

Map Production From An Active Object Database, Using Dynamic Representation and Automated Generalisation

P.G. Hardy

Laser-Scan Ltd, Science Park, Milton Road, Cambridge, CB4 4FY, UK.
email paul@lsl.co.uk

ABSTRACT

Map production has been seen previously as a standalone task, compiling information together from various sources to produce a particular cartographic product. This traditional view is now challenged by the availability of new environments which bring together databasing, object data modelling, image analysis, active agent behaviour and automated cartography to produce a unified flowline for production of multiple maps, charts, and geospatial data. This paper overviews the capabilities of a modern map production application built around an object-oriented geospatial database, and specifically highlights its capabilities for active representation, multiple geometry and automated generalisation, in order to explore the benefits and future directions of active object mapping.

1. INTRODUCTION

1.1 Why an object database for mapping?

A map is a model of part of the surface of the Earth presented conventionally as a graphical illustration. Maps are produced for different purposes and will tend to exaggerate relevant features while minimising or suppressing irrelevant detail. The term 'map' is used in this paper to cover the range of mapping products such as topographic maps, thematic maps, charts, plans, atlases and geodata (e.g. CD-ROMs). Producing such mapping products used to be a manual draughting task, but increasingly relies on computer cartography.

Understandably, the first stages of the evolution of digital mapping mimicked the conventional process. Map production was commonly done by capturing and compiling the data needed to produce a particular map or chart, usually using file-based feature mapping or graphics software. Increasingly though, the wasteful nature of such one-off capture has been recognised, and there is a move to a database-centric approach in which a geospatial model of the world is captured, stored, and updated [Cameron & Hardy 1998]. Starting from the database, it is then possible to produce a range of products at differing scales and to different specifications [Woodsford 1995a].

Traditional relational databases are not designed for holding the complex data models and large volumes of data involved in building and ensuring the ongoing integrity of a real-world geographic mapping database. Neither is it easy to produce a range of cartographic products from such data using the static

representation facilities found in traditional GIS and mapping software.

Recently however, Object-Oriented (O-O) geospatial databases and associated mapping products have appeared which provide the technology to step into a new world of active objects and product-independent geodata storage. The later sections of this paper cover the O-O paradigm in more detail and put forward its strengths for geographic databasing and map production. They use as an exemplar, the Gothic O-O database and LAMPS2 mapping system from Laser-Scan [Laser-Scan 1994].

2. OBJECT DATA MODELLING

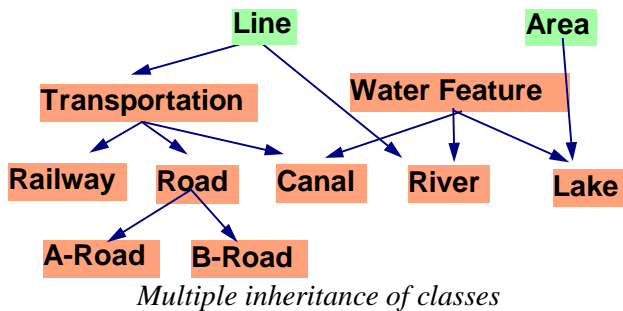
2.1 Object-Orientation and object data model

In an O-O database, real world entities are abstracted and held as objects. All objects belong to object classes. For each class there may be many objects, but each object belongs to only one class. The class defines what values can be held by an object. Values can be simple datatypes (integers, strings, dates, etc.) together with more specialist types (geometries, locations, rasters, tables). Furthermore, objects can hold structural information or references between objects.

A key, and defining, concept of O-O is that of methods defined on objects. These methods are bound to behaviours. When a method on an object is invoked by sending a message to the object, the behaviour bound to it is executed, possibly using values and references also held by the object. The ability to define behaviours as part of the database

schema, rather than as part of the application, is a fundamental concept of the O-O paradigm.

A further key concept of O-O is that of inheritance, which provides the means to define a new object class in terms of existing classes. The new class inherits the characteristics (values, references, behaviour methods) of its parent class or classes, unless superseded or redefined. Using inheritance, hierarchies of classes can be created and maintained in a systematic manner.



True O-O has gained much popularity in software engineering and computer graphics [Taylor 1990], and is gaining ground in GIS. O-O is now appearing in cartography and geodata production. In reality, however, there are still few commercially available systems that support all the key elements to a level that can successfully support mapping, charting and geodata production applications.

2.2 Methods and behaviours

Methods are central to the O-O technology. Each object class will have inherited basic methods from its parent classes, and can have other methods and specific behaviours for standard methods defined on itself. Methods are of several types:

- Value methods (also known as derived attributes) return an answer to a message. The results appear as attributes on enquiry, e.g. area, length, description
- Reflex methods occur automatically at milestones in an object's lifecycle: creation, modification or deletion (before and after). They are used to set up consequences of actions.
- Validation reflex methods enforce integrity, and allow you to put your own rules on each object class (see 2.3 below).
- Change to a referenced object is a reflex method which can trigger the propagation of effects from one object to another.
- Display methods give active representation (see 2.4 and 3 below)
- Process methods happen at operator request. They are used to carry out data cleaning, data

checking, polygon formation, and generalisation on defined sets of objects.

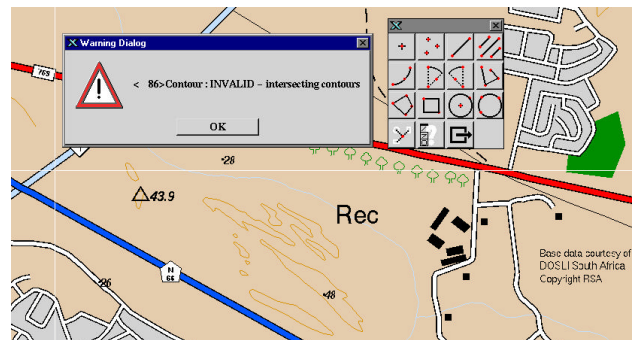
2.3 Validation methods for data integrity

Data integrity is a major issue for agencies who invest large amounts of money in capturing and maintaining a large geospatial database. The object data model allows the agency to define its geodata logic and business rules as reflex methods in the database schema. This means that the database will enforce these rules as the objects are entered into the dataset, and whenever they are modified.

Whenever an object is being modified, messages are sent automatically to the object at the various stages:

- a message before the modification is started, so that it can check that it is allowable (e.g. can't move a lighthouse unless you are a supervisor).
- a message after the modification is finished, so that it can check that it has been done validly (e.g. can't edit a contour so that it crosses another contour).

If any of the validation reflex methods return a "not OK" result, then the complete transaction is rolled back as if it hadn't started. Note that such validation methods in the database are not just applied during interactive edit, but also during other operations such as bulk data loading from external data sources (e.g. legacy data).



Validation method checks on contour crossing

2.4 Display methods and active representation

In an O-O mapping system, the appearance of an object on the screen or on hardcopy is generated at draw time by execution of an arbitrary 'display method' defined on the object class and stored in the database under the direct control of the customer. See Section 3 for more detail on active representation.

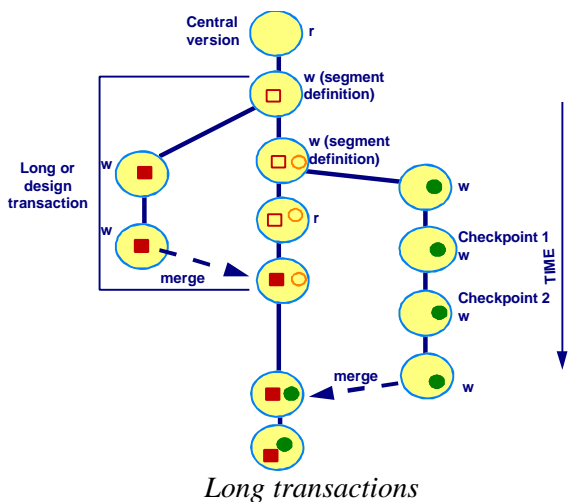
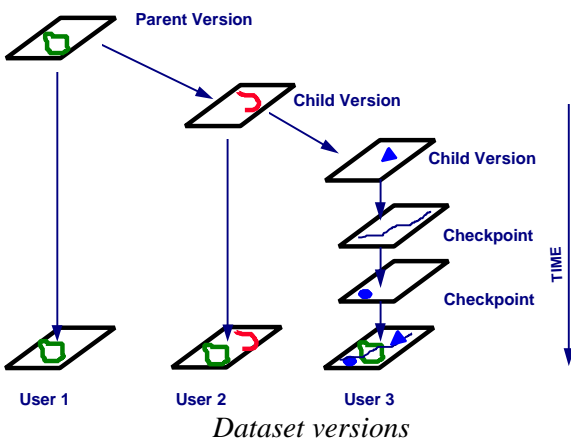
The advent of the object-oriented paradigm opens up new strategies for generalisation, in which behaviours allow the object to decide for itself what

to do, e.g. it can inspect its relationships with its neighbours to decide whether to move itself. See Section 6 for more detail on active generalisation.

2.5 Object Versioning and Long Transactions

One of the problems of traditional relational databases is that their transaction model is designed round the rapid lock-update-unlock scenario common in financial and business transactions. However, completing the update of geodata and mapping for an area is often a long drawn-out process, taking several hours, days, or even weeks [Hardy 1995]. In the interim, the half-updated state must not be allowed to be used for production tasks, but also the unchanged data must not remain locked

The object database, with its encapsulation of all the data and behaviour for each object, lends itself to a different transaction model. In this, each user has a stable view of a 'version' of the dataset. Only changes made by that user are stored in the version, the unchanged objects are accessed from the previous version. Versioning of datasets solves the problem of long transactions and allows the sharing of very large data volumes between multiple users needing write access [Woodsford and Hardy 1997].

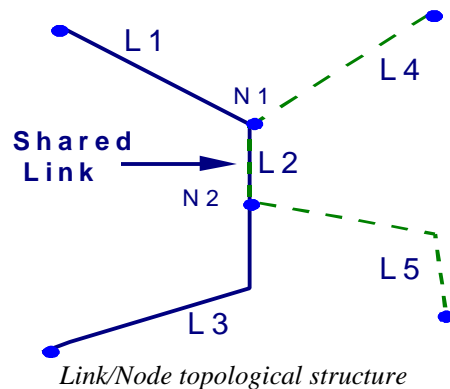


In the diagram above, two users each reserve a segment of a continuous dataset for update. After several checkpoints (e.g. stops for lunch), the changes are merged to give an updated mainstream version. This is discussed further in previous papers on spatial databasing [Woodsford 1996].

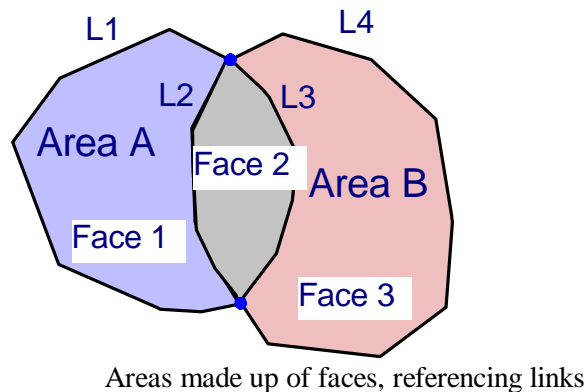
Within mapping, charting and geodata production flowlines, versioning allows efficient multi-user update access to a continuous dataset of master features. In addition, it allows product-specific versions to be subset from the continuum, e.g. to include all the sheet furniture (grids, borders, titles, legends, North arrow, etc.), without having to copy all the unchanged geodata.

2.6 Topology and Structure

The Gothic object spatial database uses methods and object references to implement in-built support for topology structure. The user can choose Spaghetti or Structured for each class, and then define snapping tolerances between pairs of classes. The database will then apply these rules and create the necessary links and nodes as objects are digitised or imported.



In addition to line topology which is made up of links, areas can be topologically structured, referencing links as their boundaries and one or more faces as their interior.



Areas made up of faces, referencing links

Uniquely, the topology is maintained automatically as the data is edited, avoiding risks of overshoots, undershoots, slivers etc., and obviating the need for subsequent error-prone building of coverages. Polygons can be formed out of existing linework, either directly or as a set of faces (the atomic entities of area).

3. ACTIVE OBJECT REPRESENTATION

3.1 What is Active Representation ?

In an O-O mapping system, the appearance of an object on the screen or on hardcopy is generated at draw time by execution of an arbitrary 'display method' defined on the object class and stored in the database under the direct control of the customer. This contrasts with the traditional approach as indicated in the following table.

O-O Active Representation	Feature Representation
Dynamic - objects can draw themselves differently each time	Static - defined by feature class
Defined in the database	Defined in application
Can be defined and enhanced by the customer	Can only be enhanced by the software supplier
Can be influenced by combinations of attributes	Indexed by single feature code attribute
Can be influenced by attributes derived from other referenced objects	Each feature is represented in isolation
Can adapt to external influences (e.g. required map scale)	Not adaptive

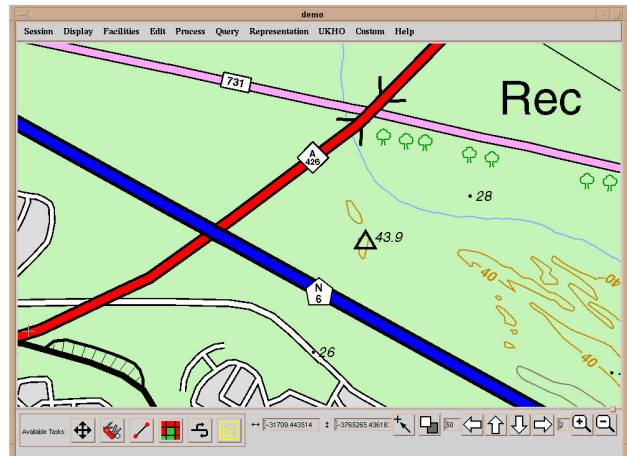
3.2 Results of Active Representation

The following example illustrates the benefits of active object representation, in the production of a topographic 1:50,000 map from a product-independent database. In particular, the following features should be noted:

1. The road number symbols are objects of a single class which have no attributes and no stored coordinates. When digitised, a pre-commit reflex method defined in the database locates the nearest road object and sets up a direct object reference. Whenever a message is sent to the symbol object,

it follows the reference to determine the road number, road type, position, and orientation. It then uses this information to draw itself with the appropriate text and box.

2. The contour label objects similarly extract height, position, orientation from the contour object, and use the information gained to blank out a section of the contour and draw the label, oriented so that it reads correctly going up hill.
3. The bridge object is a short line object which has its ends snapped onto the road (A425), and therefore shares link geometry for the section. It has no line representation of its own, but repeats the representation of the road to which it is attached, at a higher drawing priority to ensure that it overdraws the crossing road (731). The display method calculates the position and orientation of the bridge abutments (> <) and draws those also.
4. Roads are drawn as two parts with differing priorities, firstly in black to get the casings, then more narrowly in the infill colour. The priorities can come from combinations of attributes on the road class.
5. The railway cutting (bottom left), complete with its embankment ticks is generated as needed for scale from the outline of the cutting using a representation generation method.
6. The trig point uses multiple representations derived from combinations of attributes to display both the triangular symbol and the text label for the height.



Base map data courtesy of DOSLI South Africa. Copyright RSA.

The functionality and benefits of active object representation are applicable to a wide range of cartographic products, from topographic mapping to nautical charting [Hardy & Woodsford 1997]. Active representation is not the only useful object-oriented facility to apply when producing multiple mapping products from a product-independent database. Data view selection, map generalisation,

furniture generalisation, and product-specific alternatives are all tools in the armoury of the O-O multi-product map producer [Hardy 1998].

4. OBJECT DATABASE VIEWS

4.1 Why do we need object database views ?

The new database-centric approach to map production means that a single continuous dataset which models the real world is created and maintained. The contents of this dataset (sometimes called the Master Features Dataset or MFD) have to be the source for all the products to be generated, which may range from topographic mapping, through navigation atlases to thematic maps.

Any one of these products will depict a subset of the MFD - topographic maps may not show postcode boundaries; navigation atlases may not show contours, and thematic maps may not show roads.

In practice, the selection criteria are much wider ranging than just by feature class - Topographic maps may need to show contours only if their heights are a multiple of 100m, or may show urban areas if their population is greater than a minimum value. Atlases may show villages only if they lie on a through route. A thematic map may only show points of interest if they lie within 2km of a trunk road. The efficient and flexible mechanism for implementing all these types of selection is the object database view.

4.2 What are object database views ?

A database view is a way of hiding some of the information in a database, and presenting a simplified subset to the user. Relational database views are traditionally read-only, and can only subset by hiding specific columns of tables, or hiding rows by value range of specific fields.

In contrast, object database views are read/write, and the selection depends on the result of any true/false value method defined on the object class. Hence, whether a specific object is in the view or not is determined when the method behaviour is invoked at draw time (or whenever). View methods are defined in the database schema (or can be automatically created based on query forms).

The view method can check multiple attribute values, can follow pointer references to related objects to retrieve information, and can use the power of the spatial index and spatial toolkit to calculate whether this object should appear in this view or not.

The richness of the object database views capability allows the embodiment of a cartographic or geodata product specification as a saved specification, which can then be used to produce successive versions of products.

5. MULTIPLE GEOMETRY OBJECTS

5.1 Why multiple geometries ?

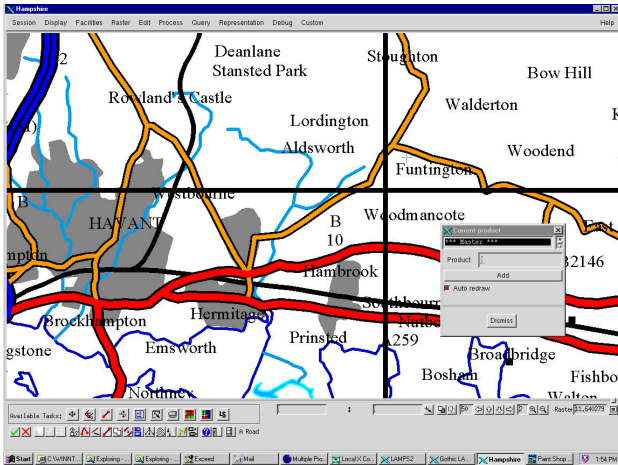
The object database holds a model of the real-world state of the geographic features it includes. However in any particular map product, it may be necessary for cartographic reasons to show the feature in a different position, or in a modified shape.

One way to handle this might be to have separate product-specific datasets, one for each product, containing the modified features. However this would be difficult to maintain and would necessitate much labour to keep the master and derivatives in step in the light of change in the real world.

5.2 What are multiple geometry objects ?

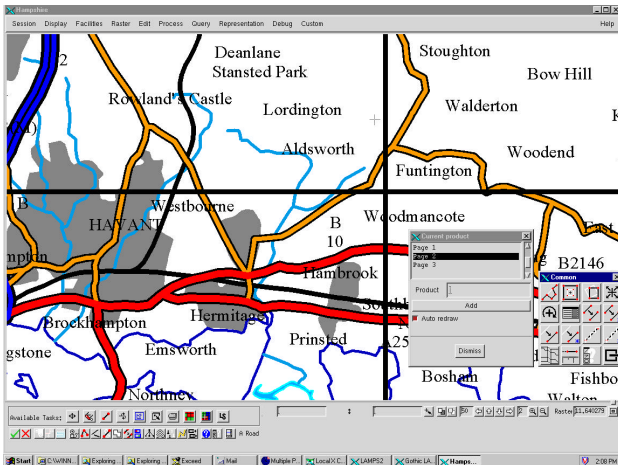
A better solution is made possible by encapsulation, which is one of the basic tenets of object-orientation. In the object database system, access to all properties of the object can only be via methods defined on the object class. This is true not just for simple values such as attributes, but also for more complex values such as the point, line, or area geometries that contain the defining coordinates for the features.

Hence, by overriding the two methods 'set-geometry' and 'get-geometry', it has proved easy to implement in Gothic an efficient mechanism for storing multiple alternative geometries on the object, only one of which is active at a particular time. This mechanism is transparent to the application and allows normal editing commands to be used for setting the alternative geometries, and normal display and hardcopy facilities to be used for output.



*Display using Master geometry
(Base data courtesy of AA)*

The above illustration shows the master geometry for all objects, while the following illustration shows the appearance when a product alternative has been selected. The purpose of the alternative is to produce an atlas page covering the top right quadrant of the screen. Note the movement of some features (e.g. “Stoughton”), and the change in shape of others (e.g. the road near “Funtington”). All these changes are to alternative object geometry, so attributes (such as the village name “Stoughton” are common and not stored twice.



Display using Alternative geometry

The facility for multiple alternative geometries on objects in LAMPS2 allows a single dataset to hold not just the master geometry (the real-world position), but alternatives suitable for a range of derived products. At the same time, only a single object exists with a single set of attributes, so space is conserved and the complexities of parallel update are not needed to control multiple objects.

6. GENERALISATION

6.1 Map Generalisation

Map generalisation is the science (and art) of exaggerating those aspects that are important for this particular map purpose and scale, and removing irrelevant detail that would clutter the map and confuse the user.

Generalisation has traditionally been a hard task to automate, being dependent on the skills of the human cartographer. People have tried for years to build centralised ‘knowledge bases’ of generalisation rules, with very limited success. In such systems, the map features themselves have just been passive items containing coordinates and attributes, acted upon by the centralised rules.

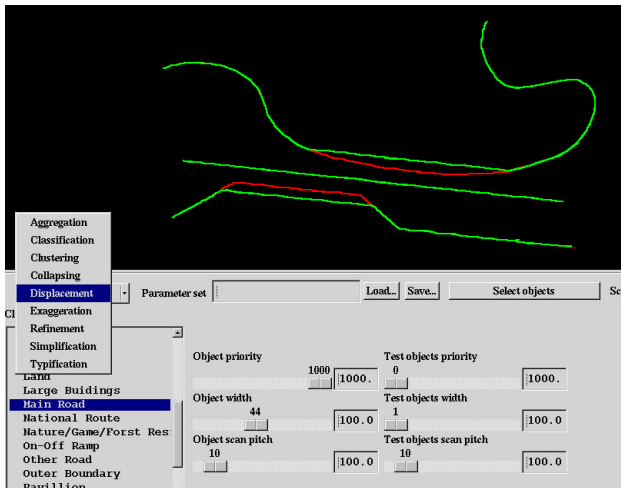
In the object-oriented world, this is turned upside down. The map features themselves become objects which have generalisation behaviours defined in the database schema. The application itself becomes much thinner, and contains no knowledge about what, how, or when. It merely provides a framework for invoking and sequencing the generalisation processes by sending messages to selected objects.

Each such object inherits behaviours from its object class definition and from superclasses in its class inheritance hierarchy. These behaviours allow the object to decide for itself what to do when receiving a message to generalise itself, e.g. it can inspect its relationships with its neighbours to decide whether to move itself. However, as the object modifies itself, any objects which are directly linked to it or spatially adjacent can also be told to reassess themselves, so that effects propagate.

The advent of the object-oriented paradigm therefore opens up new strategies for generalisation. LAMPS2 includes an object-oriented generalisation facility which allows the user to define the strategy for generalisation in terms of methods on the object classes [Hardy 1996].

Generalisation base classes are provided which supply generalisation process methods for multi-object combinational operations (aggregation, typification, displacement) and others for single object generalisation (collapsing, refinement, exaggeration and simplification). Note that these are implemented as behaviours of the objects in the database, not as commands within a program.

Preparation for generalisation using these methods is aided by a visual interface to setting controlling parameters which allows the map designer to see in real-time the effects of tuning parameters. Once set up, then a process sequencing mechanism allows unattended execution of complex generalisation runs on multiple object classes for chosen areas of the continuum to produce generalised products.



Visual interface to generalisation parameters

6.2 Multi-Agent Generalisation

Further dramatic developments of the LAMPS2 generalisation facilities are under way, driven by the AGENT project on multi-agent generalisation. This project [AGENT 1997] is a collaboration under the ESPRIT programme (LTR/24939) involving Laser-Scan as providers of object technology together with a national mapping agency (IGN) as prime contractor, and academic partners, some of whom provide in-depth knowledge of generalisation algorithms, while others provide insight into multi-agent modelling.

In this context, agents are self-aware active software objects which co-operate, subject to a set of constraints, to achieve a goal. For map generalisation, it is the geographic objects such as houses and roads which become active agents and co-operate through simplification, typification and displacement of themselves to achieve a cartographically acceptable generalised result [Baejis et al 1996].

One outcome of the AGENT project is to be a set of base classes in LAMPS2, the methods of which embody the measures, constraints, algorithms and goals of map generalisation for specific feature classes. Future papers will expand on this new approach to automated generalisation.

6.3 Mapping on the Internet

The O-O paradigm is increasingly fundamental to the evolution of the Internet, as witness the dominant role of JAVA and CORBA. Internet access to rich object models of geodata have already been shown to provide flexible and responsive maps in Internet web browsers, and rapid progress in this area is underway. See examples using Gothic active representation at the on-line Ad Hoc guide to Cambridge available on <http://www.adhoc-guides.com>.

To allow true on-demand mapping in response to user requests over the Internet, then the decisions which traditionally have been made by the human cartographer will in future need to be made by the automated geodata web server. This automation has proven insurmountable for traditional feature-based digital mapping systems.

However, the active object generalisation paradigm has shown the way forward and Laser-Scan and the AGENT project are working towards the goal of on-demand Internet mapping through O-O and multi-agent generalisation.

7. BENEFITS

7.1 Strengths of Object-Oriented for Mapping

The description of O-O given in earlier sections applies to almost any geospatial data application [Woodsford 1995b]. The particular strengths of O-O in respect to mapping and cartography relate to:

- Data storage and retrieval of a model of the real world in an object database, including object versioning, long transactions, complex data modelling, validity checks and data integrity.
- Data visualisation and cartographic product generation, including active representation, multi-geometry alternatives, and automated generalisation.

7.2 Benefits of O-O Mapping

The benefits that arise from the above strengths include:

- The object data model allows accurate modelling of the real world, including behaviours.
- The dataset versioning and long transactions allow efficient multi-user access to a true continuous map dataset.
- The validation methods of the O-O data model can prevent invalid data being captured, allowing immediate rectification of operator error.

- Active representation allows efficient generation of a range of cartographic quality products from a common database.
- Multi-product alternative geometries stored on single objects allow sheet-specific modifications to be stored in the master dataset, obviating the need for tracking of similar changes through multiple product datasets, and hence reducing the costs of product update.
- O-O generalisation methods provide an automated solution to what is historically a labour-intensive and expensive task. This new ability for dynamic generalisation is timely, given the increasing requirements for on-demand mapping, such as for presenting responsive maps in an Internet web browser.

8. CONCLUSION

Object-oriented mapping and charting software as typified by Laser-Scan's Gothic LAMPS2 provides a set of versatile capabilities which enable a range of visual and data products to be generated from a common spatial database.

Recent advances include object database views and multiple geometry objects. These supplement the existing database versioning, active representation and automated generalisation capabilities to give a complete and cost-effective flowline for multi-product generation.

REFERENCES

Lamy, S., 1997, "Project AGENT", on-line paper at <http://agent.ign.fr>.

Baeijs, C. Demazeau, Y. and Alvares, L., 1996, "Application of Multi Agent Systems to Cartographic Generalisation", 7th European workshop on Modelling Autonomous Agents in a Multi Agent World (MAA-MAW).

Cameron, E.C.M. and Hardy, P.G., 1998, "Stereo Images with Active Objects - integrating photogrammetry with an object database for map production", International Archives of Photogrammetry and Remote Sensing. Vol. XXXII, part 2, Commission II, Cambridge, pp 35-40.

Hardy, P.G. and Wright, P., 1995, "Techniques for Update in Raster and Vector Cartography". ICA/ACI Conference Proceedings, September 1995, Barcelona, Spain.

Hardy P.G., 1996, "Map Generalisation - The Laser-Scan Way", on-line paper at

<http://www.Laser-Scan.com/papers/lamps2mapgen.htm>

Hardy P.G. and Woodsford, P. A., 1997, "Mapping with Live Features - Object-Oriented Representation", ICC Conference Proceedings, June 1997, Stockholm, Sweden.

Hardy P.G., 1998, "LAMPS2 Multi-Product Generation", on-line paper at <http://www.Laser-Scan.com/papers/L2prod.htm>

Laser-Scan, 1994, "The Gothic Versioned Object-Oriented Database: an Introduction". November 1994, Laser-Scan Ltd, Cambridge UK. See also up-to-date information on the Internet web pages at <http://www.laser-scan.com/>.

Taylor, David A., 1990, "Object-Oriented Technology: A Manager's Guide", Addison-Wesley Publishing Company.

Woodsford, P. A., 1995a, "Object Orientation, Cartographic Generalisation and Multi-Product Databases", ICA/ACI Conference Proceedings, September 1995, Barcelona, Spain.

Woodsford, P. A., 1995b, "The Significance of Object-Orientation for GIS" IUSM Working Group on GIS/LIS, September 25-28, Hannover, Germany

Woodsford, P. A., 1996, "Spatial Database Update - the Key to Effective Automation" International Archives of Photogrammetry and Remote Sensing. Vol. XXXI, (B4), Vienna, pp 955-61.

Woodsford, P. A. and Hardy P.G., 1997, "Databases for Cartography and Navigation", ICC Conference Proceedings, June 1997, Stockholm, Sweden.

[Original 1998/08/4, Revision 1.2 1998/10/15]