To appear in
Proceedings of the 6th Enterprise
Distributed Objects Computing Conference
(EDOC'2002)

Model Driven Geo-Information System's Development

Javier Morales
International Institute for Geo-Information Science and Earth Observation (ITC)
Enschede, The Netherlands
jmorales@itc.nl

Luís Ferreira Pires
Centre for Telematics and Information Technology (CTIT), University of Twente
Enschede, The Netherlands
pires@cs.utwente.nl

Marten van Sinderen
Centre for Telematics and Information Technology (CTIT), University of Twente
Enschede, The Netherlands
sinderen@ctit.utwente.nl

Abstract

Continuous change of user requirements has become a constant for geo-information systems. Designing systems that can adapt to such changes requires an appropriate design methodology that supports abstraction, modularity and other mechanisms to capture the essence of the system and help controlling complexity. An important factor in the conceptual modelling of information systems is to describe structural and behavioural aspects of these systems. Traditionally, however, these aspects of geo-information systems have been modelled separately, and little attention has been given to the behavioural part of such systems. Numerous techniques for the conceptual modelling of systems have emerged in the last years. We build upon these techniques, combining behavioural modelling with concepts from information modelling and formal system specifications, to define a methodology that enables the modelling of both behavioural and informational aspects of geo-information systems. The methodology aims at supporting the design process of the system, and also helps enforcing that geo-information systems are reliable, maintainable and compliant with the changing requirements.

1. Introduction

Geographic data has become widely available due to advances in data transmission technologies and efforts made by standardisation bodies to define proper transfer formats. However, the usefulness of geographic data can only be measured from the point of view of the users who can satisfactorily exploit this data to perform their work and consequently achieve planned goals. This satisfaction depends not only on the availability of the data, but also on its suitability to the users' particular application domains. Through the incorporation of processes, operations and applications that generate more specialised information services for specific user communities, highly suitable value added data could be obtained.

The reliability of the information services delivered by any geo-information provision system determines its success in the market. In order to fulfil stringent reliability requirements, processes, data, operations and applications have to be put together in a service chain. A static implementation of this service chain would suffice if the requirements remained constant during the whole life-cycle of the product, which unfortunately is not the case. Requirements do not remain static, but they keep on changing, mainly because users want to influence the products in many different ways, competition has intensified, and new technology offers a bunch of opportunities. All these circumstances make change a constant factor in geo-information systems. The design of geo-information provision systems that cope with dynamically changing requirements can become rather complex. Static pre-defined ever lasting services do not fulfil the dynamic requirements of the service users, so we have to define new dynamic and adaptive architectures for these services. This can be achieved, for example, by identifying core elements that may be used in multiple combinations to deliver different services.

The development of these dynamic and adaptable systems requires methods to analyse, model, design and redesign the system as a set of engineering artefacts. With the help of these methods, knowledge about the system can be captured in conceptual models that represent aspects of the system such as structure, behaviour and functionality. Traditionally, methods for geo-information system design put emphasis on static aspects of the system, such as the data and its structure. System behaviour is often neglected in most design methodologies for geo-information systems.

In this paper we present a novel approach to the design of geo-information systems. We have combined behavioural modelling techniques with abstract data types and formalisms, to define a methodology in which the model-

ling of both structural and behavioural aspects of geo-information systems plays a central role. The methodology focuses on the definition of adaptive architectures. We start by identifying and describing core elements or system components [3]. Each element has a definition that can be used to make assertions about system properties. Then, the logic of the interactions between elements is captured using behaviour blocks [4]. Pre and post conditions are used to check inputs and outputs of the behaviour blocks. Finally, functional specifications are obtained by composing the defined elements such that their composition satisfies the given sets of requirements.

The remaining part of this paper is organised as follows: section 2 discusses the business environment and requirements of geo-information systems, section 3 describes the general methodology to model geo-information systems, section 4 introduces the case study that is used to illustrate our approach, section 5 presents the design elements used to define core elements and their interactions, and section 6 presents the procedure to combine elements into a service chain represented by functional specifications.

2. Geo-information systems

This section provides a description of our application domain, namely geo-information. We introduce the relevant aspects of this application domain, including system requirements and the environment that geo-information systems have to deal with.

Geo-information systems are information systems that handle information about geographical areas and their natural, social and artificial elements. The change of emphasis in geo-information production from just data towards specific information services forces a change in the design of geo-information systems. Diverse and changing products and services form a set of dynamic requirements to geo-information systems, which can only be effective if these systems are developed to be adaptable, as opposed to static systems that are re-engineered every time the requirements change.

Figure 1 shows the environment of geo-information systems. External factors like technology, competition, policies, etc., are the forces that drive the requirements and define the responsibilities (functionality) of the system at any point in time. Continuous change of requirements [$R1..n$] through time affects the system and subsequently impacts its behaviour and responsiveness. Possible outputs in terms of Products [P] or Services [S] are defined by the inputs [R] to the system. Inputs varying over time mean that the system has to adapt continuously in order to respond satisfactorily. We believe that if geo-information systems are to remain accepted as part of large scale business applications, the use of suitable design methodologies is crucial to cope with this challenge.

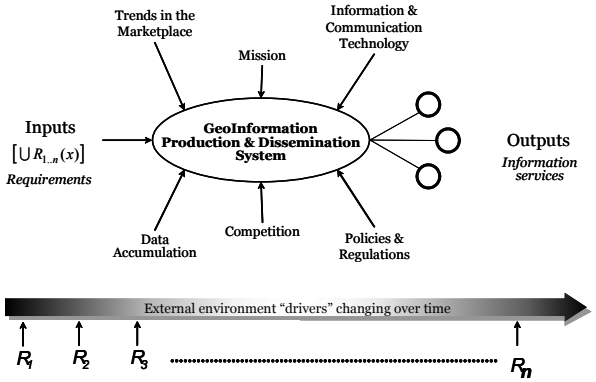

Figure 1. Geoinformation provision environment and the drivers of change

When studying existing geo-information systems, we observe that most of them have been designed using data-oriented design methods and only a few use some basic forms of object-orientation. Their architectures are suitable to acquire and store geographic data that can be retrieved at will and facilitate the production of certain maps. In contrast, these architectures have not been designed to deliver diverse products and services. Traditional geo-information systems offer interesting mechanisms for providing access to geo-information data produced by these systems. These mechanisms are altogether known as the *Geospatial Data Infrastructure* (GDI). A GDI is defined as the collection of technologies, policies and institutional arrangements that facilitate the availability of and access to spatial data. The GDI provides a basis for spatial data discovery, evaluation, and application for users and providers within all levels of government, the commercial sector, the non-profit sector, academia and by citizens in general [2].

We want to take advantage of the existence, understanding and maturity of GDI as an infrastructure for data sharing, and build upon it a system for the provision of a wider range of quality value-added information services as demanded by the geo-information market. We term this system a Geo-information Service Infrastructure (GSI). A GSI helps its users obtaining specialised information products and services by exploiting an infrastructure of interconnected data nodes (data repositories), data brokers, service providers, service brokers and clients. The benefits of a GSI come from the set of artefacts located along the distributed nodes. These artefacts have economic value and can be assembled together to perform operations within the infrastructure, resulting in specialised artefacts that have equal or higher value than the original artefacts. A service is created by integrating these artefacts and generating functionality that satisfies a par-

ticular set of requirements. Composing connectable artefacts in a flexible way to generate functionality is an appealing approach to cope with the dynamic requirements of geo-information systems, but it imposes a great challenge on the design of these systems.

When we consider that geo-information systems have to form a GSI, as introduced above, many implications arise especially at the meta-model level. Enough mechanisms and concepts have been developed in the last years for the meta-modelling of geo-spatial data, aiming at facilitating the search and discovery of this data. The challenge, however, is the definition, storage, management and publishing of meta-data for more complex products and services. Interest in this area is just beginning to emerge and, consequently, mechanisms and criteria for that are still scarce. We believe that if geo-information systems are to form a GSI, meta-data has to be generated for their products and services. Textual and potentially ambiguous descriptions of functions will not suffice, but rather should leave room for formal definitions of the observable behaviour of the service and of the composition of elements that support it. Figure 2 illustrates that services can be generated by combining available components (data and processes). This service realisation can be specified in terms of a composition of the models of these components.

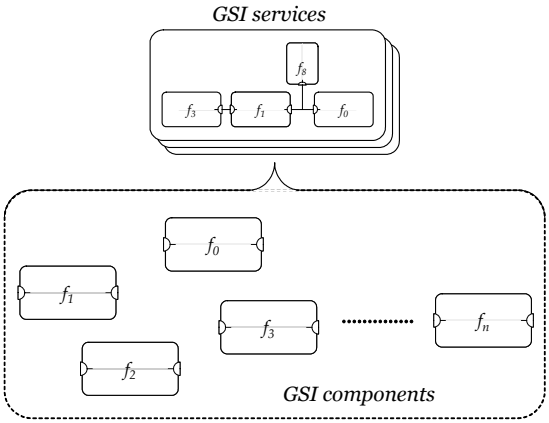

Figure 2. GSI system service realisation

The remaining sections of this paper present a design methodology to support the development of sufficiently conformant, reliable and maintainable GSI systems.

3. Our development methodology

The development trajectory of an information system should start with a conceptual modelling phase, during which models that describe structural and behavioural aspects of the system are produced [6]. Traditionally, in the design of geo-information systems these two aspects have not received the same amount of attention, since little or no focus has been put on the behavioural aspects of these systems [7]. Numerous techniques for conceptual modelling have been developed in the last years like, for example, UML [10] and IDEF [9]. Our group at the University of Twente has developed a methodology for the design of distributed systems that stresses conceptual modelling of behaviour [4]. This methodology is supported by a consistent set of design concepts, and by a language that allows designers to represent instances of these concepts in designs. The language has been formalised [11] in order to enforce precision and to allow comparison of designs. In particular, For example, the formalisation makes it possible to establish the mutual correctness of designs that are specified at different abstraction levels. AMBER [5] is a specialisation of this design language for business processes.

The design methodology considered in this paper combines abstract data types with the concepts for behaviour modelling supported by AMBER. This methodology aims at defining both structural and behavioural aspects of geo-information systems, so that they can be developed to form a GSI.

Our methodology focuses on the definition of the services provided by a GSI. A service is delivered by combining functions that are offered by independent components available at GSI participating nodes. A service has to be formally defined before it can be properly implemented. The internal structure of the service (i.e., the service realisation) describes how different components interact to generate a desired service. To create a service realisation, proper understanding of the available components is required. For this purpose, it is necessary to pay attention to the identification and the definition of the (functions offered by) the components. We have identified three dimensions to describe the different aspects of a GSI system, facilitating the identification and arrangement of components for the purpose of constructing a service realisation (see Figure 3):

- the *data dimension* represents the information that is used, manipulated and/or generated by the GSI system. Related to this dimension one can identify *data components*. Data components are responsible for the storage and provision of geo-information, represented as data sets and proper documentation (meta-data);
- the *process dimension* represents the geo-processing capabilities of the GSI system. Such capabilities are provided by *process components*. Process components are used to generate or modify data sets. Modification of data sets can be accomplished through interaction with data components, in which case the process component can be considered independent of the data sets;

- the *control dimension* represents the conditions and constraints that govern the interactions between components.

The origin of the reference system determined by these dimensions in Figure 3 represents the starting point of a processing flow that is activated by a design problem. Design problems are the various inputs (requests) to the GSI system that have to be converted into products and or services. The path taken through the cube shown in Figure 3 corresponds to the functional specification of a service that satisfies a specific design problem by making use of a proper combination of components. The execution of a predefined service definition generates a desired product or service.

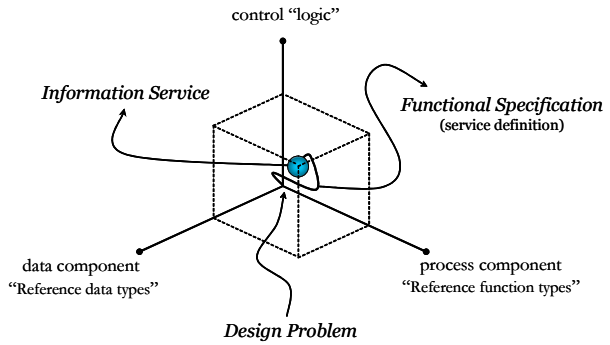

Figure 3. Modelling dimensions

In order to properly describe service definitions our structuring approach focuses on two main perspectives:

- the *external perspective*, which captures the responsibilities of the system defined by the relations between the system and its environment;
- the *internal perspective*, which describes the system components and their possible interactions that are necessary to achieve the desired service.

The internal perspective actually covers two related concerns, namely the set of components and how they interact to provide the service. Therefore, in order to obtain the internal perspective initially one is forced to consider each individual potential component separately. This separate consideration is essential when determining the suitability of a component for its participation in a service realisation.

Figure 4 shows how the external perspective, the set of components, the internal perspective and the GSI systems influence each other.

The external perspective aims at identifying and explicitly delimiting the scope of the system under development and helps determining the objectives of the development process [1]. Initially this definition of scope and objectives does not necessarily have to be as detailed as a complete service description. This step can be performed in such a way that more detailed specifications are added later in the development process. The system scope can be initially defined, for example, as short statements characterising the products and services to be delivered by the system to its users.

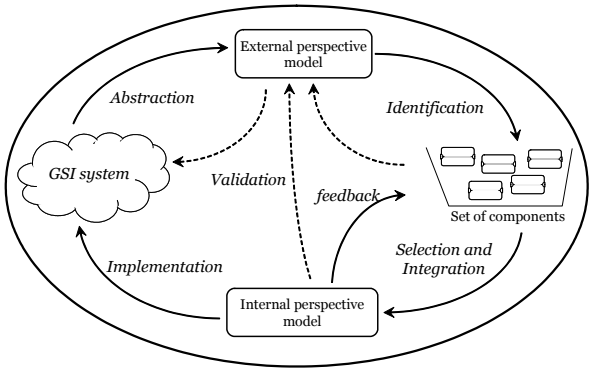

Figure 4. Modelling steps

The internal perspective aims at describing the internal system structure in terms of a composition of components that interact to perform the system functions. This arrangement of components determines the actual system behaviour and shows how the interacting components work together to achieve the system functionality. Many different alternative compositions are allowed that implement the same functionality in a different way.

We need to be able to focus on the behaviour of each component that can be used to compose a service. Each component requires a mechanism by which it can be accessed, so that it can be connected to other components. The internal structure of each component has to be modelled and properly described as well. As a result, a library of components (data definitions and process definitions) becomes available for the designers to use and generate service realisations (Figure 5).

In order to acquire the necessary information for the generation of the reference library and the service realisation, one needs to apply some structuring. Figure 5 shows structuring principles for the GSI system. Figure 5 (top-left) shows how the system has to be analysed as a collection of components that are in principle independent from each other. Using categorisation, those components are arranged according to their correspondence to the three dimensions shown in Figure 3. Each component is described in order to be able to assess its contribution to different system services. For that purpose, formal definitions of the components are produced and stored in the reference library. The design concepts to generate these definitions are explained in section 5. At this point each component has its own definition and can be used for different service definitions that deliver results to the clients.

To define these service definitions we use the responsibilities or requirements that are being continuously identified from the geo-information market (Figure 5, top-right) and that have to be supported by the underlying system. These responsibilities (sets of requirements) are used as design problems.

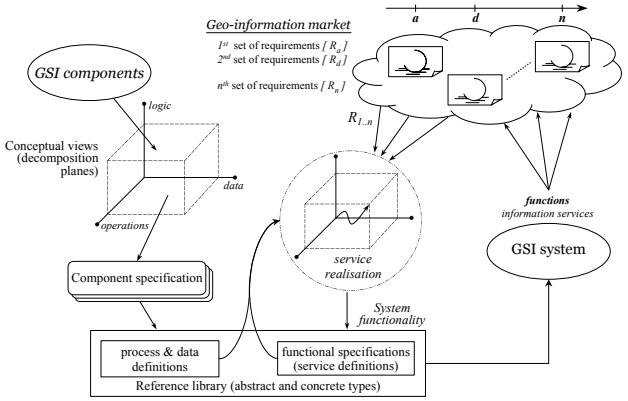

Figure 5. Development methodology

When generating a service realisation, we define how components interact with each other in a meaningful way to provide the required service. During this step, a path through the cube shown in Figure 3 is defined and determines a behaviour definition for component assembly. Each behaviour definition is a sequential arrangement represented by the relations and interactions between the participating components, including conditions and alternatives. The logic that governs these relations is based on pre and post conditions that are evaluated to identify the correct sequences. Behaviour definitions that are suitable to fulfil particular requirements are stored in the reference library. Behaviour definitions can also be reused as components in combination with additional basic components to define specifications that provide more complex functions. All these stored specifications can be enacted with a workflow management system or a similar tool to physically generate a desired service. Section 6 shows the design concepts used for the creation of full service definitions.

4. The TIC-service case study

This section describes informally the *tax identification and calculation* (TIC) service case study that we use in this paper to illustrate the various concepts of our methodology. The TIC-service is a fictitious service based on actual legislation that defines the rules for the fare taxation system in agricultural areas in some countries. Tax payers are registered at the regional tax office. Every agricultural piece of land (field) has a tax payer associated to it, who can be the owner or the tenant of the land. Taxes are determined twice per year.

The taxation regulation states that the tax pertaining to a field should be calculated as a function of three main aspects: 1) crop types, 2) crop productivity and 3) crop delivery. The crop types determine the base tax percentages for a field. Crop productivity is a function of the crop types, planted area, and the amount of rainfall and average temperature during the crop's life span (approximately six months). Crop delivery depends on the accessibility of the fields, which implies distance to urban centres, transportation means, such as rivers and roads, and the condition of the available roads. Some extra factors may also be considered when determining the tax to be paid, such as, whether fields have one or multiple crop types distributed in different percentages according to the wish of the farmer, whether crops are harvested in seasonal bases after which new crops are introduced, whether the introduction of new crops influence their types and distribution, or whether fields are not cultivated in their totality or are not cultivated at all in some periods.

The tax office, however, does not hold all the data necessary for determining the taxes. It does not have the capabilities and it is not responsible for generating and maintaining this data. The tax office does not have the expertise required to implement and maintain the complete system to generate the necessary information and determine the tax values. This would be extremely expensive, especially if you consider the periodicity of the tax calculation (typically every six months in this example). Nonetheless, the tax office is responsible for the tax calculation. This requires an information service conceived on the principles of the GSI system to support the tax office. We call this the TIC-service and we use it to illustrate the main ideas of our methodology.

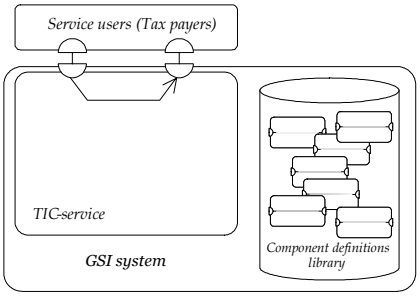

Figure 6. TIC-service external perspective

Figure 6 shows that tax payers interact with the GSI system to obtain some service. The service normally consists of a registration for taxation, which takes place when a user acquires some land for exploitation, and tax invoices that are delivered to the user on periodical basis.

This case study contains some typical characteristics and requirements of geo-information services, giving a good indication of the conditions in which these services have to operate. In this particular case study, some of the necessary data sets are collected and maintained by other private or governmental offices. Other sets of data have to be collected periodically and can be obtained by making use of functions (processes) delivered by other service providers. We abstract from the organisational details and focus on the service itself to get the external perspective of this particular case.

This case study contains some typical characteristics and requirements of geo-information services, giving a good indication of the conditions in which these services have to operate. In this particular case study, some of the necessary data is collected and maintained by other private or governmental offices. Other sets of data have to be collected periodically and can be obtained by making use of functions (processes) delivered by other service providers. We abstract form all the organizational detail and focus on the service itself to get the external perspective of this particular case.

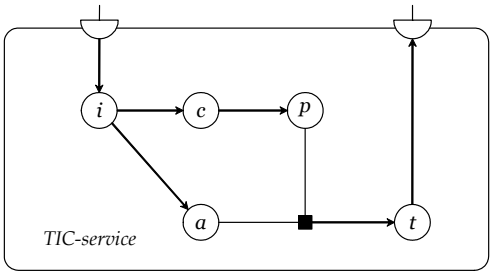

Figure 7. Tax calculation process

The tax calculation process for a specific tax payer is composed of the following steps (see Figure 7):

- *field identification* {*i*}, in which the geographical areas that correspond to the fields associated with the tax payer are identified;
- *crop classification* {*c*}, in which the crop types and areas planted in the fields are identified. This comprises the identification of the existing crops in every taxable field for the period being taxed;
- *determination of productivity factors* {*p*}, in which the average rainfall and temperature factors for the period being taxed are assigned to each field. This information is collected from weather stations;
- *determination of accessibility factors* {*a*}, in which a accessibility category is assigned to each field based on the existing surrounding transportation infrastructure and distance to crop distribution centres;
- *tax calculation* {*t*}, in which a tax form is generated with the corresponding amounts using all the information obtained for the fields. This tax form is notified to the tax payer afterwards.

From this description, we claim that the TIC-service has to be designed as a composition of components that are in principle independent from each other. We justify this claim by noticing, amongst others, that certain data (meteorological data in this case) does not have to be produced by the tax office, but rather obtained from the responsible provider, although some processing may be necessary to derive the values required for the tax calculation, and that periodic information about crops requires raw data (satellite images) from which crop information for the particular period can be obtained, this making use of a process that comply with the requirements. With the general description of the tax calculation process and the external perspective defined, we are set to identify and define individual components in section 5 and to generate a simplified TIC-service realisation definition in section 6. In both sections the necessary design concepts to generate component and service definitions are introduced.

5. GSI components specification

Each individual GSI component has to be specified before it can be included in a service realisation. This section discusses some design concepts used in our design methodology, namely for the definition of data and process components.

5.1 Data definitions

Data definitions are used to describe data elements, which are units of data structure similar to objects in object-oriented models. Therefore, we can describe data elements in terms of their classes or data types. Data elements can be defined as complex classes using aggregation or generalisation. In addition to the conventional predefined data types (integer, Boolean, chars, etc.) we introduce extra data types that are required for the definition of spatial data. By spatial data we mean geo-referenced real world objects that are represented in terms of geometric features with a relative location. These features represent diverse phenomena, about which relevant data is collected, maintained and disseminated. These features can be organised and classified into feature classes (like transportation, hydrography, relief, etc.) in accordance with the needs of applications with similar data requirements. Spatial data is therefore commonly arranged in *data sets* that can be composed of one or many feature classes. Data sets composed in this way form a feature catalogue [14]. Available feature catalogues are also modelled as data-components within the GSI system. The basic types for spatial data

types are given below using an abstract data type notation [8,12]:

Type constructors

	→	Geobj	*points / lines / regions*
Geobj	→	Theme	*theme*
Theme	→	Comp	*composite*

Geographic object types (*Geobj*) are used to represent single entities in the real world. A geographic object has two main characteristics: a *description-part* or set of attributes that define its nature, and a *geometry-part* that defines the way the object is to be represented. For example, a *city* type may be defined in its description-part by a schema containing its *name*, *population* and *foundation-date*, and in its geometry-part by a polygon in 2D space (region), which implies an area, a boundary, and the topological relations with the neighbouring geographic objects.

A theme type (*Theme*) represents a group of homogeneous geographic objects that share the same structure, in terms of their description-part and geometry-part. A theme definition must include a domain that defines the scope of the elements that can belong to the theme. The domain, which can be geometric or thematic, is mainly used to define pre and post conditions for the manipulation of theme elements. For example, a city theme can incorporate any real entity that complies with the predefined structure (any city), but this alone may still mean that entities or cities that are irrelevant for the purpose of the theme can be added to the theme.

Composite types (*Comp*) represent groups of heterogeneous geographic objects as collection of elements that still keep their individual structure, but represent a predefined snapshot of reality based on the users requirements. Composites are also used for the manipulation of raster-base information that includes satellite images, digital aerial photographs, etc. Final products generated by GSI components often contain data of the type composites.

For the case of the TIC-service we require these data types to include fields (legal parcels) that are taxable, and cultivated areas within the fields. These objects are captured as *Themes* of regions. Data about temperature is also represented as a *Theme*, but in this case as *points* containing the geographic coordinates of the spots that represent weather stations, and the additional attributes with relevant weather data required for this problem.

Figure 8 shows a class diagram that represents some of the data-elements required by the TIC-service. The diagram includes one main aggregated class `Taxform` that represents the data produced by the TIC-service. Each instance of this data-element encloses the details required by the tax office to generate the invoices to the various tax payers. The `TaxForm` contains two other data-elements `TaxPayer` and `Field` as components. The element `Field` itself contains a `Crop` element. The diagram specifies implicitly that a `TaxForm` may include multiple `Fields`, and that the `Field` element is defined as a spatial element of type *theme*. This implies that it can store the boundaries of the `Fields` that are used for the calculation of the tax for the referred tax payer. Figure 8 also shows three specialisations of the `TaxRate` class that represent the different components of the tax calculation.

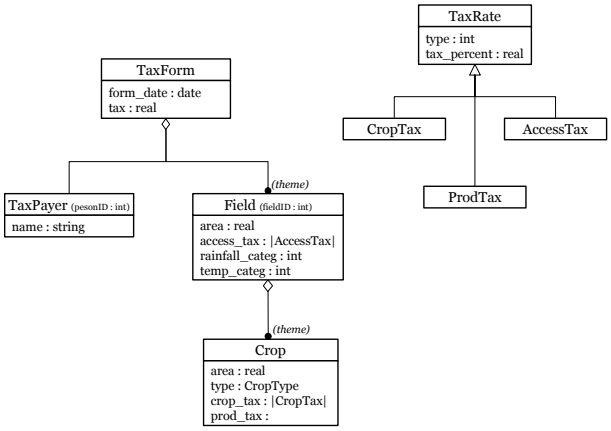

Figure 8. Data-elements diagram

data element **Field**
  components
    crop(ID : *int*) : Crop → *theme(regions)*
  attributes
    area : *real*
    access_tax : |AccessTax|
    rainfall_categ : *int*
    temp_categ : *int*
    num_cr : *int*
    location : *geobj(regions)*
  operations
    addcrop (ID : *int*)
  constraints
    crop(id).area +...+ crop(id+num_cr).area ≤ area
end;

List 1. Field definition

In addition to a data-elements diagram, it is necessary to produce a textual specification of the data-elements containing more details on the data-elements. This includes user defined data types, components, internal operations, as well as all the necessary attributes, together with their initialisation procedures if required.

List 1 shows the specification of the Field data-element, which, in addition to the attributes and components, also defines 1) a local operation *addcrop* that adds a crop region to the crop *theme* of the Field element; and 2) a constraint that assures that the total area occupied by Crops is not bigger than the total area of the Field itself.

This constraint can be extended to guarantee that each Crop region is inside the Field region.

List 2 shows the textual specification of the TaxForm data-element, with its components, attributes and local operations.

```
data element TaxForm
  components
    taxpayer (personID : code) : TaxPayer
    field (fieldID : int) : Field → theme(regions)
    weather_data : array(WeatherData)
  attributes
    form_date : date cons
    tax : real
  operations
    createForm (payer : code, c_date : date)
    addfield (fieldID : int)
end;
```

List 2. Taxform definition

Besides the data-elements above, there are other data-elements that require additional information to enable their discovery within the information infrastructure. An example is the meteorological data (rainfall and temperature). Since this data is required during the execution of the TIC-service, it is necessary to define a profile that can be used by a process to search and retrieve this data. This profile should contain details that properly describe the characteristics of the desired data.

```
data element WeatherData
  referential attributes
    thematic_content : array → [rainfall, temperature]
    spat_res : record(val : real, unit : str) → [1,"degrees"]
    temporal_res : (date) → [month]
    structure : type → [text]
    coverage : |Field|.polygon
  components
    ...
  attributes
    ...
  operations
    average(...)
    …
end;
```

List 3. WheatherData referential attributes

List 3 illustrates the set of so called *referential attributes* of a data-element that are used when a data description is necessary.

5.2. Process definitions

Processes are the core of a service definition; interactions between processes that manipulate data definitions convey the overall behaviour of the GSI system. A process component may be capable of performing a single activity or a collection of related activities. Process components are modelled using *behaviour blocks*, which are defined by composing actions, interactions and their causality relations [4,11].

An *action* represents a real world activity. The occurrence of an action represents that its corresponding activity has been successfully performed. Each action is uniquely identified and either occurs or does not occur. An action is an indivisible unit of activity (atomic) at a certain abstraction level. Action decomposition is needed whenever the activity represented by the action has to be defined in more detail. An action has attributes that represent the most relevant characteristics of the activity it represents, namely the result produced by the activity, the time moment when the result becomes available, and the location where the result can be found. Therefore an action has three attributes:

- the *information attribute* models the result of an action;
- the *location attribute* models the physical or logical location at which the result of the activity is made available;
- the *time attribute* models the time at which the established result becomes available.

In the case of GSI components, the information attribute of an action often refers to instances of a data definition.

An *interaction* is an abstract concept introduced to model an activity performed by two or more components in cooperation. An interaction can be considered as a refinement of an action, since it defines a specific distribution of the responsibilities for performing the interaction over multiple components. Interactions have the same properties and attributes as actions. However, when multiple components interact, each of them may define its own constraints on the possible values that can be established in the interaction attributes. An attribute value is established based on the negotiation constraints imposed by these components. Concerning these constraints, three basic forms of value establishment are distinguished:

- *value checking*, which represents that one component requires a specific value x to be established, while the other component requires a specific value y to be established. In order to allow the interaction to happen the following condition must hold: $x = y$;
- *value passing*, which represents that one component requires a specific value x to be established, while the other entity allows any value from a set of values Y to be established. In order to allow the interaction to happen the following condition must hold: $x \in Y$;
- *value generation*, which represents that one entity allows any value from a set of values X to be estab-

lished, while the other entity allows any value from a set of values Y to be established. In order to allow the interaction to happen the following condition must hold: $X \cap Y \neq \emptyset$.

GSI components interact with each other in order to deliver a service. Each component has interaction constraints that define under which conditions a component can participate in an interaction with other components. Therefore, two or more components can only be combined if the intersection of their constraints on each of the interaction attributes does not result in an empty set of values.

Causality relations are used to model the relationships between actions (interactions). A causality relation models the conditions under which a particular action becomes enabled, by defining how the occurrence of an action depends on the occurrences or non-occurrences of other actions. Figure 9 shows the basic causality relations: Figure 9(i) depicts *independence*, in which action *a* is independent of any other action and can occur at any point in time; Figure 9 (ii) shows *enabling*, in which the occurrence of action *b* enables the occurrence of action *a*; Figure 9 (iii) shows *disabling*, in which the occurrence of action *b* disables the occurrence of action *a*; Figure 9 (iv) shows *synchronisation*, in which both actions *a* and *b* occur simultaneously if they occur.

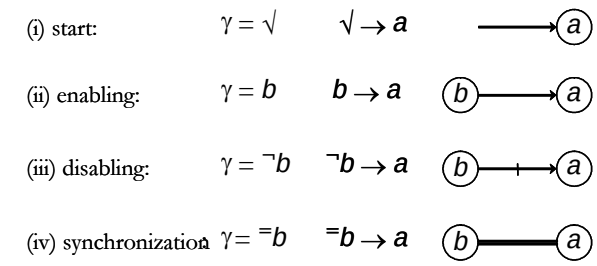

Figure 9. Causality relations

Composite causality conditions can be defined by combining the basic causality conditions using the (*and*) and (*or*) operators. More complex behaviours can also be modelled by combining basic relations with constraints on the enabling condition. Constraints can be based on the information, time and location attributes of the related actions and they are of three types:

- the *attribute value domain*, which defines the values that are allowed as a result of an action;
- the *attribute reference relation*, which defines how the attribute values of an action depends on the values of the attributes of the actions referred to in its causality condition;
- the *attribute causality relation*, which defines how the occurrence of an action depends on the values of the attributes of the actions referred to in its causality condition.

Behaviour blocks can be constructed by using actions, interactions and causality relations. Process components are modelled in terms of behaviour blocks, in a form that is suited for automated manipulation by, for example, a workflow management system.

In a behaviour block that specifies a process component, interactions are used to allow behaviour blocks to be composed together, representing the actual interactions between process components. An interaction contribution defines the constraints of a component to participate in interactions. These constraints restrict the establishment of the information, time and location values of the possible interactions between components, since interactions can only take place if all their constraints are satisfied.

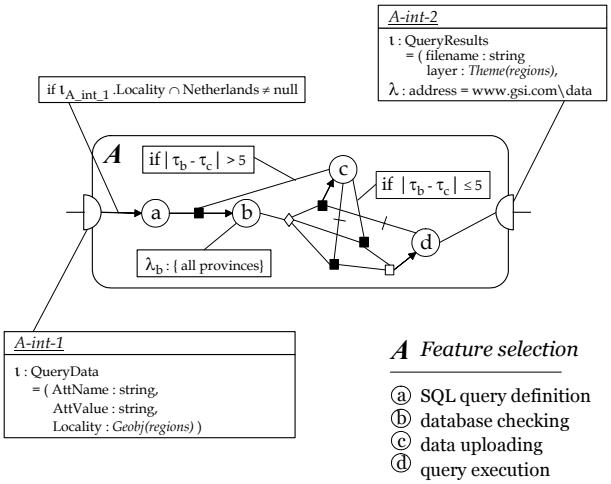

Figure 10. Behaviour block diagram

Figure 10 shows the behaviour block used to represent process component `FeatureSelection`. This component has two uniquely identified interactions *`A-int-1`*, *`A-int-2`*. When performing *`A-int-1`*, this component accepts three parameters: an attribute name `Attname`, an attribute value `Attvalue`, and a `Locality` or a polygon that represents the geographical area over which a query has to be performed. This process component delivers as output 1) a *theme* in interaction *`A-int-2`*, containing a set of features selected based on the attribute name and the attribute value, 2) the file name where the data is stored, and 3) the location where the file is available.

The `FeatureSelection` process extracts features from a dataset and can be used within the TIC-service for various tasks. A possible application of this process component is to select the parcels that a tax payer possesses in a municipality. For this purpose, the `FeatureSelection` component would receive values `AttName` = "Owner", `AttValue` = "PersonID" as selection criteria, and a poly-

gon object representing a municipality in `A-int-1`. Similarly, this process can determine from which weather stations data for the calculation of the crop productivity can be obtained. The behaviour block in Figure 10 also contains four internal actions that have to be performed in a particular way to generate the desired output in `A-int-2`. Alternatively, a behaviour block can be described using a textual notation, which we omit in this paper for the sake of conciseness. We refer to [4] for information on the textual notation to define behaviour blocks.

6. Service specification

After data and process components are specified, we can build a specification for the service realisation. This specification consists of four sections: attributes, data-elements, process-elements and behaviour. The attributes section defines all of those attributes required to qualify the service, the data-elements section includes all the data-elements from which instances of objects are required for use in the service, the process-elements section includes references to all the processes components that are required to realise the service, and the behaviour section defines the behaviours of the process components and how they interact.

service element **TIC-service**
  attributes
    …
  data elements
    taxform : (ID : *int*) : set(TaxForm)
    …
  process elements
    FeatureSelection {*A*}
    CropClassification {*C*}
    AccessFactorsdefinition {*B*}
    ProductivityFactorsdefinition {*P*}
    TaxCalculation {*T*}
    …
  behaviour
    A : { $\surd \rightarrow \underline{e}$
        $\underline{e} \rightarrow a$,
        $a \vee \{\, c \,[|\, \tau_b - \tau_c \,|\, > 5\,]\,\} \rightarrow b$,
        $(\, b \wedge \neg\, d\,) \rightarrow c$,
        $(\, b \wedge \neg\, c\,) \vee (\, b \wedge \{\, c \,[|\, \tau_d - \tau_c \,|\, \leq 5\,]\,\}\,) \rightarrow d$,
        $d \rightarrow \underline{i}$ };
    P : { $\surd \rightarrow \underline{i}$
        $(\, \underline{i} \wedge \neg\, o\,) \rightarrow$ m,
        $(\, \underline{i} \wedge \neg\, m\,) \rightarrow$ o,
        $o \rightarrow p$, $m \rightarrow e$,
        $m \rightarrow q$, $(\, e \wedge q\,) \vee\ p \rightarrow \underline{w}$ };
    B : { $\surd \rightarrow \underline{i}$
        $\underline{i} \rightarrow s$, $\underline{s} \rightarrow t$,
        $\underline{s} \rightarrow r$, $t \wedge r \rightarrow v$
        $v \rightarrow \underline{w}$ };
    …
    A, *C*, *B*, interact on $\underline{i}$
    B, *P*, *T*, interact on $\underline{w}$
end;

List 4. TIC-service realisation specification

List 4 gives the specification of the TIC-service realisation, omitting action and interaction attributes for the sake of conciseness. The data-elements section in this specification defines a set of objects of the type `TaxForm`. To simplify the behaviour section of this example, in the process-elements section we have renamed the process components with shorter names, such as, *A* for *FieldIdentification*, *C* for C*ropClassification*, etc.

The behaviour section of a service realisation specification consists of a composition of interacting components (sub-behaviours) related by means of their common interactions. The set of relationships between components in terms of their common interactions is called the *interaction structure*. The interaction structure of a behaviour defines for each interaction which components participate in the interaction. The textual representation of the interaction structure is a list of 'interact on' statements separated by semi-colons (see List 4). An '*interact on*' statement consists of a list of two or more behaviours, followed by the keyword 'interact on', followed by a list of one or more interactions. For example, the '*interact on*' statement *B*, *P*, *T*, interact on $\underline{w}$ defines that behaviours *B*, *P* and *T* participate in interaction $\underline{w}$, and no other behaviours participate in this interaction. Each interaction instance appears only once in the definition of the interaction structure of some behaviour.

Figure 11 gives an excerpt of the graphical representation of the behaviour section for the TIC-service. It shows that the interaction structure is graphically represented by linking the corresponding interaction contributions with solid lines.

The behaviour definition section of the TIC-service realisation shown in List 4 defines all the possible interactions between the participating components.

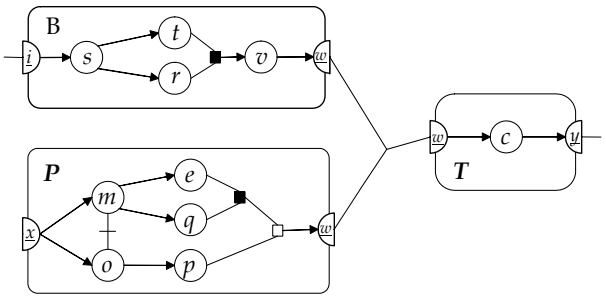

Figure 11. Service realisation diagram

All processes are instantiated from the beginning of the service execution context, but process *FieldSelection* {*A*}

is the starting point of the service realisation, since interaction *e* is enabled from the beginning (√) and can take place at any time, whenever the TIC-service users wish to request the service. Processes *FieldSelection* {*A*}, *CropClassification* {*C*} and Acc*essFactorsDefinition* {*B*} interact on $\underline{i}$, which can only occur when all involved components allow this interaction to happen. Similarly, interaction $\underline{w}$ can only occur when behaviour blocks *B*, *P* and *T* are ready to participate in it. Figure 11 shows that in component *B* action *v* must have occurred and, in component *P* action *p* or actions *e* and *q* must have occurred for the interaction $\underline{w}$ to happen ($v \wedge (p \vee (e \wedge q)) \rightarrow \underline{w}$). After interaction $\underline{w}$ has occurred, action *c* in component *T* can take place. The actions in behaviour blocks represent activities that are local to their corresponding components. We do not elaborate further on the descriptions of the actions of each component since a complete detailed specification of the TIC-service is beyond the scope of this paper.

Process-elements are in principle independent of each other, they may exist within one organization or they may be supplied by an external service provider. They are specified in a way such that they can be used in a service realisation in spite of their physical location or specific implementation. Another benefit of specifying process components as behaviour blocks is that these blocks do not only specify the local actions and the interactions between the components that form the service, but they can be used to automate the service by means of a workflow management tool. Because the GSI system is made of collaborating entities that provide geo-processing functions or spatial data or both, having a mechanism to describe, publish and manage formal definitions of components enables the integration of functions and, consequently, the provision of specialised services.

7. Conclusions

This paper reports on research that is being currently carried out in a cooperation between the Centre for Telematics and Information Technology of the University of Twente (CTIT/UT), and the International Institute for Geo Information Science and Earth Observation (ITC). Our aim in this research is to define a methodology for the development of flexible geo-information systems. This methodology builds upon our past experience with the use abstract behaviour modelling concepts to the engineering of business process and to the development of telematics systems.

This paper introduces the principles of what we call a Geo-Information Service Infrastructure (GSI), which should allows added-valued geo-information services to be delivered to the users of geo-information, as opposed to the raw data as it is often the case now-a-days. A GSI is expected to offer facilities for the integration of available service components to form tailored services, which is an approach that is well-suited to cope with the ever changing requirements of geo-information users.

In this paper we introduce the concepts and deliverables of our methodology, and illustrate them with a simple case study of a tax identification and calculation service. Although the case study is relatively simple, it is representative enough for the environment of a GSI. We have shown with the case study that our methodology is capable of delivering a service development platform that complies with the requirements of a GSI.

In our research we have observed that mechanisms are necessary to define, manage and publish meta-data on the behaviour of service components. This observation applies to a multitude of systems that need to provide services in a flexible way to their users, from which geo-information systems are just a special case. Our approach is novel to the area of geo-information systems, particularly to what concerns the use of models to represent and compose behaviours (processes).

Tool support is a crucial factor for the success of large scale development processes, such as the development of a GSI. We have some tools already available for editing specifications and performing some simple tests using simulation. Tools to automate validation of specifications are being currently developed. Some other tools could be developed to manage service repositories that support a GSI, for instance, for version control. Considering the example of a service component that is stored in a repository and is used in a service realisation, if this service component is modified, the consistency of the service realisation cannot be guaranteed and has to be assessed again. Tool support could (partly) automate this task.

References

[1] T. H. Davenport. *Process Innovation: Reengineering Work Through Information Technology*. Harvard Business School Press, Boston, Massachusetts, 1993.

[2] N. D. Douglas. Developing spatial data infrastructures: The SDI cookbook. Technical report, Global Spatial Data Infrastructure Organization, May 2001.

[3] D. F. D'Souza and A. C. Wills. *Objects, Components and Frameworks with UML: The Catalysis Approach*. The Addison-Wesley object technology series. Addison-Wesley, Reading, Massachusetts, 1999.

[4] L. Ferreira Pires. *Architectural Notes: a Framework for Distributed Systems Development*. PhD thesis, University of Twente, Enschede, The Netherlands, September 1994. Ph.D. Thesis, no. 94-01.

[5] H. M. Franken, M. K. de Weger, D. A. C. Quartel, and L. F. Ferreira Pires. On engineering support for business process modelling and redesign. In G. Doumeingts and J. Browne, editors, *Modelling Techniques for Business Process Re-engineering and Benchmarking*, chapter 10, pages 103–120. Chapman & Hall, London, UK, 1^{st} edition, 1997.

[6] I. Graham. *Object-Oriented Methods: Principles and Practice*. The Addison-Wesley object technology series. Addison-Wesley, Reading, Massachusetts, 3^{rd} edition, 2001.

[7] R. Groot and J. McLaughlin. *Geospatial Data Infrastructures: Concepts, cases and good practice*. Oxford University Press, New York, United Sates, 2000.

[8] R. Hartmut Güting and M. Schneider. Realm-based spatial data types: The ROSE algebra. *VLDB Journal*, 4(2): 243–286, 1995.

[9] R. J. Mayer, C. P. Menzel, M. K. Painter, P. S. deWitte, T. Blinn, and B. Perakath. Information integration for concurrent engineering (IICE): IDEF3 process description capture method report. Technical report, Knowledge Based Systems, Incorporated, September 1995.

[10] Object Management Group. Unified modeling language specification. Technical Report Version 1.3, Object Management Group, June 1999.

[11] D. Quartel. *Action Relations: Basic design concepts for behaviour modelling and refinement*. PhD thesis, University of Twente, Enschede, The Netherlands, February 1998. Ph.D. Thesis, no. 98-18.

[12] P. Rigaux, M. Scholl, and A. Voisard. *Spatial Databases: With Application to GIS*. The Addison-Wesley object technology series. Morgan Kaufman Publishers, San Francisco, California, 2001.

[13] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorenson. *Object-Oriented Modeling and Design*. Englewood Cliffs, Prentice Hall, New Jersey, 1^{st} edition, 1991.

[14] ISO. 19110 Geographic information - Feature cataloguing methodology. Technical report, ISO/TC 211, December 2001.