Database Design in Migration from Traditional to Object-Oriented GIS - the Evolution Story of the Topographic Database of Finland

Ms. Leena Salo-Merta
Helsinki University Of Technology,
Laboratory of Cartography and GIS
Genimap Oy
Postitorvenkatu 16
FIN-33840 Tampere
Finland
Fax +358 2001 340 589
Email:Leena.Salo-Merta@genimap.fi

Mr. Ari Tella
National Land Survey
of Finland,
Development Centre
P.O.BOX 84
FIN-00521 Helsinki
Finland
Fax. +358 20541 5454
Email: Ari.Tella@nls.fi

Mr. Ilkka Vanhamaa
National Land Survey
Of Finland,
Development Centre
P.O. BOX 84,
FIN-00521 Helsinki
Finland
Fax + 358 205 41 5454
Email: Ilkka.Vanhamaa@nls.fi

Abstract

Data intensity brings special challenges to the development of organisations' next-generation GISs. The paper addresses database design and schema evolution of existing very-large geospatial datasets, using the Topographic Database of Finland (TDB) as a case study. The focus is on the latest step in evolution, the change from traditional file-based system to GE Smallworld's object-oriented GIS. The migration strategy, database design, populating the new database and further data restructuring processes are described. The practise of committing to the old in the design for the next-generation system and the stable nature of 'data infrastructure' are concluded.

1. Introduction

1.1. Challenges in the Development of the Next-Generation GIS

Geographic information systems (GIS) in computerised environments have become a fundamental part of the work of many federal authorities and mapping agencies. *Data intensity* is a characteristic feature of these systems. First of all, geographic databases are usually very large (VLDB). Secondly, data appears in several different forms like vector or raster and it may be in 2, 3 or even 4 dimensional spatio-temporal form. Thirdly, the management of geographic data, e.g. vector topologies, is often complex. Fourthly, the lifetimes of the datasets tend to be long. Even the first-phase data acquisition for a nation wide dataset may be a several year project.

Data intensity brings special challenges to the development of organisations' next-generation GISs. The lifetime of a dataset often exceeds the lifetime of the database management system (DBMS) that is used to maintain it. Database technology has matured to a stage where commercial object-oriented and object-relational DBMSs for GIS are available and make it possible to implement elaborate object-oriented data models. This paper addresses database design for the next-generation GIS and schema evolution of very-large geospatial datasets. The Topographic Database of Finland (TDB) is used as a case study.

1.2. Schema Evolution within a DBMSs Life-Cycle

We distinguish schema evolution within a DBMSs life cycle from the changes connected to the change of the DBMS. The need to make changes in the data model emerge during the lifetime of most systems.

Performing these changes are called *schema evolution*, and the DBMSs functions that support it are called schema evolution facilities.

One proposed taxonomy for schema evolution is to distinguish changes according to the *type of the schema object* to which the change is addressed. This leads into three main classes of changes:

- Changes to the *components of a class* (that is: to properties and methods)
- Changes to *relationships* of classes
- Changes to *classes themselves*.

The taxonomy is elaborated by considering the *type of the change*, which may be adding, dropping, modifying or inheriting. (Banerjee et al. 1987 according to Wachowich, 1999).

Schema evolution is fairly easy during the development phase of the system, but problems may arise in practise if the database is already populated and used in production. For example, adding an optional attribute to an existing database class that has 20 000 instances is simple as a technical operation. Committing a single schema operation makes the data model richer, but the data itself remains just as poor as it was - 20 000 missing values for an attribute. On the other hand, schema operations that alter the structure of the data may be prohibited or corruptive for a populated database. The only option is to export the data to an external file and then import it back into the modified structure. Scheduling tasks of this kind for a VLDB in operational use is difficult and the schema is kept as stable as possible.

1.3. Database Design for the Next-Generation System

The change of the underlying DBMS brings a natural breakpoint where to re-design the schema. In theory the database design process contains conceptual, logical and implementation level modelling, which should also be included in the development of the next-generation system. In practise the design process does not follow the waterfall model only, but the design is influenced by the previous systems solutions on each level (Fig 1). Translation of geographic VLDB from one system to another is a major effort, even though the structural changes in the design are kept as minimal as possible. Converting the existing data into the new system as quickly and painlessly as possible is usually a key question in an organisation's DBMS implementation project. Commitment to the old design may become a critical factor in the design of the application data model for the next generation.

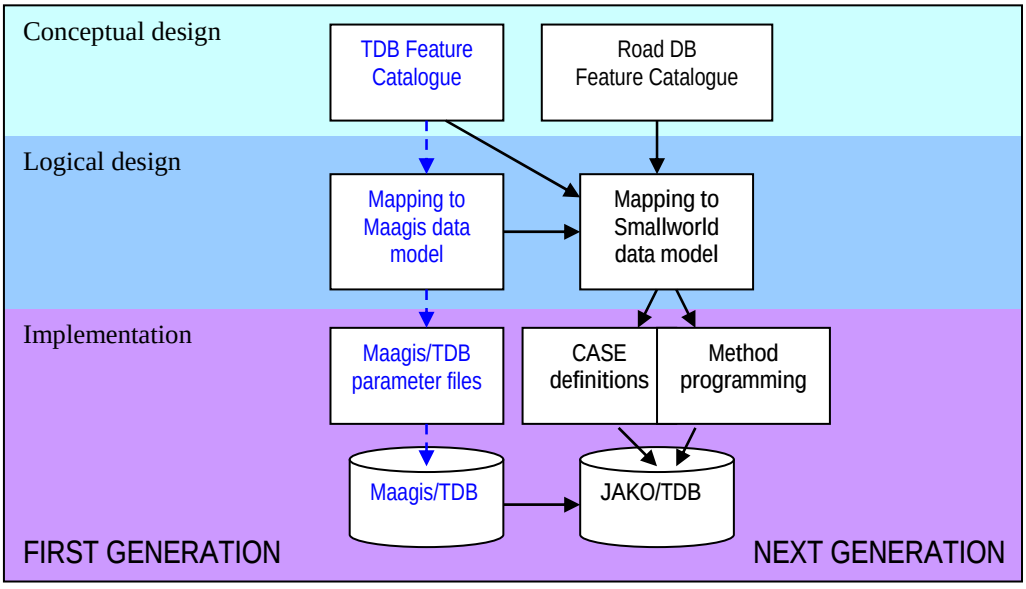

Fig 1. The database design process for the next-generation Topographic Database.

2. Evolution of the Topographic Database of Finland

2.1. General Description of the Topographic Database

The Topographic Database comprises the most detailed general topographic information of Finland with full coverage. TDB is one of the general mapping assignments of the National Land Survey (NLS), financed by the State.

TDB is primarily a vector-based geodatabase with a nominal scale of 1:5 000 – 1:10 000. The data acquisition started in 1992 and the coverage is currently 85 per cent. The content of the TDB is published in a feature catalogue (National Land Survey, 1996), with more than 100 geographic features. TDB is not entirely a map dataset, although its content is closely related to the tradition of topographic mapping (Fig. 2). Furthermore, it is a multi-product database and the source for printed topographic maps in scales 1:20 000 and 1:50 000 as well as a set of digital products. In some cases the small-scale datasets are shown as part of the TDB, but we prefer to leave them out because they are managed by a different DBMS with no true integration to the core TDB.

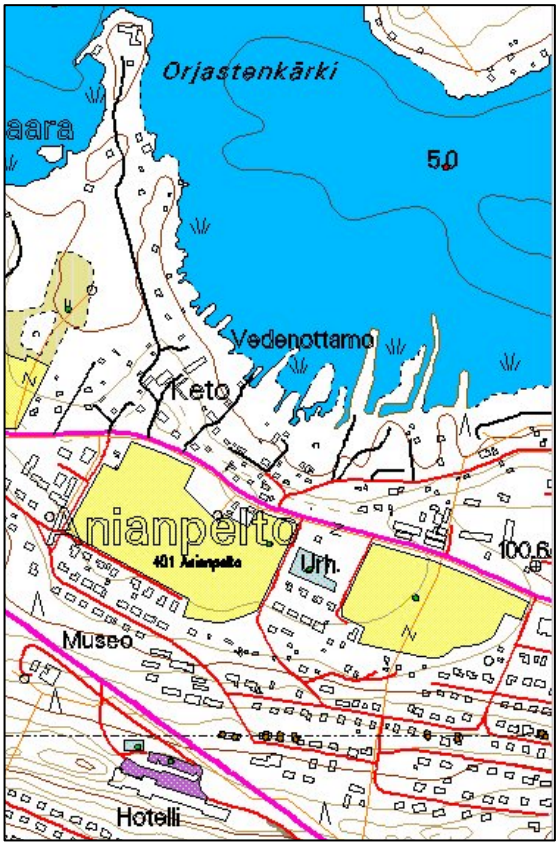

Fig 2. An example of TDB data

The NLS is responsible for both topographic and cadastral data, which has been managed separately by different departments. The NLS has a strong tradition of in-house software development. The first generation of digital cadastral maps was based on Maagis software, an in-house-built, file-based system for managing and processing geographic information. In the mid-90's the next generation cadastral data system, called JAKO (Tuomaala et al., 1998, Myllymaki, 1998), was developed on the Smallworld platform (GE Smallworld, 2001). The main arguments for choosing this system were support for long transactions and database distribution, good performance on large datasets and a sophisticated application programming interface with GIS functionality.

2.2. Steps of Evolution

The history of the TDB follows the same baseline as the cadastral system that comes to the GIS platforms. Three phases can be identified (Fig. 3): The first phase was the *establishment* of the TDB. The core of the system was Maagis software.

The second phase called the branching phase began in the mid-1990 due to emerging customer-driven needs to improve the content, quality and delivery of road information. This lead to the parallel use of several GIS environments and redundant datasets with severe updating problems.

The third phase can be considered as the *unifying* phase. It is unifying in two senses – first of all, it brings the major geodatabases (TDB and cadastral database) of the NLS back to shared GIS platform. Secondly, the branches of road datasets have been merged back to the core of the TDB. This major data restructuring process has been recently carried out in the National Land Survey of Finland as the object-oriented Smallworld GIS has been introduced as a new database and production environment. The new topographic information system is called Topographic Data System JAKO (JAKO/TDS).

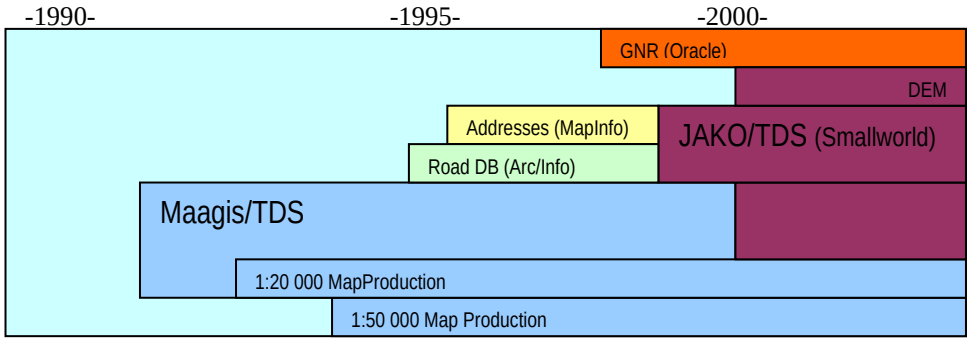

Fig. 3. Evolution of the Topographic Database of Finland.

3. Migration Strategy for the Next-Generation System

During the past decade the production system around the TDB has grown to cover a whole range of tasks from stereo digitising to map publishing. It consists of several specialised subsystems with tailored functionality. Replacing all of them at one go was considered too much, regardless what the new GIS tool might have been. A *migration strategy* was needed to breakdown the effort into smaller tasks. The migration started with the most urgent case – merging the three road datasets. The application for managing road data was designed and built in winter 1998-99. The major components of the system were the new database, data translator module, and interactive editing tools for merging the road objects that originally came from different sources and had different attributes and structure.

The second step in migration was to upgrade the application so that it would serve as the whole TDB's DBMS. The data acquisition and editing environment needed to be renewed as well to support 3D data. A Finnish company *ESPA Systems Oy* incorporated 3D digitising facilities into Smallworld software via a stereo workstation data transfer link (ESPA Systems, 2001). Both of them were challenging projects, but they still did not cover the full range of tasks. Smallworld is not designed especially for map production. TDB/Maagis has adequate applications for producing topographic maps in 1:20 000 and 1:50 000 in operation, including cartographic representation, some simple but well-proven generalisation processes, and interactive editing, proof plotting, generating layouts and legends and postscript output for film plotting. So the short-term goal was not to give up the TDB/Maagis system completely but to change its role from a data storage and primary production environment to a map publishing system only.

4. Database Design for Geographic Features

The production environment of the next generation system included new datasets: a digital elevation model (DEM), orthophoto database and also small-scale raster datasets for zooming-in the working area. When considering the geographic features *at the conceptual level* as they were described in the TDB's feature catalogue, no significant changes were introduced. The scope of the project was the technical production environment and thereby neither additional geographic features were introduced, nor were the classifications changed.

Keeping the mapping between the classes of the old and the new design as straightforward as possible was essential, because the data was translated forth and back between the systems. The implementation of the classes was quite different due to the diverse nature of the data models of the systems, and there was a need to pay a lot of attention to the validity of the data.

4.1. Differences in Object Models

The Maagis database was based on a *hierarchic data model*. The user-defined real-world object classes refer to *base object types*: linestring, area, text, and two different symbol types. Base objects have a fixed

record structure with a limited set of attributes. Linestrings consisted of a header record and a set of line points and possibly symbol points. *User classes* were defined to belong to groups of *'topological universes'*, levels, at which topology could be formed. Topological relationships were stored as direct references in the object records. The same record structure was used for both objects with topology and simple geometries. The formation of topological relationship was controlled by a user-defined distance parameter called working tolerance. There was no support for complex objects and topology split geographic features of line-type into several linestring objects in the database. Data was physically stored in map-sheet files, and this caused the fragmentation of real-world-objects at the edges of the map sheets. The management of area-type objects was the speciality of the Maagis system. Only reference point and attributes were stored for an object of area-type, but the DBMS managed the area 'on-the-fly' by searching the bounding topological linestrings around the reference point. This was convenient for digitising, because the area boundaries were truly shared and could also have attributes of their own.

The object model of Smallworld is more flexible and sophisticated in many senses, and the application programming language *Magik* is an object-oriented language. Because Smallworld VMDS is a *relational database*, there are no limitations to the number of attributes of a class. Objects of classes may be joined with other objects, and defining methods on classes is also possible. In addition to basic data types, the type of an attribute may be a *codelist* or a system object class called *geometry*. One user class may have several geometry type attributes, and one user class instance may have several geometry type attributes simultaneously. In the case of alternative geometries, a class method is programmed to decide which one to choose. The system provides a true continuum of tile-free geographical space and line-type geographic objects that support topology, even though they do not have to be split into topological primitives. Smallworld uses the concept *manifold* for a topological universe. The database designer defines the manifolds in the Smallworld's case tool and sets the topological rules between the classes within the manifold. Depending on which rules are chosen, the topological operations either split the real-world-object or keep it as a whole, storing the topological primitive objects as an underlying layer.

4.2. One-to-One Mapping Using Class Templates

The basic principle in the design was to create one-to-one mapping both *between the features of the feature catalogue and the database classes* and between the *old and the new design*. Although this meant that a large number of classes had to be dealt with, it brought some significant advantages as well. First of all – the semantics was clear at all levels of design. Capturing objects from different feature classes into one database class is not sound, although they would have similar record structure and behaviour at the moment. By doing so, one might save a little time when constructing the database, but what would that class really be and how would one deal with it's need for evolution in the future? We thought it wiser to treat the *database class as a logical unit in the design* and specialise a class rather than specialise within a class. There are other mechanisms to deal with the number of classes. For example, *class templates* that contain the attributes and methods common to several classes can be utilised in the creation of new classes. We defined class templates for linestring, area, text and symbol types.

4.3. Shared Geometries and Area Management

One speciality of the TDB has been the modelling and management of shared boundaries. Area objects that belong to the same topological universe truly shared the common boundary lines, and the boundary lines may have attributes of their own. In the previous system the boundary line could be classified as a real-world-feature (e.g. a river or a ditch) or as a boundary line only. Actually, one attribute of a boundary line is a method by nature, it is called *cartographic code,* and is used for defining the representation of the line (e.g. "draw me as a field boundary", or "draw me as an indefinite border of a silting water area"). The cartographic codes were set by running a process based on an area priority stack.

In the new design, two types of geometries were introduced for area type features: area and seed point. The areas were constructed by using *area seeding* technique, and both the area polygon and the seed point were stored in the database. The fluent updating and management of areas and shared boundaries was one

of the major challenges in the design. A set of application tools has been developed for editing the topological nodes without dropping and re-creating area geometries. Also Smallworld has developed the area management capabilities of its product significantly. The cartographic codes were implemented by using class methods, based on the same area priority principle, but with more dynamic database management.

4.4. Complex and Associated Objects

By complex objects we mean here other more fundamental associations between object classes than system supported topology, and the simple storage of geometries as classes' attributes. Additional cartographic objects can be classified into different categories (Salo-Merta, 1999). Additional cartographic symbols that belong to a line-type geographic feature were 'joined' to the master object with a topological rule. Smallworld supports the orientation of symbols according to the direction of the line for objects with topological relationships. Other additional cartographic objects (e.g. symbols for area objects, explanatory texts) were not associated with the real-world geographic features as a main rule, a couple of exceptions were made though, but they can be considered as curiosities. Joins between user-classes were defined as optional, and cascading delete was implemented on the application, not as a schema rule.

4.5. Transaction Management

Smallworld supports long transactions based on optimistic transaction policy and database versioning. The different versions that the users see are called *sub-alternatives*. Long transaction is started by a 'create alternative' operation, which branches a new sub-alternative for the user. During the long transaction the sub-alternative can be updated by 'merge changes' operation to see the changes that have taken place in the top alternative. The long transaction is committed by posting the changes and merging the sub-alternative back to the top.

Validation of constraints was implemented in the application program, rather than as schema rules, because different constraints are applied to the valid state of a completed long transaction, than during the intermediate states within long transactions.

4.6. Temporal Elements and Metadata

Two types of temporal elements were introduced in the design. Firstly, each class has two attributes for time stamping: birth date and death date, representing database transaction dates. Secondly, a concept of history object was introduced. When a long transaction is committed, each object that has been changed during the transaction, is copied to a history object. It is not an identical copy; the history object has no topology, it gets a new unique object identifier, and its death date is set. The 'present' object gets a new birth date, but there is no direct linking between present and past objects.

This idea of temporal elements was copied in somewhat simplified form to JAKO/TDS from the cadastral system JAKO. JAKO links the present and past objects to keep record of the full history of parcels and real estates. With topographic data that was not regarded necessary, timestamps and preserving the past stages of objects as history objects make it possible to resume a certain date's state in the database (or actually, come quite close to it) if there is a need to look back. So far the utilisation of temporal elements to serve the updating of small-scale databases is under consideration.

In the TDB two *metadata* elements are brought to attributes on the instance level: the positional accuracy code, which has been present already from the very beginning of the TDB, and the accuracy code for the z co-ordinate, which was inserted now. The rest of the meta-information, consisting the lineage, is stored into objects called production plans. The management of *production plans* and single *tasks* within a production plan plays a key role in the system, but is not in the scope of this paper.

4.7. Connections to External Databases

Shared GIS platform enables the cadastral system JAKO to view and copy data from the topographic system, and vice versa.

The NLS keeps geographic place names in a centralised database: *Geographic Names Register* (GNR) (Leskinen, 1999). It is a register-type database implemented on Oracle. The TDB is the master dataset for place names, and updates are propagated to GNR. GNR stores more information about place names than the TDB, e.g. details for producing small-scale maps. For tracking the details of the update, each place name attribute in TDB is time-stamped, and with this information an application generates the SQL-script that updates the GNR.

5. Populating the Database

5.1. Primary Data Loading

The primary data loading was started in December 1999 with 30 NT workstations and 12 personnel and was estimated to take six months. Table 1 shows some figures about the database's volume. A rectangle of 30 x 40 kilometres (1:100 000 map sheet) was used as a loading unit. (Tella, 2000).

Table 1. The volume of the TDB in figures. (Gigabytes represent the size of the database in April 2001, numbers of objects are estimations for the full coverage.)

DATABASE SIZE IN GIGABYTES		DATABASE SIZE IN NUMBER OF OBJECTS	
Raster datasets		Areas	9.5 million
• Small-scale maps	4 GB	Symbols	8.5 million
• Orthophotos	125 GB	Texts	1.5 million
• Digital elevation model	4-5 GB	• Incl. geographic names	0.7 million
Vector dataset	102 GB	Linestrings	45 million
		Spot heights in DEM	5 600 million

The first tasks were pre-load operations, including edge matching in the Maagis/TDS and the generation of transfer files. The software for both the transfer file generation and import into the new system were programmed in the NLS. An ad-hoc transfer file format with record structures similar to the database record structures of the old system was used. The translation was managed in three transfer files per loading unit: *1. Area features, 2. Contours* and *3. Other features*. The data of a loading unit was imported into a new sub-alternative of JAKO/TDS. The area-type features were reconstructed using the area seeding method. A set of validation tools was developed to catch objects with invalid structure or classification errors. Post-load validation operations for analysing possible problems in geometry and topology are described in fig. 4. The inconsistencies noticed were corrected at the workstation before the sub-alternative was merged to the top. It was laborious, but significantly improved the quality of the data. The true throughput was roughly one week for a loading unit (Tella, 2000).

- Overlapping linestrings (1)
- Intersecting linestrings (2)
- Self-intersecting linestrings (3)
- Linestrings with incomplete connectivity (4)
- Reference point of an area too close to a linestring (5)
- Short linestrings (6)
- Linestrings that were dropped in the area seeding process (7)

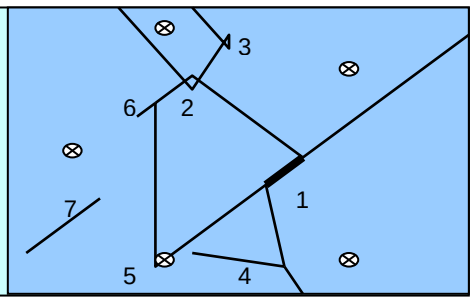

Fig 4. Post-load validation operations for object geometries.

5.2. Further Data Restructuring

Further data restructuring was carried out later to take advantage of the spatial continuum. Area features that had previously been fragmented by mapsheet's edges were merged with each other, as well as road objects that connect with each other topologically and have same attribute values. For the rest of the features, restructuring was not considered important.

6. Conclusions

In an organisation's DBMS implementation project the translation of the existing VLDB into the new system as quickly and smoothly as possible is vital. The need to minimize changes in the interfaces to other systems in operation are another matter of concern. Commitment to the old data and existing interfaces may become a critical factor in the design of the application data model for the next generation. This practise makes the 'data infrastructure' even more stable in nature.

In the design of the next generation database for the TDB of Finland, keeping the mapping of classes between the old and the new design as straight-forward as possible was essential, because the data was to be translated back for map publishing. However, it does not involve the implementation of the classes themselves, and a richer data model could be implemented. There was a need to pay a lot of attention to the validity of the data, and post-load validation operations became an important part of the process. Translation of a VLDB is always a major effort, even though the steps of migration are made small.

7. Acknowledgements

The authors wish to acknowledge Metsamiesten Saatio Foundation for funding the participation of the ICC2001 conference and Mr. Henry Methuen for checking the language.

8. Authors' Contributions

This article is based on the experiences on the development of the Topographic Database of Finland. The responsible author Leena Salo-Merta was employed by the NLS during 1990-2000 and in her last project in NLS she was the key designer of the TDB database for the Smallworld system. Ilkka Vanhamaa carried out the major part of the implementation work and has assisted in writing chapters 4 and 5. Ari Tella was the project manager and has acted as NLS's supervisor during the writing process.

9. References

ESPA Systems Oy (2000). http://www.espasystems.fi/
GE SmallWorld (2000). http://www.gesmallworld.com/
Leskinen, Teemu (1999). National Place Name Register Integrated with Cartographic Names Register for Multiple Scales and Products. ICA/ACI Conference Proceedings, August 1999, Ottawa, Canada.
Myllymäki, Tarja (1998). How to update a Cadastre – a Long Transaction in Modern Cadastral System. FIG Conference Proceedings 1998.
Salo-Merta, Leena (1999). Object-Oriented Modelling Approach on Geo-Databases and Cartographic Products. ICA/ACI Conference Proceedings, August 1999, Ottawa, Canada.
National Land Survey (1996). Maastotietojen kohdemalli. Maanmittauslaitoksen julkaisuja 71. Maanmittauslaitos, Helsinki, Finland. (Feature Catalog of the Topographic Database. In Finnish.)
Tella, Ari (2000). JAKO/TDS – The National Topographic Data System In Finland. Smallworld user conference 2000, Freiburg.
Tuomaala, Juha & Uimonen, Mikko (1998). Introducing the New Object-Oriented Cadastral Infromation System (JAKO) of Finland. FIG Conference Proceedings 1998.
Wachowicz, Monica (1999). Object-Oriented Design for Temporal GIS. Taylor & Francis.