The challenge of training new architects: an ontological and reinforcement-learning methodology[1]

Anabel Fraga, Juan LLorens
Universidad Carlos III de Madrid, Computer Science Department, Madrid, Spain
Email: {afarga, llorens}@uc3m.es

***Abstract*— This paper describes the importance of new skilled architects in the discipline of Software and Enterprise Architecture. Architects are often idealized as super heroes with a lot of qualities that are very infrequent in contemporary people. The Enterprise/Software Architect role could be assumed by a group of people able to manage the qualities needed by the role. But, a group or a single person must be taught and skilled for the discipline by training them using new learning methodologies or even by traditional university classes. In order to improve the process of becoming a new architect we propose a new methodology based on synergic ontological structures and reinforcement learning that aids in the education process for new skilled architects.**

***Index Terms*— Software/Enterprise Architecture, Education, Training, Methodology.**

I. INTRODUCTION

In the field of Software/Enterprise Architecture, it is important to offer methodological and applied support for new architects, but it is essential that they experience enthusiasm to be involved in the process of been a new Software/Enterprise Architect. College students may feel attracted to the process of analysis and design of new software; as far as they develop an abstract map, it could be for example the architectural design.

Nowadays, the Software/Enterprise Architecture courses are not standard in all universities. It is not the intention of this article to precise university by university if it is teaching more or less architecture courses, but some universities include this topic in a Software Engineering course, but not as crucial. Some others universities includes the subject as non major courses. And finally some others do not include the subject at all. The Software Architecture is treated as Design but at high level, the difference between applying styles or applying patterns is not evident and students does not feel the difference or even does not know that this subject is available as a discipline. Even more, students do not know the difference between an UML (*Unified Modelling Language*) Component Diagram and a Class Diagram, or even a deployment Diagram. The dissimilarity between the diagrams is high and the application is different as well. But, how could it be possible for a student to learn the difference if no architectural subject is taken or even provided in the universities?

Involving universities and students in the process of training new architects will be an incentive to obtain new skilled people in the field.

It is important to track architectural decisions because a subtle change in a security concern decision, for example, could change the global design of an application, and in this case ontologies guarantee this foundation. The Knowledge Organization Systems (KOS) domain is a formal and well studied area aiming with creating accurate structured knowledge, where Ontologies have a fundamental role. Ontologies play a great role in Knowledge Reuse as they could serve as repositories or even be reusable assets as well.

New methodologies concerning architectural decisions, case studies and so on, is needed. A practical methodology to be applied is suggested for that purpose. The methodology involves a forward and backward process that makes it an iterative process for the student. The first course must be an introduction to the basic and history knowledge. And then in the next ones, a deep practical study is essential. The existence of a good structural knowledge representation is very important for supporting student in the decision making process.

The reminder of this article is structured as follows: Section 2 include review of education concerning architecture courses and certifications, Section 3 includes some ontological references, Section 4 explains a training methodology based on ontological structures and reinforcement learning, and Section 5 covers Future Work and Conclusions.

[1] This paper is an enhanced version of the paper published as: *Fraga, A., Llorens, J. "Training Initiative for New Software/Enterprise Architects: An Ontological Approach," p. 19, The Working IEEE/IFIP Conference on Software Architecture (WICSA'07), 2007.*

II. SOFTWARE/ENTERPRISE ARCHITECTURE TRAINING AND CERTIFICATIONS

Software/Enterprise Architecture is quite a new topic in Software Engineering, the knowledge of this discipline is not showed in specific courses or even formally placed in the structure of a study plan.

Searching on diverse universities throughout internet, the plans are inclined to be located in one of the six intensities or levels of education or training for new architects in the field of Software Engineering. In this article, we called each level by a number from the basic level, level 0; to the higher level, level 5. Each level is specified as follows:

- Level 0: There is no existence of Enterprise/Software Architecture training at all. No evidence of courses, seminars or training is available for any student in the university.
- Level 1: Some courses are available, but not formally established. The level of the Architecture topic is treated at the same level as Design with no difference at all. Some courses are available for postgraduate and bachelor degree, but not mandatory.
- Level 2: Some courses are available. Architecture is treated as a high level in the design process. Some seminars or courses are available for postgraduate or last years of career, but they are not established as formal courses, the level of abstraction for the Software Architecture knowledge is just Design without detail, no reference to authors like Shaw [3], Garlan [17], Bass [1] or Perry&Wolf [2] are mentioned. Some times Software Architecture is mixed up in the Software Engineering course.
- Level 3: Some courses are available. Software/Enterprise Architecture is positioned slightly. Some seminars or courses are available for postgraduate or last years of career, but they are not established as formal courses, the level of abstraction for the Software Architecture knowledge is explained a little following the Shaw&Garlan book as primary book of the course [17]. Some seminars are available showing some history of the Software Architecture.
- Level 4: Some courses are formally established in the career, but as optional. It is the case of seminars and courses offered to students who are willing to know about this matter. As optional courses, the students are often previously involved with the matter because of working experiences.
- Level 5: Some courses are formally established in a career, so every student must take it. The students even without having previous contact with the discipline must study the history and abstraction level of a Software Architect role. It helps to make them interested in this relatively new discipline in the Software Engineering.

At this point, the SEI (Software Engineering Institute) [22] or the IASA (International Association of Software Architects) [14] are working constantly offering training courses, certifications, and working groups in order to coach new architects in the role.

The IASA group offers a new project called the IT Architect Skill Project. The library currently contains articles for each primary skill of the IT architect. IASA is currently in the process of augmenting the skills library with training materials, active communities of experts for peer interaction (IASA refers to these as "knowledge communities"), and links to other available resources for each topic such as books, book reviews and events. By the end of 2007, the IT architect skills library will be the largest compressive collection of resources for the practicing architect. The project fulfils a primary component of the IASA mission which is to allow the open architect community to guide the overall direction of the profession [23]. A new project on the go in the IASA Association is the project called IT Architect Training Program, industry is aware of new architects needs to be skilled, and maybe they are not upcoming prepared from college, so it is an initiative of architects that wants to teach new architects to be prepared in the field [24]. This initiative is a good solution in the industry world, a world that indeed needs well formed architects, but it is not a solution for in progress computer engineers. In the last case, universities must set up a position and also be involved in the needs that companies have at the moment. In that path, universities must try to satisfy real needs because student will require to know what they will be called to apply, and specially because they are engineers and they must solve the problems any company has, as expected.

Some authors wrote about the history of the discipline and they had given a light on for new architects interested in gain new knowledge concerning the profession [21].

The significance of the IASA active working group for creating a Framework of Knowledge for new architects is relevant and important for the mission and vision in the Software/Enterprise Architecture discipline. The role that an architect group have in the field is so essential in the SDP (Software Development Process) that it makes imperative to support this kind of initiatives. The reuse of Architectural Knowledge in the SDP process, helps new architects in the learning process in order to obtain new qualified personnel in a undersized quantity of time, and promote the discipline as a well formed and mature one.

Some companies like IBM agreed, as we do also, to treat the Architect role not just as one person, this role must be conceived for a group [15]. The Software Architecture and Enterprise Architecture roles are both often idealized as super heroes [16] [18] [19], with a lot of qualities that are very infrequent in contemporary people. The Software/Enterprise Architect role could be assumed by a group of people able to manage the qualities for the role.

In the Carlos III of Madrid University, the course of Software Engineering contains the Software Architecture matters, but not at all, it is positioned in the level 3. It is considered as design just at high level, some efforts of teachers are involving the students with the discipline but it is not enough time to include all valuable knowledge in a couple of months. A seminar is dictated once a year for postgraduate students, but it is optional and students

previously involved with Software Architecture matters are often captivated for that course, but not all are in the same situation.

Lago and Van Vliet, from the Vrije Universiteit of The Netherlands, treated the problem and integrated in the community a paper about a course on Software Architects [20], as an intention to achieve a level 4, in the levels that we distinguish in this article.

It is important to improve the methods and encourage the teaching of Software Architecture in the profession. It is important to set off transformations in the plans of study to include the Software Architecture as a reference course.

In our experience teaching Software Engineering courses for the last three years, the perspective of students is that functional requirements are more valuable that non functional requirements for Software Engineering. It is an opinion cultivated among students because of the unknown Software Architect point of view in the profession due to lack in the education process.

III. Ontological References in Software Architecture

The word ontology has been used to describe artifacts with different degrees of structure. These range from simple taxonomies, to metadata schemes, even to logical theories. The need to specify descriptions bring into being the following concepts:

- Classes (general artifacts) in all the domains of interest.
- The relationships that can exist among Classes.
- The properties (or attributes) that Classes may have.

An ontology defines the terms used to describe and represent an area of knowledge. Ontologies are used by people, DBMS, and applications that need to share domain information (a domain is just a specific subject area or area of knowledge, like medicine, tool manufacturing, real estate, automobile repair, financial management, etc.).

Ontologies include computer-usable definitions of basic concepts in the domain and the relationships among them, throughout this document, this definition is not used in the technical sense understood by logicians. They encode knowledge in a domain and also knowledge that spans domains. In this way, they make that knowledge reusable. Ontologies are critical for applications that want to search across or merge information from diverse communities [9].

Some works are available on the field of Software Architecture Knowledge Representation like Mehta´s paper [11] for Software Architecture Connectors Taxonomies, or Keshav´s investigation on a Taxonomy of Architecture Integration Strategies [12], or maybe a Taxonomy of Orthogonal Properties of Software Architectures [13], a Taxonomy for the Software Architecture Domain as a whole at the moment is available [4][5]. But none implementation of it in a CASE tool is available, for example; some projects are in progress but not completed yet.

The importance of this kind of taxonomies is clear if we take a look into groups of architects like the IASA group (International Association of Software Architects). Last year, the association has created a working group interested in the foundations of the discipline of Software Architecture in order to chart the largely uncharted profession of software architecture in a form that can be understood and leveraged consistently by not only professionals but also as clients and institutions [14]. An available ontology of Software Architecture is in both Spanish and English idioms [5].

IV. Training Methodology in Theory

After looking into the need of new architects and the need of new courses, a methodology based in ontologies and reinforcement learning could be of value for ongoing new architects.

The methodology core is a Software Architecture Ontology. The first step is to teach the basic theory of Software Architecture that a student should know about the discipline. Next step by step, practical classes are needed using the Ontology and a Case-Based Reasoning system to introduce problems from different levels to the student.

The levels of the practical problems to be solved must be from low to high levels, and they must grow little by little including some steps forward and backward to be sure the student gain a solid understanding of the solution applied.

It is known in Software Engineering and Reuse arenas that as systems become old, people come and go, and technology changes. The lack of memory, which is unavoidable without a solid change management system, results in a significant loss of efficiency, because vital decisions made in the development of the system are lost. If students are smart enough and think in future projects, then a question may come to their minds, how could be possible to know if a decision made in an architecture could be used in a similar architecture in an afterward development for a new system? Therefore, a key benefit of this methodology could be the traceability throughout the life cycle. It is ensured that tracks could be trailed because of the ontology, and at last the reuse of decisions for future projects could be achieved. It is an impressive goal to be reached thanks to the use of ontologies and the power that this kind of knowledge representations give us for sure.

The architectural decisions are a cardinal part of the early stages of a system. Prepared students since the very beginning of the process is crucial. Through these decisions it will be possible to maintain and evolve the system to be developed. If decisions are not good enough then the rigidity and no adaptation of the new software to changes in the environment could be a problem in the future.

It is important to track architectural decisions because a subtle change in a security concern decision could change the global design of an application, in that sense ontologies guarantee this foundation. For example, if we change the authentication process of a secure system, then the change in the non functional requirement will point to an alteration in diverse components of the system, because security is an aspect to be considered in the whole software system. The reason of the change should be kept and changes as well. If we keep this information then potential changes will be easy to accomplish, the past and the evolution of the system is stored and known by the architect if she/he needs to make a forthcoming decision.

The figure 1 show the iterative process of learning. The practical effort will be greater than the theory effort. The lines in the figure 1 that goes from the courses boxes to the practical training boxes hint a dependency between the courses and the practice. The dependence level is represented by the weight of the line, for the first course the practical need will be lower than the needed for the advance course.

The forward and backward process makes it an iterative process for the student. It is suggested to include the basic and history knowledge in a first course.

After the introductory course will follow two courses. They will be the substantial teaching of knowledge and practical solutions of Software Architectures examples. This courses could be optional and the student after knowing the existence of the field could select to study it deeply or not.

It is imperative to include the importance of non functional requirements in the study of the discipline, systems without quality attributes consideration are not complete architectural studies. It seems to be obvious but it is important to be settled for students. The students by experience are not quite in knowledge of it.

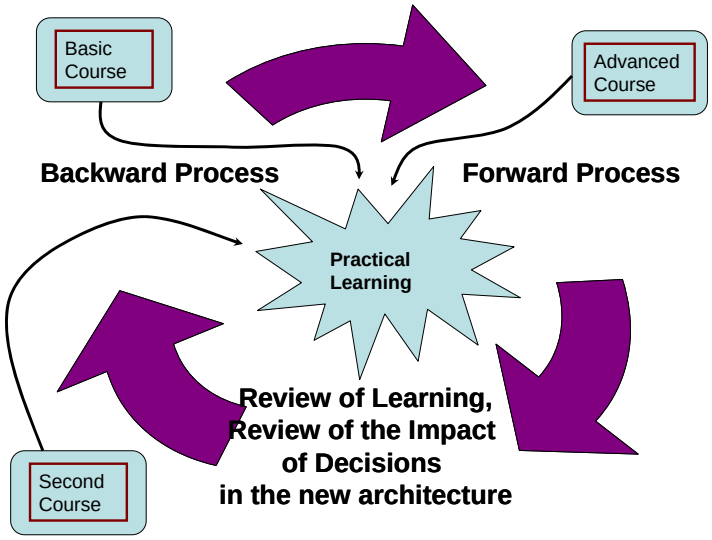

Figure 1. Illustrative Learning Methodology Process Diagram.

The methodology implies also an implicit reuse of knowledge from preceding projects and from previous architectural decisions.

IV. CONCUSIONS

In this work, it is described the importance of new architects in the discipline of Software Architecture and Enterprise Architecture. Architects are often idealized as super heroes with a lot of qualities that are very infrequent in contemporary people. The Software/Enterprise Architect role could be assumed by a group of people able to manage the qualities for the role. In any case, even a group or a single person must be raised in the discipline by training courses, new methodologies of learning, or traditional university studies. In order to improve the process of becoming a new architect we propose a methodology based on ontological structures and reinforcement learning.

Knowledge is very difficult to accumulate, be sought and be integrated for new needs. One of the basic problems with different types of knowledge is that reusers do not always get what they need from repositories, for reasons that have to do in part with how repositories are created, in part with not up-to-date retrieval techniques, and with almost not existing solutions for smart merging and integrating knowledge within other knowledge. This is a big part of the "window" to be covered by the Knowledge Reuse area.

The importance of new courses in the discipline and the awareness of teachers to be involved and prepared in the discipline as well is significant. The methodology is waiting to be implanted as soon as a new emerging course of Software Architecture will be introduced in the University.

It is important to track architectural decisions, and even reuse them, because a subtle change in a security concern decision, for example, could change the global design of an application, and in this case ontologies guarantee this foundation. The Knowledge Organization Systems (KOS) domain is a formal and well studied area aiming with creating accurate structured knowledge, where Ontologies have a fundamental role. Ontologies play a great role in Knowledge Reuse as they could serve as repositories or even be reusable assets as well.

New methodologies concerning architectural decisions, case studies and so on, is needed. A practical methodology to be applied is suggested for that purpose. The methodology involves a forward and backward process that makes it an iterative process for the student. The first course must be an introduction to the basic and history knowledge. And then in the next ones, a deep practical study is essential. The existence of a good structural knowledge representation is very important for supporting student in the decision making process.

The core of the recommended methodology is a Software Architecture Ontology supported by a Case-Based Reasoning software that could help students to apply knowledge and build solutions of new architectures depending on the problem created by increasing levels of difficulty from low to high, and a backward-forward

methodology that helps students to fix the knowledge and practice them in practical situations. Because tracking of architectural decisions is important, a slight change in a non functional concern could change the global design of an application, in that sense ontologies guarantee this knowledge basis. The tool is still under construction, and waiting for a ministry scholarship for undergraduate computer engineers, and it will help students to create new architectures based on cases and reasoning of the cases as well. In that sense, the ontology will be growing year by year and it is expected to have a solid knowledge representation after two years, since the application of the methodology.

REFERENCES

[1] Bass, L., Clements, P. and Kazman, R. Software Architecture in Practice. Addison Wesley. Second Edition. (2003)

[2] Perry, D.E. and Wolf A.L." Foundations for the Study of Software Architecture". ACM Software Eng. Notes, vol. 17, no. 4, 1992, pp. 40–52.

[3] Shaw, M. "The Coming-of-Age of Software Architecture Research". Proc. 23rd Int'l Conf. Software Eng. (ICSE 01), IEEE CS Press, pp. 656–664a. (2001)

[4] Fraga, Anabel; Sanchez, Sonia; Lloréns, Juan. "Creación de un Tesauro Manual y Automático para el dominio de Arquitectura de Software". Workshop Ingeniería del Software (WIS2005) de las Jornadas Chilenas de Computación. Chile. (2005)

[5] Fraga, Anabel; Sanchez, Sonia; Lloréns, Juan; Astudillo, Hernán." Knowledge Representation for Software Architecture Domain by Manual and Automatic Methodologies". Electronic Journal of CLEI 2006 (EJCLEI2006) (http://www.clei.cl/cleiej). (2006)

[6] The Reuse Company. http://www.reusecompany.com [July, 2005]

[7] Llorens, J.; Fuentes J.: "Computer Aided Knowledge Environment CAKE". The Reuse Company, Winter (2004). http://www.reusecompany.com [23.06.2005]

[8] Díaz, I. ; Llorens, J.; Génova, G.; Fuentes, J.M.: "Generating domain representations using a relationship model." Information Systems, 30(1): 1-19, March (2005).

[9] W3C. Ontologies. http://www.w3c.es/Traducciones/es/SW/2005/owlfaq [July, 2007]

[10] Software Product Management Network. http://www.spmn.com/16CSP.html [July, 2007]

[11] Mehta, Nikunj R.; Medvidovic, Nenad; Phadke, Sandeep. "Towards a taxonomy of software connectors". Proceedings of the 22nd international conference on Software engineering. (178-187). Ireland. (2000).

[12] Keshav, R.; Gamble, R. "Towards a taxonomy of architecture integration strategies". Proceedings of the third international workshop on Software architecture (89-92). (1998).

[13] Bratthall, Lars; Runeson, Per. "A Taxonomy of Orthogonal Properties of Software Architectures". Workshop in NOSA. (1999).

[14] International Association of Software Architects. http://www.iasahome.org/iasaweb/appmanager/home/workgroup [July, 2007]

[15] IBM. http://www-128.ibm.com/developerworks/rational/library/mar06/eeles/index.html [July, 2007]

[16] Architecture Roles. http://stevendwright.home.comcast.net/ArchRoles.htm [July, 2006]

[17] Shaw, M. and Garlan, D. "An Introduction to Software Architecture". V. Ambriola and G. Tortora, eds., Advances in Software Engineering and Knowledge Engineering, vol. 2, World Scientific Publishing, pp. 1–39. (1993)

[18] Astudillo, H. and Hammer. "Understanding the architect´s job". Software Architecture Workshop of OOPSLA. (1998)

[19] Bredemeyer, D. and Malan, R.. "The Role of the Architect. Bredemeyer Consulting". White paper. (2002)

[20] Lago, P. and Van Vliet, H. "Teaching a Course on Software Architecture". Software Engineering Education and Training. CSEE&T 2005. Proceedings. 18th Conference, pp. 35-42. (2005)

[21] Kruchten, P., Obbink, H., Stafford, J. "The Past, Present, and Future of Software Architecture". Software, IEEE. (23): 2, pp. 22-30. (2006)

[22] SEI: Software Engineering Institute. Carnegie Mellon. http://www.sei.cmu.edu/ [July, 2007]

[23] IT Architect Skill Library. IASA. http://www.iasahome.org/web/home/skillset [July, 2007]

[24] IT Architect Training Program. IASA. http://www.iasahome.org/web/training [October 2007]]

Anabel Fraga is a Computer Engineering professional. She committed her efforts in the industry as UNIX/Linux/Windows Administrator, Application Administrator and Project Management. She obtained her E-commerce Master degree and the Advance Studies Diploma in the Carlos III of Madrid University, and she is working on the dissertation for her PhD in Computer Science in the Carlos III University. Research areas are: Software Architecture, Information Engineering and Reuse, ethics and innovative learning methods for supporting new software architects. Professor of Software Engineering and Information Engineering. Member of ACM CSTA and IASA.

Juan Llorens is an Industrial Engineer. PhD in Industrial Engineering and robotics. He started his own company dealing with Artificial Intelligence applied to Database systems. In 1989 the whole company was sold to the European Multinational Advanced Software Technology and he become the Marketing Director. His main subject is Information representation and processing for Software Reuse. He splits his educational activities between Madrid and the ATL (Mariehamn, Finland). He is member of the IEEE Computer Society and ACM.