Anticipation, Interoperability and Service's Quality

Dr. Juan Jesús Torres Carbonell*, Dra. P. Paderewski-Rodríguez**
Members of GEDES Research Group (Group on Specification, Development and Evolution of Software)
E.T.S. Ingenierías Informática y de Telecomuniación. University of Granada
* jjtc28@hotmail.es, ** patricia@ugr.es

Abstract
A software system relates with others, and receives or provides services. The way services are provided requires interoperability, and software system adapts and evolves according to the necessities. As consequence, the quality of the service depends on the interaction of several software systems. We approach software evolution under the point of view of Anticipation and Incursion. So, we try to define which could be the parameters and elements that define the future states of the service provision framework. We use the concepts of anticipatory systems, incursion and hyperincursion, in the area of software engineering, evolution and adaptation of systems that form part of a complex system interoperable structure for the provision of services. The final objective is the use of these concepts to improve the service provision quality.
Keywords: Software Evolution, Interoperability, Anticipation, Incursivity.

1 Introduction

Applications and software systems are not isolated entities, they have relations with the environment. Furthermore these relations can include others software systems from which it receives services o to which it provides services. So, there exists a close relation between different software systems to produce a concrete service. This situation implies that the quality of the service offered to the end user, depends on the interaction of several software systems that perhaps are dependant of more than one organization. The way the service is provided requires interoperability and also that the software systems within the framework of the service provision infrastructure adapts and evolves according to the necessities and taking into account the existing relationships.

The novelty of this paper is to approach software system evolution and adaptation under the point of view of Anticipation and Incursion. So, we consider the whole framework of systems that participates in the service provision through the use of web-services [4] and UDDI [2] directory, as a macro-system in which the next state can be established according to the current state, the previous states, a set of specific parameters and the future state that responds to the service provision necessities. Thus, the evolutionary process tries to reach and maintain and improve the quality of the service provided to the end user.

We use the concepts of anticipatory systems, incursion and hyperincursion, in the area of software engineering and specifically in approaching the evolution and adaptation of systems that form part of a complex system interoperable structure for the

provision of services using web-services. The final objective is the use of these concepts to improve the service provision quality.

We present in Section 2 a revision of the characteristics of service provision based in web-services. In Section 3 we expose how to approach the quality necessity of the services and how system's evolution is approached. In Section 4 we introduce the anticipatory model and how it can be applied to software systems. In Section 5 we present how the next state in an evolutionary process of software system that request or receive web-services is reached based in anticipation and incursivity concepts. Finally, conclusions are presented in Section 6.

2 Interrelation Services

A change in the way services are provided has been produced. From a vertical structure that was monolithic and independent from any other structure (designed not to suffer changes in time), we are moving towards a new paradigm: the use of web-services. Under this new paradigm the design aims to obtain as much as possible benefit from the services offered by other systems and subsystems, simplifying in that way the production and delivery of new services.

Such a structure of this type produces a close relationship between its elements, and these elements are organized in a structure that produces services for the end user. The whole structure is organized in a multilayer component set interrelated according to their necessities. Within an organization based in the use of web-services we can find a base of structured information that is used by all the applications (either directly or through web-services). This structured information makes easier the development of applications. So (see Figure 1) applications access to different data bases in different ways.

Additionally, applications can be used through different channels (PDA, Internet, Intranet, TDT, etc.) and the access can be originated outside of our organization, when it is an access using web-services that our organization provides to others. So, web-services can be used by systems of the same organization and systems that belong to other organizations, in order to provide services to final users.

As we have mentioned, services can be provided internally or externally and conform a network of relationships in which one system depends on the others. In this paper we will refer to systems, either if it is a whole system or if we refer to subsystems that provide web-services one to another. So a "macro-system" will be a set that include all the systems belonging to all the organizations that take part in the composition of a service. Additionally, according to the proposal by MDA [3]:

"A system may include anything: a program, a single computer system, some combination of parts of different systems, a federation of systems, each under separate control, people, an enterprise, a federation of enterprises ..."

"A model of a system is a description or specification of that system and its environment for certain purpose".

Also, we call service to the final service, that is, the service provided to the user, and web-service to the service that one system provides to another.

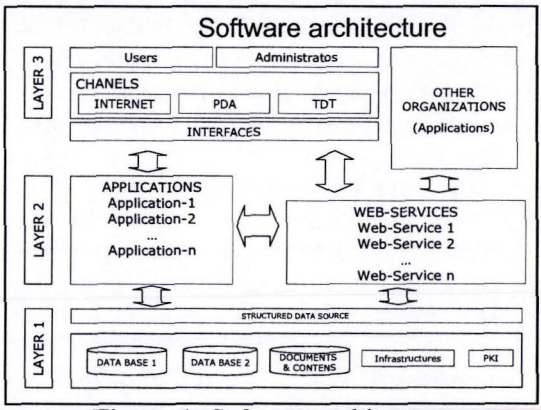

Figure 1: Software architecture

In Figure 2 we present in a schematic way how the final service is provided to the end user. This service uses web-services provided by the organization itself, obtained using the organization's Intranet, and web-services provided by others organizations and obtained through Internet.

When a service is settled there a contract between the client and the server exists, in which service's characteristics are gathered. Additionally, a SLA (Service Level Agreement) is defined, and is compulsory for both of them, the user and the provider of the service. The existence of a SLA is useful to check at each moment of time if commitments included in services contracts are accomplished, and if they are still useful or it is time to modify them. In this last case an evolutionary process should be followed by the system, as we will see in the following sections.

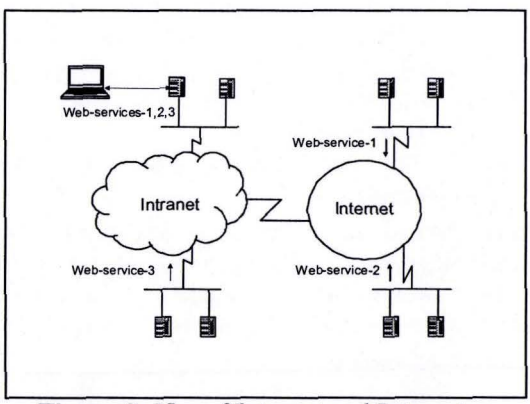

Figure 2: Use of Intranet and Internet

This macro-systems' complexity may require the availability of information for the manager. This information ought to be easy to understand and allows the manager to

have the necessary information about the final service's provided quality. Thus, the manager of the organization will access to the information about the behaviour of the whole system and make, if necessary, the right decision about service composition. Nevertheless, these decisions must be as few as possible, for the system to evolve and adapt by itself according to the pressure of the environment. This evolution could require complex decisions in an environment in which available information about systems that are providing services isn't complete.

In this situation (lots of relationships between several systems and incomplete information) it seems appropriate to have some kind of reproduction of the whole macro-system. This reproduction should allow the service provision to be done based in a model of the system (or macro-system). This model allows making the right decision to undergo the evolutionary process.

So, there exists a complex macro-structure that provides services. In this macro-structure systems use the services provided by other systems, using web-services. This implies that the way for an application to solve its necessities isn't to solve them on its own but to use web-service provided by others systems, and then services must be published.

Services relationships may be coordinated by a technology interchange point in which services users and providers met. Additionally, the necessity of using an UDII (Universal Description, Discovery and Integration) directory can rise. An UDDI directory can support services search. Applications access UDII directory searching for the web-services' address they need and then they ask for the service provision to the provider organization (system) at the address obtained in the directory.

The sequence to follow for creating, publishing and use of web-services is the following (Figure 3):

1. Organization O-1 designs and offer web-services. These web-services are provided by the organization.
2. The organization O-1 registers the web-service in the UDII directory for the address to be published. After this, applications will be able to access to the UDII directory to obtain the address in which the service is provided.
3. Whenever an application Ap-i needs the web-service it is located at the address provided by the UDII directory.
4. Application Ap-i will connect to the web-service. Provider and requester interchange messages and data.

These interrelations between applications and systems are the reason for one to owe another par of its functionality. As a consequence, the rise on functionality because of interrelations makes ROI (Return Of Investment) rise as well. Then, applications and systems using web-services are more productive for the organization as they use functionalities that are yet developed.

As the systems that provide services can evolve or the quality of the service provided can change, we must provide ours systems and applications (that use web-service) with capacity for evolution and adaptation to preserve the quality of the final service that we provide to users.

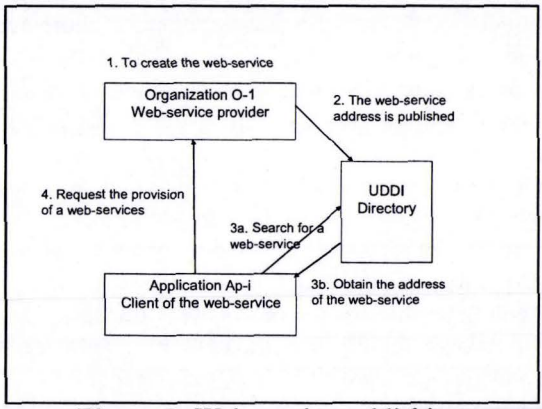

Figure 3: Web-service publishing.

3 System Evolution and Quality

We are going now to approach how to assure a higher quality obtained as a result of the interrelation between systems. To reach these benefits we need feedback about the current state and do accurate prediction in advance, because a collapse of the system or a lack of functionality of any of the web-service used by our system, could cause problems to the service provided to the end user, or put in danger our organization.

There could be doubts and scepticism if as consequence of the dependence on other systems a weakness appears in the value chain of the service provision. This weakness must be controlled and reduced as much as possible to be sure that the quality maintains a level the higher the better. In the case the levels of quality couldn't be kept in proper standards it should be necessary to think about if the use of web-services is a good solution for our system or not.

In this situation there is a series of questions that could be set out:

- How can be measured the whole chain of service provision?
- How to establish SLAs?
- How to specify SLAs?
- How to deal with service degradation?
- How to fix and solve errors during service execution?
- How to adapt the use of the service according to circumstances?
- How to change and evolve SLAs?

These questions are difficult to deal with, because answers depend very much on the objective of the organization. Furthermore, more questions could be added to the list that reflect the point of view of the organization, but all of them aim to a higher quality and imply evolution and adaptation.

Related with the necessity of quality, software system's evolution and adaptation requirement must be approached. The system must undergo modifications for the service provision to reach the highest level of quality. Because of the macro-system's complexity the searching and attaining of quality is a difficult task, due to dependence

of one system on another, and even of one organization on another. Additionally, the complexity is consequence of the lack of one responsible (there are several of them) of the whole chain of value from the web-services providers to the en users.

The evolutionary process of a system that provides a service to end-users, will depend on the factors and characteristics of the systems that provide to it web-services and information that allows the client system to make accurate decisions during its evolutionary process. This evolutionary process additionally requires certain amount of knowledge. With regard to systems evolutionary characteristics, we consider the following [5]: Our understanding of Software System characteristics consists in considering its functionality and its evolution as no hazardous processes, but certain knowledge about the characteristics of the next state is needed.

As a mechanism to evolve taking into account the necessary knowledge about the next state, we think that the anticipatory model can be a solution to offer a theoretical framework. In the next section we present the anticipatory model and the way it can help to understand the evolution of software system that use web-services.

4 Anticipatory Model

4.1 Anticipation and Software Systems

We have previously presented our point of view of Software Systems evolution under the perspective of Anticipation [5] [6]]. Following [1], we have presented the way we consider the definition of an Incursive Discrete Strong Anticipatory System: an incursive discrete system is a system which computes its current state at time t, as a function of its states at past times, ..., t-3, t-2, t-1, present time, t, and even its states at future times t+1, t+2, t+3, ..., that is to say:

$$x(t+1) = A(..., x(t-2), x(t-1), x(t), x(t+1); p) \quad (1)$$

In the framework of Software System, we can similarly write:

$$S_{t+1} = M(..., S_{t-2}, S_{t-1}, S_t, S_{t+1}; OS) \quad (2)$$

Where:

- .., St-i, Are the previous states of the Software System known through the Memory of the system.
- St Is the current state of the Software System.
- St+1 Is the next state of the Software System, the state at the following evolutionary instant.
- OS Is the parameter that indicates the evolutionary modifications to be performed.

That is to say, the next state of the evolutionary process of the Software System is decided according to the previous states, the current state and also the following state itself, applying additionally an evolutionary operator that is defined by the parameter OS.

We think on the evolutionary processes followed by a Software System as an Incursive Discrete Strong Anticipatory System (IDSAS) [1]. As IDSAS the next state is influenced by the present state and the next state, according to a parameter that indicates the evolutionary modifications or performed. But, as presented in the next section, we accept also the existence of Incursive Discrete Weak Anticipatory System (IDWAS) framework. Furthermore, even if we don't present this concept in this paper, we think that Software Systems are Hyperincursive Systems because during their evolution multiple possible states could be reached, although the selection process will collapse in one state.

4.2 Inter-anticipation and Web-Service

In the anticipatory framework the coexistence of strong and weak anticipation is possible when considering the evolution of Software System. The strong or weak character will depend on the kind of knowledge the system has about state t+1 and the following ones. If the knowledge is the result of a model of the system, then it is said to be weak anticipation.

We have previously proposed that anticipation between subsystems could be considered as weak anticipation [7] :"...That is to say, subsystems could manage a model of other subsystems and anticipate the future states according to that model, as a complement to strong anticipation, characteristics of the intra-anticipation, in which a system/subsystem anticipates it s own evolutionary process..." .

In this paper we consider that we are dealing with a software system inter-anticipation mechanism in which the system uses a model of the system (or systems) that provides web-services. This model is used to make decisions on the next state in time t+1. According to this point of view, the system behaves as an Incursive Discrete Weak Anticipatory System

To perform anticipation, we follow the proposal from [7], based in the following sequence:

1. Creating the internal representation of the system's state (that is to say, a model of the system that provides web-services).
   The subsystem/system must have a representation of the elements of the service provision that are relevant for the evolution and quality of the service. This implies that the model is to be updated.
2. Proposing and selecting operators for the evolution.
   According to the current state and the knowledge provided by the model, a selection of the next operator to be applied is carried out.
3. Simulating the execution of operators.
   Using the model, the application of the operator is simulated in order to decide the viability of the selection.

4. Deciding that anticipation is useful.
   According to the results of the simulation it is decided whether the anticipation is useful or not.
5. Using anticipation to select other operators.
   The anticipation process could be the basis for further decisions.

In point 1) it must be taken into account that a model of every system provider of web-services must be available, at least it should be modeled the elements of those systems that are relevant in making decisions. So the result is the set of all web-services received models:

$$\sum_{j=1}^{n}\sum_{k=1}^{m} Model(S_j WS_k) \qquad (3)$$

Where Sj-WSk is the model that the receiver system has of the web-service WSk provided by the system S_j. The model is the formal representation of all functional characteristics (and other elements) that should be taken into account for evolutionary process objective. In the next Section we explain how to obtain the necessary information for decision making.

5 Deciding Next State

As we have previously stated, several systems participate in the global mechanism aiming to provide a complex service. These systems may depend on different organizations, and can have been developed with different objectives. Nevertheless these differences several common elements can be settled:

- There exist web-services defined as XML elements that are interchanged between systems.
- There exist SLAs (Service Level Agreements) that establish mutual commitments. These agreements are defined and accepted by provider an user of the web-service.
- There exist quality indicators, derived from the SLA, that allow the provider to be sure it is performing well and accomplishes its responsibilities. Indicators make easier for the receiver to be sure about the quality of the service provision as well.

As the web-service structure, the SLA and indicators will be available, System-i that receives service WS_k provided by System-j will be able to have a representation of the service, in such a way that the service is reduced to a model based in these three elements:

$$FunctionWebService_{jk} = \{WS_{jk}, SLA_{jk}, indicators_{jk}\} \qquad (4)$$

As an example of the information provided by this function:

1. WS_{jk} : Is the web-service structure, and it can be represented as a result

of:

i. Detected errors
ii. Out of range values
iii. Empty fields

2. SLA_{jk} : that could be represented as:
   i. % of error
   ii. Hours of service interruption
   iii. availability
   iv. Penalizations
3. $Indicators_{jk}$: established according to the services provided and the SLA.

As consequence, when making a decision on the next evolutionary state of the system, the decision could be based on:

$$S_{t+1} = M(...,S_{t-2},S_{t-1},S_t,S_{t+1};OS) \quad (5)$$

- .., St-2, Are the previous states of the Software System known through the Memory of the system (M).
- St Is the current state of the Software System.
- St+1 Is the next state of the Software System, the state at the following evolutionary instant.
- OS Is the parameter that indicates the evolutionary modifications to be performed. It can refer to:
  - a new or different web-service to ask for at an UDDI,
  - an alert to be fire and alert the Administrator.

Each evolutionary modification is the necessary change to obtain the following evolutionary state, having into account that the value of the current model of the web-service function is:

$$Model(S_j WS_k)_t = \left[FunctionWebService_{jk}(WS_k, SLA_{jk}, indicators_{jk})\right]_t \quad (6)$$

That is to say, the value of the function $FunctionWebService_k(WS_k, SLA_k, indicators_k)$ in time t (the result of the provision of the web-service in t), is the model of the system in relation with the web-service in instant t. Depending on this model an operator OS_i will be selected and will be applied to produce an evolutionary process in the system. Furthermore, there can be several web-services provided by one or several systems. So, we have that the set of necessary modifications will be:

$$\sum_{j=1}^{n}\sum_{k=1}^{m} Model(S_j WS_k)_t = \sum_{\substack{j=1 \\ k=1}}^{\substack{j=n \\ k=m}} \left[FunctionWebService_{jk}(WS_{jk}, SLA_{jk}, indicators_{jk}\right]_t \quad (7)$$

That is the information provided by the model about a series of services (k=1, …, m) provided by a series of systems(j=1, …, n) that are services providers.

We have, then, that the model is a formal representation of functional or structural characteristics of the systems, and we must take them account within the evolutionary process.

The function $FunctionWebService_{jk}(WS_{jk}, SLA_{jk}, indicators_{jk})$ will return a value of the function depending on:

- If the WebService WS_{jk} accomplishes the established standards: fields, values, structure, etc.
- If the SLA_{jk} is within the established margins :
    - Commitment 1: value 1.
    - Commitment 2: value 2.
    - ...
    - Global value SLA_{jk}.
- If indicators have specified values:
    - Indicator 1: value 1.
    - Indicator 2: value 2.
    - ...
    - Global Indicators value.
- Then, the value of the function will be one of the values in an accepted range of values previously defined.

Additionally, the knowledge of the evolutionary process followed by the system through previous states produces an expected quality level. This expected quality together with the model establishes an evolutionary operator to be applied:

Table 1: Version Space.

Expected Quality	$Model(S_iWS_k)_t = FunctionWebService_{jk}(WS_{jk}, SLA_{jk}, indicators_{jk})t$			
	Range 1	*Range 2*	...	*Range n*
Range 1	OS_1	OS_2	...	OS_n
Range 2	OS_{n+1}	OS_{n+2}	...	OS_{2n}
...	...	...	...	...

Expected Quality will depend on the knowledge about previous states ..., S_{t-2}, S_{t-1}, S_t,, that is to say previous experience of the system evolutionary history and can be organized in ranges that help to formalize and systematize the followed process, for example, using decision rules.

What is established in Table 1 is the set of evolutionary operators that can produce modifications in the software system, that is to say, the set of operators that produce the Version Space of Possible Configurations. This Version Space will include the possible and viable configurations that obey to the temporal, ontological and functional necessities [8]

As consequence of the information available an operator can be applied that replaces the request of a web-service by a different one. For example if the quality of the service provided by WS_{il}, goes down it can be replaced by the web-service WS_{jk} by the modification of the request. Now we have all the necessary elements for the system (or the Administrator) to make a decision about the next state in the evolutionary process:

Previous states (... S_{t-2}, S_{t-1}), Current State (S_t), $FunctionWebService_{jk}$, and expected quality. The only remaining step is to decide which are the "weights" of the relations between these elements in order to decide the evolutionary modification to perform that is represented by OS (Figure 4).

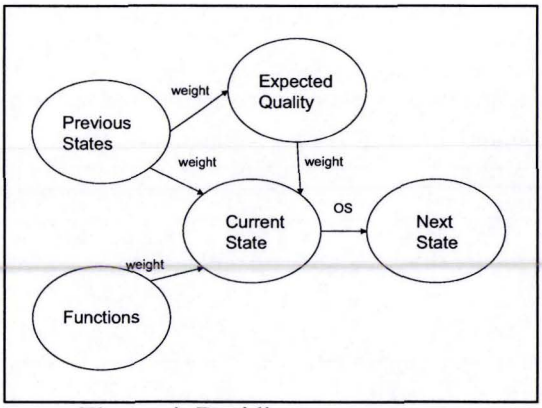

Figure 4: Deciding next state.

Now we must propose our future work: to study which kind of changes and modifications of a system that uses web-services, should imply changes in the system and in the systems that provide web-services as well. This interaction could be dealt by the designing of events plus a manager-agent that fires evolutionary actions to activate the web-services involved.

6 Conclusions

Most of the current computing systems built services (to be offered to end-users) using web-services provided by other systems (or subsystems). As consequence a framework that allows the evolution and adaptation of software systems to assure the quality of the provision is needed, even more having into account the dependence on other systems. Anticipation framework and its applications to the evolutionary process makes easier to set up a version space. In this version space operators are applied depending on the model used to establish the next state St+1: according to the available information provided by the model of the web-services used (that depend on web-service received, SLAs and indicators), the previous states (..., S_{t-2}, S_{t-1}, S_t) and the expected quality. Future work will be focused on applying the framework to an example case and the definitions of a Version Space of Possible Configurations.

References

[1] Dubois, D.M. (2000): "Review of Incursive, Hyperincursive and Anticipatory Systems-Foundation of Anticipation in Electromagnetism" in Computing

Anticipatory Systems: CASYS'99 – Third International Conference, edited by D.M. Dubois, AIP Conference Proceedings 517, Melville, New York, 2000, pp. 3-30.

[2] OASIS 2004. UDDI UDDI Version 3.0.2, UDDI Spec Technical Committee Draft. http://uddi.org/pubs/uddi-v3.0.2-20041019.pdf

[3] OMG 2003. "MDA Guide Version 1.0.1" http://www.omg.org/docs/omg/03-06-01.pdf

[4] Thomas Erl . 2005. "Service-Oriented Architecture. Concepts, Technology, and Design". Prentice Hall. ISBN 0-13-185858-0. First edition, 2005.

[5] Torres-Carbonell, J.J. and J. Parets-Llorca (2001): "Software Evolution. What kind of Evolution?", in Computing Anticipatory Systems: CASYS 2000 – Fourth International Conference, edited by D.M. Dubois, AIP Conference Proceedings 573, Melville, New York, 2001, pp. 412-421.

[6] Torres-Carbonell, J.J.; Parets-Llorca, J.; Dubois, D.M. (2002). "Software Systems Evolution, Free Will and Hyperincursivity". International Journal of Computing Anticipatory Systems, CHAOS, Liège, Belgium. Volume 12. Soft Computing and Computational Intelligence; Cognitive, Neural, and Psychoanalytical Anticipation; Computer Science, Data and Knowledge Systems. Partial Proceedings of the Fifth International Conference CASYS'01 on Computing Anticipatory Systems, Liège, Belgium, August 13-18, 2001, D. M. Dubois (Ed.), pp. 3-24.

[7] Torres-Carbonell, J. J.; Paderewski-Rodríguez, P. and J. Parets-Llorca, (2004). "Software Systems Intra/Inter-Anticipation", in Dubois D. (Ed.): "Computing Anticipatory Systems: CASYS 2003 – Sixth International Conference". Published by The American Institute of Physics, AIP Conference Proceedings.

[8] Torres-Carbonell, J. J.; Parets-Llorca, J. and D.M. Dubois (2005) "Emergencies Scope of Complex Systems. Software Version Space and Incursivity" , published in the book ANTICIPATIVE AND PREDICTIVE MODELS IN SYSTEMS SCIENCE, Volume I, pp 45-50; by George E. Lasker and Daniel M. Dubois (Eds), published by the IIAS, in Windsor, Ontario, Canada in May 2005. ISBN 1-894613-49-X.