Active Object Techniques for Production of Multiple Map and Geodata Products from a Spatial Database

MM
P.G. Hardy
Laser-Scan Ltd,
Science Park, Milton Road, Cambridge, CB4 4FY, UK.
Phone +44 1223 420414, Fax +44 1223 420044
Email: Paul @ lsl.co.uk

Historically, map production has been an isolated task, compiling information 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, geospatial data, and on-demand spatial visualisations such as Internet web mapping.

This paper overviews a modern production application built on an object-oriented geospatial database, and specifically highlights its capabilities for active representation, multiple geometry and active object selection, in order to explore the benefits and future directions of active object mapping. Recent advances in handling spatial objects allow for the first time the efficient storing and retrieval of a geographical feature as an object with a single set of attributes, but with multiple alternative sets of spatial information (geometries). This allows one continuous spatial dataset to hold data appropriate to multiple map and geodata products, reducing the overheads of update as the outside world changes.

Complementing the multiple geometry capability are new mechanisms for selection of objects from the database for product generation. Object database views are implemented as object behaviours, and are the means by which geographical feature objects can decide for themselves whether they should be included in a specific map or geodata product. Once selected for inclusion, further object behaviour techniques come into play. Active generalisation methods rely on message passing to objects to ask them to simplify or displace themselves. Dynamic representation implemented as active object display behaviours allows individual objects to draw themselves differently according to the surroundings.

Distributing the knowledge of selection, generalisation and representation into object behaviours in this way overcomes many of the problems previously encountered in embedding the skill of the human cartographer into a software solution. For the first time the cartographer's dream of the scale-free map can become reality, not only for generation of a range of paper-based mapping products, but also for on-demand visualisation in new media.

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

Understandably, the first stages of the evolution of digital mapping mimicked the conventional production process, 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, one can produce a range of products at differing scales and to different specifications, as described later in this paper.

Traditional relational databases are not designed for holding the complex data models and large volumes of variable length 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. Now, Object-Oriented (O-O) geospatial databases and associated mapping products have appeared [Warboys et al 1990], which provide the technology for a new world of active objects and product-independent geodata storage. The later sections of this paper cover the O-O paradigm, 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], shown schematically in the product family diagram below right.

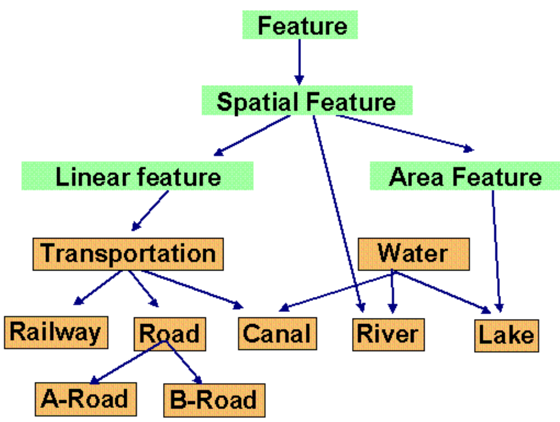
2. OBJECT DATA MODELLING

2.1 Object-Orientation and object data model

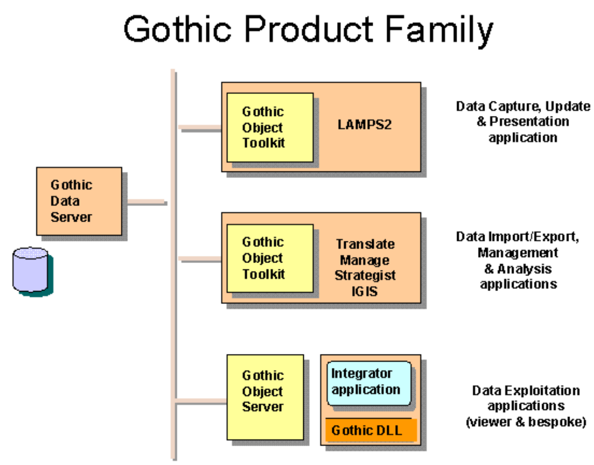
In an O-O database, real world entities are abstracted and held as objects. All objects belong to object classes. For each class there may be many objects, but each object belongs to only one class. The class defines what values can be held by an object. Values can be simple datatypes (integers, strings,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



Multiple inheritance of classes



O-O Product Family

True O-O has gained much popularity in software engineering and computer graphics [Taylor 1990], and is appearing in GIS, 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 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 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 again whenever they are modified.

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

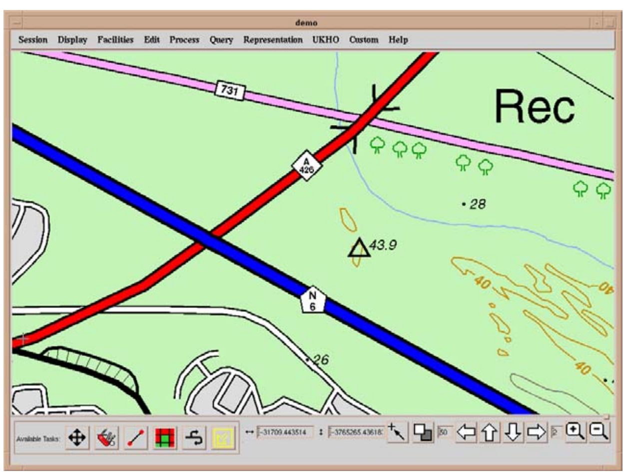
- One before the modification is started, so that it can check that it is allowable (e.g. can't move a lighthouse unless you are a supervisor).
- One after the modification is finished, so that it can check that it has been done validly (e.g. can't edit a contour so that it crosses another contour).

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).

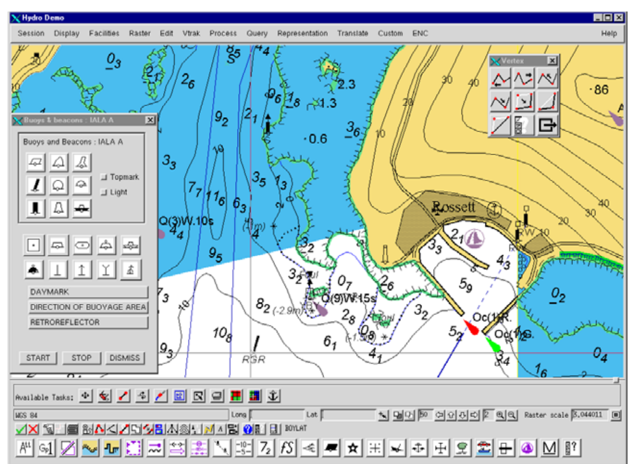
2.4 Display methods and active representation

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'. Such methods are 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.

Dynamic O-O Active Representation	Traditional static feature-based representation
• Objects can draw themselves differently each time, adapting to external influences (e.g. map scale, product) • Behaviour defined in database by customer • Can be influenced by combinations of attributes including attributes derived from other referenced objects	• Single unchanging appearance, requiring separate datasets and representations to achieve multiple products • Behaviour defined in application by supplier • Influenced only by single feature code attribute, and each feature is represented in isolation



Topographic map active representation
Base map data courtesy of DOSLI South Africa. Copyright RSA.



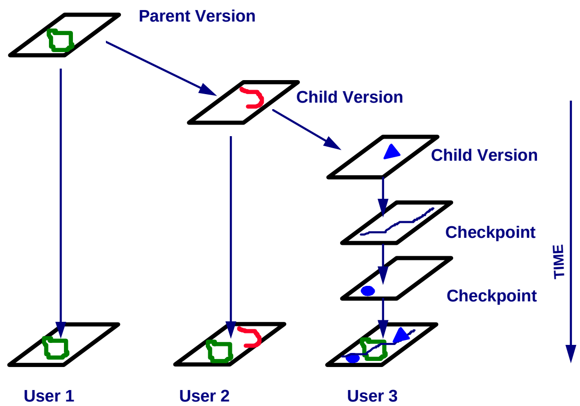
Nautical chart active representation
Base data and representation courtesy UKHO, Crown copyright

The functionality and benefits of active object representation [Hardy & Woodsford 1997] are applicable to a wide range of cartographic products, from topographic mapping to nautical charting, as shown above.

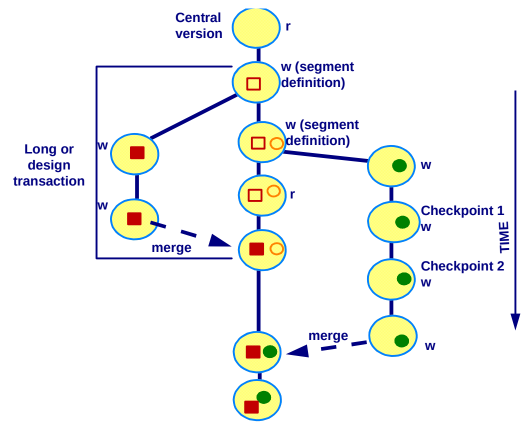
2.5 Object Versioning and Long Transactions

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].



Dataset versions



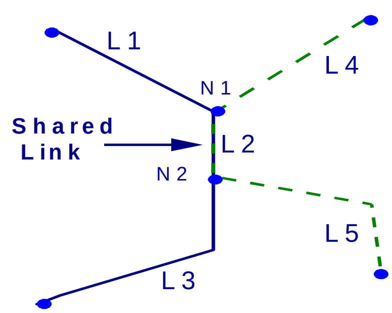
Long transactions

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].

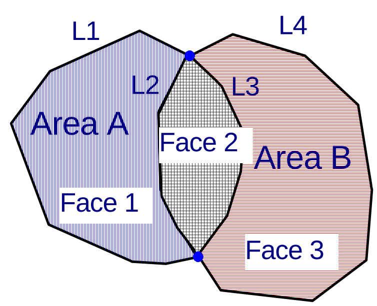
Within mapping, charting and geodata production flowlines, versioning allows efficient multi-user update access to a continuous dataset of master features. In addition, it allows product-specific versions to be subset from the continuum, e.g. to include all the sheet furniture (grids, borders, titles, 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.



Link/Node topological structure



Areas made up of faces, referencing links

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. 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).

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 that 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, 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.
Because views are defined and held on the database, rather than in the application, they are applicable to a variety of kinds of database access, whether for screen display, object search, hardcopy plotting, or export to external product formats. 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 which contain the defining coordinates for the features.

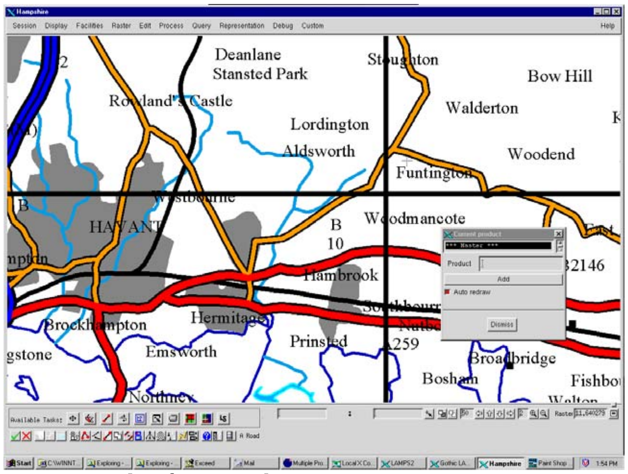
Hence, by overriding the two methods 'set-geometry' and 'get-geometry', it has proved easy to implement in Gothic an efficient mechanism for storing multiple alternative geometries on the object, only one of which is active at a particular time. This mechanism is transparent to the application and allows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

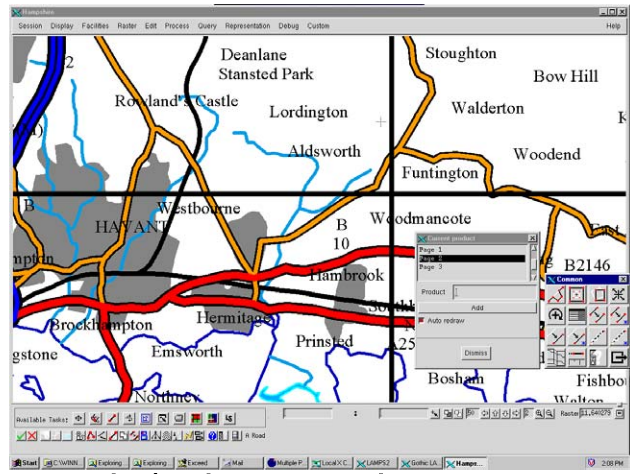
5.3 How are multiple geometry objects used?

This facility can be used for the many situations in cartography where alternatives are needed. In particular it has shown worth in:

- Sheet dependent information, where the constraints of the sheet edge necessitate cartographic amendments to the map data.
- Scale dependent ‘patches’ of modified data, stored in the main dataset, but replacing the true-to-life basic scale data in cases where cartographic generalisation for scale has forced deviation from reality because of constraints of clarity.

The left illustration below shows the master geometry for all objects, while the right 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 for map sheet using Master geometry *Display for atlas page using Alternative geometry*

(Base data courtesy of Automobile Association, UK)

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. The following example shows the same area at three different scales, showing different levels of detail, and different objects for the same entity (e.g. area at detailed scale goes to symbol at small scale).

1:25,000 *1:100,000* *1:250,000*

Map extracts courtesy of IGN France, Copyright IGN.

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 [McMaster, 1991].

In the object-oriented world, this is turned upside down. The map features themselves become objects that 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.

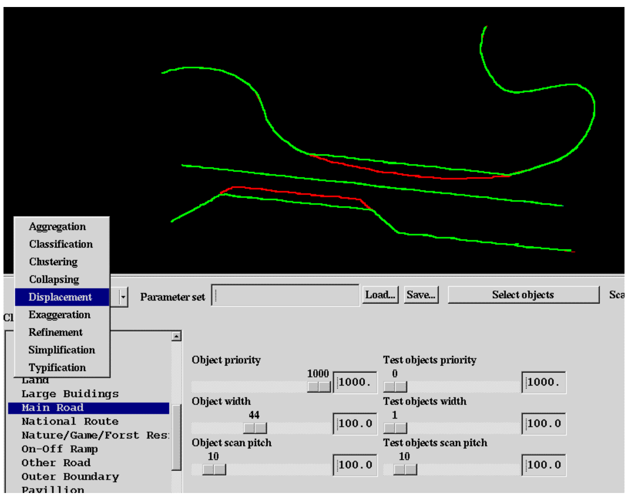
Each such object inherits behaviours from its object class definition and from superclasses in its class inheritance hierarchy. These behaviours allow the object to decide for itself what to do when receiving a message to generalise itself, e.g. it can inspect its relationships with its neighbours to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

One of the fundamental tenets of O-O is polymorphism, in that different object classes may respond to the same message by different method behaviours. For generalisation, this has particular strengths in that the 'simplify outline' generalisation method may have very different behaviours defined for man-made objects like buildings, to natural objects like lakes, even though they are both area objects [Ormsby and Mackaness 1999].

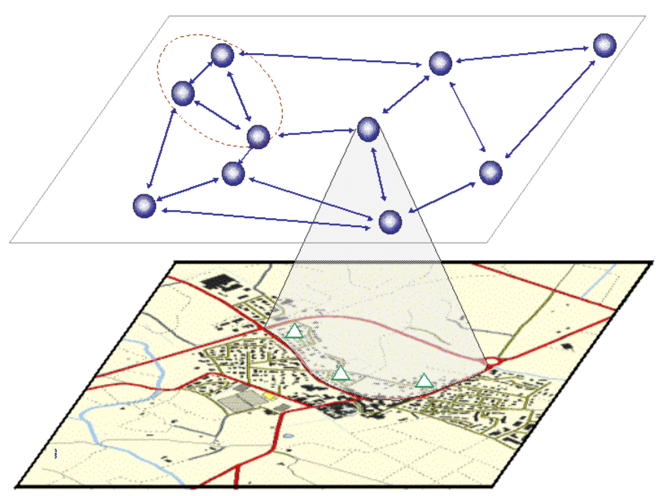
The advent of the object-oriented paradigm therefore opens up new strategies for generalisation [Buttenfield 1995]. These apply particularly to single datasets used for multiple products, but also for maintaining a series of related but distinct datasets [Kilpeläinen 1997], [Harrie 1998].

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 (see figure below left). This 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



Geographic objects become co-operating agents

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 (Edinburgh & Zurich, INPG). Some partners provide in-depth knowledge of generalisation algorithms, while others provide insight into multi-agent modelling. The contract involves 48 person years of effort over a 3-year period.

In this context, agents are self-aware active software objects that 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]. The illustration above right shows co-operating meso-agents handling urban blocks, communicating with the micro-agents that are the buildings and roads.

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.

To allow true on-demand mapping in response to user requests over the Internet, then the decisions that 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 adaptive Internet mapping through O-O and multi-agent generalisation.

7. MAPPING ON THE INTERNET

7.1 Web client-Server

The Web typically relies on a multi-tier architecture. The most common formulation consists of three tiers:

1. Data storage
2. Business logic
3. Presentation

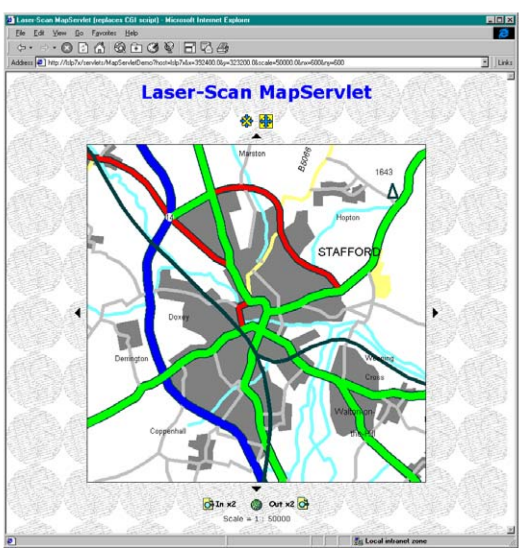
The Internet is used to remote the ‘presentation’ tier from the data and business logic. Furthermore, it is common to expect the presentation tier to be thin, both in its footprint and in its bandwidth requirements in communicating with the business logic tier. The ‘business’ behind most successful geospatial applications, Internet based or otherwise, is a sound and extensible model of the real world. Object-orientation provides the most successful software modelling tool to date. Thus, one can regard an object-oriented database as a natural fit for the data and business logic tiers in a Web-enabled geospatial solution.

To keep the presentation tier thin means that higher level abstractions are required for communication over the Internet. The most common abstraction with geospatial databases is the feature. However in many cases it is necessary to communicate a ‘map’ to the presentation tier, and this is ideal work for the O-O selection, generalisation and representation mechanisms described previously. An object-oriented database therefore is not only capable of answering web queries about the properties of individual features, but can also serve up fully-symbolised maps.

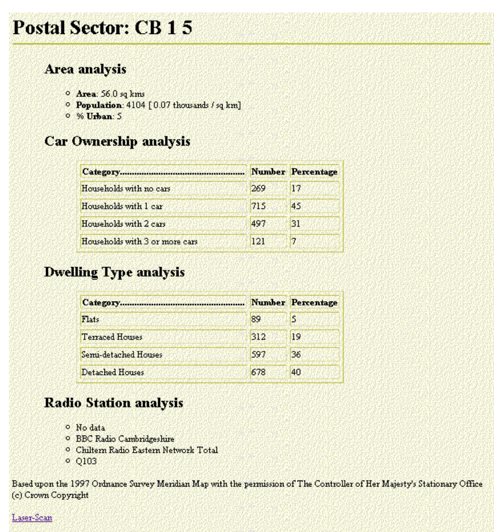
Java applets using a feature communication protocol provide one important way of delivering an easy-to-use presentation tier. A great deal of user-interaction can be handled locally, from simple tasks like drawing out a zoom-box to manipulating individual vertices in a feature geometry. For more analytically rich and cartographically complete situations, it is sometimes better to let the object database take the responsibility for validation and preparation for presentation, and let the O-O server supply a symbolised map as an image to a very thin client. Both types of client are not mutually exclusive, and it is common to server the same object database in both vector and image modes to clients, depending on their capabilities and needs.

7.1 Web presentation of feature attributes and context

Object-oriented techniques also facilitate web presentation of attributes and context. Benefits come because the client does not even need to know what type of feature is being manipulated. All features in an object-oriented database might support a common method that returns an HTML page describing the feature. Furthermore the ability of an object-oriented database to deal with a variety of data-types means that it can be readily customised to return HTML that is generated as required. The illustrations below show web-based access to a Gothic object dataset, and a generated web page showing derived properties of a particular feature.



Web applet displaying Gothic object data
(Data Crown Copyright)



Postal Sector: CB 1 5

Area analysis

- **Area**: 56.0 sq kms
- **Population**: 4104 [0.07 thousands / sq km]
- % **Urban**: 5

Car Ownership analysis

Category......................................	Number	Percentage
Households with no cars	269	17
Households with 1 car	715	45
Households with 2 cars	497	31
Households with 3 or more cars	121	7

Dwelling Type analysis

Category......................................	Number	Percentage
Flats	89	5
Terraced Houses	312	19
Semi-detached Houses	597	36
Detached Houses	678	40

Radio Station analysis

- No data
- BBC Radio Cambridgeshire
- Chiltern Radio Eastern Network Total
- Q103

Based upon the 1997 Ordnance Survey Meridian Map with the permission of The Controller of Her Majesty's Stationary Office
(c) Crown Copyright

Laser-Scan

Derived attributes as a web page,
Generated as HTML on-demand

The O-O paradigm is increasingly fundamental to the evolution of the Internet, as witness the dominant role of Java and CORBA.
A recent workshop was held at the AGI (Association for Geographic Information in UK) on a Java/CORBA approach to delivering geographic information over the Internet, and this approach is described further in [Laser-Scan 1998].

8. BENEFITS OF O-O

8.1 Strengths of Object-Orientation for Cartographic Production

The description of O-O given in earlier sections applies to almost any geospatial data application [Woodsford 1995]. 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, automated generalisation, and on-demand web access.

8.2 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. This obviates the need for tracking of similar changes through multiple product datasets, and hence reduces the costs of product update.
- An object-oriented database provides the ideal middle tier for Internet based geospatial solutions. Object-oriented databases not only provide an ideal framework in which to implement new functionality but their open architecture allows them to be readily adapted to work with the emerging standards of the Web.
- 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.

9. CONCLUSION

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

Recent advances include object database view, multiple geometry objects, and web interfaces. 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

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).

Buttenfield, B. P., 1995, "Object Oriented Map Generalization: Modelling and Cartographic Considerations", in J C Muller, J. P. L. a. R. W., ed., GIS and Generalization: Methodology and Practise, Bristol, Taylor and Francis, p. 91-105.

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
Harrie L., 1998, "Generalisation Methods for Propagating Updates between Cartographic Data Sets", Licentiate thesis, University of Lund, Sweden.
Kilpeläinen T., 1997, "Multiple Representation and Generalisation of Geo-Databases for Topographic Maps", Doctorate thesis, Publications of the Finnish Geodetic Institute, No. 124
Lamy, S., 1997, "Project AGENT", on-line paper at http://agent.ign.fr/.
Laser-Scan, 1994, "The Gothic Versioned Object-Oriented Database: an Introduction". Laser-Scan Ltd, Cambridge UK. See also up-to-date information on the Internet web pages at http://www.laser-scan.com/.
Laser-Scan, 1998, "Delivering Geographic Information over the Internet", On-line paper at http://www.Laser-Scan.com/events/GIS98pages/workshop/index.html
McMaster, R.B., 1991, "Conceptual Frameworks for Geographical Knowledge" in Buttenfield B.P. and McMaster R.B. "Map Generalisation: Making Rules for Knowledge Representation", Longmans.
Ormsby, D., and Mackaness, W. A., 1999, "The Development of Phenomenonological Generalisation Within an Object Oriented Paradigm": Cartography and Geographical Information Systems, v. 26, p. 70-80.
Taylor, David A., 1990, "Object-Oriented Technology: A Manager's Guide", Addison-Wesley Publishing Company.
Woodsford, P. A., 1995, "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.
Worboys, M. F., Hearnshaw, H. M., and Maguire, D. J., 1990, "Object-oriented data modelling for spatial databases": International Journal of Geographical Information Systems, v. 4, p. 369-384.

Note: Some background material used in this paper is updated from that in Hardy, P.G., 1996, "Map Production From An Active Object Database, Using Dynamic Representation and Automated Generalisation", British Cartographic Society Annual Symposium, Keele, UK.

[Original 1999-03-01, Revised 1999-03-29]