HIGAIA METHODOLOGY

A. Anguera, A. Gutierrez, M.A. Diaz

Abstract: *At present there is a deficiency in the field of scientific theories that support software development. On the other hand, the few existing scientific theories do not provide methodological support for all phases in software development. It is necessary to combine both aspects and develop a methodology, supported by a scientific theory, which extends these methodological support to all phases of software life cycle.*

The proposed Software Development methodology combines Holons and Informons theory with GAIA, a well known methodology in the field of MultiaAgent Systems (MAS).

The elements defined in scientific theory are used in the description of the software development phases included in GAIA, extending them to complete the software life cycle.

Analysis, Architectural Design and Detailed Design phases of GAIA have been completed with Requirements Elicitation and Implementation phases, the latter based on the AUML standard.

In this way we obtain a complete methodology supported by a scientific theory that allows develop software systems based on Holonic Integrated Systems (HIS).

Keywords: *HIGAIA,HIS, HIP, Holon, Models, MAS, Software Development Methods.*

ACM **Classification Keywords**: *C.2.4 Dsitributed Systems – Distributed applications, D.2.11 Software Architectures-Languages*

Introduction

There are several schools of thought on the current state of scientific theories that support software development. All of them describe a slightly optimistic panorama.

Some authors understand that while the scientific theory is not developed, this is not the same with the engineering component, which itself is developed but in many cases without the required scientific theoretical support. The solution might be to address the software development taking into account that is treated as an engineering discipline but in any case must be supported by an underlying scientific theory.

There is a emerging scientific theory proposal to support the software development made by Alonso [Alonso 2004] where are introduced the holon and informon concepts as basic elements of this theory. Later, Martinez [Martínez, 2008] develops these concepts in the theory of holons and informons for software development , which is used by Suárez [Suárez, 2009] to formulate a specific proposal on the scientific theory on computational and conceptual model of software development.

According to [Suárez, 2009], the basic element of information that makes sense for a holon and allows to make decisions and implement appropriate actions is called "informon". The informon can take the form of facts, news or knowledge.

On the other hand, a "holon" can be generically defined as an element in its own right which has an autonomous, cooperative and self-organized behavior.

Autonomy means the ability of an entity to create their own behavior, that is, create their own plans and behavior strategies and monitoring execution as well as their own state.

Cooperation in this context is a process by which a set of entities develops plans, commonly accepted, that run in a distributed manner. Finally, it should be noted that one of the main characteristics of holons is their self-replication, which provides the ability of recursion and self-organization.

In building software terms, a holonic computational model consists of different levels [Martinez 2008], as detailed below:

- *Instruction*: Primary holons that are their own and cooperatives entities. They deal only with data and produce new data or simple news. They are specialized and perform basic operations.
- *Component*: The result of a holarctic structure (hierarchical/heterarchical) of primary level "instruction” holons. The holon component has a functionality greater than the sum of "instruction" holons and has capacity to produce more elaborate news and/or knowledge.
- *Entity*: Holarctic relationships formed by holon components. An “entity” holon presents beliefs, motivations and intentions and change their behavior based on previous experience. Incorporates proactive property, so it is able to act on their own initiative to achieve goals that itself is able to generate. An “entity” holon deals with news and knowledge informons. It can act as a software agent.
- *Organization*: A holonic organization is a collaborative holarchy entity. An “organization” holon means a formal and stable group of entity holons and provides a well defined interface for external communication. The activities of each organization are determined by cooperation processes with other entities or organizations. Presents a common goal and the entity holons, which have their their own goals, have a cooperative behavior led to organization common goal. An organization holon is similar but not equal, to a MultiAgent System (MAS).
- *Domain*: Is the logical space in which entity or organization holons, each with their own goals within the domain, operate and communicate with each other. Providing the context in which these holons may locate, contact and interact with others. A domain can be simple, anyone committed to working with any other following a set of established interaction protocols; or complex, designed to achieve a goal shared by several holons. In the domain, the holons are viewed as members of a society whose global activities should be contemplated. The social structure of the domain is determined by roles, social relationships identified between holons and social laws that govern these relationships. Holons are incorporated in the domain to achieve their own goals, and accept or reject their entry depending on which role they used to participate, if appropriate for this domain and accept conventions and social laws.

Software engineering has gone through different stages marked by the programming paradigms. Since the structured approach or object-oriented approach to the Unified Process. Today there are great challenges to these standards resulting from new technologies such as distributed Web services and agents. Agents are computational entities that differ from objects in several respects, such us autonomy, social ability and proactivity among others [Wooldridge, 1995], [Franklin, 1996], [Hernandez, 1999]. All these properties defined for the agents, and more, can also be attributed to the holon element defined in the holon and informons theory.

The presented scientific theory provides the basic tools for modeling computer systems being a formal support for the development of such models, but is in early maturity stage and does not include some software life cycle phases needed to complete development of such systems. This paper presents a methodology having its roots in the abovementioned theory approach to the software design phase. Taking as its starting point the main element of this theory, the holon, to make a comparison between an entity holon organization and a software agent. It can be inferred that they share more features than differ. If we accept this assertion and focus on a more complex relationship form, holons in Organizations and agents in MultiAgent Systems (MAS) found that both forms of organization share many similarities. An holon organization may seem like a MultiAgent System (MAS), although they are not alike.

HIGAIA

The general diagram that defines the phases of the HIGAIA methodology builds on the general model proposed in GAIA [Wooldridge, 2000], [Zambonelly, 2000] within the field of development methodologies Multiagent Systems (MAS) [Henderson, 2005]. Developing and adapting the methodology to the elements of the holons and informons theory for software development. The following figure represents the GAIA phases taken as a starting point.

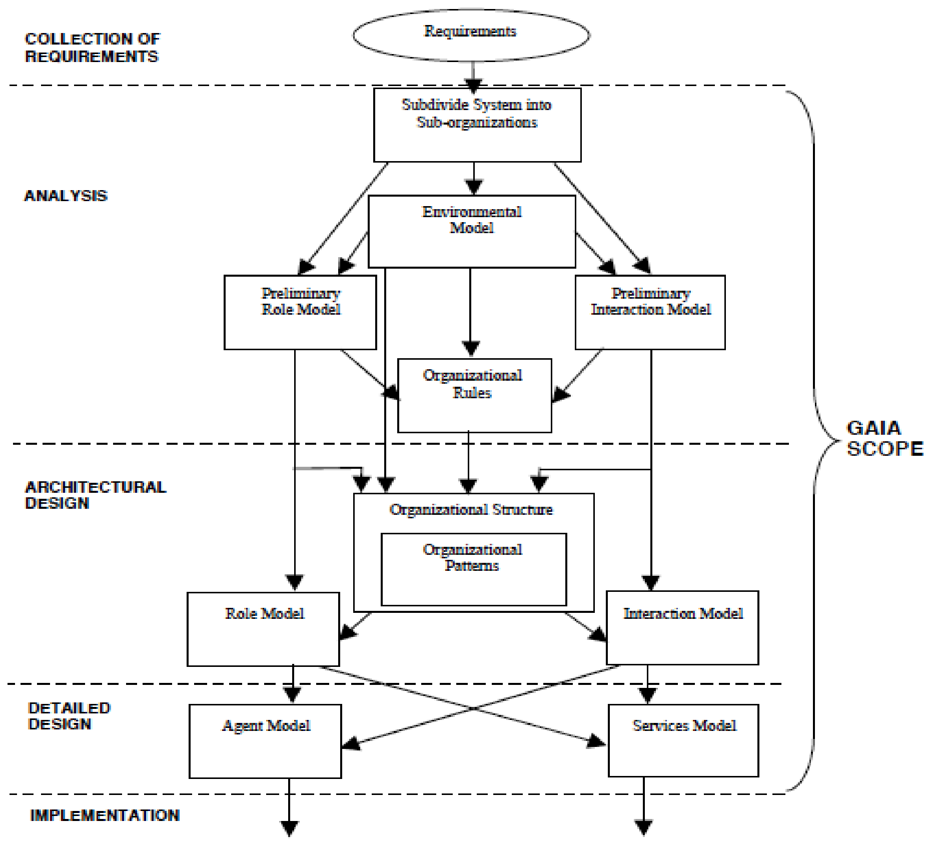

Figure 1: Phases of software develop.

As shown in Figure 1, the model consists of five phases outlined below.

1. *System Requirements elicitation phase*: The goal of this phase is to obtain a detailed specification of the information system that meets the users information needs and serve for further analysis and system design.

2. *Analysis phase*: The analysis stage output systematically documented all the functionalities and features that the system has to express, together with the characteristics of the operational environment in which it is located. Therefore, the main objective of the analysis phase is to organize the specifications and system requirements to become an environmental model. The preliminary roles, interaction patterns, and a set of rules of organization, for each of the sub-organizations that make up the system in general. This phase is divided in turn into five phases.

   2.1. Determination of the organizations. The first phase of HIGAIA analysis deals with the identification of multiple organizations that exist in the system and interact to become an Integrated Holonic System (IHS). The identification of these organizations is reasonably easy if (i) system identifies naturally or (ii) the system mirrors the structure of the real world, and it involves multiple agencies that interact. An organization consists of a set of related roles, these roles will be assigned to holons depending on skills and abilities of the latter. A set of holons can be grouped into sub-organizations, each of which will have its own goals and cooperate to achieve the overall goal of the organization.

   2.2. Environment Model: A IHS is always in an environment and we believe this should be considered as a primary abstraction for the analysis and design phases. In general terms, identify and model the environment is to identify all entities and the resources the IHS can exploit, control or consume when you are working toward the goal of the organization. This model consists of a list of abstract computational resources that can be drawn at this stage of design. Collect the system inputs, outputs and resources associated with sub-organizations identified in the previous phase. It should include active components that interact with the organization.

   2.3. Preliminary model of roles: The analysis phase does not result in the design of the current SHI organization (this is the purpose of the subsequent architectural design phase). However, even without knowing what the structure of the organization will, it is possible, in most cases, be able to identify some characteristics of the system that remains independently the final organizational structure. This phase detects the basic skills (preliminary roles) that organization requires to achieve its goals. To represent these preliminary roles uses a semiformal and abstract description to describe their capabilities and expected behaviors. Using two main class features respectively, permissions and responsibilities. The latter are subdivided into two types of properties, liveness and safety. The atomic components of an vitality expression are protocols and activities. Respectively are short-term actions that require interaction with other roles and can be done without third-party roles.

   2.4. Preliminary interaction model: Define the interactions by a reference to an institutionalized model of exchange of messages, such as FIPA (The Foundation for Intelligent Physical Agents), FIPA-Request [FIPA 1964]. This model captures the dependencies and relationships between the different organization roles, specifying the characteristics of the protocols concerned. Interaction organization models describe the protocols that govern the interactions between roles. Moreover, the interaction model describes the characteristics and dynamics of each protocol (for example, when, how and by whom a protocol should be executed). As the role model is still preliminary at this stage, the protocols for this model must necessarily be preliminary, for the same reasons.

2.5. Organizational rules: preliminary functions and interaction models capture the basic features, functions, and patterns of interaction that must realize in the MAS, irrespective of any pre-defined organizational structure. But at this stage can be captured functions and protocols to the organization. Those the rules that control the overall relations between the different roles of the organization and interactions between different protocols. Among others, the organizational rules establish the order of execution of the roles, time constraints in the execution of the same role for different entities and the maximum number of times you can run a role.

3. *Design phase*: *Architectural Design*. The specifications provided in the analysis phase should be structured and used in architectural design to identify an efficient and reliable SHI's organizational structure, and thus complete the preliminary tasks and interaction models. It is worth emphasizing that although the analysis phase is primarily concerned with understanding what the SHI will be, the design phase is where decisions are made to bear on the actual characteristics of SHI. At this stage, there are three different phases:

3.1. Organizations structure: The choice of the organizational structure is a very critical stage in the development of SHI, which affects all subsequent stages. Unfortunately, as always with the architectural design, it is not possible to identify a precise and formal methodology with which to obtain the "best" design. However, a design methodology can and should provide guidelines to help designers make decisions. In this case seeks to achieve maximum simplicity and organizational efficiency. As a general rule, choose the simplest topology that enables the organization to properly handle both the computational complexity and the complexity of coordination between different actors. Another factor that impacts the choice of organizational structure is the need for the SHI should respect rules of organization and be able to adopt them for implementation. We analyze each of the organizations that have been described in the analysis phase, as well as control schemes adopted (dependence, equality, or control) between the roles that are part of each, in order to maximize efficiency and simplicity of the system. If you increase significantly the number of members of different organizations, would opt for a multilevel hierarchical system in which some roles may partially control the actions of other roles.

3.2. Ultimate role model, complete the set of roles defined in the initial roles model with different organizational structures designed in step 3.1.

3.3. Final interaction model: the incorporation of new roles in the previous phase, implies a redefinition of the initial interaction model.

4. *Design Phase: Detailed Design*. This phase is responsible for the eventual identification of the type of holons and service model that, in turn, act as guidelines for actual implementation of the agents and their activities. The objective of this phase is to define two models:

4.1. Holons model. HIGAIA context distinguish between holon and holonic organization. A holon is a software entity that runs a set of roles. Therefore, the definition of holon model will determine what kinds of holon are defined to perform specific roles and how many instances of each of them must be implemented in the real system. Normally there will be a one-to-one relationship between the roles and holons so that a given role is taken by a holon responsible for implementing that role. The adoption of a role for a particular holon depends on the skills of the holon that can deliver the goals that represent the role.

An holonic organization shall consist of holons which are in turn dependent on the acceptance of social rules of the organization and should develop cooperative behavior led to the common goal. This model is described by one side each of the holons which are part of the system using Table 2 and the other holonic organizations that conform the system as seen in Table 1.

holonic organization class		Organization identifier.
Servicces	Name	Organization name.
	Goals (description)	Organization objective.
	Skills (description)	Holons capabilities as part of the organization to realize its goals
	preconditions	Conditions to be satisfied in the system just before the start of the execution of any element that is part of the organization.
	postconditions	Declare what should be true in the system when is to successfully complete the runnig of the organization.
	input	External ítems that the organization needs to running its behavior.
	output	items to get de organizations after running the organization behavior.
Strategies		Descriptions of algorithms or procedures that represent the behavior of the organization to reaching its goals.
Set of Roles		Roles that are part of the organization at all times.

Table 1: Clase Organización Holónica

Holon class		Holón identifier.
Services	Name	Holon name.
	Goals (description)	Holón objetive.
	Skills (description)	Holon capabilities to reach their goals. A skill is always linked to a goal.
	preconditions	Conditions to be satisfied in the system just before the start of the running of the holon
	postconditions	Declare what should be true in the system when is to successfully complete the running of the holon.
	input	Elements takes to run your behavior holon
	output	items to get the holon after running the holon behavior
Strategies		Strategies
Set of Roles		Set of Roles

Table 2: Holón Class

4.2. Service model. This model allows us to identify the services associated with each holon in the system, or equivalent, with each of the roles that run the holon. Thus, the service model applies in the case of static allocation of functions to holons as well as in the case of functions they can assume dynamically. Must be defined for each of the holons that form part of the system (as defined in the previous phase, 4.1), indicating the inputs and outputs, as well as necessary preconditions or postconditions needed to perform the role assigned to holon. The services that compose a holon are derived from the list of protocols, activities, responsibilities and liveliness properties of the functions it implements. At one extreme, there is at least one service for each parallel activity that holon has to execute. However, even for sequential execution activities, there may be a need to introduce more services to represent the different stages of holon implementation.

5. *Implementation phase*. GAIA notations, probably due to their simplicity, are poor and far from being widely accepted as industry solutions (unlike UML in object oriented software engineering). This seems to be quite clear in the specification of the interactions between holons within HIGAIA methodology. In fact, the Gaia protocol model, despite rigorously specify the actors and the inputs and outputs of the protocol, use informal natural language to specify the semantics, including dependence and interaction of the actors involved. AUML is not itself a complete and comprehensive methodology, however, is based on the acknowledged success of UML to support software engineering for industrial processes. The AUML core are

interaction protocols between agents (AIP). The proposed methodology use an adaptation of AUML AIP interaction model to capture the interaction between holons (HIP- Holons Interaction Protocol). Will, therefore, the central element of the specification of the HIP, completing the present model in GAIA.

The following figure shows a generic class hierarchy, which can be used to implement any holonic system (SHI). As can be seen are established the essential elements of the organizational model as well as the relationships established between each of them.

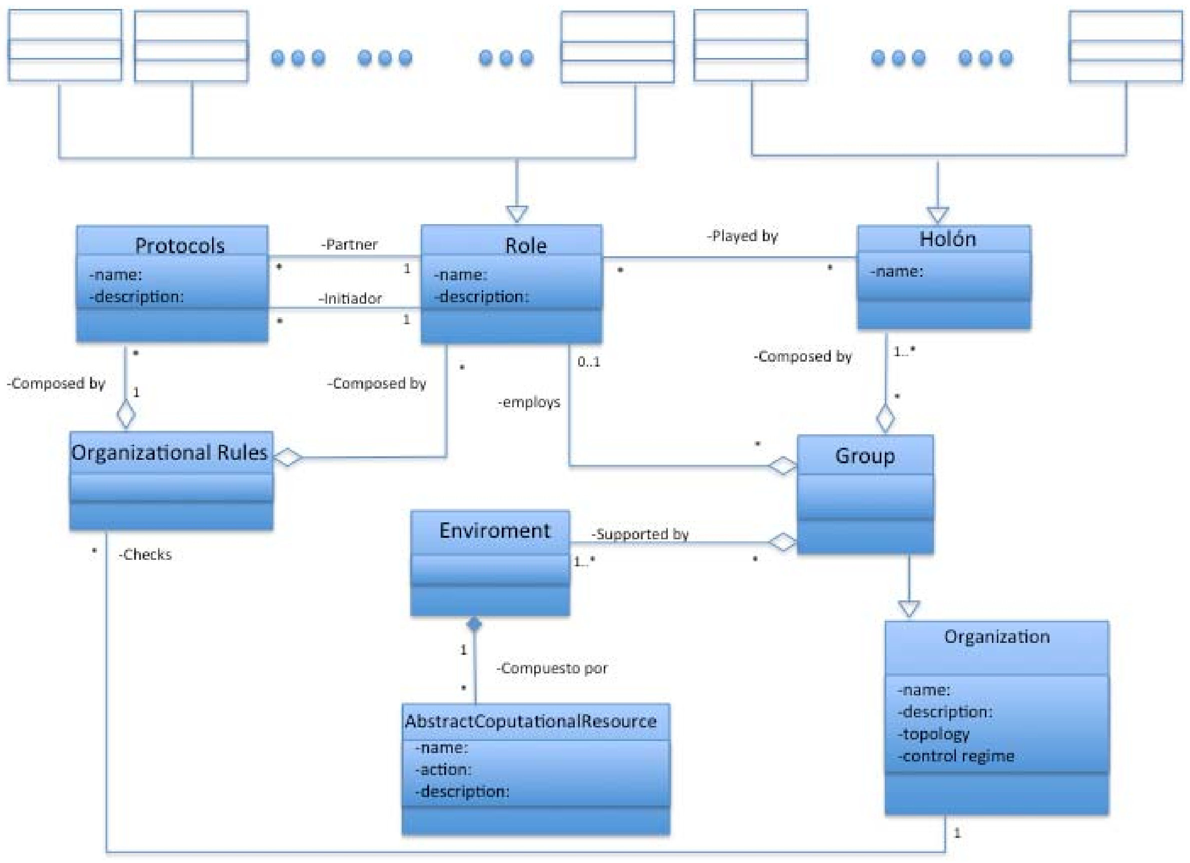

Figure 2: Generic hierarchical class model

Conclusions

This paper develops a methodology based on the scientific theory of holons and informons that unlike existing theories covers all phases of software development. Completing the scientific theories with development methodologies. Achieving a balance between the two sides of software, the science base and its development as seen from the point of software engineering.

HIGAIA methodology is proposed in order to be applied in any field where multi-agent systems (MAS) can be used. HIGAIA is a versatile methodology that allows easy application and adjustment to the software development industry. Finally, the HIGAIA methodology is independent of programming language chosen to implement the system.

Bibliography

[Alonso, 2004] Alonso F., López G., Pazos J., Rodríguez-Patón A., Silva A., Soriano F.J., Fundamental Elements of a Software Design and Construction Theory: Informons and Holons, Proceedings of the International Symposium of Santa Caterina on Challenges in the Internet and Interdisciplinary Research (SSCCII), Italia, 2004, pp. 21-35.

[FIPA, 1964] FIPA, 1964 The FIPA Specification Repository, http://www.fipa.org/repository/index.html Fletcher, R., & C.M. Reeves, "Functions minimization by conjugate gradients", Computer Journal, vol. 7, pp. 149-154, 1964

[Franklin 1996] Franklin, S., Graesser, A.: Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages. Springer-Verlag (1996).

[Henderson, 2005]. Brian Henderson-Sellers, P.G.Agent-Oriented Methodologies, 2005, Idea Group.

[Martínez, 2008] Martínez, M.A.: Una Teoría para el Desarrollo Software construída mediante técnicas y modelos de Gestión del Conocimiento. Tesis Doctoral. Universidade da Coruña. Mayo 2008.

[Suárez, 2009] Suárez, S.: La experimentación Crucial Basada en una Teoría de Holones e Informones para el Desarrollo del Software. Tesis Doctoral. Universidade da Coruña. Junio 2009.

[Wooldridge 1995] M. Wooldridge and N. R. Jennings, editors. Intelligent Agents --- Theories, Architectures, and Languages. Volume 890 of Lecture Notes in Artificial Intelligence, Springer-Verlag, January 1995.

[Wooldridge et al. 00] Wooldridge, M., Jennings, N. R., and Kinny, D., The Gaia Methodology for Agent-Oriented Analysis and Design, Journal of Autonomous Agents and Multi-Agent Systems, vol. 15 2000.

[Zambonelly, 2000] Zambonelly, F., Wooldridge M., and Jennings N.R., Organisational Rules as an Abstraction for the Analysis and Design of Multi- Agent Systems, International Journal of Software Engineering and knowledge Engineering, 2000

Authors' Information

Aurea Anguera de Sojo Hernández- Associate professor U.P.M. Crtra Valencia km 7, Madrid 28031, Spain;

e-mail: aanguera@eui.upm.es

Miguel Angel Díaz Martínez- Associate professor U.P.M. Crtra Valencia km 7, Madrid 28031, Spain;

e-mail:mdiaz@eui.upm.es

Abraham Gutiérrez Rodríguez- Asocciate professor U.P.M. Crtra Valencia km 7, Madrid 28031, Spain;

e-mail:abraham@eui.upm.es