Proceso marco orientado a aspectos en las etapas tempranas del ciclo de vida del desarrollo de software para una transición en la industria

Fernando Pinciroli
Instituto de Investigaciones de la Facultad de Informática y Diseño
Universidad Champagnat, Mendoza, Argentina
pincirolifernando@uch.edu.ar

Director: Dr. Raymundo Forradellas, U.N. de San Juan, U.N. de Cuyo, Argentina
Codirector: Dr. José Luis Barros Justo, Universidad de Vigo, España

Doctorado en Ciencias de la Informática
Universidad Nacional de San Juan, Argentina
Fecha de defensa: 19 de noviembre de 2020
Tribunal: Dr. Luis Olsina, Universidad Nacional de La Pampa, Argentina
Dr. Santiago Matalonga, University of the West of Scotland, Escocia
Dr. Leandro Antonelli, Universidad Nacional de La Plata, Argentina

1. Introducción

Por una parte, se halló evidencia de la escasa presencia del paradigma de aspectos en la industria, pero, por otro lado, también se observó que sus beneficios, largamente mencionados en la literatura, sí se pudieron alcanzar en aquellos casos en los que se utilizó la orientación a aspectos en el mundo real, más allá de los ámbitos académicos [1]. Al mismo tiempo, esa evidencia también mostró que las propuestas existentes son incompletas y muy pocas llegan a cubrir tan solo dos fases del ciclo de vida del desarrollo de software (en adelante, SDLC) [1]. Por esto es que surgió la motivación de elaborar una alternativa metodológica que permitiera su aplicación de inmediato en proyectos e iniciativas en el mundo real.

Así, el objetivo de nuestro trabajo consistió en definir un proceso marco para las etapas tempranas del ciclo de vida del desarrollo de software, desde el modelo de negocios hasta la especificación completa de requisitos de software y empleando el paradigma de la orientación a aspectos. A la vez, se buscó propiciar el empleo en la industria de este paradigma para obtener sus beneficios, al aprovechar las herramientas y técnicas estándares disponibles actualmente en el mercado, mientras se siguen desarrollando otras específicas y alcanzan la madurez suficiente. Por tal razón, se decidió llamar a esta propuesta AOP4ST, sigla derivada de Aspect-Oriented Process for a Smooth Transition [2].

Se trata de un proceso marco, no específico, de modo de permitir su empleo con diferentes modelos del ciclo de vida del desarrollo de software a lo largo de sus etapas tempranas y hasta obtener una especificación de requisitos completa y coherente, incluyendo tres vistas: funcional, estática y de estados. Este proceso emplea herramientas y técnicas estándares, de amplia difusión en la industria, para facilitar su adopción inmediata y, también, utiliza notaciones estándares, para permitir elaborar modelos y especificaciones comprensibles y no ambiguas, que puedan contar con el soporte de las herramientas de software actualmente disponibles en el mercado. Se procura que esta alternativa sea completamente orientada a aspectos, que facilite la obtención de las incumbencias en forma progresiva a lo largo de todos los modelos y, al mismo tiempo, las mantenga siempre separadas y asegurando la trazabilidad bidireccional entre ellas. Estas incumbencias deben obtenerse en forma natural a lo largo de todos los

modelos, de manera de no afectar los objetivos propios de cada uno de ellos y, de este modo, potenciar los beneficios que se esperan en cada modelo mediante el empleo del paradigma de aspectos.

En la sección 2 se presenta la motivación de este trabajo, incluyendo el estado de la cuestión y la problemática que se pretende resolver, en la sección 3 se describe la solución diseñada para dar solución a los inconvenientes mencionados y los aportes a la disciplina y, finalmente, la sección 4 presenta las líneas de investigación que quedan abiertas a partir de este trabajo.

2. Motivación

2.1. Estado de la cuestión

A lo largo de la historia del desarrollo de software se produjeron numerosos cambios paradigmáticos que, paulatinamente, fueron adoptados por la industria, por ejemplo, la programación imperativa, la descomposición funcional, la orientación a objetos. Estos paradigmas nacieron en ambientes de investigación y desarrollo, para luego ser ofrecidos a través como productos de mercado [3].

La orientación a aspectos nació en 1996 con la propuesta de Gregor Kiczales, denominada Programación Orientada a Aspectos (*Aspect-Oriented Programming, AOP*) [4]. Su objetivo principal consistió en mejorar la división del código en módulos, para dar solución a los problemas de desparramo (*scattering*) y enredo (*tangling*) de las funciones que se encuentran diseminadas a lo largo de todo un sistema y entremezcladas con otras porciones de código. A estas funciones se las conoce como incumbencias transversales (*crosscutting concerns*) [5]. Con el paso del tiempo aparecieron el diseño orientado a aspectos, el análisis de sistemas orientado a aspectos y, finalmente, se denominó Desarrollo de Software Orientado a Aspectos (Aspect-Oriented Software Development) a su empleo en todas las fases del SDLC. Para el caso particular de la aplicación de la orientación a aspectos en las fases tempranas del SDLC, se emplea el nombre de “aspecto temprano” (*early aspect*), que es “una incumbencia que corta transversalmente una descomposición dominante de artefactos o módulos base derivados del criterio de separación dominante de incumbencias, en las primeras etapas del ciclo de vida del software” [6].

La separación de incumbencias es un principio muy importante de la ingeniería de software [7-9]. Esta posibilidad de contar con una mayor modularidad ofrece una serie de beneficios a lo largo de todas las fases del SDLC: comprensibilidad [10-17], reusabilidad [10-14] [17-18], mantenibilidad [10-11] [15-20], reducción de la complejidad [12] [21-25], capacidad de evolución [16] [25-27], extensibilidad [10] [17], flexibilidad [10] [28], reducción en los costos del desarrollo [20] [27], escalabilidad [25] [29], adaptabilidad [13], calidad [26], entre otros.

Estas ventajas son alentadoras e, incluso, aplicables a todas las etapas del SDLC [30-32], pero su explotación en la industria todavía no se evidencia claramente. Aún no existen propuestas lo suficientemente maduras y con una presencia real en el mercado, a través de productos concretos y con soporte de parte de la industria [1]. Esta escasez de propuestas acarrea la falta de aplicación efectiva del paradigma de aspectos en aquel ámbito, aunque en algunas empresas puedan observarse implementaciones en las que se realiza la separación de algunas incumbencias transversales correspondientes a requisitos no funcionales (incumbencias asimétricas) y, en la mayoría de estos casos, sin emplear lenguajes orientados a aspectos [33].

Si bien en el ámbito académico existen numerosas propuestas, su implementación y aplicación están restringidas actualmente al ámbito privado [1]. Por ejemplo, el MIT había pronosticado en 2001 que la programación orientada a aspectos sería una de las diez tecnologías emergentes que cambiarían el mundo y, sin embargo, doce años después prácticamente todavía no se había adoptado [34]. En un estudio de Muñoz et al. [35] se menciona que, casi al momento de cumplirse la previsión del MIT, solamente se

había utilizado programación orientada a aspectos en el 0,5% de los proyectos escritos en Java, subidos al repositorio open source sourceforge.net, entre 2001 y 2008. Por lo tanto, la orientación a aspectos se presenta hoy más como un área de investigación, que necesita ser explotada con mayor profundidad [1]. Esta es la razón que motiva a buscar un enfoque que permita acelerar la adopción del paradigma en el ámbito industrial.

2.2. Dificultades para aplicar las propuestas disponibles

El empleo de la orientación a aspectos en proyectos reales en la industria, tal como se la encuentra disponible actualmente, llevaría a enfrentar algunos obstáculos, impedimentos y dificultades importantes:

1. No existe una propuesta para el SDLC completo. Para cubrirlo totalmente deberían tomarse alternativas de diferentes autores [1] [36-38]. Pero este collage derivaría en un método heterogéneo y prácticamente inaplicable. Además, las alternativas actualmente ofrecidas y más difundidas no abarcan siquiera más de dos fases del SDLC [1].
2. Las propuestas disponibles no cuentan con una difusión suficiente en la industria debido a que en muchos casos se emplean notaciones no estándares [1]; la mayor parte de las propuestas no cuentan con herramientas de modelado que las soporten o, si las poseen, no son asequibles en el mercado ni tienen el soporte técnico necesario [1] [39]; y, por todo esto, se puede concluir que no existe una masa crítica de profesionales formados que puedan incorporarse rápidamente a un equipo de proyecto.
3. Si no hay aplicación en el mundo real, entonces no hay madurez suficiente en técnicas necesarias para conducir adecuadamente los proyectos. Por ejemplo, las técnicas de estimación requieren contar con una base estadística importante para poder asegurar un grado de certeza razonable en sus resultados [40].
4. La aceptación comercial o no de una técnica o producto gravita considerablemente en su posibilidad de uso en el mundo real. Un caso emblemático es el de las bases de datos orientadas a objetos [41], que nunca fueron adoptadas suficientemente por la industria y perteneciente a un paradigma ampliamente difundido y aceptado, no pasaron de ser materia de discusión e implementadas en algunos productos comerciales que no alcanzaron el éxito esperado. Y no hay suficiente evidencia de la aplicación de la orientación a aspectos en la industria [1].

3. Aporte a la disciplina

3.1. Objetivos de la solución

Los objetivos generales de nuestra tesis, junto con sus respectivos objetivos específicos, son:

1. Describir un nuevo proceso marco para las etapas tempranas del SDLC, desde el modelo de negocios hasta la especificación de requisitos de software completa.
   a. Ofrecer un proceso marco, no específico y liviano, de modo que permita su empleo con diferentes modelos del SDLC.
   b. Cubrir las etapas tempranas del SDLC, desde el modelado de negocio hasta la obtención de una especificación de requisitos completa y coherente.
   c. Emplear herramientas y técnicas estándares, de amplia difusión en la industria, para facilitar su adopción inmediata.
   d. Emplear notaciones estándares, para lograr modelos y especificaciones comprensibles y no ambiguas, que puedan contar con soporte de herramientas de software disponibles en el mercado.
2. Describir ese proceso marco para las etapas tempranas del SDLC empleando el paradigma de la orientación a aspectos.
   a. Ofrecer una alternativa que sea completamente orientada a aspectos.

b. Obtener las incumbencias en forma progresiva a lo largo de todos los modelos.
c. Mantener la separación de incumbencias a lo largo de todos los modelos.
d. Mantener la trazabilidad bidireccional de las incumbencias de punta a punta.
e. Obtener las incumbencias en forma natural a lo largo de todos los modelos, de manera de no afectar la obtención de los objetivos de cada modelo.
f. Potenciar los beneficios que se esperan de cada modelo mediante el empleo del paradigma de aspectos.

3.2. Características de la solución

La propuesta descripta en nuestra tesis doctoral consiste en la definición de un proceso marco genérico e independiente de los modelos del SDLC, que cubre las etapas tempranas de ese ciclo de vida con los siguientes modelos: de procesos de negocio, de requisitos de usuario y de requisitos de software; este último constituido por tres vistas: de procesos, estática y de estados. El modelo de procesos de negocio permite el gobierno de TI con soluciones de Arquitectura Empresarial; el de requisitos de usuario da soporte a las dos partes de la ingeniería de requisitos: el desarrollo y la administración de los requisitos, además de la gestión de la demanda para la etapa de mantenimiento; y el modelo de requisitos de software está constituido por tres vistas complementarias: de procesos, estática y de estados, lo que posibilita una definición completa de los requisitos de software (Figura 1).

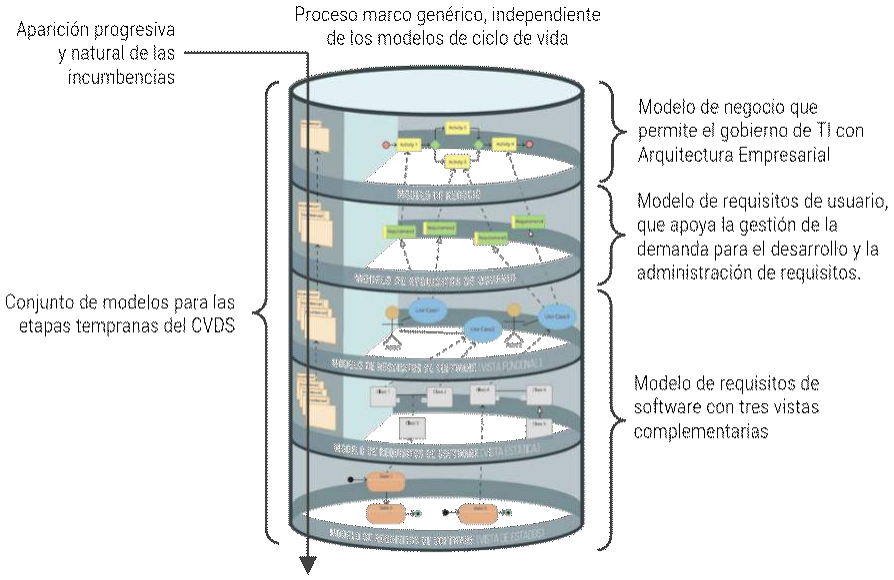

Figura 1. Esquema de AOP4ST: modelos y vistas.

En AOP4ST las incumbencias se detectan en forma progresiva y se mantienen perfectamente separadas a lo largo de los diferentes niveles de abstracción, acrecentándose en número y aumentando en granularidad, además de mantener una trazabilidad bidireccional entre ellas. Esta trazabilidad bidireccional también se mantiene entre los elementos constitutivos de cada uno de los modelos. Se puso especial énfasis en el empleo de notaciones, herramientas y técnicas ampliamente difundidas en la industria [1]. Estas cuestiones permiten que se ponga el foco en los objetivos que se esperan en cada una de las fases del SDLC, con el objetivo de hacer software de calidad, y no distraerse con las exigencias que impone el paradigma de aspectos, sino que, por el contrario, este enfoque sea de verdadera ayuda y un instrumento para alcanzar aquel objetivo (Figura 2).

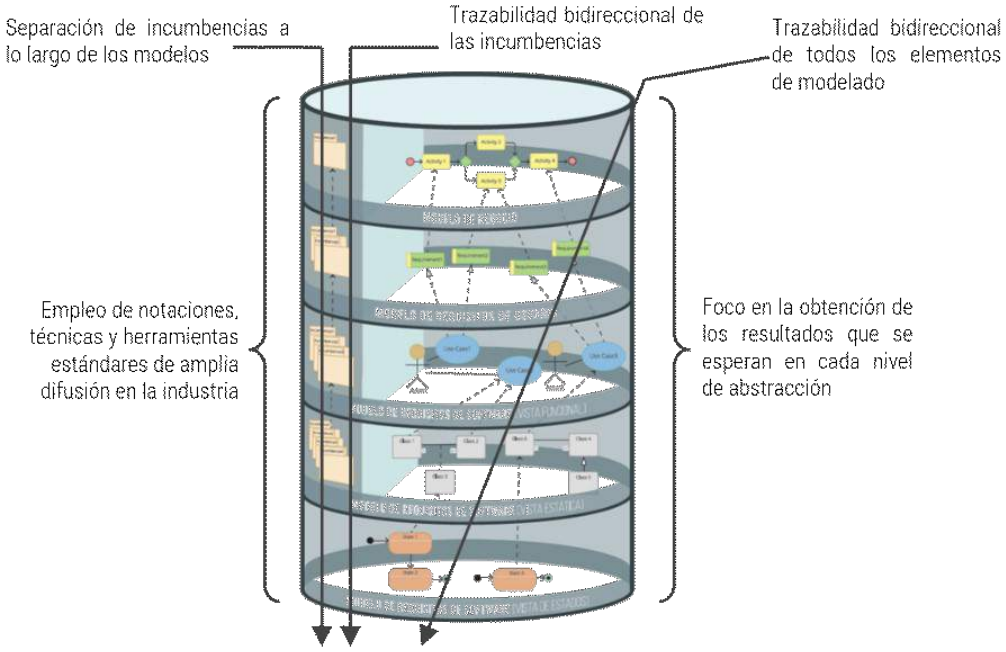

Figura 2. Esquema de AOP4ST: separación de incumbencias y trazabilidad.

Igualmente, la orientación a aspectos exige realizar ciertas actividades para las que se propusieron herramientas y técnicas para reducir su impacto, a la vez que se diseñaron modelos específicos que, además de que se pueden generar en forma automática por medio de herramientas de software, permiten analizar la calidad de los modelos y aportar información de valor para la mejor toma de decisiones (Figura 3).

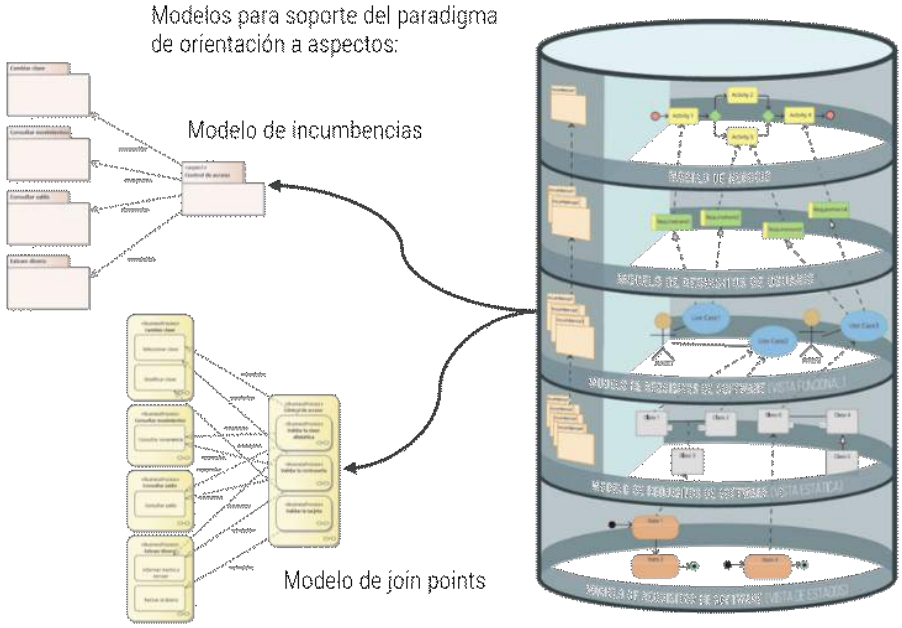

Figura 3. Esquema de AOP4ST: modelos y mecanismos para la administración de los aspectos.

Algunas características destacadas de AOP4ST hacen que ocupe una posición de privilegio en el concierto de las propuestas existentes para el empleo de la orientación a aspectos en el modelo de negocio (Figura 4) según el método de evaluación propuesto por Jalali [42].

Propuesta	Grupos de requisitos						Total general
	Requisitos básicos	Exposición de firma	Composición de reglas	Relaciones de los advices	Patrones de transformación	Soporte de fases	
AOP4ST	**3 de 3**	**3**	**3,33**	**4**	**1**	**0,8**	**12,13**
Jalali et al.	3 de 3	1	2	2,67	2	3,2	10,87
Cappelli et al.	3 de 3	1	2,67	2	1	0,8	7,47
Jabeen et al.	3 de 3	1	0,67	2,67	1	0,8	6,14
AO4BPMN	3 de 3	1	0,67	2,67	1	0,8	6,14
Patiniotakis et al.	3 de 3	1	0,67	2	1	0,8	5,47
AO4BPEL	2 de 3	0	0	0	0	0	0
Colleli	2 de 3	0	0	0	0	0	0
AOBPMN	2 de 3	0	0	0	0	0	0
Wang et al.	1 de 3	0	0	0	0	0	0
Shankardass	1 de 3	0	0	0	0	0	0

Figura 4. Ubicación de AOP4ST con respecto al resto de las propuestas de aspectos existentes para el modelo de negocio.

Todos los modelos contemplan la detección, separación y encapsulamiento de las incumbencias, el establecimiento de las relaciones entre los elementos del mismo nivel de abstracción y con elementos del nivel superior, la posterior composición de las incumbencias y la resolución de conflictos tras la composición, además de prácticas de modelado recomendadas. Además cuenta con un conjunto de reglas de composición de incumbencias en el modelo de negocio; con el empleo de las técnicas de análisis de las relaciones de contribución positivas y negativas entre requisitos de usuario; con la administración de requisitos de software en su vista funcional que permite el empleo de diferentes técnicas de especificación (por ejemplo, casos de uso e historias de usuario); con las técnicas de composición de las clases minimizando los conflictos; y con la composición de las máquinas de estados a partir de la composición de las clases.

3.3. Validaciones

La investigación en el ámbito de la ingeniería de software exige el estudio empírico de los fenómenos que suceden en el mundo real, entre los que se encuentra el desarrollo de nuevos modelos de proceso [43], tema central de este trabajo. De los métodos de investigación empíricos que se emplean en la ingeniería de software fueron utilizados dos métodos cualitativos y uno cualicuantitativo:

- **Caso de estudio:** en el que se evaluó la aplicación de AOP4ST en un proyecto real completo, pero un tiempo después de la culminación de ese proyecto (post-mortem análisis [44]).
  Este caso de aplicación fue el remodelado, usando AOP4ST, de todos los procesos del Laboratorio Hidalgo, de la provincia de Buenos Aires, que se habían llevado a cabo con el empleo de un enfoque orientado a objetos y con UML. Los modelos son complejos y, los más de ochenta procesos punta a punta modelados, describen los procesos de funcionamiento del laboratorio para el procesamiento de todas las determinaciones de análisis bioquímicos, tanto para los análisis corrientes como para los correspondientes a la investigación de nuevas drogas para laboratorios de Norteamérica. Los modelos originales fueron auditados por una empresa italiana, que debía asegurar el cumplimiento con las normas establecidas por la Food and Drug Administration (FDA) de los Estados Unidos.
- **Proyectos de investigación-acción:** en la que investigadores y profesionales trabajaron en forma combinada utilizando las técnicas de AOP4ST en tres proyectos reales en la industria, aunque aplicándolas solo a algunos modelos de la propuesta.
  El primero de los casos fue el modelado de los procesos corporativos de la empresa Prosegur, en donde se detectaron, separaron, encapsularon y compusieron las incumbencias del modelo de negocio del área de Vigilancia en tres países de la compañía: España, Brasil y Argentina. A partir de estos procesos de negocio se detectaron los requisitos de usuario, tal como se propone en este trabajo, y, con ellos, se elaboraron los nuevos procesos corporativos y se estableció el alcance del software a adquirir para dar soporte a tales procesos.

El segundo caso correspondió al modelado de los procesos de la Dirección de Estadísticas e Investigaciones Económicas del Gobierno de la Provincia de Mendoza, ejecutado durante 2019, y en donde nuevamente se realizaron la detección, separación, encapsulamiento y composición de los procesos de negocio.
El tercer proyecto en el que pudieron emplearse las ideas que se presentan en el capítulo dedicado al modelo de negocio de AOP4ST consistió en el modelado de los procesos internos de la empresa de gestión de transacciones bancarias Red Link, en Buenos Aires, proyecto iniciado a finales de 2019 y aún en marcha.

- **Investigación secundaria:** se llevó a cabo un estudio de mapeo sistemático para validar las hipótesis de trabajo y aportar evidencia científica a las afirmaciones vertidas en la tesis, dado que las revisiones sistemáticas son uno de los bloques de construcción clave de la Ingeniería de Software Basada en Evidencia (EBSE) [43]. El protocolo de este estudio fue presentado en un evento en la Argentina en 2018 [45] y el mapeo sistemático fue publicado por la revista científica "Journal of King Saud University – Computer and Information Sciences" [1].

Adicionalmente, las diferentes ideas de AOP4ST fueron desarrolladas a lo largo de cinco proyectos de investigación consecutivos, en los que se produjo una veintena de publicaciones, la mayoría de ellas en congresos y revistas internacionales:

	Proyecto	Institución	Período	Avances y publicaciones
1	"Procesos de desarrollo de software de calidad basados en aspectos"	Universidad Tecnológica Nacional FRM	2010 a 2012	Sentó las bases principales de un proceso marco de desarrollo de software que permita aprovechar los beneficios de la orientación a aspectos [46].
2	"Detección temprana de aspectos en el modelado de negocios y el desarrollo de los requisitos"	Instituto de investigaciones de la Facultad de Informática y Diseño de la Universidad Champagnat	2013 a 2015	Se definieron los modelos y vistas de AOP4ST [2].
3	"Modelado de procesos de negocio orientados a aspectos con BPMN"		2015 a 2017	Abordó el modelo de negocio en AOP4ST [47-50] y se obtuvo la principal evidencia con un estudio de mapeo sistemático [1] [45].
4	"Ingeniería de requisitos de usuario en AOP4ST"		2018 a 2020	Se centró en el modelo de requisitos de usuario de AOP4ST [51-57].
5	"Ingeniería de requisitos de software orientada a aspectos en AOP4ST"		2020 a 2022	Se está trabajando sobre los requisitos de software, con énfasis en la vista estática [58].

Las ideas de AOP4ST también fueron aceptadas y expuestas en el "Simposio de Tesis Doctorales del IEEE – 11 Congreso Colombiano de Computación (11 CCC)" llevado a cabo en Popayán, Colombia, en 2016 [59], donde se recibieron opiniones muy enriquecedoras de parte del tribunal, que se plasmaron en la tesis.

4. Líneas de investigación futuras

Se considera de especial interés continuar las investigaciones sobre la detección de patrones, para el análisis de impacto ante cambios y para la resolución de conflictos. La evidencia obtenida de la aplicación de la orientación a aspectos en la industria también da cuenta de esta necesidad [1]. Sería oportuno dar un mayor tratamiento al modelo de requisitos de usuario, para abordar requisitos para sistemas más específicos. En nuestra tesis se trabajó sobre sistemas de propósito general, pero también se entiende que sería de mucho valor el considerar requisitos de sistemas de tiempo real, de uso intensivo de interfaz humano-computador, de computación paralela, de ingeniería, etc.

Un avance importante sería, también, el desarrollo de un soporte más específico de aplicaciones de software. Los lenguajes estándares empleados permitirán la interpretación de los diagramas, ya que en el modelo de negocio se utilizó BPMN que cuenta con BPEL WS-BPEL, y en el resto de los modelos se empleó el UML, que hace uso del OCL, también estándar, para su formalización. Además, se harían grandes aportes con la simulación para analizar la integridad de los modelos tras la composición, especialmente con los diagramas de comportamiento, como son los diagramas de procesos de negocio, los diagramas de actividades y de secuencias, que pueden obtenerse de las especificaciones estructuradas de los casos de uso, y los diagramas de estados, con la inclusión de variables de control para análisis de escenarios. También se observa necesario adaptar a este paradigma las técnicas de estimaciones de software existentes, evaluar el impacto en los roles de los integrantes de los equipos de proyecto a partir de las actividades que surgen del empleo de la orientación a aspectos, especialmente con el empleo de metodologías ágiles para el desarrollo de software [60].

Referencias

[1] Pinciroli, F., Barros Justo, J.L. and Forradellas, R. (2020). "Systematic Mapping Study: on the coverage of aspect-oriented methodologies for the early phases of the software development life cycle". In: Elsevier, Journal of King Saud University – Computer and Information Sciences.

[2] Pinciroli, F. (2015). AOP4ST – Aspect-Oriented Process for a Smooth Transition. In: Proceedings of the XVII Workshop de Investigadores en Ciencias de la Computación, WICC 2015, Salta.

[3] Gabbrielli, M. and Martini, S. (2010). Programming languages: principles and paradigms. Springer, London.

[4] Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Videira Lopes, C., Loingtier, J.M. and Irwin, J. (1997). Aspect-Oriented Programming. In: Proceedings of the European Conference on Object-Oriented Programming, ECOOP, Finland. Springer-Verlag, Lecture Notes in Computer Science, no. 1241.

[5] Wimmer, M., Schauerhuber, A., Kappel, G., Retschitzegger, W., Schwinger, W. and Kapsammer, E.E. (2011). A survey on UML-based aspect-oriented design modeling. ACM Computing Surveys, vol. 43, no. 4, pp. 1–33.

[6] Baniassad, E., Clements, P.C., Araújo, J., Moreira, A., Rashid, A. and Tekinerdoğan, B. (2006). Discovering early aspects. In: IEEE Software, vol. 23, pp. 61–70.

[7] Parnas, DL. (1972). "Information distribution aspects of design methodology". In: Communications of the ACM, vol.15, no.12, pp.1053-1058.

[8] Dijkstra, E.W. (1982). On the role of scientific thought. In: Selected Writings on Computing: A Personal Perspective, Springer-Verlag, New York, pp. 60–66.

[9] Ye, S. and He, C. (2013). A comparison of methods for identification of early aspects. In: Proceedings of the 3rd International Conference on Computer Science and Network Technology, ICCSNT 2013, pp. 275-279.

[10] Mehmood, A. and Jawawi, D.N.A. (2014). A quantitative assessment of aspect design notations with respect to reusability and maintainability of models. In: Proceedings of the 8th Malaysian Software Engineering Conference, MySEC 2014, Langkawi, pp. 136-141.

[11] Witteborg H., Charfi A., Colomer Collell D., Mezini M. (2014). Weaving Aspects and Business Processes through Model Transformation. In: Villari M., Zimmermann W., Lau KK. (eds) Service-Oriented and Cloud Computing, ESOCC 2014, Lecture Notes in Computer Science, vol 8745, Springer, Berlin, Heidelberg.

[12] Faßbender, S., Heisel, M. and Meis, R. (2015). A Problem-, Quality-, and Aspect-Oriented Requirements Engineering Method. In: Communications in Computer and Information Science, pp. 291–310.

[13] FanJiang Y.Y., Kuo, J.Y., Ma, S.P., Huang, W.R. (2010). An Aspect-Oriented Approach for Mobile Embedded Software Modeling. In: Taniar D., Gervasi O., Murgante B., Pardede E., Apduhan B.O. (eds) Computational Science and Its Applications, ICCSA 2010, Lecture Notes in Computer Science, vol 6017, Springer, Berlin, Heidelberg, pp. 257–272.

[14] Wang, Y., Singh, S., Hosking, J. and Grundy, J. (2006). An aspect-oriented UML tool for software development with early aspects. In: Proceedings of the 2006 International Workshop on Early Aspects at ICSE, EA '06, pp. 51-58.

[15] Liu, X, Liu, S. and Zheng, X. (2009). Adapting the NFR Framework to Aspectual Use-Case Driven Approach. In: Proceedings of the Seventh ACIS International Conference on Software Engineering Research, Management and Applications, Haikou, pp. 209-214.

[16] Mosconi, M., Charfi, A., Svacina, J. and Wloka, J. (2008). Applying and evaluating AOM for platform independent behavioral UML models. In: Proceedings of the 2008 AOSD Workshop on Aspect-Oriented Modeling, AOM '08, pp. 19–24.
[17] Zhou, Y., Lei, G., Li, P. and Kong, L. (2008). Realizing extension use cases with AOP. In: Proceedings of the 2008 IEEE International Symposium on IT in Medicine and Education, Xiamen, 2008, pp. 1040-1044.
[18] Jalali, A. (2015). Static Weaving in Aspect Oriented Business Process Management. In: Proceedings of the Conceptual Modeling, 34th International Conference, ER 2015, Stockholm, Springer, pp. 548–557.
[19] De Oliveira, K.S., Franca, J.M.S. and Soares, M.S. (2013). Extensions of SysML for Modeling an Aspect Oriented Software Architecture with Multiple Views. In: Proceedings of the 10th International Conference on Information Technology: New Generations, Las Vegas, pp. 680–685.
[20] Budwell, C.C. and Mitropoulos, F.J. (2008). The SLAI Methodology: An Aspect-Oriented Requirement Identification Process. In: Proceedings of the 2008 International Conference on Computer Science and Software Engineering, vol. 2, pp. 296–301.
[21] Jalali, A., Maggi, F.M. and Reijers, H.A. (2015). Enhancing AO-BPM through declarative rules. In: Proceedings of the Conceptual Modeling, 34th International Conference, ER 2015, Stockholm, pp. 108–118.
[22] Jalali, A., Ouyang, C., Wohed, P. and Johannesson, P. (2017). Supporting aspect orientation in business process management. In: Software and Systems Modeling, vol. 16, no. 3, pp. 903–925.
[23] Mehner, K., Monga, M. and Taentzer, G. (2006). Interaction Analysis in Aspect-Oriented Models. In: Proceedings of the 14th IEEE International Requirements Engineering Conference, RE'06, pp. 69–78.
[24] Amirat, A., Mohamed, L. and Khammaci, T. (2008). Modularization of Crosscutting Concerns in Requirements Engineering. In: The International Arab Journal of Information Technology, vol. 5, no. 2, pp. 120-125.
[25] Amálio, N., Kelsen, P., Ma, Q. and Glodt, C. (2010). Using VCL as an Aspect-Oriented Approach to Requirements Modelling. In: Transactions on Aspect-Oriented Software Development VII: A Common Case Study for Aspect-Oriented Modeling, pp. 151–199.
[26] Conejero, J.M., Hernández, J., Jurado, E. and van den Berg, K. (2010). Mining early aspects based on syntactical and dependency analyses. In: Science of Computer Programming, vol. 75, no. 11, pp. 1113–1141.
[27] Singh, N. and Singh Gill, N. (2012). Towards an Integrated AORE Process Model for Handling Crosscutting Concerns. In: International Journal of Computer Applications, vol. 37, no. 3, pp. 18–24.
[28] Keriakos, M., Hosny, H. and Aly, S.G. (2012). Context aware business process aspect modeler. In: Proceedings of the 2nd International Conference on Pervasive Embedded Computing and Communication Systems, PECCS-2012, pp. 206-213.
[29] Bošković, M., Mussbacher, G., Bagheri, E., Amyot, D., Gašević, D. and Hatala, M. (2011). Aspect-Oriented Feature Models. In: Dingel J., Solberg A. (eds) Models in Software Engineering, MODELS 2010, Lecture Notes in Computer Science, vol 6627, Springer, Berlin, Heidelberg, pp. 110–124.
[30] Rashid A. and Moreira A. (2006) Domain Models Are NOT Aspect Free. In: Nierstrasz O., Whittle J., Harel D., Reggio G. (eds) Model Driven Engineering Languages and Systems. MODELS 2006. Lecture Notes in Computer Science, vol. 4199. Springer, Berlin, Heidelberg, pp. 155–169.
[31] Gray, J., Bapty, T., Neema, S. and Tuck, J. (2001). Handling crosscutting constraints in domain-specific modeling. In: Communications of the ACM, vol. 44, no. 10, pp. 87–93.
[32] Niu, N., Easterbrook, S. and Yu, Y. (2007). A Taxonomy of Asymmetric Requirements Aspects. In: Proceedings of the 10th International Conference on Early Aspects: Current Challenges and Future Directions, pp. 1–18.
[33] Rashid, A., Cottenier, T., Greenwood, P., Chitchyan, R., Meunier, R., Coelho, R., Südholt, M. and Joosen, W. (2010). "Aspect-oriented software development in practice: Tales from AOSD-Europe". In: Computer, California, vol. 43, no. 2, pp. 19-26.
[34] Munoz, F., Baudry, B., Delamare, R. and Le Traon, Y. (2013). "Usage and testability of AOP: An empirical study of AspectJ". In: Information and Software Technology, vol. 55, no. 2, pp. 252–266.
[35] Munoz, F., Baudry, B., Delamare, R. and Le Traon, Y. (2009). "Inquiring the usage of aspect-oriented programming: An empirical study". In: 2009 IEEE International Conference on Software Maintenance, Edmonton, pp. 137-146.
[36] Magableh, A., Shukur, Z. and Ali, N.M. (2013). Systematic review on aspect-oriented UML modeling: A complete aspectual UML modeling framework. In: Journal of Applied Sciences, vol. 13, no. 1, pp. 1–13.
[37] Mohite, S., Phalnikar, R., Joshi, M., Joshi, S.D. and Jadhav, S. (2014). Requirement and interaction analysis using aspect-oriented modeling. In: IEEE International Advance Computing Conference, IACC 2014, pp. 1448–1453.

[38] Gerami, M. and Ramsin, R. (2011). A framework for extending agile methodologies with aspect-oriented features. In: Proceedings of the Fifth International Conference on Research Challenges in Information Science, Gosier, pp. 1-6.
[39] Elrad, T., Aldawud, O. and Bader, A. (2005). Expressing Aspects using UML Behavioral and Structural Diagrams. In: Filman, R., Elrad, T., Clarke, S. and Akşit, M. (eds) Aspect-Oriented Software Develpoment, Addison Wesley, 2005, pp. 459-478.
[40] McConnell, S (2006). Software Estimation: Demystifying the Black Art. Microsoft Press.
[41] Leavitt, N. (2000). Whatever Happened to Object-Oriented Databases? In: Computer, vol. 33, no. 8, Long Beach, pp. 16–19.
[42] Jalali, A. (2014). Assessing Aspect Oriented Approaches in Business Process Management. In: Johansson B., Andersson B., Holmberg N. (eds) Perspectives in Business Informatics Research, BIR 2014, Lecture Notes in Business Information Processing, vol 194, Springer, Cham, pp. 231–245.
[43] Sjoberg, D.I.K., Dyba, T. and Jorgensen, M. (2007). The Future of Empirical Methods in Software Engineering Research. In: Future of Software Engineering, FOSE '07, Minneapolis, pp. 358-378.
[44] Wohlin, C., Höst, M., Henningsson, K. (2006). Empirical Research Methods in Web and Software Engineering. In: Mendes, E., Mosley, N. (eds), Web Engineering, Springer, Berlin.
[45] Pinciroli, F., Barros Justo, J.L., Zeligueta, L. and Palma, M. (2018) Systematic Mapping Protocol. Coverage of Aspect-Oriented Methodologies for the Early Phases of the Software Development Life Cycle. In: Proceedings of the II International Congress on Computer Sciences and Information Systems, CICCSI 2018, Mendoza.
[46] Pinciroli, F. (2011). Consideraciones para un proceso de desarrollo de software de calidad orientado a aspectos. In: Proceedings of the Sexto Encuentro de Investigadores y Docentes de Ingeniería, EnIDI 2011, San Rafael.
[47] Pinciroli, F., Barros Justo, J.L. and Forradellas, R. (2017). Aspect-Oriented Business Process Modeling Approaches: An assessment of AOP4ST. In: Proceedings of the 46 Jornadas Argentinas de Informática, JAIIO 2017, pp. 40-47.
[48] Pinciroli, F. and Zeligueta, L. (2016). El modelo de negocio en AOP4ST. In: Proceedings of the XVIII Workshop de Investigadores en Ciencias de la Computación, WICC 2016, Entre Ríos, pp. 423-426.
[49] Pinciroli, F. (2016). Aspect-oriented business process composition rules in AOP4ST. In: Proceedings of the 35th International Conference of the Chilean Computer Science Society, Valparaíso, pp. 1–6.
[50] Pinciroli, F. and Albino, G. (2020). Reglas de composición para modelos de procesos orientados a aspectos con BPMN 2.0. In: Proceedings of the 8° Congreso Nacional de Ingeniería en Informática e Ingeniería en Sistemas de Información, CONAIISI 2020, San Francisco.
[51] Pinciroli, F. and Barros Justo, J.L. (2017). Early aspects in 'Aspect-Oriented Process for a Smooth Transition'. In: Proceedings of the XXIII Argentine Congress of Computer Science, La Plata, pp. 692-701.
[52] Pinciroli, F. (2017). Concern detection along the requirement development. In: Entre Ciencia e Ingeniería, vol. 12, no. 23, Pereira, pp. 117-122.
[53] Pinciroli, F. (2018). Requisitos de usuario y gestión de la demanda en AOP4ST. In: Proceedings of the XX Workshop de Investigadores en Ciencias de la Computación, WICC 2018, Buenos Aires, pp. 511-515.
[54] Pinciroli, F. and Palma M. (2020). Desarrollo de requisitos en "Aspect-Oriented Process for a Smooth Transition". In: Proceedings of the XXII Workshop de Investigadores en Ciencias de la Computación, WICC 2020, El Calafate.
[55] Pinciroli, F. (2020). Explicit and implicit join point designation in aspect-oriented business modeling. In: Proceedings of the XLVI Conferencia Latinoamericana de Informática – CLEI 2020, Loja.
[56] Pinciroli, F. (2016). Improving software applications quality by considering the contribution relationship among quality attributes. In: Proceedings of the 3rd International Workshop of Computer Antifragility and Antifragile Engineering, ANTIFRAGILE 2016, Procedia Computer Science, vol. 83, pp. 970–975.
[57] Pinciroli, F. (2016). An HCI quality attributes taxonomy for an impact analysis to interactive systems design and improvement. In: Pesado, P.M. et al. (eds) Computer Science and Technology Series, Proceedings of the XXII Argentine Congress of Computer Science, Selected Papers, La Plata, pp. 133-143.
[58] Pinciroli, F. (2019). Modeling the Static View in Aspect-Oriented Software Development. In: Proceedins of the III International Congress on Computer Sciences and Information Systems, CICCSI 2019, Mendoza.
[59] Pinciroli, F. (2016). Aspect-Oriented Process for a Smooth Transition. In: Proceedings of the Ph.D. Symposium, IEEE 11 Congreso Colombiano de Computacion, Popayán.
[60] Pinciroli, F. (2020). A maturity model for the Integrated Agile Transformation Model™. In: Proceedins of the IV International Congress on Computer Sciences and Information Systems, CICCSI 2020, Mendoza.