Computers, Enviornment and Urban Systems, 28 (2004), S. 471 - 486.

Processes in a cadastre

Gerhard Navratil Andrew U. Frank
Technical University Vienna, Institute of Geoinformation
navratil/frank @ geoinfo.tuwien.ac.at

Abstract
A cadastre is a system of major importance for economy and planning. A cadastre provides data on land. It is the basis for legal aspects like ownership as well as fiscal aspects like taxation of land. The cadastre also provides data for planning assignments (for example, boundaries of constructions, land use, and soil). Storage and update of these data require a complex system that had been developed in Austria during more than 100 years.
Understanding, using, and improving a cadastre requires knowledge on the cadastral processes. The problems a cadastre must solve are important to understand the needs for a cadastre. It is also important to understand the processes of a cadastre to see how a cadastre works. These processes define the way a cadastre handles data and what prerequisites the data must fulfil to be accepted by the cadastre. Improving a system requires analysis of the processes. The user wants to work with a cadastre. He needs processes that meet his demands. Improving the efficiency of the processes, therefore, improves the efficiency of the cadastre directly because then the user will be satisfied (his work will be done better or faster).
The paper discusses the tasks of a cadastre. The starting point is the needs of users (owners, government and many others). The needs define the tasks and the data needed to fulfil the tasks. The next step is the definition of the processes to fulfil the tasks. The paper then formalizes these processes. Finally, implementations for two different cadastral systems prove the general validity of the processes.

1. Introduction

A cadastre is the basis for solving three different social needs. Cadastral systems developed from the need for a **guarantee of rights** (ownership, mortgages) and a **fair taxation** based on the size of parcels. Today a cadastre is also used as a **basis for planning assignments** like the dedication of land.

The organization of the cadastre developed during a long time. The Austrian cadastral system, for example, started in the middle ages as a simple list naming areas of land and their possessors (not the owners). The possessor is the person using the land while the owner is the person that may do anything with the land (including sale). Each part of the country had lists that looked different because there were no uniform rules for the system. The system was replaced by a uniform system in the 19th century, which was designed to provide more information (for example, ownership and encumbrance information, and a graphical description of the boundaries). Legal changes made it necessary to add planning data in a limited amount. Today the system is computerized. However, the principles of the system and the administrative structure are almost the same as in the 19th century (Lego 1968).

Improvement of the organization requires analysis of the cadastral processes. An effective method to improve an organization is business reengineering (Hammer and Champy 1995). The method starts with an analysis of the processes and the user's needs and then regroups the organization to improve the efficiency of the processes.

The paper discusses the processes of a cadastre. The assumption is that it is possible to define general processes that, in principle, will fit for each type of cadastral system (Frank 1996). Therefore the processes are general methods without relation to a specific cadastral system. The focus of the paper is on the ownership data but the processes are similar for the other parts of the cadastre.

2. Organization of a cadastre

This section contains basic information on cadastral systems. An important part of a cadastre is the land registration, which handles ownership and encumbrance data. Section 2.1 provides a discussion of the most contrary types of land registration, the registration of title and the registration of deeds. A cadastre also provides data on land. Section 2.2 discusses the different types of data and how they are connected with land. Land and constructions on land can be seen as objects and people as subjects, which are a special kind of objects because they can act themselves. Section 2.3 shows these subjects and objects and the links between them. Finally, section 2.4 discusses the types of documents registered in a cadastre.

2.1 Land registration systems

There are two opposite types of land registration systems (Dale and McLaughlin 1988; Hofmeister and Auer 1992) with many mixed systems between them:

- Registration of title (for example, Austria):
  The registered (publicized) rights serve as the basis of this system and only these rights are valid to a third party. That principle works in both directions. Everyone may rely on the data listed in the cadastre, but there is also the compulsion to inspect that data (for example, to check if the person selling land is the owner).
  Documents like a contract or any other **textual** (for example, a certificate of heirship or a judgment) or **graphical document** (a plat) are the basis for changes in the cadastral data. Text documents are the basis for rights (for example, ownership). Graphical documents only affect the number of different areas or the boundaries of these areas. Each document requires tests to prove its legality because the cadastre guarantees the correctness of the registered documents.
- Registration of deeds (for example, USA):
  The action of signing a contract or creating a plat forms the basis of civil rights. The rights are valid even if the documents are not registered. The registration only secures the priority against later documents. The registration grants no protection against existing but unregistered (and therefore unknown) documents. This requires a title search to find out the actual legal situation or the correct boundaries.

There are only minor differences between the systems in the way they store documents. The most important difference is that the registration of title grants the validity of the documents stored and, also, grants the validity of the registered rights. The registration of deeds only lists documents and the user has to check their validity. However, the checks themselves are similar. Both systems create a chain of owners for a specified piece of land and allow proving the validity of a right.

2.2 Cadastral data

A cadastre must hold a variety of data. These data are necessary for the different tasks the cadastre must solve. The most important data are (Navratil 1998):

- **Technical data:** A cadastre must provide technical data for three tasks:
  - *Positioning:* The storage of the boundaries of the areas provides a link to the real world and therefore provides the positional reference.
  - *Taxation:* Calculation of the land tax requires the size of the piece of land and data on the land use. The land use affects the productivity of land and, therefore, the value of land. Then this value is the base for the land tax.
  - *Planning:* Planning requires data on land use and constructions. Existing constructions sometimes prevent other constructions (playgrounds should not be next to highways). The current land use also provides evidence for the value of land and therefore allows an estimation of the construction costs.
- **Legal data:** The legal data splits into two parts. The first part is ownership data, the second part are encumbrances like mortgages. Legal data serves as guarantee for rights in a system with registration of title, whereas a system using registration of deeds treats it as additional information and does not claim completeness or correctness. Contrary to technical data, legal data is either wrong or right. A boundary, for example, may have an error band, which is not possible for legal data.
- **Additional data:** A cadastre may also hold data that is neither legal nor technical data. These data are additional data (e.g. postal address).

Documents may not be removed from the register even if they become invalid. A cadastre establishes a chain of owners and stores the restrictions of ownership. Documentation of the chain of owners requires permanent storage of all changes in the ownership situation. This requires permanent storage of the documents and therefore physically removing a document is prohibited.

The different types of data create a layer structure. Technical data affects small pieces of land with unique land use and soil. There must be some kind of positional reference for that area. Other data, like the address, will be correct for several such areas. Therefore, the types of data form three layers with different spatial resolution. The elements of each layer consist of one or more elements of the lower layer. Figure 1 provides an example. Layer 1 contains the technical data (like boundary, soil type, and land use). It consists of four separate areas. The higher layers also have defined boundaries because they are merges of areas from the technical layer. Layer 2 shows the additional data. It only consists of two different areas. Finally, the legal data forms layer 3. This layer stores the legal data like ownership and encumbrances.

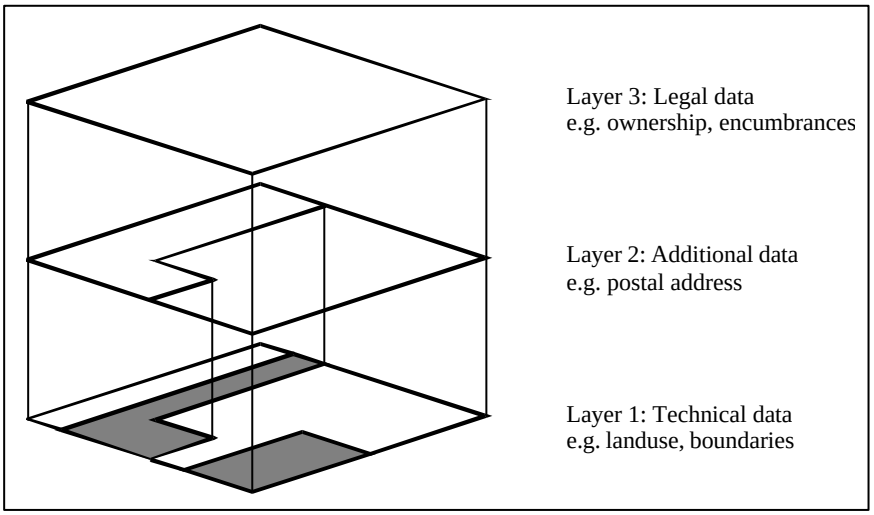

Figure 1: Layers of cadastral data

2.3 Subjects and objects in a cadastre

A cadastre deals with subjects, objects, and rights as connections between these two. A cadastre provides data on land. Most of the data (for example, ownership data) relates to people, too. Rights provide connections between subjects and objects (Twaroch and Muggenhuber 1997). A right is an idea and not a physical object. A right creates the connection by defining what the subject may do with the object. Figure 2 shows this connection. The subjects split into natural and juridical persons. A natural person is a human being. A juridical person is an organization (for example, the country itself or a church) or a company. The figure lists pieces of land and buildings as examples for objects.

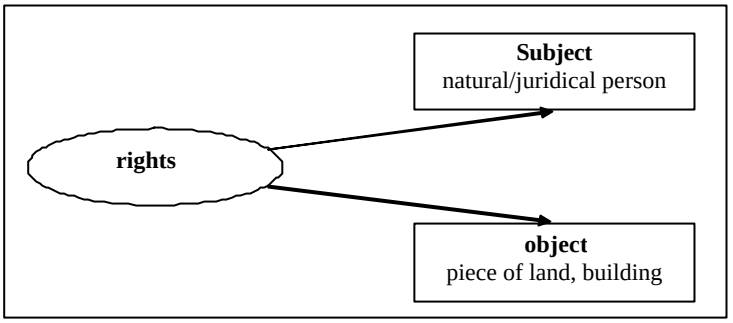

Figure 2: Subjects, objects, and rights

Identifiers represent subjects and objects. An identifier allows access to the subject or object. Subjects/objects have properties and relations to other subjects/objects. Identifiers also provide short references for these relations. The identifiers must be unambiguous within the cadastre. Therefore using the name as an identifier for a person would not be sufficient because there may be other persons with the same name. However old cadastral systems use the name as an identifier and have to deal with the problems arising from the ambiguities. The Austrian system, for example, tries to reduce these problems by adding the year of birth.

Documents require an identifier, too. A cadastre only holds the documents as evidence for rights or the transfer of rights and not the rights themselves. The documents must be accessible and therefore require identifiers.

2.4 Documents in a cadastre

Changes in cadastral data require a representation. Documents provide this representation because they exist in reality and are objects describing cadastral data. A bill of sale, for example, shows the old and the new owner and that reflects the change of ownerships. A computer representation must store the contents of the document. Typically cadastral documents contain two signatures (the owner of land and the beneficiary). In some cases other people apart from the owner of the land must agree to the document. The definition of a boundary, for example, requires the agreement of the owners of the neighbouring land. Their signatures show their compliance with the boundary resulting in the legal meaning of the boundary. The computer representation must reflect these aspects and must contain the signatures of these people, too.

The documents split into the following three categories:

- **Legal changes:** There are three types of legal changes:
  - **Transfer of rights:** Sales or inheritances transfer the right of ownership from one person to another.

  - **Establishment of rights:** If the owner of a parcel transfers a part of his right of ownership to another person (e.g. he allows that the other person walks across the parcel), he establishes a new right (the right of way in the example). This is a new right because it did not exist as a separate right although the owner has it implicitly as a part of the right of ownership.
  - **Deletion of rights:** Rights created as described above can be deleted if they are not necessary any more.
- **Changes of technical data:** The owner of land must subdivide the area he owns if he wants to sell a specific part of his land. This subdivision changes the number of areas stored in the cadastre. The action requires a document to reflect the change. Changes of land use also change the cadastral data and, therefore, require documents.
- **Changes of additional data:** Additional data may change, too. The postal address may change (e.g. if the name of the street changes). A cadastre must reflect that change. However, the old address must remain in the cadastral data because old documents may name that address. Therefore, the change requires a document.

3. User interaction

People want to use the cadastre if they want to secure or acquire rights on land. The pieces of land a cadastre deals with are called parcels. They are pieces of land with a determined boundary and a unique ownership situation. In case of a sale the person buying a parcel wants to inspect data (before the sale), change data (change the person registered as the owner), and (eventually) add new data (like a mortgage) Therefore, everybody must have the right to inspect data stored in a cadastre to check the legal situation, the position (in the real world) and the size of a parcel. Everybody with a valid document of sale must be able to register the document. However, the other actions must be restricted to the owners of the land. The owner must agree to the inscription of a mortgage, for example. Owners may also invalidate rights if they have the approval of the beneficiary of the right. Each of these actions changes the contents of the cadastre. Figure 3 shows these user actions.

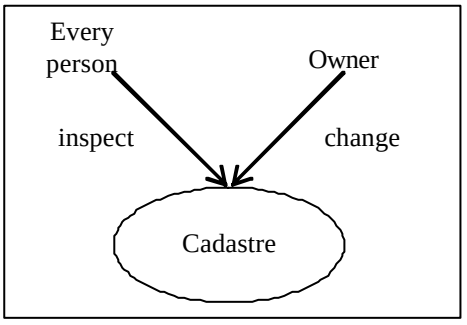

Figure 3: User actions

The actions must fulfil two prerequisites for the user's acceptance:

- **Simplicity:** The actions must not be complicated or time consuming for the user.
- **Reliability:** The result of the action must be valid. The confidence of the users will decrease if there are errors and invalid results.

The user must not be involved in the process of inscription to fulfil the simplicity. The user only starts the process and the cadastral organization performs it. Then the user does not need to know internal details and he does not need to spend a lot of time.

The reliability of the cadastral system depends on the completeness of the cadastral data. The user cannot rely on the cadastre if the data stored in the cadastre is incomplete because unregistered data could invalidate his document (for example, the person who wants to sell land is not the owner of the land). Then the user would question the suitability of the inscription and he would not register his documents, too. The reliability will decrease further. Therefore it is important that the user can rely on the cadastre because then he will register all changes to secure his own rights on the land. This becomes important especially in cases where the registration is not an obligation like inheritance in the Austrian cadastre.

4. Processes in a cadastre

A cadastre requires two processes: inscription and data retrieve. The inscription adds a new document to the document register. This adds a new right to the cadastre because rights are based on documents (see section 2.4) and should result in the publication of the data by making it accessible. Retrieving data returns information to the user.

A cadastre can be seen as a database and thus the processes are processes necessary for databases. A cadastre stores data on land. Documents represent the data and therefore a cadastre must store the documents. A database system provides the storage. A database must also provide methods to add new data, change data, delete data, and retrieve data. Adding documents to the cadastre provides adding, changing, and deleting data. Therefore the methods discussed in section 3 are database processes. Figure 4 shows what the system looks like.

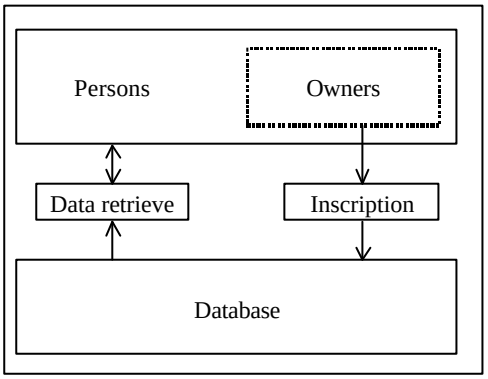

Figure 4: Processes in a cadastre (Navratil 1998)

4.1 Inscription

The inscription splits into three categories with different effects:

- **Transfer or creation of a right:** A document is an evidence for a right. Therefore, adding a document to the document register changes an existing right or creates a new right.
- **Deletion of a right:** The deletion of a right requires the approval of the beneficiary of the right (or a judgement). A document is necessary to prove that prerequisite. Therefore adding a document of deletion to the document register deletes a right. Then the affected right gets invalid.
- **Changing data:** One of the most important changes in the data of a cadastre is the subdivision or merge of land. A cadastre must create or invalidate land data sets if a

cadastre stores a separate data set for each piece of land. A cadastre must store the document showing the change (for example, a plat) to publicize the change.

The inscription itself consists of several steps (see Figure 5). Documents may pass several checks before the changes take place in the database. The process consists of the following steps:

- **Arrival:** The user files for inscription of the document.
- **Formal checks:** The operator of the cadastre checks the completeness of the data. These checks should be as short as possible to minimize the time the user has to wait and as detailed as necessary to detect a high percentage of errors at that early point.
- **Registration:** The operator adds the document to the document register. The register requires an identifier for the document. A simple solution for such an identifier would be continuous numbering.
  After the registration the public 'knows' the document. Everybody inspecting the cadastre will get data on the document. However it is necessary to place a mark for that document, so the public can also see that the document has not yet passed the checks for validity. This mark will be removed when the database changes took place.
- **Detailed tests:** The operator checks the validity of the document in detail. If the document fails the checks, the document may be
  - corrected by the author and again tested, or
  - removed from the register.
- **Database changes:** Some documents affect parts of the database and require changes. The deletion of a right, for example, sets the status of the affected right from 'valid' to 'deleted'. However, it does not delete the document completely.

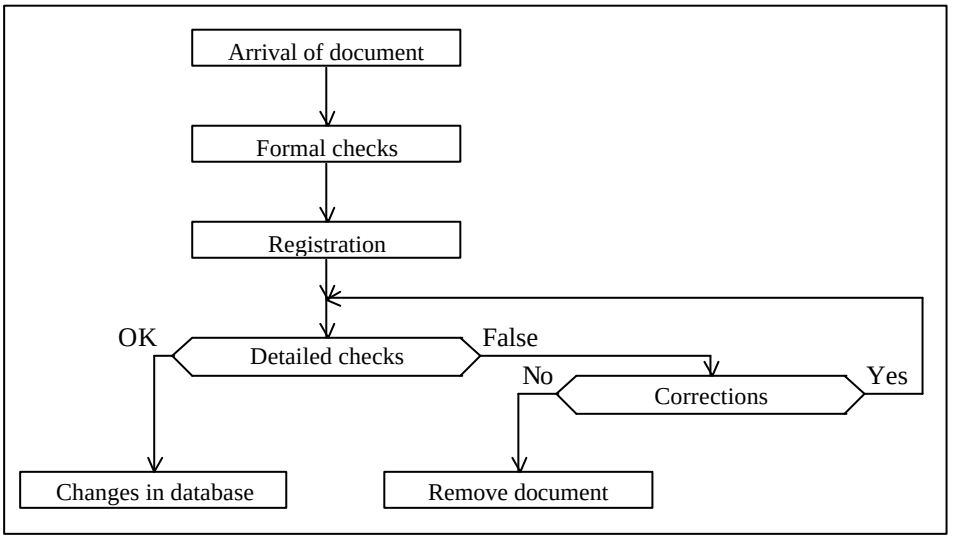

Figure 5: The process 'inscription' (Navratil 1998)

4.2 Data request

Data requests may be limited due to the law. In Austria reading all data concerning one person would contradict the data protection law. Therefore, using the name of the owner as a search criterion is restricted to people who need this possibility (e.g. courts, tax authority, and credit institutes). Similar situations may occur in other cadastral systems, too. Therefore, the process

must first test whether the request is allowed or not. The result of the request is only calculated if the request is allowed (see Figure 6).

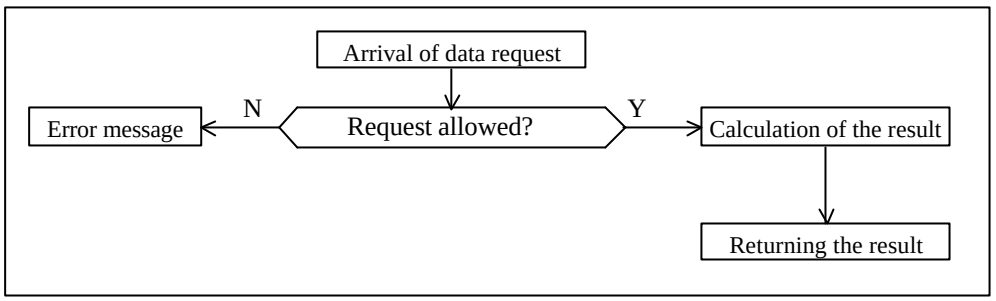

Figure 6: The process ‘data request’ (Navratil 1998)

The request must allow different keys. It is not granted that there is a single key suitable for all requests (for example, the identifier of the parcel). Therefore several possibilities must be provided. Some examples for keys are:

- Identifier of the parcel
- Postal address of the parcel
- Name of a person owning land

5. Formalization of the processes

The formalization starts with a database and then expands to a model of a cadastre. The database forms the core of the cadastre. Section 5.1 describes a simple database system. Section 5.2 then introduces the data types necessary for a cadastre and section 5.3 implements a general model of a cadastre.

5.1 Database system

A database stores data with data sets representing parts of the real world. The data sets are objects if using an object-oriented programming language for formalization. The data sets require unique identifiers to access the data. Medak defines the principal functionality of a database as follows (Medak 2001):

```
class Eq i => IDs i where
      nextID :: i -> i
      sameID :: i -> i -> Bool
      sameID i j = i == j

class Objects o where
      putID :: ID -> o -> o
      getID :: o -> ID
      isID  :: ID -> o -> Bool
      isID  i o = sameID i (getID o)

class Objects t => Databases w t where
      createObj  :: t -> w t -> w t
      getObj     :: ID -> w t -> t
      destroyObj :: ID -> w t -> w t
      getLastID  :: w t -> ID
      perform    :: (t -> t) -> ID -> w t -> w t
      thatAre    :: (t -> Bool) -> w t -> [ID]
      exists     :: ID -> w t -> Bool
```

The programming language used here is Haskell, a functional programming language (Thompson 1996; Hudak, Peterson et al. 1997; Bird 1998). The advantages of the language are mathematically correct definitions, the support of object orientation, and the short source code. Contrary to other programming languages the methods of functional programming languages (functions) always have a result that is returned.

Identifiers have two methods: `nextID` and `sameID`. The method `nextID` creates a new identifier from a specified identifier. This may be done, for example, by continuous numbering. The method `sameID` compares two identifiers and returns `True` if they are equal. This method is defined using an axiom.

Objects have three methods: `putID`, `getID`, and `isID`. The method `putID` adds an identifier to an object and the method `getID` retrieves the identifier. Finally, `isID` tests if the identifier of a given object is equal to a specified identifier. The axiom retrieves the identifier of the object using the method `getID` and then compares the resulting identifier to the specified identifier by using the method `sameID`.

Databases require a number of functions. Data shall be added to, retrieved from, or deleted from the database. The last two functions use the identifier to determine the data set. It should also be possible to get all data sets that fulfil a specified condition (`thatAre`) and to perform an action on all data sets (`perform`). There is no function to change a data set because changing data can be seen as deletion of an old data set and creation of a new one (this also allows tracking the changes in the database).

5.2 Data types for a cadastral system

The data types described in this section provide a simple solution for a cadastre. See (Navratil 1998) for more detailed data types for the Austrian cadastre.

The identifier may be of type integer for a simple demonstration. The range of integer numbers is sufficient for testing purposes because test data sets typically contain only a few data sets. Practical realizations require better solutions like, for example, adding the year of inscription to limit the range of necessary identifiers. The data type for the identifier and the corresponding instance of the class `IDs` has the following code:

```
type ID = Int
instance IDs ID where
      nextID i = i + 1
```

The cadastre has to deal with documents. The model requires a general data type and realizations for each type of document. The number of specifications is limited because the number of document types is limited. The code provides a realization for a sale. The data type `Sale` holds the identifiers of the seller and buyer (both of type `PID`), the identifier of the piece of land (`ID`), and the date of signing. The identifiers of the people are defined as strings.

```
type PID  = String
data Date = Date Int Int Int
data Sale = Sale PID PID ID Date
data Document d = D ID d
```

A cadastre is a list of documents. Documents provide data on each change of cadastral data. A document can also provide a starting point. An owner, for example, receives his ownership from the previous owner. The previous owner received his ownership from his predecessor and so on. Finally, at the time of the first registration there is an owner, who has no predecessor. Typically that owner is either a nation or a monarch. Therefore a gift from

somebody to that first owner provides a starting point for the cadastral data concerning ownership of land. A data type for a cadastre therefore looks like the following:

```
data Cadastre t = C ID [t]
```

The identifier used here is the identifier of a cadastre. Organization becomes simpler if the country is split into smaller parts with separate cadastres (based on the same principles, however). Each of these cadastres then requires an identifier.

5.3 General implementation of a cadastre

First of all a cadastre requires an implementation of the database for the data type `Cadastre t`. The method `createObj` takes a new entry `t` and an existing database, creates a new identifier for the entry (`nextID`) and adds it to the database. The function `getObj` reads the entry, `destroyObj` removes an entry, and `exists` checks if there is an entry. All three functions use an identifier to address the entry. The method `perform` applies a function to an entry specified by its identifier. Finally `thatAre` filters the database and returns all entries that fulfil a condition given by the function `f`.

```
instance Databases Cadastre t where
      createObj  t (C i s) = C i' (t':s) where
            i' = nextID i
            t' = putID i' t
      getObj i (C j os) = head (filter ((==i).getID) os)
      destroyObj i (C j os) = C j os' where
            os' = filter ((/= i).getID) os
      getLastID (C i s) = i
      perform f i (C i s) = C i [if isID i x then f x else x | x <- s]
      thatAre f (C i s) = map getID [x | x <- s, f x]
      exists I (C j s) = elem i (map getID s)
```

The inscription requires a class with the following methods: The method `inscribe` performs the steps defined in section 4.1. It consists of `formalTest`, `rc`, `test`, and `dbChanges`. The method `formalTest` applies the formal tests to the document and returns `False`, if the document fails. The local function `rc` registers the document using the method register. After the registration the document must be read from the result of `rc` using the local function `nd` because the data set of the document changes during the registration (the document gets an identifier). The method `test` then applies the detailed tests. Finally, the changes in the database take place (using the method `dbChanges`). The simple implementation of `inscribe` that is shown here does not allow corrections as shown in Figure 5 because this would require user interaction. User interaction requires complex code in functional programming languages and this would complicate the solution significantly. Therefore, corrections are not allowed in the implementation presented here.

```
class (Databases c a, ListFuncs a) => Cadastres c a where
      inscribe   :: c a -> a -> c a

      data2Doc   :: c a -> a -> a
      formalTest :: c a -> a -> Bool
      register   :: c a -> a -> c a
      test       :: c a -> a -> Bool
      dbChanges  :: c a -> a -> c a

      inscribe c a = if (formalTest c a) && (test rc nd)
                        then dbChanges rc nd else c
            where rc = register c a
                  nd = data2Doc rc a
```

```
class ListFuncs a where
      sameData :: a -> a -> Bool
```

Data requests need different methods because the result of the requests may differ. The result of a request may be a document, a list of documents concerning a specified area of land or simply an identifier. It is possible to model this in a single method. However, the use of that method would be complicated because it would require an explicit specification of the type of result. Methods for data requests could look like the following:

```
class Cadastres c d => Requests c d where
      owner     :: c d -> ID -> d
      documents :: c d -> ID -> [d]
```

The method `owner` takes a cadastre and the identifier of a piece of land and returns the document containing the last ownership change concerning the specified area. The method `documents` takes the same parameters and returns the list of all documents concerning the specified piece of land.

6. Implementation of two different cadastres

The general methods discussed so far require an implementation to prove their validity. Section 6.1 shows a coarse implementation of the Austrian cadastre to prove that the methods satisfy the needs of a registration of title. Section 6.2 then does the same for the cadastre of the USA (registration of deeds).

6.1 Simple implementation of the Austrian cadastre (registration of title)

A simple implementation shall grant that the processes satisfy the needs of the Austrian cadastre. The code only implements sales as an example for a document type. It shows, however, what the implementation of the Austrian cadastre would look like:

```
instance Cadastres Cadastre (Document Sale) where
      data2Doc (C id ds) dd = getDoc ds dd
            where getDoc [] dd = error “Document not found“
                  getDoc (d:ds) dd = if sameData d dd then d
                                          else getDoc ds dd
      formalTests c (D id (Sale s b p dt)) =
            (id < 0) && (s /= “”) && (b /= “”) &&
            (p > 0) && (dt /= (Date 0 0 0))
      register (C id ds@(d1:dd)) nd =
                        C id ((putID (inc (getID d1)) nd):ds)
            where inc a = a + 1
      test c (D i (Sale s b p dt)) =
                        (isOwner c s p) && (mayDoContract p)
            where isOwner c s p = True
                  mayDoContract p = True
      dbChanges c d = c
```

The formal tests must prove the completeness of the data. The method `formalTests` only checks the existence of the contents in the document data set. The identifier must be a negative number, which marks the identifier as not existing. This must be true because the registration sets the identifier of the document and the formal tests are prior to the registration.

The registration must add the document to the document register and set the document identifier. The method `register` reads the identifier of the last document registered and

increments it. The result is the identifier for the new document. This simple solution implements continuous numbering for the identifiers.

The detailed tests must prove the correctness of the document. A sale must fulfil two prerequisites to be valid:

- The seller must be the owner of the land. The local function `isOwner` provides that check. The code here sets the result to `True`. A complete model must derive the result from the cadastral database and compare the results to the seller. A solution for this problem can be found in (Frank 1996).
- The seller must be allowed to sign contracts. The Austrian law defines that a contract is only valid if all people who are signing the contract have full contractual capacity. Minors, for example, have limited contractual capacity and therefore contracts signed by minors are invalid. The model assumes that the people signing the contract have full contractual capacity because the check would require a more detailed model of people.

The function for the database changes updates the organizational data sets. There are no changes in the database because all of them take place in the organizational data sets.

6.2 Simple implementation of the cadastre of the USA (registration of deeds)

The instance for the class `Cadastres` looks almost like the instance for the Austrian cadastre. The only differences are that the American cadastre does not check the validity of the documents after registration and there are no database changes. Therefore the implementation for the methods `test` and `dbChange` looks like the following:

```
instance Cadastres Cadastre (Document Sale) where
      test _ _ = True
      dbChanges c _ = c
```

The instance for the class `Requests` has a different implementation for the method `owner`. Ownership of land in the USA is based upon the validity of the chain of owners from the beginning to the latest sale. The cadastre of the USA must prove that validity. This is done by comparing the buyer of a sale with the seller of the following sale. The document list is valid if the results of the comparisons are all `True`. The result then is the buyer in the latest sale. Otherwise the result is undefined because the chain is not correct and therefore at least one invalid document is registered, or one document is missing. The model creates an error in this case.

```
instance Requests Cadastre (Document Sale) where
      owner c pc = if foldl (&&) True
                        (map selBuyOk (pairLst (documents c pc)))
                   then head (documents c pc)
                   else error “Invalid document in document register”
            where pairLst (d:[]) = []
                  pairLst (d1:d2:ds) = (d1,d2):(pairLst (d2:ds))
                  selBuyOk (d1,d2) = seller d1 == buyer d2
```

7. Conclusions

A cadastre has two processes: inscription and retrieval. The inscription provides changes of cadastral data by inserting new documents. The process of retrieving data allows extracting data from a cadastre. As section 6 showed the implementation of the process depends on the

type of cadastre. A cadastre requires no process to remove rights because the documents registered to the cadastre may have effects like the invalidation of rights.

Inscription of data is a process that requires human operators. The process inserts a new document to a cadastre. The document must be stored within the database. A cadastre requires a collection of the documents even if the system is based upon computers because the system demands high security against computer failures. Only human operators can handle the collection of documents because computers cannot work with paper documents.

Computers can retrieve data if the cadastral data is stored in a form that the computer can access. The process of retrieving data is simple in the case of the registration of title. The registration of deeds, however, requires title search and computers cannot do this because there may be unregistered documents. However, computers can provide the process if they have access to all document data. This also includes retrieving data by using the Internet.

8. References

Bird, R. (1998). Introduction to Functional Programming Using Haskell. Hemel Hempstead, UK, Prentice Hall Europe.

Dale, P. F. and J. D. McLaughlin (1988). Land Information Management - an introduction with special reference to cadastral problems in third World countries. Oxford, Oxford University Press.

Frank, A. U. (1996). An object-oriented, formal approach to the design of cadastral systems. 7th Int. Symposium on Spatial Data Handling, SDH'96, Delft, The Netherlands, IGU.

Hammer, M. and J. Champy (1995). Business Reengineering: Die Radikalkur für das Unternehmen. Frankfurt/Main, Campus Verlag.

Hofmeister, H. and H. Auer (1992). Das moderne Grundbuch. Vienna, Bundesministerium f. Justiz.

Hudak, P., J. Peterson, et al. (1997). A Gentle Introduction to Haskell, Yale University.

Lego, K. (1968) Geschichte des österreichischen Grundkatasters. Vienna, Bundesamt f. Eich- und Vermessungswesen.

Medak, D. (2001). Lifestyles. Life and Motion of Socio-Economic Units. A. U. Frank, J. Raper and J.-P. Cheylan. London, Taylor & Francis: 140-153.

Navratil, G. (1998). An object-oriented approach to a model of a cadaster. Department of Geoinformation. Vienna, Technical University of Vienna.

Thompson, S. (1996). Haskell - The Craft of Functional Programming. Harlow, UK, Addison-Wesley.

Twaroch, C. and G. Muggenhuber (1997). Evolution of Land Registration and Cadastre - Case Study: Austria. Joint European Conference on geographical information.