Revista de Ingeniería
ISSN: 0121-4993
reingeri@uniandes.edu.co
Universidad de Los Andes
Colombia

Jiménez Guarín, Claudia; Gómez, Juan Erasmo
GENESIS: Agile Generation of Information Management Oriented Software
Revista de Ingeniería, núm. 31, enero-junio, 2010, pp. 30-39
Universidad de Los Andes
Bogotá, Colombia

GENESIS: Agile Generation of Information Management Oriented Software

GENESIS: Generación ágil de software orientado a gestión de información

Claudia Jiménez Guarín[a], Juan Erasmo Gómez[b]

KEY WORDS

Agile development, information requirements.

ABSTRACT

The specification for an information system can be clear from the beginning: it must acquire, display, query and modify data, using a database. The main issue is to decide which information to manage. In the case originating this work, information was always evolving, even up to the end of the project. This implies the construction of a new system each time the information is redefined. This article presents Genesis, an agile development infrastructure, and proposes an approach for the immediate construction of required information systems. Experts describe their information needs and queries, and Genesis generates the corresponding application, with the appropriate graphical interfaces and database.

PALABRAS CLAVES

Desarrollo ágil de software, requerimientos de información.

RESUMEN

La especificación de un sistema de información puede estar clara desde el principio: debe adquirir, desplegar, consultar y modificar datos, usando una base de datos. El asunto es decidir cuál información manejar. En el caso que origina este trabajo, la información evoluciona permanentemente, incluso hasta el final del proyecto. Esto implica la construcción de un nuevo sistema cada vez que se redefine la información. Este artículo presenta Genesis, una infraestructura ágil para la construcción inmediata del sistema de información que sea requerido. Los expertos describen su información y consultas. Genesis produce el software correspondiente, generando las interfaces gráficas y la base de datos apropiados.

a PhD. en Informática. Profesora asociada, Departamento de Ingeniería de Sistemas y Computación, Universidad de los Andes. Bogotá D.C., Colombia. ✉ cjimenez@uniandes.edu.co

b M.Sc. en Sistemas y Computación. Instructor, Departamento de Ingeniería de Sistemas y Computación, Universidad de los Andes. Bogotá D.C., Colombia. ✉ jua-gome@uniandes.edu.co

INTRODUCTION

The software conception and development process usually begins with a requirement gathering phase. These requirements define the scope, functionality and technical conditions for the software product, as well as the "business" needs, that is, those conditions justifying the software process construction and further use [1].

In the software construction process for supporting information based systems, one frequently finds functional requirement descriptions in terms of "enter, modify, delete or obtain some information according to a given criteria". These operations are usually denominated CRUD operations (Create, Retrieve, Update, Delete) [2]. In the non functional aspects, it is usual to require database support, distribution and a graphical user interface. The database supports mainly data storing, efficient querying, transactional features, and guarantees information consistency. The graphical interface must allow significant, robust and ordered interactions between the user and the information.

Thus, the challenge for the analysts and developers in this kind of information systems is to accurately determine the relevant information in order to guarantee the information integrity rules and the needed query constructions. As a result of this process, the appropriate database and user interfaces are designed

In the beginning, the information system originating this work was non-surprising classical CRUD software. It had to catalog the data obtained from several diagnosis methods, on a same patient data. The diagnosis methods were defined by clinical, medical and engineering experts. Their corresponding approaches had to be evaluated using the results of queries obtained with the information system [3]. Experts were radiologists, neurologists, mechanical engineers and computer medical image treatment engineers.

From the software requirement point of view, the initial problem was quite simple and not really challenging. In the earlier days of the project, all the functional and technical requirements were clear, as well as the system scope. However, it was impossible to begin the software construction process! Given the project's nature, it was impossible to determine, in an earlier phase of the project, which information could be considered.

The information description and structure, and the expert knowledge, continually evolved over time. As the diagnosis methods research evolved, experts were defining the relevant information. In this way, the information definition, restrictions and rules were highly changing during the whole project evolution. In practical terms, once the information requirements were introduced in the software process, most of the time, they became obsolete rather quickly. Traditional software developing methodologies [4], as well as known agile methodologies like extreme programming [5] were revealed to be unsuitable. In this case, the functional or technical requirements are not evolving, as it is usual in many software projects leading to complex architectures dealing with requirement administration [6] [7] [8]. The challenge is to deal with highly evolving information specification. Changes to the information definition can occur in data types, structuring complex information, enumeration ranges, or, even, all new information definition. These changes have impact in the database definition, integrity constraints, input data interface, browsing and querying interface. None of these aspects are covered by traditional methodologies or tools for managing software evolution, adaptability or refactoring.

This paper presents Genesis, an agile construction approach for developing information oriented management systems in the context of evolving information requirements. The information requirements, as well as queries, are described by the experts, using a declarative and simple to use XML-based language. Genesis constructs, on the fly, the corresponding CRUD information system, including the graphical user interface and the database support. This approach allows the software product construction to be highly efficient, robust and reliable, and as fast as the expert knowledge evolves.

In part 2 we present the Genesis solution approach. Next, in part 3, we present the proposed information description and query languages. Part 4 presents Genesis architecture and design. Part 5 describes the specific example of Genesis use in the context of the comparison of diagnosis methods for carotidal stenosis. Finally, conclusions and future work are presented.

THE GENESIS APPROACH

The main challenge in Genesis is to quickly produce an information system corresponding to the information requirements established by the experts. The information definition can evolve by introducing new attributes, changing data types, reorganizing complex structures or modifying data integrity restrictions as possible values, ranges or enumeration values. This produces a new information system specification, and consequently a new software product.

In a first version of the project, a traditional information system was developed, with a relational database to support data management. This solution was quickly obsolete because of project evolution and the resulting information system was nearly unused. As the research project started a second phase [9], it was clear that new, dynamic and quickly reactive development software process was required.

The chosen approach was to develop a software environment capable of producing a functional software system for CRUD information management, using a declarative descriptor of the desired data and queries. From the software developing process point of view, the main goal is to produce the required information system decoupling the specific information definitions, in order to avoid source code modification caused by evolution of information requirements. An additional issue for Genesis is the independence of the initial medical research project needs, such that it can be applied to other contexts.

With this approach, the experts make a description of their research information, using a proposed syntax. Based on this description, the complete software system must be constructed in an automatic procedure. This procedure includes the graphical user interface for information input, the interface for browsing and editing the existing information and the corresponding database generation. Both software components must take into account the defined information integrity constraints. Then, by using an expert query language, the generated information system must produce the desired comparison for the medical research purposes.

Two different technical solutions were developed using this approach: a first phase, using relational database technology; and Genesis, the second phase, using XML technologies intensively. Genesis description and results are the main contribution of this paper.

BEFORE GENESIS

In the first phase, technical decisions were guided by two main goals: a very simple description language for the experts, and the use of traditional relational database technology. This solution was fully implemented, using java and an XML-like language for data description.

The resulting product [10] was functional, but highly coupled with the medical research problem and their information structure. It also reflected information representation features of relational database systems; in particular non hierarchical information units were allowed and relationships between units were defined in a separate descriptor. The description language allowed complex units, data types and integrity restrictions but intentionally avoided introducing information attributes needed to fit relational definitions. Extra data identifiers, subrogates or primary keys were introduced by the system at generation time. Hibernate [11] and introspection were the key tools for producing the desired system and database.

Once the information description was given by the experts, they established the queries to be validated using the resulting information system. Complex queries were really hard to produce, mainly due to the relational normalization of hierarchical information:

experts were forced to introduce extra attributes, unknown to them. These extra unknown attributes corresponded to integrity foreign keys, and the results were unexpected cross products or uncompressible query expressions. Resolving this complexity introduced many features in the software completely dependent on the medical case study.

GENESIS PROPOSAL

Using the experience obtained in the first phase, the decision was to preserve the solution approach, but to use an all new technological approach. The present phase, Genesis, eliminates relational database technology, data model impedance and translation; and simplifies software design and implementation. An all new expert language has been designed, allowing hierarchical information definition and easier queries.

In Genesis, data model descriptors and queries are written using an expert appropriated languages. In the present Genesis version those languages are simplifications of XML; those expert languages are automatically traduced to XML, making the Genesis processing module as powerful as XML is. Before system information execution, the expert data model description document is translated to the corresponding Genesis descriptor schema, which is expressed in `XML Schema` [12]. Figure 1 presents an example of a Genesis information definition and Figure 2 presents the corresponding Genesis generated `XML Schema`. Then, at runtime, using the generated schema, the corresponding information system is generated, including the appropriate database and user interfaces for data entry and browsing.

At query time, the expert query language is transformed to XQuery and then it is processed. In this way, XML is not only used in the classic way, for data representation. The additional and innovative role of XML in Genesis is to be a multipurpose tool, used both for the expert model and query description, as well as for the automatic generation of the application database schema and the software's graphic interface.

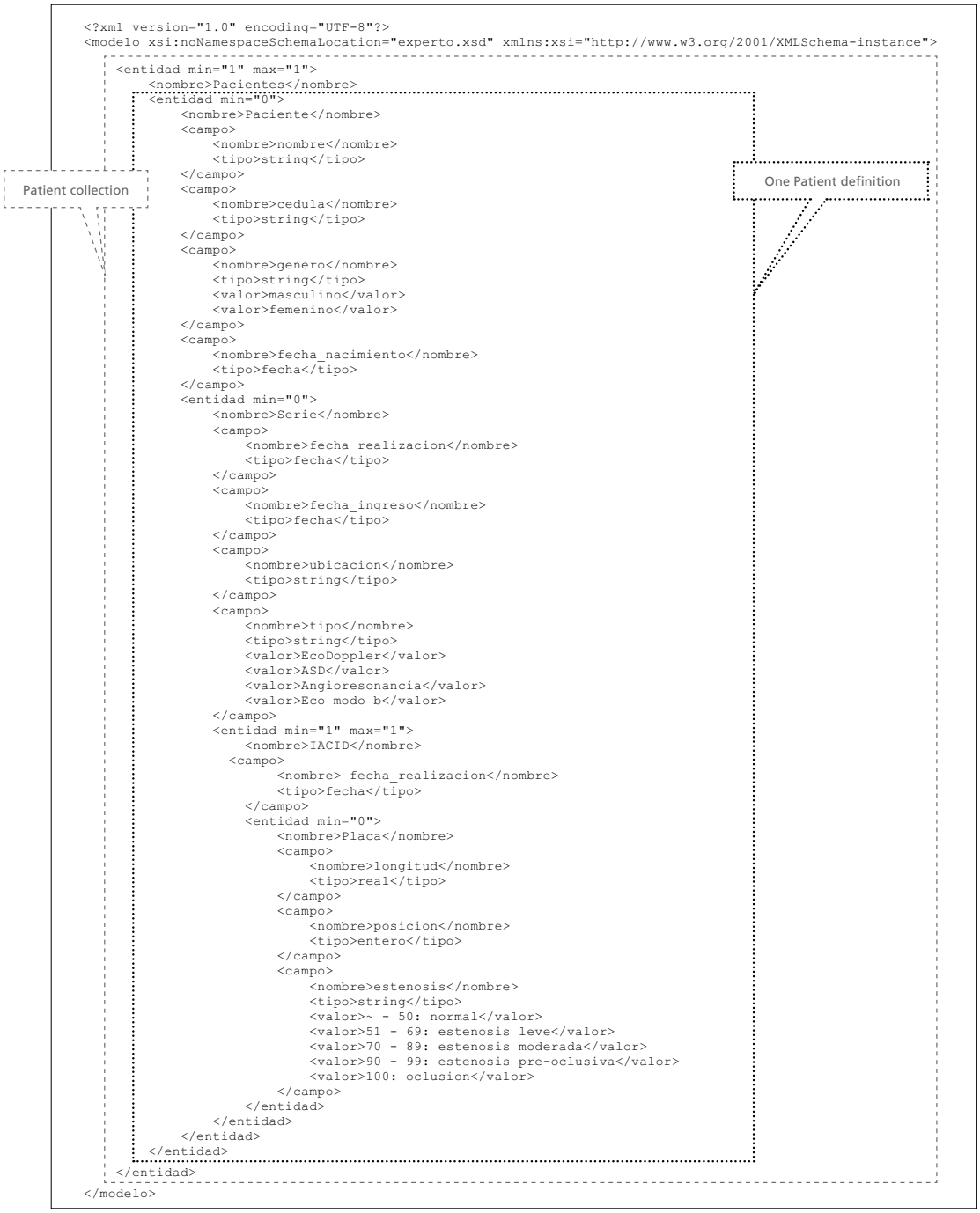
THE GENESIS EXPERT INFORMATION MODEL AND QUERY DESCRIPTION

In this chapter we present the designed Genesis definition and query expert languages, and the resulting graphical interfaces. The expressivity of the expert definition language must include data types, hierarchical structures as complex as necessary, cardinality expressions (min, max) for any attribute, and controlled values. The examples are shown for the context of the original case study, the Carotid Stenosis Diagnosis. Genesis is completely decoupled of the case study and the corresponding information descriptors.

As presented in the Genesis approach, at runtime Genesis uses a `Genesis XML Schema` for the automatic generation of the desired information system (Figure 2). `XML Schema` is not friendly and not appropriate for final experts of the system. Then, we proposed to the experts a data model definition syntax, which in fact corresponds to a XML simplification, avoiding mainly the syntactical overhead concerning the namespaces (Figure 1). It also offers XML significant `tags`, making easier for them to define and to verify the completeness of the desired integrity constraints.

In order to describe the application data model, the expert produces an XML Genesis compliant document. Using this information, Genesis produces a complete `Genesis XML Schema` syntax, using XSLT. The result is a computational description usable with tools like DOM [13] and Xerces [14].

This processing approach allows decoupling the expert data model descriptor language and the actual schema definition managed by Genesis. In this way, another expert language can be chosen in order to adapt it to more specific purposes, making the data model definition phase more user-friendly for non-technical users. In this case, the only extra work is the construction of the corresponding XSLT, to produce the generated `Genesis XML Schema`. The main advantage on this approach is that it allows decoupling of the processing technology from the expert languages, requiring only language translation modules for new expert languages when needed.




```
<?xml version="1.0" encoding="UTF-8"?>
<modelo xsi:noNamespaceSchemaLocation="experto.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <entidad min="1" max="1">
        <nombre>Pacientes</nombre>
        <entidad min="0">
            <nombre>Paciente</nombre>
            <campo>
                <nombre>nombre</nombre>
                <tipo>string</tipo>
            </campo>
            <campo>
                <nombre>cedula</nombre>
                <tipo>string</tipo>
            </campo>
            <campo>
                <nombre>genero</nombre>
                <tipo>string</tipo>
                <valor>masculino</valor>
                <valor>femenino</valor>
            </campo>
            <campo>
                <nombre>fecha_nacimiento</nombre>
                <tipo>fecha</tipo>
            </campo>
            <entidad min="0">
                <nombre>Serie</nombre>
                <campo>
                    <nombre>fecha_realizacion</nombre>
                    <tipo>fecha</tipo>
                </campo>
                <campo>
                    <nombre>fecha_ingreso</nombre>
                    <tipo>fecha</tipo>
                </campo>
                <campo>
                    <nombre>ubicacion</nombre>
                    <tipo>string</tipo>
                </campo>
                <campo>
                    <nombre>tipo</nombre>
                    <tipo>string</tipo>
                    <valor>EcoDoppler</valor>
                    <valor>ASD</valor>
                    <valor>Angioresonancia</valor>
                    <valor>Eco modo b</valor>
                </campo>
                <entidad min="1" max="1">
                    <nombre>IACID</nombre>
                  <campo>
                        <nombre> fecha_realizacion</nombre>
                        <tipo>fecha</tipo>
                    </campo>
                    <entidad min="0">
                        <nombre>Placa</nombre>
                        <campo>
                            <nombre>longitud</nombre>
                            <tipo>real</tipo>
                        </campo>
                        <campo>
                            <nombre>posicion</nombre>
                            <tipo>entero</tipo>
                        </campo>
                        <campo>
                            <nombre>estenosis</nombre>
                            <tipo>string</tipo>
                            <valor>~ - 50: normal</valor>
                            <valor>51 - 69: estenosis leve</valor>
                            <valor>70 - 89: estenosis moderada</valor>
                            <valor>90 - 99: estenosis pre-oclusiva</valor>
                            <valor>100: oclusion</valor>
                        </campo>
                    </entidad>
                </entidad>
            </entidad>
        </entidad>
    </entidad>
</modelo>
```

Figure 1. Genesis expert information definition example

Using the `Genesis XML Schema`, at runtime, the graphical interface is automatically generated. Figure 3 shows a generated interface for the case study. It shows complex and simple elements, as well as the interaction elements for collection browsing. A particular graphical element is generated for each information type. Basic data, complex unique subelements, and complex multi-valued entities have a specific graphical representation in the user interface, as shown in Figure 3.

The Genesis Query language is XQuery with syntactical simplifications; mainly, the syntactical XQuery overhead dealing with namespaces is eliminated. This allows expressions as complex as required, using only the originally described elements. Figure 4 presents a Genesis query example. Existing XML technology, mainly parsers and native-XML databases, are used in Genesis implementation. This allows delegating to known and reusable technology much of the responsibilities of information integrity validation.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:ca="http://cifi65/carotida"
targetNamespace="http://cifi65/carotida" elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:complexType name="Pacientes_TYP">
        <xs:sequence>
            <xs:element ref="ca:Paciente" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="Pacientes" type="ca:Pacientes_TYP"/>
    <xs:complexType name="Serie_TYP">
        <xs:sequence>
            <xs:element name="fecha_realizacion" type="xs:date" id="hgf"/>
            <xs:element name="IACID" type="ca:IACID_TYP" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="Serie" type="ca:Serie_TYP"/>
    <xs:complexType name="Paciente_TYP">
        <xs:sequence>
            <xs:element ref="ca:Serie" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="ca:Laboratorio" minOccurs="0"/>
            <xs:element ref="ca:Reporte_Hemodinamico" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="cedula" type="xs:string" use="required"/>
    </xs:complexType>
    <xs:element name="Paciente" type="ca:Paciente_TYP"/>
        <xs:complexType name="IACID_TYP">
        <xs:sequence>
            <xs:element name="fecha_realizacion" type="xs:date"/>
            <xs:element name="placas" type="ca:Placa_IACID_TYP" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="IACID" type="ca:IACID_TYP"/>
        <xs:complexType name="Placa_IACID_TYP">
        <xs:sequence>
            <xs:element name="longitud" type="xs:double"/>
            <xs:element name="posicion" type="xs:int"/>
            <xs:element name="estenosis" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="Placa_IACID" type="ca:Placa_IACID_TYP"/>
</xs:schema>
```

Figure 2. `Genesis XML Schema` Schema definition

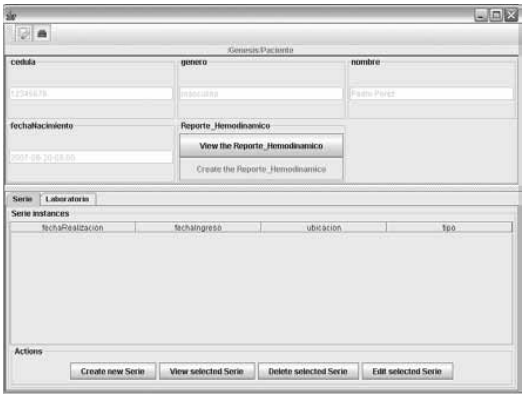


Figure 3. Genesis generated interface for data entry

Figure 4 shows an example of a simple expert query. It looks for the patient's name and carotid plaques, for all patients. For queries, Genesis offers an assisted graphical interface to the experts. The information definition tree is displayed, in order to avoid missing syntax or additional XML viewers. The information elements can be selected and used in a query expert language, edited in the Genesis provided query user interface.

The expert can edit, save, retrieve and reuse queries in order to find new relationships in the stored information.

Genesis allows more complex queries, indicating graphical preferences for result display. Experts can use pie charts, bars, lines, scattered graphics or table displays. In figure 5 we present a complex query, producing a pie chart of the total plaques found in

```
for $x in /Genesis/Paciente
return      <result>{$x/nombre}<placas>{$x//Placa}</placas>
</result>
```

Figure 4. Simple expert query

patients, according to the grading diagnosis values (`"normal"`, `"leve"`, `"moderada"`, `"preoclusiva"` and `"oclusion"`). These values correspond to the defined grades found in the Placa possible values, in the expert definition language. Figure 6 shows some examples of generated graphical interface for Figure 5 expert query.

The expert query language is translated by Genesis to standard XML query languages, using `XQuery` and `XPath`, in a similar way as it is done with the definition languages. All the corresponding overhead syntax is not visible to the expert. The expert query language can also be replaced, by providing the corresponding translation module to `XQuery`.

```
show pie chart
<PieDataset>
 <Item>
       <Key>Estenosis normal</Key>
       <Value>{
        for $x in /Genesis
        return fn:count($x//Placa[./estenosis='~ - 50: normal'])
       }
       </Value>
 </Item>
 <Item>
  <Key>Estenosis leve</Key>
       <Value>{
        for $x in /Genesis
        return fn:count($x//Placa[./estenosis='51 - 69: estenosis leve'])
       }
       </Value>
 </Item>
 <Item>
       <Key>Estenosis moderada</Key>
       <Value>{
        for $x in /Genesis
        return fn:count($x//Placa[./estenosis='70 - 89: estenosis moderada'])
       }
       </Value>
 </Item>
 <Item>
       <Key>Estenosis pre oclusiva</Key>
       <Value>{
        for $x in /Genesis
        return fn:count($x//Placa[./estenosis='90 - 99: estenosis pre-oclusiva'])
       }
       </Value>
 </Item>
 <Item>
       <Key>Oclusicion</Key>

       <Value>{
        for $x in /Genesis
        return fn:count($x//Placa[./estenosis='100: oclusion'])
       }
       </Value>
 </Item>
</PieDataset>
```

Figure 5. Complex expert query example

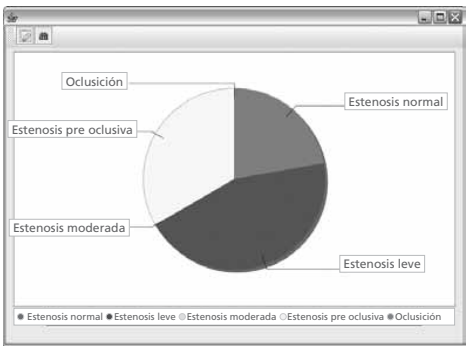


Figure 6. Graphical result for a complex expert query

GENESIS: SOFTWARE ARCHITECTURE AND GENERATED INFORMATION SYSTEM DESCRIPTION

Genesis is a Java and XML based distributed system. Figure 7 shows the general software architecture of Genesis.

The Genesis software architecture is mainly structured in components allowing the data model discovering, the generic model management, the information relationship analyzing, and the query management. A native XML database, eXist [15] is the chosen product for persistence. This technology decision guarantees no impedance in the data manipulation model and the persistent database model. None the less, the architecture is designed so that other persistence technologies (i.e. relational databases) could be use in place XML based products. Figure 7 shows the general design of the main Genesis components.

The Model Analyzer (MA) component is responsible for the model manipulation and analysis. It uses the generated `Genesis XML Schema` and produces an internal representation of the data model, considering the hierarchical definitions and the integrity constraints. Its main objective is to provide an abstraction of the data model and the necessary services to process it. The GUI Generator uses the services offered by the MA to generate a user interface that corresponds to the user defined data model. It is responsible for the GUI data entry interface, the data browsing user interface and the query user interface. This includes input fields and dialog for different kind of data types, and output elements. It also considers the structure of the information and integrity restrictions.

The XML Database Manager is the interface with eXist. The main objective of this component is to isolate the specific code related to eXist in order to reduce the dependency on any specific product. The CRUD Server is responsible for the CRUD data operations and the query execution. Its main goal is to make the architecture flexible enough to support different kind of persistence technology (currently, a XML persistence implementation is used).

RESULTS, CONCLUSIONS AND FUTURE WORK

Genesis is a fully functional solution for the agile construction of information systems based on CRUD information requirements. Experts are responsible for the information definition, which guarantees easy integrity and completeness validation, both to experts and technology. Genesis produces a tailored informa-

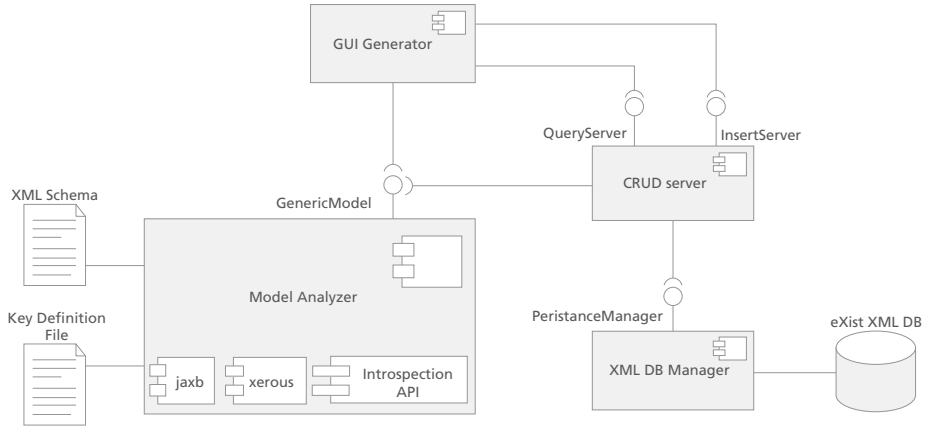


Figure 7. Genesis General Software Arquitecture

tion system, where all the interfaces and database constraints are based on expert information description. Thus, Genesis constitutes an effective way for software construction in the case of evolving information requirements: no program code changes are needed when information definition changes, and all the information features are well considered. This approach has a very efficient trade-off between expert training and flexibility, and efficient computational results. The desired new information system is generated in few minutes, for very complex information description.

The proposed expert languages have been shown to be expressive enough for the experts in the case study project [3]. The expert descriptors, expressed in an XML-like language, have show to be complete and appropriate, even if they are not so friendly. Little training is necessary for the description language and for simple queries. Complex queries may consider additional support, so graphical tools to improve end-user interaction and complex query construction are considered as future work.

Semantic problems may arise in updating the information definition if information is already present. Genesis does not discover or resolve information definition incompatibilities with previous versions. In the project experience about evolving information definitions we found that the encountered changes are almost unpredictable. For this reason, a new information system is constructed for each new information description. If information is present in the old version, an additional process for information migration is needed. In the Genesis architecture proposal we achieved a fully decoupling the medical research case study from the implementation. Then, future uses of Genesis do not require additional developing work.

For a next version of Genesis, some work is to be done, allowing the automation of some tasks, as starting XSLT translations between expert languages and schema. In order to introduce specific business requirements other than CRUD operations, further work should include more complex information relationships or the inclusion of other business operations. The expert proposed languages can be adapted or replaced if new information description needs are found.

REFERENCES

[1] B. Bruegge and A.H. Dutoit.
Object-oriented software engineering : using UML, patterns and Java. New Jersey : Prentice Hall, 2004.

[2] H. Kilov.
"From semantic to object-oriented data modeling". *Proceedings of the First International Conference on Systems Integration*. IEEE Computer Society Press, Los Alamitos, 1990, pp. 384–393.

[3] L. F. Uriza, J.A. Arias, E.M. Nieto, M. Hernández Hoyos, J.C. Briceño.
"Desarrollo de un modelo computacional para estudio de la enfermedad arteriosclerótica carotídea". *XXIX Congreso Nacional de Radiología*. Cartagena : Octubre de 2004.

[4] A.J.C. Blyth, J. Chudge, J.E.Dobson and M.R.Strens.
"A Framework for Modelling Evolving Requirements". IEEE Xplore [ed.]. *Computer Software and Applications Conference, COMPSAC 93*. 1993, pp. 356- 361. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=404219&isnumber=9093.10.1109/CMPSAC.1993.404219

[5] K. Beck and C. Andres.
Extreme Programming Explained: Embrace Change. 2nd ed. Addison-Wesley Professional, 2004.

[6] D. Kulak and E. Guiney.
Use Cases: Requirements in Context. 2nd ed. Addison-Wesley Professional, 2003.

[7] M.A. Corsello.
"System-of-Systems Architectural Considerations for Complex Environments and Evolving Requirements". *IEEE Systems Journal*. Vol. 2, No. 3, September 2008, pp. 312. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4595695&isnumber=4626033.10.1109/JSYST.2008.925972

[8] M.W. Godfrey and D.M. German.
"The Past, Present, and Future of Software Evolution". IEEE Xplore. [ed.]. *Frontiers of Software Maintenance, FoSM 2008. September 2008*. pp 10. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4659256&isnumber=4659234.10.1109/FOSM.2008.4659256

[9] I. Fernández, M.A. Navas, M.A. Zuluaga, L.F. Uriza, M. Hernández Hoyos, J.C. Briceño.
"Carotid stenosis disease: use of mathematical and computational analysis as a diagnostic aid". *53nd Annual Conference of American Society for Artificial Internal Organs (ASAIO)*. June 7-9, 2007, Chicago, USA.

[10] C. L. Jiménez, J. Grijalba, C. H. Jiménez, S. Moreno, J. E. Gómez.
"Software construction for evolving systems with incomplete information definition". *XML 2006 Conference*. Boston, Dec 2006.

[11] C. Bauer and G. King.
Hibernate in action. Greenwich: Manning Publications Co., 2005.

[12] World Wide Web Consortium.
XML Schema. 2007. Available: http://www.w3.org/XML/Schema

[13] World Wide Web Consurtium W3C.
Document Object Model. Cited: August 13, 2008. Available: http://www.w3.org/DOM/

[14] The Apache Software Fundation.
Xerces Java Parser. Cited: August 13, 2008. Available: http://xerces.apache.org/xerces-j/

[15] W. Meier.
Open Source Native XML Database. Cited: August 13, 2008. Available: http://exist.sourceforge.net/

Recibido 26 de Septiembre de 2008, modificado 16 de Septiembre de 2009, aprobado 30 de Marzo de 2010.