UNIVERSITY OF LIVERPOOL

AN OPEN SOURCE INFORMATION SYSTEM FOR INTEGRATED WATER BASIN MANAGEMENT

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor in Philosophy

2011

ANDREA LEONE

SCHOOL OF ENGINEERING

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENT

My profound gratitude goes to my supervisor Prof. Daoyi Chen for his precious guidance, patience, trust, wisdom and flexibility.
I would like to thank Dr. Shahriar Shams for his valuable support throughout this research work.
I extend my thanks to my present employer, the Joint Research Centres of the European Commission, for supporting and giving me the time for the completion of this Thesis, in particular to Dr. Cesar Carmona Moreno.
A special thank goes to my partner Giorgia for helping me in finding back the enthusiasm and the strength to continue this work.
Special warm thanks go to my parents, Mariella e Gianni, to my sister Francesca, and to my entire family for their unconditional love and support.

THE AUTHOR

Andrea Leone owns a Master's Degree in Civil and Environmental Engineering from the Faculty of Engineering of the University of Palermo (Italy).

He started his career as researcher in hydroinformatics for private consultants in Italy.
He completed a Master of Science in Innovation and New Technologies Management at the Area Science Park of Trieste (Italy).
He has been working as programme officer for international development and research management for the past 5 years for the European Commission.

He has published three International Journal papers related to this Thesis:

- Leone, A, Shams, S., Chen, D., 2006. An Object-Oriented and OpenGIS Supported Hydro Information System (30-HIS) for Upper Mersey River Basin Management. Journal of River Basin Management, Vol. 4 issue 2 pp 1-9.

- Leone, A., Chen, D., 2007. Implementation of an object oriented data model in an information system for water catchment management: Java JDO and Db4o Object Database. Environmental Modelling & Software, Elsevier, Vol. 22, Issue 12, December 2007, pp 1805-1810.

- Chen, D., Shams, S., Carmona-Moreno, C., Leone, A., 2010. Assessment of Open Source GIS Software for Water Resources Management in Developing Countries. J of Hydro-Environment Research, Vol. 4, Issue 3, October 2010, pp 253-264.

DECLARATION

No portion of the work referred to in the dissertation has been submitted in support of an application for another degree or qualification of this or any other University or other institute of learning.

COPYRIGHT

LIST OF THE MOST USED ABBREVIATIONS

API	Application Programming Interface
COBRA	Concise Object Relational Architecture
DBMS	Database Management System
RDBMS	Relational Database Management System
DSS	Decision Support System
ESRI	Environmental Systems Research Institute
EU	European Union
FOSS	Free and Open Source Software
GFOSS	Geographic Free and Open Source Software
GIS	Geographic Information Systems
GUI	Graphical User Interface
GWP	Global Water Partnership
DB	Data Base
HIS	Hydro Information System
IWRM	Integrated Water Resources Management
IWBM	Integrated Water Basin Management
IS	Information System
IT	Information Technology
JDBC	Java Database Connectivity
JDO	Java Data Object
JDOQL	JDO Query Language
NOAA	National Oceanic and Atmospheric Administration
ODBC	Open Database Connectivity
OGC	Open GIS Consortium
ODBMS	Object Database Management System
ODB	Object Data Base
OO	Object Oriented
OpenMI	Open Modelling Interface
RMI	Java Remote Method Invocation
SQL	Structured Query Language

WFD	Water Framework Directive
XML	Extensible Markup Language
USDA	United States Department of Agriculture

THESIS ABSTRACT

This thesis aims at researching on how the current stage of hydroinformatics modelling could evolve with relevance to new European (and international) water resource management policies analysing IT solutions such as standardisation and openness (intended as open source code). This thesis demonstrates the applicability of the open source approach to hydroinformatics and its relevance to integration of software components through a case study. This case study consists of the development of an Object-Oriented and OpenGIS supported pilot information system taking into consideration the fundamental Open technologies and Open Standards. The pilot information system uses Java as core programming language; SQL as data base language in its first implementation; the OpenGIS consortium Specifications as standard for the GIS enabled components. The system integrates open source projects like OpenMap as geographical information system and PostgreSQL with PostGIS extension as relational database.

Besides, this research work intends to demonstrate the importance and the relevance of the object oriented approach at data management level in hydroinformatic applications. Consequently an evolution of the IS developed is also proposed. At a first stage all the component of the IS are based on object oriented technologies except for the data management level based on a relational database. This is because the relational logic still represents the uncontested choice of information system for developers in hydroinformatics despite the "Object - Relational impedance mismatch". We studied solutions to address this issue and therefore we presented our experience in testing two different technologies for developing an object oriented data management layer: (i) the Java solution to obtain transparent persistence, the Java Data Object Technology (JDO); (ii) a purer Object solution with a light Open Source Object Database, Db4o. The process for implementing the two technologies in the hydro-Information System is described and the two different solutions are analysed and compared.

CHAPTER I

1 INTRODUCTION AND BACKGROUND

1.1 Background

There is a general consensus on the fact that water resources should be holistically managed to improve environmental sustainability and an integrated water resources management is needed to achieve it. However, its application is rather complex and new and still finds many unresolved issues in Europe and all over the world.

The political endorsement to the importance of the rational use of water resources, with the concept of Integrated Water Resources Management (IWRM), has been agreed only since the major international environmental conferences of the early 1990s (RIO 1992[1], Dublin 1992, Johannesburg 2002, etc.). IWRM was considered as a step towards sustainable development. The international community asked to "develop integrated water resource management and water efficiency plans by 2005" in World Summit on Sustainable Development (WSSD) held in Johannesburg, 2002.

The Technical Advisory Committee of the Global Water Partnership defined Integrated Water Resources Management as a "process which promotes the coordinated development and management of water, land and related resources in order to maximize the resultant economic and social welfare in an equitable manner without compromising the sustainability of vital ecosystems," (GWP[2] IWRM, 2004).

Therefore the IWRM approach aims at addressing water-related development problems by including different sub-sectors — such as waste water management, irrigation, hydropower production, industrial uses, etc. as a single unit with simultaneous interactions with the others more effectively and efficiently than using traditional approaches where sub-sectors are considered separately. IWRM aims to ensure that current demands for water are met without

1 UN Conference on Environment and Development in Rio de Janeiro 1992, International Conference on Water and Environment of Dublin 1992, World Summit on Sustainable Development of Johannesburg 2002.

2 Global Water Partnership (GWP) is an agency created by the World Bank, the United Nations Development Program (UNDP) and the Swedish International Development Agency (SIDA) to "support countries in the sustainable management of their water resources." (www.gwpforum.org)

jeopardizing the ability of future generations to meet their water demand (GWP IWRM, 2004).

It is worth to mention that beyond the traditional sub-sectors already cited, many other functions of a river basin ought to be accepted, such as tourism, nature, biodiversity conservation and cultural heritage.

This in contrast to the fact that due to the intrinsic complexity of water resources management, to lack of long term programming by the responsible institutions and to the complexity of the institutional set up itself, traditionally river basin management has been seen only in terms of water supply. The focus of management has emphasised on the various uses of water rather than on various sub-sectors and their integration with one another.

Consequently, following the outlined idea, and moving from fragmented towards integrated management, according to the World Summit for Sustainable Development (WSSD) Plan of Implementation Directive, one of the most important actions to undertake is to "develop and implement national/regional strategies, plans and programmes with regard to integrated river basin, watershed and groundwater management..." (GWP IWRM, 2004).

At the European level, the Water Framework Directive (WFD), adopted on 23rd October 2000, seeks to improve the sustainability of water use and water demand management thereby establishing a European water policy. The WFD is the most substantial piece of the European Union water sector related legislation. It requires all inland and coastal waters to reach "good ecological status" by 2015. This task will be achieved by drawing the framework for management and protection of water bodies by river basin district. The implementation of the WFD identifies the river basin management as the foundation for integrated water resources planning.

River basins and catchment areas have always been recognised as a useful rational unit of analysis for water management. Therefore the basin is the "geographical unit" to consider when balancing all the possible use of water and their impact on the environment. The integrated river basin management will be carried out by establishing a river basin district structure within which demanding environmental objectives will be set out, including ecological targets for surface and ground waters. As rivers and catchment areas are physical and ecological entities, changing patterns of land and resource use in upstream areas will have an impact on downstream areas (GWP IWBM, 2004).

It is therefore justified that the approach of the Integrated Water Resources Management had to be implemented through an effective integrated water basin management (IWBM).

In this sense, the goals of integrated river basin management fall within the framework of sustainable development principles according to which environmental conservation is of equal importance to economic efficiency and social equity, all sought in a long-term perspective on the basis of intergenerational equity.

From an institutional and organizational point of view, an environmental system and its complex nature requires a high level of integration within and between managing structures. To be effective river basin and catchment management organizations will have to encourage co-operation between stakeholders. A high degree of horizontal co-operation is required principally among interdisciplinary institutions at the preparation and designing stage and a high level of vertical collaboration is required between the institutions at the stage of implementation (UNEP, 1999).

From a scientific and technical point of view, integrated river basin management entails managing simultaneously the whole basin knowledge base. Managing independently environmental processes cannot produce sensible and supported decisions as when taking into account the wider vision suggested by the IRBM approach.

The water catchment policy makers and the water catchment managers need to understand all the possible impacts of pursuing any given policy. Implicitly this involves the need both to understand and to be able to model not only the individual catchment processes but also its interactions to have the necessary decision support tools.

Environmental processes are linked to and influenced by interactions of many users. Besides the multiple use of the water resource in a river basin and the several decision makers involved make the system even more complex. Considering these interactions, interdependencies and interrelations among the above mentioned processes and components and given the quantity and quality of data analysis involved in all the relevant processes, as a consequence the analysis of river basin systems and related decision-making processes has to be supported by the application of advanced Information and Communication Technology Tools (ICT- Tools).

This variety of decision support tools to be employed in River Basin Management includes: data-bases (DB) for managing and structuring the collections of data, Geographic Information Systems (GIS) for geospatial data analysis and management and modelling and simulation tools for a variety of purposes such as uncertainty and risk analysis, Environmental Impact Assessment (EIA), Strategic Environmental Assessment (SEA), economic evaluation of costs and benefits, environment-development scenarios and many other kinds of models. All those components are essential elements of Decision Support Systems (DSS) comprising of a class of computer-based information systems directly supporting users and managers in monitoring tasks and decision making processes in order to apply sustainable development principles (Becker, 2005).

In the last ten years, in order to support IWRM, increasingly powerful software tools and systems have been developed and used, coping with the complexity of environmental systems in general and basin systems in particular. The constant development of more complex software systems on one side and the improved processing capacities of hardware on the other side fostered the enhancement of performances of information systems (software + hardware). Therefore users and managers can benefit, through a more complete and reliable integrated modelling exercise, from a better representation and interpretation capabilities of natural phenomena. However in an operational context, the difficulties of implementing, handling and integrating such tools still create important obstacles for applications oriented to support the tasks of managers of river basin systems.

The analysis and adaptation of software architecture schemes towards the integration and implementation of software systems for simulation models and knowledge management related to integrated river basin management has been the major focus of this research. The application of computer science to water resources management, particularly related to hydraulic modelling (e.g. computational fluid dynamics), already includes an extensive literature. The term hydroinformatics is referenced from the defining text on the subject, Abbott (1991). Abbott clarified the meaning and defined the context for the discipline further than the mere application of information technology to problems of "the aquatic environment". According to Abbott, the Hydroinformatics is more about people, in the sense of socio-economic perspective, rather than just about technology. This approach tries to highlight the importance of representing a complex system such as the environment of which the evolution is interdependent with human activities. The use of hydroinformatics tools has to be seen through its direct or indirect impact on socio-economical issues. Abbot explains that it cannot be considered a science in the purest sense "because common-sense truths have also to be accommodated within Hydroinformatics, this subject cannot itself be a science, even though it draws upon many sciences" (Abbott, 1991). Abbott highlights the fact that large amounts of data about a "water system" have always been *"available but not readily accessible"*. The immediate example is that usually hydrological data have always been stored in the technically unreachable files of local administrations. With the development of hydraulic simulation models, these data had *"ceased to act upon our present world"*. Finally that data can be *"immediately accessible"*, the role of hydroinformatics is to treat the data in order to allow decision-makers to read through the resulting information carrying out their tasks.

Besides, Abbott (1999) argued also that the application of Hydroinformatics can be an effective support to decision making processes for users, developer and mangers of water resources. Abbott reaffirms, underlines and expands similar concepts later (Abbott, 2007) "Applications of numerical modelling, like applications of almost everything else nowadays, are characterised by a change in the way that knowledge is produced and employed within society". This change has to be taken into account in the way that knowledge is produced and employed within information systems. Therefore this evolution in the way that knowledge is treated in society is reflected in parallel in the evolution of generations of numerical modelling as it is explained below.

The research described in this work takes its origins from the fact that the current state of modelling technology could be adequate to support effectively the decision making process with the necessary improvements to the integration capabilities of the variety of tools and methods (DSS, GIS, DB, etc.) used in River Basin Management in offline and real-time contexts.

The development of Hydroinformatics — and the further development of those tools which are fundamental to Hydroinformatics— has been limited by the *ad hoc* way in which Hydroinformatics systems have been developed.
There has been a lack of research into (and thereafter development of) software architectures for the construction of flexible modelling engines, and as a result for the overall systems to be able to deal with the integrated water environment complexity (Harvey, 2002).
Trying to define simulation models in hydroinformatics in a generic way, they can be thought as piece of software constituted by an engine (in general an algorithm composed of mathematical and logical relations that are analysed by numerical methods) treating an input and providing an output. Simulation models in hydroinformatics generally aim at representing, interpreting, studying, foreseeing, etc. phenomena related to the environment. Simulation models *per se* are becoming continuously more powerful thanks to the extensive research and development work going on in the sector. However, models have different strengths and weaknesses and during the various phases of a modelling exercise it may be appropriate to replace one model by another or to upgrade a model with the new findings. Different situations require different combinations of models, and the different models need to be integrated in an environmental information system capable of supporting the managers in their decision making processes. It is not possible to create a "definitive" information system which can be adapted in every situation of river basin management. Being “definitive” in this case used to identify static system that cannot be updated or modified.
Abbott (1991) expected that the next generation of modelling was ought to develop a wrapping layer embedding the numerical model as means of connection to the hydroinformatics system. Nowadays, the approaches are either "adapting" existing modelling tools as components of larger systems or "copying and pasting" existing modelling algorithms and codes into another modelling tool, which is more adapted to new tasks of dealing with river basin complexity (for example Tomicic and Yde, 1998).
An elucidating approach on modelling evolution given by Abbott (2007), proposes to organise the different stages of development in five generations. The first generation was focusing on algorithms based on a series of equations, the second on tailoring solutions case by case through dedicated projects, the third generation introduced the concept of modelling systems

allowing developers to build customised models. In the fourth generation those modelling systems were organised in packages until getting to the fifth generation with the introduction of the concept, as defined by Abbott, of 'software-as-a-service'. The most important variation for the modelling-knowledge needed by users was introduced in the fourth generation in which companies started to give active support services (training courses, telephonic and on line support, etc.). In the transformation from third to fourth generation modelling, more than just a change on the level of knowledge needed by the users, there was also a change on the kind of knowledge needed. In fact, the level of knowledge needed related to computational-hydraulic mathematics modelling related was reduced dramatically, with on the other hand a strong increase on knowledge needed related to the other scientific domains involved in an information system in which all needed environmental models should be integrated.

In the fifth generation of modelling, this need for integration of different kinds of domain of knowledge is still increasing. The new issue in the information systems for water basin management and for environmental modelling in general, as mentioned above, is no longer numerical-computational software but a complete integration of environmental, hydraulic and mathematical and computational models.

It is important to follow the strategies of international IT corporations, such as IBM as these give an interesting overview of the general development trends of the IT sector to which hydroinformatics is directly related. Therefore, supporting the development of the fifth generation of modelling in hydroinformatics, it is interesting to cite the IBM's recent important change of focus towards "middleware" (Bradbury, 2008). Middleware is defined as the piece of software which acts as a layer between the level of the operating systems and the applications in such a way as to promote a wide range of new applications and their integration. This middleware, which is not just a piece of software but also a conceptual approach to IS architecture, can be identified in the case of the fifth generation modelling defined by Abbot, as the interface between the generic knowledge and the particular domain knowledge needed in the IS and included in the application modelling tools.

The approach of treating the different kinds of knowledge with different applications and integrating them together through a middleware brings to a complete rethinking of modelling system and supports the evolution of the fifth generation. As already emphasised, the

knowledge needed to undertake the approach supported by integrated water basin management is significant in terms of quantity and variety, and even so difficult to identify that the one-provider only strategy cannot possibly work. Sources of knowledge rely increasingly upon other participants on the supply side to contribute much more easily and effectively in the processes of knowledge provision.
This multi-actors approach to knowledge provision entails a need for a heterogeneous approach to knowledge gathering interfacing that can only be realised by providing connection channels to the knowledge producers based on open-source software which they can most conveniently adapt to their own standards and peculiarities (Abbott, 2007). This open-source based strategy is a consistent thread all over this research work being the solution proposed for the software technology approach to move towards some of the issues highlighted in the previous paragraphs and related to the integrated water basin management. The necessity to involve heterogeneous source of knowledge to the need for constant evolution and adaptability of the software, for integration of different modelling tools to the need for an open approach to information system development, the open-source software can be a good solution, as discussed below and in the next chapters.

1.2 Research contributions

This research is meant to undertake an exploratory approach to information system development, with a technology transfer from informatics research to hydroinformatics practice following analysis of requirements of integrated water resources management..
In particular this thesis aims at demonstrating the importance of the concept of integration in the current stage of hydroinformatics modelling evolution with relevance to the new European (and international) water resource management policies and analyse solutions such as standardisation and openness (intended as open source code). In order to achieve these objectives, the information system architecture will be tailored to become a framework with different levels in which different components are able to work together as one "flexible and replaceable" part of the larger hydroinformatics software system.

Based on the usefulness and the relevance of open source code to hydroinformatics, this thesis aims at demonstrating the applicability of the open source approach to hydroinformatics through a case study by developing an open source based pilot information system. This development entailed the approach of integration of existing open source code through the development of software interfaces.
Finally the thesis intends to demonstrate the importance and the relevance of the object oriented approach at data management level in hydroinformatic applications.

1.3 Rational and methodology

According to ATIS (Alliance for Telecommunication Industry Solutions, www.atis.org) an Information System (IS) can be defined as:

1. A system, whether automated or manual, that comprises people, machines, and/or methods organized to collect, process, transmit, and disseminate data that represent user information.
2. Any telecommunications and/or computer related equipment or interconnected system or subsystems of equipment that is used in the acquisition, storage, manipulation, management, movement, control, display, switching, interchange, transmission, or reception of voice and/or data, and includes software, firmware, and hardware.
3. The entire infrastructure, organization, personnel, and components for the collection, processing, storage, transmission, display, dissemination, and disposition of information.

Following this schema, an appropriate definition for our hydroinformatics information system can be: a group of interdependent software tools (also components) that interact regularly to

perform a task; a system that collects, stores, process and visualize data through a user interface.
The main task of the information system is to support the end user in the integrated management of a water basin. Being the IS a set of interdependent software tools, its design and implementation focuses on the linkages and integration among the different components that constitute its structure. Besides, this research further looks into the flexibility of its structure in integrating a new innovative component such as an object oriented database (as it will be defined in the next chapters).
Particularly, the information system shall allow the end user to model and manage environmental systems such as a water basin, dealing with its complexity, variability, dynamism, its continuing evolution and with the amount of complex knowledge to manage.
Therefore the software system shall allow the end user to better manage water quantity models, water quality models, geomorphologic analysis models, GIS, other water basin and environmental models, and in general to manage and process all the "environmental knowledge" related to the IWBM.
To meet these goals, we can first set a list of general requirements of such an information system: integrability, extensibility, portability and open source based.
According to Bass, Clements, and Kazman, "integrability" refers to the possibility for separately developed elements (in our case geoDB, GIS, Simulation models, DSS, etc.) to work together fulfilling software's requirements (Bass, 2003).
Extensibility is the degree to which something is able to be enhanced in the future to meet the changing requirements or goals. It is necessary because the platform can be extended with new simulation models and/or new management tools useful to the end users.
Since there are various organisational management approaches in the basin environment, which use different kinds of information systems in terms of software and hardware; portability is a fundamental feature to encourage effective scientific, technical and administrative cooperation. Another important feature of the portability is the support of knowledge sharing using standard technologies to make the information system widely compatible and accessible, above all in various web client contexts (Intranet, Internet, etc.).

The forth general requirement is the open source technology. Open source is a development method for software that harnesses the power of distributed peer review and transparency of the process. The advantages of open source are better quality, higher reliability, more flexibility, lower cost, and an end to predatory vendor lock-in (www.opensource.org).
Hence, we used for our information system, when possible, open source technologies, because we reckon them a fundamental requirement in order to meet the other three requirements and for two other main reasons exposed below.
First, this is an academic research software and its results can be more interesting and more easily diffused if they are not tied to closed or proprietary technology; Second, the research subject is connected to the most basic and vital resource necessary to life, water, hence its result should be open and useable by everybody even without any software licence.
We reckon important at this moment affirm that this thesis focuses on hydroinformatics research and development but not on pure informatics research. Although the content is strongly related to IT, the main innovative points discussed and proposed are related to the applications of IT to water resources management and hydroinformatics in general. Therefore this research has not focused on developing and designing innovative technologies for information systems; rather we aim at researching on the best available technologies for information system development suitable to fit the needs of hydroinformatics in the field of IWBM/IWRM.
The main question is: how hydroinformatics should support and address the holistic approach introduced as a principle of IWRM?
Consequently the objective of this thesis is to describe experiences and present related case studies in conceiving and developing integrating data management and modelling systems for water catchment. To answer these questions, an overall framework was defined, outlining a series of criteria, describing and analysing a list of relevant projects.
As a result, the rational of this research, is to find, assess, adapt and implement existing technological solutions, allowing them to interoperate in an information system for integrated water resources management at water catchment level.
While many of the software components which would need to be embedded in our integrated systems (GIS, DB, geoDb, etc.) already exist and research in this field is continuous and

extensive, their integration in hydroinformatics can be considered still relatively new as showed in the literature review (next chapter).

1.4 Thesis Structure

The organisation of this thesis is intended to approach the subject of this research firstly from a theoretical point view in a broader way, secondly to demonstrate a general possible application and thirdly to focus in particular on some interesting issues we faced and reckoned interesting. There are eight chapters and each chapter is briefly presented below:

Chapter I: This chapter provides introduction and background information with the objectives of the research.

Chapter II: The literature review with a list of similar and relevant research projects and information systems.

Chapter III: The presentation of the approach to the research, the methodology and the explanation of technology choices.

Chapter IV: The design of a prototype of the software information system and its architecture. The development sequence which makes use of existing open source software and their characteristics.

Chapter V: The implementation and the case study.

Chapter VI: The particular and innovative design, development and implementation of the data layer of the information system with an application of an object oriented database to hydroinformatics.

Chapter VII: Conclusions and further work.

Chapter VIII: References.

The three sections in sequence "methodology and technology", "system design" and "implementation and case study", in conformity with the overall thesis structure, first initiate and illustrate ideas from a broader theoretical approach and then progress focusing on concrete applications.

A particular explanation is worth for the section "Development of the data layer". In fact, in the first part of the research summarised in the "implementation and case study" section, some of the most innovative software technologies relevant to IWBM have been chosen for the different part of the system, nonetheless for the data layer the alternative of the "relational logic" was preferred to the more innovative "object logic" because of the inadequate literature existing in hydroinformatics and in general also in applied informatics. This issue is then treated in deep in the "Development of the data layer" section, in which an application of the "object logic" to the data layer is developed with a case study analysed and assessed for this particular purpose.

CHAPTER II

2 LITERATURE AND PROJECTS REVIEW

2.1 Review

Nowadays it is widely accepted that integrated water basin management requires a holistic, transversal and wider view of human impacts on water resources. A modelling framework for integrated catchment management must allow all domains of the catchment to be modelled and must allow the linking of, for example, environmental and economic models to hydrological models.

This recognition and the new legislation building on it, such as, the EU Water Framework Directive has led to numerous efforts in the integration of various information systems towards a holistic, long-term approach of management in the environmental context. Many of the current modelling solutions use integrated modelling systems that aim to contain all of the required domain components. A number of modelling frameworks have been developed that appear to have promising features which could allow the linking of further domain modules and swapping of existing ones. In important ongoing projects (e.g. Open-MI) there is strong research towards integration and standardisation, however, many of them do not seem to be used outside their own development environment, suggesting that in any case they lack extensibility or integrability, being possibly closed, vendor tied or having a rigid monolithic architecture.

It not easy to draw a line between what can be considered on one side an information system or a modelling framework or what can be considered a non-modular software program for hydroinformatics. In the review below, the features which brought to include one projects in this section are underlined. In general, the projects analysed are about modelling frameworks and integrated modelling information systems.

In object oriented programming, by definition, a framework is an object-oriented design. It does not have to be implemented in an object-oriented language, though it usually is. Large-scale reuse of object-oriented libraries requires frameworks (Linthicum, 1997).

According to Linthicum a framework is a semi-complete application that programmers can customize to form complete applications by inheriting from and instantiating classes in the framework.

To better explain the concept, it is useful to compare them with integrated models particularly as it related to our research subject. Modelling frameworks, generally, are open information systems that leave the end user the room to chose which models to use, whereas integrated models/modelling systems are a group of simulation models that have been combined or just linked together in order to represent a specific system or set of systems. The intent of integrated models is to provide a new modelling tool, whereas the objective of a framework is usually to provide a mechanism, often represented as a system architecture, by which existing models can be linked or organised in an effective way.
Some of the most interesting modelling frameworks and integrated modelling systems developed in the last few years have been analysed.

Generic Framework

The Generic Framework (Blind et al., 2001) is an open framework for linking models and decision supports that have been developed by a large consortium of companies, commissioned by RIZA, STOWA, Alterra and RIVM in the Netherlands.
The purpose of the framework is to link all types of models relevant to integrated catchment modelling and potentially environmental modelling in general, on a time-step in memory basis, which also allows iterations and feedback loops. The types of models include user-function models (e.g. shipping, drinking water, agricultural costs).
The philosophy employed in building the GF version 0.9 was to focus on generic linkages and functional modularity. As a result, the "scientific logic" for automated linkage is not yet implemented and many functions are only available in their most simple forms. The intention is that future modelling projects will provide the required extensions of (generic) functionality within the GF.
A useful categorisation of model domains relevant to HarmonIT was developed within the GF project. Here the model domain was studied from three angles in order to gain a generic description. The angles (or views) were: physical domains, modelling and simulating approaches and; environment for development.
The "modelling and simulating" domain view expressed in the GF is based around the concept of model elements and connectors which are combined to form model components.

Model components are combined to form model applications. This approach has been identified as being applicable for 1D, 2D, 3D modelling concepts, for finite element, analytical element and random walk solutions.
The "environment for development" domain view simply defines two framework components: model programs (computational kernels plus the required data) and generic tools. The generic tools are further sub-divided as "in-process" tools (which in general allow a number of model programs to communicate e.g. a process chain management tool) and "linked" tools (which have no influence on the results calculated, e.g. graphical results presentation tool).

Open Modelling System

The Open Modelling System (Goede, 2005) is dedicated to the development of an operational modelling system for complex 2D/3D simulation of flow and transport processes, by integrating the Delft3D system (Delft) with the SIMONA system (by RIKZ, Netherlands).
The OMS software architecture is a flexible environment developed to accommodate data exchange and synchronisation between legacy computational cores. Its main focus is on the low-level communication, by way of standardisation of unambiguous defined interfaces, data structures, communication protocols and file formats. A major role is being played by a standardized input/output library (DelftIO for Delft3D and Couple for Simona).
The Delft Cluster-OMS (DC-OMS) project is based on the same ideas, namely focus on the data exchange level. Through a co-operation between WL Delft Hydraulics, GeoDelft (geotechnics) and TNO-Bouw (mechanical engineering), the DelftIO library has been extended to include data structures and file formats relevant to non-water related civil engineering disciplines.
DC-OMS is linked to the Generic Framework and OMS, but additionally includes elements of water management such as hydraulic engineering works, modelling dam stresses etc.

Object Modelling System

The Object Modelling System (OMS) is described as a Java-based modelling framework consisting of a library of science, control and database components, which facilitates the

assembly of selected modelling components into a modelling package suited to the problem, data constraints and scale of application (OMS, 2005; Ahuja et al., 2004). OMS is a modelling framework that facilitates model development, evaluation and deployment (David et al., 2004). The concept behind OMS is to create all system and model tools as independent reusable components that may be coupled using standardised software interfaces to create an application-specific modelling package (Kralisch et al., 2004). Development of OMS began in 1996 in the Institute for Geography at Friedrich Schiller University, Jena, Germany (Ahuja et al., 2004). Since October 2000 development of OMS has continued as an inter-agency project supported by the United States Department of Agriculture - Agricultural Research Service (USDA-ARS), United States Geological Survey (USGS) and United States Department of Agriculture – Natural Resources Conservation Service (USDA-NRCS) (Ahuja et al., 2004).
The two main types of components in OMS are model components and system components. Model components are the building blocks from which models are created within OMS. System components are those used to assemble user selected model components to create an application specific model, populate the model with suitable data and then execute the model. OMS is supported by graphical user interface (GUI) components and utility components which include a data dictionary, data retrieval and storage, GIS, graphical visualisation and statistical analysis (OMS, 2005; Ahuja et al., 2004). OMS uses custom metadata tags to support component documentation, testing, proper component integration into a model, automatic user interface generation and model execution (David et al., 2004). Tools are available in OMS to assist in migrating legacy models either by direct implementation in OMS or by means of wrappers. It appears that individual models built from OMS model components would be able to be linked.

TIME and the Catchment Modelling Toolkit

The Invisible Modelling Environment (TIME) is described as a modelling and programming system for developing, applying and deploying environmental models (Murray et al., 2004). TIME has been developed on the Microsoft.Net platform and is a collection of .Net classes, libraries and visualisation components for the development of models and model applications. The Catchment Modelling Toolkit is a system of environmental modelling software which integrates a new generation of catchment models and modelling support tools (Marston et al., 2002). The aim of the Catchment Modelling Toolkit is to provide land and water managers, researchers and educators with an integrated collection of software tools and components to simulate catchment response to management and climate variability, at a range of scales and using a variety of approaches (Marston et al., 2002). To achieve this aim a modelling framework was required which allowed models to be developed and integrated quickly and consistently (Rahman et al., 2003). TIME is an environmental modelling framework developed to meet these requirements and is the foundation on which the models, model applications and other modelling tools included in the Catchment Modelling Toolkit are built (Catchment Modelling Toolkit, 2005). TIME and the Catchment Modelling Toolkit were developed by the Cooperative Research Centre for Catchment Hydrology (CRCCH) in Australia (Rahman et al., 2003).

The architecture of TIME is divided conceptually into five layers: Kernel, Data, Models, Tools, and Visualisation and User Interface. Each layer consists of a family of classes, with the classes in the upper layers using services provided by classes in lower layers. Developers create models in the Model layer using classes in the Kernel and Data layers. The Tools and Visualisation and User Interface layers contain classes that interact with models and provide most of the framework functionality, such as user interface generation and model linking. The use of TIME custom metadata tags allows models to remain independent of these tools, resulting in better model stability (Rahman et al., 2004). TIME makes use of .Net's introspection capabilities for dynamic discovery of system properties at runtime (Rahman et al., 2003). These system properties include class structure, class fields and methods, and custom metadata tags which allow TIME to automate several tasks which facilitate model integration and automatic user interface generation (Rahman et al., 2003). TIME seems to support the approach of restructuring models into a set of linked modules and does not appear to make provision for wrapping or linking to legacy models.

HarmonIT and OpenMI

The Generic Framework and the OMS have been joined by the HarmonIT project (Gijsbers et al., 2002), which is largely a result of the Generic Framework. HarmonIT was a 6 million

Euro project, funded half by the European Union under the 5th Framework Programme for research of the European Union, and half by the project participants, which include major water modelling software developers from across Europe. HarmonIT has been strongly influenced by its lineage and sharing of technical staff with the Generic Framework project resulting in strong similarities of vision and outline design at present. Gijsbers et al. (2002) provide a list of issues to be resolved in model linking. This list makes no attempt to separate the intrinsic and extrinsic issues —that is, those which have technical solutions and those which are part of the essence of modelling and can only be solved on a case by case basis by model developers and model integrators. They do however provide a more comprehensive overview of the different levels of users of the outputs of the HarmonIT project.
A number of issues which are raised as being of critical importance in this thesis are addressed (at least explicitly) by none of the above projects. These include:

- the need to ensure, at the core of current model linking difficulties, that the tool under development is a complete application and will not be used as a part of a larger system;
- the need to provide scientists and other developers of component process models with more appropriate tools than programming languages for experimental model development
- the need to provide model developers and integrators at all levels with hierarchical tools for managing model complexity.

The Open Modelling Interface (**OpenMI**) is described as a generic interface allowing models simulating different water- related processes to be linked on a temporal and spatial basis, allowing simulation of process interactions (Gijsbers, 2003; HarmonIT, 2004a). The objective of OpenMI is to simplify the linking of models running in parallel and which operate at different spatial and temporal scales by means of direct transfer of data values between models without writing to or reading from intermediate text files. OpenMI focuses on resolving or improving several complicated model-linking issues, including differences in spatial and temporal scales, feedback loops, differences in spatial and temporal concepts (distributed vs. lumped, steady state vs. dynamic), different units and naming of variables, and distributed computing (Blind and Gregersen, 2004). OpenMI simplifies the linking of models from a computer science point of view, allowing modellers to concentrate on the complexities of linking models from a hydrological point of view. OpenMI has been developed under the European Commission-funded HarmonIT project towards meeting the goals of the European Union's Water Framework Directive. The project partners include three commercial partners (DHI – Water & Environment, WL | Delft Hydraulics, HR Wallingford Group) and several other partners from research institutes and universities in Europe. The fact that competing software vendors have joined forces in creating OpenMI, is a key advantage to achieving the objective of setting a standard (Blind and Gregersen, 2004).

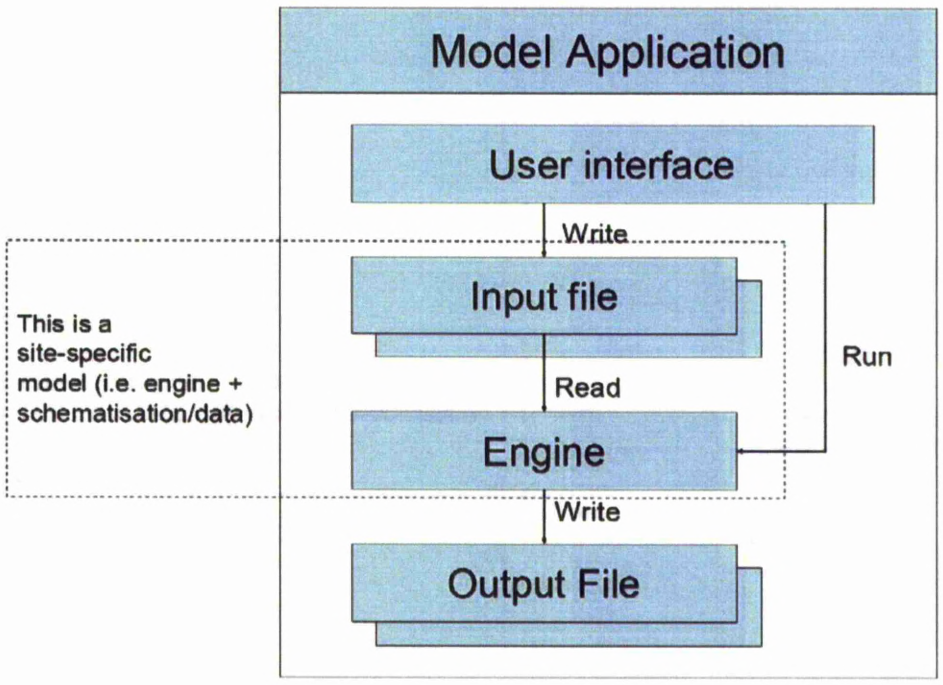


Figure 1.1 : the Architecture of the OpenMI project (Moore et al., 2007).

The architecture for OpenMI is shown in Fig. 1. The most important part of OpenMI is the Standard (org.OpenMI.Standard), which consists of a set of interfaces. The standardisation does not concern a piece of software in particular but an overall approach and set of procedures, so may be implemented in any object-oriented programming language and related computing platform. Any model implementing the relevant interfaces contained in the OpenMI Standard is described as being OpenMI compliant and may be linked to any other OpenMI compliant model. While OpenMI focuses on data exchange between models at runtime, it may also be used to link models to databases and user interfaces (Blind and Gregersen, 2004). The HarmonIT project has concentrated on implementing the Standard in the C# programming language on the Microsoft .Net computing platform but will also provide a Java implementation (HarmonIT, 2004). The other namespaces in the OpenMI architecture form the Open Modelling Environment, and provide a set of classes whose purpose is to simplify the migration process and to facilitate the linking and running of the OpenMI compliant models (HarmonIT, 2002). A default implementation of each interface in the Standard has been created to form a group of classes known as the Backbone (org.OpenMI.Backbone). The org.OpenMI.DevelopmentSupport namespace contains a generic set of low-level classes that can be used in the development of an OpenMI modelling environment. The org.OpenMI.Utilities namespace contains a set of classes that have been created to reduce the amount of re-engineering required to migrate existing model engines and software systems to become OpenMI compliant. A primary design objective for OpenMI was that the cost, skill and time required to migrate an existing model to the standard should not be a deterrent to its use (HarmonIT, 2002). The HarmonIT project recognises that there are many legacy models written in programming languages such as FORTRAN. To facilitate the linking of legacy models, a group of wrapper classes have been created which allow linking to these legacy models, without rewriting these models in an object-oriented programming language to meet the OpenMI Standard, and with a minimum of changes to the legacy models themselves. These wrapper classes take care of the entire book keeping associated with handling links, events, exceptions, buffering and basic spatial and temporal interpolation. For a legacy model engine not written in a .Net language or Java to be suitable for wrapping it must be compiled to a dynamic link library (DLL) and must be able to separately initialise, perform single time-steps, finalise and to be disposed of (HarmonIT, 2004). The org.OpenMI.Configuration namespace contains a set of classes created to help developers to administer, configure and deploy coupled OpenMI compliant modelling systems. The org.OpenMI.Tools namespace contains user interface components and other classes to facilitate user interaction with OpenMI compliant models at configuration and run-time.

One of the most interesting aspects of OpenMI is that it does not concern technical solutions chosen but the set up and the governance of the project. The OpenMI is in fact an Association of an entirely open international group of organisations and people. As it is stated in its website, www.openmi.org, the OpenMI association provides a small core team that supports, responds to and is guided by a growing active worldwide user community.

It is a not for profit organisation and therefore depends on the willingness of its members. Currently it is funded under the EC LIFE Environment Programme of the European Union.

Therefore it is evident that the structure that this project took in its evolution from “General Framework”, through “Harmon-IT”, until its actual shape is the one of an Open Source Community (as it will be defined in other sections of the thesis).

MIKE SHE/SHYLOC

“Système Hydrologique Européen” (SHE) is an integrated catchment model that simulates the entire land phase of the hydrological cycle. It was developed by the Danish Hydraulic Institute, Sogreah of France and the Institute of Hydrology (CEH).

MIKE SHE (http://www.dhisoftware.com/mikeshe/) is a modelling tool for analysis, planning and management of a wide range of water resources and environmental problems related to surface water and groundwater. MIKE SHE is an integrated modelling environment with a modular structure. Individual components can be used independently and customized to local needs, depending on data availability and aims of the given study.

Pre-processing and results presentation tools are included in the MIKE SHE software package. MIKE SHE consists of a hydrological surface model and a groundwater flow model, based on MODFLOW. MIKE SHE can read in MODFLOW files and a dynamic link to MODFLOW is being developed at DHI. Typical applications of MIKE SHE include investigations of the:

- Use of water

- Surface and groundwater interaction
- Change in land use
- Impact of farming practice
- Wetland protection
- Transport of contaminants

SHYLOC is a procedure to integrate Landsat TM satellite images with ground-based data for the determination of surface water storage in ditch networks. The procedure has been developed as part of the SHYLOC project to improve hydrological predictions in wetland systems, which are characterised by water storage rather than flow.

SWAT

The Soil and Water Assessment Tool (SWAT), produced and supported by the USDA Agricultural Research Service is the hydrological model used in the SWAT/GRASS linkage (Arnold et al. At http://www.wiz.uni-kassel.de/model_db/mdb/swat.html). SWAT is a continuous-time, basin-scale hydrological model capable of complex long-term simulations including hydrology, pesticide and nutrient cycling, erosion and sediment transport. The SWAT hydrology model is based on the water balance equation. It simulates soil water content in relation to daily precipitation, run-off, evapotranspiration and percolation. SWAT calculates parameters such as surface run-off, sediment yield and nutrient budgets. A soil database is used to obtain information on soil type, texture, depth and hydrological classification. In SWAT, soil profiles can be divided into as many as 10 layers. Such information is critical to SWAT's crop growth modelling, sediment yield and runoff and water balance calculations. Among the management options that can be investigated are irrigation water transfer.

MIKE BASIN

MIKE BASIN is a mathematical representation of the river basin encompassing the configuration of the main rivers and their tributaries, the hydrology of the basin in space and time, and existing as well as potential major schemes and their various demands of water.

MIKE BASIN is a network model in which the rivers and their main tributaries are represented by a network of branches and nodes. The branches represent individual stream sections while the nodes represent confluences, diversions, locations where certain water activities may occur, or important locations where model results are required. The model operates on the basis of a digitized river network generated directly on the computer screen in ArcView GIS. All information regarding the configuration of the flow simulation network, location of water users, reservoirs and intakes and outlets of return flow are also defined by on-screen editing.

SMS/WMS/GMS

SMS/WMS/GMS (http://www.scisoft-gms.com/) is a group of models distributed by US based Scientific Software Group to simulate watersheds, groundwater and surface water flow and quality using well-known core functions such as MODFLOW, the rational method, Reynolds equations etc. The core functions are embedded in a large data visualization system.
The Surface Water Modelling System (SMS) is a comprehensive environment for one-, two-, and three-dimensional hydrodynamic modelling. A pre- and post-processor for surface water modelling and design, SMS includes 2D finite element, 2D finite difference, and 3D finite element modelling tools. Supported models include the USACE-ERDC supported TABS-MD (GFGEN, RMA2, RMA4, SED2D-WES, HIVEL2D), ADCIRC, CGWAVE, STWAVE, BOUSS2D, M2D, GENESIS, and WABED models. A comprehensive interface has also been developed for facilitating the use of the FHWA commissioned analysis package FESWMS. The TUFLOW numerical model with powerful flood analysis, wave analysis, and hurricane analysis is now supported. SMS also includes a generic model interface, which can be used to support models which have not been officially incorporated into the system.
The numeric models supported in SMS compute a variety of information applicable to surface water modelling. Primary applications of the models include calculation of water surface elevations and flow velocities for shallow water flow problems, for both steady-state or dynamic conditions. Additional applications include the modelling of contaminant migration, salinity intrusion, sediment transport (scour and deposition), wave energy dispersion, wave properties (directions, magnitudes and amplitudes) and others.

The Watershed Modelling System (WMS) is a comprehensive graphical modelling environment for all phases of watershed hydrology and hydraulics. WMS includes powerful tools to automate modelling processes such as automated basin delineation, geometric parameter calculations, GIS overlay computations (CN, rainfall depth, roughness coefficients, etc.), cross-section extraction from terrain data, and many more! With the release of WMS 8, the software supports hydrologic modelling with HEC-1 (HEC-HMS), TR-20, TR-55, Rational Method, NFF, MODRAT, and HSPF. Hydraulic models supported include HEC-RAS and Simplified DAMBREAK. 2D integrated hydrology (including channel hydraulics and groundwater interaction) can now be modelled with GSSHA.
The program's modular design enables the user to select modules in custom combinations, allowing the user to choose only those hydrologic modelling capabilities that are required. Additional WMS modules can be purchased and added at any time. The software will dynamically link to these subsequent modules at run time and automatically adding additional modelling capability to the software.
GMS is internationally widespread used. It has been proven to be an effective and exciting modelling system. GMS provides tools for every phase of a groundwater simulation including site characterization, model development, calibration, post-processing, and visualization. GMS supports both finite-difference and finite-element models in 2D and 3D including MODFLOW 2000, MODPATH, MT3DMS/RT3D, SEAM3D, ART3D, UTCHEM, FEMWATER, PEST, UCODE, MODAEM and SEEP2D.
The program's modular design enables the user to select modules in custom combinations, allowing the user to choose only those groundwater modelling capabilities that are required. Additional GMS modules can be purchased and added at any time. The software will dynamically link to these subsequent modules at run time - automatically adding additional modelling capability to the software.

InfoWorks (http://www.wallingfordsoftware.com/products/infoworks/)
In the Wallingford Software InfoWorks system the models are hard wired into the system to provide a detailed description of the individual model parameters that allows version control and audit trails through detailed knowledge of the models. InfoWorks gives a choice of MS

Access, MSDE, SQL Server or Oracle master databases. The InfoWorks system contains rainfall runoff models, hydraulic models, water quality models, sedimentation models, and flow routing models. Different versions of InfoWorks can be used for urban drainage modelling, river modelling, and water supply modelling. The system aims to provide a single environment that integrates asset planning with detailed modelling. The aim is to provide planners and engineers with a predictive tool to assess environmental impact.

BASINS

As stated in the United States Environmental Protection Agency website (http://www.epa.gov/waterscience/basins/), BASINS is a multi-purpose environmental analysis system that integrates a geographical information system (GIS), national watershed data, and state-of-the-art environmental assessment and modelling tools into one convenient package. The American approach to IT applied to water basin management is then more practice than the European one. More than setting up a common approach to modelling, the EPA gives a solution integrating all the different modules needed. To be noticed that from version 3.1 to version 4.0 the software package has been completed with a non-proprietary, open source, free GIS system. The move towards open source solutions, seeking for more flexibility and a widespread use, is also undertaken in this case.

From a technical point of view, the Better Assessment Science Integrating point and Nonpoint Sources (BASINS) BASINS was developed by the United States Environmental Protection Agency's (USEPA's) Office of Water and was first released in 1996, currently we are at release 4.0.

The BASINS system is described in its website as a multipurpose environmental analysis system designed for use by regional, state and local agencies in the United States of America (USA) to perform catchment and water quality-based studies (USEPA, 2001). The package is designed to support the estimation of total maximum daily loads (TMDLs) using a catchment-based approach including both point and non-point sources for a variety of pollutants, at a variety of scales (USEPA, 2001). BASINS is also a decision support system developed to facilitate collection and analysis of environmental information interoperating and simulating environmental systems and providing a framework for the continuous work of practitioners, managers and decision makers.

As already mentioned BASINS is a system integrating GIS (open source from version 4.0), data analysis and modelling.

Concerning data sources, the BASINS system includes a set of custom databases compiled from a wide range of federal sources in the USA including national databases at the USEPA and United States Geological Survey (USGS). Simply data are extracted from databases and stored in Water Data Management (WDM) and DBF files, a kind of very long lasting database management system files (http://www.dbase.com/), which may be accessed by BASINS models and tools. BASINS used ArcView to provide a customised user interface until version 3.1 and is now using Mapwindow GIS (http://www.mapwindow.org/) as visualisation layer, which is a ready-to-use open source spatial data viewer and tool that can be modified into custom applications. The user interface contains all standard and customised on purpose Mapwindow menu, button, and toolbar items which access query, spatial analysis, and map generation tools. Tools and menus specific to BASINS are accessed through the BASINS Extension Manager and include assessment tools, data management utilities, catchment characterisation reports, water quality models and catchment hydrology models. BASINS uses the GenScn post-processing tool to facilitate the display and interpretation of observed water quality and other time series data, and the analysis of model output data. Models to be run from BASINS would need to be able to read from and write to the relevant BASINS files in WDM and DBF format.

Confirming its integrated approach, BASINS include a series of existing and developed in parallel watershed model application models. In particular, currently they are:

- HSPF, Hydrological Simulation Program FORTRAN, http://www.aquaterra.com/resources/hspfsupport/index.php,
- SWAT, Soil and Water Assessment Tool, http://swatmodel.tamu.edu,
- QUAL2K, River and Stream Water Quality Model, http://www.epa.gov/athens/wwqtsc/html/qual2k.html

- PLOAD models, an ArcView GIS Tool to Calculate Nonpoint Sources of Pollution in Watershed and Stormwater Projects http://www.epa.gov/waterscience/basins/b3docs/PLOAD_v3.pdf)

CUAHSI HIS

Another large USA project, this time with a similar approach to OpenMI project, is the CUAHSI (http://www.cuahsi.org/). CUAHSI (Consortium of Universities for the Advancement of Hydrologic Science, Inc.) is an organisation representing more than one hundred United States universities receiving support from the National Science Foundation to develop infrastructure and services for the advancement of hydrologic science and education (Tarboton et al., 2007).

CUAHSI is Hydro Information System having as primary objective the integration of the USA water information in order to make data widely useful and accessible. This purpose is pursued by providing access to the data sources, tools and models supporting the user throughout analysis, treatment, visualisation and evaluation of behaviours of hydrologic systems. The CUAHSI Hydrologic Information System bases its architecture on distributed computing. Its network is constituted of geographically distributed hydrologic data sources and models integrated using web services. The result is that the system works and acts as one integrated platform.

As it happens in Europe, environmental data is collected by a long series of defend institutions, at federal level, at state level, by local agencies, by academic scientists as well as by consultancy firms and do on. Therefore, linked to this variety of available sources there are a series of issues that CUAHSI tries to address (Johnson, 2008):

- Access protocols and data exporting formats can be highly heterogeneous from source to source, and when sources are several, handling this variety can be a serious issue. Although the internet has improved access to these disparate data sources in terms of speed and reliability, standardisation of protocols is still very far to be achieved.

- A particular problem is not related to IT, but the fact that often developers and data gatherers in hydroinformatics are not IT specialist but individual researchers or practitioners with a strong expertise in hydraulics or water resources management related fields. When hydraulics experts publish their data, they do not have the needed informatics expertise to provide suited interactive data retrieval systems.

It is to address these issues that the CUAHSI Hydrologic Information System was developed providing a series of relevant services, as reported in the website:

- Data discovery — A map-based viewer will display the locations where data collected by various entities in one location, including both government-collected and university-collected data.
- Data delivery — Through the use of programmatic calls, users will be able to retrieve data directly into databases, spreadsheets, and analysis packages using a single syntax regardless of data source.
- Data publication — Academic scientists will be able to easily publish data they collected so that appears within the common data viewer and responds to the same data retrieval calls as government sources.
- Data duration — A HIS Data Centre (HISDAC), will provide a repository for archival data. These data can be viewed and delivered using the same mechanisms as described above so that their delivery is seamless to the user.

It is important to report here this list of services because, these kinds of services are also a basis for the research of this thesis and represent a similar architecture to the one of the information system developed in this research.

WATERWARE

WaterWare is a very typical example of a project financed by the public sector consequently to the establishment of a new normative, the EU Water Framework directive.

As stated in its website, WaterWare (http://www.ess.co.at/WATERWARE/) is an integrated, model-based information and decision support system for water resources management. Given the complexity of the applicability of the normative, the EU decided to finance a series of related projects, trying to increase the level of knowledge needed for integrated management of the resource and related availability of software support systems. Therefore WaterWare integrates results of the EUREKA projects EU487, and the related cluster of projects of the Framework Programme of research financed by the European Union; The EU financed both the development of the system and its pilot applications.

Therefore, the information system was developed with the objective of supporting the responsible institutions to implement the European Water Framework Directive or similar national legislation (http://ec.europa.eu/environment/water/water-framework/). The system has been developed through a series of applications to: River Thames in England; Lerma-Chapala basin in Mexico; West Bank and Gaza in Palestine; Kelantan River in Malaysia; Yangtze River in China. Applications around the Mediterranean with other EU sponsored projects include river basins in Cyprus, Turkey, Lebanon, Jordan, Palestine, Egypt, Tunisia and Morocco.

Considering the software architecture, WaterWare is very similar to the IS developed in this research. The architecture is open and object-oriented with a client-server orientation, web-based. WaterWare integrates a series of different components needed for water resources management oriented information systems such as databases, GIS, simulation and decision support models and analytical tools. The client side is implemented through a graphical user interface integrated with a GIS and through hierarchical map layers. On the serves side the system relies on object data bases, time series analysis, reporting functions, an embedded expert system for estimation, classification and impact assessment tasks, and a hypermedia help and explain system.

The system can perform real-time data management, simulation and decision support modelling, supporting data storage, forecasting, and reporting, and support for operational management that can be provided with a real-time rule-based expert system.

2.2 Analysis and conclusions

As it is straightforward to appreciate even with a rapid research through the relevant literature, there is a very long list of projects which could be relevant to this thesis. The quantity, complexity and variety of those are extremely important. Therefore it is not possible to carry out an exhaustive analysis of the list of those projects. Such an analysis would have been too time consuming and out of the scope of this research.

Besides it is important to mention that other projects, not present in the above list, have been used, analysed or taken as examples or model to develop pilot applications for this research. These other projects have been detailed in the appropriate relevant sections either concerning technology analysis or pilot applications.

The criteria to select the above listed projects have been based on the size and the importance of the projects and/or of the institution responsible for its development or use. Most of these projects come from the public sector, sometimes public/private at the same time, with few exceptions. This is because the environmental sector is a public concern on its own nature.

Although the variety of technology used, the variety of actors involved and the different requirement of each project developed, it is possible to extrapolate some general ideas from the list and from its analysis.

From the technology point of view:

- technology involved are many and varied,
- data sources are heterogeneous,
- there is an high level of IT knowledge involved but often scientist of this field do not have the needed expertise,
- there is a strong tendency towards standardisation and openness.

From the organisational and project management point of view:

- given the integrated management approach to water resources which nowadays is often normative, the number of institutions involved is quite high,
- given the public scope of the software developed often the public sector if the funders/client and final user,
- as it is happening in general in the IT world, always more and more institutions are going towards open source and towards defining a community of users and developers,
- even if the water resources management tends to be a niche sector, the availability of projects and experiences is important.

In this review we focused on the analysis of projects related to information systems and modelling frameworks. Nonetheless even though hydroinformatics has been a topic of active research for more than 20 years, it seems that inadequate consideration has been given to research on architectural design dedicated to information systems and modelling frameworks and to the associated software development (Harvey, Han, 2002). This has changed only in the last years, as above showed, under the impulse of the new approach to water management (IWRM) and related legislations mentioned in the background paragraph (first chapter).
Even under this new impulse, the focus of research in the field was mainly oriented to modelling or to the development of commercial "closed" framework information systems trying to embrace as many requirements of IWRM as possible. This often resulted in an untargeted increased complexity. This "untargeted increased complexity" could come for instance from coupling underground, surface and atmospheric phases of the water cycle when maybe not all these functionalities at the same time were needed for all users. This is because a closed commercial framework (or information systems) to be as most competitive as possible in the market would probably need to be as complete as possible.
To address all needs of water resources management at the same time is obviously too complex. This complexity has resulted into an increasing demand for tools, which integrates or/and allows the interaction of software tools and models of different origins, and with different purposes. Besides, this increasing complexity makes systems always more difficult

to be used influencing performances and costs for human resources training needs even for developers or for user specialists.

Consequently the idea of a software toolbox has become constantly more attractive (Wasson et al., 2003). According to this solution, integration and interoperation of different components will be organised and tailored in order to adapt the overall system to the final application.

Usually different tools based on the same equations and algorithms and with similar data formats are customised and to be used for different purposes or area of utilisation. This dispersion of efforts could be avoided for instance structuring the tools in two different parts, the core containing equations and algorithms (the model) and an external layer that could be customised according to its final use to deal with other tools and to be managed by the final user (Wasson et al., 2003). These suggestions and experiences are close to many kinds of similar problems that it is possible to find in the IT literature.

It may be concluded that similarity can be found among various researchers like Harvey (2002) with a dissertation on models structure and models coupling.

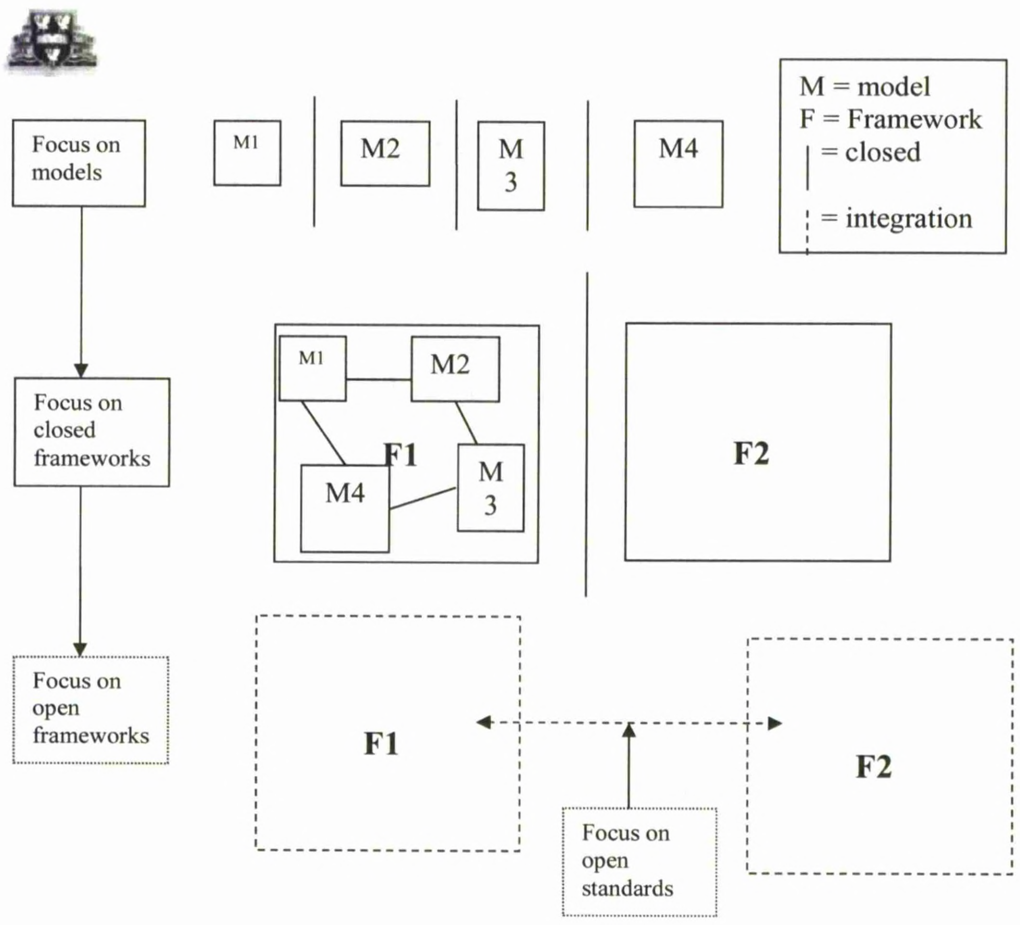


Figure 2.1: evolution of modelling integration

In figure 2.1, our interpretation of the evolution of modelling integration in modelling frameworks (or information systems) is showed. The three phases show a change of focus from the models themselves, to the framework, the openness and standardisations which are considered in parallel.

Considering the last evolutions, at European level the best examples of this kind of projects are HarmonIT and OpenMI, which aim at defining a standard modelling interface and environment to simplify the linkage of models related to water resources management.

Even recognising the importance of this kind of works, which are at a higher level of the research described in this thesis, both in terms of human and financial resources, we thought it would be useful to give our contribution approaching the problem of software integration in IWRM from a similar point of view and bringing additional reflections to the debate.

The definition of an open standard would be a fundamental step in making hydroinformatics able to deal with integrated resources management (see OpenMI), but at this early stage of research on system integration in hydroinformatics towards IWBM, a thorough work would also be needed on the interoperability.

To clarify this difference between standardisation and interoperability it is interesting to cite Shirky (2001), who in his work "Interoperability, Not Standards", oriented to informatics in general, says that the coupling of standards and interoperability is the default for any widely dispersed technology. However, there is one critical period where interoperability is not synonymous with standardization, and that is in the earliest phases of work, when it is not entirely clear what, if anything, should be standardized and how it should be done.

The way we chose to undertake our research on software interoperability in IWRM is the way of openness and standard open technologies. We tried to make use of all already existing open standards to try to develop the different components of the system as more interoperable as possible. Besides, we tried to build an information system keeping its structure and its architecture as most open and open standard based as possible, always giving the right important to the quality of the technology or software component and tools used.

On the link between open standards and interoperability it is interesting to know also what a once "the closed" standard oriented software multinational, IBM (www.ibm.com), states on the subject:

"Open computing standards and platforms mean much more than just helping to reduce licensing costs by eliminating the need to buy and manage proprietary solutions. Openness addresses the need for interoperability and provides for the broadest choice of IT solutions to meet business needs. It can also help incorporating technology innovation more seamlessly to meet the needs of any IT environment" (www.ibm.com/systems/why/openness/).

All these points have been taken into account as the frame and the background for the development of this research and have strongly influenced its approach as shown in next sections.

CHAPTER III

3 METHODOLOGY AND TECHNOLOGY ANALYSIS

3.1 Information System Background

The aim of this chapter is to research into the design of an information system for IWRM starting from the design of its architecture/structure and its components, considering technological solutions and integration through open source technologies and (if existing) open standards. The reason to perform this "integration" exercise, as explained in the previous chapter, is to research into the "interoperability" of models and software tools towards the integrated management of water resources. Therefore, the architecture of the IS shall be designed to integrate the most appropriate, innovative and open technologies for IWBM making them able to interoperate.

As discussed in the literature review, a hydroinformatics system for IWRM consists of many different components. Since, decision makers and policy implementers need to retrieve, store, analyse and present always more and more data, this amount of heterogeneous information requires the development, use and interoperation of an increasingly wide variety of tools. Whenever, a developer has to design an information systems, the most appropriate tools for acquisition, analysis or/and presentation have to be selected on a project basis by considering characteristics of the catchment area and the requirements of end users such as decision makers. Generally this is not possible with most of the suites and monolithic analysis packages presented in previous chapters where tools are fixed, closed, not exchangeable and not modifiable. This is the main issue that this technology analysis aims at resolving.

The technology screening process is then followed by the development of consistent interfaces between each of the components chosen for the information system. These interfaces are not usually provided as a native attribute of the tool. Therefore, at this stage the use of standard open technologies helps to a great extent for the work of developers. In fact, open source code and projects can be modified, adapted integrated as needed.

The requirements identified in the previous paragraphs (1.2) for the IS are integrability, extensibility, portability and open standards based. Recalling that integrability refers to the capability of the different components of the IS to work together, extensibility to the degree to which the IS can be extended to meet new goals. Portability allows the information system to

work in different operating systems and hardware environments while the use of open source technologies will help meeting the first three requirements as showed in the next sections.
Following these requirements and introducing the IS architecture, the information system can be seen as a general framework embedding the following common kind of subsystems: Databases, Simulation Models and Management Models, GIS, everything connected and manageable, in different ways, from the User Interface (UI) level. The GIS intervenes at the UI level and at the simulation models and management models level.
This information system shall operate as a platform which can facilitate standard formats for data exchange and code sharing. Besides given the variety and quantity of end user organisations involved in the management of water resources, other important specific requirements of the system are portability, meaning the ability to function in different OS and hardware environments, and Web orientation of applications with possible developments through distributed computing.

3.2 Modelling environment approach

In informatics, frameworks, also called modelling environments, support modular and integrated model development providing the structure where environmental modelling modules can be integrated. In general they aim at a dynamic modular approach to information systems strongly oriented to modelling. More flexible and complete frameworks allow integrating reusable tools for data management, analysis and visualisation.
In informatics, the term framework is also sometimes used for underlying classes and libraries (intended as in programming languages). In this case the framework is included in a system which uses these software frameworks to support module development, model construction, and execution.
We prefer to use the term framework in a broader sense including software tools, its architecture as well as modules, modelling tools and its data model.
Frameworks have a series of characteristics and requirements related to their most important feature, their architecture. Since frameworks are in fact integrating environment their

performances are based on their architecture and linking capabilities as protocols and methods of operation.

Environmental sciences related frameworks find their reason to exist facing the integration issue in trying to monitor and represent a complicated and dynamic system such as the environment itself. For instance, if taking the scientific research approach, individual researcher and research groups in specific environmental disciplines, such as ecology and hydrology, tailor models' architecture for their own purposes focusing with a specific objective to a particular concern of modelling potential.

This approach to model and modelling framework development, based on *ad hoc* situations on a project basis, even considering just the field of hydroinformatics, leads to a large number of different approaches to models' architectures and modelling framework architectures. These models have different structures, algorithms, input fields and boundary conditions which are not compatible or interoperable. When dealing with complex problems requiring the combination of multiple models or even frameworks of models, these are difficult to integrate and an important effort in harmonisation is needed.

With the stronger and increasing impact that informatics has now on other scientific disciplines, more researchers and practitioners are trained in modern software engineering techniques. Besides, more information technology experts have become specialists in other fields like in our case environmental science disciplines. This exchange trend between IT and environmental science led to the development of standardised approaches to modelling in the environmental modelling problem domain.

Models dealing with discrete time and discrete entities can be structured through approaches based on object oriented logic, and are consequently managed using appropriate software engineering design and development methods.

These new approaches have brought to the development of the concept of environmental modelling frameworks with the opportunity to bring together a modelling suites or libraries of modules, with architectures designed to well represent basic and natural features of environmental problem situations (e.g. Reed et al., 1999). The advantages of using frameworks to treat the environmental information of a water catchment include standardisation of features such as data manipulation and analysis, interoperation and exchanges between models and data sets, standardisation of structures and coding of modules, and integrated visualisation of model outputs.

A range of environmental modelling frameworks exist and have been under development for some time in the last 20 years. Beside the examples cited in the literature review chapter which are more recent, other interesting projects, many of them already discontinued but in anyways contributing to the advancement of the filed include: the Dynamic Integration Architecture System (DIAS) (Sydelko et al., 2001), the Interactive Component Modelling System (ICMS) (Reed et al., 1999), the Spatial Modelling Environment (SME) (Costanza and Voinov, 2004), TIME (Rahman et al., 2004).

It is interesting then to describe in detail an example of modelling framework. Probably the most relevant example of implementation and concrete application of a modelling framework in the United States is the case of the United States Department of Agriculture as national intersection point of all information at basin level. As typical for an institution that worked long time at basin level, the USDA's Agricultural Resource Services (ARS) had more than 100 models for a variety of purposes that had been developed on a case-by-case basis, using the technology that was available at the time of development (Gonsalves, 2007). As it happened in Europe, when at a certain point the policy focus was re-oriented towards IWRM, the USDA's Agricultural Resource Services started facing the integration problem in terms of management, performances and financial issues with a disorganised amount of knowledge; difficult and expensive to maintain and to adapt to the needs of the new policy. Efforts to edit and improve the models making them interoperable on an ad-hoc basis were complicated. The result of the reflection USDA's Agricultural Resource Services to address these problems were that a new object-modelling system that supported Web-based team communication was needed with the series of features such as issue tracking, user access permissions, collaborative code and document management. All this system had to take into account the need for supporting USDA's existing systems and not losing any of the knowledge that was collected during many year of work concerning algorithms, functions, modelling etc. all related to water resource management. Besides, one of the major requirements of the system was that it had to be easily installed and maintained.

Therefore, a modelling framework called OMS (object modelling system) was implemented in a platform consisting of an open-source development tools from Sun Microsystems (NetBeans, www.netbeans.org) supported by collaborative development and distributed computing software such as Intlands CodeBeamer (www.intland.com) and Subversion from Collab_net (www.open.collab.net). The system developed allowed ARS to reach the level of collaboration fundamental for an institution working with the IWRM policies. In particular the system was able to mange up to 300 simultaneous software development projects at the same time. More than having just a framework for modelling integration with a step forward they developed an integrated development infrastructure allowing OMS developers to collaborate, share knowledge and work effectively as a team from different research institutions and/or locations.
It is important to highlight that it is not just about developers working on code, there are also data analysts and sector experts, as for instance on geospatial processing, fixing the parameters of the models based on data sets such as elevation and vegetation layers. Some of them are pure scientists who are not trained in coding. The aim of the system is to support all of them.
We noticed that one of the first real concrete implementations of the IWRM management requirements has been developed using open and cross-platform technologies such as JAVA and NetBeans. In fact efforts were made to develop scientific models and migrate to open and cross platform technologies as described also in some cases in the literature review.
While the conceptual approach of the OpenMI is the most complete response to integration problems that allows existing models and tools to be linked together, approach of the USDA's ARS is a practical response to a concrete integration problem that allows a continuous and dynamic development focusing more on the content of the scientific models than on the time consuming IT side.
Besides, application of theoretical concepts to practise underlines the fact that "closed software" (typically commercial software) are *per se* obviously less oriented towards integration. When such an extensive integration exercise as the one needed in the case of IWRM is undertaken, it is probably more economical and scientific to develop open source software. Also, the degree of *ad-hoc* solutions needed is so high that it would not be possible

for a single private entity (firm, institution, etc.) to develop software able to encounter all the functionalities needed for IWRM. Finally, the degree of standardisation in the actual IT market in the field is not enough to easily develop automatic integration of different commercial closed solutions.
Another point in favour of this solution is that the OMS was also successfully used for applications in Europe in line with the EU Water Framework Directive (Kralisch et al., 2005).

3.3 Software technologies and system architecture

An information system supporting decisions for IWBM encompasses a wide-range of software applications with a precise morphology. The core part is constituted by a series of analytical models and for the information system to work, these models need to be connected to a data layer, typically a database management system. If the data layer can be considered as the roots of the tree, the core applications the trunk, towards the higher superficial level the branches are represented by the graphical services and spatial analysis models, typically defined as Geographic Information Systems. The different software components which fit in an overall framework (as previously described) are developed to allow component interoperability. This structure allows the information system to become an advanced decision system across a wide range of actors (users, managers, etc.) involved in the "life" of the catchment area. An overview of software components and related technologies typically included and needed in a hydroinformatics information system is given in following sections.

3.3.1 Database Management Systems

The Data Base Management System (DBMS) is the foundation of almost every modern business information system. Virtually every administrative process in business, science or government relies on a data base. A database is a reflection of the world, a reflection governed by the beliefs, axioms, and perspectives of the database designers and implementers as well as by the context of the application at hand. As such, a database inherently encapsulates a view of what should be modelled and represented, as well as the relationships between the real-world and the modelled data entities. The decision on what should be modelled and how such a model relates to a real world entity is in the hands of the database designer. This characteristic of a somewhat subjective human perspective is likely not to become an obstacle if a single coherent database design and implementation process is carried out. In such a scenario, the application and the database according to which it was tailored comprise an internally consistent reflection of the real world, thus providing a consistent ontology and semantics framework. This situation changes radically when attempting to utilise different databases that were designed and implemented in several different contexts. While each of these databases is internally a consistent reflection of the world, this consistency is not maintained between the databases due to a lack of a common agreement on the meaning and the representation of data. In such scenarios, semantic heterogeneity is often inevitable.

From the informatics point of view, a data base management system is a very complex piece of system software. A single DBMS can manage multiple data bases, each one usually consisting of many different tables full of data. The DBMS includes mechanisms for application programs to store, retrieve and modify this data and also allows users to query it interactively to answer specific questions. Specialists, known as Data Base Administrators (DBAs) control the operation of the DBMS and are responsible for the creation of new data bases and the definition of the table structures used to store data.

In the case of this research the focus is on spatial databases. A spatial database can be defined as a collection of spatially referenced data that acts as a model of reality. The concept of GIS and spatial databases are strongly related. The actual trend of spatial databases is related to the distributed geospatial infrastructure paradigm which continues to emerge as a powerful and versatile framework, making spatial database systems more distributed. Moreover, with the increased number and availability of spatial databases, the incentive to utilize various different data sources is greater than ever. Semantic heterogeneity and its consequential ambiguities are therefore almost inevitable.

In particular, addressing water resources management issues, there is a need to focus on reliable and adequate environmental data. What is needed is a careful approach to database design that goes through the identification of data requirements. Some of the information parameters needed specifically for the water resources management are shown in table 3.1 for instance for the case of information systems for water supply and for wastewater management.

Acquisition	**Treatment**	**Delivery**	**Support**
Water Flows, Water Balances	Operational records	System flow and pressure	Financial database
Water quality	Reporting records	Water quality	Engineering records
Water sales	Maintenance records	System inventory and maintenance	Administrative records

Table 3.1: Example of data needed for Information Systems for water resources management.

As shown in table 3.1, parameters of the IS cover a wide array of fields or disciplines related to water resources management. A well-designed database for WRM should enable the water manager or the decision-maker to have coherent, clear, and concise information about the water sector in general.

In particular, concerning data required for water quantity and quality analysis and management in watersheds typically include:

- Climatic factors such as temperature, wind, solar radiation, and rainfall;
- Geomorphic and land-use information such as slopes, drainage density, geology, soils, land covers, channel cross-sections, and groundwater depths;
- Hydrologic data that include flows, water levels, depths, and velocities;

- Point pollutant loads from point sources such as large industries, cities, and wastewater treatment plants that discharge their wastes into surface waters at a specific locations;
- Diffuse loads from nonpoint sources that enter surface waters along an entire stretch of the river;
- Ecological attributes including an inventory of existing habitats and their condition;
- Water quantity and quality demands over time and space that in some cases can be compatible and in other cases conflicting; and
- Information on the institutional framework in which management decisions are to be made, such as laws pertaining to the **allocation** of water to various users and the various standards set by public health and environmental agencies.

One important definition that we will look into this research is the difference between Relational DBMS and Object DBMS. A Relational database management system maintains a set of separate, related files (tables), but combines data elements from the files for queries and reports when required. The concept was developed in the early 1970s to accommodate users' ad hoc requests for selected data. The most used DBMS now in the market are relational database management system, including Oracle, DB2, SQL Server, MySQL, etc.
Alternatively object database management systems are closely aligned with object-oriented programming languages and enable the data in the objects to be persistently stored without requiring conversion to relational database logic and structures.

3.3.2 Modelling theory

The concept of modelling and models, being central to this work, had already to be approached in the previous sections of the thesis, meanwhile this paragraph will be complementary and specify some of the concepts already presented.

Models are of central importance in many scientific contexts and in particular in environmental sciences where highly complex natural phenomena are difficult to be monitored and need to be modelled for understanding and representation. In the literature review, there is already an extensive list of possible water quality and quantity models that can be integrated in an information system for IWBM. Concerning our research, a model can refer to many different kinds of tools to be integrated in the information system.

From a theoretical point of view, we can first define the real meaning of what we intend as model and modelling and the relationships with the various related software tools. In modern logic an abstract model is a theoretical construct that represents something, with a set of variables and a set of logical and quantitative relationships between them (Frigg, 2006). Khatibi (2002), for instance, argues that "a model is the dataset of a physical system described by a particular modelling technique". This definition, probably very similar to the definition of algorithm, identifies a model as a mathematical description of a process that together with a set of particular parameters fit the description of a particular physical system to represent. On the other hand, the modelling technique is just the mathematical description of that process in general, its conceptual abstraction.

Researchers studying catchment are used to the difference between formal models of some process in general and the application of that model representing a particular natural phenomenon. The ability to make this distinction allows modellers to choose some set of parameters to fit the general model to a natural system (Harvey, 2002).

In managing catchment, for instance, a formal unit hydrograph model of the excess rainfall–runoff transformation represents this natural process as linear, time invariant, and independent from spatial distribution of rainfall (Shaw, 1994). To customise the model for the relationship between excess rainfall and runoff in a particular catchment, it is necessary to provide parameters defining the shape of the unit hydrograph.

In principle, as one of the kind of tools required for the information systems, models have to fit all general requirements already defined as integrability, extensibility, portability and open standards based. Therefore, the real topic of this research is not related to modelling *per se* but more precisely to models integration, which is consequently the central requirement to analyse. Given that even a single simple model includes a set of different sub-processes (operations), the distinction between a single model and the point at which a model is the result of the integration of two or more models can not be other than subjective. Hence, in theory the distinction between models and integrated models seems to be meaningless. As presented in several examples in the literature review and in previous chapters, software for water resources management and modelling (the correct term would be modelling framework or environment) often embed a series of different models towards a certain objective. As it has been argued, these software are not designed to be integrated or interoperated with other models. Therefore, in this case the definition of the process of integration of different models becomes useful.

Moreover, the process of integration of two (or more) of such existing models from the hydroinformatics point of view is conceptually and practically different from designing and implementing a single model embedding two (or more) sub-processes.

To clarify this concept, considering a phenomenon called A happening in a water catchment (i.e. rainfall) and a phenomenon called B (i.e. runoff), the phenomenon C (rainfall-runoff) is the result of A+B. We can have a software model for A and a separated software model for B, integrating the two, we take the output of A and use it as input for B the output of B will be the output of A+B = C. In the same way we can have (as it is often the case for rainfall-runoff models) a single software model for the phenomenon C. The main difference between the two approaches (apart from possible algorithm simplifications) from the IT point of view is the work needed to make the output of A compatible as input of B.

In particular, in environmental modelling, given the complexity of the process to be represented by the modelling exercise, model development is always at a certain level a model integration issue. Besides, integration can not be static. For every particular purpose of the information system design, a different combination of models would be required. This can happen not only for the purpose of adapting an information system to different water basins, but in the framework of different activities for the same basin.

Often in IWBM more than just water quality and quantity models, other kind of environmental and socio-economical models are required. Nevertheless the case study approached in our research will be specifically water quantity oriented.

3.3.3 Geographic data management and information system

Geographic data play an important role in the framework of information systems oriented to environmental data management. In the European context, the directive of the European Union on Infrastructure for Spatial Information called INSPIRE (INSPIRE Directive 2007/2/EC), states that: "Community policy on the environment must aim at a high level of protection taking into account the diversity of situations in the various regions of the Community. Moreover, information, including spatial information, is needed for the formulation and implementation of this policy and other Community policies, which must integrate environmental protection requirements. In order to bring about such integration, it is necessary to establish a measure of coordination between the users and providers of the information so that information and knowledge from different sectors can be combined." This directive not only highlights the importance of the usefulness of spatial information for environmental protection but also defines the spatial information as fundamental. Therefore, emphasis has been given for standardisation of data management procedures at European level.

On the other side of the Atlantic, in the United States, a Federal Directive (USA, Executive Order 12906), with a more concrete approach, requires directly the development of a National Spatial Data Infrastructure defined as "the technologies, policies, and people necessary to promote sharing of geospatial data throughout all levels of government, the private and non-profit sectors, and the academic community". With an innovative and open minded approach, the directive states that sharing geographic data among all users could produce significant savings for data collection and use and enhance decision making. The final goal of all these governmental policy initiatives is harmonising to reduce duplication efforts among agencies and institutions by improving quality and reducing costs related to geographic information, to make geographic data more accessible to the public and thereby increasing the benefits of using available data.

In the private sector, management of geographic data has seen one of the biggest joint standardisation initiatives of the software industry, the Open Geospatial Consortium (OGC) (www.openspatial.org). OGC is an international industry consortium of 352 companies, government agencies and universities participating in a consensus process to develop publicly available interface specifications. It is the most important effort at global level to standardise the use of geographic information in the software industry and in particular the standardisation of geographic information system technology (GIS). The OGC uses as definition for GIS the definition of the Association for Geographic Information (AGI) glossary (www.agi.org.uk/) affirming that a Geographic Information System is "A computer system for capturing, storing, checking, integrating, manipulating, analyzing and displaying data related to positions on the Earth`s surface".

Citing one of the many existing definitions, Geographic Information Systems are databases that have a spatial component to the storage and processing of the data. Hence, they have the potential to both store and create map like products. They also offer the potential for performing multiple analyses or evaluations of scenarios such as model simulations (Lyon, 2003). Typically, a Geographical Information System (or Spatial Information System) is used for handling different kinds of maps at the same time bringing complete thematic information to the end user. These might be represented as several different layers where each layer holds data about a particular kind of feature. Each feature is linked to a position on the graphical image of a map.

From the IT point of view there are many different ways/technologies to deal with data. In general, to define geographical information in a very simple way, it is about data that can be stored in one or multiple files, each file containing a set of a reference to a coordinate system. The reference is used to define the position for each data entity. Each entity has one or multiple features which are also stored, typically as attributes.

When a database of individual files is developed and the combined files may contain characteristics or attributes such as, in the case of hydroinformatics, water or soil chemical sampling monitoring information, ownership, time series information, river locations, topography, management procedures, point sources, and any other information that can be collected and have a reason to be stored for the analysis, modelling, representation, etc.

Each variable or set of variables can be included in a layer. This layer can then communicate to the applications, for instance with the kinds of data mentioned above, on the characteristics of water resources or watersheds.

Being not just a database with geo-referenced data but a complex information system *per se*, an important capability of GIS is the fact that it can directly be interfaced with models for simulation of physical, chemical, and biological processes. The models required should have the capability and the need to take spatial and/or multiple file or "layer" data as input to computations and to return results also considering the spatial reference. Therefore, GIS can potentially be used both with deterministic or complex models based on algorithms simulating processes and it can also be interfaced with statistical models. The condition in both cases is that data have spatial reference.

In GIS usually layers (fig. 3.1) of heterogeneous data can be structured in a way that it is possible to perform statistical analysis. In general these data are used for town planning, local authority and public utility management, environmental, resource management, engineering, business, marketing, and distribution (Maguire et al., 1991).

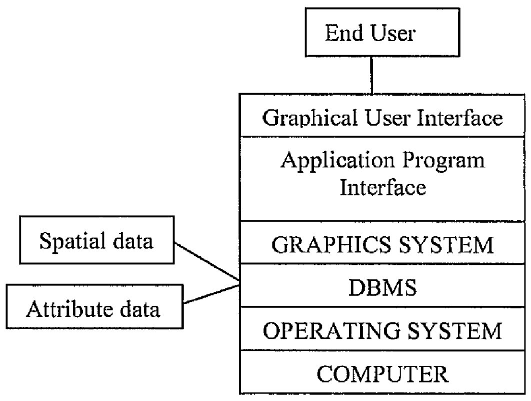


Figure 3.1: the physical component of a GIS according to AGI, Association for Geographical Information (http://www.agi.org.uk/).

According to ESRI (1998) a GIS is a collection of computer hardware, software, and geographic data for capturing, managing, analyzing, and displaying all forms of geographically referenced information. Besides ESRI states that with a geographic

information system, it is possible to link information (attributes) to location data, such as people to addresses, buildings to parcels, or streets within a network. It is then possible "to layer" the information to give a better understanding of how it all works together. The information can be quickly visualised if right combination of layers are chosen.
Following ESRI definition, a GIS can provide three different ways of supporting the problem-solving process:

1. **The Database View:** A GIS can be used as Graphical User Interface for a geographic database or geo-database. It is itself an "Information System for Geography".
2. **The Map View:** A GIS is a series of improved maps and other views that show features and feature relationships on the earth's surface. Maps of the underlying geographic information can be integrated in different layers and used as "windows into the database" to support queries, analysis, and editing of the information.
3. **The Model View:** A GIS is a set of information tools that derive new geographic datasets from existing datasets. These **geoprocessing** functions take information from existing datasets, apply analytic functions, and write results into new derived datasets.

All the three applications of a GIS are fundamental to an information system for IWBM. Combining data and applying modelling algorithms, the decision support system will support mangers and decision makers.
Concerning the particular scope of our research, GIS are particularly relevant and have a rapidly increasing role in the field of hydrology and water resources development (Lyon, 2003).
A number of GIS applications to water resources and watersheds have been completed over the years. Many of these efforts have resulted from the need to address difficult-to-achieve project goals. GIS applications are applied due to the variability of the resources over time and space, and the number of variables that must be evaluated.
GIS links geographic information (where things are) with descriptive information (what things are). Unlike a flat paper map, where "what you see is what you get," a GIS can present many layers of different information (Maidment, 2002).

GIS also allows for advanced analysis and modelling methods to be implemented in support of research efforts. The use of simulations models provides detail on water movement and transport of materials. GIS databases and technologies allow for optimization of model results. The running of model scenarios supplies detailed information for the development of plans, management decisions, and informed leadership. Repetitive processing facilitates predictions and forecasting of events. Certain characteristics of and applications in water resources research led themselves to GIS databases and GIS analyses. Water flows downhill and supplies a directional characteristic to the modelling efforts. The flow can be determined by gauging, and the simulations of water discharge and water quality at a gauge can be compared with the field measured value at the gauge point.

The history of watershed and water resources research has included many models that utilize spatial data. These "traditional" models incorporate data with a spatial basis or the components of spatially averaged data. This is because applications have been dictated by needs, and models exhibit sensitivity to these variables. The future promises a variety of enhanced applications of GIS and allied technologies for water resources and watershed research. We will see better integration of data within databases, and the advent of more uses of three-dimensional visualization of data. Databases will become available in greater numbers and detail, and will be procured through the Internet and the World Wide Web.

The advantage of using GIS for watershed studies has been recognised (Lyon, 2003). It is evident that the capabilities of GIS are potentially valuable in a number of efforts. Over time the need for information resulted in the development of appropriate algorithms to facilitate the utility of GIS and statistical or deterministic models. Traditional modelling and analysis preceded the advent of GIS and many efforts are ongoing to join the approaches and optimise their capabilities.

3.3.4 Object oriented programming language

Last but not the least, the selection of programming language is a fundamental step to undertake for all hydroinformatics research and in this case for the software technologies of an

IWBM. Research on programming languages is more than just a part of informatics and can be approached as a science on its own in terms of importance in the scientific and business world. Consequently the relevant scientific literature is vast and extremely complex. Here we expose some of the concepts relevant to our research that have been analysed. The programming language can be seen as the element able to glue and communicate all the tools and technologies already presented. Besides, as core element, the choice of the programming language is influenced by the information system requirements. The general requirements we identified earlier are: integrability, extensibility, portability and open standards based. In general there are numerous examples in the scientific and technical literature using object oriented logic for modelling and information management in environmental sciences. Focusing on our research, for the particular case of hydroinformatics, it has been demonstrated in a series of case studies, as for instance in Spanou and Chen (2000, 2001 and 2002), in Maidment (Maidment et al., 2002), etc., that object oriented approaches offer great benefit to river water quality and catchment hydrological modelling.

It is also straightforward to support from a conceptual point of view the choice of object oriented logic for an information system dedicated to water resources management and modelling. We can simply argue that in informatics the comparison between models and object logic is quite straightforward. For instance in Java, a class is a blueprint or prototype from which objects are created. In object oriented terms, a particular model is an instance (or object) of the class of objects known as models. An object is a software bundle of related state and behaviour. Software objects are often used to model the real-world objects, giving a more direct and easy to understand structure than for instance the procedural programming where statements are written in the form of a batch.

In the concrete implementation process of the information systems, this parallelism between objects and models demonstrates to be even stronger, as simple "models" can be directly implemented as single objects. A set of objects can also be seen as more complex model. In terms of object oriented language during a simulation, a running instance of the model implementation exists.

This strong conceptual and concrete parallelism between models intended as in environmental sciences and objects of in object oriented language justifies on one hand the utilisation of this technologies for water catchment management in our IS and on the other hand the fact that we consider at the same level a modelling tool or a modelling application. Particularly interesting is the approach of Keogh and Giannini (2005), which explain the object oriented logic beginning from a point of view which is particularly relevant to environmental modelling: "how we see the world". They argue that the real world is made of objects and objects must be represented.

The fundamental concepts that are related to object oriented programming languages and that are strictly related to the history of their development have been identified by Armstrong (2006). In a the research entitled "The Quarks of Object-Oriented Development" she analyses nearly 40 years of computing literature in order to find a number of quarks, or fundamental concepts, common to the strong majority of object oriented languages and to their definitions . They are the following:

- **Class**, a class defines the abstract characteristics of an object, including the object's characteristics (defined as its attributes, fields or properties) and the object's behaviours
- **Object**, a particular instance of a class.
- **Method**, an object's capability.
- **Message** passing, the process by which an object sends data to another object or asks the other object to invoke a method. Also known to other programming languages as interfacing.
- **Inheritance** is a feature of a subclass, a more specialised version of a class, which inherit attributes and behaviours from their parent classes, and can introduce their own.
- **Multiple inheritance** is the inheritance from more than one parent class, neither of these ancestors being an ancestor of the other.
- **Encapsulation**, which conceals the functional details of a class from objects that send messages to it. Encapsulation is achieved by specifying which classes may use the

members of an object. The result is that each object exposes to any class a certain interface — those members accessible to that class.

- **Abstraction**, which is simplifying complex reality by modelling classes appropriate to the problem, and working at the most appropriate level of inheritance for a given aspect of the problem.
- **Polymorphism**, which allows treating derived class members just like their parent class's members. More precisely, Polymorphism in object-oriented programming is the ability of objects belonging to different data types to respond to method calls of methods of the same name, each one according to an appropriate type-specific behaviour.

These concepts are not common to all object oriented programming languages, but most of them apply to the major languages as Eiffel, C++, C#, Java (Joyner, 1999).

3.4 Human Interface and the Management of Information

Management of information systems became an overall challenge with the advent of new technologies. Being information *per se* complex to be managed, management of the information systems area is even more complex and unpredictable. A great impact on user's effectiveness of management of information systems is related with the efficiency of the "human - computer interface". According to one of the first definitions in modern computing (Shneiderman, 1986), the human - computer interface refers to "the way a person experiences the computer, its application programmes, hardware components, output devices and functionality. It includes all aspects of the human's experience from the obvious ones of screen layout and selection options as well as input and output devices, reliability and accessibility."

It is then obvious that technology and software development, nowadays ubiquitous, come with the need to increase human user effectiveness of information management. According to Barrier (2002), organisations are beginning to understand that information systems are not just

a monolithic and static tool that improves efficiency but a dynamic and heterogeneous instrument which need to be managed.
After the first phase of development of IT (in terms of innovation management principles), it became anti-economic to accept pre-conceived components into information systems without analysing their effectiveness, feasibility and efficiency and determining in advance the requirements. In fact, given certain characteristic of the software and IS market and of organisations' approach to information technologies, it happened and happens often even nowadays that end users are adapting their skills and workflow to features of information systems. This is true above all for the most sensitive part of the human- computer interface, the Graphical User Interface (GUI). This is often the cause for a steep learning curve for the end users and consequently the need for a strong effort to adapt and time consuming training activities. In the field of IWBM, where many different organisations (Ministries, Agencies, utilities, users, etc.) are involved, GUI implies a great challenge regarding data management. If we also follow the participative approach, as one of the principle of the IWRM (Dublin, 1992), at consultation level we should allow the greatest number of people to access the information system for the largest number of purposes and in the widest number of contexts. In general, the list of queries regarding the physical requirements for human – computer interfacing could be infinite. The common solution in designing requires that organisation must determine in advance IS features.
The use of human Interface for management of information is particularly relevant to this research. West (2000) states that GIS are powerful information management systems particularly suitable for decision support and end-user application. However, complexity in data accessibility, cartographic principles and analysis do not make them fully user friendly. The solution is to address a series of mitigation measures in order to increase and simplify the usability of the GIS interface always in line with identified requirements.

3.5 Open information technologies

3.5.1 Open source technologies

In the era of information technology, the development of open source has made rapid "movement" and playing even greater role day by day, directly and indirectly, a bigger portion of the IT market. The 2007 report from IDC, one of the biggest software market analyst companies in the world (www.idc.com), affirms that open source is one of the biggest, most important trends in software in the last twenty years "and it's only gaining momentum". A senior analyst from IDC working on global software research argues that open source is used in most of all organizations worldwide and includes hundreds of thousands of projects (Gillen, 2009). The study, including thousands of developers from 116 countries, found that open source is in production in over half of the organizations surveyed and in use in 71 % of them. Also, despite the several open source licenses in existence, IDC affirms that only three kinds of revenue models work with vendors and in the market: the software revenue model, the public collective model and the service broker model. Besides, after the 2009 world economy recession, the impact of open source and free software on the market is expected to grow even more.

SUN Microsystems (www.sun.com), one of the biggest IT companies promoting open standards, focuses the definition of open standards and openness to the impact on the market. They argue that the best test for "openness" for a technical standard is its ability to promote competing and interoperable implementations. Also supported by SUN is the definition of Ghosh (2005) which, in its paper on economic analysis of open standards; affirms that "open standards should be defined in terms of a desired economic effect: supporting full competition in the marketplace for suppliers of a technology and related products and services". A standard can be defined as open only if more than one company (and its partners) can implement it freely. In theory, as affirmed by Ghosh, this openness should be able to support full competition.

To better define, avoid confusion on open standards and foster the true development of open standards, SUN created an open standards checklist. This list has a series of pro-competitive criteria to address technical, business and legal components of interoperability, with a focus on the final product. This check list is an interesting guidance in this quite complex world which is located in the border line between private and public interests. In fact, often private companies, probably with a shorter term vision, can not afford or do not perceive the real interest of open standards, therefore to set up clear boundaries can be very useful.

In the SUN check list, two kinds of issues are important at same level in determining a truly an open standard technical specification: "how it is created and managed" and "how it can be used". From SUN check list for open standards (SUN, 2007):

Creation and Management of an Open Standard

- Its development and management process must be collaborative and democratic:
    - Participation must be accessible to all those who wish to participate and can meet fair and reasonable criteria imposed by the organization under which it is developed and managed.
    - The processes must be documented and, through a known method, can be changed through input from all participants.
    - The process must be based on formal and binding commitments for the disclosure and licensing of intellectual property rights.
    - Development and management should strive for consensus, and an appeals process must be clearly outlined.
    - The standard specification must be open to extensive public review at least once in its life-cycle, with comments duly discussed and acted upon, if required.

Use and Licensing of an Open Standard

- The standard must describe an interface, not an implementation, and the industry must be capable of creating multiple, competing implementations to the interface described in the standard without undue or restrictive constraints. Interfaces include APIs, protocols, schemas, data formats and their encoding.
- The standard must not contain any proprietary "hooks" that create a technical or economic barriers
- Faithful implementations of the standard must interoperate. **Interoperability** means the ability of a computer program to communicate and exchange information with other computer programs and mutually to use the information which has been exchanged. This includes the ability to use, convert, or exchange file formats, protocols, schemas, interface information or conventions, so as to permit the computer program to work with other computer programs and users in all the ways in which they are intended to function.
- It must be permissible for anyone to copy, distribute and read the standard for a nominal fee, or even no fee. If there is a fee, it must be low enough to not preclude widespread use.
- It must be possible for anyone to obtain free (no royalties or fees; also known as "royalty free"), worldwide, non-exclusive and perpetual licenses to all essential patent claims to make, use and sell products based on the standard. The only exceptions are terminations per the reciprocity and defensive suspension terms outlined below. Essential patent claims include pending, unpublished patents, published patents, and patent applications. The license is only for the exact scope of the standard in question.
    - May be conditioned only on reciprocal licenses to any of licensees' patent claims essential to practice that standard (also known as a reciprocity clause)
    - May be terminated as to any licensee who sues the licensor or any other licensee for infringement of patent claims essential to practice that standard (also known as a "defensive suspension" clause
    - The same licensing terms are available to every potential licensor

- The licensing terms of an open standards must not preclude implementations of that standard under open source licensing terms or restricted licensing terms.

Schmidt & Porter (2001) argued that the Open Source can be successful within communities with unresolved software needs, being in this case the scientific community, as one of the most important examples. They contend that the market of the "scientific software" is not attractive enough, in terms of dimension and potentiality, for the software industry, which could be true in general. Besides considering the fact that the most part of "scientific software" is developed in academic projects, which means in general within organization supported by public funds, it is understandable that the "closed software" industry is rather careful in approaching that market.

The business model of software industry, essentially based on the "software as a product with a price", is not easily well-matching with the business model of open source software.

Therefore, with regards to requirements of open standards in the field of Integrated Water Basin Management, which is a niche market, probably no private company or other organization could provide all of the specialist knowledge required for the variety of software tools and technologies that are required for these software development projects.

Concerning the development of open standards in pure hydroinformatics application as in water quantity and water quality models, we are still at the beginning of the dedicated research but an increasing interest in the last ten years is recognisable.

Even if standard for hydroinformatics do not yet exist, most of the software components needed for the information system have already their open standard reference, as for instance the Open Geospatial Consortium for geospatial information.

This is why we consider fundamental for our integration exercise to choose and use the best existing open standards for the components to be integrated in the IS.

3.5.1 Open source development trends and the research community

Officially, the first free and open source project begun with the GNU project (www.gnu.org) in the early 80's. According to von Krogh and Spaetha (2007), from its beginning the open source developers' community has triggered a vast volume of research. They defines a list of characteristics that made the phenomenon successful and worth of examination. In particular, we are interested on what they argue about the open source developers' community and the innovation process and its similarity with scientific community. Concerning the first, they support that open source software developers frequently engage in a dialog on its functioning (it also has its own research community). Concerning the innovation process in open source software resembles knowledge production in science (in many instances, open source software is an output of research processes). Therefore there is a strongest connection in the *modus operandi* between teams of open source developers and scientific research communities. The software development virtual team around open source growths in a similar way and for similar reasons of the virtual scientific communities, where several researchers interested in a particular phenomenon resolve a question or problem sharing their experiences and their findings. Talking in particular about the specific subject of hydroinformatics, another similarity between the two development paths is that it takes a long time to filter a software development project into practice, if it ever breaks through at all (Harvey, 2002). This happens not only because the hydroinformatics software market is extremely limited. Two other reasons, we can try to address, are that software produced by research groups are in general closed source and standalone, and unable to be integrated with the systems of general use in the industry. Therefore, obstacles of entry in the software market are high, so that much of this software never succeeds the stage of being experimentation, even if binaries are distributed at no cost. This suggests that in hydroinformatics it remains difficult to utilise the results of other scientific research projects and often similar models are implemented in software programs many times with a great waste of resources.

3.5.2 Open systems and interoperability

The Carnegie Mellon Software Engineering Institute (http://www.sei.cmu.edu/) states that Open Systems promise the faster and more economical development of high-quality systems that are technologically up-to-date. An open systems approach is important to improve acquisition efficiency and system interoperability. In the rapidly changing world of software acquisition, open systems continue to grow in importance. There are many definitions of open systems. One of the most quoted is from the Open Systems Joint Task Force of the USA Department of Defence: "An open system is a system that implements sufficient open specifications for interfaces, services, and supporting formats to enable properly engineered components to be utilised across a wide range of systems with minimal changes, to interoperate with other components on local and remote systems, and to interact with users in a style that facilitates portability". From this definition it is possible to see that interoperability and open systems are strongly related.
This definition provides a good high-level vision of what open systems are all about. For a more operational definition, we can turn to this one.
An open system is a collection of interacting software, hardware, and human components

- designed to satisfy stated needs
- with interface specifications of its components that are
    - fully defined
    - available to the public
    - maintained according to group consensus
- in which the implementations of the components conform to the interface specifications

For our purpose it could be defined as an open source operating system, typically composed of coordinated modular components from a number of sources and not reliant upon any proprietary elements. Characteristics of open systems include the exposure of the source code, which is thus available for understanding and possible modification and improvement;

portability, which allows the system to be used in a variety of environments, and interoperability, which allows the system to function with other systems. The interoperability in this context could be referred as the ability of the components to work together in the system. For large systems, whose evolution is necessarily incremental, component interoperability is a key requirement. Openness appears to be a fundamental prerequisite for interoperability. Integration and interoperability, with the priorities defined in the IWRM policy (Dublin, 1992) and in the European Water Framework Directive (Wasson, 2003 and Uslander, 2005), can be included among the most important topics of research in hydroinformatics. This is demonstrated by the increasing number of collaborative projects being undertaken in many fields of hydroinformatics, one of the most important already mentioned is the HarmonIT project financed by the Framework Programme of the European Commission (Gijsbers, 2002) which has released its standard, the OpenMI (Gijsbers, 2004). This initiative is far for being considered concluded as still much work is needed to get to stable standards.

On the other hand, the growing formation of strategic links between otherwise competing institutions, as for instance the Open Modelling System (OMS) (Goede, 2005), mentioned in the literature review, is another confirmation. The freedom gained by utilising the open standard model, could accelerate the development, supporting a more rapid expansion in the market. It is probably complicated to foster this sort of cooperative approach in a competitive marketplace, but with the support of the policy makers funding, we reckon that this could be achieved.

In conclusion, according to the IWRM approach and principles, water models should work in a bigger framework dealing with other components of an information system. Hence an open standard for interoperability or integration of hydroinformatics should be dealing with existing and stable open standards. Not having still a stable hydroinformatics open standard, our focus will be oriented towards interoperability in hydroinformatics of existing open standards and related software components for IWBM.

CHAPTER IV

4 SYSTEM DESIGN

4.1 Summary of the chapter

Following the general analysis of the technologies relevant to this research carried out in the previous chapter, this chapter intends to illustrate in detail the choices made in designing the structure of the information system and in selecting its different components. After this introduction the chapter continues with the description of the information system three-tier architecture. Subsequently to the software architecture design, the chapter presents the analysis of the "glue" of the components of the structure: the core programming language. Finally the following paragraphs focus on the elements to be interconnected that are presented and analysed. Particular attention is given to the description of the language used for the data management layer and to the management layer itself, defined as the Geo-database.

4.2 Information system design

The development of information technology has a high impact on human activities and development. Poor or wrong information are followed by wrong results of the analysis tools, which could bring the policy makers to inappropriate policies or the decision makers and the managers to natural resources mismanagement. Therefore information management has also environmental, governmental and socio-economic aspects. Water resources management is a complex problem involving many variables and as a result combinations of many variables need to be monitored and interpreted with the right models and procedures. An information system has the task to collect, analyse and process existing information and present it to the decision-maker level. It is a dynamic set of tools dealing with information and information processes. The methodology and technology analysis is approached here considering that integrated water basin management asks for integrated analysis needing integrated information and modelling systems. The different tools integrated in the information system have to interoperate exchanging data through a multithreaded architecture where an instance of a linkable component handle data requests. If necessary, components can start their own computing process to produce the requested data. As presented in chapter 3, openness and open standard technologies have been the central thread to create the information system, the designing of the software architecture has been the structure that supported the development towards the objectives of this implementation.

An information system for IWBM results then in a data exchange and analysis environment, embedding software components storing, treating and presenting information. In order for such a complex collection of models, databases and processors to function both as individual components and as part of an integrated framework, an object-oriented architecture with simple and well-defined interfaces is required. The procedure followed to transform our general idea of integrated water basin management and the deriving requirements in an information system has been organised through the following stages:

1. Analysis and identification of current issue in hydroinformatics research for IWRM (previous chapters);
2. Analysis of past and ongoing related projects (previous chapters);
3. Analysis of the state of the art of relevant technologies (previous chapters);
4. Design of the software architecture of the information system;
5. Identification of the different categories of components or software tools making part of the entire system;
6. Selection of the existing open standard technologies
7. Identification of the software components to be integrated in the system
8. Set up interconnections between the different technologies to be implemented in the pilot information system.
9. Testing the pilot through a case study.

The sixth and the seventh points are strictly interdependent and they have been carried out together.

4.3 System Architecture

Software architecture forms the backbone for building successful software-intensive systems. Architecture is fundamental for system's quality attributes such as performance, extensibility or reliability. As defined by the Institute of Electrical and Electronics Engineers (IEEE) Recommended Practice for Architecture Description of Software-Intensive Systems (IEEE standard 1471-2000), architecture is "the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution". Concerning the process of developing an IS architecture for IWBM, even though the need for formalisation of modelling techniques is commonly accepted, not much literature is dedicated to the concrete development and standardisation of procedures implicated. Conventionally, information systems has been a fairly informal field of research approached from several other fields more than directly focused and many methods and techniques have been introduced without a formal foundation.

Hence our approach the development of an information system from the IWBM point of view consisted of analysing the available hydroinformatics literature designing our own development process based on the general requirements identified for our system. Usländer (2005) affirms that to meet the requirements of an Integrated Water Basin Management approach, considering a river basin as reporting unit, an IT Architecture must follow a trans-organisational approach. He shows that this approach should be undertaken from the organizational, process, data and functional point of view. He conceptualise an IS organised in two different levels, a thematic layer where information is treated and a co-operation layer where information is exchanged. In order to efficiently master the information flow between the thematic layer and the co-operation layer, he suggests that thematic modules, such as environmental models, should be extended and directly connected to a content manager component in the co-operation layer over an Intranet or the Internet.

Another example of integrated architecture for IWBM is the Danubia project (Barth et al., 2004) where thirteen models of meteorology, land surface, water research and social sciences are integrated. As the previous one the system is organised in different levels dealing with

thematic and communication layers. The implementation language is Java and the network communication is realised by Java's RMI technology. One of the most complete information systems for IWBM close to our requirements, except for the fact that is not open source based, is WaterWare (http://www.ess.co.at/WATERWARE/). WaterWare is an integrated, model-based information and decision support system for water resources management. WaterWare is implemented in an open source, object-oriented client-server architecture, fully web-enabled and Internet based, supporting the integration of databases, GIS, simulation and optimization models and analytical tools. WaterWare is designed as a modular system; it can integrate a range of information resources. Linked to this data layer are a set of models, which can perform scenario analysis, questions for various water quantity and quality issues, as well as related engineering, environmental, and economic aspects.
From the analysis of the above mentioned IS and from the other cited in the literature review, it is possible to affirm that the main components typically constituting the architecture of information systems for IWBM are:

- User interface
- Knowledge and data management
- Simulation engine
- Communication standards and process control

Several aspects of the software architecture, in particular the different software technologies chosen, have been analysed in the following paragraphs.

4.3.1 Object oriented architecture for models and components linking

In their early stage of development GIS in hydroinformatics were used to deal with modelling of surface water and groundwater (Abbot, 1991). Further extension took place by including chemical and bio-chemical modules to monitor and assess water quality. However, most of these integrated codes and algorithms were customized for specific simulation engines and required significant resource efforts and commercial justification to include new/alternative engines. Apart from it, they also required significant resources when upgrading model software to new versions, as many of the internal links and interfaces will need to be re-written.

The growing demand for more open systems towards modularity and allowing greater flexibility, has been driven partly by the increasing complexity (and cost) of integrated modelling systems, but also by increasing acceptance of integrated water management as a fundamental approach. In order for such a complex collection of models, databases and processors to function both as individual components and as part of an integrated framework, an object oriented architecture with simple and well-defined interfaces is required.

A generic object based modelling system described in Havno et al. (2001) is showed below.

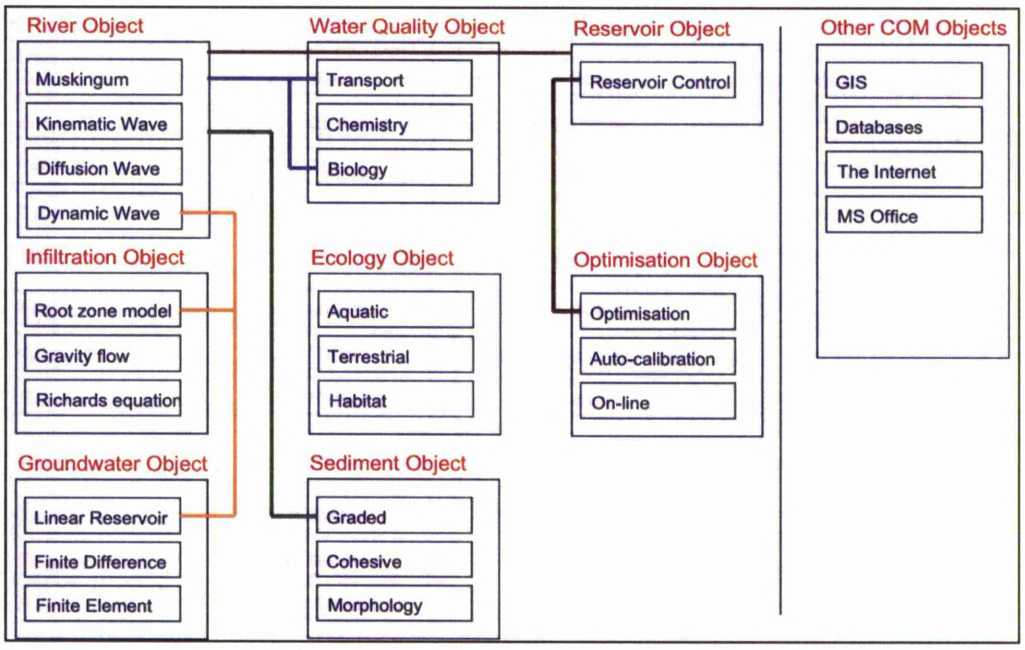


Figure 4.1: Structure of an Object Oriented Integrated Water Resources Management System (Havno et al. 2001).

Coping with the complexity of the system and depending on how closely the models are linked with their architecture, the work of developing such links requires several man/months. In addition to the limitation of such links, they can only be used for the specific models. In order to address problems associated to model coupling and to form the basis for new opportunities, a new architecture for water resource models has to be implemented.
It is obvious that the communication problem resides in the links between the different components. The critical point is to find out a way to set up a standard communication protocol. When speaking the object oriented language, these standard communication protocols can be identified as interfaces, being an interface defined as a set of methods and properties that can be accessed by other objects.
The advantage of using this approach is that once an object is implementing a standard interface, it will immediately be ready for coupling with any other objects that comply with this standard; the same applies at the level of a component. In our case, the communication protocol, other than just being a set of properties and methods (interface), will also implement a process control. Data interchange will follow a particular and defined path that will allow the knowledge base of the system to be accessible not only from the simulation models but for all other internal components.
The process control improves the knowledge management capacity of the entire system. The information always passes through the component of the system entitled to manage the data. This component, being particularly indicated and tailored for this task, will have the capabilities to structure the knowledge and to make it widely accessible from inside and outside the system (figure 4.2).

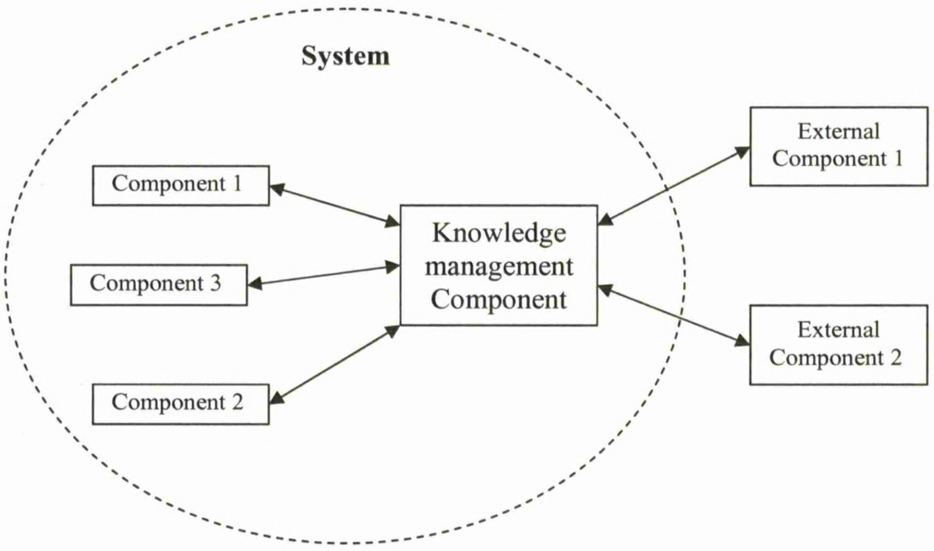


Figure 4.2: Internal and external data interchange paths.

The challenge is to define a standard interface and a communication protocol, which together can work as a bridge between the components and the knowledge core of the information system.

To improve inter-flexibility the linkage between components will be composed of two levels. The level of the interface closest to the component will be a sort of customized interpreter.

In figure 4.3 are shown the two levels of the linkage of the interface between the knowledge management component and the other components.

In this case the first level is a standard one, and it is responsible to interact with the Knowledge management Component and with the second level. The component's interpreter interacts with the first level and the other components.

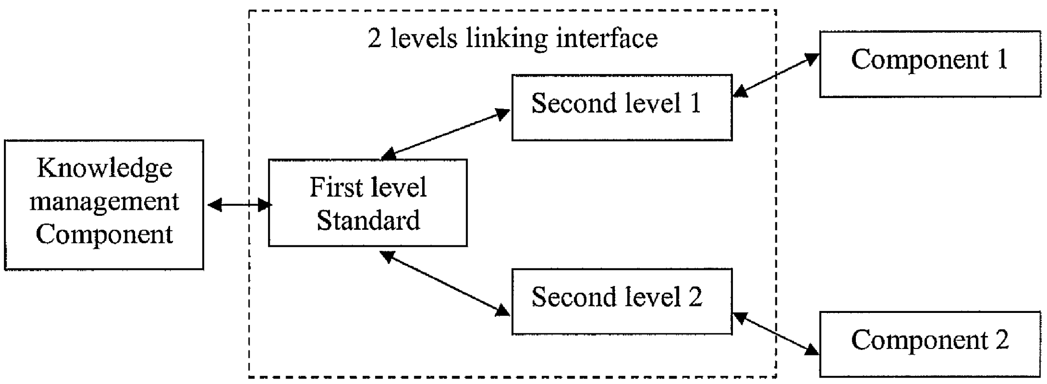


Figure 4.3: the two levels of the linking interface, the first one is standard, the second one customized on every component.

4.3.2 Client – server architecture

A client is defined as a requester of services while the server is known as the provider of services. A single machine can be both a client and a server depending on the software configuration. The actual client/server model started gaining acceptance in the late 1980s. The client/server software architecture is a versatile, message-based and modular infrastructure (Bass, 2003). The first kind of software systems developed were not client/server based, but rather based on mainframe architecture. With mainframe software architectures, knowledge and information are captured within the central host computer. Users interact with the host through a terminal, which captures keystrokes and in turn sends that information back to the host. Mainframe software architectures are not tied to a hardware platform. User interaction can be done using PCs and UNIX workstations. The limitation of mainframe software architectures is the lacking of graphical user interfaces support or access to multiple databases from geographically dispersed sites. In the last few years, mainframes have been widely used as servers in distributed client/server architectures (Edelstein, 94). The original PC networks were based on file sharing architectures, where the server downloads files from the shared location to the desktop environment. The requested user job is then run (including logic and data) in the desktop environment. File sharing architectures work if shared usage is low, update contention is low, and the volume of data to be transferred is low. As a result of the limitations of file sharing architectures, the client/server architecture emerged. This approach introduced a database server to replace the file server. Using a relational database management system (DBMS), user queries could be answered directly. The client/server architecture reduced network traffic by providing a query response rather than total file transfer. It improves multi-user updating through a GUI front end to a shared database. In client/server architectures, Remote Procedure Calls (RPCs) or standard query language (SQL) statements are typically used to communicate between the client and server (Edelstein, 94).

4.3.3 The N-Tier software architecture

Large monolithic applications are difficult to maintain and enhance, while application built on layer are more flexible, easier to assemble, maintain and extend. This is why, in order to meet general requirements, the architecture of the IS has to be designed as a multi-layered system, starting with a low-level back-bone (e.g. basic and general functionalities for code developers) upon which more customised functionality layers are added (following the user type hierarchy), until the requirements for a certain user type have been met. The multi-layered system technologies most suitable for this kind of information system are the N-Tier architecture models, N-Tier meaning "Any Number of Tiers" (layer and tier will be treated here as synonymous). The N-tier architecture is a kind of Client/server architecture. It provides a model for developers to create a flexible and reusable application by breaking up an application into tiers. Developers only have to modify or add a specific layer, rather than having to rewrite the entire application over again if they decide to change technologies, scale them up or re-customise a certain layer. The N-tier architecture is categorised according to the number of layers implemented.

With two tier client/server architectures, the user system interface (upper layer) is usually located in the final user's desktop environment and the database management services (base layer) are usually found in a more powerful machine server, which can service many clients. It is a split between the user system interface environment and the database management server environment. The database management server provides stored procedures, standard triggers and in general a set of common procedures in order to reduce client side workload. There are a number of software vendors that provide tools to simplify development of applications for the two-tier client/server architecture (Edelstein, 94).

The two-tier client/server architecture is a good solution for distributed computing when dealing with work groups comprised of a dozen to 100 people interacting on a LAN simultaneously. However, it has a number of limitations. When the number of users exceeds 100, performance begins to deteriorate. This limitation is a result of the server maintaining a connection via "keep-alive" messages with each client, even when no work is being done. A second limitation of the two-tier architecture is that implementation of processing management services using proprietary database procedures restricts flexibility and choice of DBMS for applications. Finally, current implementations of the two-tier architecture provide limited flexibility in moving (repartitioning) program functionality from one server to another without manually regenerating procedural code. (Edelstein, 94).

The three-tier architecture (also referred to as the multi-tier architecture) emerged to overcome the limitations of the two-tier architecture. In the three-tier architecture, a middle tier is added between the user system interface client environment and the database management server environment. There are a variety of ways of implementing this middle tier, such as transaction processing monitors, message servers, or application servers. The middle tier can perform queuing, application execution, and database staging. For example, if the middle tier provides queuing, the client can deliver its request to the middle layer and disengage because the middle tier will access the data and return the answer to the client. In addition, the middle layer adds scheduling and prioritization for work in progress. The three-tier client/server architecture has been shown to improve performance for groups with a large number of users (in the thousands) and improves flexibility when compared to the two tier approach. Flexibility in partitioning can be a simple as "dragging and dropping" application code modules onto different computers in some three-tier architectures. A limitation with three-tier architectures is that the development environment is reportedly more difficult to use than the visually-oriented development of two-tier applications (Edelstein, 94). Recently, mainframes have been used as servers in three-tier architectures (figure 4.4).

DATA TIER	Storage Query and storage optimization Performance (indexing, etc...)
DATA ACCESS TIER	Interface with the database Handles all data I/O Made to scale, usually stateless
BUSINESS TIER	Business objects and rules Data manipulation, analysis and processing Data and knowledge managemet
Presentation - GUI	End user system End user interface

Figure 4.4: A typical N-Tier model, Application Architecture. Robert Chartier. (http://www.15seconds.com/Issue/011023.htm)

In this example the Data tier is intended to deal with the storage and retrieval of information. The data access layer is the location where generic and standard methods interact with data. It can implement various methods, for instance, for creating and opening a connection object (internal) and for creating and using command object. It can also have some specific storing methods, so that it can persist to the Data Tier. This data layer is a reusable interface to interact with the database.

The business layer is basically where the brain of the application resides; it contains for instance, business rules, data manipulation, modelling tools, etc. This layer normally does not have any code to access the database or the like. These tasks are assigned to each corresponding layer above or below it.

The business layer provides an interface for the end-user into the framework's application. It works with the results/output of the business tier to handle transformation into something usable and readable for the end user. The main features of the N-Tier architecture are the following:

- Different layers carry out different tasks;
- The tiers may contain one or more components of the application

- The components in one tier can communicate only with the components in the tiers above and below
- Different layers or subsystem will be developed separately;
- Interface layers and communication protocols have a fundamental role in the framework development;
- Modularity, extensibility, reusability.

4.3.3.1 Client/server approach in IWBM remote modelling

Several approaches and technologies to implement interactive programming and remote modelling using Internet are reported in the literature (Linthicum 1996, Sutherland 1997). Current developments in Internet are typically focused on an object-oriented implementation (Solomatine, 1996) that improves maintenance, security and enables software reuse. Depending on where the computational process takes place, at the client or server side, there are three client/server architectures on which remote modelling using Internet can be implemented:

1. Client oriented approach, where the main computational process goes to client. This architecture can be used for distance learning and training purposes and for limited hydroinformatics models and modelling systems.
2. Server oriented approach, where hydroinformatics modelling system resides at the server side, and the client has controlled access to it (account and password). At the client side, usually there are user friendly interfaces for passing the input data and displaying the results in the browser.
3. Combined approach, where the main computational process resides at the server side, and pre-processing post-processing or some modules go to the client side.

4.4 Information system implementing Technologies

Reviewing the literature of frameworks and integrated modelling systems for IWBM, it has been possible to categorize the main software tools (components of the system) that the information system needs in: data-bases, the core of the system responsible for storing and managing the knowledge; Geographic Information Systems, the geospatial data visualization and geospatial data management tool; the User Interface tools responsible of the interaction between the end user and the system; simulation models and knowledge management tools responsible for data process. The last two kinds of components are conceptually different from the Hydroinformatics point of view but quite similar from the informatics point of view. In fact, even if knowledge management tools on the contrary of models do not imply simulation, the implementation phase is quite similar. Hence it is reasonable to treat them together.

To implement these components in the system we analysed several available software solutions. They have been examined on the basis of the technology analysis carried out in previous chapters and as implementations of these technologies. In the software solutions analysis, the choice to restrict the research only to open source software, more than just a general requirement of the IS, has been an inevitability. In fact, in order to integrate different software together, we had to customise their input/output interfaces or API, modifying and re-programming often several *classes* (in Java almost everything is a class) of the software. Therefore, the design of the information system, according to the general requirements, goes towards the development of an object-oriented hydro information system taking into consideration the fundamental open technologies and open standards.

Hence, being prohibitive in terms of time, resources and know-how and also out of the scope of this research, to develop our own GIS and Database, while referring to the general principles exposed above, we have used, adapted and integrated in the system several open source projects.

The fundamental technologies and open standards used in the research and implemented in the system include:

- Java as core programming language;
- Java technologies (JDBC, RMI, etc.) as backbone for the IS structure and architecture;
- The Open Geospatial Consortium specifications as standard technology for the GIS application;
- The standard Internet technologies (TCP/IP, HTTP, etc.) as communication support over the client/server architecture;
- SQL as information interchange and query language to the relational database;
- The relational logic for the database (Relational Database Management Systems);

While referring to the general principles exposed above, and as implementation of the above mentioned technologies, we have used, adapted and integrated in the system several open source projects and among them the most important:

- OpenMap as geographical information system (http://openmap.bbn.com);
- PostgreSQL + PostGIS (www.postgresql.org) as relational database with spatial extensions in the first implementation;
- JPOX (www.jpox.org) and DB4O (www.db4o.com) as improvement of the data layer, in the second implementation of the information system, exposed in the next chapter of the thesis.

The details of the open technologies and of the open source software chosen and above listed are exposed in the following paragraphs of the chapter.

4.4.1 Programming languages analysis

Concerning the use of fundamental characteristics of the core programming language in this research, we can assert that our information system intends to deal with incompatibility between hydroinformatics systems, tools and components in an internet based and client/server environment.

In the scientific and technical literature and in the Internet there is a huge amount of analysis and benchmarking dedicated to comparison among object oriented programming languages. Benchmarking different programming languages is a difficult exercise mostly because different languages have been designed for different purposes and in different periods of hardware development. This is probably why debates about different programming languages remain often inconclusive (Prechelt, 2000). Benchmarking test cannot be exhaustive and relies on a heavy amount of variable factors. Even in the framework of the same software project, for different kinds of functions or operations, different benchmarking test can be designed. Benchmarking tests are often based on quantity related parameters directly influencing performance. Following parameters are often considered during benchmarking:

- Level of Object-Orientation
- Static / Dynamic Typing
- Inheritance
- Garbage Collection
- Class Variables / Methods
- Reflection
- Access Control
- Multithreading
- Regular Expressions
- Built-In Security
- Portability

A concise explanation of the list of parameters is given below based on the book "Object-Oriented Software Construction" (Meyer, 1997). For an extensive analysis of programming languages and their impact on hydroinformatics a thorough research would be needed. The languages cited are the ones that are currently used in the market and could be easily adapted to development of this research. These are: Ruby, Eiffel, C, C++, Python, Visual Basic, Perl and Java.

Level of Object-Orientation

Object orientation has been widely discussed in paragraph 3.3.4 where basic definitions of object oriented languages, logic and technology have been given. Many languages claim to be Object-Oriented, while the exact definition of the term is highly variable.

As for instance among the most well known and diffused programming languages Eiffel, Smalltalk, and Ruby are all pure Object-Oriented languages, supporting all qualities listed in paragraph 3.3.4. Java claims to be a pure Object-Oriented language (SUN, 2010), but by its inclusion of "basic" data types that are not objects, fails to meet one of the parameters according to many IT experts. Since version 5.0, however, the "autoboxing" function enables programmers to proceed as if primitive types were instances of their wrapper class (www.java.sun.com).

C++ is considered to be a multi-paradigm language, of which one paradigm it supports is Object-Orientation. Thus, C++ is not (nor does it contend to be) a pure Object-Oriented language.

Python is often proclaimed as one of the purest Object Oriented language. Therefore some operations are implemented as methods, while others are implemented as global functions. Some analysts complain about Python's lack of "private" or "hidden" attributes, which goes against the Encapsulation/Information Hiding principle. The Ruby language, on the other hand, was created in part as a reaction to Python. The designer of Ruby decided that he wanted something "more powerful than Perl, and more Object-Oriented than Python." Visual Basic and Perl are both procedural languages having Object-Oriented support added on as the languages have matured.

Static / Dynamic Typing

There has always been a strong debate in informatics between the supporters of static and the support of dynamic typing in the environment of Object-Oriented specialists. The title of a

well known paper by Erik Meijer and Peter Drayton (Meijer, 2004), "Static Typing Where Possible, Dynamic Typing When Needed: The End of the Cold War between Programming Languages", gives the idea of the complexity of the topic. Supporters of dynamic typing argue that it is more flexible and allows for increased productivity. Static typing supporters claim that it improves safety, reliability and efficiency of code. It goes beyond the scope of this research to analyse in deep performances of the two approaches but it is relevant to the choice of the programming language to say that that a statically-typed language requires a very well-defined type system in order to be as flexible as the dynamically-typed.
Smalltalk and Ruby are two pure Object-Oriented languages that use dynamic typing. Eiffel is a statically-typed language but manages to remain nearly as flexible as its dynamic counterparts. C++ also offers generic classes (known as "templates" in the C++ parlance), as well as multiple inheritance. Java was seriously hindered by a lack of generic classes which have been added in 2004 as part of J2SE 5.0. They allow "a type or method to operate on objects of various types while providing compile-time type safety" (http://java.sun.com/). Java also allows type casting, but some rudimentary type checks can be made by the compiler, making casts in Java somewhat safer than in C++ and other languages.

Inheritance

Inheritance, as defined in 3.3.4, is a cornerstone for object oriented programming. It is the ability for a class or object to be defined as an extension or specialization of another class or object. Characteristics in object-oriented programming terms are attributes and behaviours of a class— that is, the data and methods of a class.
Object-oriented languages in general support class-based inheritance, for some programming languages such as SELF and JavaScript the support is oriented to object-based inheritance. Just few languages, as for instance Python and Ruby, support both class- and object-based inheritance, in which a class can inherit from another class and objects can be modified at run time with the features of other objects.
There are three ways to implement inheritance: simple inheritance, multiple inheritance, and level inheritance. Each different option enables a class to access attributes and behaviours of another class using slightly different techniques. Multiple inheritance for instance is the ability

for a class to inherit from more than one super (or base) class. Java does not support multiple inheritance, instead it supports interfaces, which somehow have to operate at the same level of multiple inheritance. However, in Java, interfaces are probably conceptually closer to the definition of abstract classes. Concerning the different possible uses of inheritance, Bertrand Meyer (1997) identified and classified 17 different kinds of inheritance. Despite this richness of opportunities, most programming languages allow only some syntactic constructs for inheritance. These anyways, are generally enough to take advantage of the usefulness of the inheritance. As example of the usefulness of multiple inheritance performances, an application object called PersistentGraphics could inherit from two different classes called GraphicalObject and PersistentObject, this in order to be used both as graphical object that can be showed on the screen in addition to persistence capability that allows the object to be stored in a database. In this example, multiple inheritance is described as an essential feature of every programming language, because when two or more distinct hierarchies have to be merged into one application object the only direct solution would be the multiple inheritance itself. But if multiple inheritance is not implemented by quite few languages, as Java, there are serious reasons. The most important reason is that multiple inheritance leads to additional complexity and complications into the programming language implementing it, such as name clashes and ambiguities in the object model.

For instance, the use of multiple inheritance in C++ is not as flexible and performing as one would expect from probably one of the most used programming language in the world. On the other hand, Eiffel is known for its carefully and thoroughly well-designed support for multiple inheritance, as stated in its web site it implements "robust multiple inheritance facilities".

Multiple inheritance is a conceptual feature of object oriented languages which can be implemented in one or more different forms as OO languages can support different forms of single inheritance (e.g. implementation and subtype inheritance). C++ and Eiffel support in the same way pure implementation inheritance as well as subtype inheritance. These two languages being probably more developed as far as inheritance is concerned also implement multiple inheritances in both forms.

Java, even though does not implement a pure support for inheritance, supports two different inheritance mechanisms. The implements-tool implements a pure subtype (interface)

inheritance, the extends-tool provides a combination of implementation and subtype inheritance.
Smalltalk implements just a single feature of inheritance: single inheritance of interface and single inheritance of implementation. In this case, a class can just only inherit from another class inheriting at the same time implementation and interface. Python supports one form of inheritance (implementation and subtype) similar to Smalltalk but it can perform multiple inheritance increasing its flexibility.
Ruby performs in between the two last languages described. Ruby implements single inheritance with an additional feature, it allows classes to implement a chosen number of modules. This implementation of inheritance similar but to some extent more limited compared to Python.

Garbage collection

Another important feature for programming languages is garbage collection. Garbage collection is a technology that allows a language implementation to free memory from unused objects freeing programmers from explicitly managing memory allocation for every object created. The main issue with garbage collection has always been that the automation needed heavy computational resources. Nowadays with increased hardware computational power and increased software efficiency, meaning better algorithms and techniques, have decreased computational burden increasing garbage collection impact on overall performances.
There are different kinds of possible implementation for garbage collection which can be adopted in language implementations. As for instance, reference counting is the simplest scheme which can be defined as a form of automatic memory management where each object has a pointer to the number of a reference. The reference number to an object increases when a reference to it is created, and decreases when a reference is cancelled. When the reference count becomes zero the object is automatically deleted. This way the memory where the object was stored is freed.
This kind of garbage collection technology cannot handle cycles. As for instance when two or more objects refer to each other, they can create a cycle. This is the scheme used by Visual Basic and Python, even if in the case of Python a solution for cycles handling is implemented.

The most common form of garbage collection supported by most language implementations, as for instance by Eiffel, Smalltalk, Ruby, and Java, is the "mark and sweep" technology. This garbage collection scheme overcomes the limitation of the reference counting scheme, the cycle handling. On the other hand the main disadvantage of mark and sweep garbage collection is that it is non-deterministic, this means that the sweep phase, the deletion, is not related to timing during the execution of the program.
Another process for garbage collection, known as "mark and sweep" but not as common as the other forms that can be found in some implementations of Eiffel, Smalltalk, Ruby, and Java, is the generational garbage collection. The generational garbage collector organises objects into "generations" based on their existence time. This reduces the time spent in the mark and sweep phases.
Concerning garbage collection a particular case is the one C++ which does not implement any sort of garbage collection. This is because for reasons of convenience and performance related to its development.

Class Variables / Methods

An important feature of object oriented languages is that class variables and methods pertain to a class and even if their direct existence is related to the instance of a class, they refer exclusively to the class itself. This brings to the fact that at a given point in time only one copy of each class variable/method exists even if together with many instances of the same class. The same class variable/method is shared by all the instance of that class.
There are different levels of implementation of this concept, as for instance the most advanced class variables and methods handling features are perhaps in Smalltalk and Ruby. In these languages classes are also objects and moreover they support meta-classes. Less advanced is the implementation in Java and C++. They provide "static members" with static variables and static methods, which compare class variables and methods but are more limited since they do not support inheritance. We are in even less advanced case with Eiffel, which does not provide direct support to class variables or methods. The solution comes with the function called "once" similar but more limited than in the previous cases. The least developed

example is the one of Python where class methods or variables are not at all supported. In this case the feature called "module" is an alternative solution.

Reflection

Computational reflection (Ferber, 1989) is the ability to monitor and intervene on the computational behaviour of a system through a process called reification. This idea, originated in the field of logic, more recently has been recognised as one of the key features in the construction of object-oriented systems. Through this technology, systems can keep information about itself (meta-information) and reuse this information to influence its future behaviour. Operations of the object model can be captured and modified using reflection.
Object-oriented programming languages in general support reflection even if in quite dissimilar ways. Ruby, Smalltalk, and Python support reflection mechanisms with very powerful and flexible features. Java's support to reflection is less flexible and dynamic than the others, even though it has improved in last versions. C++ does not give full support to reflection but it implements a run-time information tracking technique allowing programs to determine the type of objects. Eiffel improved in the most recent versions its limited support to reflection, including the information tracking for features contained in objects.

Access Control

The access control feature is related to the encapsulation hiding principle of object oriented languages. It allows modules implementation to keep hidden behind its public interface. In the particular case of hydroinformatics with time series and several kinds of environmental data to manage, access control is a very important feature. These are the methods that, for instance, could be in charge of connecting to databases for storing, accessing and retrieving information. Access control allows the application program interface (API) to connect to the database without influencing the client code and without being exposed. API hide database related code, such us SQL (not object oriented code), without impact on the software architecture.
As for the other features, there are different interpretation and implementation of the access control programming capability. Starting from the less developed approach, probably Python

which does not provide any level of access control but instead provides a sort of labelling. This means that with that label clients code can anyways use the labelled feature with an indicator that is intended to show that the feature should not be used for that since it is not meant for this.

Object oriented languages provide usually two levels of access control defined as public and protected. Protected features, like variables, are not available outside of the class in which they are contained, except in the case of inherited properties like for subclasses. In Smalltalk for instance all methods are public and all attributes are protected. Methods are not protected in Smalltalk, so Smalltalk programmers use the solution of "private protocol" in the class. Visual Basic supports the same two levels of access control with the difference that without inheritance in Visual Basic, protected features are completely private.

More developed is the approach of some other languages like Java and C++. They implement also a third and even more hidden level of access control called "private". Private features are just available in the class in which they are declared, inheritance is not applied and even for subclasses features are not available. Ruby also provides these three levels of access control, but they work slightly differently. Java provides a very useful feature with a fourth level of access control called "package private", making public features just inside a package of classes. The most powerful and flexible access control scheme of all analysed languages is provided by Eiffel and it is called the selective export. Features are by default public and for any of this feature of a certain class it is possible to specify the export clause which lists explicitly what other classes may access that feature. The only limitation Eiffel encounters is that the definition of "private" as there is in Java and C++ it is not present.

Multithreading

The word multithreading can be translated as many threads of control. Multithreading allows single process to access to two or more threads concurrently. Since operating systems support for threads has became widespread, the use of multithreading became a common feature. Because each thread runs independently, multithreading allows to (SunSoft, 1994):

- improve application responsiveness,

- use multiprocessors more efficiently,
- improve program structure,
- use fewer system resources,
- in general improve performance.

Regular Expressions

Programmers can save a lot of coding time by using regular expressions. Regular expressions (regex) allow powerful string parsing in much shorter lines of code. Considering the same instruction as example, regex are faster to write, easier to debug and maintain than the correspondent lines of normal code.

Almost every language has a support for libraries of regular expression. Native support to regular expressions has become increasingly important with the development of Perl. This integration improves performances and Perl was the initiator of this kind of model. Also Python and Java (since Java 4), have included regular expression libraries as part of the standard base library distribution.

Built-In Security

Built-in security allows languages to improve their stability. Pieces of code can come from different sources and in order for a language to be able to check if the piece of code is trusted (such as the hard disk) or not, it has to implement some kind of security capability. For the particular case of Java, applets are considered not trusted pieces of code. This is why they have limited scope when coming from internet browsers compared to piece of code coming from local machines. Almost all the languages under analysis, including Java, Ruby, and Perl have by default the built-in security. Other languages delegate this protection to the machine operating system.

Portability

Portability has already been described from different angles in this document. Concerning this research, Portability, more than just one of the key concepts of high-level programming, is a conceptual requirement of hydroinformatics related to integrated water resources management. This is why it needs a deeper analysis. Below of the programming language described, the most relevant to portability are listed from Raymond (2003).

Perl Portability

According to Raymond (2003) Perl's performance concerning portability are good. Certain version of Perl implementations offers direct support to portable GUIs across Unix, MacOS and Windows. The most important limitation is that Perl scripts works with external libraries from CPAN (the Comprehensive Perl Archive Network) which are not available in every Perl implementation.

Python Portability

Better than Perl, Python has excellent portability skills. Also several implementations of Python come with a toolkit supporting portable GUIs across Unix, MacOS, and Windows. In general, being Python an object oriented born language, it has a much richer standard library than does Perl and overcomes its main limitation.

C Portability

The core C language is extremely portable. The standard open source compiler is the GNU C compiler, which has been ported to UNIX, Windows and classic MacOS, with the limitation that the native GUI is not as portable as the core.
Portability of C codes depends strongly on programmer skills and ability. Several books have been written on the subject of portable C code, as for instance "The Practice of Programming" (Kernighan and Pike, 1999).

C++ Portability

C++ has the same portability issues as C plus, an additional feature related to their open source compilers developments which do not get to reach the level of its commercial implementations. Besides many open source compilers exist and thus it is difficult to define a de-facto standard.

Java Portability

Despite the fact that one of the main goals of Java was portability, the motto was "write once, run everywhere", in its early years of development Java portability encountered many problems. The difficulties were mostly related to the Abstract Window Toolkit (AWT), Java's original platform-independent windowing, graphics, and user-interface widget toolkit, than replace with the Swing, widget toolkit developed to provide a more sophisticated set of GUI components than the AWT. In general we can affirm that portability fails to be perfect but it is still excellent.

Ruby Portability

Ruby is highly portable with its to two main implementations: the de facto standard Ruby interpreter, the Matz's Ruby Interpreter or MRI, which is the most widely used, and JRuby, a Java-based implementation that runs on the Java Virtual Machine. Both of them have level of portability similar to Java itself.

4.4.2 The recommended core programming language: JAVA

In this paragraph after a brief description of main Java features, the reasons for choosing Java have been listed and justified.

Java was built by Sun's programmers in 1995, in reply to the incompatibility between various systems, hardware and software, and libraries they have to deal with (Sikora, 2003).

There were four primary goals in the creation of the Java language:

- It should be object-oriented.
- It should be independent of the host platform (more or less).
- It should contain language facilities and libraries for networking.
- It should be designed to execute code from remote sources securely.

The objective of Java was portability and after the first phase of development in the early 90s with the emergence of the Web, Java was propelled to the forefront of computer language design to meet the requirements for web portability. While, it was the desire for an architecture-neutral programming language that provided the initial spark, it was the Internet that ultimately led to Java's large-scale success.
The key considerations were summed up by the Java design team in the following list of buzzwords (www.java.sun.com). These represent the usual benefits of programming with the Java programming language using its APIs:

- **Simple**

Java has a concise, cohesive set of features that makes it easy to learn and use.

- **Secure**

Java provides a secure means of creating Internet applications.

- **Portable**

Java programs can execute in any environment for which there is a Java run-time system.

- **Object-oriented**

Java embodies the modern, object-oriented programming philosophy.

- **Robust**

Java encourages error-free programming by being strictly typed and performing run-time checks.

- **Robust security architecture**

It is possible to deploy fine-grained security control if needed.

- **Multithreaded**

Java provides integrated support for multithreaded programming. Threads are native in the Java platform. Therefore multi-threaded solutions can be implemented with ease.

- **Architecture-neutral**

Java is not tied to a specific machine or operating system architecture.

- **Interpreted**

Java supports cross-platform code through the use of Java byte code.

- **High performance**

The Java byte code is highly optimized for speed of execution.

- **Distributed**

Java was designed with the distributed environment of the Internet in mind.

- **Dynamic**

Java programs carry with them substantial amounts of run-time type information that is used to verify and resolve accesses to objects at run time,

- **Garbage collection**

No stray pointer problems,

- **Classloading**

It is possible to dynamically deploy more code after the device has been deployed,

- **Networking**

Standard APIs allow easily networking.

A particular mention is needed here in order to differentiate between compiled and interpreted languages. In the early stages of software technology development there was an important distinction between "real" compiled programming languages such as C, and simpler, slower languages called "script languages" such as Bourne Shell (the default Unix shell of Unix Version 7) or Awk (a script language designed for processing text-based data). With technology improvement in the '80s and above all in the '90s, performances differences decreased and interpreted languages like Java, Lisp, Perl and Python for different reasons became important players in the programming language world and started being considered general-purpose programming languages and not just useful for a limited range of applications. Script languages are often, but not always, interpreted from the source code or semi-compiled to bytecode (like Java) at run time. On the other hand compiled languages are implemented typically through compilers that generate machine code from source code), and not interpreters. Benefits of using scripting languages are the following:

Isolation

Scripting languages only works with the Application Program Interfaces needed to actually run requested actions. Being at a higher level they do not intervene directly in the device. This brings to an improved stability of the critical code which cannot be manipulated by the developer. Isolation has a direct impact on other issues related to programming performances explained below.

Human Resource Options

An important issue related to the previous one and that is really important when dealing with open source software and integrated water resources management concerns human resources. IWRM related to water basin management systems with the many different possible uses of water holistically managed could foster the intervention and the interaction in the development of several possible organisations. Each one of these organisations does not need to have developers expert in the critical code but they can focus on content. This brings two advantages, more stability for a system subjected to many inputs and interactions and more effectiveness for developers specialised on the content. This allows for instance to manage development trough content or software development kit (SDK) publicly available as it should happen in an open source community. The system manager can let the other responsible organisations (i.e. for environmental issue, for energy management and/or other utilities, etc.) develop the content for the device they need and work for without having to worry about systems stability and security. Developers of the system need just the APIs and the scripting language.

Upgradeability

Since scripting languages are interpreted there is a list of advantages linked to an easier code management process and networking. Operation like bug fixing, code maintenance are easier because could only interest certain modules that can be changed without going through critical code. Since, integrated water resources environmental management systems are flexible and adapting to continuous changes, additional content needs to be upgraded all the time by different actors. The alternative solution to this approach in hydroinformatics could be to develop a general dedicated scripting language, or even a particular scripting language per application. On the other hand it is also true that such a language is requested to run a complex system having all the features and functionality like that of a full programming language.

Java technology as scripting engine

A mature technology like Java has an incontestable series of advantages against customised and dedicated scripting languages. Specifications are publicly available from various sources, even for not specialised developers it is possible to quickly acquire needed skills since available documentation is extensive (books, online resources, tutorials). Therefore developers specialised in other fields and even new developers can become productive. All the organisations that are going to contribute to the development of the systems or even end user developers can have a more efficient economic approach to skills development. Even though it is a complex technology and in continuous process for development and improvement, Java technology is probably mature enough, tested and deployed toward many kind of purposes and industries. It is mature both in terms of implementation as well as APIs. Given the amount of developers and the openness of the technology approach there is an enormous quantity of code already available and reusable. In fact there is a large community of companies and developers specialised in working on the Java platform developing software and information systems.

To identify the popularity of programming languages one of the most used references is the TIOBE index (www.tiobe.com/tiobe_index/index.htm). The ratings are based on the number of skilled engineers world-wide, courses and third party vendors. The popular search engines Google, MSN, Yahoo!, and YouTube are used to calculate the ratings. According to TIOBE Programming Community Index for March 2009, the most used language by developers is JAVA, ahead of C and C++.

This gives an important insight also from the commercial point of view, since the high dissemination of the technology, multiple vendors and not just SUN provides compliant implementations of the Java platform for different purposes and resources. The availability of development tools, open source and commercial, is important. For instance, for the debugging there is a large availability of tools with different features and very small need for producing *ad hoc* tools. This allows firms focusing human and financial resources more on the content of the development than on the process with important gains in productivity and efficiency.

It is noteworthy to mention that there is a particular distinction between traditional scripting and Java. Traditional interpreters do a simple translation from source code into faster code that works but still close to the original. Java byte code is someway closer to a real machine language because for each line of source code there are several byte code instructions. This is related to the concept of just-in-time (JIT) compiler. In the Java programming language and environment, a just-in-time (JIT) compiler is a program that turns Java bytecode into instructions that can be sent directly to the processor. The bytecode is platform-independent code that can be sent to any platform and run on that platform. JIT technology that makes interpreted code run at compiled code speed, reduce the main problem of scripting language against compiled languages.

Looking also at the other side of the coin related to Java portability, one of the main critics that can be moved to Java is that, the main objective, the vision, of the Java platform's, cited already several times, Write-Once-Run-Anywhere (WORA), in practice, it is not completely acquired (and maybe it cannot be). Several technical and market related reasons causes these limitations. This vision remains anyways the strongest features and innovative added value of Java.

Besides, Java has a series of characteristics over other scripting languages. Since, Java took much of its inspiration from the design of C++ in terms of logic and semantics, Java is a complete programming language with well-defined syntax and grammar. Developers can work complex logic as expected from any programming language as it is from C++ itself. The JAVA platform also comes with a rich set of APIs. The fact that this is one of the biggest community of developers and users means that there is a huge availability of support libraries, leaving again more time to developers to focus on the content having as baseline a solid technology.

After highlighting all Java features, and in the previous paragraph comparing it to other programming languages, the question that can be raised in order to justify the choice are probably: why the Java platform is relevant to this research and to hydroinformatics in general and why Java is probably better suited than other available alternatives?
At the level of development of Java, considering interpreted programming language platforms with features similar to what above exposed, there is the .NET (to be pronounced as "dot net") Microsoft platform (http://www.microsoft.com/NET/).
The development of the .NET platform started considerably after (considering informatics timing) the development of Java platform. This can be considered in itself already a strong sign of the impact that the Java platform had on the market: the world market leader software company, Microsoft, decided to go for a similar solution as Java, meaning to develop a runtime interpreted platform. The main issue related to .NET, as for all other Microsoft technologies, is that there is, and probably there will always be, only one vendor for it i.e. Microsoft. On the open source side of .NET now there are other actors such as Novell (www.novell.com) with the MONO project (www.mono-project.com). Novell, continuing its aggressive approach to the open source market, after entering in the Linux world with SUSE Linux (www.novell.com/linux/), started this ambitious open source project for .NET. The Mono Project is an open development initiative to develop an open source UNIX version of the Microsoft .NET platform. Its objective is to enable UNIX developers to build and deploy cross-platform .NET applications. In any case, even considering MONO, the choice for the .NET related projects is far much limited that the JAVA based one.
In view of the analysis described in previous paragraphs, there is also a series of other alternatives for scripting languages like Python, Ruby, etc. These are interesting technologies but they have not been chosen because of a series of technological and market related concerns (not every concern apply to all of the previous mentioned technologies):

- They are not as mature as the Java platform.
- They lack of a strong community of developers

- Some features (e.g. multi-threading, security) may not be as developed as in Java or even not implemented.
- They lack of APIs. Standard APIs can be limited or extremely limited, above all if compared to the level of development of the Java platform.
- As mentioned Java has the singularity of being interpreted through its bytecode which through the Java JIT technology brings to very good performances compared to text-based interpretation usually much slower.
- Related to the fact that they can have limited developers' community, development tools (e.g. debuggers) can be available only if developed on purpose, increasing consistently organisational resources required.
- As showed with the TIOBE index (previously mentioned), Java is the most diffused languages today, meaning with the most number of available developers. For the law of supply and demand, to have developers in other fields of less diffused language it would probably require more resources.

Despite the fact that only some of the competing technologies have all the problems above mentioned, it is certain that none of them can have all the advantages of the Java platform all together. Again, this means that for an organisation choosing another technology, the work to fill the gap with JAVA performances would require important investments.
To be honest, it is possible that for very particular applications, JAVA customisation could be as important as for another technology and choice would require a deeper analysis and probably also important testing. At the end of the day JAVA represents a mature technology which has in one package all important features to develop IS for water resources management. An investment in a technology has also to be sustainable in the long term. A good investment has to be rentable not loosing value too quickly. This means in the case of developing information systems for IWRM that the organisation should reuse the majority of the existing software components to minimise further developments in case of requirements for additional functionalities or in case of changes or upgrading of hardware. In technical

terms, this implies to manage cost of components, different hardware scenarios and hardware obsolescence.

Therefore in order to minimise costs and improve investment profitability, once again the Java platform's WORA (Write Once, Run Anywhere. the JAVA motto) characteristic can be the answer. The majority of the code developed, also in terms of modules will be reusable with new hardware. Besides, openness of the system will allow an easier development and integration of new or updated modules.

So far, Java has been compared to similar interpreted technologies but all these consideration on different scripting and interpreted languages are not true when speaking about compiled languages, native solution, like C or C++. They can ensure same features as the Java platform and there are a lot of developers prepared in C/C++ and Linux (they are second and third in the TIOBE index). This implies that there are a lot of development tools available. One of the main limitations compared to Java is the fact that they do not implement isolation (as previously described), with less security in flexible development. They have less flexibility also for upgrading and extensions, as for instance every time the code needs to be upgraded it needs also to be recompiled, the same if new hardware is installed then testing procedures are to be carried out. Similar issues arise when dealing with maintenance.

In practise, also the Java code has to be ported and retested. The difference is that porting cost is reduced because it is limited to the Java platform instead of the entire code base. Besides, thanks to the open source community, it is highly probable, if using standard hardware, that there can be a free open source solution already available.

Regarding testing, fortunately with Java there are standard suites that check for a series of tests (API conformance, virtual machine behaviour, etc.). These tests are known as the Technology Compliance Kits (TCKs). TCK are suite of tests, tools, and documentation that provide a standard way of testing an implementation for compliance with a Java technology specification. In 2008, Sun made the OpenJDK Community TCK license available, therefore the community has the means to certify compatibility. This opens the door to free, compatible Java platform (standard edition) implementations also for GNU/Linux distributions. Test tools are however somewhat limited in terms of coverage. Sun (www.java.sun.com) and other big

Java developers provides also more complete solutions that are not free but covering a large range of testing options and requirements.

Java and the internet

Java and the Internet saw the light together at the beginning of the '90s. After its creation, the relationship between Java and Internet, oriented the evolution of Java itself. Java can be defined as a network programming language (Jonoski, 2002). Since the beginning, Java's success is brought by its own particular programming features related to networking, the applets. They can be embedded in Web Pages and run in all kind of platform. In fact given the heterogeneous spectrum of platform and OS that are used to access the Internet, many Web developers began developing Java applets to perform complex Web page graphics operations that were previously not possible to be Implemented in a portable way. These programmers encountered many early Java problems transporting their applets across Web browsers and across OS graphical user interfaces. While the Java Community Process (JCP) considerably improved the GUI capabilities of the language, since the early days of Java's release many public and private organizations have softly adopted Java to develop platform independent server functionalities which are not dependent on the GUI.

Although the applets are still very important features, nowadays the real power of Java is in the general programming capabilities related to networking represented in its built-in classes, such as, socket connections, connections with common getaway interfaces (CCGI), servlets, the remote method invocation (RMI), etc. (Jonoski, 2002). Therefore Java is now one of the most widely used programming languages for technology integration. This is another reason for us to choose Java in order to address interoperability and integration in hydroinformatics.

Nowadays Java-capable servers dominate the Internet as an established standard for server side Internet development. According to the Netcraft, one of the most important Internet services analyst consultant, (http://news.netcraft.com/) in January 2009 close to 60% of Internet servers used Java-compatible web server software, including Apache (www.apache.org), Sun One (www.sun.com), Zeus (www.zeus.com) and Google Web Server

GWS, a customised version of Apache by Google. This percentage attained approximately 90% when considering that Microsoft's IIS can also be configured to run Java server software with third party software. In the remaining 10% there are also many Java-compatible web servers, which are not specified by Netcraft.

Java and open source

It is natural that the open source community adopted a community maintained programming languages as a suitable implementation tools for open projects. Sun developed the Java Community Process (http://jcp.org), as a forum for changing the Java language to meet the evolving requirements of the information technology industry. The Java Community Process (JCP) includes companies, individuals, and agencies that desire to have input on the growth and change of the definition of Java. According to the JCP forum, since its introduction in 1998 as the open, participative process to develop and revise the Java technology specifications, reference implementations, and test suites, JCP program has fostered the evolution of the Java platform in cooperation with the international Java developer community.

In the last years there has been a big boost for open sourcing the Java platform. In 2006 SUN started releasing parts of the JAVA SE platform under the GPLv2 licence (The GNU General Public License is a free, "copyleft" license for software and other kinds of works) now the GPLv3. In 2008 OpenJDK-based implementations are appearing in Free software repositories of major GNU/Linux distributions. There has been a big cultural change on development strategies, SUN developers are encouraged to work with free Java developers on a broad range of packaging, porting, language and implementation projects. SUN is part of the free and open-source Java community now. This is in line with the work of an open scientific community that is needed in such complex fields such as integrated water resources management. Only through openness, projects can tend to a stronger integration.

Conclusions

Although the fundamental reason that necessitated the development of Java is portability, other factors played an important role in shaping the final features of the programming

language. Many of these factors supported the choice of Java as core language to develop an IWRM information system, despite the effort required to apply, convert and structure the hydraulic modelling knowledge in Java/object oriented language.
However, all the components of our information system are Java based except the data base management layer.

4.4.3 Java Developing Environment: NETBEANS

In order to facilitate programming the needed code for the information system was developed using an integrated development environment (IDE) also known as integrated design environment. This is a fundamental productivity tool, above all in the case of our information system, developed by hydroinformatics researchers and not by Java expert developers. An IDE Normally consists of an editor of source code, a compiler and/or an interpreter, an automatic builder tool, and possibly a debugger. Sometimes it is integrated with a system version control and with one or more tools simplifying the development of the GUI. IDE for object oriented programming includes also navigators of classes, analysers of objects and other programmers' supporting and exploring tools such as diagrams representing classes' hierarchy, etc.
Although there are also multi-language IDE like Eclipse, NetBeans and Visual Studio, generally IDE are oriented to a specific programming language. The most important open source JAVA IDE projects are NetBeans (www.netbeans.org) by SUN and Eclipse (www.eclipse.org) by IBM.
Eclipse began as an IBM Canada project and it became in 2001 an open source project with an open source foundation. Eclipse is a multi-language software development platform comprising an IDE and a plug-in system to extend it. It is written primarily in Java and it is used to develop applications in this language and, by means of the various plug-ins, in other languages such as C/C++, Cobol, Python, Perl, PHP and others.
NetBeans is the most important competitor of Eclipse and it is a SUN based open source project. The NetBeans IDE is written entirely in Java using the NetBeans Platform. NetBeans

IDE supports development of all Java application types (Java SE, web, EJB and mobile applications) out of the box. Among other features are an Ant-based project system, version control and refactoring (www.netbeans.org).

Most important feature of NetBeans is Modularity. All the functions of NetBeans are provided by modules. Each module provides a well defined function, such as support for the Java language, editing, or support for the CVS versioning system, and SVN. NetBeans contains all the modules needed for Java development in a single package. Modules also allow NetBeans to be extended. New features, such as support for other programming languages, can be added by installing additional modules. For instance, Sun Studio, Sun Java Studio Enterprise, and Sun Java Studio Creator from Sun Microsystems are all based on the NetBeans IDE.

The IDE chosen for this research was NetBeans. This choice was made because of its modularity and user friendliness. Besides, because its main developer is SUN, the developer of Java itself, probably a strong particular attention will be always given to all Java related features.

4.4.4 SQL the data management language

SQL stands for Structured Query Language. SQL is an ANSI (American National Standards Institute), standard computer language for accessing and manipulating databases. It is the standard language for relational database management systems. SQL statements are used to retrieve and update data in a database.

Over the last two decades, SQL has grown from its first commercial use into a computer product and services market segment worth tens of billions of dollars per year, and SQL stands today as the standard computer database language. Literally hundreds of database products now support SQL, running on computer systems from mainframes to personal computers and even handheld devices. An official international SQL standard has been adopted and expanded twice. Virtually every major enterprise software product relies on SQL for its data management, and SQL is at the core of the database products from Microsoft, Oracle, and IBM, the three largest software companies in the world. SQL is also at the heart

of open-source database products that are helping to fuel the popularity of Linux and the open-source movement.
Among the most diffused relational database management systems (RDBMS) in the world that use SQL, are: Oracle, Sybase, Microsoft SQL Server, and Access, and among the Open Source ones MySQL and PostgreSQL.
There are many different versions of the SQL language, but to be in compliance with the ANSI standard, they must support the same major keywords in a similar manner such as SELECT, UPDATE, DELETE, INSERT, WHERE, and others, that can be used to accomplish almost everything that one needs to do with a database. Most of the SQL database programs also have their own proprietary extensions in addition to the SQL standard (www.w3school.org/sql).
A list of general operations that SQL can operate is the following (www.w3school.org/sql):

- SQL can execute queries against a database,
- SQL can retrieve data from a database,
- SQL can insert records in a database,
- SQL can update records in a database,
- SQL can delete records from a database,
- SQL can create new databases,
- SQL can create new tables in a database,
- SQL can create stored procedures in a database,
- SQL can create views in a database,
- SQL can set permissions on tables, procedures, and views.

Java and SQL
The difference between the fundamental logics which are at the basis of JAVA, as object oriented programming language, and SQL, as language to deal with relational databases, has always been one of the main issues for computer developers. The conceptual difference

between the two logics, called object-relational impedance mismatch, will be explained and treated in details later on in this thesis. Different kinds of solutions have been proposed and are used in the market and in the open source community.
Therefore, concerning the particular case of JAVA and its integration with SQL has been one of the major areas of SQL development over the last five to ten years. Considering the need to link the Java language to relational databases, Sun Microsystems introduced Java Database Connectivity (JDBC), a standard API, developed by SUN, which allows Java programs to use SQL for database access. JDBC is the standard Java technology to RDBMS database connectivity. SUN gives a complete and always updated overview of Java database related new development, both relational and object oriented, in their related web page (http://java.sun.com/javase/technologies/database/). JDBC received a further boost when it was adopted as the data access standard within the Java2 Enterprise Edition (J2EE) specification, which defines the operating environment provided by all of the leading Internet application servers. In addition to its role as a programming language from which databases are used, many of the leading database vendors have also announced or implemented Java support within their database systems, allowing Java to be used as a language for stored procedures and business logic within the database itself. This trend toward integration between Java and SQL will ensure the continued importance of SQL in a new era of Java-based programming (Groff et al., 2002). This standardised solution addresses only the connectivity issue, meaning that it allows a system to interact with a database exchanging information. Even though SUN built the bridge for connectivity, the mismatching between the two logics remained. How to use the relational database to store object oriented information, this remains the main challenge of developers. Two different solutions will be proposed in this research. One dealing with the mismatch and the other one eliminating the mismatch with the help of object oriented databases.

4.4.5 GIS, JAVA and the Open Geospatial Consortium specifications

In the last 15 years, geographic information systems (as defined in paragraph 3.3.3) evolved from a highly specialized, technically "impenetrable" discipline into a highly disseminated crosscutting technology. GIS included hardware development, mapping, data production, data analysis, and complex software integration. At the same time and for this reason, GIS technology adapted to its new role becoming almost ubiquitous in several sectors (i.e. environment, transport, etc.) interacting with different kinds of hardware, data storage tools, and software, therefore with new programming languages. As often happens for new and innovative technologies, at the beginning GIS producers and developers did not worked towards standardisation and integration keeping the high costs for customisation, interoperability and development of new GIS oriented services. With the success of GIS in the market, the need for using GIS with JAVA became significant. One of the reasons why this link between GIS technologies and JAVA as its developing language has been increasing, according to Andrews (2004), can be identified with the increasing need of GIS for integrability and interoperability, main features and reason for the development of JAVA itself. In fact, as explained in previous chapters, the fundamental concern around this new programming language, when it was first released, focused on the interoperability on multiple operating system platforms, ability which is called portability.

As the GIS industry stared looking forward also to the World Wide Web in order to provide improved based mapping and interactions functionality with spatial distributed computing over the Internet, Java appeared also strongly as a practicable and high level option for integrating GIS into the Web. Besides, being Java portable also across a range of different off-line hardware solutions (mobile phones, palmtops, pocket pc, etc...), it responds well to the need of a broad variety of hardware used in water resources management.

Java, however, was a free but not open source language, this made it less attractive for pure FOSS (Free and Open Source Software) developers (see e.g. http://www.gnu.org/philosophy/java-trap.html) until 2006 when the migration of JAVA toward the open source world started with the release of part of its code under the open source GNU General Public License, GPL2 and then GPL3. In spite of this, while there were other truly open source programming languages that can be used to develop GIS technology, the power and the features of Java programming language allowed the development of several

GFOSS (Geographic Free and Open Source Software) applications since its beginning, even long before 2006. Besides, the growing computing power of modern PCs makes Java programs adequate even for large applications. Several Java GFOSS programs are particularly interesting; for instance, Deegree (www.deegree.org) is a good map server, fully compliant with OGC Web Services specification as well as Clients and security components. For desktop mapping and geographical data analysis, gvSIG (www.gvsig.gva.es) is a multilingual, open source GIS developed in JAVA that can handle both vector and raster data. Java Unified Mapping Platform (http://www.vividsolutions.com/jump) has attractive features. For GRASS (C++ based project) users, a Java graphical interface is also available (http://www.hydrologis.com/html/jgrass/jgrass_en.html). Several other similar projects can be found on the Internet and new ones are continuously developed.

More than just Java GIS development, in the last years, the growing amount of similar projects developed, fostered standardisation. An interesting example of the trend to work toward standardisation and open source GIS technologies with Java is a project by Vivid Solutions, Inc. (Davis and Aquino, 2003). They implemented a robust topology toolkit called the JTS Topology Suite (JTS). The JTS, an open source Java project, is an API providing spatial object model and fundamental geometric functions. It implements the geometry model defined in the Open Geospatial Consortium Simple Features Specification for SQL. JTS provides a complete, consistent, robust implementation of fundamental 2D spatial algorithms, allowing Java programmers to perform 2D spatial operations such as identifying polygons and intersecting shapes.

The development of such standardised tools allows JTS based software to be integrated with other standards-based OGC compliant Java GIS projects. Another similar effort toward standardisation is Geo Tools (Garnett, 2007), an open source (LGPL) Java code library which provides standards compliant methods for the manipulation of geospatial data. The GeoTools library implements Open Geospatial Consortium specifications. Geotools is used by a number of other open source and commercial projects which building on its API are compatible among them.

Community supporting for GIS and Java integration grows every year above all in the open source world. This kind of market model has been pushed by the development of standards by the OGC helping developers to build applications with a set of predefined rules for constructing geospatial applications, and helping them to build on existing open source projects. One of the most important examples of this growing community is The Open Source Geospatial Foundation, or OSGeo (www.osgeo.org). The OSGeo is a non-for-profit organization whose mission is to support and promote the collaborative development of open geospatial technologies and data. The foundation provides financial, organizational and legal support to the broader open source geospatial community. These kinds of initiative support more and more OGC standardisation and the development of the open source market approach model for GIS projects. Concerning commercial software projects, also numerous proprietary Java GIS packages already exist. MapInfo Corporation (www.mapinfo.com) started exploring the Java GIS world commercializing solution with an IT-standard Java mapping package called MapXtreme Java.

The Environmental Systems Research Institute (ESRI), developed a Java oriented solution for ArcView IMS (www.esri.com/software/arcview/) with a Java Graphical User Interface. Besides, emphasising a stronger interest in Java technologies for GIS, ESRI ported its MapObjects platform to a MapObjects Java version. The market's Java-based GIS solutions are rapidly evolving from inelegant GUI-focused toolkits to fully functional geospatial advanced programming interfaces oriented toward interoperability and integration.

We can conclude that Java technology has entered the GIS world and it is at the core of many new projects representing now a key component of GIS integration with the support of OGC standards. Besides, Java is benefiting from the trend toward the development of open source technology, a marketing approach for development alternative, a major facilitator for open source implementation.

Several new open source tools such as JTS, GeoTools, or commercial tools such as the Oracle Spatial Java API (www.oracle.com/technology/software/products/spatial/) allow Java programmers unfamiliar with GIS technologies to implement in their projects with complicated GIS functions in order to store and treat geospatial data. Therefore, the development of GIS solutions in Java technology has helped GIS itself in becoming more present not just as an improving functionality for the graphical user interface level but as an integrated tool of information systems in the IT industry. These are just some of the examples of Java (and not only) related projects concerning GIS. In the next section of the thesis a deeper analysis, together with a projects' benchmarking, will expose the reasons which brought us to choose a specific project for this research.

OGC standards

The Open Geospatial Consortium, Inc. (OGC) is a non-profit, international, voluntary consensus standard organization that is leading the development of standards for geospatial and location based services. Through member-driven consensus programs, OGC works with government, private industry, and academia to create open and extensible software application programming interfaces for geographic information systems and other mainstream technologies. Adopted specifications are available for the public's use at no cost (Nebert, 2007 Since 1999 the OGC has been researching the feasibility of heterogeneous 'open' GIS systems that achieve interoperability and break with a history of proprietary 'closed' GIS technology. The mission of OGC is "to deliver spatial interface specifications that are openly available for global use", the specifications aiming to solve the "interoperability problem" in the GIS world. These problems come from industry, government and academia and span many topics. Once an interoperability problem is identified, the members work together to define requirements for a new interface specification or enhancements to an existing OpenGIS Specification. Once a specification is approved, it is made publicly available on the OGC website, without cost. Specifications are simply engineering documents that describe how the OGC membership has agreed to solve an interoperability problem.

An overview of the most commonly used OGC specifications can be found in Vretanos (2005) for the Web Feature Service (WFS), in Evans (2006) for the Web Coverage Service (WCS), and the Web Map Service (WMS) is described in detail in de la Beaujardiere (2006) give an overview of specifications co-maintained by the International Standardization Organization (ISO).

In conclusion, in order to comply with the requirements of our information system, we have focussed on the choices of open source GIS projects, OGC specifications and Java compatibility.

4.4.6 The Database/Geo-database

A Geo-database, known as “geographic database”, is a database capable of managing geographic data.
As the use of GIS has increased, the concept of water resources data, evolving in water resources management, has broadened to include geospatial data describing the water resource features of the landscape. This geo-database is build following a new concept, rather than simply applying GIS in water resources, it is possible to think that time series data on water properties (quality and quantity) and geospatial data on the water environment are both information sources that the information system and the end user wants to use.
To strengthen this synergy between geospatial data and temporal water resources information, it is fundamental not holding the information in different formats and archiving environments. Traditionally, to store geographic information, GIS users have depended upon files. One of the most used of this kind of format is probably the ESRI's shapefiles or computer aided drafting files such as DGN/DWG. Where DGN is the name used for CAD file formats supported by Bentley Systems' MicroStation (www.bentley.com/) and Intergraph's Interactive Graphics Design System (http://www.intergraph.com/) CAD programs.
The probably most wide diffused is the shapefile format. According to ESRI (1998), shapefiles are a simple, non-topological format for storing the geometric location and attribute information of geographic features. The shapefile format defines the geometry and attributes of geographically referenced features in three or more files with specific file extensions. Apart from their ease of use and marketing reasons, the file-based GIS data storage methods do not provide the performance, security, and accessibility necessary for a GIS to support a wide IWRM platform with a high level of heterogeneous integrated end uses. Besides they do not provide management systems supporting client/server architecture as instead the typical RDBMS.
Geo-databases overcome many of the limitations inherent in file-based GIS data storage architectures. Modern enterprise geo-databases offer a number of advantages over traditional shapefiles and coverage, including the ability to:

- Support multiple concurrent GIS users and workflows,
- serve GIS data in high demand environments,
- securely store GIS data and protect against unauthorized data access,
- allow long transactions, data versions, and other real-world scenarios,
- store many GIS features in a single, seamless, spatially indexed database,
- enable sophisticated GIS data models, data validation and spatial business rules.

A major benefit of the geo-database is the ability of GIS staff throughout an enterprise to centrally store and share geo-spatial data (vectors, rasters, topologies, measures, addresses, CAD, etc.). It is supported by the client/server architecture of the RDBMS.
The traditional solution to all this has been to use specialised database systems that use proprietary spatial indexing, and proprietary interfaces.

4.4.6.1 Linking relational databases with GIS functionalities

A relational database basically consists of a set of interconnected tables, where each table stores one sort of data relevant to the application (more on RDBMS and on the difference with OO DBMS in the next chapter). More than by RDBMS database developers, the need to combine SQL with GIS technologies was firstly determined by the developers of GIS applications in order to make their applications and services more powerful and capable of an integrated data management.
The organization, which took the task of formally establish and expand GIS standards to broader markets making GIS technologies available everywhere, is the Open Geospatial Consortium that we already presented. The idea of Geo-database now is combining the power of typical relational database management systems (RDBMS) with SQL spatial extensions of the Open Geospatial Consortium specifications.
These specifications are used as the baseline for almost all implementations of GIS functions within an SQL-based relational database. This standard defines the data types (figure 4.5),

operations, input and output format, functions and much more. This is the standard that is followed by almost all SQL databases with spatial extensions, including MySQL, ORACLE, PostgreSQL + PostgreGIS, etc. (Karlsson, 2003).

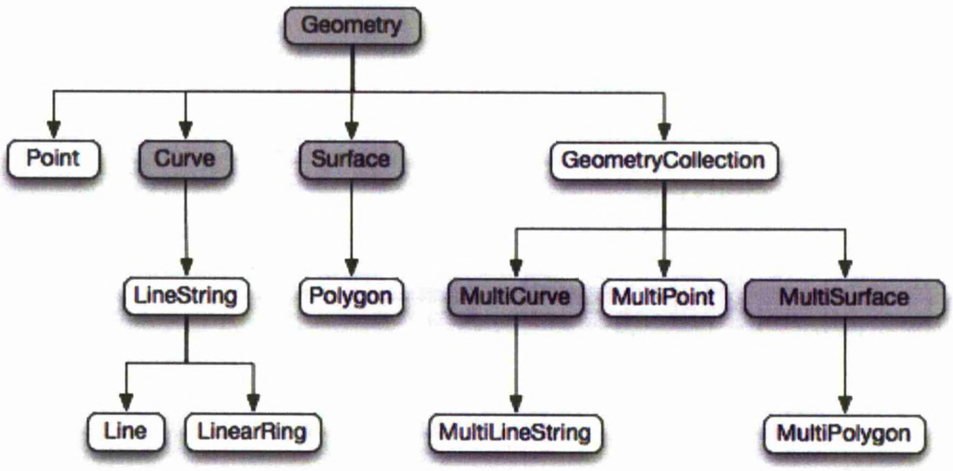


Figure 4.5: MySQL GIS Data types (abstract types in gray).

The geo-database, in the architecture of our information system, corresponds to the "data layer" and to the lowest tier of the N-Tier model. It aims not only to be the knowledge base for the information system, but also a structured data model supporting analysis and modelling in the whole information system. "Water resources data" includes time series data on observations of water resources phenomena, water quantity, rainfall, stream-flow, water quality, climate data etc.

Besides, more than being just a storage layer, the RDBMS can be considered as a whole information system dedicated to data storage. In fact, the RDBMS has a series of business tools and procedures that allow the rest of the information system to deal with it only through a series of simple commands with SQL. Moreover, typical RDBMS provide already a set of other fundamental functionality, as the client/server architecture support, and organization, storage, access, security management and integrity of data and eliminating data redundancy.

In conclusion databases are a fundamental piece of the information system that can be identified as the data-layer of N-tier architecture and also pieces of software including all the necessary functionality to data management systems.

4.4.6.2 The Water Basin data model

It has already been pointed out that all the information flowing from one component to another component of the information system should always be treated through the knowledge management component (the geo-database) of the system. Data are sent to and stored or modified in the database and then retrieved by another component of the system. The information, other than being only stored or transferred through the database, it is also structured according to the water basin data model design and implementation.
It is possible to find in the literature some interesting and relevant hydroinformatics projects in which geo-databases implement water catchment data models, as for instance in the information systems based on the Arc Hydro model (Maidment, 2002).
Arc Hydro is a data structure that provides the capacity to link hydrologic data to water resources modelling and decision making methods. In this way water resource models included in the information systems can be more closely integrated with GIS (Maidment, 2002). A detailed presentation and explanation of the data model will be given in the next chapter.
For the moment it is important to underline that the data model implemented in the geo-database of our information system is a relational oriented representation or adaptation of the object oriented data model designed in Arc Hydro to represent the water basin system.

4.4.6.3 The relational model in database management systems

The most popular databases, commercial and open source, currently in use are based on the relational model. The relational model was introduced by Codd (1969) and is the logic at the basis of relational databases. The software programs which implement relational databases are the relational database management systems (RDBMS). From the early 70's there have been many kinds of different interpretations and implementations of this model. According to Codd, in order to be compliant with the relational model, implementations should at a minimum respect the following common factors:

- present the data to the user as relations (a presentation in tabular form, i.e. as a collection of tables with each table consisting of a set of rows and columns),
- provide relational operators to manipulate the data in tabular form.

Strictly speaking and as defined by Codd (1969), a relational database is a collection of relations (commonly called tables). Other items are frequently considered part of the database, as they are part of the organisation of the structure of the data and they are also needed to the database in order to conform to a set of requirements. These principles are translated by Codd in a set of 12 rules that can be defined as a check list or guidelines for relational models compliancy and that are listed here with the original name given by Codd (1969, 1990) and with comments from Parkhurst (2002):
Rule 1: The Information Rule
All data should be presented to the user in tabular form.
Rule 2: Guaranteed Access Rule
All data should be accessible without ambiguity. This can be accomplished through a combination of the table name, primary key, and column name.
Rule 3: Systematic Treatment of Null Values
A field should be allowed to remain empty. This involves the support of a null value which is distinct from an empty string or a number with a value of zero. Of course, this cannot apply to primary keys. Also, most database implementations support the concept of a nun-null field constraint that prevents null values in a specific table column.
Rule 4: Dynamic On-Line Catalog Based on the Relational Model

A relational database must provide access to its structure through the same tools that are used to access the data. This is usually accomplished by storing the structure definition within special system tables.

Rule 5: Comprehensive Data Sublanguage Rule

The database must support at least one clearly defined language that includes functionality for data definition, data manipulation, data integrity, and database transaction control. All commercial relational databases use forms of the standard SQL (Structured Query Language) as their supported comprehensive language.

Rule 6: View Updating Rule

Data can be presented to the user in different logical combinations, called views. Each view should support the same full range of data manipulation that has a direct access to a table. In practice, providing update and delete access to logical views is difficult and is not fully supported by any current database.

Rule 7: High-level Insert, Update, and Delete

Data can be retrieved from a relational database in sets constructed of data from multiple rows and/or multiple tables.

Rule 8: Physical Data Independence

The user is isolated from the physical method of storing and retrieving information from the database. Changes can be made to the underlying architecture (hardware, disk storage methods) without affecting how the user accesses it.

Rule 9: Logical Data Independence

How a user views data should not change when the logical structure (tables structure) of the database changes. This rule is particularly difficult to satisfy. Most databases rely on strong ties between the user view of the data and the actual structure of the underlying tables.

Rule 10: Integrity Independence

The database language (like SQL) should support constraints on user input that maintain database integrity. This rule is not fully implemented by most major vendors. At a minimum, all databases do preserve two constraints through SQL.

Rule 11: Distribution Independence

A user should be totally unaware of whether or not the database is distributed (whether parts of the database exist in multiple locations). This is difficult to implement.

Rule 12: Non-subversion Rule

There should be no way to modify the database structure other than through the multiple row database language (like SQL). Most databases today support administrative tools that allow some direct manipulation of the data structure.

Relational databases, implemented in relational database management systems, have become a predominant choice for the storage of information in new databases used for financial records, manufacturing and logistical information, personnel data and much more.

Nowadays there is a huge list of different commercial, free and open source RDBMS. Concrete implementations of the relational model have oriented the evolution of the concept of relational model itself from the one defined by Codd. Therefore, practically the most accepted definition of a RDBMS is a product that represents data as collection of rows and columns (organised as tables), even though it is not based strictly upon relational theory, meaning that they typically implement some but not all of the 12 rules defined by Codd.

Relational database theory uses a different set of mathematical-based terms, which are equivalent, or roughly equivalent, to SQL database terminology, the language used to deal with RDBMS:

Relational term	**SQL equivalent**
relation base relvar	Table
derived relvar	view query result set
tuple	Row
attribute	Column

Where *relvar* is a term used by Codd meaning "relation variable".

In the relational model logic the kinds of relationships to set up between the relational elements (the tables) are:

- One-to-one: Both tables can have only one record on either side of the relationship. Each primary key value relates to only one (or none) record in the related table. Most one-to-one relationships are forced by business rules and don't flow naturally from the data.
- One-to-many: The primary key table contains only one record that relates to none, one, or many records in the related table.
- Many-to-many: Each record in both tables can relate to any number of records (or no records) in the other table. Many-to-many relationships require a third table, known as an associate or linking table, because relational systems can't directly accommodate the relationship.

The concept that will be discussed in next paragraph will oppose the relational model and rather discuss the object model which is the main model used in our information system.

4.4.6.4 The object to relational database mapping

Except the knowledge management component (the RDBMS), every component of the system, being based on Java, is programmed following the object oriented logic. Therefore the data exchange flow between these two components, based on different technologies, is a continuous and fundamental task influencing the performances of the whole information system. As a matter of fact, object oriented technologies and relational technologies are in common use in most organisations, and both are being used together to build complex software-based systems. Nevertheless,, the combination between the two technologies, and even more between the two logics, is problematic. This "conflictive" integration and interoperability is called the "object-relational impedance mismatch" in informatics (Ambler, 2003).
"Impedance mismatch" is a definition originated from electrical engineering and used in system analysis identifying the inadequate or excessive ability of one system to accommodate input from another (Ambler, 2006). It is caused from the fact that OO logic is based on

software engineering principles that model the objects in the problem domain, while the relational model is based on mathematical principles that organize data for efficient storage and retrieval (Paterson, 2004). One solution to overcome this problem is the process of "mapping" objects to relational databases. Where mapping is meant as the act of determining how objects and their relationships are persisted in permanent data storage, in this case relational databases (Ambler, 2003).
Basically, the mapping can be simplified as if object oriented classes have to be mapped to relational tables and classes' properties to tables' columns. Except for very simple databases, it is not likely to have a one-to-one mapping of classes to tables as relational databases do not natively support inheritance. Since Java is a single inheritance language, the mapping techniques we took in account are (Ambler, 2003):

- Map the entire class hierarchy to a single table,
- map each concrete class to its own table,
- map each class to its own table,
- map the classes into a generic table structure.

We choose, both for simplicity and completeness, to map each class to its own table. Following this strategy it will be created one table per class, with one column per attributes and any other necessary information. The use of the primary key and the foreign key is fundamental to map accurately relationships between classes. The inheritance relationships are simply simulated bringing in every inheriting class all the attributes of the inherited classes.

4.4.7 The Graphical User Interface

A graphical user interface (GUI) can be defined as an interface which allows a user to interact with a computer based information system. There are several elements in a GUI (i.e. pointers, windows, etc.) and devices (i.e. mouse, keyboards, etc.) which allow the user to interact with

the different elements of the GUI. The way these are designed and combined influences the performance of the GUI in facilitating the interaction between the user and the system.
When designing a GUI, the developer has to take into account the fact that the main objective of the users is to have their tasks done as soon as possible spending as small time as they can in understanding how the GUI has to be used. The GUI and the information system are the mean to accomplish an objective and not the objective in itself. Together with the users, the organisations increase their performances when the task is accomplished as quick and as easy as possible. Reducing the time that every human resource has to dedicate to a particular task and reducing costs for trainings should be the main objective of every GUI.
The organisations responsible for different tasks related to managing a water basin can perform better when more GUI developers succeed in reducing time and skills needed for using an application. In the case of software which is not going to be used by a vast public, such as our information system, the GUI is particularly designed for specialised experts. In order to design a GUI, two key points are mentioned by Cooper (2003) which are given below:

- "Imagine users as very intelligent but very busy"
- "No matter how friendly your interface is, less of it would be better"

Therefore, even if our information system is dedicated to technical experts (very knowledgeable users), simplicity and usability remain always the most important requirements.
From the software architecture point of view, the GUI corresponds to the presentation layer. Keeping in mind that the information system can be used both for modelling and monitoring purposes in IWBM, the views and controllers components of the system often shall work very closely together. For instance, a controller shall be responsible for updating a particular parameter in the model which is displayed by a view. In some cases a single object will function as both a controller and a view. Each controller-view pair is associated with only one model (simulation or data model); however a particular model can have many view-controller

pairs. The GUI will be built in different layers and can be easily extended as a new component added to the system. The general requirements of the information system (integrability, extensibility, portability and open standards based) remain valid at each level and for every component.

4.4.8 Application of simulation models

Models are one part of the larger hydroinformatics software system; they are developed like separated modules in separated layers. Models implementation will be supported either as plug–in components or as predefined composites of these components. Such composites should behave identically to basic plug–in components, and in particular should be themselves modular. It is possible to define three different kinds of models: data management, data analysis and simulation models. The first one shall include the applications that allow the user managing and controlling the data layer from the GUI. The second one shall be a package of tools allowing the user to carry out analysis on the data, retrieving the data directly through the standard interfaces. The third one includes simulation models which need the pre- and post- processing of the retrieved and updated data, aligning input and output. It is the case that the model engines are written in other languages (C, C++, Fortran, etc.) than the common components core language (Java), and the pre- and post- processor work as a wrapping layer around the engine providing the information in the right format and translating the model results to the system data exchange standard (SQL + Java).

CHAPTER V

5 IMPLEMENTATION AND CASE STUDY

5.1 Introduction

To test design, architecture and validity of the system a pilot implementation of the information system is developed using proposed methodologies and technologies discussed in the previous chapters. To validate the system, a case study with some concrete examples is conducted for the Upper Mersey (UK) river basin according to requirements presented in previous chapters. The research aims to show how the approach followed can embrace a wider range of software tools as it can support integrated water basin management. All software used to build and implement the information system is open source. The Operating System (OS) has been Windows by Microsoft Corporation (www.microsoft.com) and given the portability of the core programming language of the IS, Java, both client and server side of the information system can be used in every other OS supporting the Java Virtual Machine (e.g. MacOS, Linux, Solaris, etc.). In order to test the portability of the system some simplified tests has also been carried out for an open source OS, the Linux distribution SUSE 10 (www.opensuse.org).

Technologies presented in previous chapters are widely available in the open source and in the commercial market under several different implementations. They are developed and maintained by many different companies, open source communities and projects. Different implementations with different features are oriented to different purposes. Performances of the single components have a strong influence on the overall performances of the IS.

5.2 Development of the Information system

The definition of the research methodology and of the technology analysis provided a support framework to strategically approach the development of the information system.

In previous chapters, all the components to be implemented in the IS have been defined, also the implications of their interactions have been studied. Besides, going through the open source and open standard existing technologies, we acquired the necessary overview towards an appropriate approach to software development.

After designing the information system architecture and following the analysis of the software components to be integrated, the following step for the implementation of the system has been developed so that all the components can interoperate with each others. This programming exercise, which has as result a series of application interfaces, has been possible thanks to the fact that all the components chosen are open source and hence freely modifiable. Therefore it has been simplified through the use of components based on available standard technologies as previously defined. Beside the application interfaces, also the GUI and the simulation models have been entirely developed or adapted to the IS. In the quest for the best possible results on integration and interoperability, every other component brought into the system has been first prepared to be compatible with Java through developing the needed "glue code" using and adapting existing API (application programmes interfaces) or developing new ones. Best available standard open technologies have then been adapted toward *ad hoc* IWBM functions; chosen in particular and adapted to the case study.

5.3 Information exchange and data paths

The essential components of IS architecture are shown below (figure 5.1). In the n-Tier model (cf. 4.2.3), different layers with different responsibilities accomplish different tasks.
Every layer includes different components and software tools able to process data and others able to deal with the adjacent levels acting as interfaces. The process of integrated water resources management can be complex and take into consideration several end users and organisations. This is why the 3-tier approach can be considered useful in IWRM when compared to the two tier approach. In fact, the three tier client/server architecture has been used to improve performance for groups with a large number of users and improves flexibility when compared to the two tier approach (Helal et al., 2001).
In general, when dealing with a series of different water resources management models that have to interoperate, a middle layer used as an applications server, is a strategic support to the software infrastructure. Besides, considering the founding principles of the integrated water resources management approach, several actors (such as management agencies and

organisations) are supposed to make decisions and take actions while considering multiple viewpoints and scenarios on how water should be managed. From an informatics point of view, the 3-tier architecture is defined where a middle tier is added between the user system interface in the client side and the database management server environment. This middle tier runs as a middleware. Middleware software consists of a set of enabling connectivity services that allow multiple processes running on one or more machines to interact across a network. When dealing with distributed computing, a middleware is mostly implemented as an application server.

During the implementation of this IS, the structure has been made simpler and for the prototype of the system, the only layer implemented in the server side is the database server, while the others work in the client side. Further development efforts will increase the server oriented approach, thereby reducing client software applications and the amount of computational work executed in the client machine. In figure 5.1 it is possible to differentiate between the present implementation and the possible improvement of the client/server architecture towards a purer 3-tier architecture. From a hydroinformatics point of view, the fundamental piece of work to implement a 3-tier architecture is the structuring and implementation of the three different levels, presentation, business and data layer.

In this implementation, the hydroinformatics modelling system will reside at the server side, and the client will have controlled access to it (account and password) managed by the application server. Also the interface layer will work at the server side. At the client side it will integrate a user friendly graphical interface for passing the input data and displaying the results to the end user.

It is important to mention that the structure of this information system has been designed as Web ready 3-tier architecture. A Web presentation server (the presentation layer), the application server (the business layer) and a database server (the data layer) shall all constitute the server side of an information system that has at the client side only an Internet Web browser (Holz et al., 2004). The only addition in comparison with the 3-tier approach above presented is that also the presentation layer should work though a Web server. Having a Web GIS as presentation layer would already be a solution.

In order to design the business layer in line with the 3-tier approach, in this phase it is worthwhile to identify possible solutions for the implementation of the different layers. These different solutions imply different scenarios for the architecture with different levels of implementation of the distributed computing architecture.

Considering the business layer, in order to manage a series of modelling or management tools in a framework of a distributed architecture, the already mentioned solution of implementing an application server requires more in-depth studies and developments for scaling up the information system. An application server is a component-based piece of software which is part of the middle-tier in multi-tier server based architectures. It provides middleware services for security and state maintenance, along with data access and persistence needed for distributed computing.

The distributed nature of the IWBM management presents particular challenges. When dealing with a distributed and integrated information system, managers can easily lose sight of the fact that all components are part of the same system and should be treated as such. Besides in complex systems, with the need for interoperability of several modelling and management tools, application servers become a very important piece of software in the whole system architecture.

During the implementation of the IS, what the system really needs to do is to communicate to the database server. Therefore, Java technologies are able to work directly with a RDBMS through the appropriate technology and interfaces (as showed in the following paragraph). Besides the RDBMS have the appropriate features to work also as database server and hence Web servers or application servers are not strictly fundamental.

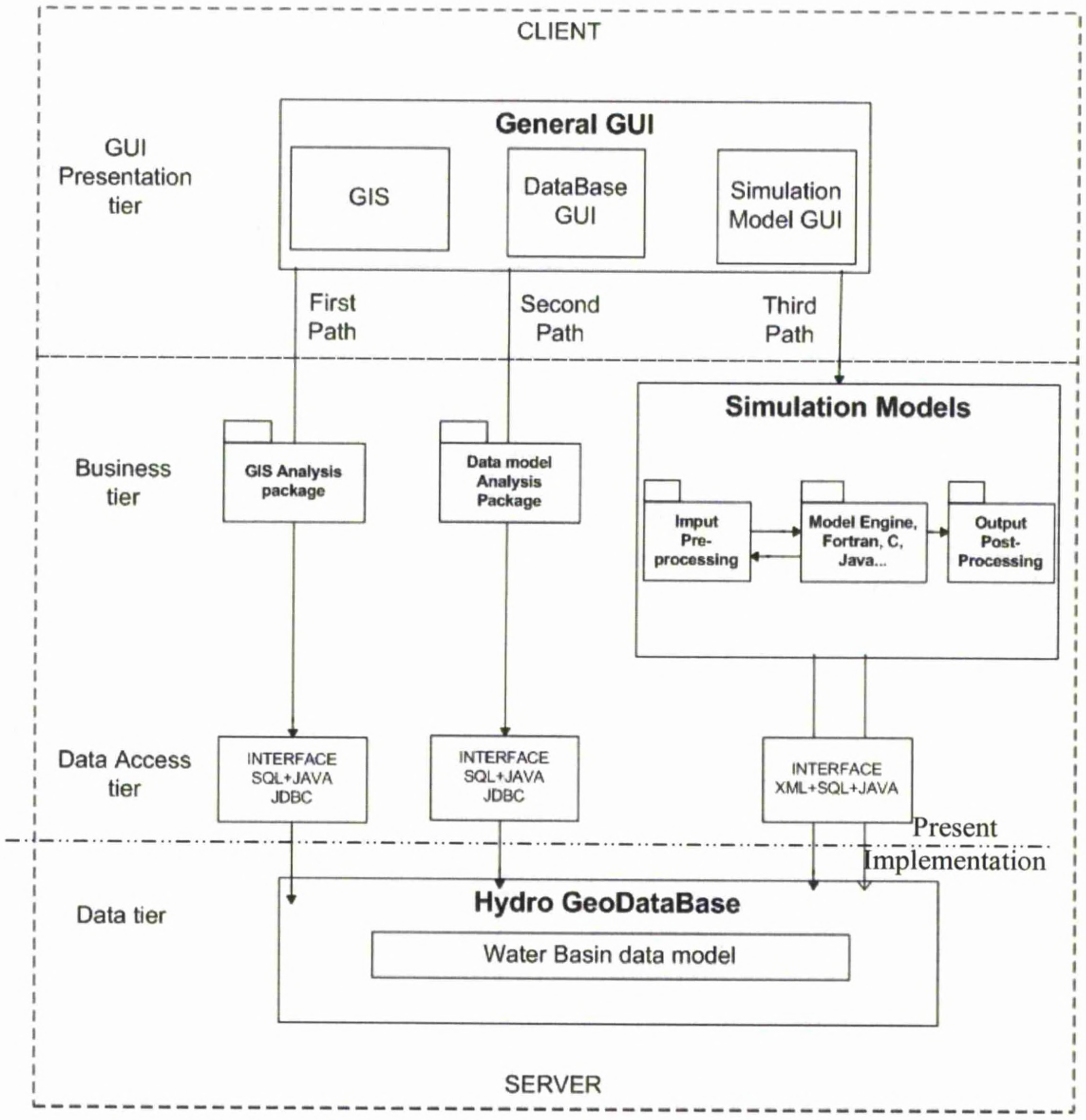


Figure 5.1: the IS architecture and the data paths.

Another solution to implement direct distributed computing in the particular case of JAVA technologies is the Java Remote Method Invocation (RMI) (http://java.sun.com/products/jdk/rmi/).

Java-RMI is a Java-specific middleware for distributed computing. Remote Method Invocation (RMI) is the object equivalent of Remote Procedure Calls (RPC). While RPC allows calling procedures over a network, RMI invokes an object's methods over a network.
In the RMI model, the server defines objects that the client can use remotely. The clients can invoke methods of this remote object as if it were a local object running in the same virtual machine as the client. RMI hides the underlying mechanism of transporting method arguments and return values across the network. In Java-RMI, an argument or return value can be of any primitive Java type or any other serializable Java object (java.sun.com).
In our IS, from the user interface application in the client side, the required remote method, belonging to a class in the server, will be called; it will execute the necessary routine and send back results to client application. Finally the client application will be utilized or simply to visualize the received data.
Besides the horizontal organisation and focus of the information, in the structure shown in figure 5.1 it is also useful to classify the vertical channelling. In fact, between the fundamental layer (the data tier) and the layer on the top (the GUI) it is possible to identify three different data paths. Every path treats information with a different analysis and is oriented to a different purpose. In the first path, the GIS, encapsulated in the GUI or presentation tier, uses the business layer both to retrieve the information from the database in an appropriate way and to execute GIS analysis typically required in hydroinformatics, like visual queries and elevation models analysis (slope calculation, flow path identification, etc.) This analysis package in the first prototype has been only partially implemented. In the second path, the aim is both to manage the database and to execute simple and basic analysis on the data model, the level of data analysis increases; in the third path, the data are processed at a higher level. Here it is necessary to separate the execution in three phases, pre-processing, core engine process and post processing. The core engine can be based on other programming language, both procedural and object oriented.

5.4 Interfaces implementation

In the n-Tier architecture, the interface layer has a central role; its task is to set up the communication protocols between the business layer and the knowledge base of the whole information system, the data layer. As it has already been argued it is necessary to establish a set of standard communication channels to allow the different applications of the business layer to access the data base, retrieving, modifying and storing the information. The interface will be responsible for connecting the Java components of the system with the geo-database. The interface will be made of two layers, two APIs (Application Program Interface), one dealing with the geo-database, the other with the other Java components of the system. The two APIs will be able to communicate between them and will be based on SQL and Java. The API communicating directly with the geo-database is the JDBC (Java Database Connectivity) driver. It allows Java programs to be connected to an RDBMS database using standard, database independent Java code. It is a pure Java implementation. Call-level interfaces such as JDBC are programming interfaces allowing external access to SQL database manipulation and update commands. They allow the integration of SQL calls into a general programming environment by providing library routines which interface with the database. JDBC API is the industry standard for database-independent connectivity between the Java programming language and a wide range of databases as for instance MySQL and Postgree SQL, the ones used in this research. The JDBC API provides a call-level API for SQL-based database access. JDBC technology is also portable following the "Write Once, Run Anywhere" capabilities for applications that require access to multiple sources of data.
The JDBC API allows the different layers of the information system to:

- Establish a connection with a database or access any tabular data source,
- send SQL statements,
- process the results.

Concerning its architecture (shown in figure 5.2), The JDBC API contains two principle series of interfaces: the upper level, the JDBC API for application writers, and the other level, the

JDBC driver API for writers. JDBC technology drivers fit into one of four categories. Applications and applets can access databases via the JDBC API using pure Java JDBC technology-based drivers, as shown in the below figure (Andersen, 2006):

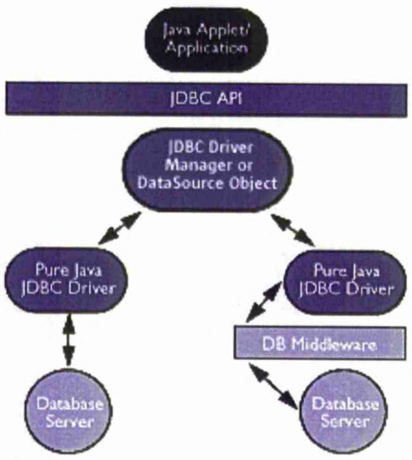


Figure 5.2: The JDBC access scheme.

For every other component of the IS a pure Java API able to communicate with RDBMS through the JDBC will be built. For some of the components, the interface will be only composed by a single layer, the JDBC. This is either because of the simplicity of the component or because our customized applications are built directly to communicate with JDBC.

5.5 Open Source Database projects comparison and selection

Given the complexity of modern database management systems there would be many considerations to take into account when choosing a database platform and not just purely technical. As for instance they can include feature completeness, vendor support and community support, performance and optimization. An organisation will not invest without knowing details about system's requirements and about the application to be built. In the end

those requirements may not be easy to identify, and even harder it could be to define them, but our experience shows that it is worth to invest time and resources on hard thinking towards cost-effective and robust solutions. It could seem obvious but in many projects does not seem to be, the design phase needs to be based on solid considerations.
Over the last couple of years, a number of open source databases have been developed becoming important alternatives to commercial databases. Projects such as MySQL (www.mysql.com) have strongly benefitted from becoming open source experiencing a significant market share increase over their biggest and previously unreachable competitors such as Oracle and Sybase. In realty MySQL marketing model is quite complex and not just open source license based, but its gains from the open source model are incontestable. In fact, MySQL offers a mix of open source and commercial licence which can be adapted to B2B (business to business) and B2C (business to customer) needs.
Other important competitors of MySQL in the open source database world are PostgreSQL (www.postgresql.org), Ingres (www.ingres.com), FireBird (www.firebirdsql.org), etc. A complete list of other open source database project would be too long and out of its scope. Therefore, this research basically focuses on the two most competitive existing databases such as MySQL and PostgreSQL. A brief description of the two database management systems with comparison is given below.

PostgreSQL (www.postgresql.org) and PostGIS (postgis.refractions.net)
One of the open source projects chosen to implement the database layer for our research is PostgreSQL (http://www.postgresql.org/). It has a spatial extension known as PostGIS (http://postgis.refractions.net/). PostgreSQL is an object-relational database management system (ORDBMS) based on POSTGRES, Version 4.2, developed by the Computer Science Department at the University of California at Berkeley. POSTGRES pioneered many concepts that only became available in some commercial database systems much later. PostgreSQL is an open-source descendant of this original Berkeley POSTGRES code. It supports SQL92 and SQL99 and offers many modern features:

- complex queries;

- foreign keys;
- triggers;
- views;
- transactional integrity;
- multiversion concurrency control.

Also, PostgreSQL can be extended by the user in many ways, for example by adding new:

- data types;
- functions;
- operators;
- aggregate functions;
- index methods;
- procedural languages.

PostgreSQL works on many different system platforms, including FreeBSD, Linux, Mac OS X, NetBSD, Novell NetWare, OpenBSD, OpenSolaris, Symbian, SunOS, and Microsoft Windows, a much longer list is available in the web site. Besides, because of the open source license (It is released under a BSD-style license, technically BSD licenses represent a family of permissive free software licenses.), PostgreSQL can be used, modified, and distributed by everyone free of charge for any purpose, be it private, commercial, or academic (http://www.postgresql.org/).

Even though PostgreSQL claims to be an object-relational database but for the prototype of our information system the object oriented (OO) features are not used. Nevertheless in the future developments the OO features of PostgreSQL will be taken into account and probably be implemented in the system, being the test of the object oriented technologies applied to a relational database one of the research objectives also presented in the last chapter of the thesis with the implementation of another OO database management system.

A particular mention when using PostgreSQL in GIS environment is to be given to PostGIS (http://postgis.refractions.net/). PostGIS is an open source software program that adds support for geographic objects to the PostgreSQL. PostGIS gives a strong impulse to PostgreSQL performances and features making the combination probably one of the most complete solutions available nowadays. PostGIS is a standardised solution since it follows the Simple Features for SQL specification from the Open Geospatial Consortium.

In particular, as mentioned in its web site PostGIS includes:

- Geometry types for points, linestrings, polygons, multipoints, multilinestrings, multipolygons and geometrycollections.
- Spatial operators for determining geospatial measurements like area, distance, length and perimeter.
- Spatial operators for determining geospatial set operations, like union, difference, symmetric difference and buffers (provided by GEOS).
- R-tree spatial indexes for high speed spatial querying.
- Index selectivity support, to provide high performance query plans for mixed spatial/non-spatial queries.

The first version was released in 2001 by Refractions Research under the GNU General Public License, and development has continued since then actively. In 2006, PostGIS was certified as a compliant Simple Features for SQL database by the Open Geospatial Consortium.

MYSQL (www.mysql.com)

MySQL is a relational database management system which, according to its developers, has more than 11 million installations. The software runs with a client server architecture providing several users to access several databases. MySQL is owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, which became recently subsidiary of Sun Microsystems. MySQL AB is the owner of the copyright to most of the codebase.

MySQL is an open source product available under terms of the GNU General Public License. Besides, MySQL is also a commercial software sold under a variety of proprietary agreements.

In fact, both the MySQL server software and the client libraries are distributed under a dual-licensing format. Users may choose the GPL, which MySQL has extended with a FLOSS License Exception. MySQL popularity is strongly linked to PHP popularity, being PHP an open source scripting language designed for producing dynamic web pages. The two together are one of the most diffused solutions for web applications. In fact several high-traffic web sites (including Flickr, Facebook, Wikipedia, Google (not for searching), Nokia and YouTube) use MySQL for its data storage and logging of user data.

Concerning portability, MySQL has strongly developed this aspect working on many different system platforms, including FreeBSD, Linux, Mac OS X, NetBSD, Novell NetWare, OpenBSD, OpenSolaris, Symbian, SunOS and Microsoft Windows. When integrating MySQL in an information system it is also fundamental the availability of Libraries for accessing MySQL databases from different components of the system. Libraries are available in all major programming languages with language-specific APIs. In addition, an ODBC interface called MyODBC (Open Database Connectivity (ODBC) provides a standard software API method for using database management systems) allows additional programming languages that support the ODBC interface to communicate with a MySQL database, such as ASP or ColdFusion. The MySQL server and official libraries are mostly implemented in ANSI C/ANSI C++.

Concerning usability and management tools for MySQL databases it is possible to use the included command-line tool (commands: mysql and mysqladmin). Also downloadable from the MySQL site are GUI administration tools: MySQL Administrator and MySQL Query Browser. Both of the GUI tools are now included in one package called tools/5.0.html MySQL GUI Tools. The user and the developers' community around MySQL is important and this is noticeable with the large variety of other tools available in addition to the above-mentioned developed by MySQL AB. As for instance there are several other commercial and non-commercial tools available. Examples include Navicat Free Lite Edition or SQLyog

Community Edition, they are free desktop based GUI tools and phpMyAdmin, a free Web-based administration interface implemented in PHP.

Comparison, MySQL vs. PostgreSQL

PostgreSQL and MySQL are probably the most important competitors in the open source world of database management systems, and there is a large list of articles, literature and websites related to their comparisons and benchmarking.

In the IT world PostgreSQL is the University based project and consequently is seen as the innovative and research oriented database, while MySQL is the typical open source project result of a commercial marketing strategy.

First thing that influenced our decision is that surprisingly even if MySQL has a much bigger market share than PostgreSQL, it is easier to find more PostgreSQL supporters than MySQL ones. There is a long list of websites with references of experts supporting PostgreSQL choice versus MySQL, as for instance below:

- http://www.vitavoom.com/postgresql.html
- http://article.gmane.org/gmane.comp.lang.ruby.rails/12576
- http://www.sitepoint.com/article/site-mysql-postgresql-1
- http://feedlounge.com/blog/2005/11/20/switched-to-postgresql/
- http://www.postgresrocks.com/

The main difference behind the philosophy and the approach to market of the two database, that confirms what previously said, is clearly that MySQL is more an open source product and PostgreSQL is more an open source project. This is maybe confirmed by the new market related developments involving MySQL AB (the company owning MySQL) which was founded in 1995 and which was recently acquired by Sun Microsystems in 2008 (the owner of Java).

Being MySQL and PostgreSQL the two main competitors in the open source market, a choice between one of the two is a decision that many must make when approaching open-source relational databases management systems. Both are time-proven solutions that compete strongly with proprietary database software. MySQL has long been perceived as the fastest of the two but with less features above all when considering innovative features, while PostgreSQL was assumed to be a more complete featured often even described as an open-source version of Oracle. In 2003, Ian Gilfillan (2003) affirms that the first sticking different between the two databases was the slower performances of PostgreSQL, being this justified with the fact that PostgreSQL was more focused on features than performances. MySQL has been popular among various software projects because of its speed and ease of use, while PostgreSQL has been preferred by developers who come from an Oracle or SQL Server background.

Over the last years the two developers' communities worked hard to make those perceptions change (i.e. PostgreSQL less solid and MySQL less performant) making become those assumptions mostly outdated and incorrect.

MySQL has come a long way in adding advanced functionality while PostgreSQL dramatically improved its speed with the last major releases. Many analysts, however, are unaware of the convergence and still base their opinions on stereotypes based on MySQL 4.1 and PostgreSQL 7.4. In fact, as affirmed by Rindal (2007), both MySQL and PostgreSQL offer a good engine geared towards the lower to middle end of database systems. PostgreSQL has a full list of advanced features that have been stable for longer. As such, it is better suited for larger database systems. Over the long run, we expect MySQL to strongly stake a claim in the higher-end of the scale. The latest release has implemented a host of advanced features to support industry-standard database systems.

Also, the speed comparisons are often referred to tests mainly based on MyISAM and PostgreSQL engines. If the comparison was done between the latest versions of InnoDB and PostgreSQL, PostgreSQL would be often faster.

5.6 The hydro geo-database

Following our design principles, the information system implements an open source relational database, with the spatial extension OGC compliant (geo-database). The geo-database will integrate the water basin data model, following the object to relational database mapping techniques. We named hydro geo-database the geo-database implementing the water basin data model.

The hydro geo-database has been designed with a data model taken from the Arc-hydro data model of ESRI (Maidment, 2002) and organised in different subsystem: water quantity drainage, channel, hydrography, network, time series, basin DTM. Every subsystem is able to deal with water quality and water quantity parameters. The data model has been designed but it has not been totally implemented in the prototype. In figure 5.3, the complete overview of the Arc-hydro data model, all the features are shown.

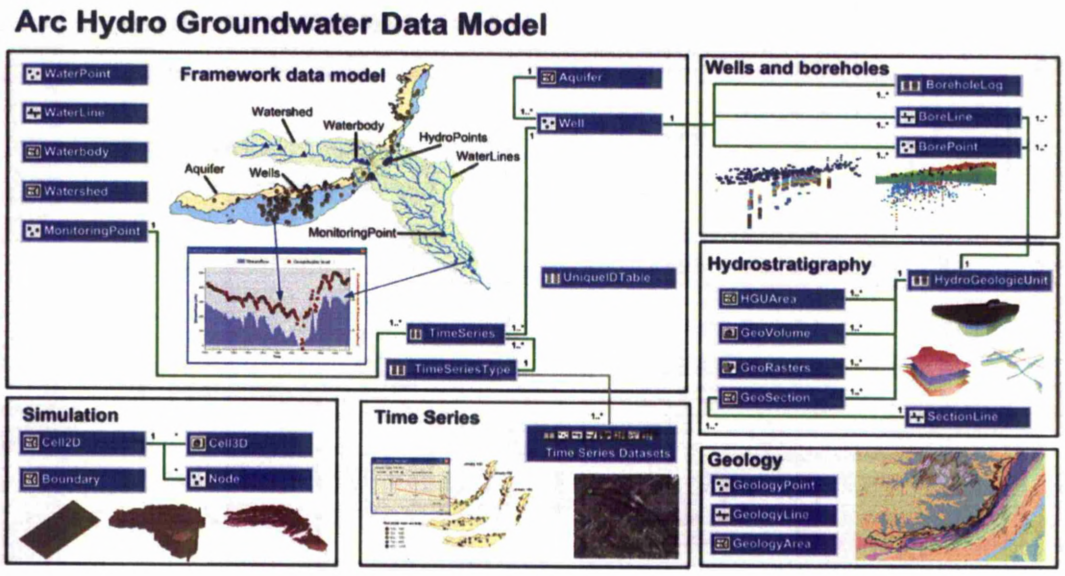


Figure 5.3: an overview of the Arc-hydro data model (Maidment, 2002)

The general features of the hydro geo-database are:

- A general geo-database built conforming to the water basin data model, using a unique ID (Maidment, 2002);
- A structured representation of the knowledge about the basin;
- Knowledge base for every kind of component (simulation model, GIS, DSS, management tool, etc.) integrated in the system;
- Knowledge base for every kind of external component dealing with the system;

The water basin is modelled through a set of generic objects, the classes (in object oriented model), which are represented as tables in the database. In the tables a set of properties are defined as columns and every time that a row is created it is like a new object of the class that is instantiated. The relationships between classes are mapped for the moment using the primary and foreign key.

5.7 Analysis of GIS projects

The GIS is at the same time an end user visualization layer and a data process package able to deal with geographic data. It shall be managed directly from its customized GUI and shall retrieve the information from the geo-database through the standard interface, the JDBC and the GIS API.
According to the design principles of our information system, the GIS tool will be open source, OGC specifications compliant and Java based. In recent years, the GIS industry has witnessed dramatic growth in the development and adoption of open source technologies. The technical GIS community adopted open source technology relatively early, and now mainstream GIS and broader IT industries have come on board as open source products have matured. Therefore there is an important amount of open source GIS projects available and they are extremely powerful and advanced. In order to analyse this large, evolving and dynamic list, open source GIS software can be organized in different non exhaustive categories, depending on the technology used for its development and also on the objective of the development itself (WEB oriented, etc.).

As in every field of open source software, among different categories, developers re-integrate continuously different projects. This happens also because some of these projects are themselves meant to develop libraries oriented to different and particular field of GIS.
The criteria that have been used to proceed with the analysis, benchmarking and selection of the GIS project to be integrated in the system are the following:

- System requirements,
- easy integration with the Information System,
- capacity, efficiency and reliability,
- built-in applications,
- interoperability,
- ease and effectiveness of use,
- speed and demand for computing resources,
- costs and need of resources for utilization,
- possible free and/or not free technical support and community of developers.

Concerning sources of Information for open sources GIS, since the speed, the variety, the quantity and the complexity of projects that are all the time launched or abandoned, the only effective way to find updated information is to collect them from all possible sources on the internet. In particular, there is a quite long series of internet repository which monitor continuously the world of GFOSS (Geographic Free and Open Source Software). Below a list of several of these very useful repositories (Chen et al., 2010).

Specialised GIS Web portals:

- Open Source GIS (http://opensourcegis.org/). This portal for GIS aims to build and maintain a large list of Open Source/Free GIS related software projects. In order to be able to include a long list of projects, the definition of GIS itself is quite large and means to encompass all kind of software or package which deals with spatial data. This site is also based on other open source related repository projects such as OSRS , FreeGIS.org, Metalab Linux Archive, and Fresh Meat.net. A list of over 250 GIS related software packages is provided.

- Freegis (http://freegis.org/). This is another repository with the particular interest that software, Geodata projects, and all related information are organised according to the OS and to the language features. A list of over 300 relevant software is provided.
- OSGeo (http://www.osgeo.org/). This is a website of the Open Source Geospatial Foundation that has been created to support and build the highest-quality open source geospatial software. The main mission of the foundation and of the website in particular is to support and promote the collaborative development of open geospatial technologies and data. It also serves a source code repository which acts as a centralized location for software developers to which community members can contribute with code, funding and other resources, securing in the knowledge that their contributions will be maintained for public benefit. A list with several popular software packages is given.
- Cascadoss (http://www.cascadoss.eu/en/index.php) is a project funded by the 6th EU framework programme for research, and it carried out a very useful benchmarking among a list of open source GIS. This benchmarking was based on general information, marketing potential, technical potential and economic potential. This more than just giving a list of open source GIS, it sets out a series of parameters and it proposes a method to analysis GIS potential and performances.
- GISwiki (http://en.giswiki.net/wiki/Category:Software) brought to the GIS world the concept of WIKI, meaning a collection of Web pages designed to enable anyone who accesses it to contribute or modify content. The main point of having a WIKI for GIS projects is to create a collaborative website empowering a community. GISwiki lists both commercial and open source software.

General repositories of software projects (mainly open source):

- SourceForge (http://web.sourceforge.com/). This, more than just being a repository for all sorts of open source software, is a source code repository which acts as a centralized location for software developers to control and manage open source software development. It helps to maintain and provide codes for downloads. SourceForge.net competes with other providers such as RubyForge, Tigris.org, BountySource, BerliOS, JavaForge and GNU Savannah. Since it is a general repository, in order to determine the number of GIS related projects a search has been carried out in the internal search engine with the GIS keyword, and a list of 240 projects is given (April 2009).

- Wikipedia, the most well known WIKI (as previously defined) provides also a list of GIS software packages (http://en.wikipedia.org/wiki/List_of_GIS_software) and comparisons of some packages were given at (http://en.wikipedia.org/wiki/Comparison_of_GIS_software). It is noted that both commercial and open source software have been included.

After an extensive search among the above mentioned references, more than 300 hundreds projects relevant to open source GIS were found. The initial selection was based on macroscopic criteria such as the fact that the project behind the software was currently active, meaning that a new version or at least an update occurred in the last year and that the project was mature enough, meaning working for at least 3 years with a series of updates and new versions.

The second criteria was the objective of the project, as for instance general purpose Desktop GIS applications, web-GIS packages, functions packages which need more development etc. A list of 31 open sources GIS software packages has been put together as a serious candidates to be considered for the project. The name, version of release, developer and homepage are shown in the following table (Chen et al., 2010).

Name	**Developer**	**Homepage**
Apache Batik	Apache	http://xmlgraphics.apache.org/batik/
DIVA GIS	CIP (International Potato Center, Peru)	http://research.cip.cgiar.org/confluence/display /divagis/Home
Deegree	lat/lon, Germany	http://www.deegree.org/
Fmaps	Fmaps team	http://sourceforge.net/projects/fmaps/

FWTools	Private	http://fwtools.maptools.org/
GeOxygene	GeOxygene Team	http://oxygene-project.sourceforge.net/
GeoServer	Geoserver team	http://geoserver.org/display/GEOS/Welcome
Generic Mapping Tools	Univ of Hawaii	http://gmt.soest.hawaii.edu/
GRASS GIS	GRASS Development Team	http://grass.itc.it/
gvSIG	Iver, Generalitat Valencia, Universidad Jaume I	http://www.gvsig.gva.es
HidroSIG	University of Colombia, Medellin	http://cancerbero.unalmed.edu.co/~hidrosig /ingles/index.php
ILWIS	52° North	52°North Product page
KOSMO	SAIG S.L.	http://www.opengis.es/
JTS Topology Suite	VIVID SOLUTIONS	http://www.vividsolutions.com/jts/jtshome.htm
Mapnik	BerliOS Developer	http://mapnik.org/
MapWindow GIS	MapWindow Open Source Team	http://www.mapwindow.org
mezoGIS	Private, frozen now	http://www.mezogis.org/
monoGIS	MonGIS team	http://www.monogis.org/
NRDB	Private	http://www.nrdb.co.uk/
OpenJUMP	OpenJUMP Team	www.openjump.org
OpenMap	BBN Technologies	http://openmap.bbn.com/
OSSIM	OSSIM team	http://www.ossim.org/OSSIM/OSSIMHome.html
PostGIS	Refractions Research	http://postgis.refractions.net/
Quantum GIS	QGIS Development Team	http://www.qgis.org
SAGA	Univ of Goettingen	http://www.saga-gis.uni-goettingen.de/
SAMT	Institute of Landscape Systems Analysis (ZALF)	http://www.zalf.de/home_samt-lsa/
SavGIS	IRD (Development Research French Institute)	http://www.savgis.org
SharpMap	sharpmap team	http://www.codeplex.com/SharpMap
TerraView	DPI of INPE	http://www.dpi.inpe.br/terraview/index.php
Thuban	Thuban Team	http://thuban.intevation.org/
uDIg	Refractions Research	http://udig.refractions.net

In order to refine this long list another very interesting research was taken into account. This was carried out by one of the market leader in open source GIS, Refractions (http://www.refractions.net/). The research tries to define a method for analysis of open source

GIS projects and software. They pointed out a series of reasons for an open source GIS to become successful and to be a good candidate to be integrated in an information system (Ramsey, 2007). It is interesting to note that criteria are particular to open source and would somehow differ from a commercial GIS software development, at list in their prioritisation. They are formulated as questions that an analyst should wonder in order to make a reasonable choice, these questions are:

- Is the project well documented?
- Is the development team transparent?
- Is the software modular?
- How wide is the development community?
- How wide is the user community?

These can be also seen as an indirect criteria which more than analyse the project itself address in depth its sustainability, success and user friendliness. It is a test of the state of healthiness of the project.
In the same direction, again very relevant to open source, goes the interesting approach that Ramsey (2007) undertakes in order to categorise projects taking into account the open source community working around the project, calling it tribe with the reference to the programming language/technology used. On the basis of this "tribe" approach, he also makes a classification of projects:

- The 'C' tribe, mainly consisting of developers working on UMN Mapserver, GRASS, GDAL/OGR, OSSIM, Proj4, GEOS, PostGIS, QGIS and MapGuide OS. The 'C' tribe also includes users of scripting languages that bind easily to C libraries, such as Python, Perl and PHP.
- The 'Java' tribe, consisting of developers working on GeoTools, uDig, GeoServer, JTS, JUMP, OpenMap and DeeGree.

- The '.Net' tribe, consisting of developers working on Worldwind, SharpMap, NTS, and MapWindow.

There are a series of these above mentioned projects that are used transversally by some of the other mentioned communities or at least by more than one. This is for instance the case of the geodatabase PostGIS/PostgreSQL thanks also to other interfaces like libpq (C/C++), ODBC, NPgSQL (.Net) and JDBC (Java). However, since it is developed in C and uses C-based GIS support libraries, PostGIS should be treated as member of the C tribe.
It is also important to note that Mapserver is also used by some Java developers via JNI (Java Native Interface) or via the OpenGIS WMS and WFS protocols. C and Java communities developed a high degree of linkage in particular with code reuse and libraries linking. There is also a mechanism for .Net to reuse both the Java and C code called the .Net "assemblies" allowing .NET to wrap C/C++ code and cross compiling of Java code using J#. The J# (J-sharp) programming language is a transitional language for Java and Microsoft's Visual J++, to use code, and applications on .NET platform. J# can work with Java bytecode and source so it can be used to transition applications that use third party libraries even if their original source is unavailable.
Another category of projects that cannot be organised around the language communities are the web applications. This category includes various toolkits and web services that provide a browser-based interface to spatial web services, like mapping servers.
In the shorter list of projects, considering now just the JAVA projects, the programming language chosen for our information system, it is possible to affirm that the "Java" world includes several independent attempts at "complete unified toolkits" – OpenMap, GeoTools, and Deegree. Besides, two of the desktop applications considered, JUMP and gvSIG, have independent implementations of core features like data access and rendering, though they do use GeoTools library code for some functionality.
Therefore considering our research there is a series of projects which would have been suited. The choice of the GIS project to be integrated, at the time of the development, was made based on the conceptual belief that the most important general requirement for the information

system, integrability, extensibility, can be better attained if the system focuses at the maximum possible level modularity of its components.

Hence, more than general performances and features of the short list of projects above cited, the focus went to modularity. Among those projects the only one making use of a particular concept related to modularity is OpenMap. OpenMap uses the JavaBeans technology (www.java.sun.com), which by definition are reusable software programs that can be developed and assembled easily to create sophisticated applications. JavaBeans technology is based on the JavaBeans specifications. In the next paragraph a thorough analysis of OpenMap and JavaBeans concepts and specifications is presented.

5.8 GIS implementation: Open Map and JavaBeans

Based on our previous analysis, the project which resulted as the most suitable for our information system is OpenMap. BBN Technologies' OpenMap package is JavaBeans based programming toolkit. OpenMap is completely open source and has a Mozilla-style source license called Mozzilla Public Licence (MPL), an open source/free software copyright license (http://www.mozilla.org/MPL/). If talking about modularity as one of the most important requirements for the information system, the concept put forward by JavaBeans (http://java.sun.com/javase/technologies/desktop/javabeans/) is extremely attractive. By definition, JavaBeans are reusable Java software components that can be manipulated visually in a builder tool. SUN claims that they are reusable software programs that can be developed and assembled easily to create sophisticated applications. The fact that OpenMap structure is based on JavaBeans has a direct impact on its development features.

In the case of organisations related to managing water resources (utilities, basin agencies, etc.), often, experts are hydroinformatics specialists rather than pure IT specialists. Therefore, water engineers have usually limited knowledge of programming. This is why implementing information systems with decreasing need for pure programming skills could be very useful features for those organisations.

According to its developer, using OpenMap, it is possible to quickly build applications and applets that can access data from legacy databases and applications in a distributed setting. OpenMap provides the means to allow users to visualise and manipulate geospatial information (openmap.bbn.com).
At its core, OpenMap is a set of swing components that understand geographic coordinates. A Swing is a widget toolkit for Java and part of Java Foundation Classes (JFC), in other words they are API for providing graphic. These Swing components help to show map data, and help to handle user input events to manipulate that data.
OpenMap has been used by BBN in a number of DARPA and military sponsored programs including: ALP (Advanced Logistics Program), Ultra*Log, AMP (Analysis of Mobility Platform), CoABS (Control of Agent Based Systems), GAMAT (Global Air Mobility Advanced Technology), JLACTD (Joint Logistics Advanced Concepts Technology Demonstration) and SensIT (from the website http://openmap.bbn.com/).
BBN has been a member of the OGC (Open GIS Consortium) since 1995. Through this membership BBN has been involved in adopting and developing global standards for sharing geospatial data. BBN worked also with a number of commercial vendors in the GIS market (Bentley Systems, ESRI, Intergraph, Oracle Corporation, MIT and UMD).
The OpenMap architecture (figure 5.4) is a two-tier client/server architecture and therefore easy to integrate in our information system.

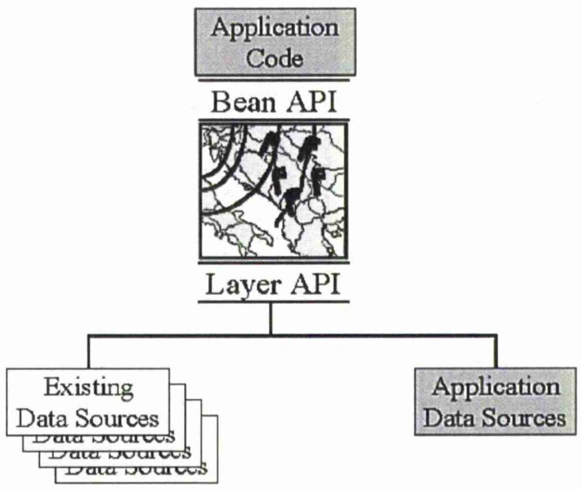


Figure 5.4: the OpenMap architecture (openmap.bbn.com).

The limitation of the OpenMap package against its use in hydroinformatics and in water resources management is that it does not come with already developed water resources management related functionalities (hydroinformatics packages). This is not relevant tor our research since one of our main objective is to study and to demonstrate the process of integration of different components of an information system and OpenMap is a very good starting point to develop the visual managing tools. Referring to the overall system structure, the integration of OpenMap has been straightforward since it covered the higher level, being at the centre of the GUI development.

Database Connectivity

The most important and first task to be carried out is to integrate the GIS into the information system and this is achieved by connecting GIS with the data management level. Therefore, first part of the development was focused on database connectivity. We developed the application program interface (API) able to connect OpenMap to the RDBMS through the JDBC. In order to have a more complete picture on the connectivity features of OpenMap and to demonstrate that it could also work with different interchangeable RDBMS, we built three APIs for two of the most important open source existing RDBMS, MySQL^{TM}, PostgreeSQL plus the commercial market leader, ORACLE^{TM}. Since the JDBC are standard APIs and the JDBC available for the three RDBMS are quite similar, also our three OpenMap API layers are quite similar. In figure 5.5, the structure of the interface between OpenMap and the above mentioned three databases are shown. The interface is constituted by two different levels which can exchange information directly. Level one is developed during this research, called OpenMap API, allows OpenMap to exchange information with the JDBC, the level two is available from the open source community i.e. JDBC API which transfers the information flow from the OpenMap API to the RDBMS, translating Java commands into SQL.

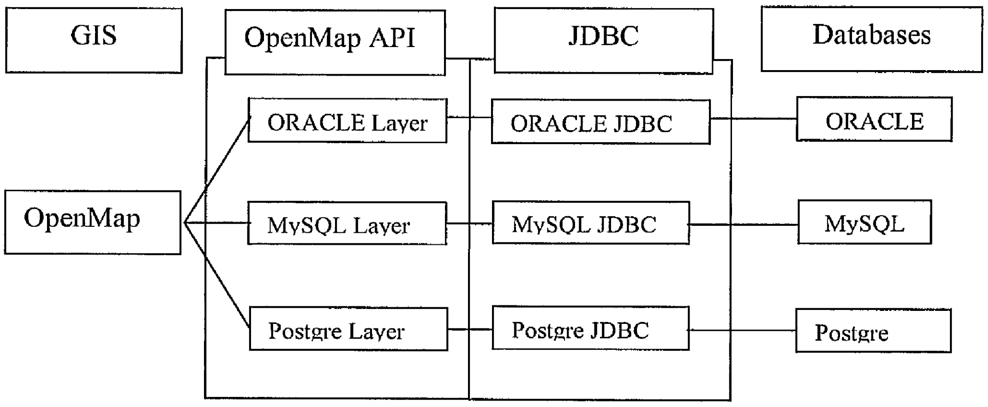


Figure 5.5: Database connectivity.

This phase involved the creation of ad hoc Java classes and APIs to visualise and then to interact with the system knowledge structured (the data model) in the geo-database. The package includes methods to establish the interaction between the mouse and the GIS, to have the possibility to trigger directly from the Map the required queries to the Database.

The development of the API was done programming a series of Java classes to be included in the same package. These classes implement methods which are called from the OpenMap GUI and that can effectuate two different kinds of operations:

- retrieve and represent, modify and store different possible types of geometry compatible with the databases (points, lines, etc.) as defined in the OGC specifications.

- Enquire the database to perform a series of operations (as defined in the OGC specifications) and then retrieve and demonstrate the results.

The step following the development of database connections was the development of a series of customized tools in the GIS packages useful to deal with water basin management.

5.9 The Graphical User Interface

The graphical user interface is constituted of a series of component developed in Java to work in the client side of the IS architecture. Being a Java portable application, the GUI is able to work with many different operative system and hardware platforms. This is more a conceptual affirmation than a concrete and proved feature, in fact for every different hardware and Operative System, the information system would need long testing and debugging before becoming operational. This is true not only for the GUI but also for every other component of the IS. Being quite simple (without considering the testing phase) to transform a Java tool in a Java applet, it will be also possible to make the application working over an Internet browser.

The GUI can be considered as an extension of the GIS. It completes OpenMap with a series of modules allowing typical other automatic functions like presenting data in Charts, etc. The GUI is directly linked to the Hydrogeo-database through the JDBC. Even though the data exchange is carried out just with the DB, the GUI manages the other components of the system with direct calls to their methods. While the data exchange paths respects the process control principles we set up, as in a typical OO software the methods' calls made in a component go directly to the other components of the IS entitled to perform the action. With the further improvement of the client/server architecture, calls should be addressed through the implementation of the RMI. Modularity has been fundamental in the development of the IS, more than just OpenMap GIS other projects has been integrated in order to have a more complete GUI.

In particular, to develop all the Java classes needed in the GUI for data analysis representation we used among other open source projects, Jfreechart (www.jfree.org), which is a powerful free Java class library for generating charts. This gave to the GUI a series of functionalities comparable to a basic spreadsheet processor (such as a simplified version of Microsoft Excel). The GUI structure integrating the different projects mentioned can be represented as in figure 5.6.

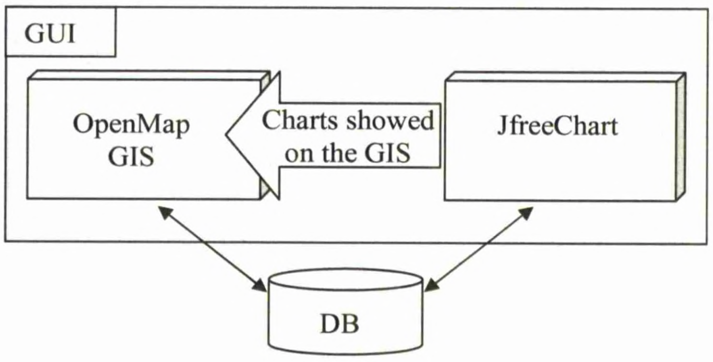


Figure 5.6: the structure of the GUI showing the open source projects integrated.

JFreeChart is an open source Java chart library allowing developers to display professional quality charts in their applications. JFreeChart provides the system and the GUI a set of useful features. Its most important features are:

- It supports a wide range of chart types;
- Its design is quite flexible allowing easy extensibility;
- It can be used both in server-side and client-side applications;
- It supports many output types, including Swing components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG);
- It is an open source project distributed under the terms of the GNU Lesser General Public Licence (LGPL), which allows its use also in proprietary applications.

In particular it is interesting to notice that Jfreechart supports a series of useful chart types:

- X-Y charts (line, spline and scatter),
- Pie charts,

- Gantt charts,
- Bar charts (horizontal and vertical, stacked and independent),
- Single valued (thermometer, compass, speedometer).

As for the other open source projects integrated in the information system one of the most important criteria is the developers' community. The JFreeChart project was founded in February 2000 and nowadays it is used by approximately by 50,000 developers which represent an extensive user community.

One other interesting parallel project, extension of JFreeChart, is JWebChart (http://jwebchart.sourceforge.net/), a chart servlet that renders JFreeChart generated charts using URL.

We can conclude that the GUI has been built with major contributions and extensions from two JAVA open source projects, OpenMap and JFreeChart (figures 5.8, 5.9). These two projects are completely different and have different objectives but one is completing the other and their integration was facilitated by their common programming language, JAVA, and their openness. The contribution from the Open Source community for the GUI was substantial, and it allowed us to add many powerful features without having the burden to focus on heavy programming. Modularity and openness have been once again fundamental in the development of this component of the IS.

5.10 Implementation of simulation models

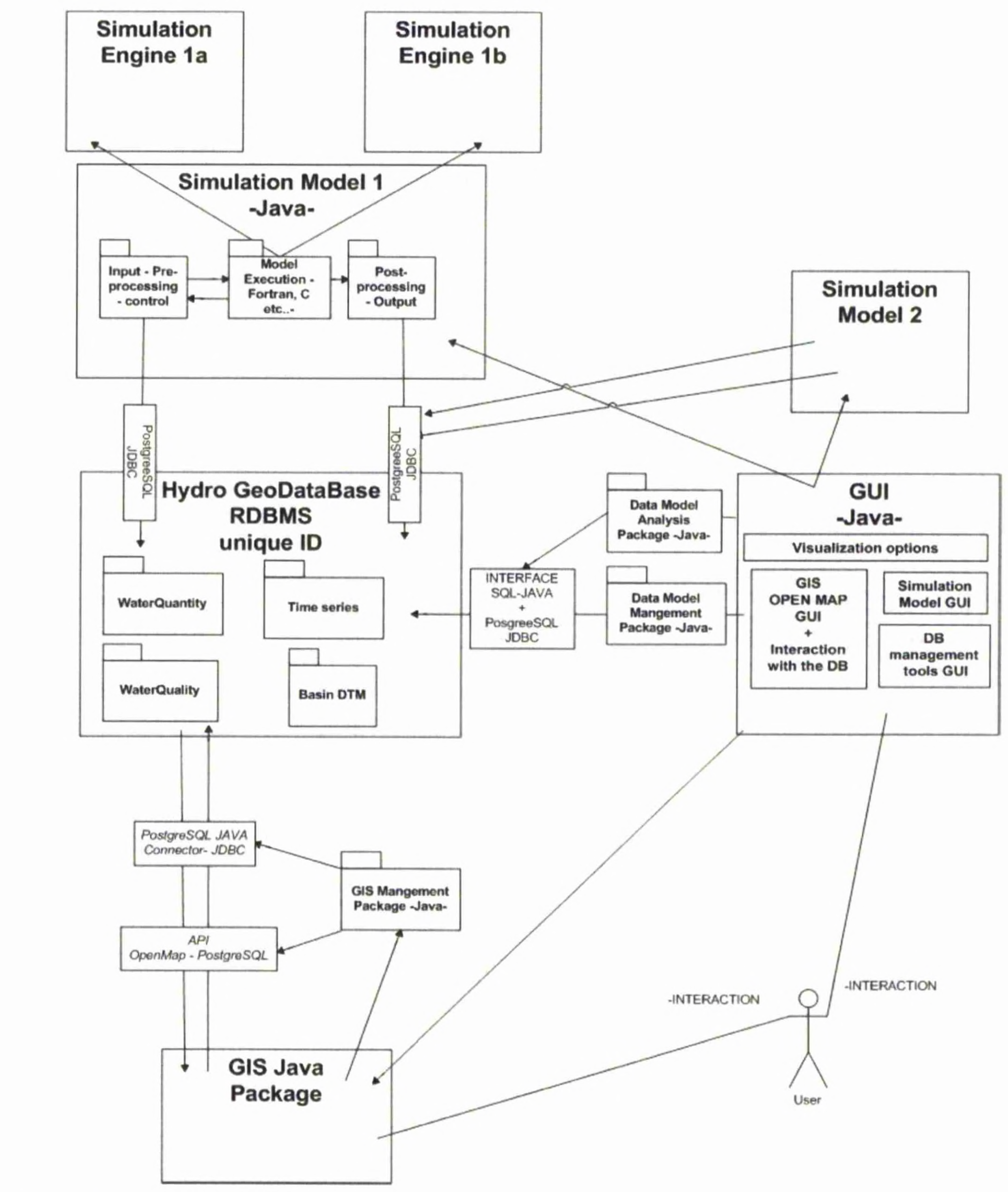
Some basic hydrological simulation models have been developed in Java to test information system functionalities and its flexibility. Their development followed the IS architecture design and the case study performed by Spanou (2001) "Water quality modelling of the upper river Mersey system using an object oriented framework" continuing the hydroinformatics research work of Professor Chen research team (Chen, 2002, Leone et al. 2006, etc.). Currently two of the tools (regarding flow analysis at gauged river sites) of Spanou work have been integrated in the system.

Moreover some other general packages (statistical, mathematical, etc.) needed as basic support component of the IS have been developed at the business layer level. These can be useful for supporting basic functionalities of the IS and of other simulation models to be implemented in the IS, allowing the developers to continue the research testing extensibility and integrability of the system architecture.

5.11 Information System architecture

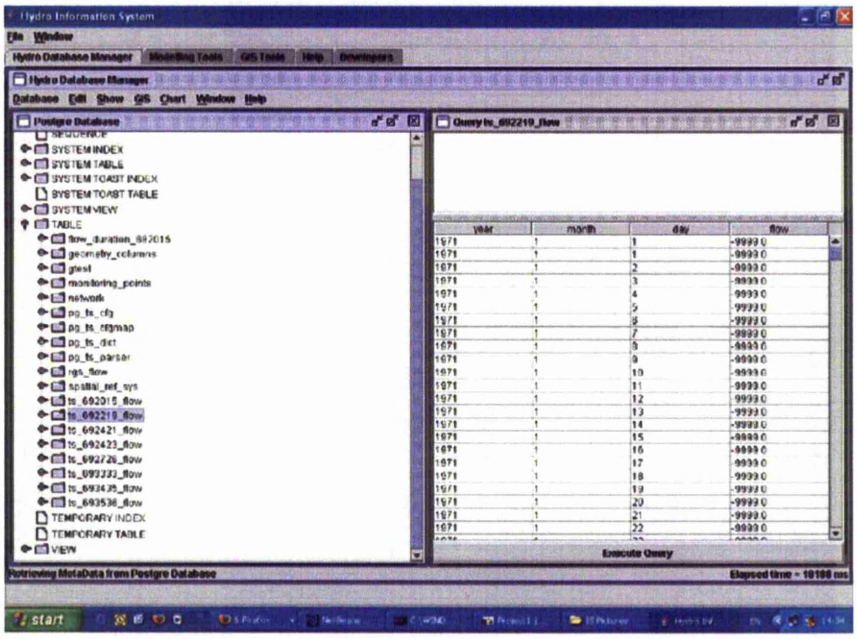
Finally the whole architecture of the Information System following the n-Tier model, the software requirements and the integrated river basin management principles is shown below (figure 5.7).
Summarizing the most important open source projects/technologies we integrated in the IS are the following:

- The core programming language is Java;
- the query language is the SQL;
- the spatial extensions for the GIS and for the geodatabase are based on the OGC specifications,
- the database connectivity is based on the JDBC driver;
- the software used to develop the geo-database and the data model is PostgreSQL with PostGIS as spatial extension;
- for the visualisation layer the OpenMap GIS project has been used as baseline;
- for standard (not directly related to hydraulics) data analysis and visualisation most important project implemented in the information system GUI was JFreechart.



Simulation Engine 1a
Simulation Engine 1b
Simulation Model 1 -Java-
Input - Pre-processing - control
Model Execution - Fortran, C etc..-
Post-processing - Output
Simulation Model 2
PostgreSQL JDBC
PostgreeSQL JDBC
Hydro GeoDataBase RDBMS unique ID
WaterQuantity
Time series
WaterQuality
Basin DTM
Data Model Analysis Package -Java-
INTERFACE SQL-JAVA + PosgreeSQL JDBC
Data Model Mangement Package -Java-
GUI -Java-
Visualization options
GIS OPEN MAP GUI + Interaction with the DB
Simulation Model GUI
DB management tools GUI
PostgreSQL JAVA Connector- JDBC
GIS Mangement Package -Java-
API OpenMap - PostgreSQL
GIS Java Package
-INTERACTION
-INTERACTION
User

Figure 5.7: the IS architecture.



year	month	day	flow
1971	1	1	-9999.0
1971	1	1	-9999.0
1971	1	2	-9999.0
1971	1	3	-9999.0
1971	1	4	-9999.0
1971	1	5	-9999.0
1971	1	6	-9999.0
1971	1	7	-9999.0
1971	1	8	-9999.0
1971	1	9	-9999.0
1971	1	10	-9999.0
1971	1	11	-9999.0
1971	1	12	-9999.0
1971	1	13	-9999.0
1971	1	14	-9999.0
1971	1	15	-9999.0
1971	1	16	-9999.0
1971	1	17	-9999.0
1971	1	18	-9999.0
1971	1	19	-9999.0
1971	1	20	-9999.0
1971	1	21	-9999.0
1971	1	22	-9999.0

Figure 5.8: GUI of IS showing the database manger tool.

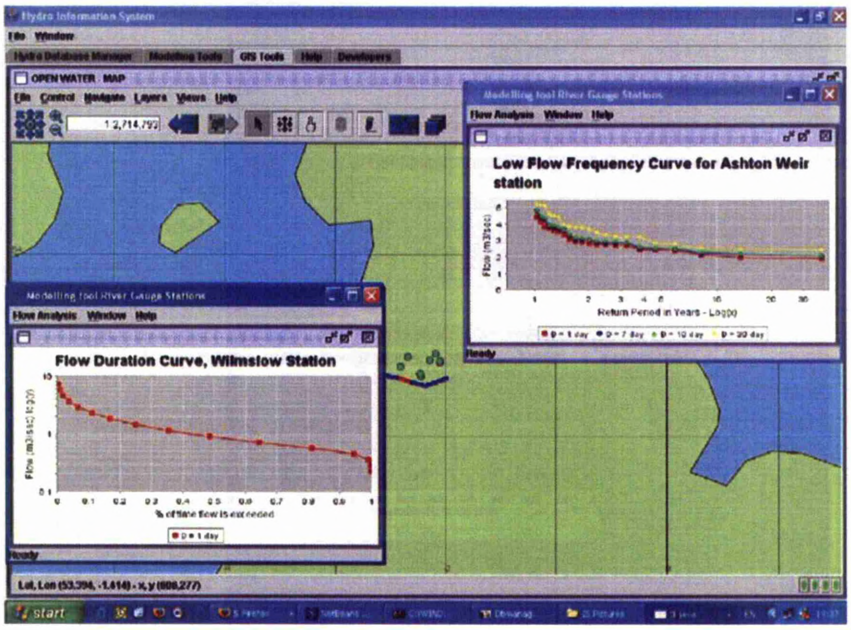

Hydro Information System
File Window
Hydro Database Manager
Modelling Tools
GIS Tools
Help
Developers
OPEN WATER MAP
File Control Navigate Layers Views Help
1:2,714,793
Modelling Tool River Gauge Stations
Flow Analysis Window Help
Low Flow Frequency Curve for Ashton Weir station
Flow (m3/sec)
Return Period in Years - Log(x)
D = 1 day
D = 7 day
D = 10 day
D = 30 day
Flow Duration Curve, Wilmslow Station
Flow (m3/sec) log(x)
% of time flow is exceeded
D = 1 day
Ready
Lat, Lon (53.394, -1.414) - x, y (808,277)
start

Figure 5.9: GUI of IS showing the GIS and the modelling tools.

CHAPTER VI

6 APPLICATION OF THE OBJECT ORIENTATION TO THE DATA LAYER

6.1 Introduction

It has been shown in the previous sections that the object oriented approach allows to develop IS for river water quality and catchment hydrological modelling integration. The hydro information systems (HIS) developed, is an OO based system with extensions of OpenGIS standards and connections to various relational database management systems (RDBMS). Such HIS integrates a series of subsystems: GIS, Simulation and Management Models, Databases and other external data sources, and a GUI. These different components of hydro information systems have been developed with the use of OO logic and technology except for the data layer, where RDBMS are still the preferred solution in terms of the reliability of performances.

These integrated tools allow users to analyse real-time environmental conditions through integrated models, environmental trends and possible development for early warnings (flooding, environmental hazards, etc.). The increasing complexity of environmental data, data structure of data models, amount of information and broadcasting channels (LAN, internet, etc.), needs the power of the OO logic to be managed as already stated in the previous sections of the thesis.

In general, in modern information systems for water resources management data model is designed with object oriented logic and is then integrated in the system as persistent layer through manual Object-Relational (O-R) mapping with RDBMS. This path has been followed (as illustrated in previous chapters) in the first implementation of our information system. The main problem we encountered is that the O-R mapping is not standardized and that the interface is complicated, also at conceptual level, by the "impedance-mismatch" between the domains object model of the application and the relational model of the RDBMS (Ambler, 2006).

Therefore, the incongruence between the “Relational Logic” of data layers and the “Object Logic” of other components of the HIS need to be solved to increase overall performances. The obvious solution to solve the impedance mismatch in Information System above described is developing a data layer with object oriented logic. In this chapter , the experience of testing two different object oriented solutions for data management will is presented: the Java Data Object technology (JDO, Jordan and Russell, 2003) and an Object Database (ODB) (Leone and Chen, 2007). Our Java based Hydro Information System (Leone et al., 2006) presented in the previous chapter is used as test bed. It is noteworthy to re-affirm that this IS has been developed using only open source technologies and software and its evolution will follow the same path. Technology solutions and implementation choices aim at making our Information System evolve as an alternative to its first and more conventional implementation where the IS was interfaced and "manually mapped" with a RDBMS.

6.2 System evolution towards Object Oriented persistence

The definition of persistence with regards to Java technologies, the core programming language of the IS, and more in general to OO technologies, is the ability of an application or service to shuttle state between a transient objects and some type of data store. In contrast, objects which "die" at the end of a process are called transient.

Persistence, also called object durability, is a term often used in conjunction with the issue of storing objects in databases.

In order to improve the IS data management capacities, the ultimate goal is to have an OO implementation of the logical data model in the data layer, able to make objects persistent, directly managed from the IS.

The tests related to the implementation of two different solutions for OO data management resolving the O-R impedance-mismatch of the system are presented. In practise, two different object oriented data layers are developed with the following functionalities requirements:

- Capacity to manage and store objects;
- Adaptability in the number of objects and storage capacity;
- Working on both sides of client – server architectures;
- Managing complex queries, similar to SQL query capacity level;
- Real-time data management from data sources across the Internet or sensor networks;

- Open Source software based;
- Ability to integrate existing data sources (stored in RDBMS) with data object model.

6.3 Choice of Persistence Standard

As mentioned, persistence is the ability of data to outlive an instance of a program (Bauer and King, 2006). There are several open source technologies available to implement persistence standard in Java based Information Systems. Some these technologies, used to implement OO persistence in general and Java object persistence in particular, such as Hibernate (www.hybernate.com) or JDO (Java Data Object) (Korb, 2004) are designed to provide the developer with transparent persistence; the application deals with persistent objects without the need for SQL to be embedded in the Java code. These solutions are called transparent because the developer does not directly deal (see) the implementation of the data layer.
Another solution like Container Managed Persistence (CMP) has similar performances for EJB (Enterprise Java Beans) containers, but it is not a general persistence facility for the Java platform. In all these technologies, the objects are "automatically" mapped to tables of a RDBMS by the underlying framework. The developer does not have to deal with time and performing consuming RDBMS development tasks but he has to perform time consuming O-R mapping operations.
To define the mappings, the developer needs to create descriptors, typically XML files. Inheritance and many-to-many relationships augment complexity as these cannot be conceptually managed in the relational model. The more complex the data model, the more difficult the O-R mapping. Purer solutions, conceptually different from O-R mapping, are databases based on the object model. An object database management system (ODBMS), is a DBMS that supports the modelling and creation of data as objects (www.odbms.org).
There is no official Standard for ODB. The de facto standard is the final release of the last Object Database Management Group (ODMG, www.odmg.com), the ODMG 3.0. The ODMG Java binding has been now superseded by JDO.
ODB combine the elements of object orientation and OO programming languages with database capabilities. They extend the functionality of OO programming languages (e.g., C++,

Smalltalk, Java) to provide full-featured database programming capability. The result is a high level of congruence between the data model for the application and the data model of the database, resulting in less code (25% to 35% less), more natural data structures, and better maintainability and reusability (McClure, 1997).

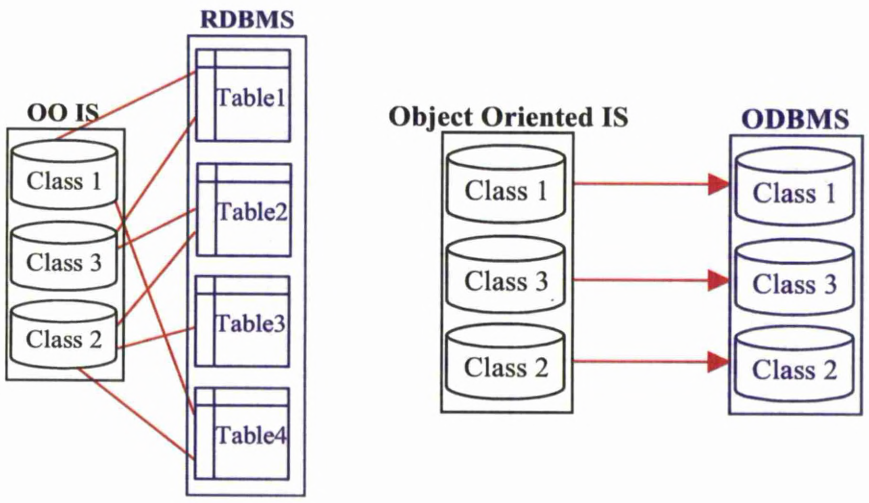


Figure 6.1: a) O-R Mapping; b) Object DB.

In figure 6.1 a), classes of an OO IS have to be mapped onto RDBMS tables; additional tables are needed to represent OO relationships in relational logic. In figure 6.1 b), classes of an OO IS are directly stored in an ODB.

6.4 Implementation and Case Study

Two different kinds of technologies, the transparent persistence and the ODBMS, have been implemented in the IS data model in order to analyse their performances dealing with hydroinformatics data. The data model previously implemented in the IS with relational logic, a customized version of the well known ArcHydro from the Centre for Research in Water Resources of the University Of Texas (Maidment, 2002), also used by ESRI (www.esri.com) for water resources management applications, will be implemented with OO logic. This data model (figure 6.2) is conceived as geographic database, GIS oriented representation of a HIS.

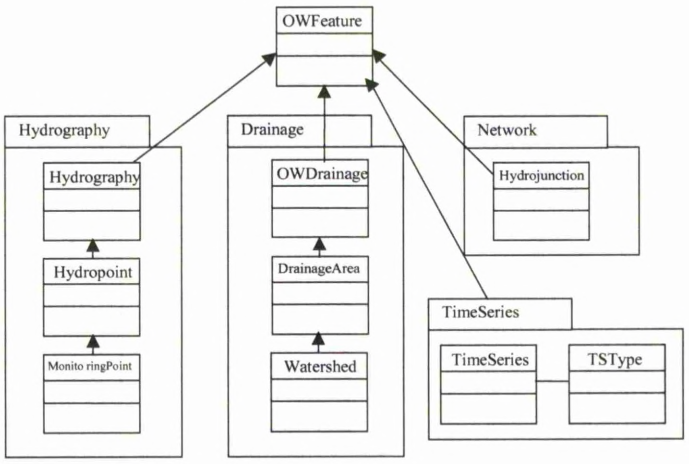


Figure 6.2: Partial representation of the data model.

6.4.1 The transparent persistence solution

The transparent persistence solution chosen is the Java Data Object. JDO is the standard for persistence of Java objects (Jordan and Russell, 2003), which basically provides two important advantages: transparent persistence and runtime database portability across the compliant implementations (www.jdocentral.com).

There are several JDO implementations, most of them Open Source. JPOX (www.jpox.org) for instance is an Open Source and fully compliant implementation of the JDO 1.0 and 2.0 specifications.

In the JDO implementations, information is virtually stored in the data model, but physically goes through the JDO API to be stored in the RDBMS. Neither the IS, the developer nor the user developer have to care about the way the information is managed or the O-R mapping is done by the JDO API.

In general, for every JDO implementation as well as for JPOX, there are 6 different phases:

1. Design the domain/model classes;
2. Define their persistence using Meta-Data;
3. Compile the classes, and instrument them (using a JDO enhancer);
4. Generate the database tables where the classes are to be persisted;
5. Write the code to persist objects within the DAO layer;
6. Configure and run the application.

Every time the data model is updated with new classes or just changed, the steps from 2 to 5 have to be repeated.

In the first step, the data model is normally designed. Then, the developer defines how the classes should be persisted, in terms of which fields are persisted and the way they are persisted etc. This is performed by writing a Meta-Data persistence definition for each class. Here an extract from an XML file of our implementation:

```
<?xml version="1.0"?>
<!DOCTYPE jdo PUBLIC "-//Sun Microsystems, Inc.//DTD Java Data Objects Metadata 1.0//EN"
"http://java.sun.com/dtd/jdo_1_0.dtd">
<jdo>
  <package name="datamodel.hydro">
<package name="Hydrography">
 <class name="hydrography" identity-type="datastore" persistence-capable -
superclass="datamodel.hydro.OWFeature">
    <field name="FType" persistence-modifier="persistent">
       <extension vendor-name="jpox" key="length" value="max 100"/>
     </field>
...............
</class>
 <class name="HydroPoint" identity-type="datastore" persistence-capable-superclass="hydrography">
     <field name="JunctionID" persistence-modifier="persistent">
        <extension vendor-name="jpox" key="length" value="max 100"/>
     </field>
 </class>
  </package>
```

The inheritance relationship is defined (with the "superclass" descriptor) between the class hydrography and the class OWFeature.

The third step makes the classes PersistenceCapable through this Java interface. JPOX and other JDO standard implementations provide their own byte-code enhancer for instrumenting/enhancing classes. Finally the data model is persistent. Interaction with the persistence framework of JDO is performed via a PersistenceManager. This provides methods for storing, querying, removing objects, etc.

Now that the data model is persistent, the IS is ready to be developed dealing with the data layer. The query language to perform all the interactions between the IS and the data model is the JDO Query Language (JDOQL). JDOQL uses Java syntax, keeping SQL heritage, its power and even improving its simplicity as shown in the following examples. In our implementation, considering the "Irwell1" monitoring station of the Irwell river :

Example 1, JDOQL request:

- **JDOQL**

```
MonitoringPoint.class,"flow>Irwell1.flow"
```

- **Equivalent SQL**

```
SELECT MonitoringPoint.*
FROM MonitoringPoint INNER JOIN MonitoringPoint AS Irwell1 ON MonitoringPoint. Irwell1 = Irwell1.ID
WHERE ((MonitoringPoint.flow)>[ Irwell1].[flow]);
```

Example 2, JDOQL request:

- **JDOQL**

```
MonitoringPoint.class,"(flow> Irwell1.flow) && (temp > Irwell1.temp)"
```

- **Equivalent SQL**

```
SELECT MonitoringPoint.*
FROM ((MonitoringPoint INNER JOIN MonitoringPoint AS Irwell1 ON MonitoringPoint.Irwell1= Irwell1.ID)
INNER JOIN Temperature AS TemperatureMonitoringPoint ON MonitoringPoint.temp =
TemperatureMonitoringPoint.ID)
INNER JOIN Temperature AS TemperatureIrwell1 ON Irwell1.temp = TemperatureIrwell1.ID
WHERE (((MonitoringPoint.flow)>[Irwell1].[flow]) AND ((ParametersMonitoringPoint.temp)>[
ParametersIrwell1].[temp]));
```

The above examples allow comparing the two approaches. Going through the code of the examples it is clear the "shock" between the two logics, the OO and the relational logic. Being obliged to manage RDBMS queries from JAVA the amount of code and requests to be done to the system are important. This shows how powerfully can be managed an OO data model through very simply queries even for complex environmental data and how ineffective and heavy can be the use of the relational SQL in the same case.

6.4.2 Object Database solution

The other tested solution to achieve persistence is using an embedded ODB engine in the IS. This is the case of Db4o (Carl Rosenberg, www.db4o.com), available commercially and through Open Source license (GPL). Db4o has been chosen for applications in embedded systems in which zero administration, reliability, and low footprint, high performance, smooth synchronisation and easy refactorability are critical features. There are several implementations of Db4o in natural sciences, like biotechnologies and geotechnologies and for networking, energy and water utilities (www.db4o.com). Db4o allows developers to work with persistent data without the distraction of conflicting data models and without the need to spend time learning to use a tool such as Hibernate or the JPOX JDO or a complex ODBMS.
Other similar open source solution for embedded OO database in JAVA has been developed in the last years, PERST by MCOBJECTS (http://www.mcobject.com/perst). The two projects are at a quite different stage of development, PERST being much younger, with limited performance and applicability.
Besides, other different solutions are available for OO JAVA databases, some of them include "not embedded" databases, others are not pure OO, others are at a very early stage of development.
Considering that the scope of this part of the research was to test an OO solution, and not to perform a complete benchmark exercise of OO JAVA databases, being Dbo4 an acceptable solution, we undertook its implementation.

6.4.3 ODBMS implementation

The implementation process for Db4o and its use are quite simple (Edlich et al., 2006), direct and intuitive. The same data model (Maidment, 2002) already implemented with JDO has been used. The following steps have been followed:

1. Design the domain classes;
2. Define their persistence definition;
3. Configure the application.

Firstly the data model is normally designed as, for instance, in the previous example (JDO implementation). Then, since the objects are directly made persistent and stored, there is no need for writing "Meta-Data persistence descriptor" files.

Db4o includes the S.O.D.A. API (Simple Object Database Access, http://sourceforge.net/projects/sodaquery/) to provide powerful querying.

This is an example of S.O.D.A. API application that shows the power of this ODB managing data with complex structure (as the environmental data):

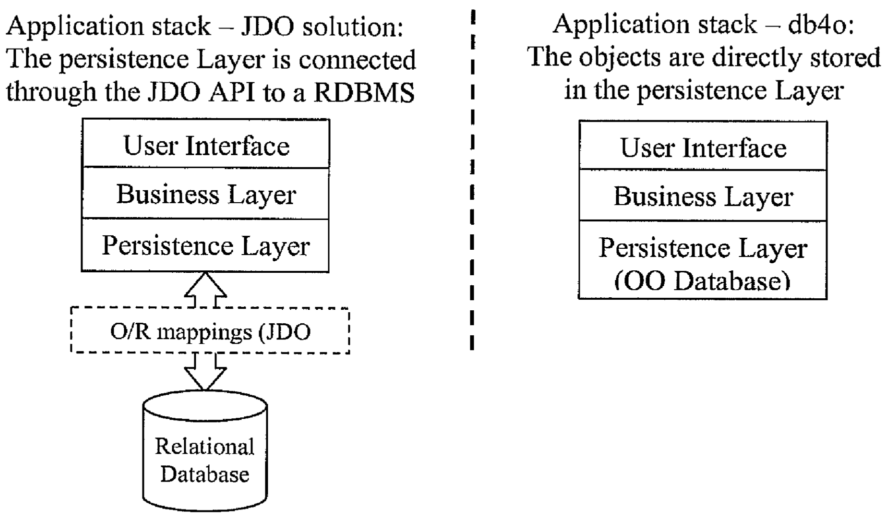
```
Query q = db.query();
q.constrain(HydroPoint.class);
q.descend("flow"). constrain(new Float(0.3f)).greater();
ObjectSet result = q.execute();
```

At first glance, this performs a similar query to an SQL query, like "SELECT * FROM HydroPoint WHERE flow > 0.3

However, the design of the HydroPoint class allows inverse relationships to be created between HydroPoint and Hydrography (parent class) objects. A Hydrography class has a reference to a list of HydroPoint objects, while each HydroPoint has a reference to a

Hydrography class. This means that the result of this query contains HydroPoint and Hydrography objects.

This works because of the inverse relationship designed into the object model. ODB are navigational; data are retrieved following the direction of predefined relationships. Given the right object relationships, related objects can be retrieved from the object database with very little programming effort. This is a fundamental feature for modelling and representing environmental systems being the OO design logic closer to the "real world" (George, 2003) and more adaptable to environmental data structures. Concerning the developer point of view, the simplicity of the installation and use, the lack of impedance mismatch between the objects and the data model make Db4o quick and flexible to be implemented in the IS.



Application stack – JDO solution:
The persistence Layer is connected through the JDO API to a RDBMS
User Interface
Business Layer
Persistence Layer
O/R mappings (JDO
Relational Database
Application stack – db4o:
The objects are directly stored in the persistence Layer
User Interface
Business Layer
Persistence Layer (OO Database)

Figure 6.3: comparison of the implementation stack for the two solutions.

6.5 Quantitative comparison

To compare quantitatively between the two solutions, we used an Open Source benchmark test suite, the PolePosition test (www.polepos.org). This comparison is not meant to give a complete overview of the performance differences between Object Databases and O/R mapping solution, which is not our aim, but to give an example for the data layer of a hydro information system for our case study. Hence, customized tests have been built to adapt and to implement PolePosition in our IS and in particular in our data model. We run two kind of different tests for the two solutions (the JDO and the ODBMS). One test was designed to work with the part of the data model dedicated to time series with simple level of inheritance. Another comparative test was still designed to work with the same "time series" data but this time through the complication of the geo-referenced data of the "monitoring station" class with 4 level of inheritance. The first test has been built just to show the performances with simple level of inheritance but usually all the data requests of our IS are made as in the second test, through several levels of inheritance. The tests run for writing, reading and deleting. Both tests show for the solutions transaction performances over 10,000 objects. From table 1 and 2, we can see that the first test with the flatter kind of data gives alternating results with anyhow not a strong difference between the two solutions. On the other hand the second test shows that resolving the O-R mismatch by using an O-R mapper like JDO costs a lot in terms of performance when dealing with more level of inheritance and with more structured data as the one often used in hydro information systems.

Similar conclusions arise from the standard tests of the benchmark test suite (www.polepos.org). Usually the flatter and simpler the data structures, the better are the performances with relational databases.

t [time in ms]	Write	Read	Delete
Db4o	809	650	680
JDO/MySQL	2885	121	180

Table 6.1: Results of the test 1 "time series" data.

t [time in ms]	Write	Read	Delete
Db4o	1895	950	4250
JDO/MySQL	15340	3225	9265

Table 6.2: Results of the test 2 "monitoring station" data.

This experience suggests that if the data model is complicated in its object structure, as in environmental data structures, and if the quantitative performances are important in the IS to be developed, the O-R mapper solutions will have a negative impact on performances. On the other hand, if the use of O-R mapping technology can be compensated by reducing the hardware, the data model is not complex and the amount of data treated is reduced, O-R mapping technology can bring back performances.

6.6 Discussions and Conclusions

The IS for water catchment management are currently developed with OO technology except for their data layer where the RDBMS are still the preferred choice. An interesting analogy, suggested at the beginning of the work carried out in the sector to include OO logic in data layers of information systems (McClure, 1997), is the following: using RDBMS to store OO data structures would be the requirement of disassembling a car before storing it in a garage each night and then reassembling it before using it each morning. This has a series of clear development productivity, runtime performance and maintenance issues, etc. Since the increasing complexity of the environmental data, the need to represent them through object structures and the O-R impedance mismatch, we tested two solutions to implement OO at data layer level: the transparent persistence of JDO technology and a purer OO solution of an ODB, Db4o. The transparent persistence of JDO is a bridge through its API towards the data layer. Behind the OO data model, implemented in JAVA by the developer, there is a "hidden" RDBMS. The developer does not interact with the RDBMS behind the data model and the JDO APIs are supported by its power, performances and consistency.
Nonetheless it seems that JDO is worthy to be used for an IS which needs strong reliability, which does not have to be upgraded frequently and when the structure of the data and of the data model is quite simple. The ODB which we tested is another effective solution. Db4o provides a flexible, extensible option for applications that must act quickly to match the capabilities of their IS with environmental complex data structures.
The implementation of Db4o in our IS has been quick, intuitive, direct and the Object Oriented developer had not to be familiar with new concepts or procedures. Besides, since

time and effort for the implementation are relatively reduced, the IS is more flexible, easier to upgrade and maintain. We can conclude that even though currently ODB are still rarely used for Hydro Information Systems, our experience has been positive in terms of simplicity, adaptability, extensibility and performance.

CHAPTER VII

7 CONCLUSION AND FURTHER STUDY

7.1 Conclusion

The main reason behind the work done for this research and this thesis was to try to find an approach to the development of information systems for water resources management which could be adequate to:

- the needs of the water sector re-defined in the last 20 years by the integrated water resources management principles then translated in laws such as the EU water framework directive,
- the strong rising of new trends in the IT world, such as the Open Source movement, which seemed to be particularly adapted to scientific communities and integrated management principles,
- consequently, new trends of hydroinformatics switching the focus from "modelling" to "integration".

An interesting work of research on existing relevant projects was carried out. Trends were identified and main technology issues were acknowledged. Consequently, as a pre-requisite for this research, an analysis on possible approaches and technologies brought to define a list of requirements and strategic ideas as baseline for the work. The approach proposed has then been tested with the development of Information Systems for water resources management with pilot applications. Besides, also "new" (new for the field of hydroinformatics) technologies, such as object oriented databases management systems, were studied and implemented in the system. This was interested for two reasons, first of all allowed showing the interest and the relevance of these technologies to hydroinformatics proposing and testing solutions for well-know issues, secondly demonstrated that applying open source technologies allows researchers to be at the edge of technological development also with small means.
We summarise research achievements, analysing the IS advantage / complexities both from the point of view of the developer and of the end user. The analysis has taken into account software architecture issues and technology choices.
The main advantages of the IS are:

- Clear separation of user-interface-control and data presentation from application-logic;
- Change in business logic will not need change in other layers;
- The user interface can be used both as web-based application and as desktop application;
- Being internet based, the IS supports the distributed computing which is also the reason to develop such kind of three layer architecture;
- Easy to manage: if each layer has its own functionality, when something needs to be changed it easy to know what to change;
- Easy to reuse: if another application is developed for the same domain, it can be integrated in one of the system's layer;
- Easy to develop: each layer can be developed by separate teams, and focus only on their specific problems (it not necessary for the developer to know SQL, ORACLE, JAVA, XML, etc. at the same time).

The source of complexity:

- Every data that goes in the system from the user to the database must pass through the components of the middle layers, and therefore the response time of the system can be slower.
- The developing of the interface layer can be a strong source of complexity.
- It is necessary to establish a standard data exchange format in the whole information system, which should be respected in every step between different layers.

Concerning the particular development dedicated to the object oriented database management system, the applications of the objects oriented technologies to relational databases with the implementation of object-relational database management systems showed that integrating different kinds of layers based on the same logic could open opportunities for better integration. A thorough work would be needed to test this kind of approach with large amount

of data as it is needed in hydroinformatics. It is already known that OODMS do not give yet enough guarantees from some points of views such as security, stability, data consistence, etc. Those issues are strongly depending on the development of the market (open source and commercial) in the field.
Besides, the opportunities of further development can be identified can go over the following main points:

- Improvement of the client / server architecture with the distribute computing through the implementation of the Java Remote Method Invocation or other suitable technologies;
- Increase the integration of the hydro geo-database data model with its representation in the GIS visualization layer;
- Improvement of the GIS analysis package which can support all the further modelling developments in the system;
- Testing of the OODMS in real working condition with important amount of data and different data providers at the same time.

Finally we reckon that is it important to point out that the open source community has been a continuous provider of information, inspiration and solutions. It allowed us, even not being experienced programmers, to design and to build an ambitious Information System. Hence, after this experience, we started considering the open source not only an easy way to "pick up" free software solutions, but rather a strategic approach to software development also in hydroinformatics.
A particular mention is needed for the improvement of the client-server architecture. First of all giving the high-computing requirements of hydroinformatics models, the whole business layer must stay in the server side, making a difference between the business layer server and the data layer servers. This will reduce the need of important hardware resources in the client side and would also consistently limit the amount of data communicated through the network, reducing costs and increasing speed.

7.2 Further study

Finally, we would like to give a vision for a next stage of development of modelling integration for water resources oriented information systems in figure 7.1.

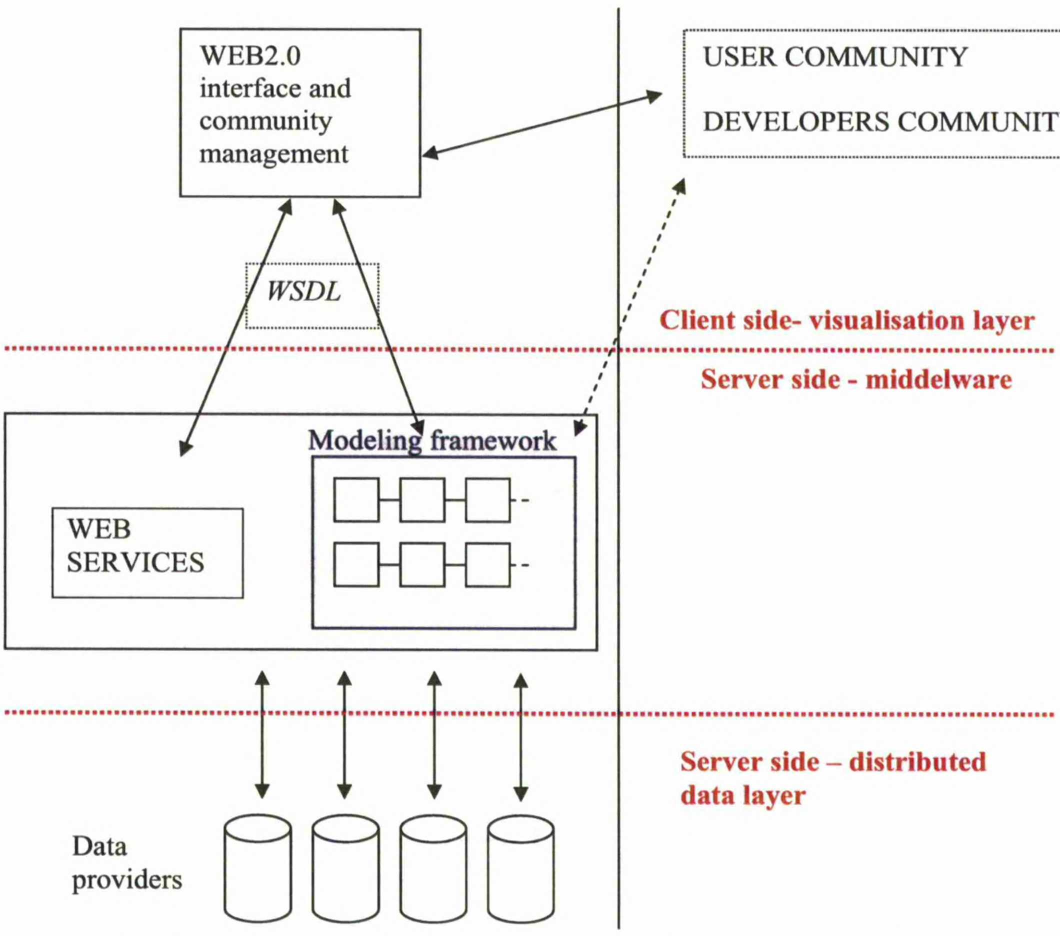

WEB2.0 interface and community management
USER COMMUNITY
DEVELOPERS COMMUNITY
WSDL
Client side- visualisation layer
Server side - middelware
Modeling framework
WEB SERVICES
Server side – distributed data layer
Data providers

Figure 7.1: Information system developed and managed by the community of users.

Many of the concepts shown in figure 7.1 have already been deeply discussed in the previous chapters. Therefore only the new concepts of this "vision", suggestion for further studies, will be listed below:

- The community approach, very much related to the open source software world, is central. The IS can be seen as a support tool for the community of experts and users of the water sector to develop and orient their work,
- consequently, the IS is shaped for and by the community of users-developers. Not only data (such as time series and models results) are shared through the IS but the knowledge which is inside the development of models themselves,
- the modelling framework included in the business layer is a modular Web service infrastructure made by input-engine-output modules which can be combined by users-developers following their needs and objectives and then plugged to data providers,
- all the heavy-computing work is pushed toward the server side,
- providers of data are distributed and chosen according to users' needs,
- WSDL, the Web Service Description Language (http://www.w3.org/TR/wsdl) is used by the users-developers to manage and make communicating Web services including models and other modular Web-services.

This approach for information systems for water resources management built around and by a community of users-developers is coherent with new trends of hydroinformatics and of the IT world combining the following:

- integrated water resources management principles,
- open source software and development approach,
- research and user community orientation,
- the participatory information sharing, interoperability, user-centered design and collaboration principles of the present World Wide Web, known as Web 2.0.

8 REFERENCES

REFERENCES

1 Abbott, Michael B., 1991. Hydroinformatics: Information technology and the aquatic environment. Aldershot (UK), Avebury Technical.

2 Abbott, Michael B., 1999. Introducing hydroinformatics. J. Hydroinformatics. 1(1):3–19.

3 Abbott, Michael B., 2007. Applications of Numerical Modelling in Hydroinformatics. http://www.knowledge-engineering.org/PDF/Paper-112.pdf

4 Ahuja, L.R., David, O and Ascough, J.C., 2004. Developing natural resource models using the object modelling system: Feasibility and challenges. Trans. 2nd Bienn. Meeting Int. Environ. Modelling and Software Soc., iEMSs 2004. Osnabrück, Germany, 14-17 June 2004.

5 Ambler S. W., 2006. Agile Database Techniques: Effective Strategies for the Agile Software Developer, John Wiley & Sons Inc, 480 pp.

6 Ambler, S. W., 2003. The Fundamentals of Mapping Objects to Relational Databases. www.agiledata.org. The Object-Relational Impedance Mismatch. Publisher. www.agiledata.org

7 Andersen, L., 2006. JDBC™ 4.0 Specification. Sun Microsystems, Inc. www.sun.com.

8 Andrews, C., 2004. Java: Why GIS People Should Care. Earth Observation Magazine.http://www.eomonline.com/Common/Archives/2004junlul/04junjul_Andrews.html. GITC America, Inc.

9 Armstrong, D.J., 2006. The Quarks of Object-Oriented Development. Communications of the ACM — Association for Computing Machinery. ACM New York, NY, USA.

10 Barrier, T. (Ed.), 2002. Human Computer Interaction Development and Management. Hershey, PA, USA: Idea Group Publishing.

11 Barth, M., Hennicker, R., Kraus, A., Ludwig, M., 2004. DANUBIA: an integrative simulation system for global change research in the upper Danube basin. Cybernetics and Systems, Volume 35, Numbers 7-8.

12 Bass, L., Clements, P. et al., 2003. Software Architecture in Practice, Second ed. Reading, MA, Addison-Wesley.

13 Bauer, C., King, G., 2006. Java Persistence With Hibernate, Manning Publications, 841 pp.

14 BBN Software, 2004. Open Map. http://openmap.bbn.com

15 Becker, A., Hattermann, F., 2005. Model-supported Participatory Planning for Integrated River Basin Management, Deliverable No. D3/11-13 of Harmoni-CA project (European Commission funded research project), http://www.harmoni-ca.info/.

16 Blind, M., Gregersen, J.B., 2004. Towards an Open Modelling Interface OpenMI: The HarmonIT Project. 2nd Bienn. Meeting of the Int. Environ. Modelling and Software Society, iEMSs. Osnabrück, Germany.

17 Blind, M.W., van Adrichem, B., et al., 2001. Generic Framework Water: An open modelling system for efficient model linking in integrated water management. EuroSim 2001 congress Shaping Future with Simulation, Delft.

18 Bradbury, D.,2008. Systems management: IBM middleware strategy. ComputerWeekly.com, http://www.computerweekly.com/Articles/2008/11/26/233550/Systems-management-IBM-middleware-strategy.htm.

19 CATCHMENT MODELLING TOOLKIT, 2005. Catchment modelling toolkit. Catchment modelling toolkit. Available on webpage: http://www.toolkit.net.au.

20 Chen, D., 2002. Real-time online hydrological information and modelling system using object-oriented approach and relational database for flood defence, in Wu et al. (eds.) Flood Defence'2002, USA: Science Press, New York Ltd.

21 Chen, D., Carmona-Moreno, C., Leone, A., Shams, S., 2010. Assessment of Open Source GIS Software for Water Resources Management in Developing Countries. J of Hydro-Environment Research.

22 Cluckie, I. D., D. Han, et al., 2002. Hydroinformatics 2002. London (UK), IWA Publishing.

23 Codd, E.F., 1969. Derivability, Redundancy, and Consistency of Relations Stored in Large Data Banks. IBM Research Report, 1969

24 Codd, E.F., 1990. The Relational Model for Database Management (Version 2 ed.). Addison Wesley Publishing Company.

25 Cooper, A., Reimann, R., 2003. Face 2.0: The Essentials of Interaction Design. Paperback.

26 Costanza, R., Voinov, A., 2004. Landscape Simulation Modeling: A Spatially Explicit Dynamic Approach. Springer-Verlag, New York.

27 David, O., Schneider, I.W., 2004. Metadata and modelling frameworks: The object modelling system example. Trans. 2nd Bienn. Meeting of the Int. Environ. Modelling and Software Soc., iEMSs 2004. Osnabrück, Germany, 14-17 June 2004.

28 Davis, M., Aquino, J., 2003. JTS Topology Suite Technical Specifications. Vivid Solutions. www.vividsolutions.com/JTS/JTSHome.htm.

29 De la Beaujardiere, J., 2006. OpenGIS Web Map Server Implementation Specification. Version: 1.3.0. http://www.opengeospatial.org/standards/wms

30 INSPIRE Directive 2007/2/EC of the European Parliament and of the Council of the European Union, 14 March 2007. "Establishing an Infrastructure for Spatial Information in the European Community (INSPIRE)".

31 Dublin Statement on Water and Sustainable Development, 1992. Waterlines, Volume 10, Number 4. Practical Action Publishing.

32 Edelstein, H., 1994. Unraveling Client/Server Architecture, DBMS 7, 5 (May 1994): 34(7).

33 Edlich, S., Paterson, J., 2006. The Definitive Guide to db4o, Apress.

34 ESRI (Environmental Systems Research Institute, Inc.), 1998. "Shapefile Technical Description, an ESRI White Paper". http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf.

35 Evans, J.D., 2006. OpenGIS Web Coverage Service (WCS) Implementation Specification Version: 1.1.0. http://www.opengeospatial.org/standards/wcs.

36 Ferber, J., 1989. Computational Reflection in Class-Based Object-Oriented Languages. Proc. OOPSLA'89, ACM Sigplan Notices, 24(10): 317-327.

37 Frigg, R., Hartmann, S., 2006. Models in Science. The Stanford Encyclopaedia of Philosophy. http://plato.stanford.edu/entries/models-science/.

38 Garnett, J., 2007. GeoTools Users Guide. http://docs.codehaus.org/display/geotdoc/home.

39 George, J, Batra, D., Valacich, J, Hoffer, J., 2003. Object-Oriented System Analysis and Design, Prentice Hall, 528 pp.

40 Ghosh, R. A., 2005. An Economic Basis for Open Standards. FLOSSPOLS project. http://flosspols.org/deliverables.php.

41 Gijsbers, P., Moore, R. et al., 2002. HarmonIT: Towards OMI, an Open Modelling Interface and Environment to harmonise European developments in water related simulation software.

42 Gijsbers, P.J.A. (ed.), 2003. OpenMI – Harmonizing linkages between water related models. Presented at Integrated Modelling User Group (IMUG) 2003 – International Conference on Application of Integrated Modelling, Tillburg Netherlands. Available from: http://www.wrcplc.co.uk/imug/html/imug_2003.htm

43 Gijsbers, P.J.A. (ed.), 2004. The OpenMI Architecture - Report A: Scope and Organisation, document version: 0.9. www.harmonit.org.

44 Gilfillan, I., 2003. PostgreSQL vs MySQL: Which is better?. Database Journal, http://www.databasejournal.com/features/postgresql/article.php/3288951/PostgreSQL-vs-MySQL-Which-is-better.htm

45 Gillen, A., 2009. The opportunity for Linux in a new economy. IDC, www.idc.com, http://www.linuxfoundation.org/sites/main/files/publications/.

46 Goede, E., 2005. Open Modelling System (OMS). http://www.wldelft.nl/rnd/intro/topic/2004-oms/index.html. Deltares.

47 Gonsalves. C., 2007. USDA Keeps Up with the Flow. Eweek. http://www.eweek.com/c/a/Linux-and-Open-Source/USDA-Keeps-Up-with-the-Flow/.

48 Gregersen, J. B., Blind, M., 2004. Open MI: the essential concepts and their Implications for Legacy Software, The International Environmental Modelling and Software Society Conference http://www.harmonit.org/index.php.

49 Groff, James R., Weinberg, P,N., 2002. SQL: The Complete Reference. New York, McGraw-Hill Professional.

50 GWP (Global Water Partnership) Technical Committee, 2004. Guidance in preparing a national IWRM and efficiency plan. Stockolm, Sweden.

51 HARMONIT, 2002. State of the Art Review. Document Version: 9. Hutchings C (ed.). Document produced for the HarmonIT project and available to members on registration on the HarmonIT website: www.harmonit.org.

52 HARMONIT, 2004. HarmonIT Document Series Part B – Guidelines. Document Version: 0.25. Tindall R(ed.) [Internet]. Draft document produced for the HarmonIT project and available to members on registration on the HarmonIT website: www.harmonit.org.

53 Harvey, D. P., 2002. A generic modelling framework component for hydroinformatics systems. PhD Thesis, Bristol, University of Bristol.

54 Harvey, H., Han, D., 2002. The relevance of Open Source to Hydroinformatics, Journal of Hydroinformatics.

55 Havno, K., Sorensen, H.R., Gregersen, J.B., 2001. Integrated water resources modelling and object oriented code architecture. Conference on water Resources Modelling and Management organised by the Japan academic society of hydraulics, Chuo University, Japan.

56 Helal, S., Hammer, J., Zhang, J., Khushraj, A., 2001. A Three-Tier Architecture for Ubiquitous Data Access. aiccsa, p. 0177, ACS/IEEE International Conference on Computer Systems and Applications (AICCSA'01).

57 Holz, K., Hildebrandt, G., Weber, L., 2004. Concept for a Web-based Information System for Flood Management. Journal of Natural Hazards, Volume 38, Issue 1. Springer Netherlands.

58 Johnson, L. E., 2008. Geographic Information Systems in Water Resources Engineering. University of Colorado, Denver, USA.

59 Jonoski, A., 2002. Hydroinformatics as Sociotechnology: Promoting Individual Stakeholder Participation by Using Network Distributed Decision Support Systems. ISBN 9054104279. Taylor & Francis.

60 Jordan, D., Russell, C., 2003. Java data objects. Sebastopol, California: O'Reilly.

61 Joyner, I., 1999. Objects Unencapsulated: Java, Eiffel and C++. Prentice Hall.

62 JPOX documentation. JPOX team, http://www.jpox.org/docs/1_1/index.html.

63 Karlsson, A., 2003. GIS and Spatial Extensions with MySQL, MySQL.org. http://dev.mysql.com/tech-resources/articles

64 Keogh, J., Giannini, M. (contributors), 2005. OOP Demystified. McGraw-Hill.

65 Kernighan, B., Pike, R., 1999. The Practice of Programming. ISBN 0-201-61586-X. Addison-Wesley.

66 Khatibi, R., C. Whitlow, T. Harrison, A. Akhondi-Asl, and M. Vaughan, 2002. Categorising Modelling Techniques to Progress Open Architecture in Flood Forecasting Systems. In Cluckie et al., pages 1609–1614.

67 Korb, W., 2004. Using JDO for Transparent Persistence. JavaPro, http://www.ftponline.com/javapro/2004_12/online/wkorb_12_08_04/default.aspx

68 Kralisch, S., Krause, P., David, O., 2004. Using the object modelling system for hydrological model development and application. Trans. 2nd Bienn. Meeting Int. Environ. Modelling and Software Soc. iEMSs 2004. Osnabrück, Germany, 14-17 June 2004.

69 Kralisch, S., Krause, P., David, O., 2005. Using the object modelling system for hydrological model development and application. Advances in Geosciences, 4, 75–81, 2005. European Geosciences Union.

70 Leone, A., Chen, D., 2007. Implementation of an object oriented data model in an information system for water catchment management: Java JDO and Db4o Object Database. Environmental Modelling & Software, Elsevier.

71 Leone, A., Shams, S. and Chen, D., 2006. An Object-Oriented and OpenGIS supported hydro information system (3O-HIS) for upper Mersey river basin management. Intl. J. River Basin Management Vol. 4, No. 2.

72 Linthicum, D. S., 1997. Frameworks evolve, ADTmag.com. www.adtmag.com

73 Lyon, J. 2003. GIS for Water Resources and Watershed Management. London, Taylor & Francis.

74 Maguire, D. J., Goodchild, M. F. and Rhind, D. W., 1991. Geographical Information Systems: Principles and applications. Longman

75 Maidment, D. R. (ed.), 2002. Arc Hydro, GIS for Water Resources. ESRI press.

76 Marston, F., Argent R., Vertessy, R., Cuddy, S., Rahman, J., 2002. The Status of Catchment Modelling in Australia. Technical Report No. 02/4. Cooperative Research Centre for Catchment Hydrology (CRCCH), Australia. Available from: http://www.toolkit.net.au.

77 McClure, S., 1997. Object Database vs. Object-Relational Databases. IDC Bulletin #14821E. www.idc.com.

78 Meijer, E., Drayton, P., 2004. Static Typing Where Possible, Dynamic Typing When Needed: The End of the Cold War between Programming Languages. Microsoft Research. Proc. OOPSLA 2004 Workshop on Revival of Dynamic Languages.

79 Meyer, B., 1997. Object-Oriented Software Construction, Second Edition. Prentice Hall Professional Technical Reference.

80 Moore, R., Gijsbers, P., Fortune, D., Gregersen, J., Blind, M., 2007. OpenMI Document Series: Part A - Scope for the OpenMI (version 1.4). http://www.openmi.org/reloaded/about/documents-publications/A_OpenMI_Scope.pdf. Isabella Tindall, Centre for Ecology and Hydrology, Wallingford, UK.

81 Murray, N., Perraud, J.M., Rahman, J., Seaton, S., Hotham, H., Watson F., 2004. Introduction to TIME. Workshop notes. Cooperative Research Centre for Catchment Hydrology (CRCCH), Australia. Available from: http://www.toolkit.net.au/time.

82 Nebert, D., 2007. OpenGIS catalog services specification, Version 2.0.2. OpenGIS project document OGC 07-006r1. Open GIS Consortium Inc., http://www.opengis.org.

83 OMS, 2005. OMS – Central Collaboration Platform. OMS webpage: http://oms.ars.usda.gov.

84 Paterson, J., 2004. Simple Object Persistence with the db4o Object Database. O'Reilly, http://www.onjava.com/pub/a/onjava/2004/12/01/db4o.html.

85 Prechelt, L., 2000. An Empirical Comparison of Seven Programming Languages. Computing Literature. IEEE Computer Society Press, Los Alamitos, CA, USA

86 Rahman, J.M., Seaton, S. P., and Cuddy, S. M., 2004. Making frameworks more useable: using model introspection and metadata to develop model processing tool, Environmental Modelling and Software, 19, March, 2004.

87 Rahman, J.M., Seaton, S.P., Perraud, J. M,, Hotham, H., Verrelli, D.I., Coleman, J.R., 2003. It's TIME for a New Environmental Modelling Framework. Proc. Of the MODSIM 2003 Int. Congress on Modelling and Simulation: Townsville, Modelling and Simulation Society of Australia and New Zealand inc.

88 Ramsey, P., 2007. The State of Open Source GIS. http://www.refractions.net/expertise/whitepapers/opensourcesurvey/survey-open-source-2007-12.pdf. Refractions Research Inc.

89 Raymond, E.S., 2003. The Art of Unix Programming. Addison-Wesley Professional, part of the Addison-Wesley Professional Computing Series.

90 Reed, M., Cuddy, S.M., Rizzoli, A.E., 1999. A framework for modelling multiple resource management issues – an open modelling approach. Environmental Modelling & Software 14 (1999).

91 Rindal, C. 2007. MySQL vs. PostgreSQL, white paper. Tometa software. http://www.tometasoftware.com/files/MySql-v-PostgreSQL.pdf.

92 Schmidt, D. C., Porter, A., 2001. Leveraging Open-Source Communities to Improve the Quality & Performance of Open-Source Software. 1st Workshop on Open Source Software, ICSE 2001.

93 Shaw, E. M., 1994. Hydrology in Practice. Chapman & Hall, London, UK, 3rd edition.

94 Shirky, C., 2001. Interoperability, Not Standards. O'Reilly, http://www.openp2p.com/pub/a/p2p/2001/03/15/clay_interop.html

95 Shneiderman, B., 1986. Designing the User Interface: Strategies for Effective Human-Computer Interaction. Addison-Wesley Publishing Company, Reading, Massachusetts.

96 Sikora, Z., 2003. Java: Practical Guide for Programmers. San Francisco, California, Oxford: Elsevier.

97 Solomatine, D., 1996. Object orientation in hydraulic modelling architectures. Journal of Computing in Civil Engineering. Vol.10, No.2, April.

98 Spanou, M., Chen, D., 2000. An Object-Oriented tool for the control of point-source pollution in river systems. Environmental Modelling and Software, 15(1), 35-54.

99 Spanou, M., Chen, D., 2001. A river water quality simulation system using Object-Oriented method for the Upper Mersey River system. J. of hydroinformatics, 3(3), 173-194.

100 Spanou, M., Chen, D., 2002. Integrated management of Upper Mersey River basin using the SMILE Object-Oriented software system. IWA Water Science and Technology. 46 (6-7), 105-112.

101 SUN, 2007. SUN check list for Open standards. http://www.sun.com/software/standards/definition.xml

102 SUN, 2010. Java Technology for Business Intelligence -- Sun Technology Guide Draft 4.0. http://java.sun.com/products/jmi/pres/guide.html

103 SunSoft, 1994. Multithreaded Programming Guide. Sun Microsystems, Inc. 2550 Garcia Avenue, Mountain View, California.

104 Sutherland, J., 1997. The Java revolution, SunExpert Magazine, January.

105 Sydelko, P. J., Hlohowskyj, I., Majerus, K., Christiansen, J., 2001. An object-oriented framework for dynamic ecosystem modeling: application for integrated risk assessment. The Science of The Total Environment, Volume 274, Issues 1-3, 2 July 2001.

106 Tarboton, D., Horsburgh, J., Maidment, D., 2007. CUAHSI Community Observations Data Model (ODM) Version 1.0 Design Specifications. http://www.cuahsi.org/his/docs/ODM1.pdf.

107 Tomicic, B., Yde, L., 1998. Integrated software for an integrated management and planning of urban drainage and wastewater systems. Babovic and Larsen (1998).

108 UNEP (United Nations Environmental Program)/MAP/PAP (1999). Conceptual Framework and Planning Guidelines for Integrated Coastal Area and River Basin Management. Split, Priority Actions Programme.

109 USA, Executive Order 12906, published in the April 13, 1994, edition of the Federal Register, Volume 59, Number 71, pp. 17671-17674.

110 USEPA, 2001. BASINS, 2001. Better Assessment Science Integrating Point and Nonpoint Sources (Version 3.0). User Manual. Office of Water, United States Environmental Protection Agency (USEPA), 401 Mth Street SW, Washington, DC 20460, USA. http://www.epa.gov/waterscience/basins/bsnsdocs.html.

111 Uslander, T., 2005. Trends of environmental information systems in the context of the European Water Framework Directive. Environmental Modelling & Software, Volume 20, Issue 12.

112 Von Krogh, G., Spaeth, S., 2007. The open source software phenomenon: Characteristics that promote research. The Journal of Strategic Information SystemsVolume 16, Issue 3, Pages 236-253.

113 Vretanos, P. A., 2005. OpenGIS Web Feature Service Implementation Specification Version: 1.1.0. http://www.opengeospatial.org/standards/wfs

114 Wasson, J. et al., 2003. What kind of water models are needed for the implementation of the European Water Framework Directive? Examples from France. Intl. J. River Basin Management Vol. 1, No. 2 (2003), pp. 125–135.

115 West, L. A. Jr., 2000. Designing end-user geographic information systems. Journal of End User Computing, IGI Publishing, Hershey, PA, USA