The Impact of Object Orientation on System Representation in Hydrology

Olaf David
Friedrich - Schiller - University Jena, Germany

The object orientation seems to be a promising technique in hydrological model design. In particular the potential to reflect hydrological system entities in a more transparent and sophisticated way is determining the value of object orientation. But there is a need for high level general design patterns to express hydrological coherences without providing a detailed implementation. A hydrological skeleton or framework may be useful for further adoption with specific modeling approaches. The focus of this paper is pointed to a design approach for an object oriented modeling library for distributed physically based modeling. Some concepts of considerable aspects of hydrological modeling objects and classes are presented. Several general components of such a library are introduced.

1. Introduction

The usage of an object oriented programming language does not automatically implies a better system design for (hydrological) simulation models. Knowledge about some useful approaches and techniques to arrange objects in an appropriate order helps to manage the system model. This prerequisites proper understanding of basic object-oriented techniques like object finding and expression in terms of the *internal* structure and the useful ways of *external* interaction between objects. In this paper the external relations between object are focus of interest because:

> *"One of the distinguishing features of object design is that no object is an island. All objects stand in relationship to others, on whom they rely for services and control."*
>
> *Beck and Cunningham [1989]*

An object oriented system consists of many interacting objects. While more and more systems are represented by objects, common arrangements have been pointed out. Reusable parts containing objects and classes in a specific arrangement using aggregation, association and inheritance are named *design patterns*. To a design pattern there are belonging a few classes/objects.

In hydrological modeling and simulation the water flow through the natural system is topic of interest coupled with associated and involved system components. One major value of object orientation relies on transparent system representation: a component in the natural system gets mapped into an object with

- structural information and
- behavioral information.

Because objects are not isolated, they are interacting with other objects in their environment.

The hydrological system can therefor be represented as a system of objects interacting with each other to realize the flow of water through the system. But different types of objects are moving water in a different dynamic. The object orientation provides the mechanism of polymorphism with abstracted view of the same process principle.

The paper will introduce the application of a design patterns in application for hydrological problems in simulations and will therefor illustrate the impact of such an approach in model construction.

1.1 What is a Design Pattern ?

The most comprehensive collection and abstraction of object oriented design patters is compiled in Gamma et al. [1995]. There are different scopes and fields of pattern applications in design process. The patterns separation for

- object creation
- structural design of systems and
- system behavior

are quite common. They are applicable at the object level as well as at the class level. Creational patterns concern the process of object creation. The composition of classes and objects is considered in structural patterns and behavioral patters deal with interaction and distribution of

responsibility among objects and classes. Design pattern in that sense are "*descriptions of communicating objects and classes that are customized to solve a general design problem in particular context*" Gamma et al. [1995].

The capability to apply a design pattern is bounded to the object-oriented programming language used for implementation. Than more sophisticated features of object orientation are supported than more elegant a design pattern is applicable.

The evolution of design patters was closely bounded to object oriented systems and applications, mainly frameworks for graphical user interface programming and frameworks like the ET++ (Gamma [1991]). But design patterns are more general and therefor valuable for their application also in hydrological simulation software development.

2. Design Patterns and Hydrology

There are many attempts to classify hydrological models (Flemming [1975], Refsgaard [1996]). Distributed physically-based models are representing the hydrological processes in a catchment more detailed and correct than any other approach (e.g. stochastic, empirical or lumped-conceptual model). But there are still deficits at the level of process description which is necessary to understand a catchment hydrology in three dimensions. In this context the need for modeling framework support for hydrological model development can be positioned. Design patterns may help to construct a conceptual model "skeleton", which is

- easy to understand because it reflects the objects to describe in a transparent way
- extensible due to potential "plug-in slots" for new approaches in physical process description
- separating several aspects related to model components.

Distributed physically-based models are describing the catchment in quasi three dimensions, where two major separations are reflected in the model structure:

- There is a regular or irregular two dimensional spatial disaggregation of the catchment into smaller units. This may be a result of a GIS analysis of the catchment. A raster or vector data model can be used to delineate these smaller catchment units. The stream network may belong to these units as well as land units located in direct or indirect stream neighborhood. The definition rules for delineating the units depend on the model approach.
- For each unit there are different layer models for each involved component in the third dimension. For example if the spatial unit may consider for example an interception, evaporation or infiltration model which operates with the specific properties of that spatial unit.

The entire distributed hydrological model run may consists of the proper routing of all local hydrology budgets computed within each units which finally represents the total catchment hydrology.

There are some aspects involved in such an abstraction process. There should be components for representing the units, for processes associated to components and data objects delivered by the hydrological processes. There should be also a routing scheme for lateral data flow at the unit level.

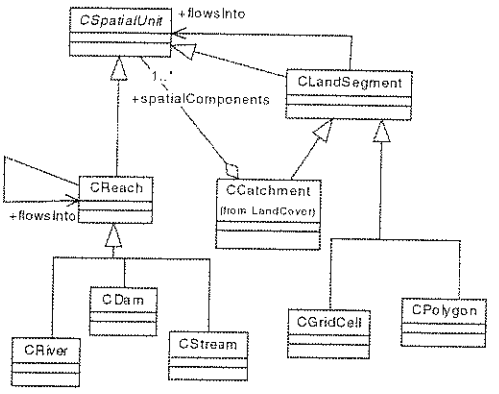

Figure 1: Spatial Component Relations

In Figure 1 the relations between the spatial components is shown in part of an UML class diagram (*Unified Modeling Language*, Rational [1997]). Base class is "CSpatialUnit" which represents any spatial distributed entity. Sub classes are the "CLandSegment" class and the "CReach" class. Both are top level classes for stream network classes (e.g. "CStream") and spatial entities which may be regular shaped as a grid cell or irregular shaped as a polygon. A "CCatchment" is a special spatial unit aggregating a set of different spatial components which is pointed out as a role. The routing is expressed by an unidirectional association from "CLandSegment" to "CSpatialUnit". This enables an discharge from an object of a sub class of "CLandSegment" to another object of a subtype of

"CSpatialUnit". Because "CSpatialUnit" is more general the destination for discharge may also be a "CStream" object. If the water resides in a stream it can only go downstream. The association in "CReach" ensures that behavior.

In object oriented design such a separation of these concerns will result in a set of *classes*. Logically together belonging classes are aggregated into *packages*.

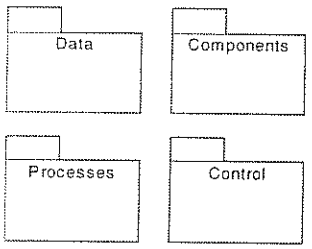

Figure 2: Hydrological Modeling Packages

In Figure 2 major packages are also shown in an UML class diagram:

- a *data* package contains classes representing physical data objects like temperature, depth, solar radiation etc. A model is operating with higher level data objects considering the physical unit belonging to a certain value or set of values.
- a *process* package encapsulates hydrological processes to model. These models reside in a abstract and concrete specific implementation in that package. Process classes are operating with data objects generated from the data classes and are delivering also typed data objects as the result of a process. For example a interception process returns a interception data object witch is an amount of water.
- a *component* package contains classes for natural and conceptual components relevant for the hydrological dynamics in the catchment. There are for example classes for vegetation, soil horizons, ground water store, snow layer, etc.
- classes residing in the *control* package take care about the proper routing of water trough the units. This finally result in the linkage management of all spatial units.

In Figure 3 some example classes from the process package describing the sub surface flow are shown. The most general process class is represented by "PProcess". The sub class "PSubSurfaceFlow" is an abstract class, means that it is not suited for object creation because the class is too general.

The "PHydro" is a interface which is an object oriented construct similar to a class but is containing only not implemented operation interfaces. Other classes implementing that interface have to provide a implementation body for each method. In the "PHydro" interface there is one operation named "flowWater()". In all sub classes of PSubSurfaceFlow this operation has to be implemented according to the approach used in that class.

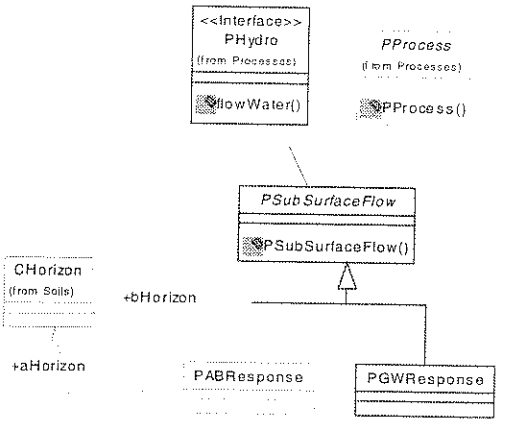

Figure 3: SubSurface Classes Diagram

The "PABResponse" is a class describing the process of water movement between the A and B soil horizon. The class encapsulates two associations to the "CHorizon" class residing in the component package.

2.1. Iterator

The *iterator pattern* provides a way to access the elements of an aggregation of objects without exposing its underlying representation [Gamma 1995]. In hydrological application using a general "CLandSegment" as a spatial unit the water movement through the system (vegetation, surface, soil horizons,...) can easily be expressed by an iterator which is using the top level interface class "PHydo".

Assume the "CLandSegment" is configured and parameterized with a set of objects representing

- the vertical components that the land segment is built up (specific vegetation, soil layer with soil type and depth, ...)
- and the hydrological processes associated with these components (interception, infiltration, surface flow, ...),

the final invocation of the model run in terms of moving water thought the simulated system of

(components) objects consists only of few lines of code using a iterator pattern This example is given in the JAVA programming language, which is in case of control statements syntax similar to C/C++:

```
for(CComponent c = getTopComp();
      c != null;
      c = c.getNextComp()) {
  for(PHydro p = c.getPHydro();
        p != null;
        p = p.getNextPHydro()) {
    p.flowWater();
  }
}
```

a)

```
ProcessIterator pi =
   getTopComp().getIterator();
while(pi.hasElements()) {
  PHydro p = pi.next();
  p.flowWater();
}
```

b)

Figure 4: Model Process Loops

The model loop (Figure 4a) iterates thought all components, starting with the top component of a spatial unit (usually the canopy layer). For each component the associated hydrological processed are extracted and the nested loop iterates through all processes. The only operation in iteration which gets called is the `flowWater()` operation bounded to an actual object of the process list. Because every process is implementing this method in its own specific way the object itself ensures the proper behavior. Polymorphism is the underlying object oriented mechanism.

A more sophisticated solution using the iterator design pattern is shown in Figure 4b. The control package contains a class "ProcessIterator" which is able to iterate trough the process list of an spatial unit.
An Iterator is a powerful concept as a behavioral design pattern. It is not only applicable for a sequential walk through a list, also the routing and movement of water at the level of the spatial units is an iteration.

3. Conclusions

The object orientation in general is a powerful concept for system abstraction in hydrology. Separating several modeling aspects into a object class library components results in a more reusable set of classes where models can build from. Object orientation alone does not guarantee a proper system representation in a model, also high level general design techniques are necessarily needed. The value of such an approach is delegation and encapsulation of responsibility. If a hydrological process class is implementing a specific algorithm for water movement only this class takes care of it. But the interface for invoking such a functionality is slim.

Design patterns are helpful to express a common applicable behavior and structure for a model. The iterator concept as one example for a behavioral pattern illustrates the power of abstraction which simplifies the top down view in a model.

4. Acknowledgements

For the discussions about the object oriented system design many thanks to Greg Kiker from the University of Natal at Pietermaritzburg.

5. References

Beck K. and W. Cunningham, A Laboratory for Teaching Object-Oriented Thinking, OOPSLA, Special issue of SIGPLAN Notices, 24(10), *1-6*, 1989.

Flemming, G., Computer Simulation Techniques in Hydrology. Elsevier, 1975.

Gamma, R. , Helm R., Johnson R. and Vlissides J., Design Patterns – Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995.

Rational Corp. The Unified Modeling Language 1.1, Rational Corporation, 1997.

Refsgaard, J.C., Terminology, Modelling Protocol and Classification of Hydrological Model Codes, M.B. Abbot and J.C. Refsgaard (eds.), Distributed Hydrological Modelling, *17-39*, Kluwer, 1996.

Kirkby M.J., Naden P.S., Burt T.P. and D.P. Butcher, Computer Simulation in Physical Geography, 2nd Ed., Wiley, 1993