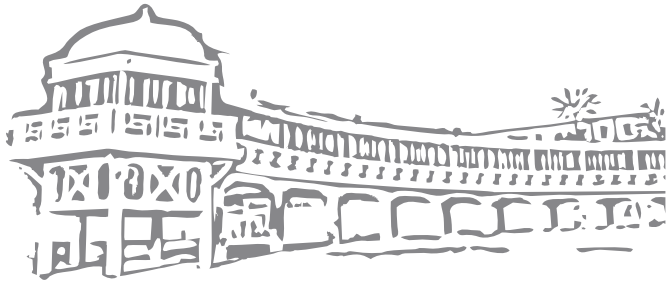


Jornadas Sistedes 2014

Cádiz, del 16 al 19 de septiembre

JISBD PROLE JCIS DC

Doctoral Consortium

ACTAS

Doctoral Consortium

Jornadas Sistedes 2014

Editores:

José Riquelme
Mercedes Ruiz
Mª Teresa García

Actas del I Doctoral Consortium de la Sociedad de Ingeniería de Software y Tecnologías de Desarrollo de Software (Sistedes)

Cádiz, 19 de septiembre 2014

Editores: José C. Riquelme, Mercedes Ruiz y Mª Teresa García

ISBN – 10: 84-697-1154-7

ISBN – 13: 978-84-697-1154-5

Prólogo

Numerosas reuniones científicas tienen entre sus actividades la presentación de los trabajos preparatorios de futuras tesis doctorales. Si bien tradicionalmente las Jornadas SISTEDES han sido un marco habitual para la presentación de trabajos predoctorales, nunca se había celebrado una sesión específica en la que se premiara un proyecto de tesis. Así pues, en esta publicación se recogen los siete trabajos presentados a la primera edición de un Doctoral Consortium dentro de las Jornadas SISTEDES, con un doble objetivo:

Por un lado, potenciar y motivar la presencia de nuestros futuros doctores en las Jornadas SISTEDES, justamente ahora que el número de doctorandos y tesis leídas en nuestros departamentos está acusando un importante descenso. La "utilidad" de una tesis doctoral es puesta en duda por la actual crisis que asfixia la investigación. Con esta actividad SISTEDES debe poner de relieve y significar el trabajo que representa una tesis doctoral.

Por otro lado, esperamos que el debate y las aportaciones del comité evaluador sirvan de ayuda y acicate para los doctorandos y sus directores. La presentación en SISTEDES proporciona la mejor audiencia posible para contrastar los objetivos que se quieren cubrir con la tesis y verificar posibles vías de mejora.

Para terminar agradecer a Javier Tuya su confianza para poder organizar esta sesión. A Mercedes Ruiz y su equipo por la inestimable ayuda prestada en la organización. A los miembros del comité evaluador por su diligencia en asumir la tarea y su dedicación. Y sobre todo, a los directores y doctorandos que han querido participar en esta primera experiencia. Esperamos que la colaboración de todos sirva para que este Doctoral Consortium sea el primero de una futura y larga tradición.

Cádiz, septiembre de 2014

José C. Riquelme

Presidente del Comité de evaluación

Prólogo de la organización

La primera edición del Doctoral Consortium (DC) de la Sociedad de Ingeniería de Software y Tecnologías de Desarrollo de Software (Sistedes) se celebra en 2014 en la ciudad de Cádiz, en el ámbito de las Jornadas Sistedes 2014. Como organizadores, agradecemos al Comité Permanente de dichas Jornadas la confianza depositada en nosotros y confiamos en que nuestro trabajo les permita a todos disfrutar de unas Jornadas enriquecedoras tanto desde el punto científico como personal.

Queremos agradecer a los miembros del comité de evaluación del DC su disponibilidad y trabajo efectuado durante las revisiones y la confección del programa científico del evento. Es imprescindible manifestar la excelente labor del presidente de este comité, José C. Riquelme, que, con su implicación en todos los detalles relacionados con el DC, nos ha facilitado en gran medida nuestra tarea.

También es necesario agradecer a las entidades colaboradoras su contribución en la celebración del evento. En tiempos como los actuales resulta, si cabe, aún más importante este agradecimiento. Agradecemos a la Universidad de Cádiz, a la Escuela Superior de Ingeniería y al Departamento de Ingeniería Informática las facilidades que nos han dado para el uso de las instalaciones y recursos. Queremos reconocer la importante contribución de los grupos de investigación de Mejora del Proceso Software y Métodos Formales y UCASE de Ingeniería del Software, porque han contribuido con lo más preciado que tienen que son sus miembros. De igual manera, queremos agradecer al Excmo. Ayuntamiento de Cádiz y a su Delegación de Turismo su atenta colaboración y atención en la organización de algunas de las actividades sociales del programa de las Jornadas. De igual manera, agradecemos a las empresas Renfe e Iberia su colaboración a la hora de poner a disposición de los asistentes los descuentos para el desplazamiento a Cádiz.

Finalmente, nuestro agradecimiento más particular y especial debe ir dedicado a todos aquellos que han colaborado en los trabajos de organización de las Jornadas. Agradecemos profundamente el tiempo, las energías, la implicación, el compromiso y el trabajo de todos los que han participado en la preparación y celebración de las Jornadas.

Os deseamos a todos una feliz estancia en Cádiz y confiamos en que las Jornadas constituyan un excelente foro de intercambio de conocimientos, experiencias y reflexión.

Cádiz, septiembre de 2014

Mercedes Ruiz

Presidenta del Comité Organizador
de las Jornadas Sistedes 2014

Comité de evaluación

PRESIDENTE DEL COMITÉ DE EVALUACIÓN

José C. Riquelme Santos (Universidad de Sevilla)

MIEMBROS DEL COMITÉ DE EVALUACIÓN

Nieves R. Brisaboa (U. Coruña)
Coral Calero (U. Castilla-La Mancha)
Oscar Diaz (U. País Vasco)
Santiago Escobar (U.P. Valencia)
Xavier Franch (U.P. Catalunya)
Juan Hernández (U. Extremadura)
Narciso Martí (U. Complutense)
Ana Mª Moreno (U.P. de Madrid)
Óscar Pastor (U.P. Valencia)
Isidro Ramos (U.P. Valencia)
Antonio Ruiz (U. Sevilla)
Goiuria Sagardui (U. Mondragón)
Miguel Toro (U. Sevilla)
Ambrosio Toval (U. Murcia)

Comité de organización

PRESIDENTA DEL COMITÉ ORGANIZADOR

Mercedes Ruiz Carreira

MIEMBROS DEL COMITÉ ORGANIZADOR

Juan Boubeta Puig
Mª del Carmen de Castro Cabrera
Pedro Delgado Pérez
Juan Manuel Dodero Beardo
Antonio García Domínguez
Mª Teresa García Horcajadas
Lorena Gutiérrez Madroñal
Nuria Hurtado Rodríguez
José Luis Isla Montes
Inmaculada Medina Bulo
José Miguel Mota Macías
Manuel Palomo Duarte
Elena Orta Cuevas
Guadalupe Ortíz Bellot
Carlos Rioja del Río
Iván Ruiz Rube
Alberto Gabriel Salguero Hidalgo

Entidades colaboradoras

Índice de contenidos

Modelos metaheurísticos para el soporte a la decisión en la construcción de software

Doctoranda: Aurora Ramírez
Directores: José Raúl Romero y Sebastián Ventura

Dpto. de Informática y Análisis Numérico
Universidad de Córdoba, Campus de Rabanales, 14071 Córdoba
{aramirez, jrromero, sventura}@uco.es

Resumen Durante el proceso de construcción del software, los ingenieros se enfrentan a tareas en las que deben evaluar y tomar decisiones sobre diferentes alternativas de diseño. Esto es especialmente relevante en la fase de análisis arquitectónico, pues aún no se dispone de toda la información sobre el sistema, si bien las elecciones realizadas afectarán al resto del proceso de diseño. Dar soporte al ingeniero en esta etapa del desarrollo mediante la obtención de modelos de decisión que tengan en consideración sus necesidades, los artefactos software que manejan y las medidas de calidad de su interés, constituye un reto interesante. La utilización de técnicas de optimización y búsqueda, especialmente las técnicas metaheurísticas bioinspiradas, para construir estos modelos de decisión constituye el principal objetivo de esta propuesta de Tesis Doctoral.

1. Introducción

El proceso de construcción de software consta de multitud de tareas en las cuales los ingenieros software deben evaluar diferentes alternativas y tomar decisiones que tendrán una gran repercusión en la calidad del software resultante. En este contexto, los sistemas de soporte a la decisión (DSS, *Decision Support Systems*) constituyen una herramienta interesante para complementar la labor del experto, ya que éste habitualmente se enfrenta a estas tareas únicamente guiado por sus habilidades y su experiencia [1].

La toma de decisiones tiene una especial relevancia en las primeras etapas del desarrollo del software, donde se concibe la estructura que mejor se adapta a los requisitos, tanto funcionales como no funcionales, impuestos por el cliente o por el propio sistema. En este punto, es importante que un experto analice rigurosamente las alternativas existentes, pues de ellas dependerá el resto del proceso de construcción del software así como el cumplimiento de los criterios de calidad establecidos.

A la hora de abordar la construcción de un DSS, la utilización de técnicas de Inteligencia Artificial (IA) puede facilitar la obtención de los modelos de decisión de manera automática. Las metaheurísticas [2], técnicas de IA centradas en la resolución de problemas de optimización y búsqueda, constituyen una alternativa interesante. En concreto, las metaheurísticas bioinspiradas se caracterizan

por simular conceptos de la naturaleza para definir una forma "inteligente" de exploración del espacio de búsqueda.

En los últimos años, la Ingeniería del Software ha encontrado en las metaheurísticas una potente herramienta para la optimización de sus tareas y procesos más complejos. La unión de ambas áreas, denominada *Search Based Software Engineering* (SBSE) [3] requiere, por tanto, la formulación de las tareas clásicas de la Ingeniería del Software como problemas susceptibles de ser resueltos mediante este tipo de técnicas.

2. Antecedentes

Desde sus inicios, el área de SBSE ha estado orientada a la resolución de problemas relacionados con la generación de casos de prueba o la planificación de recursos en proyectos software. Sin embargo, es menos frecuente encontrar propuestas destinadas a automatizar procesos propios del análisis y diseño del software. Puesto que se trata de fases fuertemente vinculadas al razonamiento humano, tanto la formulación como la resolución de los posibles problemas de optimización se convierten en tareas más complejas.

Actualmente, se están destinando importantes esfuerzos en tareas como el diseño orientado tanto a objetos como a servicios, o la optimización de arquitecturas software [4]. En este sentido, existen algoritmos bioinspirados capaces de ofrecer alternativas a los diseños originales propuestos por los ingenieros con el fin de mejorar su calidad, transformar modelos a distinto nivel de abstracción, refactorizar el código fuente, o encontrar la configuración óptima de los módulos que componen un sistema distribuido.

Si bien las propuestas anteriores han aportado métodos de apoyo a la labor de los ingenieros, la mayoría de ellas trabajan a niveles de abstracción bajos, esto es, manejan artefactos software muy específicos como el código fuente, o se basan en la existencia de modelos preexistentes. Esto implica que disponen de una gran cantidad de información sobre el software, que pueden recopilar y sobre la cual fundamentan el proceso de optimización. Sin embargo, las decisiones más relevantes para la calidad final del software son aquellas tomadas al comienzo de su construcción; por ejemplo, durante el modelado arquitectónico, cuando aún no existe un conocimiento profundo acerca de cómo será finalmente el sistema y, por tanto, es más difícil asistir al ingeniero. En este sentido, no existen propuestas que, analizando las necesidades del arquitecto en cuanto a requisitos no funcionales o estilos arquitectónicos específicos, obtengan modelos arquitectónicos bajo un enfoque metaheurístico.

3. Objetivos

Esta tesis se enmarca en el área de SBSE con el objetivo de proponer modelos de decisión basados en técnicas bioinspiradas que permitan la construcción de un DSS en el contexto del análisis y diseño del software desde sus fases más iniciales. Un aspecto importante a considerar es la obtención de modelos de decisión

comprensibles y útiles para el ingeniero software. En este sentido, los algoritmos propuestos deben ser capaces de manejar los modelos de análisis, adaptarse a las necesidades del experto, simular el proceso de búsqueda de alternativas que éste realiza habitualmente o evaluar dichas alternativas en función de medidas estándares de calidad del software. En concreto, los objetivos marcados al comienzo del desarrollo de la Tesis Doctoral son los siguientes:

1. Obtención de modelos arquitectónicos de forma semi-automática mediante algoritmos de Computación Evolutiva (EC). Para ello se formularán las distintas tareas englobadas en el análisis arquitectónico (especificación, recuperación o re-diseño de arquitecturas) como problemas de búsqueda. Esto, a su vez, permitirá identificar aspectos como la complejidad combinatoria del problema de optimización y la presencia de restricciones, que repercutirán en el diseño de los algoritmos evolutivos.
2. Análisis de las diferentes perspectivas que ofrecen las técnicas bioinspiradas con las cuales orientar el proceso de soporte a la decisión, como los enfoques multiobjetivo o los algoritmos interactivos. En concreto, se estudiará la optimización conjunta de diferentes criterios de calidad, los cuales deberán ser definidos a partir de un catálogo de medidas software estándares, o sea capaz de adaptarse dinámicamente a las preferencias del arquitecto.
3. Estudio de la aplicabilidad de la metodología propuesta en distintos dominios, como los sistemas distribuidos en la nube o las arquitecturas orientadas a servicios. De esta forma se analizará la configuración necesaria de los modelos de decisión en términos de los distintos requisitos no funcionales que pueden ser considerados en cada plataforma.
4. Abordar los desarrollos anteriores mediante un enfoque centrado en el experto, obteniendo modelos interpretables y eficientes que puedan ser integrados en herramientas de su ámbito (*MagicDraw*, *Eclipse Papyrus* o *jsUML2*), y funcionen bajo estándares de especificación de software como UML 2.

4. Metodología y plan de trabajo

Esta Tesis Doctoral ha sido iniciada en noviembre de 2013 dentro del programa de doctorado *Computación Avanzada, Energía y Plasmas* (RD99/2011) de la Universidad de Córdoba. A su vez, se enmarca dentro del programa de formación del profesorado universitario (FPU 2013) del Ministerio de Educación (inicio previsto en otoño de 2014). Con el fin de alcanzar los objetivos mencionados anteriormente, la metodología a seguir estará basada en el método científico:

- Observación: Estudio de las necesidades de soporte para la toma de decisiones en el ámbito del modelado arquitectónico.
- Inducción: Extracción de las características del escenario real y su posible resolución como problema de optimización y búsqueda.
- Hipótesis: Diseño de los modelos de decisión basados en metaheurísticas bioinspiradas para la resolución de los problemas planteados.

- Experimentación: Evaluación empírica de los algoritmos sobre sistemas software reales, bajo las medidas de rendimiento habituales en el ámbito de las metaheurísticas y considerando su utilidad para el arquitecto software.
- Antitesis: Estudio de los resultados obtenidos en relación a la hipótesis planteada y su reformulación si es necesario.

Por otro lado, el plan de trabajo previsto es el siguiente:

- Primer año: revisión bibliográfica completa de la investigación desarrollada, incluyendo técnicas metaheurísticas y SBSE. Catalogación de problemas relacionados con la construcción del software susceptibles de ser resueltos mediante técnicas metaheurísticas, centrándose en su formulación y en las medidas de calidad que pueden considerarse para la evaluación de los diseños software resultantes. Desarrollo de un modelo inicial en el contexto del descubrimiento de arquitecturas software mediante algoritmos evolutivos.
- Segundo y tercer año: desarrollo de modelos de decisión basados en diferentes tipos de algoritmos evolutivos (multiobjetivo, interactivos, etc.) para dar soporte a las tareas identificadas anteriormente dentro del análisis arquitectónico. Desarrollo de modelos metaheurísticos para la propuesta de alternativas de diseño en contextos como la transformación de modelos, la refactorización o el despliegue.
- Cuarto año: diseño e implementación de un sistema inteligente que integre los modelos anteriores con el proceso interactivo de decisión del experto. Evaluación del sistema en entornos reales e integración con herramientas del ámbito de la Ingeniería del Software. Redacción de la Tesis Doctoral.

5. Relevancia de la propuesta y primeras contribuciones

Con la realización de esta Tesis Doctoral se espera contribuir al campo científico de SBSE en el ámbito del descubrimiento y la optimización de arquitecturas software, donde las propuestas existentes se centran en aspectos de despliegue y configuración, o trabajan a bajo nivel de abstracción. A su vez, el enfoque planteado constituye también una novedad, pues se pretende hacer partícipe al ingeniero software en la construcción de los modelos de decisión.

En este sentido, es fundamental que la experiencia y razonamiento del arquitecto se reflejen en los modelos de decisión y que éstos le ofrezcan resultados comprensibles acerca del proceso de optimización, además de ciertas directrices de utilización, pues el arquitecto no está habituado a configurar y manejar algoritmos basados en metaheurísticas. De este modo, este proceso será transparente gracias a la creación de un DSS donde el arquitecto y el algoritmo de búsqueda interactuarán. A su vez, se realizarán aportaciones al campo de las metaheurísticas, pues será necesario diseñar algoritmos específicos para manejar problemas combinatorios complejos y del mundo real como los que plantea el área de la optimización arquitectónica.

Puesto que objetivo final de la Tesis es la obtención modelos metaheurísticos en el contexto del soporte a la decisión, sus posibilidades de aplicación a entornos

reales está muy presente. Por un lado, el arquitecto software encontrará útiles los modelos de decisión propuestos, ya que habrán sido creados según sus necesidades y le permitirán descubrir más alternativas arquitectónicas, y con un esfuerzo menor, que cualquier proceso manual de diseño o refactorización. Por otro lado, se ha contemplado la integración del DSS con herramientas y lenguajes de modelado, de forma que se ofrecerá al arquitecto una herramienta que realmente podrá ser utilizada durante el proceso de diseño del software. De esta forma, se espera que los resultados obtenidos sean transferibles a la comunidad de Ingeniería del Software para facilitar el desarrollo de su actividad diaria.

Nuestras primeras contribuciones realizadas en el área han estado orientadas al manejo de artefactos de análisis que permitan al ingeniero software abstraer la arquitectura subyacente a un sistema software. En concreto, se han publicado las siguientes contribuciones a congresos nacionales e internacionales:

- A. Ramírez, J.R. Romero, S. Ventura. «On the Performance of Multiple Objective Evolutionary Algorithms for Software Architecture Discovery». *16th Genetic and Evolutionary Computation Conference (GECCO)*, pp. 1287-1294, 2014. *Best paper del track de SBSE. Nominado a best paper de GECCO'14.* Conferencia *Core A*.
- A. Ramírez, J.R. Romero, S. Ventura. «Análisis de la aplicabilidad de medidas software para el diseño semi-automático de arquitecturas». *XIX Jornadas en Ingeniería del Software y Bases de Datos (JISBD)*. 2014. *Aceptado.*
- A. Ramírez, J.R. Romero, S. Ventura. «A Novel Component Identification Approach Using Evolutionary Programming». *15th Genetic and Evolutionary Computation Conference (GECCO)*, pp. 209-210. 2013. Conferencia *Core A*.
- A. Ramírez, J.R. Romero, S. Ventura. «Identificación de Componentes en Arquitecturas Software Mediante Programación Evolutiva». *XVIII Jornadas en Ingeniería del Software y Bases de Datos (JISBD)*, pp. 413-426. 2013.
- A. Ramírez, J.R. Romero, S. Ventura. «Algoritmo de programación evolutiva para identificación de arquitecturas software». *IX Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB)*, pp. 892-901. 2013. *Finalista al premio al mejor artículo de carácter metodológico.*

Referencias

1. D. Falessi, G. Cantone, R. Kazman, and P. Kruchten, "Decision-making Techniques for Software Architecture Design: A Comparative Survey," *ACM Comput. Surv.*, vol. 43, no. 4, pp. 33:1–28, 2011.
2. I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Information Sciences*, vol. 237, no. 0, pp. 82 – 117, 2013.
3. M. Harman, S. A. Mansouri, and Y. Zhang, "Search Based Software Engineering: Trends, Techniques and Applications," *ACM Comput. Surv.*, vol. 45, no. 1, pp. 11:1–61, 2012.
4. A. Aleti, B. Buhnova, L. Grunske, A. Koziolek, and I. Meedeniya, "Software Architecture Optimization Methods: A Systematic Literature Review," *IEEE Transactions on Software Engineering*, vol. 39, no. 5, pp. 658–683, 2013.

Analyzing the Correctness of Smartphones Applications using Formal Methods
(Doctoral Consortium)

Author: Ana-Rosario Espada-Sandi and Director: María-del-Mar Gallardo

Dpto. de Lenguajes y Ciencias de la Computación
University of Málaga, Spain
anarosario@lcc.uma.es gallardo@lcc.uma.es

Abstract. In this document, we present an abstract of the PhD thesis research which is being undertaken by Ana Rosario Espada Sandi. The central aim of the thesis is to provide a framework for verifying the correctness of mobile applications by combining different formal methods such as model checking, testing, runtime verification and other techniques specifically developed for this frame. The main operative system analyzed in this work is ANDROID.

1 Motivation and Relevance

Presently smartphone technology is ubiquitous and changes constantly. Users use their mobiles not only as phones, but as compact computers able to concurrently provide services which are rapidly created, updated, renewed and distributed. In this scenario of continuous evolution, different operative systems have been developed such as SYMBIAM, IOS, WINDOWS PHONE, ANDROID, which allow phones to support more and more complex applications.

These platforms define new models of execution, quite different from those used by non-mobile devices. For instance, one of the most defining characteristics of these systems is their open and event-driven nature. Mobile devices execute a continuous cycle that consists of first waiting for the user input and secondly producing a response according to that input. In addition, the internal structure of mobile systems is constructed from a *complex combination* of applications, which enable users to easily navigate through them. Thus, although, at a lower level, the execution of applications on a mobile involves the concurrent execution of several processes (for instance, in ANDROID, applications are JAVA processes executing on the underlying LINUX operative system), the way these applications interact with each other and to the environment does not correspond with the standard interleaving based concurrency model.

It is clear that the execution of applications on these new operative systems, such as ANDROID [1], may involve the appearance of undesirable bugs which may cause the the phone to malfunction. For example, mobile devices may display typical errors of concurrent systems such as violations of safety and liveness properties which may be analyzed by classic analysis techniques such as

model checking [15, 13]. However, there are other bugs inherent to the particular concurrency model supported by the new platforms which are not directly analyzable using current verification technologies. For example, applications could incorrectly implement the life cycles of their activities or services (in the case of ANDROID), or may misbehave upon the arrival of unexpected external events. In addition, there may also appear conversion errors, unhandled exceptions, errors of incompatibility API and I/O interaction errors as described in [10]. Moreover, there are other common properties that all applications should satisfy: applications should not drain the battery; applications should be reliable, this means that, for example, they should not access prohibited memory zones.

In this context, the goal of this thesis is, on the one hand, to develop of a formal model able to capture the specific characteristics of the execution of applications on mobiles and oriented to the construction of analysis tools. In addition, we are also interested in the construction of specific tools, according to this model, which may be used by developers to analyze the correctness of their applications and as a quality certification tools. We think that both the problem and the solution proposed in the present thesis project are especially relevant because current trend of the technology is towards to the massive use of mobile devices for which no well-established verification techniques exist.

The proposals is original with respect to the current approaches which deal with correctness in the context by reusing classic concurrency models of systems running on non-mobile devices. However, we think that it by is necessary to provide new interaction models adapted to mobile environments which make it possible to build specific verification tools.

Although, in a first approach, we have concentrated on ANDROID systems, the techniques developed are expected to be generally applicable to other mobile operative systems.

2 Goals of the thesis

In more detail, the goals of the thesis may be summarized as follows.

1. The development of specific *formal models* which allow us to precisely describe the execution of applications on mobile operative systems. In particular, we are working on the definition of a state-machine based language able to model the behavior of applications with different abstraction degrees.
2. The construction of specific logics to instrument and filter the events and data output to be analyzed, using patterns, temporal logic (LTL), and aspect oriented techniques.
3. The combination of existing verification techniques such as *Runtime Verification*, *Model-Based Testing* and *Model Checking* which will be adapted to the specific environment determined by the standard navigation of the mobile operative close gap.
4. The specification of the desirable/undesirable properties of mobile applications to be checked with automata-based property languages combined with

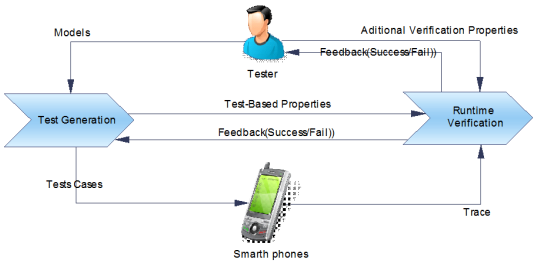


Fig. 1. Proposal of Design

domain specific languages. In addition to the typical safety and liveness properties, we are interested in the description of performance properties such as drying battery bugs, reliability problems, declining performance, communication problems.

5. The development of a verification framework of mobile operative systems based on the aforementioned formal models, logics and techniques to analyze the correctness of isolated applications running on a single device, the correctness of the interaction between multiple applications running in an single device, and the correctness of the interaction of applications running on multiple devices. The development of tools that implement the theoretical proposals, and which make it possible to compare them with other approaches.

3 Methodology

Although our proposal attempts to be generic for mobile environments, we have started by studying the case of ANDROID, since along with IOS, ANDROID is ahead of the market and due to the open distribution of applications, a number of known errors have been identified.

We are following the standard research methodology which may be summarized by the following list of tasks:

1. Studying the state of the art in verification technologies for concurrent systems, and in the concurrency model displayed by the existing mobile operative systems. This task has been completed. Section 4 presents some of the related works found in the course of studying the state of art.
2. Designing a generic concurrency model that describes the behavior of mobile applications and oriented to the construction of verification tools.
   The current state of this task can be summarized as follows. We propose to model the mobile behavior by a *complex composition* of *state machines*, each one representing the way users deal with a specific view/activities of the phone. The underlying idea is that the user interaction is dependent on the application being used and it must be specifically modeled before

the application is analyzed, *but* the complex composition of state machines is fixed (it follows the standard mobile navigation scheme), and therefore it may only be modeled once. We are also using model-based testing techniques to allow the *exhaustive generation* of possible scenarios/test cases produced by the users.

3. Developing of verification tools for ANDROID systems according to the proposed design.
   Figure 1 shows a high-level description of the current proposal of implementation. The verification tasks are divided into two main modules. On the one hand, the *test generation* module, which, by using a combination of model-based testing and model checking techniques generates scenarios/test cases that constitute the environment of the system, i.e., the user behavior. On the other hand, the *runtime verification* module that carries out the verification tasks on the ANDROID events and data, produced by the generation module, by means of runtime verification. We presently have two implementations of these modules in the case of ANDROID: the tool DRAGONFLY [7, 6] in which the user behavior is randomly generated, and an improved tool, DRACO.
4. Proving the tools with non-trivial case studies that include the analysis of complex properties (mainly performance properties) on several applications executing on several mobiles.
5. Comparing the results obtained with those provided by other approaches.
6. Extending the tools to other operative mobile systems.

4 Related Work

Several approaches for analyzing the correctness of ANDROID applications have been proposed in the literature. For instance, the JPF team has defined models of fictitious ANDROID environment which is compiled with the application, and later, analyzed using *model checking* techniques [15, 13].

Other approaches have analyzed ANDROID applications with *testing techniques* [10, 3, 12]. In this case, the environment may be simulated by a model of events, or by a sequence of random events which are fired (1) to generate an execution trace, saved in a *log* file which is later analyzed, (2) to generate test cases which are later executed.

Runtime verification (RV) [9, 11] is an other alternative method for verifying ANDROID applications. Monitors can work by analyzing a *log file* after the execution. In this case, they may follow a corrective approach such as [8]. In addition, monitors can also work at execution time. In the latter case, monitors could be embedded internally inside the application or they could be external, running in parallel with the application, either asynchronously or synchronously. For instance, a monitor could be executed synchronously with the ANDROID system, observing the evolution of the system, and firing an alert signal when a security property is violated [14, 5].

Finally, the proposal described in [9, 4] is based on the instrumentation of data provided by the traces of a JAVA application using ASPECTJ [2]. In addition,

they use several monitors embedded inside the application to verify each purpose of the property described in the SCALA language.

References

1. Android developer. http://developer.android.com/.
2. Aspectj. http://www.eclipse.org/aspectj/.
3. Domenico Amalfitano, Anna Rita Fasolino, and Porfirio Tramontana. A gui crawling-based technique for android mobile application testing. In *ICST Workshops*, pages 252–261. IEEE Computer Society, 2011.
4. H. Barringer and K. Havelund. Checking flight rules with TraceContract: Application of a Scala DSL for trace analysis. In *Proceedings of Scala Days 2011*, 2011.
5. Andreas Bauer, Jan-Christoph Kster, and Gil Vegliach. Runtime verification meets android security. In AlwynE. Goodloe and Suzette Person, editors, *NASA Formal Methods*, volume 7226 of *Lecture Notes in Computer Science*, pages 174–180. Springer Berlin Heidelberg, 2012.
6. Ana-Rosario Espada, María-del Mar Gallardo, and Damián Adalid. Dragonfly : Encapsulating android for instrumentation. In *Proceedings of the XIII Jornadas sobre Programacin y Lenguajes (PROLE13)*, 2013.
7. Ana-Rosario Espada, María-del Mar Gallardo, and Damián Adalid. A runtime verification framework for android applications. In *Proceedings of XXI Jornadas de Concurrencia y Sistemas Distribuidos*, 2013.
8. Yliès Falcone and Sebastian Currea. Weave droid: Aspect-oriented programming on android devices: Fully embedded or in the cloud. In *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, ASE 2012, pages 350–353, New York, NY, USA, 2012. ACM.
9. Klaus Havelund. Implementing runtime monitors. extended abstract., 2012. Invited talk, 2nd TORRENTS Workshop (Time ORiented Reliable Embedded NeTworked Systems.
10. Cuixiong Hu and Iulian Neamtiu. Automating gui testing for android applications. In *Proceedings of the 6th International Workshop on Automation of Software Test*, AST '11, pages 77–83, New York, NY, USA, 2011. ACM.
11. Martin Leucker. Teaching runtime verification. In *Proceedings of the Second International Conference on Runtime Verification*, RV'11, pages 34–48, Berlin, Heidelberg, 2012. Springer-Verlag.
12. R. Mahmood, N. Esfahani, T. Kacem, N. Mirzaei, S. Malek, and A. Stavrou. A whitebox approach for automated security testing of android applications on the cloud. In *Proc. of 7th International Workshop on Automation of Software Test (AST*, pages 22–28, 2012.
13. Peter Mehlitz, Oksana Tkachuk, and Mateusz Ujma. Jpf-awt: Model checking gui applications. *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*, 0:584–587, 2011.
14. Machigar Ongtang, Stephen McLaughlin, William Enck, and Patrick McDaniel. Semantically rich application-centric security in android. In *Proceedings of the 2009 Annual Computer Security Applications Conference*, ACSAC '09, pages 340–349, Washington, DC, USA, 2009. IEEE Computer Society.
15. Heila van der Merwe, Brink van der Merwe, and Willem Visser. Verifying android applications using java pathfinder. *SIGSOFT Softw. Eng. Notes*, 37(6):1–5, November 2012.

NDT-SOA, extensión de la metodología NDT para el desarrollo de Aplicaciones Web en organizaciones públicas basadas en SOA

J. Sedeño[1], M.J. Escalona[2] y M. Mejías[2]

[1]Doctorando. Grupo IWT2. ETSII, Universidad de Sevilla, Sevilla, España
jorge.sedeno@iwt2.org
[2]Directores. Lenguajes y Sistemas, ETSII, Universidad de Sevilla, Sevilla, España.
{mjescalona,risoto}@us.es

Abstract. Debido a que las administraciones públicas han de prestar a la ciudadanía los servicios a través de procesos telemáticos, la forma natural de prestarlos se realizará a través de aplicaciones Web. Cuando estas organizaciones tienen implantado un gobierno de los servicios basado en Arquitecturas Orientadas a Servicios (SOA) con cierto grado de madurez, es necesario que el paradigma de desarrollo software que utilicen incorpore metodologías que puedan trabajar con esos servicios en las fases tempranas del desarrollo, como son la fases de requisitos y análisis. En esta integración entre el gobierno de esos servicios y el desarrollo de los mismos, el paradigma de la Ingeniería Web guiada por modelo, reflejado en la propuesta NDT (Navigational Development Techniques), será el más adecuado ya que creará una integración natural y automatizada en la utilización, en las fases tempranas del desarrollo, de esos Servicios Web.

Keywords: MDWE, SOA, Governance, NDT, Public Administration, Services.

1 Introducción

En las organizaciones públicas que han de prestar a la ciudadanía los servicios públicos a través de procesos telemáticos [1,2] y con capacidad de operar bajo paradigma SOA, una de las piezas fundamentales dentro del Modelo Objetivo SOA [5] es la referente a la parte metodológica, especialmente en lo referido a la Metodología de Desarrollo del Modelo SOA, que ha de consistir en una metodología que integre la capacidad de desarrollo de Sistemas de Información Web en las que intervienen servicios, con la inclusión de las políticas de Gobierno SOA, en las etapas tempranas de dicho desarrollo.

Dentro de las soluciones metodológicas del paradigma de Ingeniería Web guiada por modelos (MDWE) [3], que se han demostrado eficaces para el desarrollo Web, hemos escogido la metodología NDT (Navigational Development Techniques) [4] con el objeto de extender sus actividades para incorporar, en las fases de toma de

requisitos y análisis, las actividades necesarias para el desarrollo de los servicios. La elección de esta metodología se ha realizo por dos razones fundamentales:

- Esta metodología ya se ha utilizado con éxito en varias administraciones públicas [8, 9, 10].
- Está encuadrada dentro de la línea de investigación del Grupo IWT2 dentro del marco de MDWE y cuenta una suite de herramientas implementadas que soportan la NDT y que han sido utilizadas en varios proyectos [8, 10].

1.1 Investigaciones previas

El objeto de la presente sección es hacer un repaso de aquellos paradigmas que se han utilizado y que representan la base para lo expuesto en las siguientes secciones:

La implantación del paradigma SOA es un proceso complejo, que implica aspectos técnicos, organizacionales y de negocio. Se requiere conjugar el conocimiento teórico de las arquitecturas orientas a servicios con un profundo conocimiento de los procesos de negocio a de la propia organización.

Como se ha presentado en el trabajo Fin de Master [5] y en otras publicaciones en congresos [6,7], en la Figura 1 se observa el Modelo Objetivo SOA que dotará a una Administración Pública de la capacidad de comenzar a operar bajo el paradigma SOA, transformando su propia naturaleza. Este Meta Modelo debe ser entendido tanto como una parte del marco propuesto como del resultado de aplicar la metodología. Uno de los objetivos de la presente tesis será mostrar como la metodología del ciclo de vida de desarrollo se ha de integrar con los Procesos de Gobierno SOA de la organización.

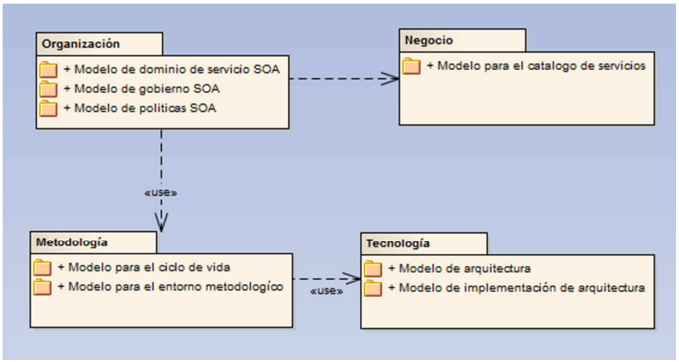


Fig. 1. Modelo de Componentes SOA

El nuevo paradigma de la Ingeniería Web guiada por modelos (MDWE, Model-Driven Web Engineering) se ha demostrado eficaz a la hora de proporcionar soluciones adecuadas en entornos de investigación de desarrollo Web dentro de las organizaciones. Estas nuevas técnicas para el desarrollo de sistemas Web introducidos por la Ingeniería Web ofrecen soluciones muy interesantes de alta calidad y reducido coste. NDT [4], es una propuesta, dentro de este paradigma, para definir y analizar

sistemas Web y la toma de requisitos. La versión práctica de este enfoque está orientada a ofrecer un entorno metodológico adecuado para el desarrollo web.

1.2 Objetivos

El objetivo principal de la tesis es la extensión de la metodología NDT para integrar dentro del ciclo de vida del desarrollo que propone, una serie de actividades que permitan integrar el desarrollo Web con las políticas de gobierno relacionadas con el ciclo de vida de los servicios SOA (Figura 2). Para ello se proponen las siguientes metas para extender la metodología NDT para entornos fuertemente adaptados a SOA:

- Identificar las actividades referentes a las fases de requisitos, de análisis y diseño.
- La definición del artefacto Servicio como núcleo de las actividades.
- La adaptación de las trasformaciones para poder operar con el artefacto Servicio.
- Diseño de un repositorio de servicios en tiempo de toma de requisitos.

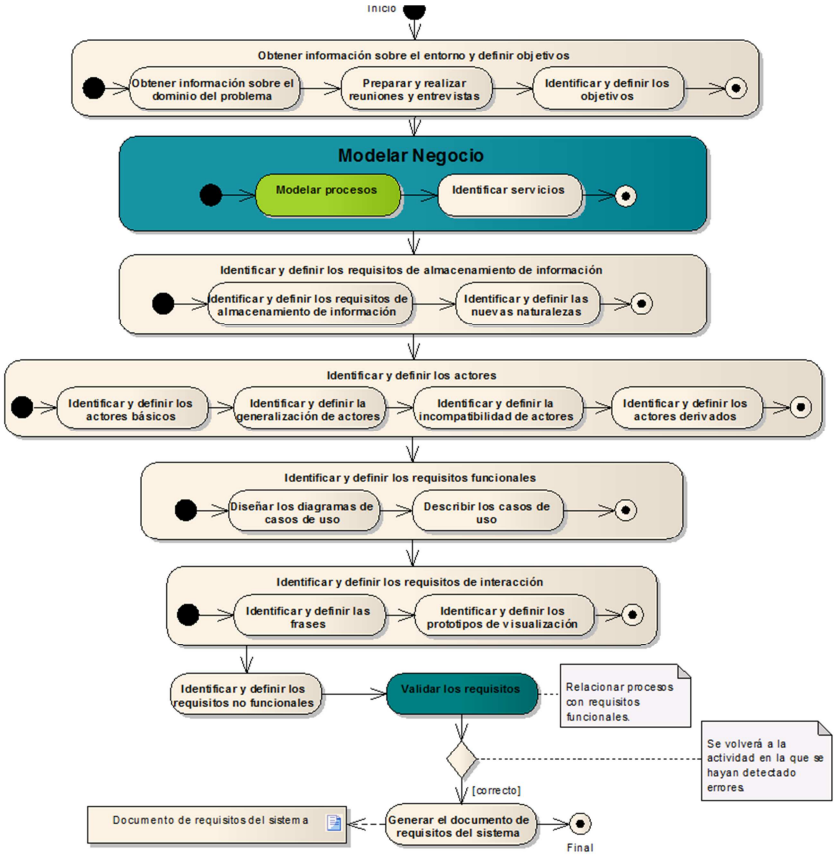


Fig. 2. Diagrama de Actividades en UML de la Fase de Requisitos

Se presentará el conjunto de actividades necesarias para que ciclo de vida de NDT pueda integrar el desarrollo bajo paradigma SOA. Para ello se deberán añadir una serie de actividades al proceso de desarrollo de NDT relacionadas con el ciclo de vida de los Servicios. De esta manera, en el proceso de toma de requisitos se podrán, no solo reutilizar los servicios existentes en la organización, sino educir nuevos servicios que podrán ser utilizados a su vez en fututos desarrollos. Como consecuencia directa, será necesario que en la fase de requisitos, por ejemplo, se pueda contener acciones (Figura 3) que permitan modelar procesos y relacionar procesos y servicios.

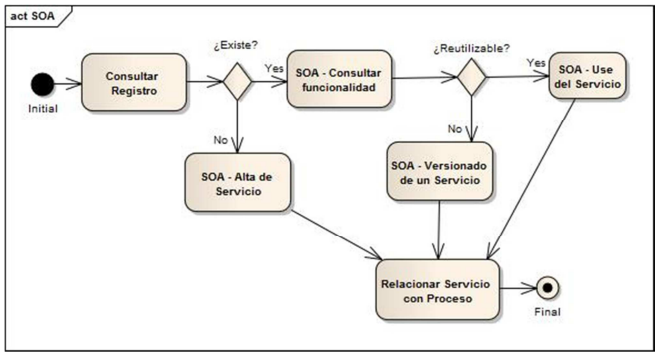


Fig. 3. Diagrama UML de Actividades para la Identificación de Servicios

Para ello se podrá consultar el repositorio de Servicios, que proporcionará información de todos los servicios existentes. En función de la existencia o no del servicio, se dispararán los Procedimientos de Gobierno SOA correspondientes.

1.3 Metodología y Plan de Trabajo.

De cara a la realización de la presente tesis doctoral se deberá seguir una metodología secuencial que abarque los siguientes hitos:

- Recopilación de la información sobre el estado de la integración entre las implantaciones de arquitecturas SOA en organizaciones públicas y el paradigma de desarrollo de aplicaciones Web basado en Ingeniería Web guiada por modelos.
- Identificación, definición y desarrollo de los puntos de integración entre la metodología de desarrollo NDT y las políticas de Gobierno SOA.
- Extensión de la metodología NDT con las nuevas actividades, adaptando las transformaciones que propone para la introducción de servicios en las etapas tempranas del desarrollo. Presentación de la extensión, denominada NDT-SOA.
- Implantación de esta metodología en una Administración Pública para refinar el modelo y comparar resultados con la actual metodología utilizada.

2 Relevancia

Este trabajo de propiciará un marco para la integración entre el ciclo de vida del desarrollo de sistemas de información Web con las políticas del ciclo de vida de los Servicios SOA en organizaciones fuertemente basadas en SOA que prestan servicios a través de medios telemáticos. Esto podría constituir un impulso importante a este campo de investigación, debido a que este tipo de organizaciones realizan no sólo el desarrollo de las aplicaciones sino el desarrollo de los servicios, en el presente de trabajo de investigación se propondrán estructuras jerárquicas para el almacenamiento de servicios en repositorios, no ya en tiempo de ejecución si no en tiempo de definición de requisitos y análisis de los mismos. Por otra parte las organizaciones que puedan adoptar este Modelo Objetivo SOA podrán realizar de forma automática la transformación de los requisitos, mediante los mecanismos de transformación que provee la metodología y la suite NDT, hacia el análisis, el diseño detallado, el modelo de navegación y el código fuente integrando la conexión con los servicios existentes en la organización, propiciando una reducción importante de los costes.

3 Referencias

1. "Ley 11/2007 de Acceso Electrónico de los Ciudadanos a los Servicios Públicos". 2007. http://www.boe.es/boe/dias/2007/06/23/pdfs/A27150-27166.pdf.
2. Plan de Acción Europeo sobre Administración Electrónica 2011-201. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2010:0743:FIN:ES:PDF.
3. OMG. Unified Modeling Language: Superstructure, version 2.0. Specification, OMG, 2005. http://www.omg.org/spec/UML/2.0/
4. M.J. Escalona, G. Aragón. NDT. A Model-Driven approach for Web requirements. IEEE Transaction on Software Engineering. Vol. 34. Nº3. pp. 370-390. 2008.
5. Sedeño J.; "Implantación de una arquitectura SOA en Administraciones Públicas", Máster de Gestión de las Tecnologías de la Información y las Comunicaciones 2011-2012.Trabajo Fin de Máster. Universidad de Sevilla. Julio 2012.
6. Sedeño J.; Torrecilla-Salinas C.J.; Escalona M.J.; Mejías M. "An approach to transform Public Administration into SOA-based organizations". 10th International Conference on Web Information Systems and Technologies, pp. 135-142 (Barcelona, Esp., 2-4 Abr 2014).
7. Sedeño J.; Torrecilla-Salinas C.J.; Escalona M.J.; Mejías M. "Propuesta para la transformación de Administraciones Públicas en organizaciones basadas en SOA". XVIII Jornadas de Ingeniería del Software y Bases de Datos, pp. 105-110. (Madrid, Esp.). 2013
8. García-García J.A., Escalona M.J, Domínguez-Mayo F.J., Salido A.; "NDT-Suite: A methodological tool solution in the Model-Driven Engineering Paradigm". Journal of Software engineering and Aplications (JSEA). 2014.
9. Cutilla, C.R., García-García, J.A., Gutiérrez, J.J., Domínguez-Mayo, P., Escalona, M.J., Rodríguez, L.; Domínguez-Mayo, F.J. "Model-driven test engineering. A practical analysis in the AQUA-WS project". Proceedings of the 7th International Conference on Software Paradigm Trends, pp. 111-119 (Rome, Italy, 2012).
10. Gutiérrez J., Aragón G., Mejías M., Domínguez F., Cutilla C.R.; "Automatic Test Case Generation from Functional Requirements in NDT". Lecture Notes in Computer Science, pp. 176-185. (Berlin, Germany, 2012).

Ejecución de Operaciones de un Esquema Conceptual de forma Persistente y Consistente

Doctorando: Xavier Oriol. Director: Ernest Teniente *

Departamento de Servicios e Ingeniería de Sistemas
Universitat Politècnica de Catalunya – BarcelonaTech
{xoriol,teniente}@essi.upc.edu

Resumen El objetivo de la presente tesis es la ejecución de operaciones de un esquema conceptual de forma *persistente* y *consistente*. Es decir, los efectos de las operaciones son persistidos automáticamente en una base de datos a la vez que se garantiza que su ejecución no conlleva, en ningún caso, la violación de restricciones de integridad definidas en el esquema. De esta forma, se automatiza la construcción de las capas de dominio y persistencia de un Sistema de Información requiriendo únicamente la implementación de un interfaz gráfica para su uso.

1 Introducción

Un esquema conceptual es la definición de un Sistema de Información (SI) en términos de qué datos debe almacenar dicho SI y qué operaciones dispondrán sus usuarios para manipularlos. Siendo los esquemas conceptuales definidos con lenguajes formales tales como UML/OCL, se erige lícitamente la siguiente cuestión: ¿Puede un esquema conceptual ser ejecutado? Es decir, ¿es posible *compilar* un esquema conceptual para obtener código o, alternativamente, es posible *interpretar* un esquema conceptual como si fuera código?

Desde los años 60 que se pretende dar respuesta a semejante reto [1]. A tal fin, diferentes propuestas han sido realizadas con distinto matiz. Por ejemplo, la propuesta del OMG, el Model Driven Architecture (MDA), se basa en compilar un esquema conceptual a código a través de una serie de transformaciones automáticas [2]. Por otro lado, el Conceptual Schema-Centric Development [3] contempla la opción de interpretar, directamente, un esquema conceptual como si fuera código.

El objetivo de la presente tesis se enmarca en la segunda vertiente: la *interpretación* de esquemas conceptuales. Dicho de otra forma, nuestra propuesta pretende, dado un esquema conceptual, ejecutar directamente sus operaciones sin necesidad de compilarlo a un lenguaje máquina.

* Este trabajo ha sido parcialmente financiado por el Ministerio de Ciencia e Innovación bajo el proyecto TIN2011-24747 y la beca FI de la Secreteria d'Universitats i Recerca de la Generalitat de Catalunya.

2 Hipótesis de Partida y Objetivos

Nuestra propuesta parte de la idea de que un esquema conceptual definido en un lenguaje formal como UML/OCL es lo suficientemente preciso como para poder ser ejecutado.

Dicho supuesto está corroborado por la existencia de herramientas de animación de esquemas conceptuales como USE [4]. Dichas herramientas son aplicaciones que, dado un esquema conceptual, permiten ejecutar sus operaciones manteniendo el estado de los datos en memoria. Obsérvese que una herramienta de animación no cumple nuestro propósito puesto que los datos no son persistidos en ninguna Base de Datos (BD) y que, además, su eficiencia, en cuanto a la comprobación de restricciones de integridad se refiere, está en entredicho [5].

Sin embargo, comprobar la satisfacción de una restricción de integridad debería poder realizarse de forma eficiente. Primero, por la existencia de trabajos específicamente enfocados a la implementación eficiente de dichas comprobaciones [6]. Segundo, porque se puede reducir el problema de comprobar una restricción de integridad a una consulta SQL [7].

Además, con la existencia de técnicas de persistencia automática (e.g. Hibernate), no parece que se debieran de padecer problemas en persistir los datos.

Todo ello nos lleva a la definición de nuestro objetivo principal:

Objetivo 1 *Proporcionar un entorno en el que un usuario pueda cargar un esquema conceptual UML/OCL y ejecutar sus operaciones persistiendo los cambios en una BD garantizando que ninguna restricción sea violada.*

Este objetivo abstracto se desglosa en los siguientes subobjetivos concretos:

Subobjetivo 1 *El entorno debe asegurar la satisfacción de toda restricción mediante una política de* checking*,* maintenance *o* enforcement*.*

Existen diferentes formas de asegurar la no violación de ninguna restricción [3]. La política de *checking* consiste en comprobar la no violación de ninguna restricción una vez ejecutada la operación. El *maintenance* se caracteriza por, al detectar una violación después de ejecutar una operación, aplicar los cambios necesarios en los datos hasta alcanzar un nuevo estado que sea consistente. Finalmente, el *enforcement* es la reescritura de las operaciones de tal forma que se asegure que, al ser ejecutadas, no se incurre nunca en violación alguna.

Subobjetivo 2 *El entorno debe aceptar operaciones tanto de inserción, como de eliminación y modificación de datos, y debe ser validado con un caso de estudio real.*

Nuestro objetivo es alcanzar un resultado utilizable en la producción real de *software*. Para ello, es necesario tener en cuenta no solo las operaciones de inserción de datos, sino también las de eliminación y modificación de datos. Para validar su correcto funcionamiento, se vuelve indispensable su puesta a prueba con un caso de estudio real.

3 Aportación

Para alcanzar los anteriores objetivos proponemos un tipo de reglas lógicas que denominamos RGDs (Repair-Generating Dependencies). Un RGD es una regla lógica que permite detectar qué cambios en los datos originan una violación de una restricción de integridad, y, en la medida de lo posible, qué cambios adicionales en los datos se pueden aplicar para reparar dicha violación.

Formalmente, un RGD tiene la siguiente forma: $l_1 \wedge ... \wedge l_n \rightarrow r_1 \vee ... \vee r_m$; donde l_i es un literal que representa un dato de la base de información, su inserción o borrado, mientras que r_j representa únicamente la inserción o borrado de datos. Por ejemplo, el siguiente RGD:

$$\iota premiumUser(X) \wedge claims(X,Y) \wedge \neg prioritized(Y) \rightarrow \delta claims(X,Y) \vee \iota prioritized(Y)$$

Indica que se produce una violación cuando se inserta un nuevo usuario X como *premium*, éste tiene una reclamación Y, e Y no es una reclamación prioritaria. Dicha violación se puede reparar o bien eliminando la reclamación o bien insertándola como prioritaria.

RGDs para codificar restricciones. El anterior ejemplo ilustra la codificación de una restricción como un RGD. Sin embargo, normalmente una restricción se debe traducir a más de un RGD puesto que existe más de una combinación de inserciones/borrados de datos que producen la violación de la misma restricción. En un trabajo previo, hemos definido la forma de automatizar la conversión de una restricción escrita en OCL a sus diferentes RGDs [8].

Ejecutando los RGDs mediante el algoritmo *chase*, se puede obtener, dado un conjunto de inserciones/borrados de datos, un superconjunto de estos mismos inserciones/borrados tal que no viola ninguna restricción de integridad. Es decir, aplicar el *chase* sobre los RGDs equivale a mantener la consistencia de los datos con una política de *maintenance*.

Más aún, los RGDs pueden ser configurados para personalizar la forma de reparación de una restricción. En el anterior ejemplo, podría considerarse el caso que reparar la violación eliminando la reclamación de un usuario *premium* no es conveniente. Siendo así, se puede reescribir el RGD de la siguiente forma:

$$\iota premiumUser(X) \wedge claims(X,Y) \wedge \neg prioritized(Y) \wedge \neg \delta claims(X,Y) \rightarrow \iota prioritized(Y)$$

El RGD solo permite ahora reparar la restricción añadiendo las reclamaciones del usuario *premium* como prioritarias. Más aún, el RGD puede ser configurado para no forzar reparación alguna:

$$\iota premiumUser(X) \wedge claims(X,Y) \wedge \neg prioritized(Y) \wedge \neg \delta claims(X,Y) \wedge \neg \iota prioritized(Y) \rightarrow \bot$$

En tal caso, $\bot$ simboliza que, de satisfacerse el antecedente, se produce una violación irreparable y se advierte de ello al usuario. En otras palabras, el RGD ha sido configurado para ejercer la política de *checking*.

RGDs para codificar operaciones. Más interesante aún, los RGDs también pueden codificar las operaciones. Una operación muestra el siguiente comportamiento: si cuando se llama la operación, se satisface la precondición, entonces se ejecutan los efectos establecidos en la postcondición. Este comportamiento se

puede simular escribiendo un RGD que contenga, en su antecedente, la llamada a la operación junto a su precondición, y, en su consecuente, los efectos de la postcondición. E.g:

$$userUpgradeCall(X) \wedge \neg problematicUser(X) \rightarrow \iota premiumUser(X)$$

El anterior RGD codifica que, si al llamarse la operación *userUpgrade* con un usuario X tal que no es problemático (precondición), se inserta X como usuario *premium* (postcondición).

Al ejecutarse este RGD, se puede violar la restricción anteriormente presentada acerca de la no existencia de reclamaciones no prioritarias para un usuario *premium*. En dicho caso, dependiendo de si la restricción ha sido configurada para la política de *checking* o *maintenance*, se advertirá al usuario de una violación o se añadirán las cambios necesarios para evitarla.

Sin embargo, existe la posibilidad de configurar la operación para ejercer la política de *enforcement*, es decir, se puede modificar de tal forma que se asegure la no violación de la restricción. Esto se puede conseguir a través de la modificación del antecedente (la precondición) o del consecuente (la postcondición). E.g:

$$userUpgradeCall(X) \wedge \neg problematicUser(X) \neg claims(X, Y) \rightarrow \iota premiumUser(X)$$

Ahora la operación no viola la restricción puesto que se ha añadido la precondición que un usuario, para poder ser *premium*, primero tiene que tener cerradas todas sus reclamaciones pendientes.

4 Plan de Trabajo

1. Conseguir un caso de estudio real Nuestro trabajo debe partir de un caso de estudio real, si bien es difícil al ser las empresas reacias a ofrecer sus artefactos de desarrollo. Más aún, estamos en busca de un esquema conceptual con restricciones bien documentadas y no un mero diagrama de clases.

2. Codificar sus restricciones como RGDs Si bien ya hemos definido una traducción de un subconjunto del OCL a RGDs, estaría por ver qué restricciones no pueden ser convertidas automáticamente con esta propuesta. Nuestro objetivo aquí es estudiar como traducir las demás restricciones, qué limitaciones ofrece la traducción, y analizar empíricamente la eficiencia de las políticas de *checking* y *maintenance*. Se espera una eficiencia razonable, al menos, en el caso de *checking*.

3. Codificar sus operaciones como RGDs Seguidamente, se procedería a definir una traducción de las operaciones a RGDs. Se espera poder reaprovechar parte de la actual traducción de restricciones OCL a RGDs. Una vez capturadas las operaciones del caso de estudio como RGDs, se analizaría como modificar los RGDs automáticamente para implementar la política de *enforcement*. Esto es, como añadir nuevas condiciones en el antecedente del RGD o como añadir nuevas acciones en al consecuente para asegurar la no violación de ninguna restricción.

4. Implementar una capa de persitencia de datos Finalmente, se procedería a implementar una capa de persistencia automática de datos. Gracias a la existencia hoy en día de técnicas de persistencia automática de datos, no se esperan problemas en esta fase.

5 Relevancia

Importancia a la sociedad No hay duda de que la automatización de la construcción de un *software* a partir de su esquema conceptual es de sumo interés para la industria. Prueba de ello es el impulso que ha recibido en los últimos años el MDA/MDD. Nuestra propuesta, sin embargo, no se basa en la transformación de modelos, sino en la directa interpretación de ellos.

Importancia científica Des de la óptica de la comunidad científica, no sólo se contribuye a la resolución de un problema planteado durante los años 60, sino que nuestra propuesta avanza en consonancia con otros problemas paradigmáticos.

Por ejemplo, el problema de *state reachability*, saber si un esquema conceptual admite un estado que tenga, por lo menos, unas instancias I dadas por el usuario, se puede reducir a *integrity maintenance*. Intuitivamente, solo se deben configuran los RGDs para quitar las reparaciones consistentes en borrar datos dejando únicamente las de inserción, y se considera entonces la inserción de todo I. El problema de *state reachability* es, a su vez, un problema clave para resolver otros como son la redundancia/contradicción de restricciones, entre otros.

6 Conclusiones

Hemos presentado un método para la ejecución consistente y persistente de esquemas conceptuales. Nuestro método se basa en convertir las restricciones y operaciones de un esquema conceptual a unas reglas lógicas que llamamos RGDs. Dichos RGDs pueden ser configurados para implementar las diferentes políticas de consistencia de *checking*, *maintenance* y *enforcement*. Esta solución no solo permite la ejecución consistente de esquemas conceptuales sino que avanza en la línia con otros problemas como el *state reachability*, que a su vez, es clave para resolver la redundancia o contradicción de restricciones, entre otros.

Referencias

1. Teichroew, D., Sayani, H.: Automation of system building. Datamation **17**(16) (1971) 25–30
2. Soley, R., et al.: Model Driven Architecture. OMG white paper **308** (2000) 308
3. Olivé, A.: Conceptual schema-centric development: A grand challenge for information systems research. In: Advanced Information Systems Engineering. Volume 3520 of Lecture Notes in Computer Science. Springer (2005) 1–15
4. Hamann, L., Hofrichter, O., Gogolla, M.: On integrating structure and behavior modeling with OCL. In: International Conference on Model Driven Engineering Languages & Systems (MODELS 2012), Springer (2012) 235–251
5. Clavel, M., Marina, E., Miguel Angel, G.d.D.: Building an efficient component for ocl evaluation. Electronic Communications of the EASST **15** (2008)
6. Cabot, J., Teniente, E.: Incremental integrity checking of UML/OCL conceptual schemas. Journal of Systems and Software **82**(9) (2009) 1459–1478
7. Egea, M., Dania, C., Clavel, M.: MySQL4OCL: A stored procedure-based MySQL code generator for OCL. Electronic Communications of the EASST **36** (2010)
8. Oriol, X., Tort, A., Teniente, E.: Fixing up non-executable operations in UML/OCL conceptual schemas. In: Conceptual Modeling–ER 2014. Springer (to appear) (2014)

Rewriting Logic Techniques for Program Analysis and Optimization *

J. Sapiña[1]

DSIC-ELP, Universitat Politècnica de València,
Camino de Vera s/n, Apdo 22012, 46071 Valencia, Spain,
jsapina@dsic.upv.es

Abstract. Debugging is the process of locating and fixing errors in computer programs. Debugging is essential in software development and almost every programming language has its own specialized tools for the task, with high variability regarding their debugging power. This paper briefly describes ongoing research towards a Ph.D. thesis on universal debugging, a proposal to develop a program debugging framework that is potentially compatible with any programming language, under the supervision of Professor of Computer Science María Alpuente.

1 Introduction

Debugging is one of the most expensive yet crucial tasks in software development. According to recent research [10], developers spend half their time finding and correcting errors. This means that by 2013, the global cost of debugging is estimated to be $312 billion. Since effective and comprehensive automated debugging is still far away [12], software debugging will still be carried out manually by developers for quite some time with the help of dedicated debugging tools. In order to lessen the debugging costs, in this thesis we focus on improving the debugging task as much as possible by advancing these tools.

The quality of debugging tools is mostly directly related to the popularity of the language to which they serve. Widespread programming languages have at their disposal a large amount of debugging tools, resulting in a fierce tool competition that ultimately results in better debugging facilities. Other not-so-popular languages barely have (quality) tools available. The availability of good debugging tools becomes then one of the determining criteria in the selection of a programming language, to the possible detriment of other languages that best fit the project. In order to unify efforts and raise the quality of debugging tools, we propose the development of a universal debugger based on the tandem matching logic/$\mathbb{K}$ [11], which is a rewriting-based, executable semantic framework in which programming languages, type systems and formal analysis tools can be defined. The main advantage of our approach is to make available almost

* This work has been partially supported by the EU (FEDER) and the Spanish MEC project ref. TIN2010-21062-C02-02, the Spanish MICINN complementary action ref. TIN2009-07495-E, and by Generalitat Valenciana ref. PROMETEO2011/052. J. Sapiña was supported by FPI-UPV grant SP2013-0083.

for free, or at a very low cost, quality debugging tools for any language whose semantics has been defined in the $\mathbb{K}$-Framework.
Plan of the paper. Section 2 gives an overview of the related literature regarding this research. Section 3 briefly describes the proposed universal debugging framework including the hypothesis, objectives, methodology, and some technical aspects. Finally, Section 4 concludes.

2 Related Work

In the past, there have been numerous attempts to develop semantics-based program analysis and debugging techniques. Field et al. proposed in [8] a parametric program slicing technique that relies on semantic specifications together with dynamic dependence tracking and an equational logic designed to serve as a transformation toolkit for the analysis and manipulation of imperative languages [7]. However, the huge gap between theory and practice has been a handicap for the adoption of this technology. In the last decade, Roşu and his team have greatly advanced in the research of semantics-based analysis techniques, thus narrowing the aforementioned gap. This has been achieved with the development of $\mathbb{K}$ [14], a rewriting-based executable semantic framework in which programming languages syntax and semantics can be defined. $\mathbb{K}$ consists of (i) $\mathbb{K}$ *rewriting* [15], a concurrent rewrite abstract machine that enhances the potential for concurrency of a rewrite theory, (ii) the $\mathbb{K}$ *technique* [13], which transforms $\mathbb{K}$ *rewriting* into an effective framework for the definition of programming languages, and (iii) a specialized notation. A prototype implementation of the $\mathbb{K}$-Framework can be found in the $\mathbb{K}$-Maude tool [16], which successfully translates $\mathbb{K}$ semantics definitions into Maude [6], a high-performance reflective specification language and system supporting both rewriting logic (RWL) and purely equational logic. This combination of features, together with the reflective ability of $\mathbb{K}$ to manipulate the source code of a program as a piece of data, allows one to execute and transform both the source code and any output results, including execution traces, by standard rewriting. Many languages that are widely distributed throughout the industry such as C, Java, Python, Haskell, and Verilog have a semantics definition written in $\mathbb{K}$. To the best of our knowledge, no extensible RWL-based universal debugging framework that applies to all these languages has been formally developed to date.

3 Universal Debugging

In this section we present the hypothesis, objectives, and methodology to be followed during the thesis development, together with some challenges that need to be addressed in order to achieve our objectives.

3.1 Hypothesis

In the $\mathbb{K}$-Framework, it is possible to virtually execute a program through the real execution of a suitable rewrite system that defines the semantics of

the language in which the program is written. Hence, it is also possible to create a parametric framework for the analysis and inspection of both, the virtual and real executions, resulting in a universal debugging framework potentially compatible with any programming language whose semantics has been previously formalized in the given scheme.

3.2 Objectives

The objectives of this thesis are as follows:

- Formal development of program transformation and analysis techniques for the inspection and analysis of RWL computations in Maude, with particular emphasis on efficiency.
- Application of these techniques to real languages whose semantics are formalized in the $\mathbb{K}$-Framework.
- Development of the first generic, extensible, modular, and language-independent, RWL-based debugging environment, which will allow the inspection of a program written in any language supported in the $\mathbb{K}$-Framework.
- Enrichment of the debugging environment by integrating different debugging paradigms (e.g., algorithmic debugging, reverse debugging, automated debugging, etc.)
- Empirical evaluation and analysis of the results.

3.3 Metodology and Workplan

This thesis will be carried out according to the terms dictated by the new doctoral program governed by the RD 99/2011, which points a tree-year limit for this task. Currently, the author is at the end of the first year. The proposed planning for the whole period is as follows.

Stage One In this first stage, we focus on developing suitable techniques to identify and extract the information that is relevant to the source program being debugged from the execution in $\mathbb{K}$ of the language semantics. The milestones of this stage are as follows:

- Development of RWL-based trace slicing, dependence analysis, and program animation techniques that help locate those pieces of code that are relevant to the analysis.
- Implementation in Maude of prototype tools for the developed techniques, with special focus on efficiency.
- Optimization of the developed techniques and tools.

This stage has been recently completed, resulting in the techniques of [1,3,4] and prototype tools *i*Julienne [2,9] and Anima [5].

Stage Two In this stage, which is currently under early development, we aim to formalize and implement the core of the universal debugger by pursuing the following specific objectives:

- Theoretical and practical study of the $\mathbb{K}$-Framework.
- Formal development of a universal debugger in $\mathbb{K}$.
- Instantiation of the debugging framework for different languages.
- Implementation of a prototype universal debugger and experimental evaluation, focusing on both efficiency and usability.

Stage Three Finally, we focus on the application of the debugging framework and the practical evaluation of the results. The milestones of this third phase are:

- Applications and extensions of universal debugging and transformation (e.g., algorithmic debugging, reverse debugging, automated debugging, dependence analysis, and program and trace slicing).
- Empirical evaluation and analysis of the results.

An international research internship is planned for this final stage. Moreover, scientific dissemination activities such as attendance to workshops and conferences and publication of journal articles will be conducted at every stage.

3.4 Technical Challenges

We have identified several challenges that must be overcome in order to achieve our goal. First, it is essential to perform detailed analysis of generic Maude execution traces, especially with regard to the application of built-in operators and equations, which are silently performed in Maude in order to optimize the performance. In fact, the details of these applications are only available in plain text format and cannot be manipulated as a meta-level expression by Maude. Our efforts so far have led us to develop a technique and corresponding implementation that faithfully reproduces instrumented versions of Maude execution traces without jeopardizing efficiency (for more details, see Section 6 of [4]). We consider this a great advance since we need not transform a given Maude specification in any way to fully capture the application of either built-in operators or equations.

Another challenge is the accurate detection and extraction of the information related to the program being debugged, getting rid of any structural information required to properly execute the generated rewriting system that models the language semantics. We plan to refine our existing trace slicing techniques to focus only on the information related to the program that is being debugged by suitably overlooking this structural information introduced by the $\mathbb{K}$-Framework.

Finally, one major difficulty is the ability to trace back to the original source code any errors that might be found during the analysis of rewriting executions. We plan to formally define the precise relation between the source code of the program being debugged and the Maude terms representing them to be able to automatically translate back and forth the original source code expressions to its Maude term counterpart.

4 Conclusions

This paper summarizes a proposal to establish the basis of universal program debugging by creating the first RWL-based, language-independent, universal debugging framework.

We consider that our proposal is very suitable for the integration of a variety of debugging techniques, regardless of how different they are, such as reverse debugging and algorithmic debugging. Concerning reverse debugging, there exists a number of schemes, each one based on a different

encoding of the machine states. By using RWL, one can provide a uniform representation and easily record the sequence of rules applied, together with their associated information. As for algorithmic debugging, which is based on the idea of an oracle (typically the user) that guides the execution, we think that the program animation techniques we formalized in [4] can support the algorithmic style.

References

1. Alpuente, M., Ballis, D., Frechina, F., Sapiña, J.: Parametric Exploration of Rewriting Logic Computations. In: Proc. SCSS 2013. EPiC, vol. 15, pp. 4–18. EasyChair (2013)
2. Alpuente, M., Ballis, D., Frechina, F., Sapiña, J.: Slicing-Based Trace Analysis of Rewriting Logic Specifications with *i*Julienne. In: Proc. ESOP 2013. LNCS, vol. 7792, pp. 121–124. Springer-Verlag (2013)
3. Alpuente, M., Ballis, D., Frechina, F., Sapiña, J.: Inspecting Rewriting Logic Computations (in a Parametric and Stepwise Way). In: Proc. SAS 2014. LNCS, vol. 8373, pp. 229–255. Springer-Verlag (2014)
4. Alpuente, M., Ballis, D., Frechina, F., Sapiña, J.: Exploring Conditional Rewriting Logic Computations. Journal of Symbolic Computation (to appear, 2014)
5. The Anima Web site (2014), Available at: `http://safe-tools.dsic.upv.es/anima`
6. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.: All About Maude: A High-Performance Logical Framework. Springer-Verlag (2007)
7. Field, J.: A Simple Rewriting Semantics for Realistic Imperative Programs and its Application to Program Analysis. In: Proc. PEPM 1992. pp. 98–107. Yale University (1992)
8. Field, J., Ramalingam, G., Tip, F.: Parametric Program Slicing. In: Proc. POPL 1995. pp. 379–392. ACM (1995)
9. The *i*Julienne Web site (2014), Available at: `http://safe-tools.dsic.upv.es/iJulienne`
10. Cambridge Venture Project. Judge Business School at the University of Cambridge, Available at: `http://www.jbs.cam.ac.uk/media/tag/cambridge-venture-project-cvp/`
11. The $\mathbb{K}$-Framework Web site (2014), Available at: `http://www.kframework.org`
12. Parnin, C., Orso, A.: Are automated debugging techniques actually helping programmers? In: Proc. ISSTA 2011. pp. 199–209. ACM (2011)
13. Roşu, G., Şerbanuţă, T.F.: An Overview of the $\mathbb{K}$ Semantic Framework. The Journal of Logic and Algebraic Programming 79(6), 397–434 (2010)
14. Roşu, G., Şerbanuţă, T.F.: $\mathbb{K}$ Overview and SIMPLE Case Study. In: Proc. K 2011. ENTCS, vol. 304, pp. 3–56. Elsevier Science (2014)
15. Şerbanuţă, T.F., Arusoaie, A., Lazar, D., Ellison, C., Lucanu, D., Roşu, G.: The $\mathbb{K}$ Primer (version 3.3). In: Proc. K 2011. ENTCS, vol. 304, pp. 57–80. Elsevier Science (2014)
16. Şerbanuţă, T.F., Roşu, G.: $\mathbb{K}$-Maude: A Rewriting Based Tool for Semantics of Programming Languages. In: Proc. WRLA 2010. LNCS, vol. 6381, pp. 104–122. Springer-Verlag (2010)

Mejorando los Sistemas Colaborativos y Post-WIMP mediante la Especificación de Requisitos

Miguel A. Teruel (doctorando), Pascual González, Elena Navarro (directores)

LoUISE Research Group, Universidad de Castilla – La Mancha
miguel@dsi.uclm.es, {Pascual.Gonzalez, Elena.Navarro}@uclm.es

Resumen. Una especificación de requisitos adecuada es fundamental para alcanzar la calidad de los productos software a desarrollar. No obstante, cuando se va a desarrollar un sistema colaborativo, las técnicas de Ingeniería de Requisitos (IR) actuales no son suficientemente expresivas para especificar los requisitos de dichos sistemas. Esto es debido a la complejidad inherente de la colaboración entre usuarios, así como a la necesidad de consciencia (*awareness*). Además, la manera de interactuar con estos sistemas colaborativos ha evolucionado enormemente hacia el uso de interfaces más complejas, más allá de los clásicos sistemas de escritorio, hacia las denominadas interface Post-WIMP (*Windows, Icons, Menus, Pointer*). En ellas, el awareness cobra aún más importancia debido a la necesidad de los usuarios de ser conscientes de su contexto: los artefactos con los que interactuar, el posicionamiento del usuario en mundos virtuales, o las capacidades del usuario o las de los demás son elementos de los que el usuario ha de ser consciente. Así, esta tesis tiene como objetivo solventar este problema mediante el desarrollo de un framework de IR capaz de especificar los requisitos de los sistemas colaborativos y Post-WIMP, con especial interés en los requisitos de awareness acerca del contexto del usuario.

1 Motivación

Un sistema colaborativo es un producto software que permite a grupos de usuarios involucrarse en una tarea u objetivo común. Por tanto, estos sistemas nos permiten comunicarnos con nuestros colaboradores, así como trabajar de forma colaborativa. Esto implica que el grupo de usuarios podrá coordinar sus actividades, solucionar problemas, editar documentos, etc., todo ello mediante el uso de tecnologías específicas. Además, la forma en la que interactuamos con un ordenador ha evolucionado hacia un nuevo paradigma más allá de las clásicas ventanas, iconos, menús y punteros (WIMP). Este nuevo paradigma, denominado Post-WIMP está basado en diferentes tecnologías como la realidad virtual, las interfaces tangibles, el reconocimiento de gestos o los cada vez más populares ordenadores corporales (*wearable computers*). Esto hace que el desarrollo de estos sistemas sea significativamente diferente del de los sistemas WIMP.

Debido a ello, uno de los principales desafíos en el desarrollo de estos sistemas, especialmente cuando consideramos a sus usuarios finales, es la especificación de requisitos. En ese sentido, esta tesis continuará con la línea seguida por el grupo de

investigación LoUISE; la mejora del desarrollo de sistemas colaborativos y Post-WIMP desde una perspectiva de calidad, centrándose en la especificación de requisitos.

2 Relevancia

Dada la popularidad de Internet en los últimos años, muchos entornos clásicos de trabajo han evolucionado hacia sistemas colaborativos basados en Web. Actualmente millones de usuarios comparten información y trabajo colaborando remotamente en diversos dominios, ya sea de forma síncrona o asíncrona. Por ello, los resultados esperados de esta tesis podrán aplicarse a un considerable número de dominios entre los que destacan la administración, negocios, educación, medicina o juegos on-line.

Además, la inclusión de nuevas características colaborativas y sociales en la mayoría de las aplicaciones existentes hace necesario el desarrollo de nuevas herramientas, entornos y metodologías que den soporte a dichas características. Concretamente, esta colaboración está tendiendo a ser realizada de una forma más natural debido al auge de la interacción Post-WIMP. Debido a ello, las limitaciones del desarrollo de los sistemas interactivos actuales deben ser solventadas, especialmente para dar soporte al awareness.

3 Cuestiones de Investigación

Teniendo en cuenta la revisión de la literatura realizada, así como el trabajo previo del grupo LoUISE, fueron identificadas las siguientes deficiencias y necesidades dentro del desarrollo de los sistemas colaborativos y Post-WIMP:

1. Se identificaron diferentes propuestas para el desarrollo de sistemas colaborativos [2, 10] y Post-WIMP [7, 11]. No obstante, éstas sólo se centran en actividades de diseño, tratando rara vez las primeras etapas del proceso de desarrollo software.
2. Algo crucial en desarrollo de los sistemas colaborativos es la identificación de las necesidades de awareness [3], o sea, la consciencia sobre qué artefactos manipular, con qué usuarios interactuar, etc. Así, dicho awareness mejorará la usabilidad de las aplicaciones a desarrollar, permitiendo la colaboración entre usuarios de una forma más natural.
3. Otro importante aspecto del desarrollo de sistemas colaborativos, y especialmente en entornos Post-WIMP, tiene que ver con la adaptación. Especialmente cuando un sistema tiene que ser usado en diferentes contextos como así señalan estándares internacionales como ISO/IEC 9126-1:2001 [6] o 25010:2011 [5].

Basándonos en estas deficiencias y necesidades, las cuestiones de investigación de esta tesis serían las siguientes:

- *RQ1*: ¿Cuáles, en el caso de haberlas, son las deficiencias de las técnicas de IR actuales cuando se usan para especificar sistemas colaborativos? De haberlas, ¿Cómo podrán ser mejoradas dichas técnicas?

- *RQ2*: ¿Cuáles, en el caso de haberlas, son las deficiencias de las técnicas de IR actuales cuando se usan para especificar sistemas Post-WIMP? De haberlas, ¿Cómo podrán ser mejoradas dichas técnicas?
- *RQ3*: ¿Qué requisitos de awareness podrían mejorar la calidad percibida por el usuario en sistemas colaborativos y Post-WIMP?
- *RQ4*: ¿Qué requisitos de adaptación ayudarían a los sistemas colaborativos y Post-WIMP a ser adaptables a su contexto de uso?

Los principales desafíos tecnológicos asociados con estas cuestiones de investigación están relacionados con cuan novedosos son. Por un lado, a pesar de que los sistemas colaborativos llevan siendo estudiados desde los años noventa, su especificación de requisitos ha sido tratada muy raramente. Por otro lado, las interfaces Post-WIMP son considerablemente novedosas, por lo que tendremos que enfrentarnos a unos sistemas emergentes para los que existen pocas alternativas para su especificación.

4 Metodología y Plan de Trabajo

Durante el desarrollo de esta tesis se seguirá la metodología *design science research* [4] y se realizarán las siguientes actividades (reflejadas en la Fig. 1):

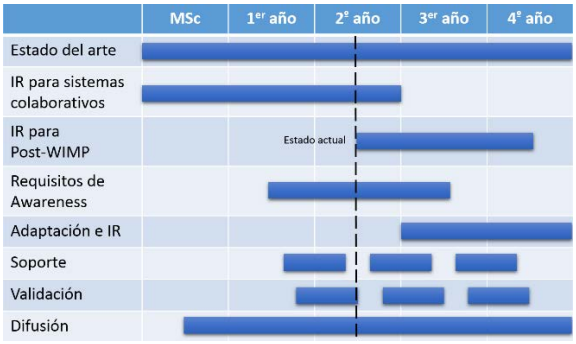


Fig. 1. Planificación de actividades durante la tesis

- *Estado del arte*: Investigación sobre IR, entornos colaborativos y Post-WIMP, interpretaciones de awareness y adaptación. Esta actividad se llevará a cabo durante toda la tesis.
- *IR para sistemas colaborativos*: Desarrollo de una propuesta integrada de IR para sistemas colaborativos, guiado por técnicas de desarrollo guiado por modelos (MDD) aprovechando la experiencia previa del grupo LoUISE en este ámbito [13]. También se considerará la modificación de los procesos descritos por las metodologías existentes para su adaptación a sistemas colaborativos.
- *IR para Post-WIMP*: Se evaluará si la propuesta desarrollada para sistemas colaborativos es adecuada para entornos Post-WIMP. Consecuentemente, se analizarán y efectuarán las adaptaciones necesarias para extender dicha propuesta a entornos Post-WIMP.

- *Requisitos de Awareness*: Identificación, desarrollo y validación de una interpretación de awareness integrada reuniendo todos los requisitos de awareness que un usuario de un sistema Post-WIMP (sea o no colaborativo) pueda necesitar o esperar.
- *Adaptación e IR*: Análisis de la propuesta de IR en relación a las características de adaptación necesarias parar entornos Post-WIMP. Además, dicha propuesta se integrará en una nueva versión de UsiXML (*USer Interface eXtensible Markup Language*) [8]. Este lenguaje, en cuya definición ha participado activamente el grupo LoUISE, es ampliamente aceptado por la comunidad por sus capacidades a la hora de especificar necesidades de adaptación en entornos WIMP. Por ello, será extendido o adaptado para tratar con sistemas Post-WIMP.
- *Soporte*: A lo lardo de esta tesis se implementarán varias herramientas para dar soporte a la propuestas de IR desarrollada.
- *Validación*: Con el objeto de validar la propuesta de IR, se implantarán diversos prototipos que serán aplicados a dominios de distinta índole como *e-learning* o rehabilitación física en los que el grupo LoUISE tiene experiencia previa [1, 9, 12].
- *Difusión*: Durante el proceso de tesis se enviarán publicaciones a congresos y revistas tanto nacionales como internacionales.

5 Resultados Actuales y Trabajo Pendiente

Esta tesis, la cual comenzó estudiando si las técnicas de IR actuales son adecuadas para especificar requisitos de sistemas colaborativos, ha tenido su principal hito con CSRML (*Collaborative Systems Requirement Modeling Language*) [16], un lenguaje de IR para dichos sistemas que está basado en *i**. Cabe destacar que el proceso de investigación ha sido completamente guiado por evaluaciones empíricas, validando cada resultado obtenido [14, 15]. No obstante, a pesar de la considerable aceptación que ha tenido CSRML, varías líneas de trabajo continúan abiertas. Entre otras cosas, en relación a la RQ1, se deberá desarrollar una metodología completa para dar soporte a CSRML con el fin de servir de ayuda a la hora de elicitar, modelar, analizar validar y verificar requisitos de sistemas colaborativos. Para ello se estudiarán las metodologías actuales de IR orientada a objetivos y su posible integración con CSRML.

Por otro lado, el trabajo relativo a la RQ2 acaba de dar comienzo. Para ello se está especificando un caso de estudio que servirá para evaluar si CSRML es adecuado para especificar requisitos de entornos Post-WIMP. En caso contrario, CSRML será adatado para poder especificar requisitos de este tipo de sistemas. En relación a la RQ3, se ha desarrollado una nueva interpretación de awareness que reúne los requisitos de awareness de sistemas colaborativos y Post-WIMP. Esta interpretación ha sipo positivamente evaluada por medio de una serie de encuestas a usuarios. No obstante, para evaluarla más exhaustivamente se deberán llevar a cabo experimentos adicionales en los que los sujetos experimentales se involucren de forma más activa. Finalmente, el análisis de la RQ4 está dando sus primeros pasos, comenzando con el estudio de una posible integración de CSRML y UsiXML con el fin de dotar a nuestro lenguaje de los mecanismos de adaptación al contexto necesarios en entornos Post-WIMP.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Economía y Competitividad y por los fondos FEDER de la UE bajo el proyecto insPIre (TIN2012-34003), así como por el Ministerio de Educación, Cultura y Deporte con la beca FPU (AP2010-0259).

Referencias

1. Fardoun, H. et al.: eLearniXML: Towards a model-based approach for the development of e-Learning systems considering quality. Adv. Eng. Softw. 40, 12, 1297–1305 (2009).
2. Garrido, J.L. et al.: A Software Architecture Intended to Design High Quality Groupware Applications. Software Engineering Research and Practice. pp. 59–65 (2005).
3. Gutwin, C., Greenberg, S.: A Descriptive Framework of Workspace Awareness for Real-Time Groupware. Comput. Support. Coop. Work. 11, 3, 411–446 (2002).
4. Hevner, A.R. et al.: Design Science in Information Systems Research. MIS Q. 28, 1, 75–105 (2004).
5. ISO/IEC 25010: Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. (2011).
6. ISO/IEC 9126-1: Software engineering - Product quality: Quality model. (2001).
7. Jacob, R.J.K. et al.: Reality-based interaction: a framework for post-WIMP interfaces. SIGCHI Conference on Human Factors in Computing Systems (CHI'08). pp. 201–210 (2008).
8. Limbourg, Q. et al.: USIXML: A Language Supporting Multi-path Development of User Interfaces. In: Bastide, R. et al. (eds.) Engineering Human Computer Interaction and Interactive Systems. pp. 200–220 Springer Berlin Heidelberg (2004).
9. Martínez, D. et al.: A Framework to Develop VR Interaction Techniques Based on OpenInterface and AFreeCA. In: Campos, P. et al. (eds.) 13th International Conference on Human-Computer Interaction (INTERACT'11). pp. 1–18 Springer Berlin Heidelberg, Lisbon, Portugal (2011).
10. Molina, A.I. et al.: CIAM: A methodology for the development of groupware user interfaces. J. Univers. Comput. Sci. 14, 9, 1435–1446 (2008).
11. Molina, J.P.: A Structured Approach to the Development of 3D User Interfaces. University of Castilla-La Mancha (2008).
12. Montero, F. et al.: Computer-aided relearning activity patterns for people with acquired brain injury. Comput. Educ. 57, 1, 1149–1159 (2011).
13. Navarro, E. et al.: A metamodeling approach for requirements specification. J. Comput. Inf. Syst. 46, 67–77 (2006).
14. Teruel, M.A. et al.: A CSCW Requirements Engineering CASE Tool: Development and Usability Evaluation. Inf. Softw. Technol. 56, 8, 922–949 (2014).
15. Teruel, M.A. et al.: Analyzing the Understandability of Requirements Engineering Languages for CSCW Systems: A Family of Experiments. Inf. Softw. Technol. 54, 11, 1215–1228 (2012).
16. Teruel, M.A. et al.: CSRML: A Goal-Oriented Approach to Model Requirements for Collaborative Systems. In: Jeusfeld, M. et al. (eds.) 30th International Conference on Conceptual Modeling (ER'11). pp. 33–46 Springer Berlin Heidelberg, Brusells, Belgium (2011).

Extracting Business Models for Software Process Support

Doctoral Student: C. Arévalo, IWT2 Group. University of Seville, Spain,
carlosarevalo@us.es

Doctoral thesis supervisors: I. Ramos, M.J. Escalona, IWT2 Group. University of Seville, Spain, {iramos, mjescalona}@us.es

1 Software Process Support

Software companies, as all over our current world, are involved in complex changes due to new globalization rules, such as joint venture, mergers and acquisitions, as well in new collaborations among stakeholders such as outsourcing, offshoring or near shoring, among others. Today, most of these organizations take into consideration the Business Process Management (BPM) [42, 43] paradigm to support its software business process (we mean: managing all kinds of software project lifecycles); these business software processes are more unpredictable [1] than other production processes, changing continuously and tightly tied to communication between single people, working teams' protocols and company contracts. These organizations also use models, computer-aided tools, standards and best practices in the market. They must adapt their knowledge to manage the requirements of a global and changing environment, using at once multiple kinds of sophisticated and automated tools to gain competitive advantage in the market. These computer-aided tools usually evolve continuously for supporting new paradigms, architectures, methodologies and languages as well as different databases; they may be tailor-made systems or customizable market products, each one with different languages and data structures and details of each business process; these tools may change for different reasons: new uncovered requirements, global contracts with providers, platform unification, etc. It is a big challenge managing all of these complex organizational factors together with the use of a population of different kind of tools that are in a production environment or well that we find them in a migration project between different tools.

It is difficult to dynamically obtain an integrated conceptual view of all of the features of the whole business software model of a complex organization. Among others goals for business processes, we need global data view. We will focus on the database structures of these computer-aided tools that usually are based on the relational model and they are the most stable view of business along the lifecycle of processes. We are looking for common data structures that lie inside the database of each automated tool, because of every business model for software has essential and common data structures for business models, business processes structures, resources involved and metrics for processes, etc.; these structures could be replicated with different names and details but they have same essential purpose.

The concept of metamodel is a key artifact to get this common structures. With regard to a metamodel for software process support, we will suggest an approach based on this concept to dynamically obtain that conceptual global data view for the business

experts. With this global conceptual business data view we look for interoperability, defined as:

Table 1. IEEE definition for interoperability

Interoperability: Ability of a system or a product to work with other systems or products without special effort on the part of the customer. Interoperability is possible by the implementation of standards.

At the best of our knowledge, for software projects management, we do not know approaches that offer a conceptual global view of the business data structures that is partially mapped inside each automated tool. The interoperability will be possible between pairs of automated tools and between each tool and a system that supports this global data view, thus allowing dynamic side-by-side migrations or bottom-up migrations of business data structures and states living in each environment, and also it could offer new facilities for global reports that are not available in local computer-aided tools.

2 Related work

We do not have space to detail the work done in this section, so briefly, to manage different computer-aided tools for software process support, each one with different software platform, we have to look at standards, best practices in software business projects, models and the technology for these systems, because of we must discover business models from these systems and we focus on their persistence layer. We have explored:

Approaches for Software Process Support.

There exists a recent systematic literature review (SLR) for *software business process support* [3], carried out by our research group[1], covering software process modelling languages, methods, standards and best practices for software process support. It briefly concludes that, among others approaches, Business Process Modeling Notation (BPMN) [9] is the preferred technology to model processes, because of its simplicity, standardization and support for execution processes. Many UML[2]-based approaches have been developed, but there are actually two main languages:

- SPEM 2.0 [4] which provides a language for software methodologies.
- ISO/IEC 24744-Software Engineering Metamodel for Development Methodologies [2] which offers guidelines to support the identification of process models as well as the improvement of consistency and uniformity in the definition of these standards.

By the relevance of both, BPMN and concept of metamodel, also our research group has proposed a BPMN metamodel [5] for software process support that extends existing metamodels for UML activity diagrams and BPMN diagrams.

[1] `http://www.iwt2.org/`

[2] Unified Modelling Language (UML) [30]

Model Driven Engineering (MDE) and Standards.

OMG Model Driven Architecture (MDA) [11] is the major exponent of MDE paradigm [25] that proposes different level of abstraction to gain independency and interoperability among different views of a universe of discourse; Meta Object Facility (MOF) [12] introduces the metamodel concept and Query/View/Transformation (QVT) [23] is a language for mapping artifacts between models and metamodels.

OMG Architecture Driven Modernization (ADM) [10] is an approach based on MDA that proposes metamodels and processes for legacy software modernization because these systems are not always well documented and, if they are highly complex, it is not easy replacing them for a new system [44, 45, 46]. Maintaining legacy software can be too expensive; very often these ones do not support the concept of BPM. We need to cope with them but also working with modern approaches as BPM to provide them with new functionality.

Also it's important reviewing the Information Management Model (IMM) [13] as an OMG proposal of interoperability [27] that, among others, establishes metamodel views for tables, keys, routines and triggers of a relational database.

Approaches for Reverse Engineering Relational Databases.

Because we are focused on the persistent layer of existing software project management tools and as ADM [10] related approaches, we have also revised among others:

- Works by Pérez-Castillo, Piattini, et al. [14, 15, 16, 17, 18] uses the concept of *process archaeology* and manages relational schemas and application code.
- Works by Cánovas Izquierdo & García Molina [19, 20, 21]. The main example is a DSL tool named *Grammar to Model Language (Gra2Mol)* [22], which acts as a bridge that goes from grammar-ware to model-ware.

  Both references are good implementations methods based on an ADM [10] approach and both work with basic relational database metamodels, although none of them cover sufficiently trigger-based rules.

We must also take into account other research contributions that extract business rules from relational databases to enrich UML class diagrams with OCL statements as [31, 32], [37] and we are interested in this aspect.

3 Proposed solution

The main goal is *"Obtaining an integrated view of essential and common data of the whole business model for software process support whose structures and statuses are replicated inside several automated tools for project life cycle; that data view must have good understandability at the business expert level".*

We will base on metamodels, selecting a BPMN metamodel proposed in [5], represented by UML2 profiles plus OCL [36] constraints. This one captures common artifacts for a software process as process models, composition of tasks, deliverable products, stakeholders, metrics, etc., offering a suitable mechanism for process definition by covering the main software process concepts and giving ISO/IEC TR 24744 [2] compliance. This one, positioned at CIM level, is proposed and based on other contributions, like Navigational Development Techniques (NDT) [33] as a methodological proposal and the software business environment named NDTQ-Framework [35], based

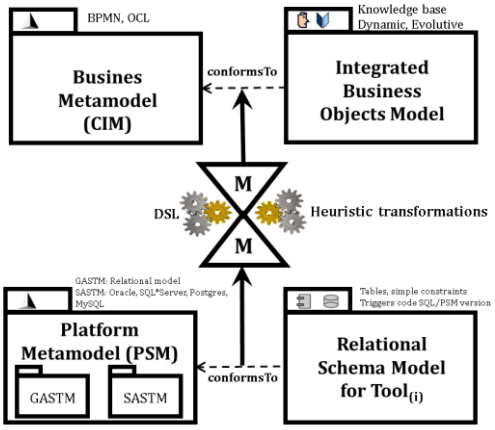
on NDT, which is being successfully used in several actual software development projects [39, 40, 41] with organizations around us.

Our approach should provide simplicity to develop a global model-based solution. We will work at different MDA levels of abstraction and we will propose specific heuristics with QVT transformations between artifacts of two levels. We will go in an ascending ADM [10] horseshoe trip, from source PSM metamodels for relational databases to our CIM target metamodel. We explain this:

- **PSM metamodel for relational behavior.** We must extend the capabilities of metamodels proposed in works [14, 15, 16, 17, 18, 19, 20, 21] and [28] and in reference with IMM [13], [27] for relational databases, specially for capturing meaning for database triggers. Triggers are implemented in proprietary procedural languages that are based on variations of ISO SQL/PSM[3] [24], so PL/SQL, Transact-SQL, pgSQL and MySQL stored procedures.
- **Heuristics from PSM to CIM.** We suggest a heuristic method to transform structures from databases at PSM level to our target BPMN metamodel of CIM level. We mean: tables, all kinds of declarative constraints and tailored triggers at each database project management tool into classes and OCL constraints of our CIM target metamodel. That heuristic may consider:
  - Grouping table structures in correspondence with CIM metamodel metaclasses, considering cardinalities or correspondences for their associations.
  - Mapping of all kinds of database constraints to OCL business rules expressed over classes discovered, considering works like [31, 32], [37] for this purpose.

  Fig.1 represents a schema of the proposal that include models, metamodels, a Domain Specific Language (DSL) and heuristic methods. The proposed method will be a model-to-model (M2M) MDA [11], where:

Fig. 1. Heuristic transformations from PSM into CIM



- Every model is an instance or *conforms to* its metamodel.

[3] SQL/PSM first appeared in ISO SQL:1999 [34] as standard procedural language to code routines like procedures, functions and triggers. SQL/PSM still remains in ISO SQL:2011.

- Metamodels, both at PSM and CIM level, are defined with a DSL based on UML2 profiles [26].
- PSM metamodel needs two packages: Generic Abstract Syntax Metamodel (GASTM) and Specific Abstract Syntax Metamodel (SASTM) to capture enough behavior of every concrete relational model for software process support.
- The heuristic transforms artifacts between different levels of abstraction and besides it merges different source models to get a unique and global business data object view that conforms to the referenced CIM metamodel.

4 Preliminary work

In the scope of this thesis work the main tasks are:

- **Detecting problems.** We have analyzed several actual software development projects (R+D+i collaborations between our research group and Private or Public institutions) [39, 40, 41] which have allowed to gather problems, proposals, desires and goals of business experts in software process support. This register has enabled us to establish the study focus.
- **State-of-the -art**. We have structured the study in a) approaches for software business process support and b) approaches for discovering models from legacy systems; both areas are briefly explained at the *related work* chapter of this document.
- **Requirements Analysis**. We need precise requirements to develop our research approach. The tasks can be grouped in lists of requeriments for: a) extending BPMN metamodel for software process support taked from [5], b) extending existing metamodels for relational databases, c) defining a concrete syntax for used metamodels, d) defining heuristic transformations between our specific PSM and CIM levels.
- **Metamodels for relational databases with triggers**. We began with Generic Abstract Syntax Trees (GASTM) and Specific Abstract Syntax Trees (SASTM) Metamodels for relational databases, used by research work such as [14, 15, 16, 17, 18] and [19, 20, 21] and existing as free downloadable Atlanmod metamodels [28]. Triggers are a kind of ECA rules whiches we are also interested in, but these works don't treat them adequately, so we have compared with standard IMM [13], [27]; thus we have been developing a metamodel that includes them; however, we are still dealing with refining these metamodels.

5 Expected contributions

We hope to reach several goals in the scope of the thesis work:

- To define needed extensions over referenced Relational Metamodels with the aim to capture structural relational behavior including triggered-based rules.
- To propose extensions to the metamodel [5] of CIM level with the aim for defining more than one software business model, software business models as software business processes and tasks and making correspondences between software business models, business processes and organization units for avoiding unnecessary redundancy. These extensions should facilitate merging several software business data models.

- To define a concrete syntax to all metamodels as a DSL: PSM and CIM based on UML2 profiles [26].
- To propose inference methods to discover database artifacts at PSM level and transform them into BPMN knowledge at CIM level:
  - Gathering both a) table structures with their embedded declarative constraints and b) external triggers that are linked to each table.
  - Establishing correspondences between relational structures of PSM level and CIM metaclasess of the extended BPMN metamodel.
  - Proposing heuristic transformations based on QVT language for metaclasess, attributes, constraints and triggered rules from PSM level to CIM level, reaching a *global business view of the business software model* for the whole organization, expressed as a UML2 profile for a BPMN metamodel enriched with OCL rules.

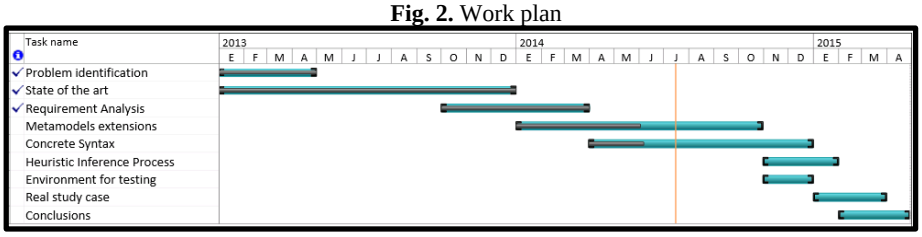
6 Plan for evaluation and validation

Our research group often collaborates with several companies and institutions in actual succeeded software projects dealing with R+D+i [39, 40, 41] along the last ten years. It is not the aim of this paper the justification of this selection but it is done in [33]. Also we have carried out a study for selecting modeling tools that supports, among other features, UML2 profiling that we need; the selected tool was Enterprise System Architect (EA) [48]; thus we will use it and also we will consider QVT transformations and Java code implementations as the environment for testing and validating selected study cases around us.

We need detailed criteria for measuring the expert understandability of the global business data view obtained.

Fig. 2 is a Gantt chart representing tasks in progress and pending tasks.

Fig. 2. Work plan



7 Acknowledgements

This research has been supported by MeGUS, Ministerio de Ciencia e Innovación (TIN2013-46928-C3-3-R), and NDTQ-Framework project (TIC-5789) of Junta de Andalucía, Spain.

8 References

1. **Ruiz-González, F.; Canfora, G.** "Software Process: Characteristics". Technology and Environments., vol. 5, pp. 5-10 (2004)
2. **ISO/IEC 24744:2007** "Software Engineering - Metamodel for Development Methodologies". ISO (2007)
3. **García-Borgoñón, L; Barcelona, M.A.; García-García, J.A.; Alba, M.; Escalona, M.J.** "Software Process Modeling Languages: a Systematic Literature Review". ISS Vol. 56. 103-116. (2014)

4. **SPEM 2.0.** "SPEM, Software & Systems Process Engineering Metamodel specification". http://www.omg.org/spec/SPEM/, (2008). Last acc. 07/2014
5. **García-Borgoñón, L.; García-García, J.A.; Escalona, M.J.; Barcelona, M.A.** "A model based solution for Business Process Modeling in Software Development Organizations". IST (2014)
6. **Kaiser, G. et al.** "Preliminary Experience with Process Modeling in the Marvel SDE Kernel," in Proceedings of IEEE 23th Hawaii ICSS Software Track. (1990)
7. **Conradi, R. et al.** "Design, use and implementation of SPELL, a language for software process modeling and evolution," SPT pp. 167–177 (1992)
8. **Sutton, S.M. et al.** "APPL/A: a language for software process programming," ACM TSEM vol. 4, no. 3, pp. 221-286 (1995)
9. **OMG.** "BPMN, Business Process Modeling Notation, Version 2.0". http://www.omg.org/spec/BPMN/2.0/. (2011). Last acc. 07/2014
10. **OMG.** "Architecture-driven modernization (ADM)". http://adm.omg.org (2010). Last acc. 07/2014
11. **OMG.** "Modern Driven Architecture (MDA)", http://www.omg.org/mda/ (2011). Last acc. 07/2014
12. **OMG.** "Meta Object Facility (MOF)", http://www.omg.org/spec/MOF/2.4.1 (2011). Last acc. 07/2014
13. **OMG.** "Information management metamodel (IMM)", http://www.omgwiki.org/imm/doku.php (2006). Last acc. 07/2014
14. **Pérez-Castillo, R.; Piattini, M. et al.** "MARBLE. A business process archeology tool". ICSM 2011:578-581, (2011)
15. **Pérez-Castillo R.; Piattini, M. et al.** "Knowledge discovery metamodel-ISO/IEC 19506: A standard to modernize legacy systems". Computer Standards & Interfaces, 33, 519–532. doi:10.1016/j.csi.2011.02.007 (2011)
16. **Pérez-Castillo R.; Piattini, M. et al.** "A family of case studies on business process mining using MARBLE". JSS 85(6):1370-1385 (2012)
17. **Pérez-Castillo R.; Piattini, M. et al**. "Database schema elicitation to modernize relational databases". ICEIS (2012)
18. **Pérez-Castillo R.; Piattini, M. et al.** "Software modernization by recovering web services from legacy databases". J. Softw. Maint. Evol.: Res. Pract. (2012)
19. **Cánovas Izquierdo, J.L. & García Molina, J.** "A domain specific language for extracting models in software modernization". In Proceedings of ECMDA-FA (pp. 82-97). ECMDA-FA (2009)
20. **Cánovas Izquierdo, J.L. & García Molina, J.** "An architecture-driven modernization tool for calculating metrics". IEEE Software Journal, 27(4), 37–43. doi:10.1109/MS.2010.61 (2010)
21. **Cánovas Izquierdo, J.L. & García Molina, J.** "Extracting models from source code in software modernization". Software & Systems Modeling. doi:10.1007/s10270-012-0270-z (2012)
22. **Cánovas Izquierdo, J.L. & García Molina, J.** "Grammar to model language (Gra2Mol)". http://code.google.com/a/eclipselabs.org/p/gra2mol . Last acc. 07/2014
23. **OMG.** "QVT, Meta Object Facility (MOF) 2.0 Query/View/Transformation (QVT)", http://www.omg.org/spec/QVT/1.1/ (2011). Last acc. 07/2014
24. **Eisenberg, A.** "New standard for stored procedures in SQL". ACM SIGMOD Record 25 (4): 81–88. doi:10.1145/245882.245907 (1996)
25. **Schmidt, D.C.** "Model-Driven Engineering," Computer, vol.39, no.2, pp.25-31 (2006)
26. **Alhir, S.S.** "Guide to applying the UML". Springer. ISBN 0-387-95209-8 (2002).
27. **Elveszeter, B. & Berre, A.J.** "OMG Specifications for Interoperability". In Interoperability for Enterprise Software and Applications: Proceedings of the Workshops and the Doctorial Symposium of the I-ESA International Conference 2010 (p. 31). John Wiley & Sons (2010)
28. **Atlanmod.** "Atlanmod Zoos", http://www.emn.fr/z-info/atlanmod/index.php/Zoos. Last acc. 07/2014
29. **Bandinelli, S.C. et al.** "Software process model evolution in the SPADE environment". IEEE TSE vol. 19, no. 12, pp. 1128-1144 (1993)

30. **OMG.** UML, Unified Modeling Language (UML 2.0)". http://www.omg.org/spec/UML/ (2011). Last acc. 07/2014
31. **Demuth, B. & Hußmann, H.** "Using UML/OCL constraints for relational database design". UML. doi:10.1007/3-540-46852-8_42 (1999)
32. **Demuth, B.; Hußmann, H.; Loecher,S.** "OCL as a specification language for business rules in database applications". UML. doi:10.1007/3-540-45441-1_9 (2001)
33. **Escalona, M.J. & Aragón, G.** "NDT. A Model-Driven Approach for Web Requirements," IEEE Transactions on Software Engineering, vol. 34, no. 3, pp. 377-390 (2008)
34. **Melton, J. & Simon, A.R.** "SQL:1999". Morgan Kaufmann. pp. 541–542. ISBN 978-1-55860-456-8 (2002)
35. **Iwt2.** "NDT-Suite", http://www.iwt2.org (2011). Last acc. 07/2014
36. **OMG.** "Object Constraint Language (OCL), Version 2.4". http://www.omg.org/spec/OCL/2.4/ (2014). Last acc. 07/2014
37. **Arévalo, C.; Gómez-López, M. T.; Reina Quintero, A.M.; Ramos, I.** "An Architecture to Infer Business Rules from Event Condition Action Rules Implemented in the Persistence Layer". In R. Pérez-Castillo, & M. Piattini (Eds.) Uncovering Essential Software Artifacts through Business Process Archeology (pp. 201-221). Hershey, PA: Business Science Reference. doi:10.4018/978-1-4666-4667-4.ch008 (2014).
38. **OMG.** "Common Warehouse Metamodel (CWM)". http://www.omg.org/spec/CWM/1.1. (2003). Last acc. 07/2014.
39. **Cutilla, C. R.; García-García, J. A.; Alba, M.; Escalona, M.J.; Ponce, J.; Rodríguez, L.** "Aplicación del paradigma MDE para la generación de pruebas funcionales; Experiencia dentro del proyecto AQUA-WS". Presented at the 6ª Conferencia Ibérica de Sistemas e Tecnologias de Informação, ISBN: 978-989-96247-4-0. (2011)
40. **Escalona, M.J.; García-García, J. A.; Más, F.; Oliva, M.; Del Valle, C.** "Applying model-driven paradigm: CALIPSOneo experience". CAiSE'13, vol. 1017, pp. 25-32. (2013).
41. **Salido, A.; García-García, J.A.; Gutiérrez, J.; Ponce, J.** "Tests Management in CALIPSOneo: A MDE Solution". Journal of Software Engineering and Applications, (2014).
42. **Van-der-Aalst, W.M.P.** "Business process management demystified: a tutorial on models, systems and standards for workflow management". Lecture Notes in Computer Science, Lectures on Concurrency and Petri Nets, vol. 3098, pp. 1-65. (2004).
43. **Van-der-Aalst, W.M.P.** "Business process management: a personal view". Business Process Management Journal, vol. 10, no. 2, p. 5. (2004).
44. **Bisbal, J.; Lawless, D; Wu, B.; Grimson, J.**: "Legacy Information Systems: Issues and Directions". 0740-7459/99 © 1999 IEEE
45. **Heuvel, W.-J.v.d.:** "Aligning Modern Business Processes and Legacy Systems: A Component-Based Perspective". (Cooperative Information Systems). The MIT Press. (2006).
46. **Paradauskas, B.; Laurikaitis, A.:** "Business Knowledge extraction from Legacy Information Systems". Journal of Information Technology and Control 35(3) 214–221,(2006)
47. **IWT2.** "Comparativa herramientas de modelado". Consejería de Cultura de la Junta de Andalucía. Proyecto Calidad. (2008). http://www.iwt2.org. Last acc. 07/2014
48. **SparxSystems**. Enterprise Architect. http://www.sparxsystems.com.au, Last acc. 07/2014.