DATABASES FOR CARTOGRAPHY AND NAVIGATION

P. A. Woodsford and P. G. Hardy
Laser-Scan Ltd., Science Park, Milton Road, Cambridge CB4 4FY, UK.
Fax: (44)-1223-420044, E-mail: peterw@lsl.co.uk and paul@lsl.co.uk

Abstract

Maps and charts have traditionally been used for navigation since the origins of cartography. In the digital cartography era, navigational use provides the strongest of motivations for database-driven map and chart production, for reasons of security, ease and regularity of update and ability to support both graphical and digital products. Navigational data, for land, sea and air, is also an activity where standardisation has, of necessity, had the most impact to date.

The paper describes the particular requirements of digital cartographic databases for navigational purposes. A key area is that of update, both for atlases and charts and for navigational data. The later has a rapid update cycle requirement, matching that achieved by conventional means such as Notices to Mariners. Update requirements may be quasi-immediate (eg. in military applications) or quantised to approximately weekly updates (as in maritime navigation) or to formally defined changeover cycles such as the 28 or 56 day cycles used in air navigation. As well as implementing such update regimes for navigationally significant information delivered in digital form, the database technology must also support the full content of traditional maps and charts with their less frequent release or print cycles. Change information must therefore be time-stamped, so that, for example, a road atlas can show future planned road changes that will only be reflected in navigational data derived from the same database when the changes actually become effective. In some situations too, it must be possible to reconstruct past states of the database to a greater or lesser extent. The underlying data model must support the structures needed for routing (eg turning restrictions) and the avoidance of hazards and maintain these over long and complex update cycles.

Experience has been gained to date in using object-oriented databases to support the production of both maps and charts and navigational information. Data standards involved include the International Hydrographic Organisation S57 and CEN TC278 GDF (Geographic Data Files) for Road Transportation Telematics and digital roadmaps.

The Central Role of the Database

There are basically two approaches to digital mapping - the graphical, sheet-based flowline which characterises desktop mapping systems, and the database driven approach. Both have their place, and they are by no means mutually exclusive. The overriding need in mapping and charting for navigation is for persistent, validated and maintained data, so there is a strong incentive to follow the database driven approach. Among the benefits arising are the following:

- Continuous data, supporting a flexible set of sheet boundaries for graphical products, and extraction areas for digital products

- Multi-user, multi-product access to the common data base

- A standardised data model, with centralised validation and control processes. In the case of an Object-Oriented data model, these can be incorporated in the database.

- Formalised management of change, particularly if the technique of versioning is available

The mechanisms for the provision of these benefits in an implementation are in practice inter-related. We will discuss a particular implementation - the Laser-Scan Gothic Object-Oriented database as used in the LAMPS2 mapping and charting application.

Data Modelling Issues - The Object Paradigm

In the object paradigm, information is represented as objects, which belong to classes. A 'class' is an abstraction of a set of entities which are typically referred to as being the same sort of thing. An 'object' is an 'instance' of a class. A class may have many instances. Each object has a unique object identifier (OID), which does not change once the object has been created. All that can be known about an object can be retrieved given the OID.

The definition of the class of an object determines both what data an object of that class can hold and how it behaves. Each object encapsulates its structure together with the methods that can operate on that structure. Objects hold values, which may be stored properties of a simple type ('attributes') or derived as the result of behaviours ('methods'). In the Laser-Scan Gothic OODB, values can also be geometries, rasters, OIDs or sets, lists or bags of OIDs. Since OIDs are invariant, they are in effect, pointers. In Gothic, values which point to other objects are always paired with another value on the remote object, which points back. Such pairs of bi-directional connections between objects are called 'references' and the integrity of references is maintained at all times by the system.

References are used to define one-to-one, one-to-many and many-to-many relationships and to set up structured data models with shared geometries, such as those required in GDF, DIGEST and the IHO S57. References also provide navigational access to the database, allowing algorithms such as those for spatial indexing and topology handling to be efficiently implemented.

The Active Database

The fact that an OODB can hold behaviours as well as attributes and structures has a profound effect on data management The database becomes an active database, incorporating in a generic manner much of the functionality that would otherwise have to be supplied at the application level

The active database can be used to hold the rule base appropriate to a particular activity, as well as to hold the formal data model. The database schema defines the data model and can also hold relevant business rules and data integrity procedures. The use of 'inheritance' to provide controlled sharing of structure and behaviour in a hierarchical manner is a powerful technique for managing data and for the creation and maintenance of databases.

In the Gothic system, topology rules are defined on a per-object-class basis, with associated snapping tolerances. Topology is maintained on the fly. Other rules are enforced by means of behaviours, invoked by 'methods', which implement business rules (e.g. to ensure that navigational aids are composed only of valid combinations of elements or that turning restrictions are applied and modified in a consistent manner).

An object encapsulates both its state and its behaviour in a manner that is common and consistent across all applications accessing the object. Applications need only be concerned with high-level relationships. The object lifecycle concept provides a framework which makes it easier to define and maintain complex data models.

Within the lifecycle of an object a number of key stages can be defined, as follows:

- construction of a new object

- modification of an existing object

- destruction of an object

- interaction with another object ('touch' another object)

The object manager uses 'reflexes' to trigger the appropriate methods at each stage in the lifecycle. Certain object lifecycle reflexes are of a fundamental nature and are defined at the base of the inheritance tree, so as to be available to all object classes. These include reflex methods for:

- constructor

- validation

- dependency

Authority checks are incorporated in the construction, modification and destruction processes. The constructor will also allocate the OID and set up initial values in a class specific manner.

One of the key problems within a database is maintaining integrity. Since integrity constraints vary from class to class, methods provide an ideal way of implementing them. Validation methods are invoked at the end of each transaction, and validate only those objects implicated in the transaction, as identified by 'dependency' methods. If in the process of committing the transaction, validation fails, then the transaction can be 'rolled-back' to a consistent state, or further changes can be made and validation attempted again.

The role of reflex methods within the object lifecycle framework is not restricted to maintaining database integrity in the narrow sense. They are also used to provide rule-based data quality checks (e.g. to ensure that contours do not cross, or that attributes are within range) and to apply business-specific rules

Since the active database ensures that reflex methods are invoked within all applications accessing it, they can also be used to automate the production of audit trail information, if required down to the per-object transaction level, and to automate the preservation of previous states of objects if necessary.

Databases and Versioning

Database versioning is an additional key current advance in the technology. It provides support for very large continuous spatial databases in an economical and manageable form. Versions are maintained in a tree structure, with only delta change information recorded rather than complete copies. Multi-user update is supported, with each update process accessing its own logical copy of the complete dataset, without the overhead of providing a physical copy to each process. The concepts are illustrated in Figure 1, which also shows how versioning provides long transaction support.
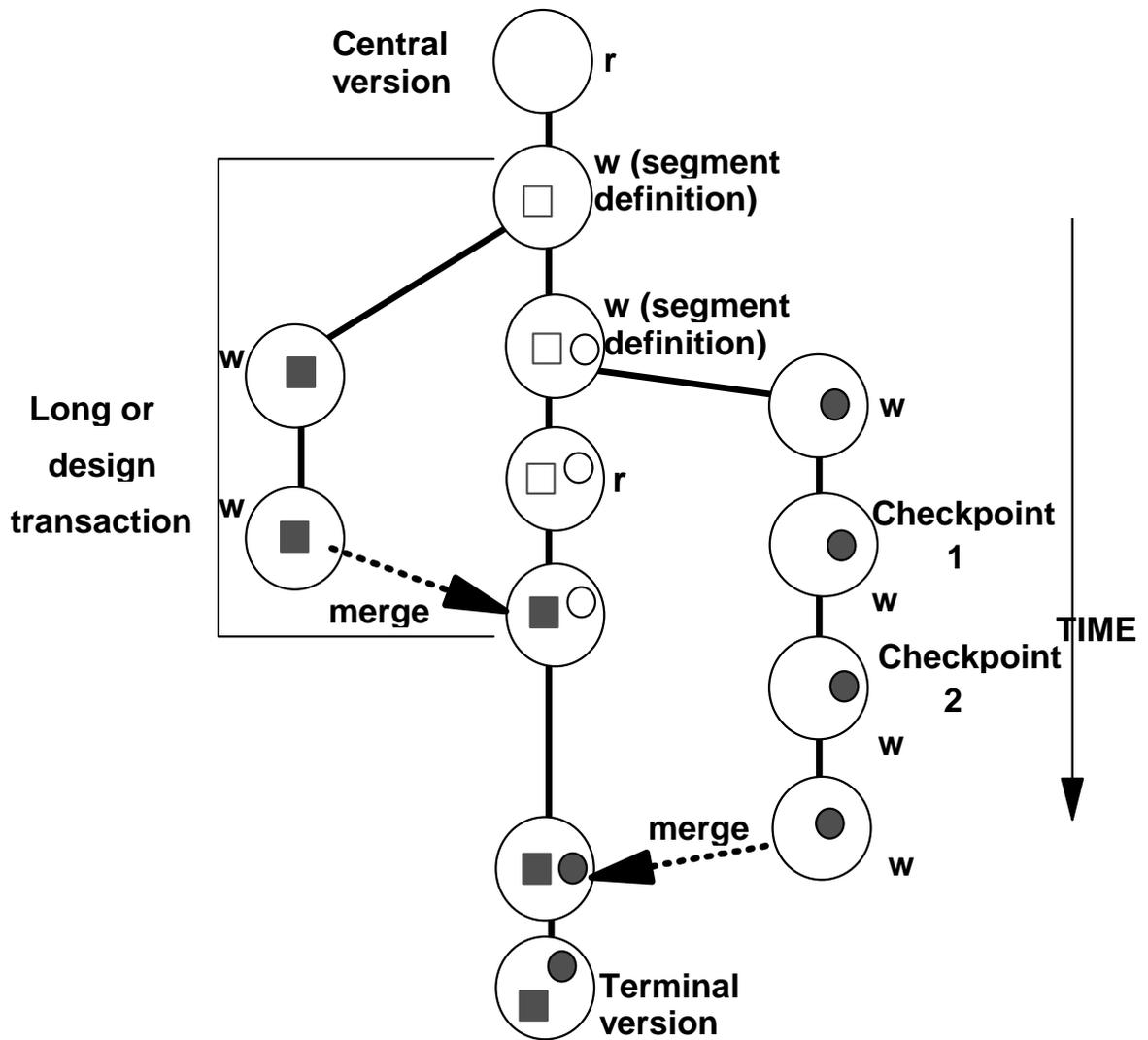
Figure 1: Versioning and Long Transaction

Validation methods are used to assure local data integrity, within the long transaction. Merge methods are used to assure global data integrity before the final commit at the end of a long transaction. Since the version holds an explicit record of all changes

made in the transaction, in delta form, the merge method can readily summarise changes for management information and metadata records purposes and for transmission in digital form to other update processes and to data users. In short, merge methods, as well as ensuring overall data integrity, play a crucial role in the overall information flows shown in Figure 2.
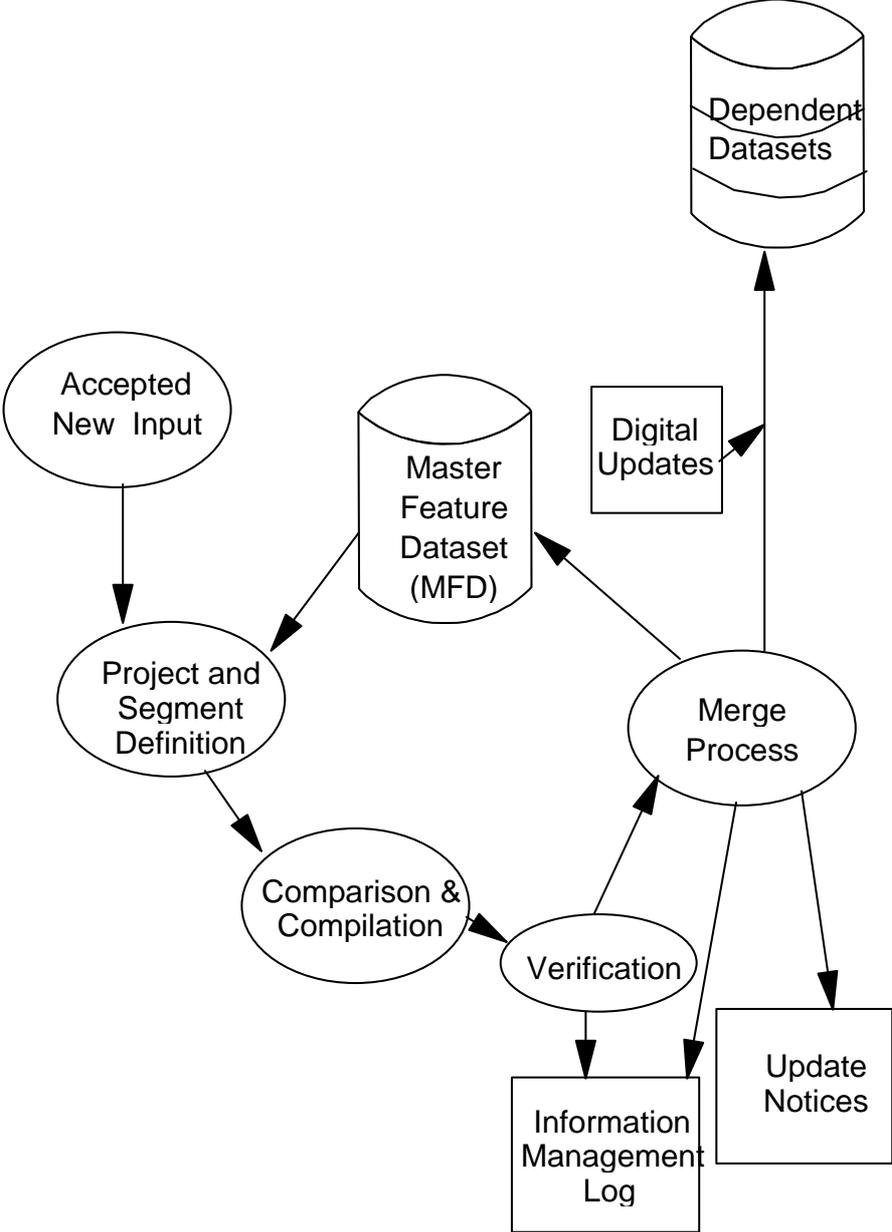
Figure 2: Information Flow and Update

Performing and Issuing Updates

The implementation of the overall information flow of Figure 2 above can be readily achieved using the Object paradigm. It provides a basis both for performing updates and for issuing updated information or updates to users. Versioning can be used to support different update scenarios, that can be made to 'go live' at controlled times depending on the Update regime required. The fact that each update can be formally reviewed as it is performed provides the opportunity to assess its consequence, and the ability thus to determine whether it requires immediate notification to users. Alternatively, the fact that versioning holds delta changes can be exploited to batch up changes against a particular update release cycle. It also provides the means for version comparison, to establish changes between particular versions.

The realisation of Public Unique Object Identifiers (PUOIDs) is the key to providing Incremental Update. OIDs are used internally by a system such as Gothic, and PUOIDs can be provided and maintained. More complex issues are involved when there are multiple data issuers, and in practice it is simpler to have a single issuer for any particular data domain. Navigational data users require a high level of update service and often carry large coverages of navigational data. There is therefore much to be gained by an efficient implementation of Incremental Update, as for example is about to be trialled in the Hydrographic community. PUOIDs also have a key role to play in establishing system Interoperability, as is being addressed by the OpenGIS initiative (where the terminology used is Feature ID rather than Object ID, in the interests of generality). A more complete treatment of Update issues is to be found in [4].

Chart Production

The object paradigm provides powerful support for cartographic representations derived from the data model and the information held therein. These techniques are described in the companion paper in these proceedings [1]. It also provides an appropriate framework for the implementation of rules-based procedures such as those required for automated text placement and cartographic generalisation [2],[3].

Conclusions

Navigational applications are characterised by rich, formalised data models, by a strong requirement to enforce data integrity across complex update sequences, and by the need to support a variety of update regimes. In addition to meeting the new demands for digital data; systems and data models have to support graphical products to at least the established specifications, with the added need for more flexibility. The Object paradigm is well suited to these requirements, and practical implementations using it are rapidly becoming established.

References

1. Hardy P. G. and Woodsford P. A., 1997. 'Mapping with Live Features: Object-Oriented Representation', ICC97, Stockholm, Sweden, June 1997.

2. Mackechnie G. and Mackaness W. ,1997. 'Detection and Simplification of Road Junctions in Automated Map Generalisation', 1997 ACSM-ASPRS/Auto-Carto 13 Conference, Seattle, Washington, USA. to appear.

3. Ormsby D. and Mackaness W. ,1997. 'The Development of a Phenomenological Approach to Cartographic Generalisation', 1997 ACSM-ASPRS/Auto-Carto 13 Conference, Seattle, Washington, USA. to appear.

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