Haklay, M., 2007, Comparing Map Calculus and Map Algebra in Dynamic GIS, in Drummond, J., Billen, R., Forrest, D. and João, E. (eds), *Dynamic & Mobile GIS: Investigating Change in Space and Time* . Taylor & Francis

5. Comparing Map Calculus and Map Algebra in Dynamic GIS

M.E. Haklay

Department of Geomatic Engineering,
University College London,
Gower St, London WC1E 6BT
Telephone: +44 20 7679 2745
Fax: +44 20 7380 0453
m.haklay@ucl.ac.uk http://www.casa.ucl.ac.uk/muki/

Keywords: dynamic entities; cartographic modelling; Map Algebra, dynamic modelling; function based layers; dynamic processes

Words for the index: cartographic modelling, map calculus, map algebra, dynamic modelling

5.1 Introduction

The integration of temporal information into Geographical Information Systems (GIS) has been the subject of extensive research for many years (Bergougnoux, 2000; Egenhofer and Golledge, 1998). This intense research effort stems from the inherent contradiction between GIS data models (be it raster or vector) and computer representations of dynamic processes. Due to their cartographic roots, data models in GIS have been designed to capture a static snapshot of reality (Albrecht, 2005). Thus, typical representations in GIS, where each layer is presented as a single file (such as those described in Tomlin, 1990), are geared towards describing the state of the study area at a single point in time. Over the years, representations that deal with temporal changes have been developed, especially within the context of spatial databases, where the changes can be handled at the feature level, instead of a monolithic handling of a whole layer (Worboys and Duckham, 2004).

Representations have been developed for dynamic phenomena where the challenges stem from the rate of updates and the need to visualise changes rapidly. Indeed, many solutions have been devised to deal with the dynamic aspect of GIS. In the vector model, for example, dynamic segmentation has been developed to allow the representation of changing events along static vector features. This representation is especially common in

transport applications of GIS (Longley *et al*., 2001). In the raster model, dynamic modelling capabilities have been developed and implemented within packages such as PCRaster (Van Deursen, 1995) or IDRISI (Park and Wagner, 1997).

Despite these developments, the modelling of dynamic entities and processes in a GIS is still an active research issue (Couclelis, 2001; Laurini, 2001). The reason for this continued interest, as Laurini (2001) noted, is that there are many applications in which real-time dynamic representation is required. These applications range from environmental monitoring of pollutants to the management of a vehicle fleet. The recent advances in real-time location tracking, communication, digital mapping availability, and the continued increase in computing power make these types of applications feasible, at least technically. As the technical challenges of implementing dynamic GIS diminish, researchers are now free to focus on the theoretical and conceptual challenges of the integration of temporal and dynamic aspects within GIS in novel ways.

This chapter focuses on Map Calculus (Haklay, 2004) and its potential applications in dynamic GIS. Map Calculus is an alternative to current representations in GIS, and is based on the use of function-based layers in a GIS (Haklay, 2004). A function-based layer is defined as the symbolic representation of a mathematical and spatial function. Map Calculus is best explained by comparing its core concepts to the current practice of representing surfaces in GIS in grids (rasters). The use of functional representation of layers existed in computer models in meteorology for many years (Goodman, 1985) and is being used in some global climate models, but it was not adopted in GIS and spatial analysis. The main strength of the new representation is the ability to treat analytical layers (layers that are based on manipulation of real world observations) in their symbolic form, in a similar way to the manipulation of mathematical functions in software packages such as MATLAB. This can increase the GIS analytical toolbox and open up new directions in spatial analysis research.

In this chapter, the application of Map Calculus for dynamic GIS is examined and explained through the comparison with Tomlin's (1990) Map Algebra and Cartographic Modelling. The reason for this comparison is the link between Map Calculus and Map Algebra at the conceptual level, as explained in Haklay (2004), and the long use of Map Algebra and Cartographic Modelling (Tomlin and Berry, 1979) in environmental modelling and in dynamic GIS. Of course, dynamic GIS can be implemented in vector-based or object-based representations. However, the comparison of Map Calculus to these representations is beyond the scope of this chapter.

This chapter opens with a general comparison of Map Calculus and Map Algebra and Cartographic Modelling using an interpolation function and a simplified environmental model. Through these examples, the main principles of Map Calculus are explained and clarified. The following section moves to discuss the challenges of dynamic modelling in GIS, exploring the ways in which it is implemented in Map Algebra (Tomlin 1990) and in PCRaster (Van Deursen, 1995) and outlining how such models can be implemented in a Map Calculus-based system. The chapter ends with conclusions and future directions for research.

5.2. Comparing Map Algebra and Map Calculus

5.2.1 Implementing spatial interpolation

The comparison of Map Calculus with Map Algebra provides a way to explain the main principles of Map Calculus, by allowing the reader to contrast them with the more familiar procedures of Map Algebra and Cartographic Modelling. For a detailed conceptual outline of Map Calculus see Haklay (2004). To make the comparison concrete, two common procedures in GIS are used here: the creation of an interpolated surface, using Inverse Distance Weighted (IDW) function and the implementation of a spatial model through overlay functions. Naturally, these two examples do not reveal the full range of GIS operations that are available under Cartographic Modelling and Map Algebra (Tomlin, 1990) which include local, neighbourhood and zonal operations. However, within the confines of this chapter, the two examples set the scene for the discussion of dynamic models in the next section.

IDW is a common interpolation method and it is used widely within GIS. Like all interpolation functions, IDW operates on a set of sampled points (L1,L2,…Ln) and calculates the value for a new location L' by using the following equation:

$$L' = \frac{\sum_{i=1}^{n} \frac{1}{d_i^{\,p}} L_i}{\sum_{i=1}^{n} \frac{1}{d_i^{\,p}}} \qquad (5.1)$$

where d_i is the distance from L' to the location L_i, and p is a power of the distance. Usually, the search radius is taken as a parameter of the function to limit the influence of remote data points. It is noteworthy that implementation of IDW function has been used for GIS research since its early days (for example Shepard, 1968).

In a GIS where Tomlin's Cartographic Modelling is implemented, IDW will be calculated in the following way. First, the user selects the spatial extent of the area for interpolation. Next, the user sets the spatial resolution (pixel size) of the grid that will be used to store the result of the IDW function. The next stage includes the main computational step – for each pixel, the computer takes the co-ordinates of the centre of the pixel, and uses them to calculate the IDW value for the cell. This is done by selecting data points from the search radius and including them in the calculations. The final value is stored in the pixel. Once all the values for all pixels have been calculated, the system writes the grid file and stores it on a mass storage unit – usually a hard disk – for future use. This process is represented in Figure 5.1(A).

In Map Calculus-enabled GIS, when the user requests the GIS to calculate the function, the system will register the manipulation in a symbolic form. If the point set is the layer "Height values", then the system will register a new layer as:

(5.2) `IDW("Height values", P1 ... Pn)`

where P_1 to P_n are the parameters needed for this instance of the generic IDW formula. These will include search radius, the number of points that can be included in the computation etc. The procedure for the definition of a function-based layer does not require any computation, but only the functionality to record the fact that the user made a decision to apply the function *f* to the data set *x* with parameter set *p*. Given the layer "Height values", with the field "Z" holding the actual values, search radius of 1000 units, power of 2, and a maximum of 12 neighbouring points, the internal representation of the layer can be:

```
Function -> "IDW"
Layer -> "Height Values"
Parameters -> ("Z",1000,12,2)
```

The stored definition of the layer can be used in other layers which are based on it, as explained in the next section. The computation of the function happens when the user requests the GIS to visualise the layer. When this happens, the GIS will calculate the value of the function for each pixel on the active display area of the screen. The GIS can use parameters, such as screen resolution and the current scale of the map, to minimise the amount of calculations. For example, if a user uses a common screen resolution of 1024x768, then the effective area of the map in a common GIS package (such as ArcGIS) is approximately 800x600 pixels, due to the elements of the graphical user interface which occupy the rest of the screen, such as the title bar, the status bar, and toolbars. The active area requires about 480,000 calculations – not a major load on modern central processing units (CPUs).

Within the active area, each pixel's location can be calculated by using the current scale of the map, and the area on which the user is focusing. This will provide the definitions for the co-ordinates of each pixel. As the user zooms out or in, the scale of the map changes, and new calculations for the currently displayed area are carried out. Hence, to the system's user, a Map Calculus-enabled GIS behaves in the same way as any other GIS. This process is depicted in Figure 5.1(B).

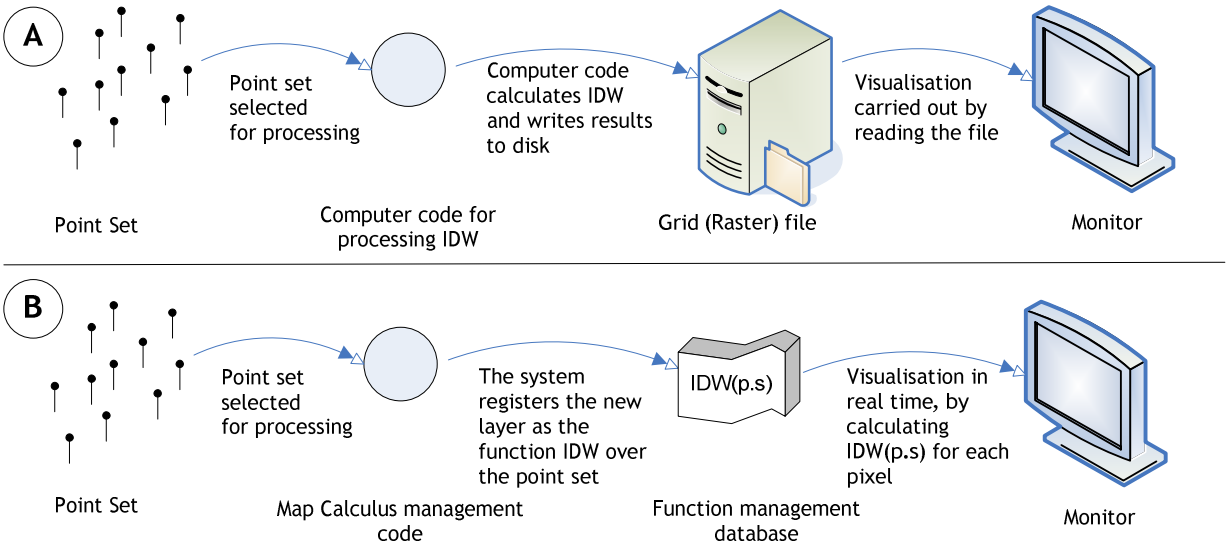

Figure 5.1 – Computation of a function in a standard GIS (A) and in a Map Calculus-enabled system (B)

Another scenario for calculation occurs when the user wants to explore the grid with other (existing) grid layers. For this, Map Calculus-enabled GIS will have a separate interface that will allow the user to define the extent of the area to which the output is required and the resolution of the output grid. This grid will be produced in the same way as in standard implementations, by calculating the value for each pixel and producing a file.

5.2.2 Site Suitability Analysis

The major difference between Map Calculus and Tomlin's representation occurs when a set of layers is manipulated in order to complete an analytical task. For example, assume the problem that the user would like to solve is to create a suitability map for the construction of a wind farm in a study area of 10 km by 10 km. There are three wind farms in a given area, as well as five farm buildings. A data set of height values for sample points in the study area was assembled through a field survey. The suitability criteria is that the location must be within 1 km of an existing farm building, but over 1.5 km from an existing wind farm and in an area with a height over 500 m above sea level. The steps of the analysis are presented in Figure 2.

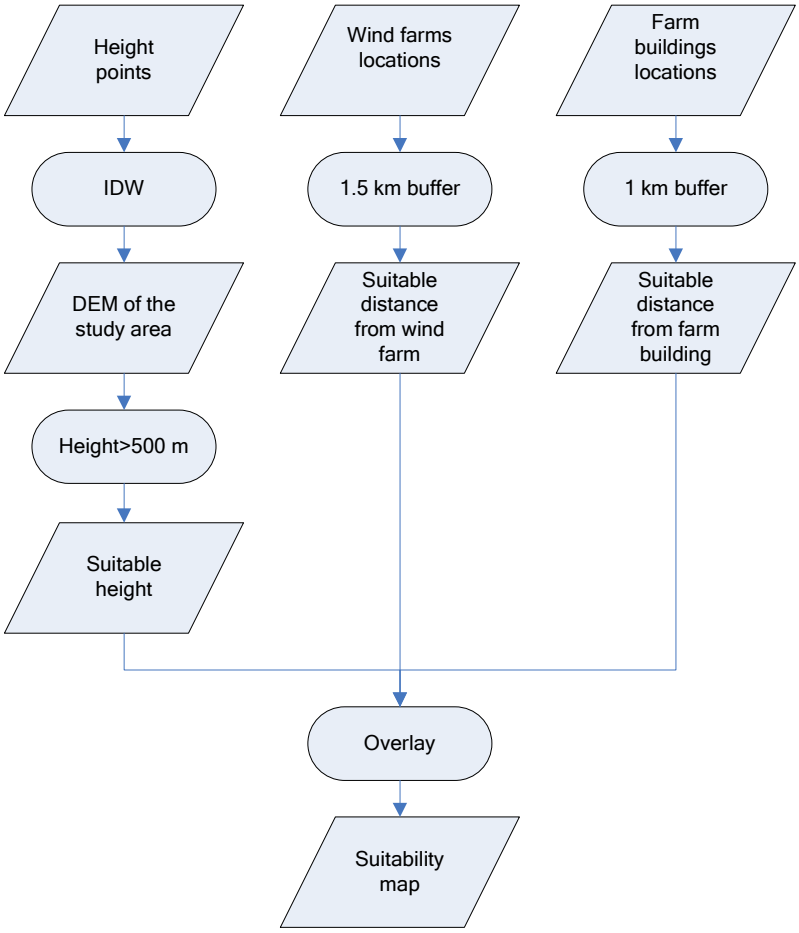

Figure 5.2 – Cartographic Modelling of Site Suitability Analysis

In a standard GIS representation it is common to use Tomlin's (1990) Map Algebra for such a task. Map Algebra provides a range of mathematical and spatial operations that can be carried out on a single layer (i.e. calculating slope) or between layers (i.e. adding cost surface to road network). This enables the user to construct sophisticated models by using these mathematical and spatial operations to merge layers. In each step of the process some operations are defined on grid layers, and, importantly, the output is a grid layer, too.

Thus, in its most generic form, the manipulation of grid layers in Map Algebra operates in the form:

$$\text{OutputRaster} = f(\text{InRaster}_1, \ldots \text{InRaster}_n, P_1, \ldots P_n) \quad (5.3)$$

where the function *f* will have 0 or more input layers (InRaster_1, … InRaster_n) and 0 or more scalar parameters (P_1, … P_n). In any function, there will be at least one input layer or parameter. The most notable aspect of this form is that each operation will result in an output layer which will be a grid. For example, the IDW function that was described in equation (5.3) can be represented in Map Algebra as:

(5.4) `IDWHeight = IDW("Height values", P1 ... Pn)`

Importantly, the difference between this form and the one presented in equation (5.2) is that in Map Calculus, the computation ends in the definition of the layer, whereas in Map Algebra, the computation ends with the production of the grid layer *IDWHeight*.

Operationally, in Map Algebra, the analysis of the site suitability problem will require the creation of five or six grid layers that will be used during the process: the IDW grid, followed by a grid containing areas above the required height, two buffer grids and the final suitability grid. In some systems, Map Algebra operations are limited to binary operation, and thus the overlay will require two steps and a temporary grid. As noted, our study area is 10 km by 10 km, and, therefore, the relevant area for the analysis can be calculated as follows: there are 5 farm buildings, and the new wind farm can be located within 1 km from an existing building. Thus, the area that is potentially suitable for the new wind farm is 15.7 km^2. Despite this, if the user selects pixel resolution of 5 m, the process includes up to 24,000,000 computations of which 83.3% are redundant. These excessive computations might lead to operational compromises such as a decision to increase pixel size, which reduces the overall accuracy of the model.

It is also notable that, while the cost of digital storage has reduced dramatically in recent years, the management of multiple grid layers is still a technical and practical problem. While the latest version of ArcGIS has no limitations on the size of a grid layer, creating a grid of 23,000x23,000 cells of random floating point numbers will lead to an output file of 2.147 gigabytes (ESRI, 2004). Thus, a model with multiple grids can lead to very significant data volumes.

In Map Calculus, the use of symbolic representation of the layers guarantees very efficient storage. In the case of the earlier analysis, the process will include only one output and this may be produced only at the end of the modelling process. The GIS will operate as follows. First, the system will record that the user requested to perform the function IDW over the "height points" layer, as well as the buffers. The internal representation may look like:

Layer ID	***Layer definition***
1	IDW("Height points","Z",100,12,2)
2	Buffer("Wind farms", 1500)
3	Buffer("Farms",1000)

As noted, each function includes reference to the input data set, and a list of parameters for this instance of the function. Hence, IDW holds four parameters as explained in the previous section. In the buffer functions, the parameter provides the distance in map units (metres in this case).

The next stage is to create the model. This can be represented in the symbolic form:

(5.5)
```
Locate(Value(IDW("Height points","Z",100,12,2))>500
     & Outside(Buffer("Wind farms", 1500)) &
     Inside(Buffer("Farms",1000)))
```

where *Locate* is a function that locates areas; *Value* provides a selection of output values from a surface function; and *Outside* and *Inside* are the definitions of the locations outside or inside the defined layer.

The final step is the visualisation of the map, possibly by creating an output grid file. In a Map Calculus-enabled GIS, the computation will include optimisation – before turning to calculate the more complex IDW function, the system will evaluate the other parts of the equation and find the area that incorporates layers 2 and 3. The resulting area will be the only part of the map for which IDW calculations will be carried out. The values will be compared to the value in the equation, and only areas above the required threshold will be marked on the final map.

This method of computation provides more precise results than it is possible to achieve with typical Map Algebra implementations, because of the compromise in pixel size that was explained in the previous section. This is especially relevant when a complex model is being computed, where it is more likely that users will choose large pixel size to allow faster computation, as the selection of small pixel size has a knock-on effect on the whole process. Another problem emerges in Map Algebra when data is computed with different pixel sizes, or when there is some shift in the origin of the grid, and, therefore, the pixels from one grid do not match the pixels of the other grid accurately. In such cases, the binary operation needs to take into account the process of combining the two grids and this introduces errors into the model. All this is eliminated in Map Calculus, as the computation is carried out in the last step in a way that takes into account the full model and the specific aspects of each function that is being used in it.

5.3. Dynamic modelling in Map Algebra and Map Calculus

5.3.1 Spatio-temporal problem solving

In their original form, Cartographic Modelling and Map Algebra did not have explicit dynamic capabilities. Temporal aspects of geographical problems were translated into static representation. For example, assume that a goods delivery company is managing a vehicle fleet. The location of all vehicles is known, as they are equipped with satellite navigation systems and radio equipment. Information about the average speed in each road in the study area is provided in real time via traffic monitoring. As part of the system, the designers want to integrate the functionality to calculate the maximum travel time of each vehicle back to the warehouse. In this case, the Cartographic Model shown in Figure 5.3 applies.

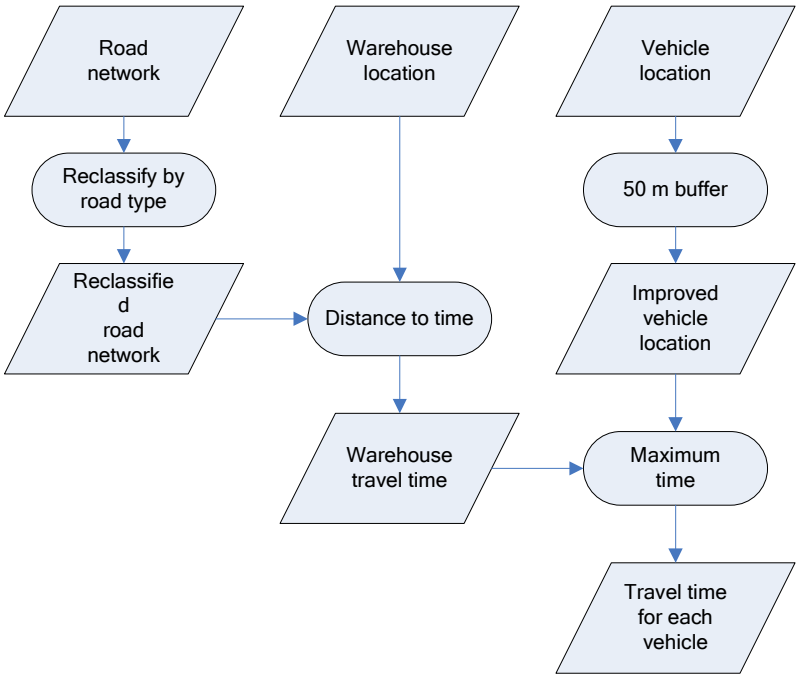

Figure 5.3 – Cartographic Modelling of Vehicle location analysis

The process in Figure 5.3 uses three input data sets – the location of the warehouse, the location of the vehicles and a map of the roads in the area. Following Tomlin's method (1990, p. 143), the first step in the computation involves reclassifying the different roads according to their travel speed, by associating a lower value with roads on which it is possible to travel fast, and vice versa. Areas that are not part of the road network are classified as "no data", to indicate that they should not be part of the calculations.

The map with the classified road network is then used to construct the travel-time map. For each location on the map, the distance from the warehouse is calculated. First, a network distance from the warehouse is calculated for each pixel, and information about the different classes of roads along the path is summarised. This information is then used to calculate the travel time by dividing the average speed for each type of road by the length of the segments belonging to this type on the path. The final value is then stored back in the pixel, representing the travel time from this point to the warehouse.

Once the travel time has been calculated, the location of the vehicles can be taken from their logs. Because it is assumed that the satellite navigation co-ordinates can give an inaccurate location of up to 30 metres and, therefore, the location can potentially be a location with "no data" value, a 50-metre buffer is created around each vehicle location to ensure that the observation is associated with a road.

The final step of the computation involves finding the pixel within the buffer with the highest time value. This procedure should be performed for each vehicle, and the output values will provide the requested answer.

The Cartographic Modelling procedure requires the creation of four grid layers within the model. It can be assumed that there is no need to compute the road network connectivity model in each run of the model, but all the rest of the information should be computed in real time – the association of road segments with average speed, the location of the vehicles, and the shortest path of the vehicles to the warehouse. This is another class of Map Algebra functions:

$$\text{Values} = f(\text{InRaster}_1, \ldots \text{InRaster}_n, P_1, \ldots P_n) \quad (5.6)$$

where, as in equation (5.3), the input is a set of grids and parameters, but the output is a set of values.

In Map Calculus, the model will be constructed from a set of building blocks, all presented as functions. The functions will include calculating a distance over a network, and calculating the nearest road segment to a given point. The layer definition will be of the form:

```
(5.7) NetworkTime(Nearest("Vehicles","Roads"),
"Warehouse","Roads","RoadTypeTable")
```

where *NetworkTime* is a function that calculates the time on the network from the vehicles to the warehouse, and *Nearest* adjusts the location of the vehicles to the nearest segment on the road network. *NetworkTime* takes four parameters: origin, destination, network and a lookup table that translates the identification code of each road segment to average speed. The lookup table can be updated continuously from real time data.

The fundamental difference between this model and the wind-farms siting case is that the current model needs to be rerun with every new input. In such situations, processing of multiple grids becomes a real hurdle in the provision of a timely response to the system's user. The main reason for the delay in response time is the redundant computations that do not contribute to the solution of the problem. This is a problem common to most Map Algebra and Cartographic Modelling implementations, where the algorithms perform the computations for all the pixels in the grid. In practice, it is unlikely that raster-based GIS will be used to solve this type of problem, and a vector-based GIS with a bespoke implementation of the *Nearest* function will be used instead of a generic buffering.

In a Map Calculus-enabled GIS the functions *Nearest* and *NetworkTime* can be optimised to ensure that they are not performing any computation that does not contribute to the final output. In this case, the optimiser can calculate first the *Nearest* function, which provides the location of the vehicle. This information will be used to calculate *NetworkTime* not for the whole network, but just for the parts of the network that contribute to the shortest path calculations for each vehicle.

The reason for the shortcoming of standard GIS implementations is that GIS is constructed as a toolbox where all operations are atomised, and, therefore, designed in a generic way that cannot take into account the specific context in which they are utilised.

In Map Calculus, the final model includes all the building blocks and the system can understand the semantics of the model. This ability will allow the creation of an optimiser that will reduce the number of calculations that are required to solve a specific problem.

5.3.2 Dynamic modelling

The example in the previous section dealt with spatio-temporal problem solving, where the GIS is required to represent information about a dynamic situation in the real world. As explained, the main challenge for the GIS is in handling temporal inputs and the need to process these inputs rapidly. Yet, the final output is static – it represents reality at a given snapshot in time when the inputs were sampled.

In contrast, in dynamic modelling within a GIS, the challenge is to represent a process in which both time and space influence the outcome. When such a process is modelled in a GIS, the ability of the GIS to integrate multiple data sources is invaluable, as it provides the framework for the model, such as the location of various physical features, the areas in which certain conditions apply etc. At the same time, the representation of dynamic phenomena within a GIS is especially challenging, due to the static nature of its data models (Albrecht, 2005).

In recent years, there has been a growth in interest in the representation of dynamic models using Cellular Automata and Agent-Based Modelling (Albrecht, 2005; Couclelis, 2001). However, most of these models are loosely linked to a GIS, and operate in a separate computing environment.

Within GIS, PCRaster (Van Deursen, 1995) provides the clearest implementation of dynamic modelling which is tightly coupled with a fully functional GIS. PCRaster has been developed as a raster-based GIS with dynamic modelling capabilities and integrates a scripting language that was deliberately designed to allow domain experts, who are not necessarily GIS experts, to create their own models (Van Deursen *et al.*, 2000). Noteworthy are PCRaster origins in physical geography with a focus on geomorphology and hydrology, although PCRaster has been used in other application areas. The system's database supports dynamic modelling by providing "…time series indexed on time and location, and by stacks of map layers representing the status of the model at different time steps" (Wesseling *et al.*, 1996).

In PCRaster, a dynamic model will be programmed by setting a script that will have the following structure (Van Deursen 1995):

```
timer BeginTime, EndTime, TimeStep;
initial InitiateModel;
dynamic ExecuteModel;
```

The timer statements inform the system about the simulation step by setting values for the beginning of the model run and the end time for the model run (thus, timer 1 20 1 will mean 20 iterations through the model). The initial statement provides instructions for

setting up the model's environment, while the dynamic part will include the instructions that will be executed at each step of the model run.

To compare the way in which a dynamic model can be executed in a Cartographic Modelling-based system and in Map Calculus, the next simple example will be used. A point-source pollution moves in a straight line, releasing a substance at a steady rate. The environment to which the pollution is released is stable and it spreads in all directions at the same rate. As a result, the pollutant will spread to a larger area, but at a reduced concentration as time passes. The model is depicted in Figure 5.4.

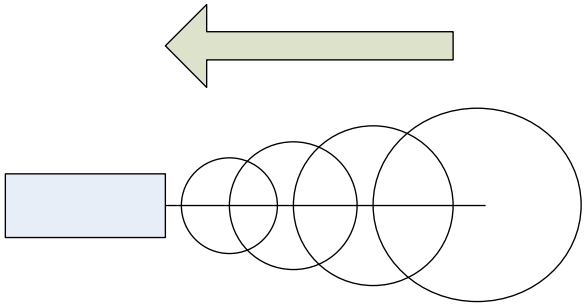

Figure 5.4 – Model of a point-source pollution: the source (rectangle) moves in the direction of the arrow, and releases substance, spreading out as a function of time.

In a system like PCRaster, the model may be implemented in the following way (described here as pseudo-code, and not in PCRaster language):

```
timer 1, 20, 1;
initial
# Location – the location of the point-source
     Location = StartLocation;

dynamic
     PollutGrid = Spread(PollutGrid);
     # SeedPollution – discharge at T0
     PollutGrid = PollutGrid + Location*SeedPollution;
     Report PollutGrid;
     Location = Location(x+dx,y+dy);
```

The core of the process is implemented through a *Spread* function. This function takes a grid, in which there are several cells with a concentration of substance, and spreads the levels of concentration to the neighbouring cells, following predefined rules. In each step of the computation, the *PollutGrid* represents the state of the substance spread at a specific point in time. The last statement advances the computation to the next location of the pollution source.

In Map Calculus, this model will be presented in the following way. The concentration of the pollutant at any given point can be represented in a differential equation, which

describes the concentration of the pollutant at a given location at time T. The definition of the function will be in the form:

```
Function -> “PointsourceSpread”
Layer -> “Start location”
Parameters -> (T0, InitialConcentration, Speed, Decay)
```

where *PointsourceSpread* is the generic function for the calculation of point-source pollution, and stores the algorithm to solve the differential equation; “*Start location*” describes the point from which the modelling will start; and the parameters describe the start time of the model (*T0*), the discharge concentration, the movement speed and the decay speed of the substance.

When the model is executed, the user is asked to enter the time over which they would like to view the dispersion model. This input, together with the stored parameters, allows the calculation of the concentration values for any location directly from the general equation. This can be done because the differential equation encapsulates the temporal dimension of the model and, therefore, can calculate the model’s output for any unit of time that passed from *T0*.

Three aspects are emerging from the comparison of the two methods. They are: the way in which the state of the computation is carried in the system, the handling of the temporal dimension and the output volume.

In Cartographic Modelling, the state of the computation is stored in the grid and not as part of the logic of the model. The state of the system is the description of the environment at each time step. The computational steps of the model deal with the procedures that advance the system from one state to the next – adding more substance as the pollution source progresses, spreading the concentration and moving the source to its next location. These, however, do not reveal anything about the state of the environmental system. Indeed, the state of the system is being carried via the computation in the data structure *PollutGrid*. This grid is acting as the memory of the system. In Map Calculus, the state of the system is explicit. The functional representation holds the information about the environmental process that is being modelled, as well as the specific aspects of the instance – the time in which the discharge started, the level of concentration etc.

As noted in the previous section, the majority of GIS implementations are based on generalised algorithms which do not take into account the specific aspects of the input or the operation. The ability of Map Calculus to take the function and the data set into account opens up the potential for more “intelligent” algorithms which “understand” the nature of the data and the manipulations that the user wishes to apply on them.

The second difference is in the handling of time. Note that in the Cartographic Modelling approach, the *Spread* function is oblivious to the specific time step – it works by spreading substance concentrations across neighbouring cells using a neighbourhood operator which is not influenced by the temporal dimension in an explicit way. This is a

good demonstration of Couclelis and Liu's (2000) insight that "…models can predict the future to the extent that they are not about the future. We can indeed predict many aspects of what is to come because events are constrained by several different kinds of determination that are in themselves outside of time" – the implementation of *Spread* is clearly a-temporal. Because the computational steps do not encapsulate the temporal element, it is the data set that stores the state of the system at a given moment, as noted previously. Importantly, the user is forced to use a discrete view of time where the progression from one step to the next occurs in discrete units, each of them presented as a full step of computation.

By contrast, in Map Calculus the temporal aspect is an integral part of the differential equation. Time is expressed explicitly as the difference between T0 and the current time. While Couclelis and Liu (2000) still holds true, as the model itself is deterministic, it is clearly spatio-temporal and linked directly to the mathematical description of the model. Here, the handling of time is as a continuous element, and the user can request the model to produce output for any given point in time.

Finally, the difference in output is noteworthy. In Cartographic Modelling, in each computational step, the user must create an output grid (the *Report* statement in the script) and store it for future reference. The reason for this is that the internal representation of the system's state (*PollutGrid*) is written over in each step, and it is essential to store a description of each step, as otherwise the GIS will need to repeat the full computation from the first step to the place that the user wants to examine (see Wesseling *et al*., 1996, for an example of such an output). In Map Calculus, the calculation is performed in real time, and there is no need to store it, as the system can always produce an output by changing the current time value. Furthermore, it is easy to see that in a Map Calculus-enabled GIS, the user can instruct the system to produce values for a specific time frame automatically and to create animations which are valuable in understanding dynamic processes (Peterson, 1993).

5.4. Conclusion and Future Developments

Map Calculus provides a different way to manipulate layers in a GIS and the comparison of its principles to Map Algebra and Cartographic Modelling demonstrates its advantages over current representations in GIS. It provides an explicit representation of the spatial functions and their manipulation, provides a compact storage of function-based layers, and enables the interrogation of function-based layers at any required resolutions.

Within the context of dynamic GIS, Map Calculus allows easier linkage to rapidly changing inputs and the implementation of a dynamic model where the model is based on differential equations. These aspects will make the GIS more accessible to domain experts, as they can focus on the construction of the model and not on the finding of a way to translate the conceptual model to the constraints of the GIS. PCRaster is already offering such a translation (Van Deursen *et al*., 2000) but is limited by the structural constraints of Cartographic Modelling and Map Algebra. Map Calculus should be seen as

the next step in making the GIS accessible to domain experts who are not familiar with GIS.

It is important to remember that Map Algebra and Cartographic Modelling have their own advantages. Of these, the simple and accessible data mode, which is based on the tessellation of the study area, holds a primary position. Indeed, the grid model is now used widely for dynamic models, especially in Cellular Automata models (Couclelis, 2001). Another advantage of these techniques is that the algorithms are simple to implement: the universal grid structure simplifies the processing and provides a clear conceptual framework. Even with the redundant computations, modern implementations provide reasonable response time and they are used for many applications of GIS.

At the same time, Map Calculus poses certain challenges. These include the reformulation of common GIS operators, creation of a function optimiser and consideration of optimal visualisation methods. Once these issues have been solved, Map Calculus can open up new avenues for spatial analysis and applications of Geographical Information Science.

Most importantly, Map Calculus is implemented only as a basic prototype. As such, its ability to handle and manipulate geographical data sets is very limited. A full evaluation of Map Calculus potential in geographical problem solving will be possible only through a full implementation of a system that will support it.

5.5. References

Albrecht, J. (forthcoming) 'Dynamic GIS', In Wilson, J. and S. Fotheringham (ed.) *Handbook of Geographic Information Science*, Blackwell publishers.

Bergougnoux, P. (2000) 'Editorial: A Perspective on Dynamic and Multi-Dimensional GIS in the 21st Century', *GeoInformatica* vol. 4, no. 4, pp. 343-348.

Couclelis, H. and Liu, X. (2000) 'The Geography of Time and Ignorance: Dynamics and Uncertainty in Integrated Urban-Environmental Process Models'. In Proceedings, GIS/EM4 Conference: www.Colorado.edu/research/cires/Banff/upload/136. Last accessed: 20th June 2005.

Couclelis, H. (2001) 'Model Frameworks, Paradigms, and Approaches', In Clarke, K.C. Parks, B.E. and Crane M.P. (ed.) *Geographic Information Systems and Environmental Modeling*, New York: Prentice Hall, pp. 34-48.

Egenhofer, M. J. and Golledge, R. G., (1998) *Spatial and Temporal Reasoning in Geographic Information Systems*. New York: Oxford University Press.

ESRI (2004) *What is the maximum size a grid can be?* http://support.esri.com/search/kbdocument.asp?dbid=14575 last accessed 25th November 2004.

Goodman, A. (1985) *Surface Analysis: a Structured Bibliography*, Working paper 17, Department of Geography, Monash University

http://www.deakin.edu.au/~agoodman/publications/biblio/surfacebiblio.html last accessed 1st August 2002.

Haklay, M. (2004) 'Map Calculus in GIS: a Proposal and Demonstration', *International Journal of Geographical Information Science (IJGIS)*, vol. 18**,** no. 1, pp. 107-125.

Laurini, R. (2001) 'Real Time Spatio-Temporal Database', *Transactions in GIS*, vol. 5, no. 2, pp. 87-97.

Longley, P., Goodchild, M., Maguire, D. and Rhind D. (2001) *Geographical Information Systems and Science*. New York: Wiley.

Park, S. and Wagner, D. F. (1997) 'Incorporating Cellular Automata Simulators as Analytical Engines in GIS' *Transactions in GIS*, vol. 2, no. 3, pp. 213-231.

Peterson, M.P. (1993) 'Interactive Cartographic Animation' *Cartography and Geographic Information Systems*, vol. 20, no. 1, pp. 40-44.

Shepard, D. (1968) 'A Two-Dimensional Interpolation Function for Irregularly Spaced Data' *Proceedings 23rd ACM National Conference*, pp. 517-524.

Tomlin, C.D. and Berry, J.K. (1979) 'A Mathematical Structure for Cartographic Modeling in Environmental Analysis' *Proceedings of the Annual Meeting of the American Congress on Surveying and Mapping and the American Society of Photogrammetry*, Falls Church, VA. pp. 269-283

Tomlin, D. (1990) *Geographic Information Systems and Cartographic Modelling*. Englewood Cliffs: Prentice Hall.

Van Deursen, W.P.A. (1995) *Geographical Information Systems and Dynamic Models: Development and Application of a Prototype Spatial Modelling Language*. Doctoral dissertation, Utrecht University, NGS 190.

Van Deursen, W.P.A., Wesseling, C.G. and Karssenberg, D. (2000) *How do we Gain Control over GIS technology?* Proceedings of the 4th International Conference on Integrating GIS and Environmental Modeling, 2–8 September, Banff, Canada. http://www.colorado.edu/research/cires/banff last accessed: 20th June 2005.

Wesseling, C. G., Van Deursen, W. P. A. and Burrough, P. A. (1996) *A Spatial Modelling Language that Unifies Dynamic Environmental Models and GIS,* in Proceedings, Third International Conference/Workshop on Integrating GIS and Environmental Modeling, Santa Fe, NM, 21-26 January 1996. Santa Barbara, CA: National Center for Geographic Information and Analysis (CD-ROM).

Worboys, M. and Duckham, M., 2004. *GIS – A Computing Perspective*, 2nd Ed., Boca Raton: CRC Press.

Biography

Dr Muki Haklay is a lecturer in Geographical Information Science in the Department of Geomatic Engineering at UCL. Dr Haklay holds a Ph.D. in Geography from UCL, an M.A. in Geography and a B.Sc. in Computer Science and Geography from the Hebrew University of Jerusalem. His research interests include public access to environmental

information, Public Participation GIS, Human-Computer Interaction in GIScience, and Spatial database and data models.