Microarquitecturas de Diseño OO para la Configuración de Escenarios de Simulación Hidrológica

Urciuolo Adriana, Iturraspe Rodolfo, Villarreal Martín
Universidad Nacional de la Patagonia San Juan Bosco – Sede Ushuaia –Darwin y Canga (9410) Ushuaia
e-mail: urciuolo@tdfuego.com, iturraspe@tdfuego.com, martinvillarreal@hotmail.com

Resumen

Los modelos hidrológicos constituyen una herramienta de predicción fundamental para tomadores de decisión ambiental. Un *modelo hidrológico* es una representación simplificada del sistema real cuyo objetivo es estudiar la operación del mismo y predecir su salida a partir de ecuaciones matemáticas que conectan entradas con salidas. Estos modelos simulan procesos ambientales del mundo real utilizando distintas estrategias para el cálculo de la transformación lluvia-caudal, mediante distintos métodos. El software del dominio provee gran diversidad de modelos hidrológicos, los cuales han ido evolucionando en sus prestaciones a través de cinco generaciones de modelos. No obstante su grado de evolución, la mayoría de ellos son estáticos, ofreciendo poca flexibilidad para la selección de la estrategia de cálculo y la construcción del sistema a simular.
Se plantea en consecuencia, la necesidad de proveer modelos de diseño apropiados para la construcción flexible de escenarios de simulación hidrológica.
En el presente trabajo se definen microarquitecturas de diseño basadas en patrones para el desarrollo de software flexible del dominio, en el marco de una arquitectura conceptual existente para modelación hidrológica integrada a Sistemas de Información Ambiental. Se muestra el uso de las mismas para la configuración de un escenario de simulación en una cuenca específica.

Palabras clave: patrones, diseño, microarquitectura, modelo hidrológico.

Introducción

Debido a las limitaciones en las técnicas de medición de fenómenos del mundo real, modelos de diferentes tipos proveen medios de extrapolación cuantitativa o predicción, que permiten simular estados de un sistema hidrológico real cuando no hay datos disponibles en el espacio o en el tiempo y conocer el impacto de futuros cambios hidrológicos, ayudando de este modo a los tomadores de decisión (Beven K, 2000).

Un *modelo hidrológico* es una representación simplificada del sistema real cuyo objetivo es estudiar la operación del sistema y predecir su salida. Sus entradas y salidas son variables hidrológicas mensurables y su estructura es un conjunto de ecuaciones que conectan las entradas con las salidas, las cuales pueden expresarse como función del tiempo (Chow, 1997). Abarcan una gran diversidad de problemas y funcionalidades tales como modelado de ríos y cuencas, calidad de aguas, predicción de crecidas, riesgos hidrológicos, etc.

Los modelos computacionales aplicables a este dominio sufrieron un proceso de cambios y evolución, determinado por los avances en la computación, a los cuales se fueron adaptando. Se distinguen cinco generaciones de modelos (Abbot, 1991), que fueron incorporando ambientes de trabajo más amigables a partir de la cuarta generación, a los fines de poder ser utilizados no sólo por especialistas calificados, sino por los tomadores de decisión en organizaciones de manejo ambiental.

Si bien existe una gran diversidad de modelos hidrológicos, en este trabajo el análisis se focaliza en los modelos lluvia-escorrentía, los cuales estiman el caudal a la salida de un sistema hidrológico (cuenca, área de aporte, lago, etc.) a partir de la precipitación y otras variables meteorológicas.

El software del dominio provee una gran diversidad de modelos basados en diferentes métodos de cálculo para cada uno de los procesos hidrológicos a simular: infiltración, escurrimiento superficial, propagación de caudal, etc. Los más modernos ofrecen facilidades para su integración a SIG (Sistemas de Información Geográfica) a través de interfaces (Maidment, 1996). No obstante, en general los modelos existentes ofrecen poca flexibilidad para la definición del escenario de simulación a ejecutar (Urciuolo et al, 2003). La mayoría de ellos presentan características estáticas, exigiendo al usuario definir de antemano los procesos, métodos y parámetros que serán utilizados en la sesión de simulación. De acuerdo a esto, una necesidad que se observa en esta etapa de la evolución del software del dominio es la de proveer mayor flexibilidad a los modelos para la configuración de escenarios de simulación, permitiendo a los usuarios la selección de los procesos hidrológicos y métodos de cálculo en forma dinámica, a partir de la información existente en un Sistema de Información Hídrica, así como la libre construcción del sistema a simular. Se plantea en consecuencia, la necesidad de proveer modelos de diseño apropiados para la construcción flexible de escenarios de simulación hidrológica.

Los patrones han sido utilizados en la Ingeniería de Software para permitir el reuso de soluciones que han probado ser exitosas para problemas recurrentes, en las diferentes etapas del proceso de desarrollo de software. Para un estilo arquitectural dado puede existir un conjunto de *patrones de diseño* que actúan como "microarquitecturas" (Monroe et al, 1996).
Los patrones de diseño proveen un esquema para refinar subsistemas o componentes de un sistema de software o las relaciones entre ellos. Describen estructuras comúnmente recurrentes de componentes comunicantes que resuelven un problema general de diseño dentro de un contexto particular (Gamma et al, 1995). El fuerte uso de mecanismos como composición y delegación de los patrones de diseño permiten el desarrollo de software de gran flexibilidad.

De acuerdo a lo expuesto, el objetivo del presente trabajo consiste en el desarrollo de microarquitecturas de diseño basadas en patrones para el componente estructural de un sistema de modelación hidrológica, que permita la configuración flexible de escenarios de simulación en el marco de una arquitectura conceptual apropiada para el desarrollo de los mismos.

Microarquitecturas de Diseño

La arquitectura de software resulta en nuestros días una parte crucial del proceso de diseño (Bass et al, 1998). Hoy se reconoce como esencial el contar con una representación arquitectural del sistema, para el análisis y la descripción de las propiedades de alto nivel. Pueden definirse diferentes vistas de la arquitectura de un sistema. Algunos autores (Booch et al, 1999) plantean que la vista arquitectural lógica de un sistema puede especificarse por diagramas de packages y de clases que permiten direccionar los requerimientos funcionales del sistema. Dada una arquitectura lógica para sistemas de un dominio, es posible definir **microarquitecturas** para los diferentes packages que componen la vista arquitectural del mismo, orientadas a resolver problemas específicos de diseño propios del servicio a proveer por cada package, las cuales puedan ser reusadas en diferentes aplicaciones.

Los **patrones de diseño** constituyen conjuntos recurrentes de relaciones entre clases, que definen soluciones apropiadas a problemas comunes de diseño orientado a objetos (Appletom, 2000).

Mediante la descripción formal de estas soluciones y sus relaciones, es posible capturar correctamente el conocimiento definido por nuestra comprensión acerca de arquitecturas adecuadas para los problemas planteados. Dado que los patrones pueden ser reusados mediante su adaptación al dominio de una aplicación, algunos autores los llaman *microarquitecturas reusables* (O'Callaghan et al, 2001).
Por lo tanto mediante un conjunto de patrones de diseño pueden definirse las microarquitecturas apropiadas para resolver problemas de un dominio, tal como el de modelación de simulación hidrológica. El uso de patrones garantiza la flexibilidad esperada.

.

Requerimientos candidatos

El modelo de microarquitecturas de diseño para la capa de simulación se construye partiendo de la siguiente *lista de requerimientos candidatos* para los modelos de diseño a construir.

- *Definición de una capa de simulación* que presente comportamiento específico del modelo a utilizar, considerando la información disponible en la capa de información ambiental.
- *Estrategia de modelación hidrológica para los componentes del Sistema de Información*. Debe existir la posibilidad de adicionar en forma dinámica a los diferentes componentes hidrológicos *métodos de cálculo apropiados para el escenario de simulación*. Se plantea la posibilidad de modelación de diferentes procesos: evapotranspiración, routing, etc., utilizando diferentes estrategias de simulación tanto para componentes individuales como compuestos.
- *Construcción del sistema hídrico a modelar*. Se plantea la necesidad de contar con una arquitectura que permita simular diferentes procesos para cualquier componente simple (tramo de río, lago, área de aporte) en forma particular o de construir flexiblemente un "sistema hídrico" (cuenca, etc.), en base a diferentes componentes runoff-routing.

Se utiliza notación UML (Booch, 1998) para los diagramas de clases y lenguaje Java para la implementación de los modelos de diseño.

Arquitectura conceptual

A los fines de utilizar una arquitectura para los modelos de simulación hidrológica que permita su integración a los Sistemas de Información ambiental de manera flexible, se utiliza una arquitectura layers para Modelación hidrológica integrada a Sistemas de Información Ambiental (Urciuolo *et al*, 2002, 2003), la cual consta de tres capas: una capa de Representación Geográfica, una de Información Ambiental y una capa de Simulación.

A partir de la Arquitectura conceptual se identifican los principales subsistemas dentro de cada capa involucrados en el problema a resolver, tal como se muestra en la Fig.1.
Dentro de la Capa de Simulación, se identifica el Subsistema Modelación Hidrológica, que incluye componentes estructurales, de estrategias de simulación, paramétricos y de datos.
En la Capa de Información ambiental, se identifica el Subsistema de Información Hídrica (SIH), el cual consta de componentes tales como Manejo del Inventario Hídrico, Tratamiento estadístico de datos, Manejo de Usos, etc.
En la Capa de Representación Geográfica el Framework conceptual GeoFrame (Lisboa et al, 2000) permite que mediante su especialización, los objetos de las capas superiores adquieran comportamiento geográfico (Gordillo et al, 1998), es decir la posibilidad de ser representados en algún sistema.

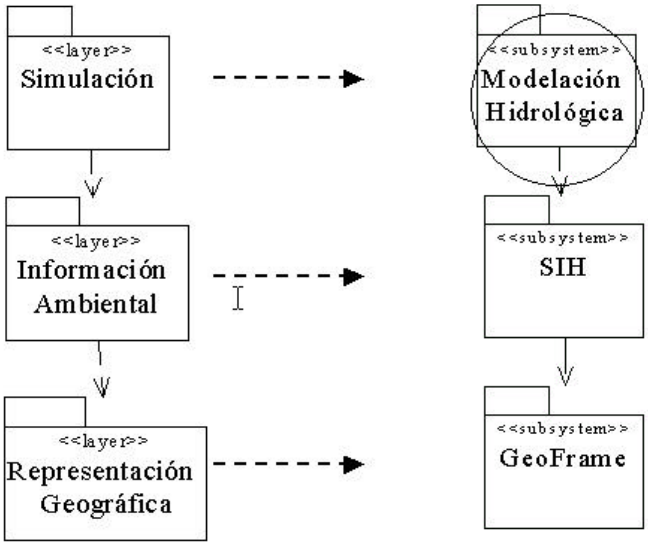

Fig. 1. Identificación de Subsistemas de interés en las Capas de la Arquitectura Conceptual

Se resalta el Subsistema Modelación Hidrológica de la capa de simulación, a los fines de analizar sus interacciones y definir microarquitecturas para el mismo a partir de información mantenida en el SIH, en base a la aplicación de patrones de diseño.

Microarquitecturas de diseño para Sistemas de Modelado de Simulación Hidrológica

Definición de una capa de simulación a partir de la capa de SIH

De acuerdo a la arquitectura conceptual utilizada, a los fines de no sobredimensionar las clases de la capa de Información Hídrica con métodos propios del escenario de simulación, se define una capa de simulación, con comportamiento específico del proceso a modelar. En esta capa, los objetos utilizados tienen comportamiento propio del escenario, agregado al comportamiento de los objetos del sistema de información, para los cuales se utiliza una adaptación de la jerarquía propuesta por Alfredsen (Alfredsen, 2000).

El Patrón de diseño Decorator (Gamma et al, 1995) puede usarse como alternativa flexible a la subclasificación, a los fines de adicionar responsabilidades en forma dinámica y transparente a objetos individuales.

El Patrón se aplica de la siguiente forma:

- CompHidro define una interface para los objetos de la capa de información ambiental que pueden tener responsabilidades adicionadas en forma dinámica.
- CompModelo mantiene una referencia a un objeto CompHidro y define una interface que replica la de CompHidro. Tiene comportamiento específico (qSimulado()).
- Los distintos componentes hidrológicos: Curso, Lago, etc. definen los objetos a los cuales pueden adicionarse responsabilidades.

En la Fig. 2 se muestra el Diagrama de Clases correspondiente a la aplicación de este patrón.

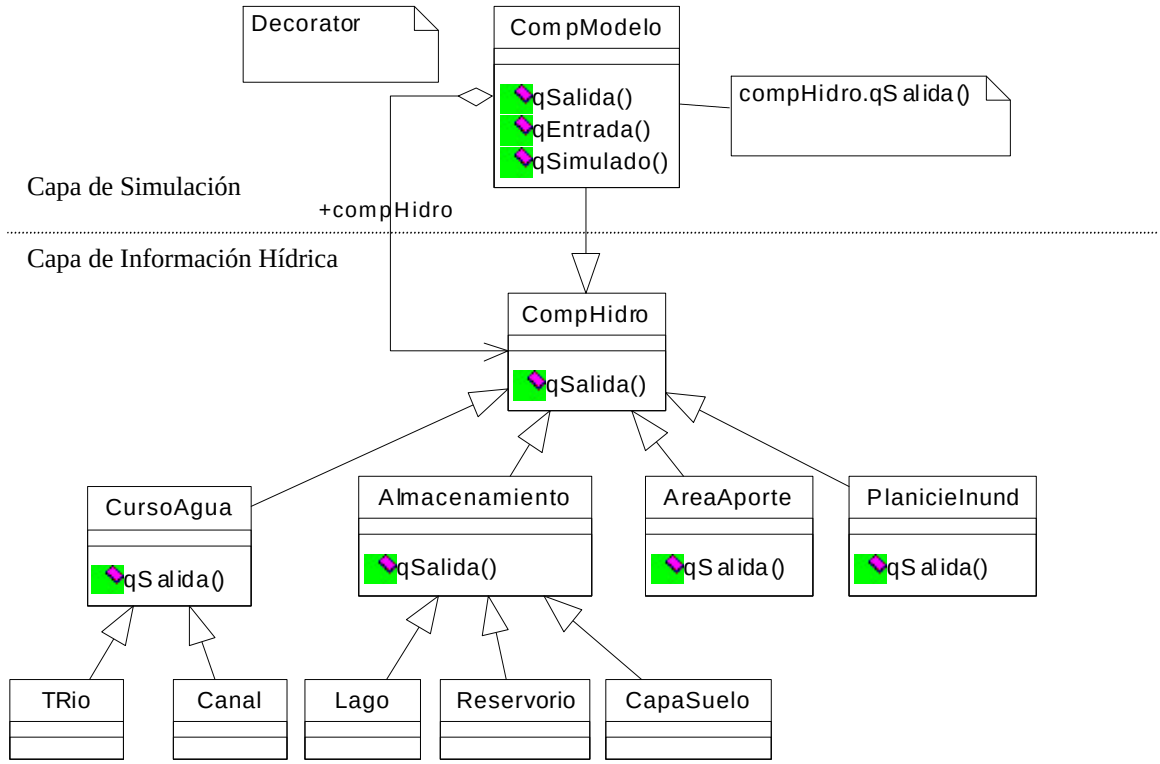

Fig 2. Decoración de Objetos Hidrológicos del SIH

Para una Interface de la Capa de Información tal como la siguiente:

```
abstract class CompHidro
{
    /…
    public abstract void input(SerieTiempo s);
    public abstract SerieTiempo qSalida(); // caudal obtenido de mediciones
}
```

se define una subclase que le agrega comportamiento específico de Simulación (Decorator):

```
public class CompModelo extends CompHidro
{
   CompHidro comph;
   HashMap series=new HashMap();
   DataContainer Da
   //…….
          public CompModelo(CompHidro H) {comph=H; }// se construye con un compHidro de algún tipo
   //.....
   public void input(String a,SerieTiempo ST){series.put(a.toString(),ST);}
   public SerieTiempo qSalida(){compHidro.qSalida());} // utiliza el método del compHidro concreto que se decora
   public void setTimeStep(){} // se adiciona comportamiento propio de la simulación
}
```

Estrategia de Modelación Hidrológica

Tal como ya se ha expuesto, la conversión del exceso de lluvia en escorrentía superficial, puede simularse por medio de distintas estrategias.

Una condición básica de flexibilidad es la de permitir definir para cada componente, diferentes métodos de cálculo que sean intercambiables.
Para cumplir esta condición, se utiliza el patrón de diseño Strategy (Gamma, 1995), que "define una familia de algoritmos, encapsula cada uno y los hace intercambiables". El estado de los objetos de la cuenca hidrográfica se actualiza mediante un método de cálculo o un modelo, que define el comportamiento hidrológico del objeto.
A los fines de incorporar la estrategia de cálculo del caudal, se aplica el patrón Strategy; como ejemplo se muestra que existen distintos métodos de propagación:

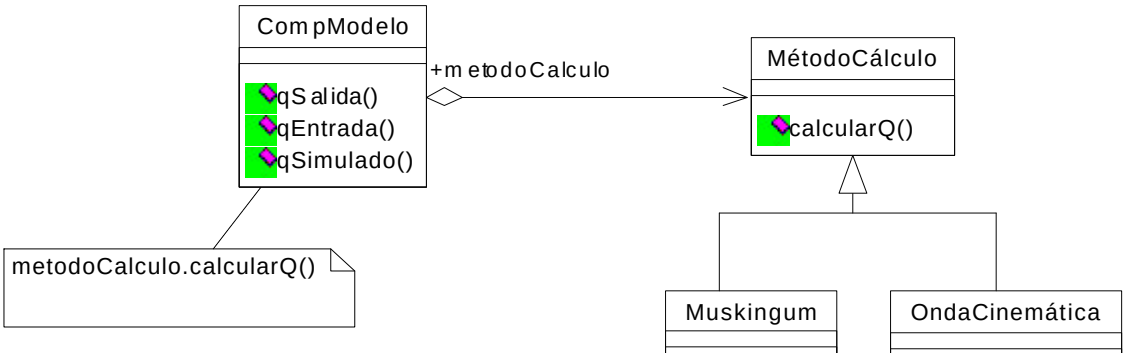

Fig 3. Selección flexible de la Estrategia de cálculo

- Método de cálculo (Strategy) define una interface común para todos los algoritmos que soporta. Puede ser de utilidad tanto para el cálculo de la propagación, como de cualquier otro proceso a modelar.
- Cada uno de los métodos concretos definidos implementa un algoritmo de cálculo de Q (caudal), utilizando la interface definida en MétodoCálculo.
- CompModelo (componente del nivel de simulación):
  - Se configura con una estrategia concreta de modelado
  - Mantiene una referencia al objeto Strategy asociado
  - Puede definir una interface que le permite a Strategy acceder a sus datos.

```
public class CompModelo extends CompHidro
{
   CompHidro comph;
   MetodoCalculo m;  //referencia a la clase Strategy

          public CompModelo(CompHidro H, MetodoCalculo N) // se construye con un método de cálculo
          {comph=H;
            m=N;
          }
   //.....
   public SerieTiempo qSimulado(){return m.metodoCalc(this);} // se calcula el caudal simulado utilizando Strategy
}
```

Se define una interface para todos los métodos de cálculo, que puede implementarse para cada método en particular:

```
public interface MetodoCalculo
{
          public SerieTiempo metodoCalculo(CompModelo componente);
}
```

Para instanciar un Componente del Modelo, se le pasa como argumento el método de cálculo concreto a ser utilizado.

Composición del Sistema Hídrico a modelar

A los fines de modelar el mundo real de manera flexible, es necesario definir diferentes alternativas de construcción del Sistema hídrico a modelar. Debe ser posible modelar tanto el comportamiento de un componente simple, por ej., calcular la propagación de agua en un tramo del río o el escurrimiento de un área de aporte, el caudal a la salida de un componente compuesto runoff-routing (elemento del modelo), así como el de un sistema hidrológico compuesto según diferentes alternativas seleccionadas (cuenca). En el último caso, se debe considerar el Transporte de fluido entre componentes, es decir que cada elemento del sistema envía un flujo de agua al siguiente.

1) Elemento Runoff-Routing del Modelo (Subcuenca o Celda)

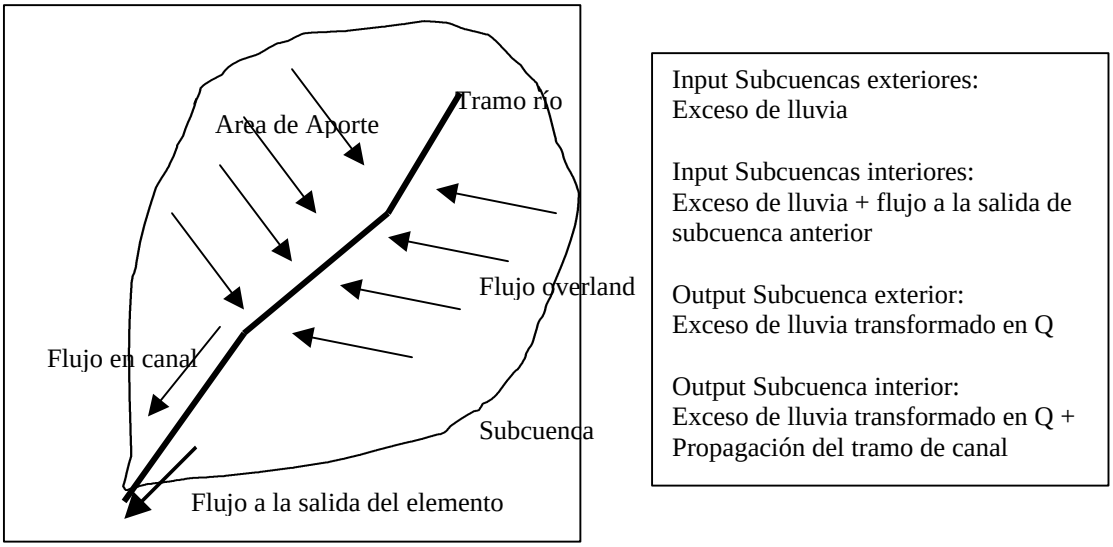

Fig 4. Esquema de una Subcuenca – Elemento Runoff-Routing

Desde el punto de vista de su comportamiento, la subcuenca contiene dos tipos de componentes a modelar: componente routing (propagación) como es el caso del canal principal de la subcuenca y componente runoff (escorrentía) como es el área de aporte (Fig. 4). Un componente de tipo runoff transforma la lluvia caída en la subcuenca en escorrentía, computándose la descarga de cada elemento a la red hídrica. El componente routing toma la descarga del elemento anterior (aguas arriba) y realiza la propagación de este caudal hacia el elemento ubicado aguas abajo.

2) Sistema a modelar (Cuenca)

El input de exceso de lluvia a cada subcuenca exterior se transforma en el correspondiente output, que es el input al canal de la próxima celda. Este caudal de entrada se propaga a través del canal y el exceso de lluvia sobre la celda interior se transforma en flujo, formando el output total a la salida de la subcuenca interior que es el input de la siguiente.
La Fig.5 muestra un Sistema hidrológico compuesto de varias subcuencas; las flechas indican la dirección del flujo desde las subcuencas superiores hacia las inferiores, que se va propagando hasta la salida de la cuenca.

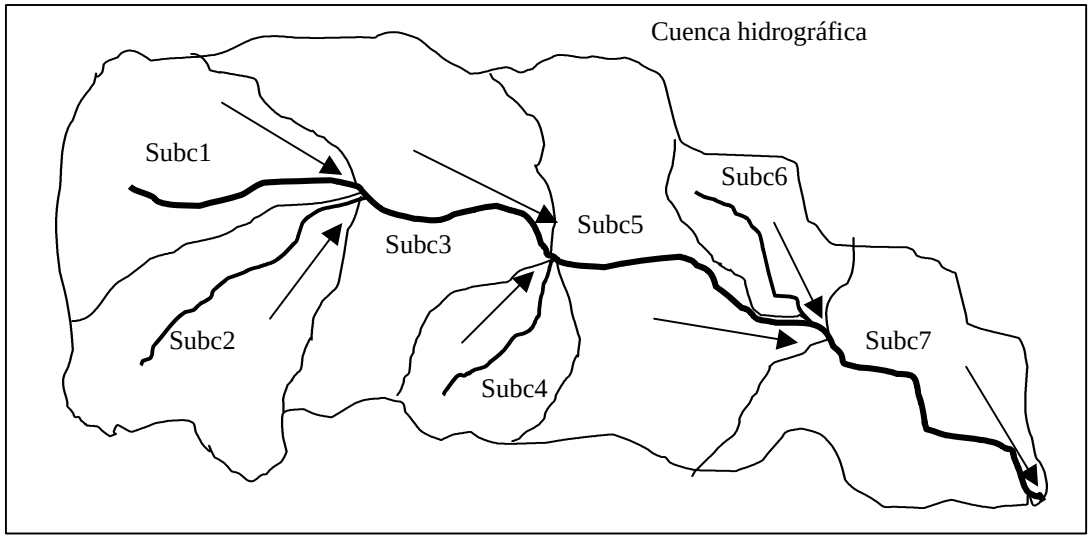

Fig 5. Esquema de un Sistema Hidrológico - Cuenca

De acuerdo a lo expuesto, deben existir diferentes alternativas de composición del sistema, según el tipo de modelo que se utilice. Es decir que debe ser posible definir el sistema a partir de la agregación de componentes: tramos de río, áreas de aporte, lagos, etc. o a partir de la agregación de *elementos del modelo* (componentes runoff-routing), ya sean subcuencas o celdas que forman una grilla.

Se utiliza el patrón de diseño Composite (Gamma et al, 1995), que permite componer objetos en estructuras de árbol para representar jerarquías todo-parte. De esta forma, se puede definir jerarquías de clases como la que se muestra en la Figura 6, con objetos primitivos y objetos compuestos por distintas combinaciones de los primitivos.

Las clases Curso, Almacenamiento, etc., definen componentes hidrológicos primitivos, cada una con su propio comportamiento. La clase HidroComposite define una agregación de componentes hidrológicos; los objetos de esta clase pueden componer otros HidroComposite recursivamente. Pueden agregarse muy fácilmente componentes a la jerarquía, tanto simples como compuestos, sin necesidad de alterar los objetos clientes de la misma, que tratan uniformemente a unos y otros.

En la Fig. 6. se muestra el diseño para la definición del sistema a modelar a partir de componentes previamente decorados. Se tiene en cuenta que cuando se utilizan en forma conjunta Decorator y Composite, normalmente ambos comparten una interface común (Gamma et al, 1995). La clase HidroComposite se define de la siguiente manera:

```
public class HidroComposite extends CompHidro
{   private ArrayList lista = new ArrayList(); // se utiliza una estructura simple como ejemplo
    public HidroComposite(){}
    //…
   public void input(SerieTiempo s){};
   public SerieTiempo qSimulado(){return new SerieTiempo();}
   public void add(CompHidro c){lista.add(c);} // permite agregar componentes hidrológicos al sistema
   public void remove(CompHidro c){}
}
```

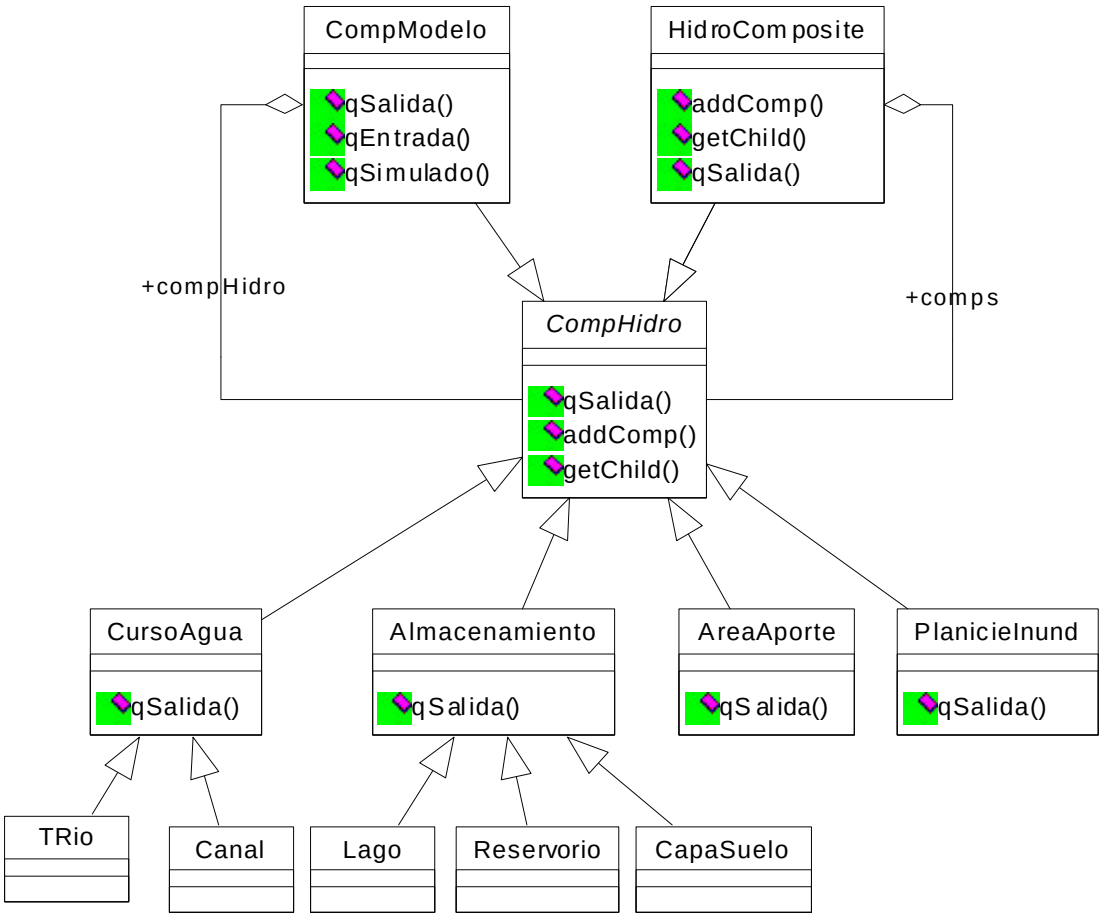

Fig. 6. Composición del Sistema Hídrológico a modelar

Una vez que se cuenta con una clase HidroComposite, se pueden definir nuevas clases que permitan representar sistemas compuestos a modelar.

1) Elemento del modelo
Un Cliente puede utilizar el diseño propuesto para construir el sistema a modelar, creando previamente los elementos compuestos runoff-routing.
La clase ElementoModelo permite crear la unidad física runoff-routing de los modelos. Se compone a partir de componentes hidrológicos (CompModelo) que generen runoff y routing. El caudal de salida simulado de ElementoModelo se computa a partir de la salida del componente runoff y la resultante de propagar sus componentes routing.

```
public class ElementoModelo extends HidroComposite
{
        boolean interior;  // indica si el elemento es exterior o interior
        public ElementoModelo()
        {}
        public boolean interior(){return interior;}
        public serieTiempo qSimulado(); // computa la salida  total del elemento = runoff+routing
}
```

2) Sistema a modelar (Cuenca)
La clase SistemaModelo almacena la topología del sistema completo a modelar y controla el traspaso de agua entre componentes:

```
public class SistemaHidro extends HidroComposite
{
        private SerieTiempo salida=new SerieTiempo();
        public SistemaHidro(){}

        public SerieTiempo qSimulado() // calcula el caudal a la salida del sistema completo
        {
        Iterator e=lista.iterator();
        ElementoModelo elem;
        while (e.hasNext())
        {
         elem=(ElementoModelo)e.next();
         if (elem.interior()==true) elem.input(salida); // traspaso de agua entre elementos
         salida=elem.output();
        }
        return salida;}
}
```

Ejemplo de utilización de las Microarquitecturas de diseño

A los fines de mostrar la utilización de las microarquitecturas definidas, se realiza un ejemplo de su implementación en lenguaje Java para la configuración del sistema real compuesto por dos subcuencas que se muestra en la Fig.7. El ejemplo se simplifica depreciando los cursos de menor orden, considerando sólo un tramo de río principal en cada subcuenca.

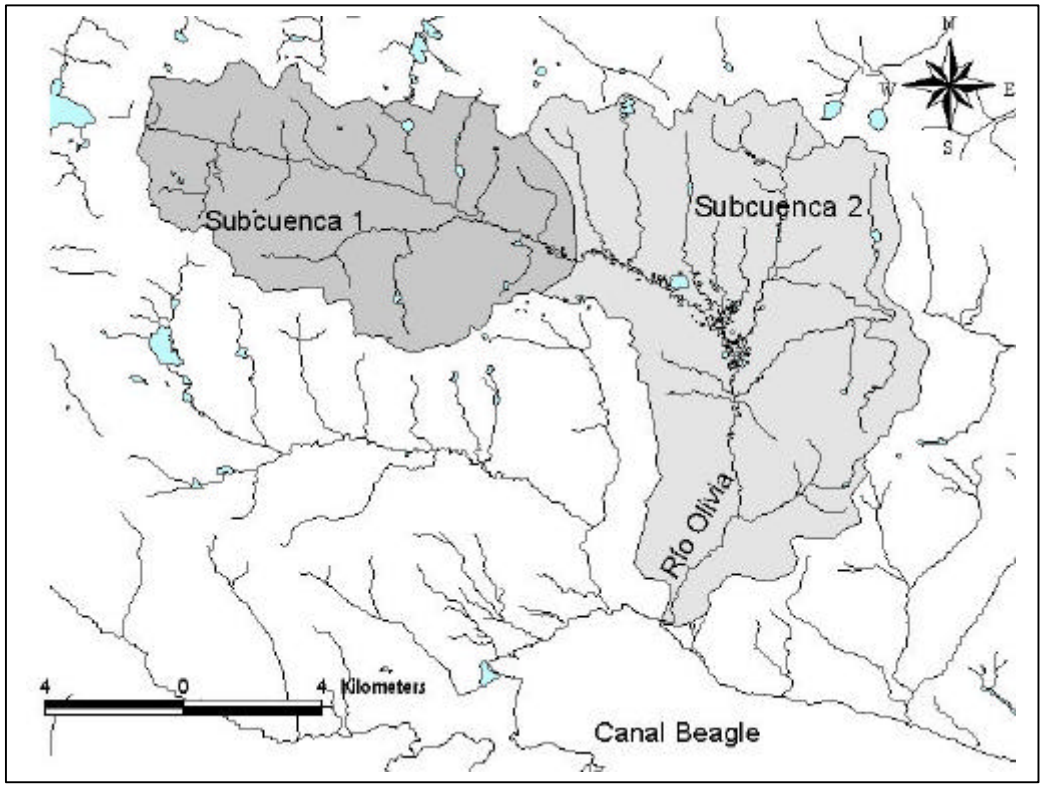

Fig 7. Sistema Hidrológico real a simular: Cuenca del Río Olivia

Para el cálculo de la escorentía (componente runoff), se utiliza el método Clark en ambas subcuencas y para el cálculo de la propagación (componente routing) se aplica un método basado en la ecuación de continuidad.

El escenario de simulación para el sistema *cuenca* a modelar, se configura de la siguiente forma:

```
** Caso de Uso Construcción del escenario de simulación**
public class Construccion_Sistema_Hidrologico
{
        public Construccion_Sistema_Hidrologico(){}
        public static void main(){
        Compmod AreaAp1,AreaAp2,TRio;
        AreaAp1=new CompModelo(new AreaAp(),new Clark(),new DataContainer());
        AreaAp2=new CompModelo(new AreaAp(),new Clark(),new DataContainer());
        TRio=new CompModelo(new TRio(15000),new Propagacion(),new DataContainer());
                    // los componentes decorados se crean con un componente del Sistema de
                    // Información, un método de cálculo y un container de datos utilizados para los
                    // cálculos como argumentos

        ElementoModelo Elemento1=new ElementoModelo(); // se crea el Objeto Subcuenca 1
        Elemento1.add(AreaAp1); // se agrega un área de aporte a la Subcuenca 1
        ElementoModelo Elemento2=new ElementoModelo(true); // se crea el Objeto Subcuenca 2
        Elemento2.add(AreaAp2); // se agrega un área de aporte a la Subcuenca 2
        Elemento2.add(TRio); // se agrega un Tramo de río a la Subcuenca 2

        SistemaHidro CuencaOlivia=new SistemaHidro(); // se crea el Objeto SistemaHidro
        CuencaOlivia.add(Elemento1); // se agrega la Subcuenca 1 al sistema CuencaOlivia
        CuencaOlivia.add(Elemento2); // se agrega la Subcuenca 2 al sistema CuencaOlivia
        (CuencaOlivia.qSimulado()).imprimir(); // se calcula la salida total del sistema CuencaOivia
        }
}
```

Como se observa, se crean los elementos runoff-routing como agregación de componentes hidrológicos (áreas de aporte, tramos de río, etc.) decorados con métodos de simulación y posteriormente se agregan al Sistema Hidrológico Cuenca. En el caso del primer elemento, no se considera la propagación en el tramo de río, por tratarse de una Subcuenca exterior (no recibe flujo para propagar de ningún elemento aguas arriba). Cada elemento puede crearse despreciando algún componente en el caso que no sea significativo. Además pueden asignarse distintos métodos de cálculo para cada uno de ellos.

Conclusiones

El presente trabajo analiza modelos de diseño flexibles para el software de modelación hidrológica. Las microarquitecturas de diseño obtenidas facilitan la construcción de aplicaciones de simulación hidrológica brindando una infraestructura de diseño básica para la simulación de procesos por distintos métodos. Mediante esta infraestructura es posible configurar escenarios de simulación en forma flexible, construyendo el sistema a modelar a partir de componentes hidrológicos tanto primitivos como compuestos, los cuales pueden seleccionar dinámicamente su estrategia de simulación.

Bibliografía

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

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

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

Bass L., Clements P., Kazman R., *Software Architecture in Practice* Addison-Wesley, 1998

Beven, K., *Rainfall-Runoff Modelling*. Wiley, 2000

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

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

Chow Ven Te, *Hidrología Superficial* Addison Wesley, 1997

Gamma, E. H *Design Patterns. Elements of Reusable OO Software* Addison-Wesley, 1995

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

Lisboa J., Iochpe C *Specifying Analysis Patterns for Geographic Databases on the basis of a conceptual framework*., 2000

Maidment D.R.. *GIS and Hydrologic Modeling-an Assessment of Progress*. The Third International Conference on GIS and Environmental Modeling. Santa Fe, NM, 1996

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

O'Callaghan A., Wills A. *Book Chapter: Architecture, Patterns and Components*, Object-Oriented Methods Principles, Products and Practices' by Ian Graham pp. 325 - 377, Addison Wesley 3rd. ed., 2001.

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

Urciuolo A., Iturraspe R., Parson Ariel, Sandoval Sandra *Patrones conceptuales para Sistemas de Información Hídrica.* Actas del CACIC 2002, Octubre de 2002, Buenos Aires