

Using the GEONAMICA[®] software environment for integrated dynamic spatial modelling

Jelle Hurkens^a, Bernhard Hahn^a and Hedwig van Delden^a

^a *Research Institute for Knowledge Systems (RIKS) BV, Maastricht, The Netherlands
(jelle@riks.nl)*

Abstract: GEONAMICA[®] is a framework developed by RIKS to support the development of integrated spatial decision support systems (ISDSS). This paper firstly presents an outline of the development process of an ISDSS, signalling potential obstacles and emphasising the need for cooperation between the involved parties. Secondly, it gives an explanation of the way in which a framework helps to reduce development costs and improve the quality of the final product, combined with a study of the requirements needed for the framework to be effective. Thirdly, the paper presents an overview of the framework, relating it to existing modelling paradigms. In the conclusion, we review the adherence of the framework to the requirements we set out and give a look into the future.

Keywords: modelling framework; model integration; modular model component; integrated assessment; DSS development

1. INTRODUCTION

Spatial planning processes are changing by the possibility to make better informed decisions on the basis of available spatial data and insight in spatial dynamics. This leads to an increase in the demand for Spatial Decision Support Systems (SDSS) [Densham, 1991]. However, the unfamiliarity of decision makers with SDSSs makes it hard to communicate the possibilities of such a system. Previous projects with similar objectives can help exemplify the benefits and elicit the kinds of questions or problems it can help answer, but the SDSS developer must make sure he can deliver what he promised with the often limited budgets that are available for such projects.

In this light, an application framework to support the development of a SDSS can prove a valuable advantage. A *framework* is a reusable, semi-complete (software) application that can be specialised to produce custom applications [Fayad, et al. 1999]. The primary benefits stem from two types of reuse: design reuse and implementation reuse. By delivering a useful set of patterns as a documented design, as well as a partial solution in the form of a skeleton application, a framework may save lots of costs for rediscovery and reinvention [Hahn & Engelen, 2000].

Besides reusability, *modularity*, *extensibility* and *inversion of control* [Fayad, et al. 1999] are benefits of an application framework that can reduce development costs. Moreover, in the domain of integrated SDSSs, these properties help to support a more advanced development process, in which a relatively simple prototype system is iteratively improved and expanded based on the users needs. Modularity eases the implementation of adaptations by localising their impact on the system. Extensibility allows a system to become more complex and specialised on the users needs without the need for a complete redesign. Inversion of control keeps the interaction procedures between components stable from one iteration in the development process to the next.

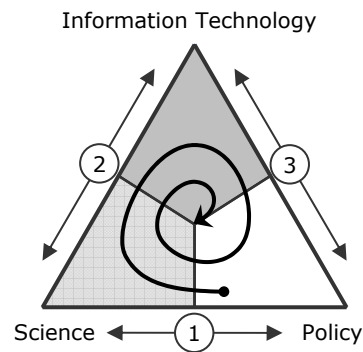
In the next part, we will elaborate on this iterative development process and the roles of the involved parties. From the insights in this process, we will derive requirements that a framework must meet in order for it to be a suitable base to build on. Section 2 gives an

explanation of the GEONAMICA® framework, which has been developed for this purpose, and relates it to the theoretical foundation it builds on. The last section shows how it meets the requirements we have set out and presents a look into the future.

1.1 Developing an Integrated Spatial Decision Support System (ISDSS)

The development of an ISDSS can best be described as an iterative process of communication and social learning amongst three involved parties – as depicted below. First, there are *policy makers*, the end-users of the system. They provide the policy context and define the problems, functions and usage of an ISDSS. Second, there are *scientists* responsible for the main model processes and choices of scale, resolution and level of detail. Third, there are *IT-specialists* who design the system architecture and carry out the software implementation of the models and user interface.

Note that for complex ISDSS projects, a fourth role is vital for success. The *DSS architect* has the main responsibility for integration, communication and management and assures the quality of the integrated model underlying the system. As a generalist, the architect bridges methodological and knowledge gaps between policy makers, scientists and IT-specialists – both between and within groups. In order to fulfil this role, the architect needs a solid and intuitive understanding of the application domain and the purpose of the system, as well as very good communication skills.



The interaction between the three groups involved is as important for the quality of the final product as the tasks carried out by each group individually. Policy makers and scientists can select policy-relevant research and models capable of answering the problems set out by the policy makers, translate policy options and external factors into model input and translate model output into policy-relevant indicators (1). Scientists and IT-specialists can work together to implement new models, link existing models and ensure consistency throughout the system (2). IT-specialists need to work with policy makers to set up a user interface that represents the relevant input and output in a comprehensible manner without overwhelming the user with the wealth of available information and possibilities (3).

The interaction helps each group to gain a better understanding of the needs the others have and the possibilities they offer. For this understanding to take full effect, an iterative approach is best suited. After the initial goals and requirements of the system have been established, IT-specialists can set up a prototype system to help assess its usability and show policy makers the possibilities. Their feedback can be used to make the model more robust, add missing input or output to the system and improve the operability of the user interface. The prototype can be used by scientists to train policy makers – and their technicians – in the use of the system. This includes translating real-world problems or questions into interventions in the system and translating an analysis of model output into concrete, valuable recommendations or conclusions.

Besides giving scientists a better understanding of the policy making process, such consultancy can help increase the practical value of the system – that is, make sure that it will be used and that it will be used appropriately. Due to its integrated approach, the utilisation of the ISDSS could even have an impact on the work-practice of the involved organisation. After some time, an extension of the system or refinement of some part may prove desirable and the development process can enter another iteration.

1.2 Perspectives on the system

Not only do the three parties involved in the development of a spatial DSS have different roles, they also have a different view on the system. These different perspectives can cause problems in communication or false expectations. What seems trivial to a policy maker

may be a complicated task for a modeller or IT-specialist. Conversely, small changes to the model or software can make a huge difference in the eye of a user of the system.

To policy makers the system acts as one whole; particular input will produce particular output. Though their understanding of the processes involved may be richer or poorer, their focus is aimed at the use of the system, not at its internal structure. Scientists and IT-specialists need a higher level of system specification [Zeigler, et al. 2000]; they need to know the internal structure. The decomposition of a system into coupled components helps scientists to understand it by dividing research or knowledge into comprehensible parts, often focused per discipline. For IT-specialists, the benefits are even greater, as they have the ability to design, implement and test each component individually. As a result, their decomposition may be more extensive than that of scientists. Finding an appropriate decomposition can be a difficult, time-consuming task. This can be lightened by the use of a framework, however.

1.3 Requirements for a framework / ISDSS

The previous sections give an outline of the process of developing an ISDSS and the common communication problems occurring between the parties involved in such development. From these insights we can derive four requirements for an application framework to successfully support this process, meaning its use will reduce development costs and result in a more stable and better suited product.

1. Modularity
2. Scalability
3. Powerful system in terms of size, speed and model complexity
4. Interactive system

The first two requirements stem from the iterative development process itself and relate more directly to the framework, while the last two are imparted common requirements of the ISDSS and relate also to the skeleton application the framework offers.

To support iterative development, a framework must reduce the costs of subsequent iterations by allowing easy adaptation or improvement of the work in the previous phase and by supporting the extension of the system with new models or functionality. This firstly requires *modularity* of the framework, meaning the entire system can be decomposed in a prescribed way into components that interact through well-defined, stable interfaces. When local changes must be made, modularity helps to keep the impact of those changes localised, thereby keeping development costs limited and allowing parallel work on distinct components. Secondly, the framework and system should be *scalable* in the sense that extensions can be made without the need to revise previous work and without the system growing excessively complex or resource consuming. This prevents the ISDSS developer from having to start from scratch when the user would like to see an extension of the ISDSS's model or functionality.

After the system has been expanded in several iterations, it can grow to a substantial piece of software, particularly compared to the prototype version. To still be able to use this application in a similar way as the first one – for example, running it on a desktop computer with computation times in the order of minutes – the final system has to be *powerful* in terms of size (of data and models) and speed (of data access and model computation). The framework that was used to build the first, lightweight version must support the final, heavyweight version as well. Besides the increased size of data or models, subsequent iterations of the development process can also raise the need for greater feedback between models, thereby raising model complexity. The model framework, upon which the models are built, should be able to support this increased complexity without the need for a complete redesign.

One of the general requirements for any DSS is user-friendliness. This requires from the framework that the DSS developer is able to design the user interface as he deems suitable. Therefore, the framework should not oppose design, layout or functionality on the user interface. Still, a common requirement will be for the user to be able to interact with the

system directly. This means the framework cannot afford to lock up during computation. It must stay responsive to user input and process these changes to display the correct output.

2. GEONAMICA®

GEONAMICA® is the object-oriented application framework [Fayad, et al. 1999] developed by RIKS to build decision support systems based on spatial modelling and (geo)simulation. It has been developed over the past 15 years and has been used to generate integrated spatial decision support systems, such as **WADBOS** [Engelen, et al. 2003a], **ENVIRONMENT EXPLORER** [Engelen, et al. 2003b], **MEDACTION** [Van Delden, et al. 2007], **XPLOAH** [Van Delden, 2008] and **MOLAND** [Barredo, et al. 2003]. Besides these, RIKS has used **GEONAMICA®** to develop **METRONAMICA**, a template SDSS that includes a local dynamic land use interaction model, a regional interaction model and/or a transport model – depending on the exact version. It can be used to set up a specific SDSS without the need for additional software development by filling the system with data, calibrating the model and training the users.

In ISDSSs, such as the ones mentioned above, we can distinguish three major components: a database to store information used by the system – mostly raster or vector map data, time series data and cross-sectional data –, a model base to manage the models that are used and a user interface to enable the user of the system to interact with it. Setting up each of these components and letting them work together should be facilitated by the application framework as much as possible.

GEONAMICA® offers set components for the storage of map data, time series and cross-sectional data. It provides a modelling framework based on the Discrete Event System Specification (DEVS) formalism [Zeigler, et al. 2000] and includes a model controller that manages the models, makes sure they interact properly and tells each model when to perform certain, predefined actions. To create a user interface, **GEONAMICA®** includes a skeleton structure and a rich class library of user interface components, such as map display and editing tools, list and table views and two-dimensional graph editing components.

The strength of **GEONAMICA®** lies in the fact that the modelling framework provides a generic structure for the models that allows them to be integrated more easily, while enabling complex dynamic models to be executed efficiently. The environment is set up in such a way as to enable users to run simulations interactively, by allowing them to intervene in the system and observe the results of their actions directly in a comprehensive manner or save the results to persistent storage for more elaborate analysis or presentational purposes.

2.1 Model blocks

The modelling framework in **GEONAMICA®** builds on the DEVS formalism. Specifically, the entire model is composed of *model blocks*, which are encapsulated parts of a model that can communicate with each other and with the system through a standardised interface. A model block contributes to the entire model – in its simplest representation as a collection of variables and equations or algorithms – with the variables it contains and the procedures associated with it to compute the value of these variables. Data hiding is particularly applied to the variables of a model block that can only be accessed through dedicated ports.

Model blocks communicate with each other through input and output ports. An input port allows a model block to gain read-only access to a variable of another model block, such that it can be used for computation. An output port allows read-only access to one of the variables of a model block. By linking all input ports of a model block to output ports of other model blocks, we create a coupled component. The collection of interlinked components forms the system that represents our entire model. For each particular system, the model is specified in an XML file. The listed model blocks are instantiated and coupled at run-time, thereby allowing a change of the model without the explicit need for additional programming and allowing different versions of a model to be maintained in parallel.

Note that the entire model can be represented by model blocks hierarchically. That is, we can define a composite model block as the collection of a set of model blocks and the links between their input and output ports. Input ports of model blocks in the composite that are not connected to output ports of model blocks in that composite are automatically rerouted to input ports of the composite model block. The set of output ports of the composite consists of all output ports of the contained model blocks.

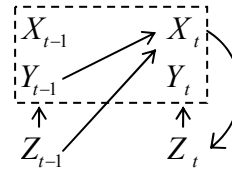
2.2 Simulation engine

The variables in a model block represent the state of a real-world or imaginary entity at some point in time or over some time period. This means that we cannot store the value of some variable for two moments in time without explicitly making a copy. The *simulation engine* tells each model block when to compute the value of its variables for a specific point in time. The model block responds by telling the simulation engine when it should calculate again. To keep the model output comprehensible for the user, the simulation engine always keeps all model blocks up to date with the global simulation time.

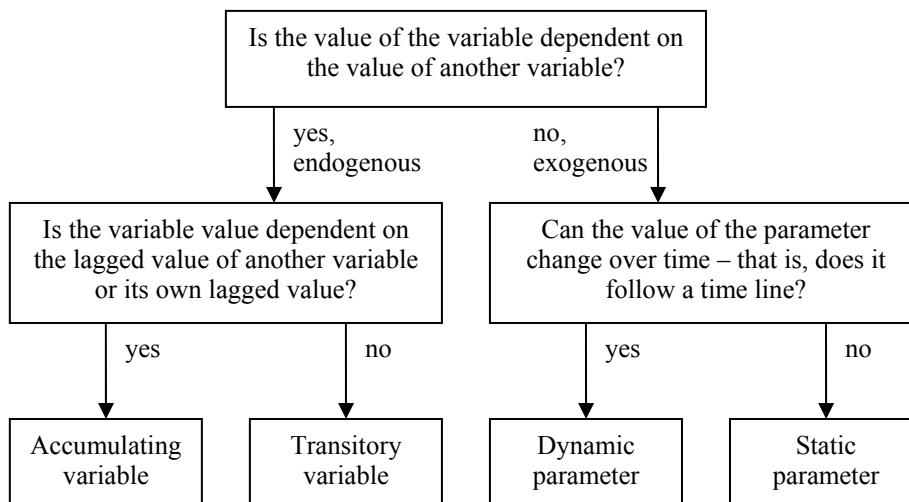
This paradigm can present a problem when we calculate more than one variable in a model block. Consider, for example, a model with two model blocks. Model block A has the variables X and Y and model block B has the variable Z . The value of the variables is calculated as:

$$A: \begin{cases} X_t = f(Y_{t-1}, Z_{t-1}) \\ Y_t = g(Z_t) \end{cases}$$

$$B: \begin{cases} Z_t = h(X_t) \end{cases}$$



On the right, these relations are depicted schematically. The arrows indicate that a variable (head) is dependent on the value of another variable (tail). We can easily see that X should be calculated before Y and Z , and Z should be calculated before Y . However, if X and Y are in the same model block, they should be calculated simultaneously, since calculation procedures are associated with model blocks, not with individual variables. To be able to represent this model in our paradigm, we have to make a distinction between variables that are dependent on a lagged value and variables that are only dependent on current values. We call the former *accumulating* variables and the latter *transitory* variables. So X is an accumulating variable and Y and Z are transitory variables. The complete determination of the type of a variable is depicted in the schema below.



In order to be able to model the above example, we have to split the computation of a model block into two parts; one in which all accumulating variables are calculated and one in which all transitory variables are calculated. First, all accumulating variables are calculated for all model blocks. Next all transitory variables are calculated. This separation

does not resolve all precedence relations. In the example above, we see that the transitory variables of model block A should still be calculated before those of model block B.

Since accumulating variables are dependent on lagged values, their value needs to be specified for the start time of a simulation. In the initialisation of the model, copying this value to the current value replaces the calculation procedure for accumulating variables.

Once we know the dependencies between variables, the remaining precedence relations can be derived automatically, as depicted below. On the left we see a non-lagged relation and on the right we see a lagged relation. Depending on the type of the variables X and Y we have six different possibilities, as listed in the table below, where T stands for a transitory variable and A for an accumulating variable.



Relation	X_{t-1}	X_t	Y_t	Meaning	Brief
$Y_t = f(X_t)$	-	T	T	X must be calculated before Y is calculated.	$X < Y$
$Y_t = f(X_t, Y_{t-1})$	-	T	A	This is impossible, since all accumulating variables must be calculated before all transitory variables.	-
$Y_t = f(X_t)$	-	A	T	This is ok; all accumulating variables are calculated before all transitory variables.	ok
$Y_t = f(X_t, Y_{t-1})$	-	A	A	X must be calculated before Y is calculated.	$X < Y$
$Y_t = f(X_{t-1}, Y_{t-1})$	T	-	A	This is ok; all accumulating variables are calculated before all transitory variables.	ok
$Y_t = f(X_{t-1}, Y_{t-1})$	A	-	A	Y must be calculated before X is calculated.	$Y < X$

The relations between variables can be derived from the connections between input and output ports. Hence, the order in which the variables of the model should be calculated can be derived once we know the model coupling. To upscale this order to model blocks, we add equality relations (in terms of precedence) between all accumulating and between all transitory variables of the same model block. Note that, since the precedence relations between variables are relevant to either the accumulating or transitory computation phase, depending on the type of variables involved, the order in which model block calculate can differ in the two phases.

The **GEONAMICA**® model framework builds on the DEVS formalism, but does not comply with it fully. In the DEVS formalism, the distinction is made between rate and state variables. First, the change of all state variables are calculated and stored in rate variables. Next, the state variables are updated with the rate variables. This method allows model blocks to calculate completely independently at the cost of having to store the change of each variable explicitly, even when a variable could be updated directly. The **GEONAMICA**® model framework takes advantage of such redundancies to reduce the strain on resources. This comes at the cost of generality. Note that any specific model that can be implemented using the DEVS formalism, can be also implemented using the **GEONAMICA**® modelling framework. Hence, the consequences are entirely practical.

2.3 Incorporating user input

We mentioned before that the simulation engine always keeps all model blocks up to date with the current simulation time to present a comprehensible output to the user. This means that in the course of a simulation, we iteratively take time steps to advance the state of the system. At the start of a time step, we update parameter values changed by the user and recalculate the transitory variables dependent on these parameters. Next, we can advance the simulation clock and calculate the accumulating variables of each model block in the

order derived from the precedence relations between variables, as explained above. Finally, the transitory variables of each model block are updated in their respective order and the new output is presented to the user. The next simulation step is performed when the user instructs the system to do so or automatically if desired.

If we strictly follow the outset above, the user could interact with the system only after a time step is completed and before the next one has started. This is, of course, an unacceptable characteristic for an interactive system. However, allowing a user to alter parameter values in the middle of a simulation step would result in undetermined behaviour of the system. Hence, we need a mechanism that allows the user to interact with the system during the course of a simulation step, but guarantees the model that parameter values remain constant during this period. This mechanism has been incorporated in the interface ports of model blocks.

Interface ports provide access to the parameter values of a model block, which can be altered by the user between simulation steps. The changes made in the user interface are cached in the interface port to which the user interface is linked. At the beginning of a simulation step, the user interface ports are instructed to relay their cached changes to the actual parameter values. This way, we require no further synchronisation between user interface and model processes, as far as the updating of parameter values is concerned, thereby greatly reducing the overhead caused by such synchronisation issues.

3 CONCLUSIONS

The **GEONAMICA®** framework has been designed to support the development of ISDSSs and to meet the requirements set out in section 1.3. In the next section, we will elaborate on the way in which those requirements are met. The last section will discuss the limitations, possibilities for improvement and future expectations.

3.1 How does GEONAMICA® meet the requirements?

The four requirements set out in section 1.3 are met by the **GEONAMICA®** framework through different concepts incorporated therein. The main requirement of modularity is met by the decomposition of the model into model blocks and the ability to specify the complete model at run-time. Model blocks offer a clear method of decomposition that allows the model framework to implement common procedures, such as reading or writing to file. The input, output and interface ports form a stable interface for interaction between model blocks themselves and between model blocks and other components.

The fact that the model is specified at run-time turns models blocks into components that can be replaced without the need for recompilation. Combined with the possibility to specify the model hierarchically, this functionality allows one to replace an entire sub-model with another one in the blink of an eye. For example, we can replace a simple model block containing scenarios for certain exogenous variables with a sub-model that computes these variables endogenously. Except for the implementation of the sub-model itself, which only has to be done once, there is no extra programming required to incorporate it into the integrated model of the system. Additional feedback loops in the model are handled by the simulation engine, which determines the computation order of the model blocks.

By reducing the storage of redundant data – by storing each variable for only one moment in time and by introducing precedence relations for the computation of model blocks –, the **GEONAMICA®** framework allows extensive, integrated models to be incorporated in an ISDSS running on a desktop computer. By building on the DEVS formalism, the model framework allows a high level of model complexity, while keeping the strain on resources limited.

The user interface that **GEONAMICA®** offers is only an empty shell. Hence, the DSS developer is free to design the user interface as best suitable for the user of the system. One could, for example, create a different user interface for different groups of users, possibly with a different background, objective or authority. The framework does, however, offer

support for synchronisation between user interface and model processes with the caching mechanism incorporated in the interface ports. This significantly lightens the load on the user interface programmer to deal with synchronisation issues, which is one of the harder problems to tackle in the design and implementation of a user interface.

3.2 A look into the future

The GEONAMICA® framework has been developed with generality in mind. Its structure is based on general requirements that follow from an understanding of the development process of an ISDSS, as set out in section 1.3. The goal, however, has not been to develop a framework that can be applied in a very wide range of conceivable applications, but to develop a framework to support the actual ISDSS development that happens in practice. As a result, the framework has a limited scope of situations in which it is useful.

GEONAMICA® is a suitable candidate for an application framework to develop a DSS that incorporates a discrete time simulation model. It is designed for systems to be used interactively on a modern desktop computer running a Windows operating system. For reasons of efficiency of execution, the framework has been implemented and can – at this point – only be used in the C++ programming language. All these are technical limitations that will render GEONAMICA® unsuitable for a good number of projects. There are, however, a significant number of projects with the goal to develop an (IS)DSS that can benefit greatly from the use of the framework. The examples listed in section 2 demonstrate the practical value.

As the requirements posed for ISDSS development get stricter and competition becomes stronger, we need to keep developing the GEONAMICA® framework to improve the available components and overcome the current limitations. At RIKS, we are working towards platform independency to allow a broader range of applications of a system – for example, web-based access to models running on a server. A next step would be greater independence of programming language, which can be achieved by the ability to incorporate models that comply with compatible standards.

REFERENCES

- Barredo, J., C. Lavalle, L. Demicheli, M. Kasanko and N. McCormick, *Sustainable urban and regional planning: The MOLAND activities on urban scenario modelling and forecast*, Office for Official Publications of the European Communities, Luxembourg, 2003.
- Densham, P.J., Spatial Decision Support Systems. In: *Maguire, D.J., M.F. Goodchild and D.W. Rhind (eds.), Geographical Information Systems: Principles and Applications*, 403–412, 1991.
- Engelen, G., I. Uljee and K. van de Ven, WadBOS: Integrating knowledge to support policy-making in the Dutch Wadden Sea, In: *Geertman, S. and J. Stillwell (eds.), Planning Support Systems in Practice*, 513–537, 2003a.
- Engelen, G., R. White and T. de Nijs, Environment Explorer: Spatial Support System for the Integrated Assessment of Socio-Economic and Environmental Policies in the Netherlands, *Integrated Assessment*, 4(2), 97–105, 2003b.
- Fayad, M.E., D.C. Schmidt and R.E. Johnson, *Building Application Frameworks: Object-Oriented Foundations of Framework Design*, Wiley Computer Publishing, 664pp., New York, 1999.
- Hahn, B. and G. Engelen, Concepts of DSS systems in German Federal Institute of Hydrology, *Decision Support Systems (DSS) for River Basin Management*, 9–44, 2000.
- Van Delden, H., P. Luja and G. Engelen, Integration of multi-scale dynamic spatial models of socio-economic and physical processes for river basin management, *Environmental Modelling and Software*, 22(2), 223–238, 2007.
- Van Delden, H., E. Gutiérrez, J. van Vliet and J. Hurkens, Xplorah: A tool supporting Integrated Spatial Planning in Puerto Rico, *paper presented at this conference*, 2008.
- Zeigler, B., H. Praehofer and T-G. Kim, *Theory of Modeling and Simulation*, Academic Press, 510 pp., 2000.