Conceptual Microarchitectures for Hydrologic Simulation Models

Urciuolo Adriana
Universidad Nacional de la Patagonia San Juan Bosco
Darwin esq. Canga- 9410 Ushuaia, Argentina, 54-2901-443533
urciuolo@tdfuego.com

Iturraspe Rodolfo
Universidad Nacional de la Patagonia San Juan Bosco
Darwin esq. Canga -9410 Ushuaia, Argentina, 54-2901-432217
iturraspe@tdfuego.com

Parsón Ariel
Universidad Nacional de la Patagonia San Juan Bosco
Darwin esq. Canga, 9410 Ushuaia, Argentina, 54-2901-430892
a-parson@infovia.com.ar

ABSTRACT

Mathematical Hydrologic models simulate real world environmental processes through different strategies. Each process is calculated by means of methods that utilize physical parameters for representing the real world system. Some parameters are obtained from tables, some of them are optimized and others may be calculated using environmental variables. Although the domain software provides a wide range of models, there is not a conceptual architecture that allows the maintenance of the vast knowledge about simulation strategies and parameters collected in environmental management organizations, facilitating the flexible simulation scenarios configuration. The present work shows how to face this problem by means of conceptual analysis models organized in the scope of a general architecture.

It's also possible for the given architecture, to analyze and define microarchitectures for software components related to particular problems.

In the present work, conceptual microarchitectures are defined to construct a knowledge level for hydrologic models systems starting from a general conceptual Environmental Information Systems architecture. To get the required flexibility for the conceptual and design models, high-level components are identified and different kinds of patterns are applied.

Keywords:
Architecture, analysis patterns, simulation models, parameter.

1. INTRODUCTION

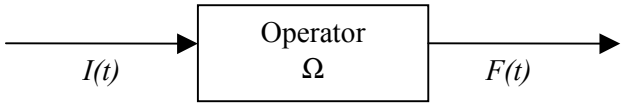
In the last years, great advances have been made in techniques for exploring the hydrologic environment, such as remote sensing, radar and others. Due to limitations of these techniques, there is only a limited range of measurements in space and time, which is normally stored in environmental organizations. Therefore, models of different types provide a means of quantitative extrapolation or prediction from the available measurements in space and time [6]. Models generally make assumptions about real world

processes in order to predict the behavior of systems under certain conditions to assess the likely impact of future hydrological change.

The ultimate aim of predicting using models is to improve decision-making about some hydrological problems in Water Resource Management, such as water resource planning, flood prediction, determination of risk areas, mitigation of contamination and so on.

Modeling has existed for at least one hundred and fifty (150) years. It is based on the idea of conservation of mass, momentum and energy. The governing equations for environmental models resulted from experimentation.

A hydrological model is a simplified representation of the real world system which aim is to study the system operation and predicting its outflow. Model inputs and outputs are hydrologic and climatic variables such as water flow, precipitation, temperature, evaporation and others. The model structure is a set of mathematical equations connecting inputs with outputs [11].



F(t) = Ω I(t) system transformation equation

As it was said, hydrological models serve a range of purposes but they are used primarily to estimate runoff from sequences of rainfall and the meteorological information needed to estimate potential evaporation. They can be used to estimate river flows at ungauged sites, fill gaps in broken records or extend flow records with respect to longer records of rainfall.

Generally, the relationship between rainfall and runoff is a central problem in hydrology. According to this criteria, models can be classified in empirical, conceptual or physically-based. Mathematical modeling of this relationship is often based on a black box type of model with less regard to the physics of the process. This is partly due to the complexity of the process and data availability. However there is a continuing interest among hydrologists to develop rainfall-runoff models with a theoretical structure based on physical laws. Powerful areally-distributed models are based on physical principles governing the movement of water within a catchment area, but they need detailed high-quality data to be used effectively. More commonly, simpler conceptual models are used to represent the basin as a whole. The main controls on water movement are represented by quasi-physical model elements whose action is governed by a set of model parameters. In some circumstances, these parameters can be adjusted to represent changes to land-use in the catchment area.

Computational models present a clear evolution process, due to the computational advances to which they were adapting. It's common, when tracing the development of models to distinguish five generations [1], but hydrologic simulation models as software products for decision-making arose in the fourth-generation, with some new requirements such as friendly job environments, use of models by non domain experts, etc., giving place to the creation of a new paradigm called hydroinformatics [1]

Despite its evolution, there are some critic questions to face in the software domain, related with the lack of a conceptual framework for Modeling in Water Resource Management. There are many different types of hydrologic models, considering the simulation strategies and methods used for flow computing, but they are often isolated from the Environmental systems, which provide basic services and information to them. The complex models used for hydrologic studies require extensive parameter input which decreases model accuracy. Sometimes it's a serious restriction for their use. On the other hand, modeling software usually is based over a static design model, which doesn't allow dynamically changing methods or parameters if they are not available or if they are not the appropriate for the real system simulation. In summary, there is not a domain software architecture that facilitates the maintenance of the hydrological

processes simulation methods and parameters knowledge for the flexible configuration of simulations scenarios and the integration of models to Environmental Information Systems.

High-level domain specific software analysis appears as a possible solution for such topics.

The problem may be characterized as follows:

- ✓ There is a great diversity of methods for each hydrological process simulation by means of different strategies (black box, conceptual and physically based models).
- ✓ There are different parameters for each type of hydrological structural component (lake, river, soil, etc.), each simulated process and selected method used by models. Some parameters can be obtained from tables, some of them can be calculated from environmental variables and others are obtained using optimization methods.
- ✓ Environmental management organizations collect important knowledge through the years about parameters that better fit the real systems conditions for a given region.
- ✓ Simulation methods and parameters may use the data from existing Environmental Information Systems for flow computing, but models are often acquired and used isolated from other systems.

The domain has been traditionally characterized by the study of efficient flow computing algorithms, but not of high-level solutions for the exposed questions. In the present work, it is proposed to introduce flexible software developing mechanisms, such as architecture, patterns and components in the construction of specific domain software.

Software architecture deals with the design and implementation of the high-level structure of the software [5]. Architectural descriptions are being recognized as an appropriate vehicle for codification and reuse of design knowledge. Software domain specific architectures provide an organizational structure for a family of applications, that allow to organize their development and sustain their evolution and reuse.

By other hand, for a given architecture style, it's possible to define a set of conceptual, design and architectural patterns, which act as "microarchitectures" [21]. They constitute appropriate techniques to solve particular problems, such as those exposed above.

Patterns are used in software engineering to allow the reuse of solutions that have been prove to be successful in the different software process stages. There are different types of patterns, which may be classified by many points of view, such as: abstraction level, scale ranges, dependency on a specific particular domain and software process stage [22].

This article provides conceptual microarchitectures for flexible hydrologic modeling software on the basis of a general conceptual architecture that facilitates simulation models integration to Environmental Information Systems. The remainder of this paper is organized as follows. Section 2 reviews software techniques for flexible software development. Section 3 presents the conceptual analysis process. Section 4 includes the proposed design models for flexible simulation scenarios configuration. Section 5 shows an example use case of the models. Section 6 is related with acknowledgements and section 7 presents conclusions and future works.

2. FLEXIBLE SOFTWARE: PATTERNS, ARCHITECTURE AND COMPONENTS.

Architecture has emerged as a crucial part of the design process. Larry Bass [5] defines software architecture as: "The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them".

From the above definition, it is inferred that when proposing specific domain software architectures, some information is abstracted to concentrate in relations and high-level structures, providing enough information to constitute the basis for analysis.

It's possible to use different architectural views to enhance the understandability of the architecture and to focus on particular concerns separately. Some studies [20] propose the following three views: conceptual, logical and execution view.

The Conceptual Architecture identifies the high-level components of the system, and the relationships among them [4]. Its purpose is to direct attention at an appropriate decomposition of the system without delving into details. It consists of the Architecture Diagram (without interface detail) and an informal component specification for each one.

In Logical Architecture, the externally visible properties of the components are made precise and unambiguous through well-defined interfaces and component specifications, and key architectural mechanisms are detailed.

Execution Architecture is created for distributed or concurrent systems. The process view shows the mapping of components onto the processes of the physical system.

In the present work the emphasis is put on the conceptual architecture analysis. Defining this basic initial structure, then it's possible to develop microarchitectures for software components related with specific domain problems. This architecture may be refined in successive iterations of the software process.

As it was said, fundamental to conceptual software architecture is the structure of the system in terms of the primary structural elements or *components of the system*, and their interrelationships. The use of components allows identifying independent units [23] for new appropriate solutions, as well as the reuse of existing solutions.

Patterns have been used in different domains to define microarchitectures for particular problems. An *architectural pattern* [10] expresses a fundamental structural organization or schema for software systems. It provides a set of predefined subsystems, specifies their responsibilities, and includes rules and guidelines for organizing the relationships between them [3]. A *conceptual (or analysis) pattern* is a pattern whose form is described by means of terms and concepts from an application domain. Analysis patterns were proposed by Fowler [13] to make possible the reuse of conceptual modeling solutions. This term is utilized for patterns used to describe solutions for problems that appear during the requirement and conceptual modeling stage. These patterns facilitate the transformation from analysis to design model. A *design pattern* describes commonly recurring structure of communicating components that solves a general design problem within a particular context. A *design pattern* can be defined as a common solution to recurring problems in software design [15]. Provides a scheme for refining the subsystems or components of a software system, or the relationships between them.

Although there are few antecedents of patterns application in hydroinformatics systems [2] [24] [25], they are important for the study of specific environmental domain architectures, as it can be seen in recent researches related with Geographic Information Systems [19] [16].

3. CONCEPTUAL ANALYSIS

3.1 Methodology

The methodology for the early stages of software development is based on the RUP (Rational Unified Process) Process Development [8], considering its basic features: case use driven, architecture centric, iterative and incremental.

First it is defined the general conceptual architecture to be used as a framework for hydrologic models software development. It is used an existing conceptual architecture for Hydrological Modeling integrated to Environmental Information Systems. It was developed on the basis of a Physical Water Resource Domain Model, which was used for the domain objects and analysis classes identification [7] [24].

This architecture primarily supports the functional requirements stated for Environmental Information Systems. UML [9] is used for the architectural representation, by means of *packages* and *class diagrams.* Packages are used as layers, subsystems and *high-level components*.

Conceptual microarchitectures are used as a vehicle from analysis to design. This way it's possible to bring solutions for specific design problems beginning from the study of high-level structures. They are developed on the following basis:

- ✓ The existing domain model and analysis classes defined for *Water Resources Information Systems* (WIRS) [24] [25].
- ✓ A feature list and an example use case, which are used as drivers for the diagrams construction.
- ✓ The identification of the main hydrological processes corresponding to each hydrologic object.
- ✓ The study of the common simulation methods and parameters used by hydrologic models.

Conceptual analysis is made applying analysis patterns [13] [14].

3.2 Conceptual General Architecture

The general architecture shown in Figure 1 is used as a scheme for the integration of Hydrologic simulation models to Environmental Information Systems.

The architecture is primarily based on the architectural pattern Layers. The *application general layer* is called: "Environmental Information Layer" [17]. In this layer environmental subsystems like: Water Resource, Soil, Climate, Vegetation are interconnected via interfaces. The *application specific layer*: "Simulation Layer" includes subsystems referred to different simulation models that use the data and services from the Environmental Layer subsystems.

In the *middleware layer*: "Geographic Representation Layer", there is a conceptual framework for SIG which is called GeoFrame[18]. Subsystems in the application layers can specialize classes and relationships in the framework for the hydrological components geographic representation. The system-software layer is not shown in the diagram.

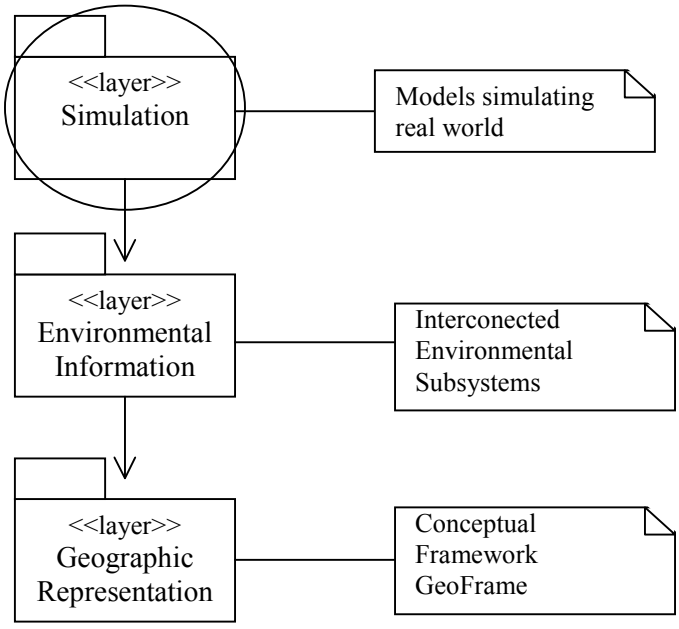


FIGURE 1. Conceptual Architecture for Environmental Information Systems

Lower Layers (middleware and system-software level) are general to several applications.

As it was said, the Environmental Information Layer contains different environmental systems: Soil, Water Resources, Climate, Terrain Ecology, Land Use, etc. The general architecture for this layer is based on the architectural pattern "System of interconnected systems" [12].

These types of systems are divided into several separate parts, each developed independently as a separate system, as it can be seen in Figure 2.

A super system is implemented by a set of interconnected systems, communicating each other through interfaces owned by the super system. One of these systems represents the overall capabilities, and it is called the **superordinate system**. The other systems represent a part of the whole, and they are called **subordinate systems.**

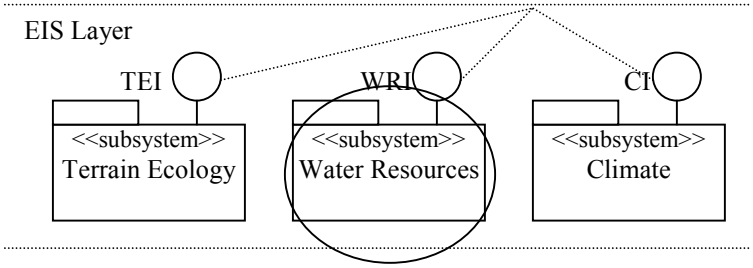


FIGURE 2. Interconected subsystems for Environmental Layer

The Geographic Representation layer contains a conceptual Framework GeoFrame [18], which is specialized in the EI Layer for the environmental objects representation in some convenient system.

3.3 Simulation Layer Conceptual modeling Requirements

It's possible now, to identify the main components of the Simulation layer analyzing their interactions and defining the required conceptual models.

When using an iterative process, it is recommended to keep the initial ideas about the system in a feature list, which is used as a starting point for the candidate requirements identification [4] and the initial system architecture definition.

Following this approach, the conceptual model requirements for the Simulation Layer are the following:

- It must bring facilities for different hydrological phenomena simulation, maintaining the knowledge about simulation strategies (methods for hydrological processes calculation).
- It must allow the parametric information maintenance either for those parameters obtained from existing tables or generated through calculations.
- It must be possible to use environmental data and services from the EIS layer.
- Simulation scenarios configuration must be flexible.

Considering the candidate requirements derived from the feature list as system use cases, in the present work it is used the system use-case: "Calculating the runoff process in a catchment using a desired flow computing method" as an example of a specific requirement for the conceptual architecture development.

3.4 Hydrologic Models Knowledge Analysis

Hydrologic models represent the nature of the system through mathematical equations which are solved in computational way [11].

Mathematical models represent the physical hydrological processes in different ways. Process models can be classified as *empirical*, *conceptual* or *physically* based.

A model that is based on analysis of input and output with the system considered as a black box is termed empirical. A conceptual model uses a combination of physical representations and empirical equations to formulate the model of the real-world system. Usually the parameters must be calibrated using observed input and output data. A completely physically based model is based upon mathematical process descriptions of the real world processes. The parameters should in principle be possible to estimate from measurements [2].

System models can further be classified as *lumped* or *distributed*. A lumped model considers the real-world system as one unit, averaging parameters and states over the system. On the other hand, a distributed model allows parameters and variables to spatially vary over the system it represents. Usually the catchment is subdivided in subcatchments or grid cells for a better representation of the real world phenomena.

4.3.1. Objects

Environmental objects can be grouped into a number of classes that structures the environment into three media: ground, water and air. This simplified taxonomy is commonly used by government agencies. For the purpose of the present work, and because of the domain features, it is used the Taxonomy for WIRS Domain Objects [24]. This classification identifies five types of objects: hydroecological objects, human activity objects, measuremnt objects, atmospheric variables and hydrologic variables.

4.3.2. Processes

Basic hydrological objects processes to be simulated by models are shown in Figure 3. They are identified from an existing Physical Domain Model [24] for WIRS.

Usually Rainfall-Runoff models include two computing components: runoff and routing. The runoff component represents hydrological processes within a subcatchment or grid cell (depending on the model type), which computes discharge to the channel network within this element.

The routing process consists in tracking the water through the river-network. This computing component gets discharge form the runoff component and river inflow from the upper reaches in the adjacent model elements and it calculates river outflows to lower reaches within the adjacent elements.

4.3.3. Methods

There are different methods for each type of process simulation, which can be found in the specific bibliography [6].

Methods related with the runoff component computing include the calculation of processes such as: evapotranspiration, infiltration, snow melting, overland flow, interflow, base flow, etc.

By the other hand, there are some methods related with the routing component computing, which take into account the selected calculus strategies like cinematic wave, dynamic wave, etc.

4.3.4. Parameters

Model parameters describe the entities that are constant in the model representation. Minimal parameters [6] required for a catchment-scale process-based model are:

- ✓ Subsurface flow parameters for each soil type, like hydraulic conductivity, porosity, etc.

- ✓ Vegetation parameters for each vegetation type, like interception storage capacity, canopy resistance, etc.
- ✓ Overland flow parameters for each slope element in the basin
- ✓ Channel flow parameters
- ✓ Snow parameters, like degree-day factor.
- ✓ Other parameters specific to methods.

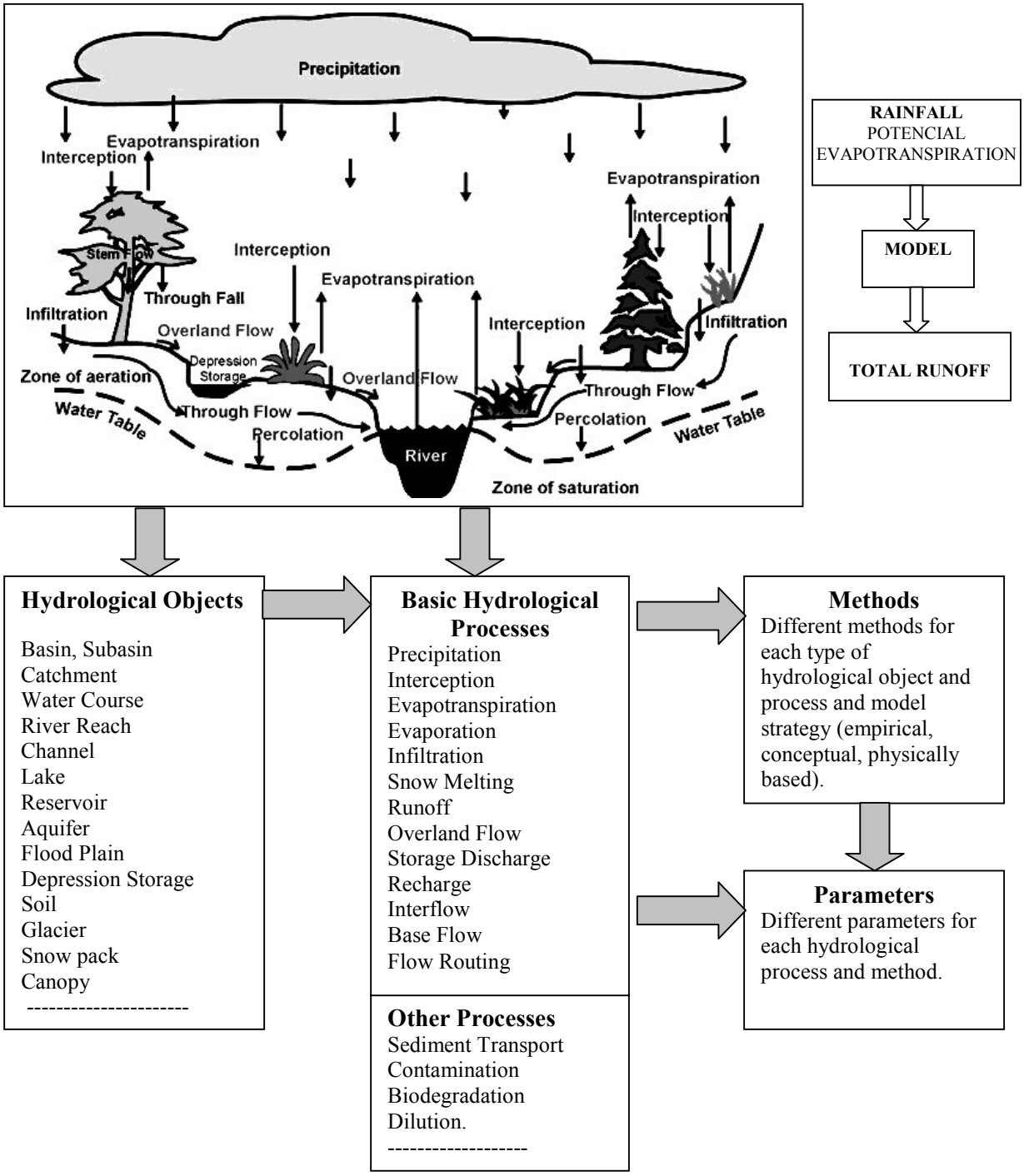


FIGURE 3. Hydrologic process simulated by models

3.5 Conceptual Microarchitectures for the Simulation layer

In the following sections there are defined conceptual microarchitectures that facilitate the flexible models scenarios configuration

3.5.1 Hydrologic Models Knowledge Maintenance

Each model may present different configurations depending on the hydrologic process involved in the simulation session and on the type of model, methods and parameters selected for the specific scenario. Then, the system must "know" all legal simulation configurations for each type of hydrological component (Lake, river, snow pack, etc.).

For that, it is applied the *Knowledge Level* analysis pattern [13] as it is shown in Figure 4.

There are two levels defined inside the Simulation Layer: *knowledge level and operational level.* The knowledge level includes a group of objects that describe how another group of objects should behave.

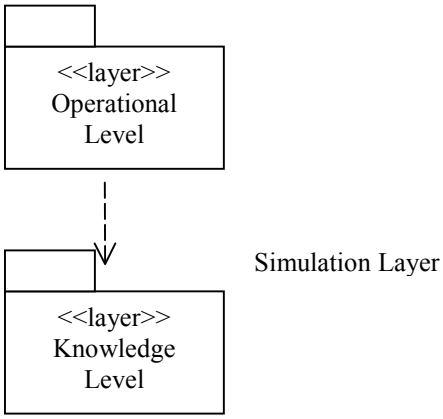


FIGURE 4. Levels for the Simulation Layer

In the knowledge level there are all possible legal modeling configurations: simulation strategies and parameters associated with different hydrological processes.

Then, it´s possible to identify two components in the knowledge level, defining independent units with the purpose of increasing flexibility: Simulation Strategies Component and Parametric Component, as it's shown in the Figure 5.

In the Operational level, it is defined a *Structural Component*, which maintains the simulation scenarios specific behavior to be added to the WRIS hydrologic objects, such as Lake, RiverReach, Catchment, etc. (EI Layer). Convenient structures and conceptual models for WRIS have been studied in previous works [2] [25] [26]. *Data containers* are generic classes that allow the efficient incorporation of state variables to the Structural Component at execution time.

The Components that have been identified for each level and their dependency relationships are shown in Figure 5, using the UML stereotype <<subsystem>> for high-level components, which internal behavior will be analyzed in the next section.

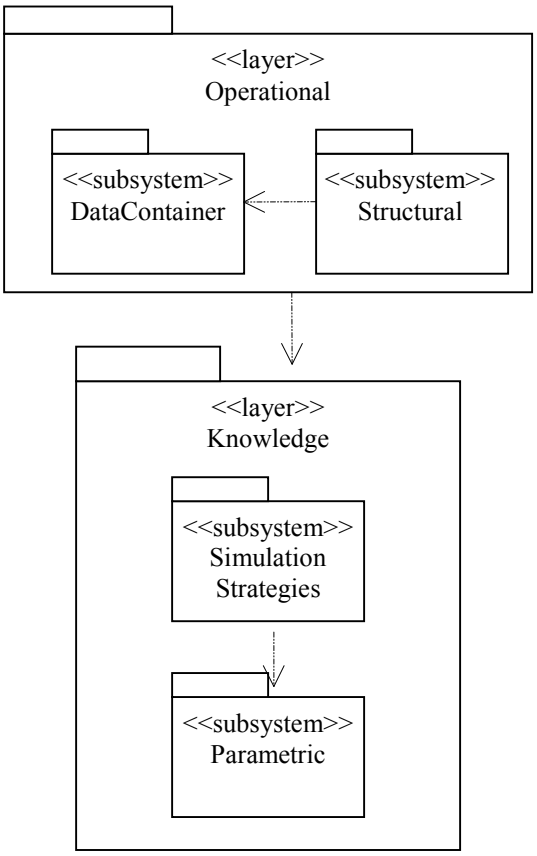


FIGURE 5. Components of Simulation Layer levels

3.5.2 Simulation Strategies.

The basic conceptual model for this component is shown in Figure 6. It allows models scenarios configuration using different processes and computing methods for each one. The *HydroProcess* class maintains information about the process being modeled (Runoff, Routing, infiltration, evapotranspiration, etc). The *Method* class contains the different computing definition for each process. *HydroProcess* may be an aggregation of processes.

Structural component objects may be configured with a Type indicating the selected modeling strategy, for example, if ModelObject is a Lake, then a possible type is Routing (process), Level Pool (method). This conceptual relation (between structural components and strategies) is shown as a dependency in Figure 5.

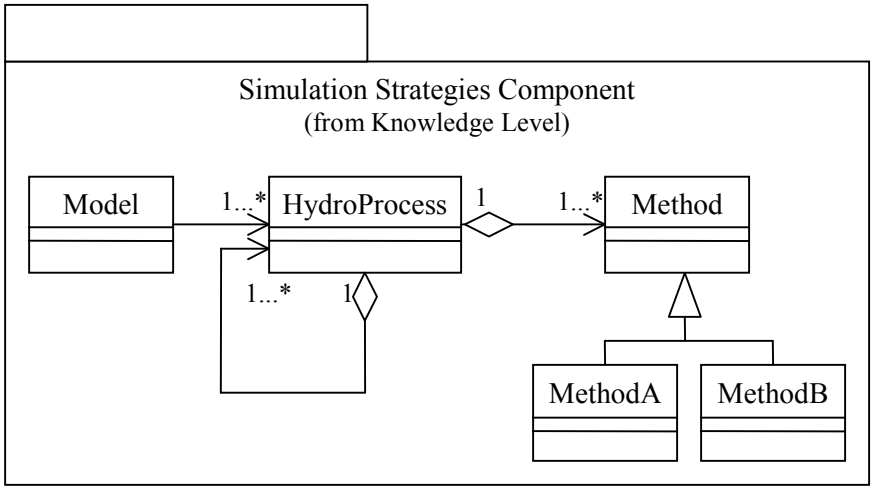


FIGURE 6. Class Diagram for Simulation Strategies Component

3.5.3 Parametric Information.

Some physical parameters are obtained as function of environmental variables such as soil type, land cover, land use, etc. They can be got from existing tables or sometimes it's possible to calculate them using specific methods.

It's convenient to define a parametric component where legal parameters for each method are maintained as well as their just known corresponding values. For instance, hydraulic conductivity may be calculated from soil type; instead of realizing computations each time this parameter is needed, the parameter value is stored as knowledge in a table related with the right environmental variable. The same thing is done with parameters that are obtained through optimization or through expert experience. Figure 7 shows how each parameter type is associated with a table where legal values and corresponding variables are recorded. Applying the analysis pattern Unit [13], Units can be represented explicitly in the model allowing this way to convert quantities from one unit to another.

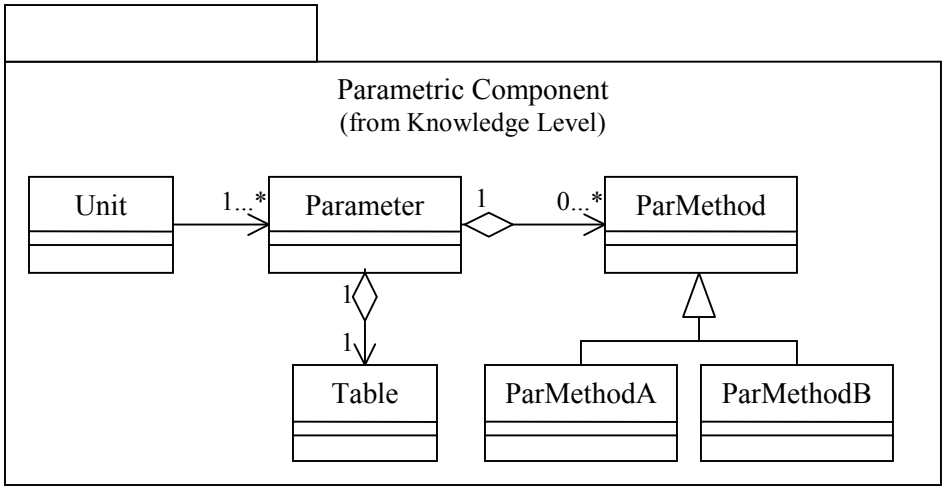


FIGURE 7. Class Diagram for Parametric Component

The class ParMethod is used to maintain the different methods for parameters calculus as it is shown in the next section.

This component must be connected to the components of the Environmental Information Layer, relating parameters with their associated environmental variables for the Table construction.

The Method class belonging to the Simulation Strategies Component is conceptually related with the Parameter class, allowing this way to define all possible legal relationships between methods and parameters and its associated values. This relation is included in the dependency relation between components shown in Figure 5.

This way, it was defined a knowledge level for models simulation strategies and parameters, which will be used by the Structural component for scenarios configuration.

3.5.4 Structural Component.

The Structural Component maintains Model objects, which will be configured with some specific simulation method and parameters. These objects add specific behavior related with flow computing simulation, to objects from Environmental layer, such as river, lake, catchment, etc. which maintains its physical features and real world measurements. They may be simple (lake) or composed objects (cathcment with river, lake, subcatchment, etc.), depending on the selected scenario configuration.
The class ParList allows to maintain the legal parameters list associated with the configured model object during a Simulation session, as it is shown in Fig. 8. They are obtained from the Knowledge level, considering the selected processes and methods for the specific scenario.

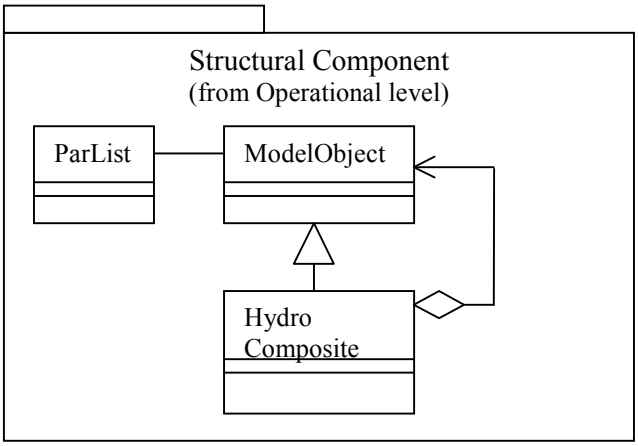


FIGURE 8. Class Diagram for Structural Component

4. DESIGN MODELS

Considering the exposed conceptual microarchitectures, in the present section design microarchitectures based on design patterns are defined, to show the way simulation scenarios are configured and how parameter values are obtained either by calculation or optimization for their associated tables construction.

4.1. Parameter Acquisition

As it was said, there are different ways to obtain parameter values. Once parameters are obtained, either by calculation or by optimization, they can be stored in tables, related with the corresponding environmental variable value. Tables can also be fed with values obtained by organizational experiences.

4.1.1. Parameter calculus.

For parameters which can be calculated as function of environmental variables such as soil type, vegetation type, land use, etc., it is defined a design microarchitecture that brings flexibility to the strategy selection, decoupling computing methods from Parameter class. This microarchitecture is shown in Figure 9 and it's made applying the Strategy design pattern [15]. This pattern defines a family of algorithms, encapsulate each one, and make them interchangeable. The *Method* class declares an interface to all supported calculating algorithms.

Facade design pattern [15] is applied for accessing environmental information subsystems appropriately. This pattern provides a unified interface to a set of interfaces in a subsystem. It defines a higher-level interface that makes the subsystem easier to use. The *FacadeEIL* class delegates Parameter class request to appropriate subsystems. The class named Parameter must know which environmental variable is conceptually associated with the parameter's value to give this information as an argument to facade.

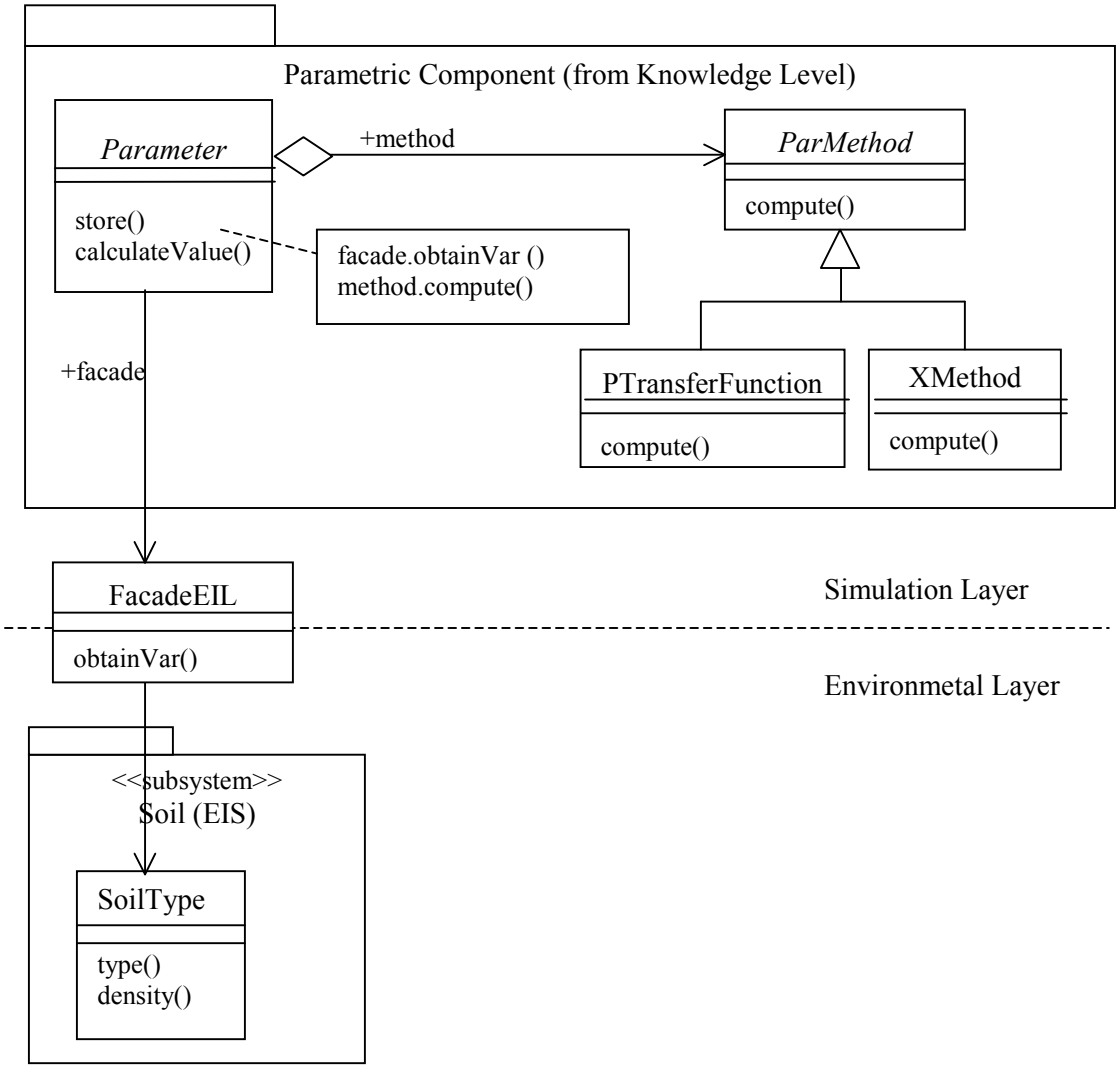


FIGURE 9. Obtaining Environmental Information for Parameters calculus

4.1.2. Parameter Optimization.

Applying the Strategy pattern it's also possible to utilize different strategies for parameter optimization by decoupling that responsibility from the Parameter class. The corresponding design microarchitecture is shown in Figure 10.

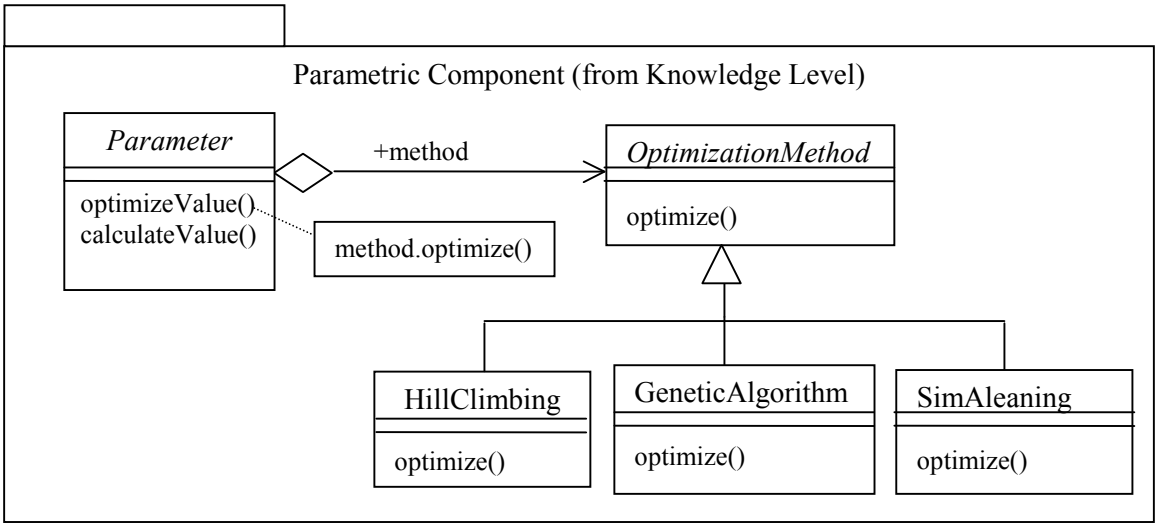


FIGURE 10. Class Diagram for Parameter Optimization

4.2. Simulation Scenarios configuration.

To configure some specific scenario, ModelObject is configured for simulation purposes, adding behavior to an appropriate Environmental object, such as lake, river, etc. Decorator pattern [15] is used to dynamically add responsibilities to objects from Environmental layer, as it can be seen in Figure 11.

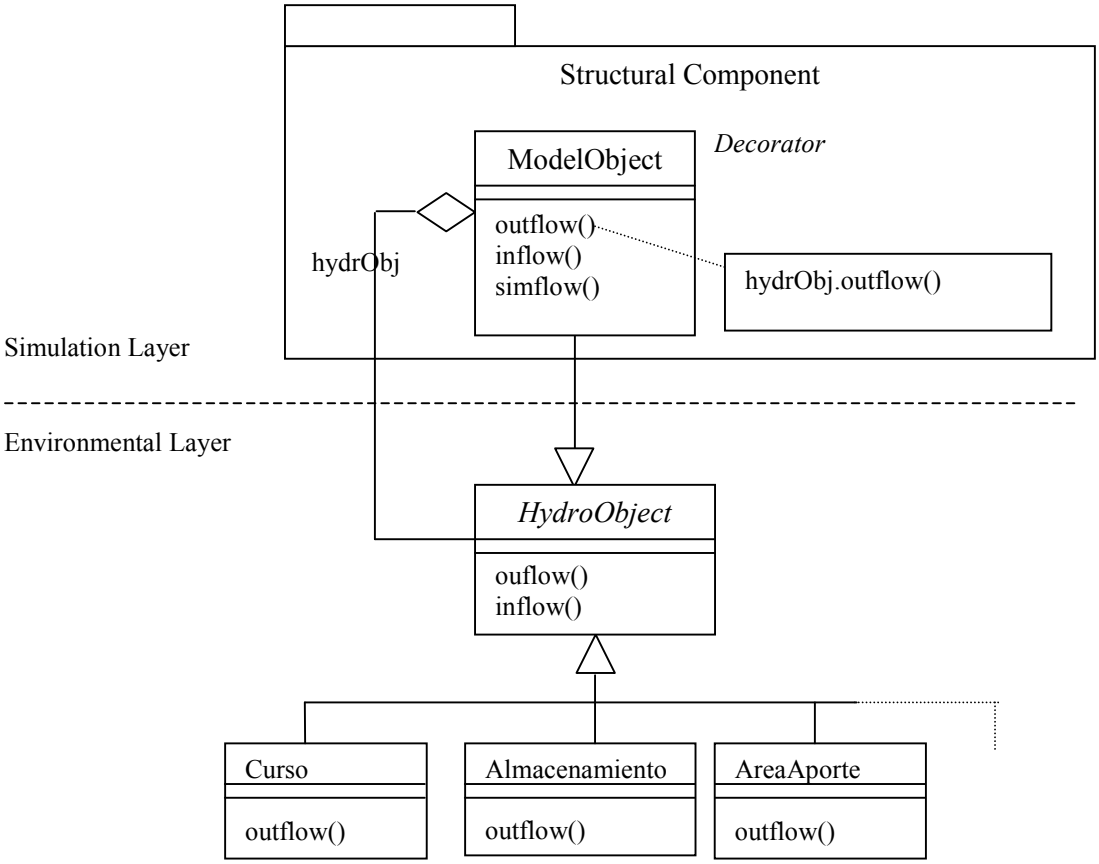


FIGURE 11. Environmental Objects Decoration for Simulation purposes

Strategy pattern is applied for updating the state of hydrological objects with added simulation behavior by some specific Computing Method. This pattern allows to decouple from the structural object the hydrological proccess to be calculated and the corresponding computing method, making them dinamically interchangeable and giving this way more flexibility to the design model. This is shown in Figure 12.

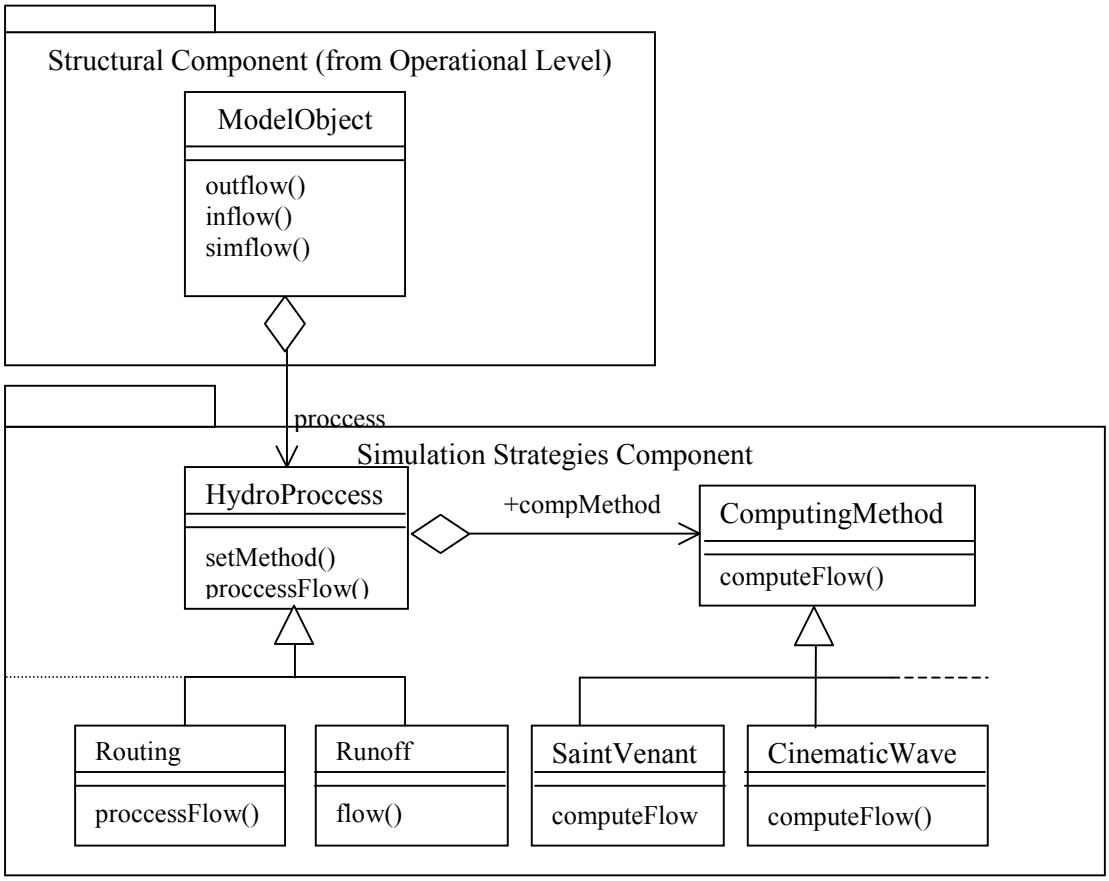


FIGURE 12. Strategy pattern for process flow definition

ComputingMethod Class is responsible of knowing which parameters are neccesary for the model object type proccess simulation and of obtaining the list of legal parameters from the Parametric Component.

Once legal parameters are obtained, ModelObject must know the values of Environmental variables related with each parameter in the list, ie., if a legal parameter in the list is hydraulic conductivity, the object must know the soil type for its geographic location. For that, it's possible to apply the Facade pattern allowing this way Model Object to access the appropriate environmental subsystem, sending object location as an argument and obtaining the desired variable value. This is possible because all environmental objects are represented in the same geographic reference system by means of GeoFrame specialization.

Now, parameters values can be obtained from tables. At execution time, parameters should be stored in a generic class, giving this way more efficience to methods execution in each time step.

5. HOW TO CONFIGURE A SIMULATION SCENARIO USING THE KNOWLEDGE LEVEL

To show the way the exposed models would be used for modeling scenarios configuration, in the present section there is an example of objects collaboration in the following use case realization: "Calculating the runoff process in a catchment using the Clark Method".

The scenario may be configured as:

Environmental Object: Catchment
ModelObject: Decorated Catchment with *Type*: Process: Runoff, Method: Clark

The specific ModelObject (from the Structural Component, Operational layer) would be asked from a Client to return its output flow generated by Runoff process, using Clark Method (Figure 14). The existing association with the corresponding environmental object allows knowing the Catchment's physical and geographic properties.

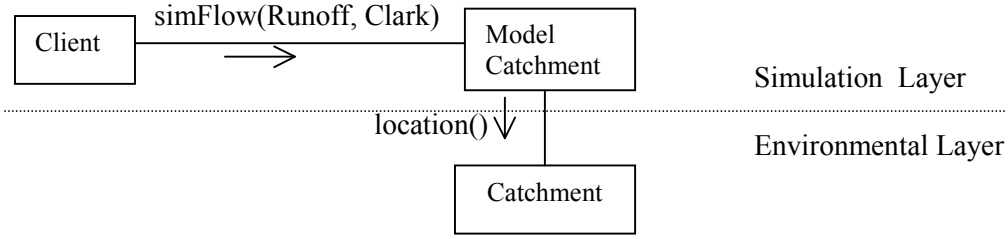


FIGURE 14. Objects collaboration showing the Client request

ModelCatchment will forward the request to the HydroProcess class (Simulation Strategies Component, Knowledge level) to verify that Clark is an appropriate method for Runoff calculus, and then it will delegate this responsibility on the specific Method class, as it can be seen in Figure 15.

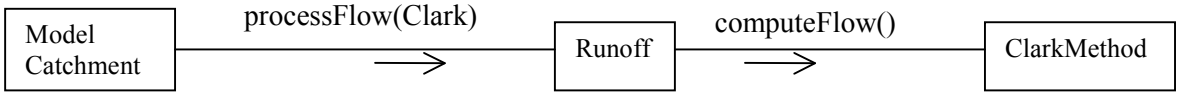


FIGURE 15. Objects collaboration showing the computing process

Before that, it was necessary to configure ModelCatchment with its associated object ParList, obtaining the legal parameters for the corresponding *TypeObject* (Knowledge level), and its related environmental variables obtained from subsystems in the Environmental Layer. This is made applying the Facade pattern, using location as argument.

As it was shown in Figure 5, there is a dependency between the Simulation Strategies Component and the Parametric Component. When calculating, Clark Method will ask the Parameter class for the CN parameter value (which depends on soil type of land use) from the appropriated table, sending the depending variables values for the catchment location as arguments.

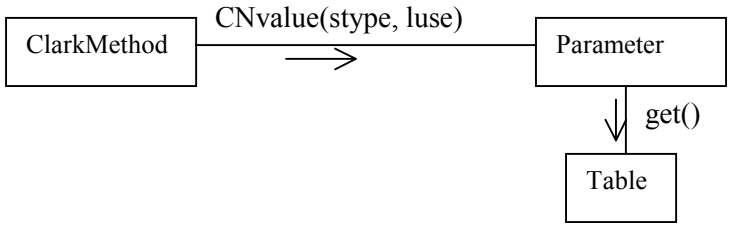


FIGURE 16. Objects collaboration showing parameter values acquisition

The same process is made for the following legal parameters used by the Clark Method in the object ParList.

As it was said, the Parameter class maintains attributes such as the parameter name and its related environmental variables. To construct its associated table, it delegates to FacadeEIL class the responsibility of finding all possible variable values from the Environmental Layer, by means of the statement facade.obtainVar() sending the variable name as an argument. Then, the message calculateValue() is sent to the ParMethod class (Figure 8) using the variable value as an argument, to calculate the appropriate parameter value.

For a given simulation session, at execution time parameter values are added to a generic class for efficient methods execution (Figure 17).

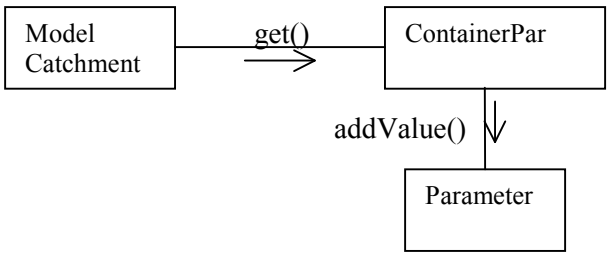


FIGURE 17. Creating a Parameter Container

It can be seen that classes which represent environmental objects (such as catchment, land use, etc.) may be related by their location, because they present geographic behavior by means of the specialization of GeoFrame services (Geographic Representation Layer - Figure 1) in the general application layer**.**

6. AKNOWLEDGEMENTS

The present work was developed in the scope of the "Design Model for Hydrologic models application domain in the EIS context" Project financed by the Facultad de Ingeniería (UNPSJB) and availed by CIUNPAT.
Research results will be applied in the Water Resource Department of Tierra del Fuego Government.

7. CONCLUSIONS

This article explores the potential of flexible software development mechanisms like architecture, patterns and components in the hydrologic modeling domain. It is shown how a layered architecture helps to integrate models and information systems in different abstraction levels. Conceptual microarchitectures for strategies and parametric information facilitate the flexible simulation scenarios configuration. Using components in a simulation layer allows either defining new design models or reusing existing strategic and parametric components. Analysis and design patterns give flexible solutions to the critical question related with parametric information management.
The results of the research will be extended in a further step formally specifying components and interfaces.

7. REFERENCES

[1] Abbot M. *Hydroinformatics: Information Technology and the acuatic environment* Avebury Technical, Adlershot, UK, 1991.

[2] Alfredsen J. *An object oriented framework for application development and integration in hydroinformatics*. Dr. Eng. Thesis, Norwegian University of Science and Technology, 1998.

[3] Appleton B., *Patterns and Software: Essential Concepts and Terminology* .Web Page www.enteract.com/~bradapp/, 2000.

[4] Bass L. and Kazman R. *Architecture Based Development.* Technical Report CMU/SEI 99-TR-007, Carnegie Mellon University, 1999.

[5] Bass L., Clements and Kazman R. *Software Architecture in Practice*, Addison-Wesley, 1997.

[6] Beven, K. *Rainfall-Runoff Modelling*. Wiley, 2000.

[7] Blind M. Adrichem B. *Generic Framework Water: An open modelling system for efficient model linking in integrated water management - current status* Paper presented at the 4th International Eurosim 2001 congress "Shaping Future with Simulation", 2001.

[8] Booch G. Jacobson I., Rumbaugh J. *The Unified Process Software Development.* Addison-Wesley Publications, 1999.

[9] Booch G. Jacobson I., Rumbaugh J. *The Unified Modeling Language* Addison-Wesley Publications, 1998.

[10] Buschmann, F. Meunier, R.; Rohnert, H.; Sommerlad, P. and Stal, M., *Pattern-Oriented Software Architecture: A system of patterns*. New York: John Wiley & Sons, 1996.

[11] Chow Ven Te. *Hidrología Superficial* Addison Wesley, 1997.

[12] Ericsson M. *Developing Large-scale Systems with the Rational Unified Process*. Rational Software White paper, Rational Software Corporation, 2000.

[13] Fowler, M. *Analysis patterns: reusable object models*. Menlo Park: Addison Wesley Longman, 1997.

[14] Fowler, M. *Patterns for things that change with time*. Fowler Web Page, 2001.

[15] Gamma, E. H *Design Patterns. Elements of Reusable OO Software* Addison-Wesley, 1997.

[16] Gordillo, S. Tesis de Magister en Ingeniería de Software, *Modelización de campos continuos en Sistemas de Información Geográfica*, UNLP, 1998.

[17] Günther, O. *Environmental Information Systems*. Springer-Verlag, Berlín, Germany, 1998.

[18] Lisboa J., Iochpe C. *Specifying Analysis Patterns for Geographic Databases on the basis of a conceptual framework*., ACM-GIS '99, Proceedings of the 7th International Symposium on Advances in Geographic Information Systems, November 2-6, 1999, Kansas City, USA, 1999.

[19] Lisboa J., Iochpe C., Beard K. *Applying Analysis Patterns in the GIS Domain* Presented at the 10th Colloquium of the Spatial Information Research Centre, University of Otago, New Zealand, 16-19 November, 1998.

[20] Malan R., Bredemeyer D. *Software Architecture: Central Concerns, Key Decisions.* Architecture Resources Pubs., Bredemeyer Consulting, 2002.

[21] Monroe R., Kompanek D. Melton R. and Garlan D. *Stylized Architecture, Design Patterns, and Objects*, 1996.

[22] Riehle D., Züllighoven H. *Understanding and Using Patterns in Software Development.* Theory and Practice of Object Systems Wiley & Sons, 1996.

[23] Szyperski C., Component Software. Beyond Object-Oriented Programming. Addison-Wesley, 1998.

[24] Urciuolo A., Iturraspe R. *Conceptual Patterns for Water Information Systems.* Journal of Computer Science & Technology, 2003.

[25] Urciuolo A., Iturraspe R., Parson Ariel, Sandoval Sandra *Patrones conceptuales para Sistemas de Información Hídrica.* Proceedings CACIC 2002, Buenos Aires, Octubre de 2002.

[26] Urciuolo A., Iturraspe R. *Microarquitecturas de Diseño OO para Modelos hidrológicos.* Jornadas Universitarias de Informática, Proceedings Congreso del Nuevo Cuyo, San Juan, Octubre de 2001.