Entre Ciencia e Ingeniería, año 12 (vol 12), no. 23 – Primer semestre de 2018, página 117 - 122 DOI: http://dx.doi.org/10.31908/19098367.3711
ISSN 1909-8367 (Impreso), ISSN 2539-4169 (En línea)

Concern detection along the requirement development[1]

Detección de incumbencias en el desarrollo de los requisitos

Detecção de incumbências no desenvolvimento dos requisitos

F. Pinciroli

Recibido: diciembre 10 de 2017 – Recibido: febrero 15 de 2018

***Resumen*—Existen numerosos enfoques orientados a aspectos presentando soluciones para las diferentes fases del Ciclo de Vida del Desarrollo de Software (SDLC, por su nombre en inglés). Pero no existen propuestas con un proceso coherente y empleando notaciones y herramientas estándares a lo largo de todo el SDLC. Hemos elaborado una alternativa llamada Aspect-Oriented Process for a Smooth Transition (AOP4ST), que permite la incorporación paulatina del paradigma orientado a aspectos en los proyectos actuales en la industria y ofrece una propuesta completa y homogénea para todas las fases del SDLC. En este artículo presentamos cómo encontrar las incumbencias en las primeras etapas de AOP4ST, cuando pasamos del modelo de negocio al modelo de requisitos de software, llevando a cabo la actividad de desarrollo de los requisitos dentro de la ingeniería de requisitos.**

***Palabras Clave*—modelo de procesos de negocio, modelo de requisitos de usuario, modelo de requisitos de software, desarrollo de software orientado a aspectos, incumbencias transversales, AOP4ST.**

***Abstract*—There are a lot of aspect-oriented approaches presenting solutions for the different phases of the Software Development Life Cycle (SDLC). However, there is no approach with a coherent process and employing standard representations and tools along the whole SDLC. We have elaborated an alternative called Aspect-Oriented Process for a Smooth Transition (AOP4ST), that allows the smooth incorporation of the aspect-oriented paradigm in the current industrial projects and offers a complete homogenous proposal for the phases of the SDLC. In this paper, we present how to find concerns at the first stages of AOP4ST, when we move from the business model to the software requirements model, performing the requirement development activity of requirement engineering.**

***Keywords*—business process model, user requirement model, software requirement model, aspect-oriented software development, crosscutting concerns, AOP4ST.**

***Resumo*—Existem inúmeros abordagens orientados a aspectos apresentando soluções para as diferentes fases do Ciclo de Vida do Desenvolvimento de Software (SDLC, pelo nome em inglês). Mas não há propostas com um processo coerente e usando notações e ferramentas padrão em todo o SDLC. Temos desenvolvida uma alternativa denominada Aspect-Oriented Process for a Smooth Transition (AOP4ST), que permite a incorporação gradual do paradigma orientado a aspectos em projetos atuais da indústria e oferece uma proposta completa e homogênea para todas as fases do SDLC. Neste artigo, apresentamos como encontrar as incumbências nas fases iniciais de AOP4ST, quando passamos do modelo de negócios para o modelo de requisitos de software, realizando a atividade de desenvolvimento de requisitos dentro da engenharia de requisitos.**

***Palavras chave*— modelo de processos de negócio, modelo de requisitos do usuário, modelo de requisitos de software, desenvolvimento de software orientado a aspectos, incumbências transversais, AOP4ST.**

I. Introduction

The emergence of the aspect-oriented paradigm brought new expectations about the possibility of building software in a more modular way and improving

[1]Producto derivado del proyecto de investigación "Ingeniería de requisitos orientada a aspectos en AOP4ST", a través del grupo de investigación de Ingeniería de Software del Instituto de Investigaciones de la Facultad de Informática y Diseño de la Universidad Champagnat, Mendoza, Argentina

F. Pinciroli, Universidad Champagnat, Mendoza, Argentina, email: pincirolifernando@uch.edu.ar

Como citar este artículo: Pinciroli, F. D. Concern detection along the requirement development, Entre Ciencia e Ingeniería, vol. 12, no. 23, pp. 117-122, enero - junio, 2018. DOI: http://dx.doi.org/10.31908/19098367.3711

its quality attributes, such as maintainability, flexibility, comprehensibility, reusability, etc. [1].

However, the new paradigm also came with new challenges, since there is not enough casuistry of its use in the industry guiding towards a way to apply it properly, particularly with regard to techniques, tools, notations and good practices.

Software development projects must deal with a large number of risks. It is not advisable to add new ones, such as incorporating an approach that is not sufficiently mature and that requires training people in poorly known tools and techniques, applying not well-tested methods, lacking support from vendors, and many more.

Similarly, there are no methodological proposals employing standard symbolizations and covering the full SDLC, so in case of applying the aspect-oriented paradigm, we are obliged to compose a method by picking up parts from different authors, who worked each phase of the SDLC in isolation [2].

In this context, we decided to move forward with an alternative that allows us to incorporate the aspect-oriented paradigm in the current projects in the industry and offering a complete proposal by unifying homogenously the different phases of the SDLC.

In this paper, we present how to find concerns during the user requirement development, from AOP4ST's business model until software requirements model. AOP4ST is a framework process for software development whose aim is the integration of the aspect-oriented paradigm in a smooth way, causing the least possible negative impact in the industry, but trying to make the most with all of the advantages that current aspect-oriented paradigm offers.

In section two (II) we briefly present the AOP4ST's schema. The subsequent sections explain the evolution of concerns along the first phases of the SDLC: in Section three (III) we offer an explanation about how to find concerns in the business model, and Section four (IV) describes how to do the same in the user requirement model. Section five (V) presents how to define new concerns in the software requirements model. Finally, Section VI presents the conclusions and highlights some open issues and future work.

II. About AOP4ST

AOP4ST is a framework process; it is not a method nor a methodology. It covers the whole SDLC, but we are presenting here its structure for the early phases, commonly known as "early aspects" [3][4].

AOP4ST's name highlights two main concepts: a) the AOP, "Aspect-oriented Process", indicates that it is truly aspect-oriented, ensuring that can be reached the widely known benefits of this paradigm; b) the 4ST, "for a Smooth Transition", points to the possibility of applying this process in the industry immediately, because it employs widespread techniques, notations, standards, tools, etc. and allows to move to an aspect-oriented reality, taking advantage of the current state of the paradigm, until their own tools, techniques, etc. were imposed and completely accepted on the market.

The problem that AOP4ST tries to solve is how to bring the benefits of the aspect-oriented paradigm to the whole SDLC at the same time that are being used techniques, tools and standards currently widespread in the industry. In addition, the use of well-known techniques and tools allows incorporating this paradigm gradually, until the different existing proposals have enough diffusion and maturity to permit their employment in real and complex projects.

The whole SDLC includes the business model, generally not considered by the authors offering aspect-oriented approaches for the early stages of the SDLC. There are few incomplete proposals about aspect-oriented business modeling [5].

Our approach arises from several factors, which we have to face in the industry and in the adoption of new technologies for software development. First, it is well known that the different software development paradigms initially appear in the programming phase and then continue their definitions upstream, along with the SDLC [6][7]. The aspect-oriented paradigm follows the same pattern, that is why it is easier to find more proposals for the programming phase than for the early phases of the SDLC.

Second, many proposals about software development sound promising and offer benefits difficult to refuse, but their massive use in industry depends on many factors. A well-known case is the object-oriented databases, which beyond the benefits they offered and the enormous popularity of the object-oriented languages and development tools today, they have not at all achieved the leading role in an industry that could be expected [8].

Finally, software development projects have to deal with many risks, and the main function of project leaders is to minimize the damage that these risks can cause. The use of immature technologies, tools newcomers to the market, techniques that have not been tested enough, etc., would be very risky decisions to take by who has the responsibility to carry out a successful software development project. On the other hand, the availability of well-known tools and techniques and the adherence to standards and best practices will help professionals to make good estimates and to take better decisions.

AOP4ST is based on the hypothesis that it is possible to design an aspect-oriented software development process that encompasses techniques, tools, notations, and standards of widespread use in the current practice. This development process is suitable for the early stages of the SDLC, making the most with the benefits of the aspect-oriented paradigm in real-world settings. Under certain circumstances, it is possible to make use of existing techniques, tools, and standard notations, right now, while specific theoretical and practical instruments are developed and introduced in the market, achieving enough dissemination and support to justify its use in real software development projects.

AOP4ST's basic structure for early aspects is composed by three models: business model, user requirements model and software requirements model. The last one is divided into three views: functional, static and state views. Concerns can be progressively discovered along these models and views.

III. Concerns in The Business Model

The business model starts the process in AOP4ST, since we seek to have it as the first layer of the enterprise architecture [9]. In enterprise architecture, this layer has to pull on the rest of the models downstream to meet the business goals. Besides this, an early concern detection can help to improve the software robustness.

Five main aspect-oriented activities must be done in AOP4ST's business model: concern detection, concern separation, and modeling, composition rules checking, conflict resolution, aspect-oriented modeling.

After concern detection, the business model is composed by three kinds of processes: primary, support and management processes [10], and besides to them, the reusable processes. The last ones are processes that are not instantiated in themselves but are done from any of the first kinds of processes and can be shared. Regardless of the type of process to which they belong, each process is located within a specific package. All these packages correspond to the concerns that are detected in this first model and will host the concerns throughout the whole SDLC. In AOP4ST, each concern must be allocated in a package.

Reusable processes can be of two types: belonging to the domain of the problem and not belonging to it. In the first case, these are typical activities of the domain of the problem, that are repeated in several processes and which we, normally, could associate with functional requirements.

In the second case, these are activities that are independent of the problem domain and can be found even in different problem domains, e.g., access control, security, audit, logging, etc. They are, typically, quality attributes, also known as non-functional requirements (Fig. 1), that are tangled with the activities belonging to the problem domain and scattered along all the processes.

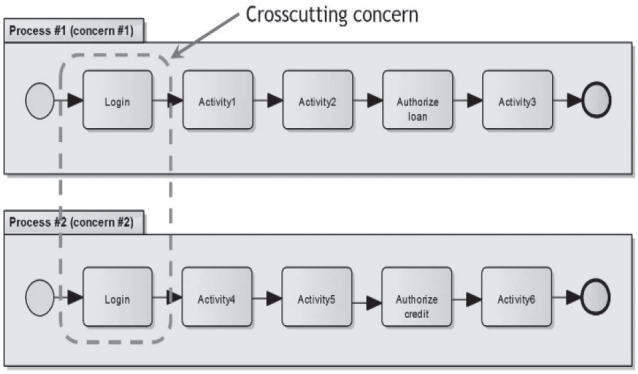

Fig. 1. Tangled and scattered crosscutting concerns.

To achieve a better success, we try to detect separately, the concerns corresponding to functional requirements and to non-functional ones. The former concerns are more difficult to be detected because they depend on the wording of the modeler. The latter are simpler, because they are clearly distinguishable from the activities belonging to the problem domain and, besides, it is possible to have a list of standard categories of non-functional requirements to follow in a systematic way. Manual or automated aspect mining techniques can be used to detect concerns [11].

Concerns must be placed into packages, along with the rest of the common elements of the model, and using a notation based on the proposal of Charfi et al. [12] although adjusted to use only elements belonging to the standard BPMN 2.0. This notation is also used to specify pointcuts, which are conditions that explicitly indicate join points. Join points are the points were the concerns will be composed again. In the base process, we use an annotation element to indicate the join point (Fig. 2), and the separated concern is modeled inside of a pool element. This pool includes a "Proceed" activity which represents the join points, and that indicates if the composition must be done before, after or around the join point (Fig. 3).

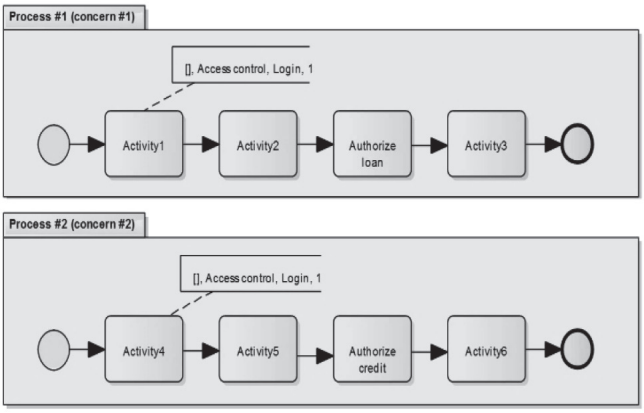

Fig. 2. Pointcut represented with annotation elements and indicating the join points.

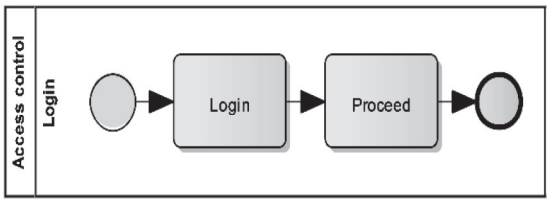

Fig. 3. Concern modeled and encapsulated, with a "Proceed" activity indicating the join points.

Since base processes and concerns are modeled within packages, the composition relationships and the relationships among concerns can be represented with a "concern model", build with UML package diagrams (Fig. 4).

In addition, it is possible to present a more detailed application of the concerns in the different join points by means of a "join point model," where the packages are presented as "white boxes", showing the join points inside (Fig. 5).

IV. The User Requirements Model in AOP4ST

The business model and the user requirements model crosscut the systems belonging to the organization. Each business process can describe activities that are supported by different systems. Similarly, the implementation of a user requirement could impact several systems, so this model of user requirements does not belong to a particular system, but to the global solution.

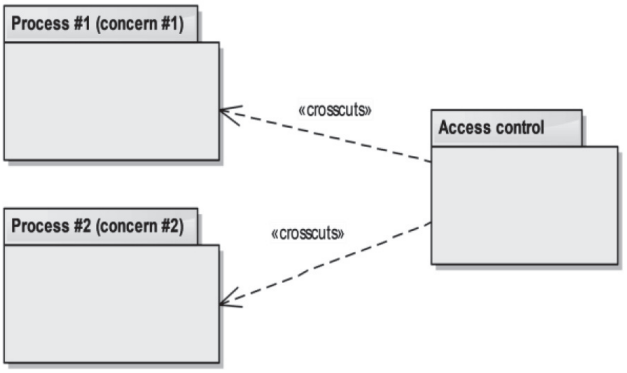

Fig. 4. Concern model.

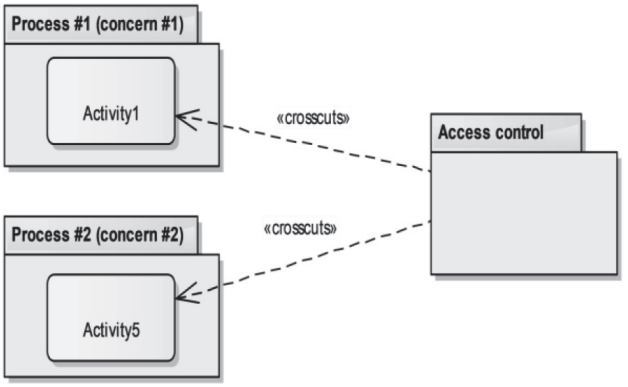

Fig. 5. Join point model.

In the business model, the processes were described by placing them into specific packages that correspond to concerns. The same packages existent in the business model must be copied into this model of user requirements so that the user requirements that are now detected are perfectly delimited to the process that requires them.

Functional user requirements will be easier to locate within a specific concern, while non-functional user requirements and business rules are more likely to be global, that is, to apply to several or even all concerns. They are usually referred to as crosscutting concerns. In the business model, several cross-cutting concerns had already been detected when we modeled processes not belonging to the domain of the problem, but in this model, will arise new crosscutting concerns (Fig. 6). These concerns also have an important influence among them, due to the positive and negative contribution relationships [13].

In addition to modeling the user requirements in the corresponding packages, it will be necessary to specify the relationships among them, so that the aspect-oriented models mentioned in the previous section (concern and join point models) are used again, now including the relationships among user requirements.

The analysis of each functional user requirements will allow finding new concerns at this stage. To do this, it is necessary to follow these rules:

- Requirement tracing against business model elements belonging to an only one concern: no new concern is added; the requirement is located into the package corresponding to the same concern coming from the previous model.
- Requirement tracing against business model elements belonging to more than one concern: it is a many-to-many relationship among concerns and requirements; we explain this below.
- Concerns with no requirements in the user requirement model: something did not work well since it is no possible to have empty packages; it is not possible not to find requirements for every business process.
- Concerns with only one user requirement: this situation may be caused because the requirement was set at a too high abstraction level (coarse-grained) and it needs a deeper analysis, or because the concern is too simple and should be revised whether it is a concern or not.

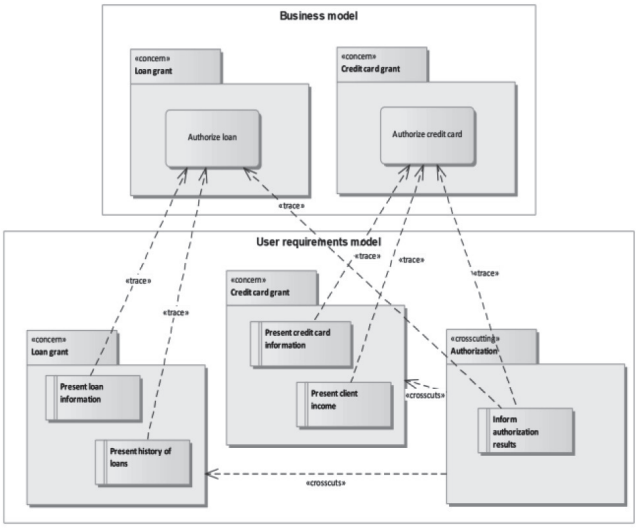

Fig. 6. Traceability between business and user requirements models.

There is no problem if we have many-to-many relationships among concerns and requirements. Of course, it is better to find one-to-many relationships [14], but in case this is not possible, the requirements must be allocated to the concern belonging to the most important business process. However, a deeper analysis should be done, because this situation is more frequent when requirements are defined at a high abstraction level. A greater granularity will improve the requirement allocation to an only one concern.

Non-functional requirements (quality attributes) will be allocated in the concern corresponding to each non-functional requirement category. Here, the problem is the positive and negative contribution relationships among them that it is necessary to analyze, because there could be an undesirable impact on the overall quality of the system if are defined non-functional requirements with too many negative contribution relationships among them [13].

Allocating business rules is easier: they must be classified into incumbencies following a criterion of homogeneity. A future analysis could be done, when the concerns can be split into new ones, depending of the degree of cohesion among the business rules sharing the same package.

Concerns found in this model are by no means the definitive ones. The next AOP4ST's model is the software requirement model, where user requirements are traced to use cases. Each use case will correspond to a concern, so it is clear that new concerns will arise in that model.

V. The Software Requirements Model in AOP4ST

We do not think that the requirement specification process will be performed in a cascade manner, but we need to describe it in any way. Of course, it can be iterative and incremental or whatever else it must be.

The use requirement must be specified in some way, and we have chosen uses cases, because we think it is possible to use them for different software development approaches: they can be developed through a traditional software development process, or their scenarios can be managed as user stories on a product backlog for agile development.

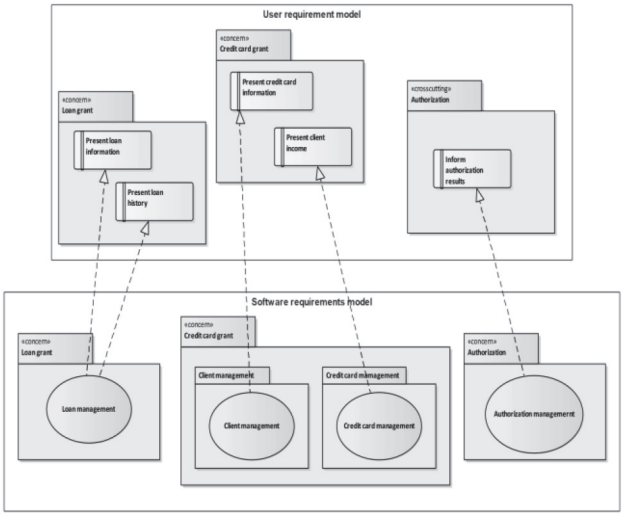

Fig. 7. Traceability between user requirements and software requirements models.

We have told that every concern is located into a package. Every package must be created again in this model, in order to continue the concern traceability and to allow the concern of discovering at this stage.

It is known that user requirements have a many-to-many relationship with use cases since these realize the first ones. However, in those cases where the relationship among them is one-to-many, we will be probably creating more than one use case for each user requirement. Moreover, since every single use case is considered a unique concern [15], we will be discovering new concerns at this phase of the SDLC. Every use case is located into a package, so we will continue having one package for each concern.

Fig.7 shows the creation of new concerns in the software requirements model. The three concerns present in the user requirements model are copied in the software requirements model. Next, user requirements are analyzed to find the use cases that must realize them. Two use cases are needed to realize the user requirements existent in "Credit card grant" concern. Since each use case is equivalent to a unique concern, we must create two new concern in this model: "Client management" and "Credit card management."

VI. Conclusions

This article presents the main ideas about how to find new concerns throughout the SDLC phases, according the AOP4ST approach. The main virtue of this proposal lies in the progressive concern appearance along the SDLC, without losing the focus on the goal of each SDLC phase.

The search of requirements and concerns are, in this way, an iterative and incremental work, where the finding and the improvement in the definition of one of them allow the finding and the improvement in the definition of the other ones.

Another very important outcome is the homogeneity and cohesion among the models, which allow for a coherent transition from one to another, enabling pre and post-requirement specification traceability and impact analysis. Moreover, another outstanding feature is that concerns are emerging naturally and progressively throughout the models.

So far, we have been able to perform some theoretical and practical validations of AOP4ST. From the theoretical point of view, we have submitted it for consideration to a symposium of doctoral theses [16], from which we have received very rich feedback. We have also applied the criteria established by Jalali [5] for the measurement of AOP4ST's business model, and the results placed it in a privileged position [17].

Regarding the practical validation, we were able to test the AOP4ST's models separately in several companies. AOP4ST was fully applied to re-modeling a whole model of one of the most important biochemical laboratories in Argentina, and at this moment, we are successfully applying it in the modeling of a beverage company. However, we believe that there is still much effort needed to validate AOP4ST and to achieve a greater maturity in the architectural and test models. We are now working on these issues. For Spanish-speaking readers, it is possible to find more information about AOP4ST in [16], [18], and about AOP4ST's business model in [19] and [20].

Referencias

[1] Kiczales G. et al., "Aspect-Oriented Programming". ACM Comput. Surv., vol. 28, no. June, pp. 220–242, 1997.

[2] Magableh, A., Shukur, Z. and Ali, N. M. "Systematic review on aspect-oriented UML modeling: A complete aspectual UML modeling framework," J. Appl. Sci., vol. 13, no. 1, pp. 1–13, 2013.

[3] Bakker, J., Tekinerdogan, B. and Aksit, M. "Characterization of Early Aspect Approaches," Early Asp. Asp. Requir. Eng. Archit. Des. Work., p. 7, 2005.

[4] Rashid, A., Moreira, A. and Tekinerdogan, B. "Early aspects: aspect-oriented requirements engineering and architecture design," Software, IEE Proc., vol. 151, no. 4, pp. 153–155, 2004.

[5] Jalali, A. "Assessing Aspect Oriented Approaches in Business Process Management," in Perspectives in Business Informatics Research, 13th International Conference, BIR 2014, pp. 231–245, 2014.

[6] Capretz, L. F. "A brief history of the object-oriented approach," SIGSOFT Softw. Eng. Notes, vol. 28, no. 2, p. 6, 2003.

[7] "History of Programming Languages," in History of Programming Lang. Conf.

[8] Leavitt, N. "Whatever Happened to Object-Oriented Databases?"

Computer (Long. Beach. Calif)., vol. 33, no. 8, pp. 16–19, 2000.
[9] Greefhorst, D. and Proper, E. Architecture Principles. The Cornerstones of Enterprise Architecture, vol. 6, no. 3, 2011.
[10] ABPMP, BPM CBOK V.3.0 - Business Process Management BPM Common Body of Knowledge, 2013.
[11] Pinciroli, F. "Considerações acerca da mineração de aspectos," Perspect. em Ciências Tecnológicas, vol. 5, no. 5, pp. 83–101, 2016.
[12] Charfi, A., Müller, H. and Mezini, M. "Aspect-oriented business process modeling with AO4BPMN," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 6138 LNCS, pp. 48–61, 2010.
[13] Pinciroli, F. "Improving software applications quality by considering the contribution relationship among quality attributes". Procedia Comput. Sci. 3rd Int. Work. Comput. Antifragility Antifragile Eng. (ANTIFRAGILE 2016), vol. 83, pp. 970–975, 2016.
[14] Clarke, S. and Baniassad, E. Aspect-oriented analysis and design. The Theme Approach. Boston: Addison-Wesley, 2005.
[15] I. Jacobson and P. Ng, Aspect-oriented software development with use cases. Addison-Wesley, 2005.
[16] Pinciroli, F. "Aspect-Oriented Process for a Smooth Transition," in Ph.D. Symposium of the IEEE 11 Congreso Colombiano de Computacion, 2016.
[17] Pinciroli, F. and Barros Justo, J.L. Early aspects in "Aspect-Oriented Process for a Smooth Transition". Accepted article at CACIC 2017, not yet published (http://cacic2017.info.unlp.edu.ar/).
[18] Pinciroli, F. "AOP4ST – Aspect-Oriented Process for a Smooth Transition," in WICC 2015 - XVII Workshop de Investigadores en Ciencias de la Computación, 2015.
[19] Pinciroli, F. and Zeligueta, L. "El modelo de negocio en AOP4ST," in WICC 2016 - XVIII Workshop de Investigadores en Ciencias de la Computación, 2016.
[20] Pinciroli, F. and Zeligueta, L. "Modelado de negocios orientado a aspectos con AOP4ST," in WICC 2017 - XIX Workshop de Investigadores en Ciencias de la Computación, 2017.

Fernando Pinciroli nació en Buenos Aires, Argentina, en 1965. Es Licenciado en Sistemas y Computación por la Universidad Católica Argentina y candidato a Dr. en Ciencias de la Informática en la Universidad Nacional de San Juan. Es Decano de la Facultad de Informática y Diseño de la Universidad Champagnat (Mendoza, Argentina) y miembro del grupo de investigaciones en ingeniería de software del Instituto de Investigaciones de esa misma facultad. Dictó cursos de posgrado y alrededor de 250 cursos y conferencias en más de veinte países. Publicó más de 40 artículos científicos en numerosos eventos internacionales y es coautor de tres libros publicados en España, Argentina e Italia. Participó en la definición del UML a través del Object Technology User Group. Es consultor en ingeniería de software desde 1993 y actualmente es el presidente de Solus S.A., en Mendoza, Argentina, y presidente de Sparx Systems Argentina, compañía hermana de Sparx Systems Ltd. Pty. de Australia, ya que produce y comercializa la versión en español de la herramienta Enterprise Architect en el mundo hispanoparlante. Contacto: pincirolifernando@uch.edu.ar.