Software Evolution Principles Applied to a Real Case: Process Information Management in a Nickel Factory

Raúl E. Menéndez Mora[1], Mauro M. García Pupo[2], Nuria Medina Medina[3], Marcelino Cabrera Cuevas[4]
[1,2]Universidad de Holguín, Cuba [3,4]Universidad de Granada, Spain
[1]raul@facinf.uho.edu.cu, [2]mauro@cristal.hlg.sld.cu, [3]nmedina@ugr.es, [4]mcabrera@ugr.es

Abstract
Maintenance is considered on many tendencies as part of a wider discipline called Software Evolution. The formal and well defined concepts provided by this discipline permit accomplishing software maintenance in a consistent and reliable way. This work is based on three foundations for evolution: an architecture, a set of mechanisms and models, and the concept of parametrizable system; and it describes the application of these concepts to a real system at the Cuban "Ernesto Che Guevara" Nickel Factory in order to automates part of its maintenance.
Keywords: Software Evolution, Metamodeling, Parameterizable Systems.

1. Introduction

Software development is an iterative process in which the product changes with extreme ease, which causes the maintenance of the software to be complex. Traditionally, the focus of the development of software for the maintenance has pursued improving this property, since maintenance takes approximately between 50% and 70% of the total cost of the development. Although the inclusion of new errors when trying to fix others causes a progressive deterioration of the product, maintenance is necessary to correct errors, enlarge or adapt the software or simply to perfect it. At the moment, numerous tendencies (Lehman et. al., 2001a, 2001b; Madhavji et. al., 2006; Medina et. al., 2002; Parets and Torres, 1996, 1999) consider maintenance a part within a much wider discipline denominated software evolution. The formal and well-defined concepts that are provided by this discipline allow carrying out the maintenance in a consistent and reliable way, reducing accordingly the deterioration of the software which is the main cause of the crisis or rather "chronic illness" that the software goes through from its birth.

The "Comandante Ernesto Che Guevara" Nickel Factory, located in eastern Cuba is one of the biggest industries in the country. In 2004, technological changes in the process of production of the factory affected significantly the information systems that supported the information management of the production process. The main objective of this paper is the combination of several evolutionary theories applied with the purpose of developing more robust and flexible computer systems in favor of an information management that represents the objective reality of the factory over time.

Specifically, the paper is structured as follows: Section 2 establishes and defines the concepts, models and architectures that constitute the main evolutionary principles applied in a practical way to improve the software systems used in the nickel factory. Section 3 describes the practical application of the presented evolutionary theory with relative details. Finally, section 4 presents brief conclusions of the work carried out.

2. Theoretical Principles

2.1 General Concepts of Evolution

A system is characterized by being active, stable and evolutionary in its environment and with reference to its purpose. In this way, the modeling of the system should cover three representations: functional (external), organic (internal) and historical (genetic). Moreover, if the modeler is capable of representing itself as any other object of the system, it constructs a systemic model of itself. On the other hand, we will take process to mean: "all change in time of matter, energy or information", activity in which two fundamental elements intervene: the changed object (processed) and the object that produces the change (processor).

From the Representation System viewpoint (Le Moigne, 1990a) it is interesting to distinguish between the active intervention of the object as processor and the performance of the object as processed. The first one shows "the operation, the almost visible immediate activity, the active and intentional intervention", the second represents "the evolution, the internal transformations, the mutations, the equilibrium ruptures that although they don't affect the nature, the uniqueness of the represented object, they can concern their forms and behaviors" (Le Moigne, 1990b).

2.2 Architecture for the Evolution of a Software System

A software system is a group of processors that interact with each other and with the environment, so that the whole system can be seen, from a functional perspective, as one processor (Parets, 1995). The modeler executes changes in the system during the development process, and also later during its functional life. These last changes modify the structure or functionality of the system in order to produce adaptations that guarantee the utility of the interaction of the software system with its environment. Therefore, the ability of a system to evolve implies the anticipation of the types of modifications a system can undergo during its development and operation. Then the developer will be able to, in the future, to carry out necessary changes (structural or functional) in the software system and adapt it to its surroundings in an easier and more flexible way (Medina et. al., 2002).

In the process of evolution of the software system, the developer is a very important element because he is responsible to model and design the capacity of evolution of the system and later to carry out evolutionary actions to produce the necessary changes on the system. Based on this, there are two ways in which a software system can evolve:

- **Evolution** of the system, lead by the developer.

- **Auto-evolution** of the system, carried out in an automatic way (depending on certain mechanisms previously defined by the developer).

For a software system to evolve, its architecture should recognize two levels of abstraction: system and meta-system. The meta-system provides the developer with evolutionary actions to create and modify the system. A system can be seen as a group of related subsystems. Therefore, to guarantee the consistency of the complete system two mechanisms are needed:

- Restrictions that check that the change on the subsystem is coherent, that is, an evolutionary action is not allowed to execute if the state of the subsystem does not satisfy the restrictions imposed for that change.
- Automatic propagation of the change. Sometimes a change affects other elements of the modified subsystem and it can even have repercussions on other related subsystem or subsystems that interact with it. This causes it to be necessary to carry out new changes to maintain a consistent state at global level.

Based on what has been explained, Parets in (1995) proposes the interaction structure shown in figure 1a.

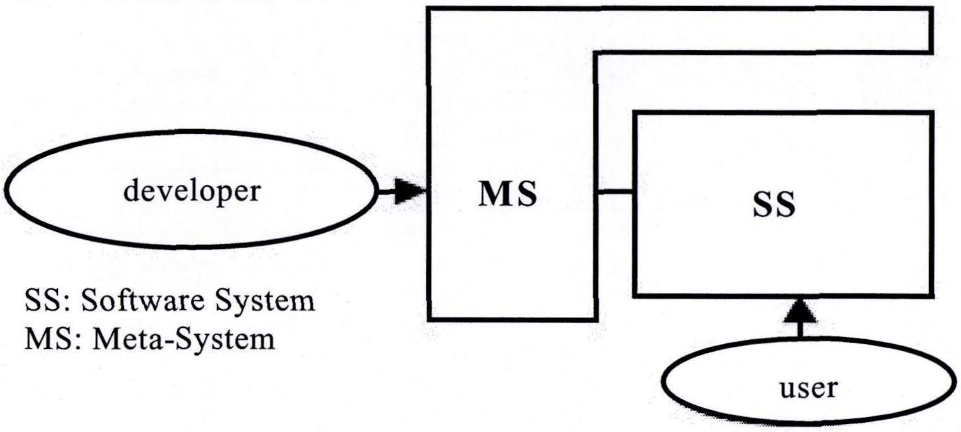


Figure 1a: Two-level architecture.

2.3 Mechanisms of Evolution

The evolution mechanisms represent different ways used by the software systems to change. Each mechanism includes a group of activities that produces the change through its coordinated execution. On the previous architecture, Torres (Torres and Parets, 1999) proposes two types of mechanisms: Adaptation and Inheritance.

- **Adaptation by Accommodation/Learning:** The systems adapt to their environment learning the best way to use their structure without changing it. This type of adaptation is carried out in functional environments where the user communicates with the software system using actions from its interface and perceiving the adaptation as changes in its responses.

- **Adaptation by Mutation/Differentiation:** This type of adaptation is more radical than the previous one because it implies changes in the structure of the system that causes changes in its functional character, but they also introduce new possibilities of adaptation by accommodation/learning. This modification type needs the intervention of the meta-system.
- **Inheritance:** The inheritance mechanism is used in the generation of descendants of software systems, which inherit the adaptations of its parents. The new system inherits the initial structure and the changes of the two types of adaptation seen previously.

2.4 Models of Evolution

An evolution model is a symbolic representation of the particular meaning of the effect of the changes in a software system. Six models are proposed in (Torres and Parets, 1999), each of which makes use of one of the described evolution mechanisms (table 1).

Table 1: Evolution Mechanisms and Models

Mechanisms of Evolution	Model of Evolution
Adaptation by mutation/differentiation	Meta-Teleology
	Teleology
	Auto-adaptation metasystem - system
Inheritance	Inheritance of the meta-characteristics
	Inheritance of the characteristics
Adaptation by accommodation/learning	Auto-adaptation of the system

1. **Meta-Teleology** directed by the modeler: The modeler carries out modifications in the structure and operation of the meta-system using actions of its evolution interface. This model applies the mechanism of adaptation by mutation/differentiation.
2. **Teleology** directed by the modeler: Changes in the structure and functioning of the system are produced by the modeler using actions of the action interface of the meta-system. The mechanism of adaptation is applied by mutation/differentiation.
3. **Inheritance of the** acquired **meta-characteristics:** If the modeler decides so, a new meta-system can be generated from the old meta-system. The obtained meta-system inherits the meta-characteristics (structural and functional) acquired by the father during its evolution. The inheritance mechanism is applied.

4. **Inheritance of the** acquired **characteristics:** This model is used by the modeler when it wants to create a new software system from an existing software system. This model inherits the new characteristics (structural and functional) of the old system acquired during its evolution. The inheritance mechanism is used.
5. **Auto-adaptation meta-system - software system:** The meta-system carries out structural and functional changes in the system without the modeler's direct intervention. This model applies mechanisms of adaptation for mutation/differentiation.
6. **Auto-adaptation of the system:** The software system develops an adaptive process with neither the modeler nor the meta-system participates actively in it. It applies the mechanism of adaptation for accommodation/learning to only carry out changes of functional type.

In figure 1b, the six evolution models are superimposed on the structure of interaction of figure 1a. Each arrow has associated the number of the represented evolution model.

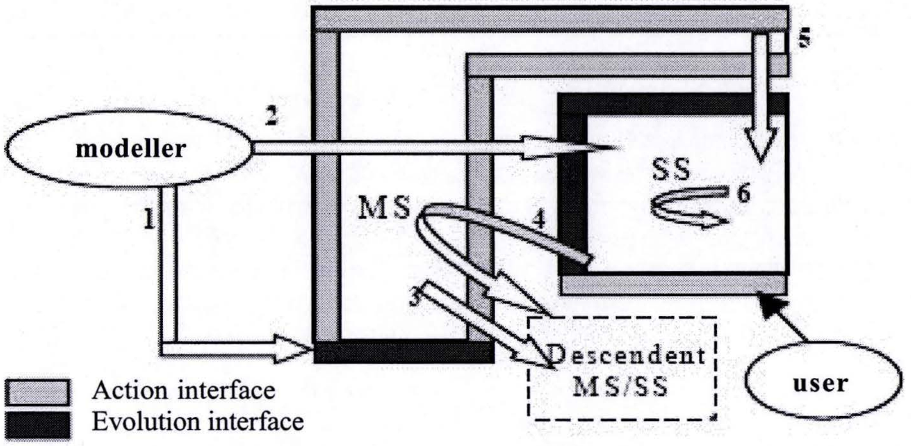


Figure 1b: Models of evolution.

2.5 Parameterizables Systems

Parameterizable systems are considered to be those that establish in advance a series of variables that allow the adaptation or modification of the system. For example, two types of systems that frequently follow this adaptation philosophy are: Adaptive Commercial Systems (SAP, 2007) and Content Management Systems (CMS) whose objective is to organize and structure contents by a wide community of users (Boiko, 2004).

The achievement of a good parameterization is a decisive step in the aspiration of achieving a better adaptation of the software system in different environmental conditions. The parameters constitute options of configuration that the system provides to the user without direct intervention of the developer or the meta-system. With time, most of these systems transform the parameters in metadata serving as a starting point for a more complete evolution.

3. Practical Application in the Nickel Factory

The "Comandante Ernesto Che Guevara" Nickel Factory has had an Intranet since 2002: "Intranet 8 Hours" whose name reflects duration of the work shifts in the company. However, at the end of 2004, technological changes that occurred in the production process implied modifications in the system of shifts that altered the sampling system, which had repercussions in the control of the mining and metallurgical processes. As a result, the system of process control that existed until then did not reflect the true operative and metallurgical state of the factory. This situation gave way to the following thought: How to facilitate the adaptation and/or evolution of the "Process Management System" in the face of changes similar to the previously mentioned changes? The solution that we found consisted on the application of the evolution model "Teleology directed by the modeler" along with parameterization techniques.

3.1 CheNET Corporate Portal

CheNET (Menéndez, 2006) is an open corporate portal, with support for distributed content that is supported by the integration and corporate collaboration of the different systems that compose it (figure 2). It is based on roles with facilities to personalize and configure the systems or applications, roles, users and access permissions. The portal is comprised of a public area where, among other services, it offers information about the nickel factory, and a restricted area that contains the main systems in charge of the management of processes and supplies, and the administration and security system.

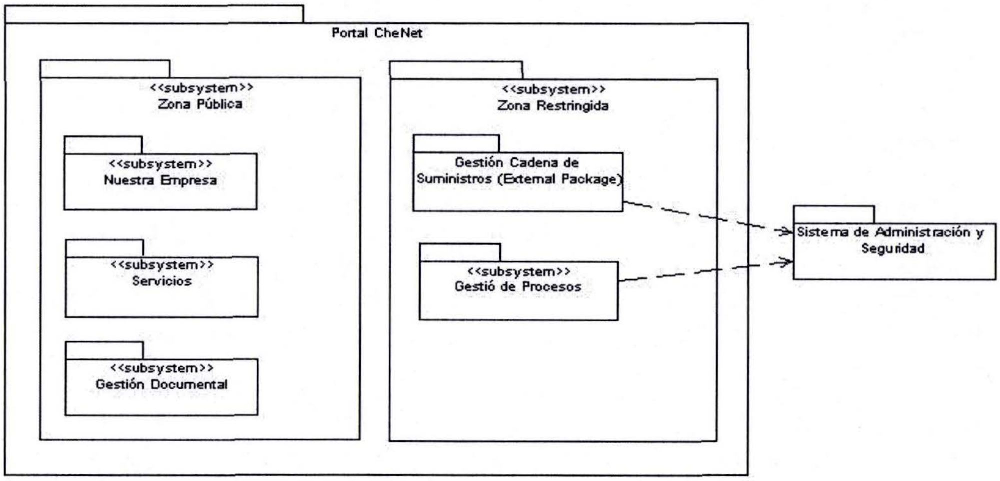


Figure 2: Portal CheNET: Public and private area (main systems).

It is a distributed web portal, which is an advantage for the process of upgrading to new versions in the client stations. In short, the portal presents a client/server architecture in four layers: Presentation (User interface), Application (Business logic), Data Access (Communication interface that separates the functionality of the system from the database system used) and Databases (Data of the business). This separation allows the logical grouping of the elements according to their features and interrelations so that if it becomes necessary to carry out modifications in the system they are only carried out in the layer or layers that contain the affected elements. Except for the layer of Access to Data, the other layers can contain structural and/or functional components of the meta-system. For example, in the database layer we can find structural elements of the meta-system like the meta-class "Muestra" where also functional components of the meta-system appear, implemented through insertion triggers in the corresponding data table (figure 6).

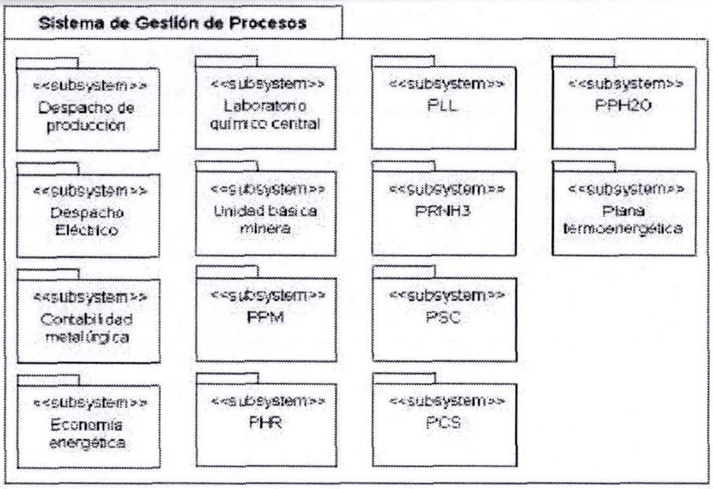

Sistema de Gestión de Procesos
<<subsystem>> Despacho de producción
<<subsystem>> Laboratorio químico central
<<subsystem>> PLL
<<subsystem>> PPH2O
<<subsystem>> Despacho Eléctrico
<<subsystem>> Unidad básica minera
<<subsystem>> PRNH3
<<subsystem>> Planta termoenergética
<<subsystem>> Contabilidad metalúrgica
<<subsystem>> PPM
<<subsystem>> PSC
<<subsystem>> Economía energética
<<subsystem>> PHR
<<subsystem>> PCS

Figure 3: Portal CheNET: Process Management System.

Process Management is an important system in the portal because it allows the management of all the information related to the production process. The system is comprised of 14 subsystems or modules that are shown in figure 3: four management and/or control subsystems (Production Dispatch, Electric Dispatch, Metallurgical Accounting, and Energy Economy), six production plants (Mineral Preparation Plant, Reduction Ovens Plant, etc.), a basic mining unit, a central chemical laboratory and two auxiliary plants (Power Station and Water Purifying).

3.2 Evolution and Adaptation in the Process Management System

The technological process of the factory is characterized by a great number of chemical analyses that are carried out on a high quantity of samples in the different stages of the process. Some of the things that are determined for each sample are: the plant where the sample is taken, the sampling type that is carried out, the chemical analyses to carry out and their frequency. The identification of the elements that change more frequently in the processes carried out is decisive in order to apply evolution techniques on the software system that controls them. The samples and their characteristics (sampling frequency, chemical analysis, etc.), the indicators of the plants and the indicators of the technical devices were identified as unstable in the productive flow of the factory.

Figure 4 shows a fragment of the Process Management Class Diagram, where we can see part of the structure that allows the above mentioned parameterization (Frequency, Plant, Sample, SampleC2, SampleC2_43, Parameters of Analysis, Generic Group, Type of Sampling, Specific Sample C61 ad Specific Sample C61_335). Since the equipment where the chemical analysis are carried out provide several results, called analysis parameters, the generic groups (figures 5 and 6) contain the group of analysis parameters that are obtained in one given moment in time. The generic goups improve the efficiency in the entry or modification of data by the technicians in charge of carrying out this task. In addition, the metaclass "Muestra" relates and describes all its instances, for example, the samples "MuestraC2" and "MuestraC61PUNTUAL" that are also classes.

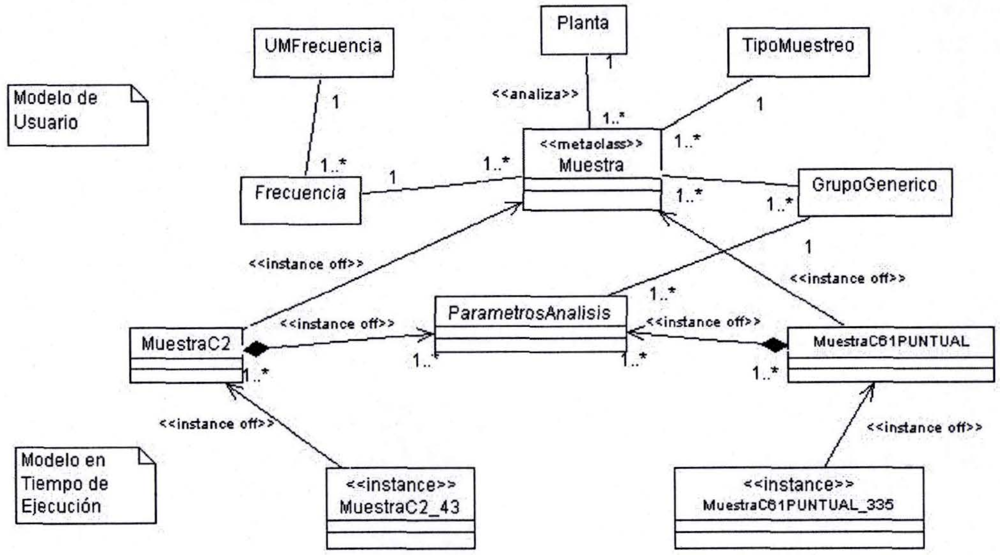


Figure 4: Process Management System class diagram fragment.

The control (component) used to enter the results of the chemical analysis carried out on a sample is shown in figure 5. It is divided in two areas: the selection of filters (frequency, plant, generic group and sample) and the entry of information that adapts to the selected filters. The analysis parameters that the technician should fill out are grouped according to their generic groups (Humidity, Granulometry, etc.).

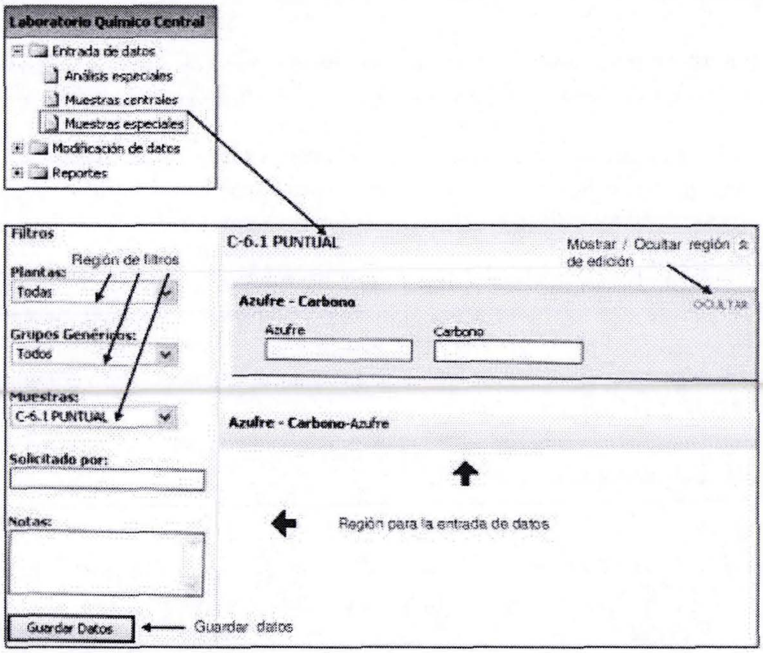
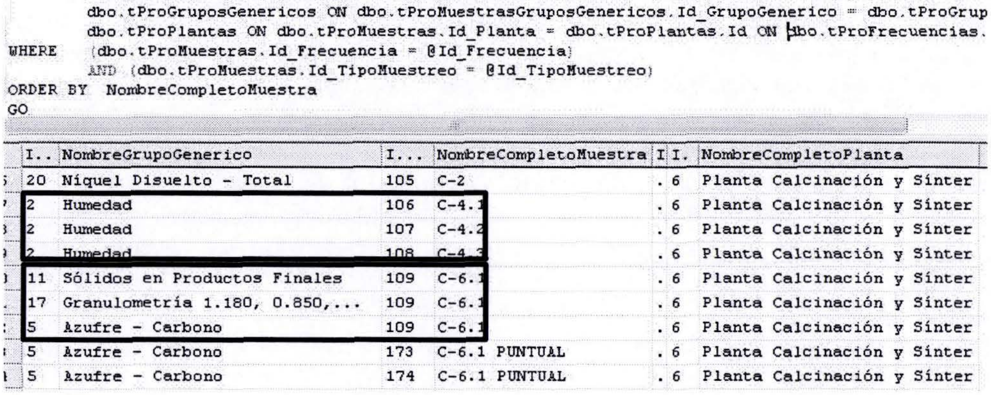


Figure 5: Central chemical laboratory: Central samples (data entry mode).



```
CREATE PROCEDURE spProMuestrasTodoPorFrecuenciaPFiltro
@Id_Frecuencia int, @Id_TipoMuestreo int
AS
SELECT  dbo.tProGruposGenericos.Id AS Id_GrupoGenerico,
        dbo.tProGruposGenericos.GrupoGenerico AS NombreGrupoGenerico,
        dbo.tProMuestras.Id AS Id_Muestra,
        dbo.tProMuestras.NombreCompleto AS NombreCompletoMuestra,
        dbo.tProMuestras.Id AS Id_Frecuencia,dbo.tProPlantas.Id AS Id_Planta,
        dbo.tProPlantas.DescPlanta AS NombreCompletoPlanta
FROM    dbo.tProFrecuencias INNER JOIN dbo.tProMuestras INNER JOIN
        dbo.tProMuestrasGruposGenericos ON dbo.tProMuestras.Id = dbo.tProMuestrasGruposGenericos.I
        dbo.tProGruposGenericos ON dbo.tProMuestrasGruposGenericos.Id_GrupoGenerico = dbo.tProGrup
        dbo.tProPlantas ON dbo.tProMuestras.Id_Planta = dbo.tProPlantas.Id ON dbo.tProFrecuencias.
WHERE   (dbo.tProMuestras.Id_Frecuencia = @Id_Frecuencia)
        AND (dbo.tProMuestras.Id_TipoMuestreo = @Id_TipoMuestreo)
ORDER BY  NombreCompletoMuestra
GO
```

```
I.. NombreGrupoGenerico              I...  NombreCompletoMuestra I I. NombreCompletoPlanta
20  Níquel Disuelto - Total          105   C-2                   . 6  Planta Calcinación y Sínter
2   Humedad                          106   C-4.1                 . 6  Planta Calcinación y Sínter
2   Humedad                          107   C-4.2                 . 6  Planta Calcinación y Sínter
2   Humedad                          108   C-4.3                 . 6  Planta Calcinación y Sínter
11  Sólidos en Productos Finales     109   C-6.1                 . 6  Planta Calcinación y Sínter
17  Granulometría 1.180, 0.850,...   109   C-6.1                 . 6  Planta Calcinación y Sínter
5   Azufre - Carbono                 109   C-6.1                 . 6  Planta Calcinación y Sínter
5   Azufre - Carbono                 173   C-6.1 PUNTUAL         . 6  Planta Calcinación y Sínter
5   Azufre - Carbono                 174   C-6.1 PUNTUAL         . 6  Planta Calcinación y Sínter
```

Figure 6: Fragment of a meta-procedure: Functional component of the meta-system in the database layer.

The main improvement added from an evolutionary point of view is that making use of the double level of abstraction proposed in the section 2.2, the control adapts at all times to the structure of the system described in the meta-system. If a parameter of the sample changes, for example, the concentrated chemical analyses in a generic group, the control automatically detects it by means of meta-procedures and updates the data-entry area. The meta-procedures (figure 6) are methods stored in the Database layer that return the meta-characteristics of the different classes (MuestraC2, MuestraC61PUNTUAL, etc.) whose meta-class is the class "Muestra". Figure 6 also shows some meta-characteristics of some samples and how a generic group (Humidity) can be used in different samples (C-4.1, C-4.2 and C-4.3) and how a sample (C-6.1) can also contain several generic groups ("Solids in final products", "Granulometry 1.180, 0.850…" and "Sulfur - Carbon").
Two evolution examples permitted to the developers and super-users of the system are described in detail in sections 3.2.1 y 3.2.2.

3.2.1 Example 1: To Modify the Frequency

In order to modify the frequency of sampling of the sample "MuestraC61PUNTUAL" (for example, from 2 to 4 hours) it is only necessary to modify the value of this property in the instance in the class "Muestra". The presence of a one to many (1 - 1.. *) relationship between the meta-class "Muestra" and the class "Frecuencia" (figure 4) it allows that each instance of the first class (each sample type) corresponds with only one of the instances of second class (a concrete frequency). Similarly, this change is done for the sampling type or the plant where the sample is carried out, among other parameters.

3.2.2 Example 2: To Add a New Central Sample

There are three types of samples: special, central and special analysis. To add a new central sample (figure 5), it is necessary to create a new instance of the class "Muestra". Let us imagine, for example, that a new sample SM9 that will be taken in the Mineral Drying Plant with a frequency of sampling of 6 hours is added. As a result, an event goes off that creates a new class "MuestraSM9" with the indicated characteristics and whose properties by default will be Id (identifier of the instances of this class), DateTime (time when the values are inserted) and AnalysisParameter1 (at least one analysis parameter for the sample is assumed). The developer or super-user should rename the AnalysisParameter1 (like Sulfur or Carbon in Figure 5) and link it with a specific generic group. As many additional analysis parameters as necessary can be created. All the added parameters are recorded as instances of the class "ParametrosAnalisis".

At the moment, these facilities are not included in the Presentation layer. However, they are carried out in a more rudimentary interface, although its inclusion in the web interface of the portal is foreseen in the immediate future. It has not been carried out for commercial reasons and not because of technical causes. Thanks to the previously defined mechanisms the structure can be changed without reprogramming the system.

4. Conclusions

A software engineering process based on a double level evolutionary architecture permits the anticipation of modification types that the software system can undergo and make them effective by means of evolutionary actions that guarantee the consistency of the change through the confirmation of restrictions and its automatic propagation. The representation of the different ways in which the software system changes by means of formal models of evolution makes the resulting system more robust and flexible. Finally, the parameterization of the system permits to foresee a group of variables that are especially susceptible to the change, establishing this way concrete evolution points.

The evolutionary model implanted in the Process Management System of the Cuban Nickel Factory "Comandante Ernesto Che Guevara" can be summarized as follows:

a) Study of the parameterizable variables, foreseeing the aspects of the productive process that can be necessary to modify during the system lifetime.
b) Definition of a meta-level to formalize the changes in the above mentioned variables.
c) Application of the evolutionary model teleology directed by the modeler based on a) and b) that permits the modification of the structure of the system by means of direct intervention of the developer who requests the changes without having to reprogram the system. This model, as described in the section 2.3, applies the mechanism Adaptation for mutation/differentiation that in turn implies new possibilities of Adaptation for accommodation/learning.

In a practical sense, from the developer's perspective all the above-mentioned allows changes to be carried out in a more flexible, quick and consistent way, by foreseeing them by means of the previously mentioned evolutionary mechanisms. From the users' viewpoint, the software reflects at all times the true reality of the enterprise.

Acknowledgements
This research is supported by ADACO project TIN2004-08000-C03-02 (CICYT, Government of Spain).

References

Boiko, B. (2004) *Content Management Bible*, 2nd Edition. ISBN: 978-0-7645-7371-2.

Le Moigne, J.L. (1990a) *La théorie du système général. Théorie de la modélisation.* PARIS. Presses Universitaires de France, ISBN: 2-13-043323-5.

Le Moigne, J.L. (1990b) *La modélisation des systèmes complexes*. PARIS. Presses Universitaires de France, ISBN : 9782100043828.

Lehman, M. Ramil, J.F. and Kahen, G. (2001a) *Thoughts on the Role of Formalisms in Studying Software Evolution*. Proceedings of Formal Foundations of Software Evolution, Lisbon, Portugal, 8 pps, in Mens T. and Wermelinger M (eds.) Tech, Report UNL-DI-1-2001.

Lehman, M. and Ramil, J.F. (2001b) *An Approach to a Theory of Software Evolution*. Position paper, Proceedings of 4th International Workshop on Principles of Software Evolution (IWPSE'01), Vienna, Austria, ISBN: 1-58113-508-4, Pp.: 70-74.

Madhavji, N. Fernández-Ramil, J.C. and Perry, D. (2006) *Software Evolution and Feedback: Theory and Practice*. John Wiley & Sons, Ltd. ISBN: 0-470-87180-6.

Medina, N. García, L. Torres, J.J. and Parets, J. (2002) *Evolution in Adaptive Hypermedia Systems*. Proceedings of International Workshop on Principles of Software Evolution (IWPSE'02), Orlando, Florida, ISBN: 1-58113-545-9, Pp.: 34-38.

Menéndez, Raúl. (2006) *CheNET: Corporate Portal of the "Comandante Ernesto Che Guevara" Nickel Factory*. Master thesis.

Parets, J. (1995) *Reflections on the Process of Conception of Complex Systems. MEDES: A method of specification, development and evolution of software system*. Ph.D. thesis.

Parets, J. and Torres, J. (1996) *Software Maintenance versus Software Evolution: An Approach to Software Systems Evolution*. Proceedings of IEEE Symposium and Workshop on Engineering of Computer Based Systems (ECBS'96), Friedrichshafen, Germany, Pp.: 134-141.

SAP Learning Solutions. (2007). http://www.sap.com

Torres, J. and Parets, J. (1999) *A Formalization of the Evolution of Software Systems*. Proceedings of International Conference on Computer Aided Systems Theory (EUROCAST'99), Vienna, Austria, Lecture Notes in Computer Science 1798, ISBN 3-540-67822-0, Pp.: 269-272.