CARRERA DEL INVESTIGADOR CIENTÍFICO Y TECNOLÓGICO

Informe Científico[1]

PERIODO [2]: 01/01/2014 al 31/12/2015.

1. **DATOS PERSONALES**

APELLIDO: MARCOS

NOMBRES: CLAUDIA ANDREA

Dirección Particular: Calle: Nº:

Localidad: TANDIL CP: 7000 Tel:

Dirección electrónica (donde desea recibir información, que no sea "Hotmail"): CMARCOS@EXA.UNICEN.EDU.AR

2. **TEMA DE INVESTIGACION**

Estrategias para Mejorar el Mantenimiento de Sistemas

3. **DATOS RELATIVOS A INGRESO Y PROMOCIONES EN LA CARRERA**

INGRESO: Categoría: ADJUNTO SIN DIRECTOR Fecha: 10/05/2012

ACTUAL: Categoría: ADJUNTO SIN DIRECTOR desde fecha: 10/05/2012

4. **INSTITUCION DONDE DESARROLLA LA TAREA**

Universidad y/o Centro: UNIVERSIDAD NACIONAL DEL CENTRO DE LA PROVINCIA DE BUENOS AIRES

Facultad: CIENCIAS EXACTAS

Departamento: ISISTAN – INSTITUTO DE SISTEMAS TANDIL

Cátedra:

Otros:

Dirección: Calle: PARAJE ARROYO SECO – CAMPUS UNIVERSITARIO Nº:

Localidad: TANDIL CP: 7000 Tel:

Cargo que ocupa: Profesor Asociado

5. **DIRECTOR DE TRABAJOS.** (En el caso que corresponda)

Apellido y Nombres:

Dirección Particular: Calle: Nº:

Localidad: CP: Tel:

Dirección electrónica:

---

1 Art. 11; Inc. "e"; Ley 9688 (Carrera del Investigador Científico y Tecnológico).

2 El informe deberá referenciar a años calendarios completos. Ej.: en el año 2014 deberá informar sobre la actividad del período 1°-01-2012 al 31-12-2013, para las presentaciones bianuales.

....

Firma del Director (si corresponde) Firma del Investigador

6. RESUMEN DE LA LABOR QUE DESARROLLA

Descripción para el repositorio institucional. Máximo 150 palabras.

El mantenimento de software es considerado una de las etapas más costosa del ciclo de vida de un sistema. Una vez que el sistema ha sido entregado, nuevos requerimientos y cambios en la funcionalidad es requerido. Estos cambios producen una degradación del código el cual comienza a tener problemas estructurales conocidos como *code smells*. Se ha desarrollado la herramienta JSpIRIT que asiste en la identificación de code smells analizando la historia de las clases de una aplicación, escenarios de modificabilidad definidos por el desarrollador, cómo se concentran los problemas en un conjunto de clases y la relevancia de cada code smell en dicha aplicación. Teniendo en cuenta dichos criterios la herramienta genera un ranking indentificando los problemas más importantes que deberían ser solucionados. Adicionalmente, se proveen diferentes alternativas de solución para el code smell Brain Method con un informe de los beneficios y problemas de cada una de esas alternativas.

7. EXPOSICION SINTETICA DE LA LABOR DESARROLLADA EN EL PERIODO.

Debe exponerse, en no más de una página, la orientación impuesta a los trabajos, técnicas y métodos empleados, principales resultados obtenidos y dificultades encontradas en el plano científico y material. Si corresponde, explicite la importancia de sus trabajos con relación a los intereses de la Provincia.

Durante el período anterior se desarrolló la herramienta SpIRIT (Identificación Inteligente de Oportunidades de Refactorización), la cual prioriza los code smells más críticos para un sistema. Dado un sistema orientado a objetos con code smells, JSpIRIT ayuda al desarrollador priorizando los code smells. La identificación de los code smells se basa en el catálogo existente de Marinescu. El aspecto novedoso de este enfoque es la priorización de code smells basado en la evaluación de sus relaciones con cuestiones de modificabilidad. La evaluación de una instancia de code smell es determinada por tres criterios: *historial de cambio,* el cual considera que cuanto más seguido los componentes se han modificado en las versiones anteriores del sistema, más inestables estos componentes son ante nuevos cambios. El segundo criterio es el *impacto de escenarios de modificabilidad,* si los componentes de un code smell también están afectados por uno o más escenarios, se tendrá un efecto en cascada de esos componentes. Es decir, si un smell está en un área de código relacionado a uno o varios escenarios importantes de modificabilidad, el desarrollador debe prestar mucha atención a la resolución de dicho smell, con el fin de mejorar la satisfacción de los escenarios. El tercer criterio es la *relevancia del code smell,* como no todos los tipos de code smells son igualmente problemáticos, la importancia que el desarrollador da a cada tipo de smell debe ser tenida en cuenta. JSpIRIT fue extendida para proponer estrategias de refactorización del code

smell Brain Method. Esta estrategia, denominada Bandago, toma como entrada el código con problemas y el tipo code smell y genera diferentes soluciones de refactorización para dicho code smell. Las soluciones propuestas pueden ser comparadas a partir de un conjunto de métricas permitiéndole al desarrollador seleccionar entre las soluciones generadas la que más se adapte a sus necesidades.

Con respecto a la modularización del sistema en las diferentes etapas del ciclo de vida, en el período anterior se identificaron los concerns relevantes en los casos de uso especificados en la etapa de captura de requerimientos. Para ello, se construyó una herramienta denominada REAssistant (REquirements Analysis Assistant) la cual ha sido implementada utilizando UIMA ya que provee componentes para realizar un análisis de las oraciones de los casos de uso. Para ello, se realiza un análisis del lenguaje natural (NLP) incorporandole información semántica que permita mejorar la identificación de los concerns no funcionales más importantes. Como entrada el analista provee los casos de uso, a los cuales se les realiza una serie de análisis, por ejemplo eliminando los stops words, llevando los verbos a su raíz por medio del stemming e identificando información importante que es representada por medio de los anotadores de UIMA. Luego, se ejecutan una serie de queries que por medio de los anotadores permiten descubrir los concerns no funcionales del sistema. Esta técnica ha sido extendida para identificar desiciones de diseño en documentos de la arquitectura (SADs). Para ello, se utilizan diversos módulos de NLP que extraen información léxica y sintáctica (palabras y partes del discurso) a partir del texto contenido en el SAD. Esta información es luego utilizada por un sistema basado en reglas para identificar oraciones que referencian (aunque parcialmente) decisiones de diseño que tienen impacto sobre uno o más atributos de calidad. Las reglas están agrupadas por tácticas y éstas a su vez por atributos de calidad. Por ejemplo, para identificar decisiones de diseño asociadas al atributo disponibilidad, se utilizan reglas que buscan tácticas como *Ping/Echo, Exception* o *Heartbeat* entre otras. Dicha técnica también utiliza UIMA para el procesamiento del texto y el motor de reglas de RUTA.

De esta manera se proponen diferentes alternativas que permitan mejorar el mantenimiento y evolución de los sistemas.

8. TRABAJOS DE INVESTIGACION REALIZADOS O PUBLICADOS EN ESTE PERIODO.

***8.1 PUBLICACIONES**. Debe hacer referencia exclusivamente a aquellas publicaciones en las que haya hecho explícita mención de su calidad de Investigador de la CIC (Ver instructivo para la publicación de trabajos, comunicaciones, tesis, etc.). Toda publicación donde no figure dicha mención no debe ser adjuntada porque no será tomada en consideración. A cada publicación, asignarle un número e indicar el nombre de los autores en el mismo orden que figuran en ella, lugar donde fue publicada, volumen, página y año. A continuación, transcribir el resumen (abstract) tal como aparece en la publicación. La copia en papel de cada publicación se presentará por separado. Para cada publicación, el investigador deberá, además, aclarar el tipo o grado de participación que le cupo en el desarrollo del trabajo y, para aquellas en las que considere que ha hecho una contribución de importancia, deberá escribir una breve justificación.*

Artículos en Revistas

1. ***Improving Use Case Specifications by means of Refactoring*. Claudia Marcos, Alejandro Rago and J. A. Díaz Pace. IEEE Latin America Transactions. Volume 13, Issue 4, April 2015. Pages 1135-1140. Publisher: IEEE Region 9. ISSN: 1548-0992. DOI: 10.1109/TLA.2015.7106367.**

   Abstract— This work presents a semi-automatic tool for use case refactoring called RE-USE. This tool discovers existing quality problems in use cases and suggests a

prioritized set of candidate refactorings to functional analysts. The analyst then reviews the recommendation list and selects the most important refactoring. The tool applies the chosen refactoring and returns an improved specification. The tool effectiveness in detecting existing quality problems and recommending proper refactorings was assessed using a set of case studies related to real-world systems obtaining encouraging results.

Grado de participación: Alto, ya que estuvo directamente involucrada en definir las estrategias de refactoring de casos de uso y particularmente en la escritura del paper. El trabajo es una parte del resultado de investigación de la tesis de doctorado de Alejandro Rago donde Claudia Marcos fue la directora.

2. ***Understanding and addressing exhibitionism in Java empirical research about method accessibility*. Santiago A. Vidal, Alexandre Bergel, Claudia Marcos & J. Andrés Díaz-Pace. Empirical Software Engineering An International Journal Vol 21 Nro 2 p 483-516. ISSN 1382-3256 DOI 10.1007/s10664-015-9365-9. Springer On line version Febrero 2015.**

Abstract. Information hiding is a positive consequence of properly defining component interfaces. Unfortunately, determining what should constitute a public interface remains difficult. We have analyzed over 3.6 million lines of Java open-source code and found that on the average, at least 20 % of defined methods are over-exposed, thus threatening public interfaces to unnecessary exposure. Such over-exposed methods may have their accessibility reduced to exactly reflect the method usage. We have identified three patterns in the source code to identify over-exposed methods. We also propose an Eclipse plugin to guide practitioners in identifying over-exposed methods and refactoring their applications. Our plugin has been successfully used to refactor a non-trivial application.

Grado de participación: Alto, este trabajo es parte de los resultados obtenidos gracias a la colaboración internacional entre Chile y Argentina, especificamente con Alexandre Bergel de la Universidad de Chile.

3. ***An Approach to Prioritize Code Smells for Refactoring* Santiago Vidal, Claudia Marcos, Andrés Díaz Pace Automated Software Engineering An International Journal. ISSN 0928-8910 DOI 10.1007/s10515-014-0175-x Vol 21 December 2014 Springer.**

Abstract. Code smells are a popular mechanism to find structural design problems in software systems. Consequently, several tools have emerged to support the detection of code smells. However, the number of smells returned by current tools usually exceeds the amount of problems that the developer can deal with, particularly when the effort available for performing refactorings is limited. Moreover, not all the code smells are equally relevant to the goals of the system or its health. This article presents a semi-automated approach that helps developers focus on the most critical problems of the system. We have developed a tool that suggests a ranking of code smells, based on a combination of three criteria, namely: past component modifications, important modifiability scenarios for the system, and relevance of the kind of smell. These criteria are complementary and enable our approach to assess the smells from different perspectives. Our approach has been evaluated in two case-studies, and the results show that the suggested code smells are useful to developers.

Grado de participación: Alto, ya que estuvo directamente involucrada en el desarrollo de la estrategia para priorizar code smells y en la escritura del paper. El trabajo es una parte del resultado de investigación de la tesis de doctorado de Santiago Vidal donde Claudia Marcos fue la directora.

4. ***Identifying duplicate functionality in textual use cases by aligning semantic actions.*** **Alejandro Rago, Claudia Marcos, Andrés Díaz Pace. Software & Systems Modeling. [JCR® 2014 I.F. 1.408] | Publisher: Springer Verlag ISSN 1619-1366 . Softw Syst Model. 1619-1374 (Online Version). DOI 10.1007/s10270-014-0431-3. August 2014.**

Abstract Developing high-quality requirements specifications often demands a thoughtful analysis and an adequate level of expertise from analysts. Although requirements modeling techniques provide mechanisms for abstraction and clarity, fostering the reuse of shared functionality (e.g., via UML relationships for use cases), they are seldom employed in practice. A particular quality problem of textual requirements, such as use cases, is that of having duplicate pieces of functionality scattered across the specifications. Duplicate functionality can sometimes improve readability for end users, but hinders development-related tasks such as effort estimation, feature prioritization, and maintenance, among others. Unfortunately, inspecting textual requirements by hand in order to deal with redundant functionality can be an arduous, time-consuming, and error-prone activity for analysts. In this context, we introduce a novel approach called ReqAligner that aids analysts to spot signs of duplication in use cases in an automated fashion. To do so, ReqAligner combines several text processing techniques, such as a use case aware classifier and a customized algorithm for sequence alignment. Essentially, the classifier converts the use cases into an abstract representation that consists of sequences of semantic actions, and then these sequences are compared pairwise in order to identify action matches, which become possible duplications. We have applied our technique to five real-world specifications, achieving promising results and identifying many sources of duplication in the use cases.

Grado de participación: Alto, ya que estuvo directamente involucrada en la estrategia para identificar funcionalidad duplicada en los casos de uso y en la escritura del paper. El trabajo es una parte del resultado de investigación de la tesis de doctorado de Alejandro Rago donde Claudia Marcos fue la directora.

5. ***An Approach for Automating Use Case Refactoring*****. Alejandro Rago, Paula Frade, Miguel Ruival y Claudia Marcos. SADIO Electronic Journal of Informaticas and Operations Research. Vol 13 - No. 1 - September 2014. Special Issue dedicated to ASSE 2013 (Argentine Symposium on Software Engineering) Invited Editor J. Andrés Díaz Pace and Pedro E. Colla. ISSN: 1514-6774**

Abstract. Carrying out requirements capture and modeling activities successfully is not easy, often requiring a thoughtful analysis of clients' needs and demanding an adequate expertise from analysts. To ensure a fluid communication among stakeholders, analysts must take advantage of modeling techniques while describing requirements and exploit reuse and abstraction practices so as to avoid redundancy (for instance, using relations between use cases). Unfortunately, these practices are seldom applied because inspecting requirements such as textual use cases by hand,

looking out for faulty or duplicate functionalities, is a challenging and error-prone activity. In this context, we introduce an assistive approach called ReUse that searches redundancy deficiencies in use case specifications and allows to fix them with relation-based refactorings. Our approach makes use of text processing and sequence alignment techniques to discover deficiencies (e.g., duplicate functionality). We have evaluated ReUse in five case studies, achieving promising results.

Grado de participación: Alto, ya que estuvo directamente involucrada en la estrategia para identificar funcionalidad duplicada en los casos de uso y en la escritura del paper. El trabajo es el resultado del trabajo final de la carrera de Ingeniería de Sistemas de Paula Frade y Miguel Ruival donde Alejandro Rago y Claudia Marcos fueron los directores.

Artículos en congresos

6. ***JSpIRIT: A Flexible Tool for the Analysis of Code Smells*. Santiago Vidal, Hernán Vázquez, J. Andrés Díaz-Pace, Claudia Marcos, Alessandro Garcia, Willian Oizumi 34th International Conference of the Chilean Computer Science Society SCCC 2015 Santiago 9-13 ISBN 978-1-4673-9816-9 pp 1-6. DOI 10.1109/SCCC.2015.7416572 Poster. Publisher IEEE Noviembre 2015.**
Abstract—Code smells are a popular mechanism to identify structural design problems in software systems. Since it is generally not feasible to fix all the smells arising in the code, they can be postponed by developers to be resolved in the future. One reason for this decision is that the improvement of the code structure, to achieve modifiability goals, needs extra efforts that developers might not always want to spend, particularly when they are focused on delivering customer-visible features. Along this line, the smells are seen as a source of technical debt. Furthermore, not all the code smells may be urgent to fix in the context of the evolution of the system or its business goals. In this article, we present a flexible tool to prioritize technical debt related to code smells. The tool easily allows developers to add new smells and to prioritize them based on the configuration of multiple criteria.

   Grado de participación: Alto, ya que el trabajo es una extensión a JSpIRIT para analizar aglomeraciones. Dicho trabajo es parte de los resultados del proyecto de cooperación entre Brasil y Argentina al cual pertenezco.

7. ***Identifying Duplicate Functionality in Textual Use Cases by Aligning Semantic Actions* (SoSyM abstract) Alejandro Rago, Claudia Marcos, Andrés Díaz-Pace. In Proceedings of the ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS'15) | Ottawa, Ontario, Canada, September 2015 | Pages 442 | Publisher: ACM/IEEE | ISBN: 978-1-4673-6908-4/15 | DOI: 10.1109/MODELS.2015.7338276 Best Paper Award.**
Abstract—Developing high-quality requirements specifications often demands a thoughtful analysis and an adequate level of expertise from analysts. Although requirements modeling techniques provide mechanisms for abstraction and clarity, fostering the reuse of shared functionality (e.g., via UML relationships for use cases), they are seldom employed in practice. A particular quality problem of textual requirements, such as use cases, is that of having duplicate pieces of functionality scattered across the specifications. Duplicate functionality can sometimes improve readability for end users, but hinders development-related tasks such as effort

estimation, feature prioritization and maintenance, among others. Unfortunately, inspecting textual requirements by hand in order to deal with redundant functionality can be an arduous, time-consuming and error-prone activity for analysts. In this context, we introduce a novel approach called ReqAligner that aids analysts to spot signs of duplication in use cases in an automated fashion. To do so, ReqAligner combines several text processing techniques, such as a use-case-aware classifier and a customized algorithm for sequence alignment. Essentially, the classifier converts the use cases into an abstract representation that consists of sequences of semantic actions, and then these sequences are compared pairwise in order to identify action matches, which become possible duplications. We have applied our technique to five real-world specifications, achieving promising results and identifying many sources of duplication in the use cases.

Grado de participación: Alto, ya que estuvo directamente involucrada en la estrategia de alineación semántica y en la escritura del paper. El trabajo es una parte del resultado de investigación de la tesis de doctorado de Alejandro Rago donde Claudia Marcos fue la directora.

8. ***REAssistant: a Tool for Identifying Crosscutting Concerns in Textual Requirements*** **(Tool Demonstration) Alejandro Rago, Claudia Marcos, Andrés Díaz-Pace In Proceedings of the Demo and Poster Session at the ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS'15) Poster and Demos Section. | Volume 1554 - Ottawa, Ontario, Canada, September 2015 | Pages 32-35 | Publisher: ACM/IEEE | ISSN: 1613-0073.**

Abstract—Use case modeling is very useful to capture requirements and communicate with the stakeholders. Use cases normally have textual specifications that describe the interactions between the system and external actors. However, since use cases are specified from a functional perspective, concerns that do not fit well this decomposition criterion are kept away from the analysts' eye and might end up intermingled in multiple use cases. These crosscutting concerns (CCCs) are generally relevant for analysis, design and implementation activities and should be dealt with from early stages. Unfortunately, identifying such concerns by hand is a cumbersome and error-prone task, mainly because it requires a semantic interpretation of textual requirements. To ease the analysis of CCCs, we have developed an automated tool called REAssistant that is able to extract semantic information from textual use cases and reveal candidate CCCs, helping analysts to reason about them before making important commitments in the development. Our tool performs a series of advanced NLP analyses based on the UIMA framework. Analysts can define concern-specific queries in the tool to search for CCCs in the requirements via a flexible SQL-like language. In this article, we briefly discuss the technologies behind the tool and explain how an end user can interact with REAssistant to analyze CCCs in use case specifications. A short video explaining the main features of the tool can be found at https://youtu.be/i3kSJil_2eg. The REAssistant tool can be downloaded from "https://code.google.com/p/reassistant"https://code.google.com/p/reassistant.

Grado de participación: Alto, ya que estuvo involucrada en la definición de las características de la tool y en la escritura del paper. El trabajo es una parte del

resultado de investigación de la tesis de doctorado de Alejandro Rago donde Claudia Marcos fue la directora.

9. ***Uso de Ontologías para mapear una Arquitectura de Software con su Implementación.*** **H. C. Vázquez , J. A. Díaz Pace, and C. Marcos. Proceedings of Simposio Argentino de Ontologías y sus Aplicaciones (SAOA) ISSN: 1850-2792 En el marco de las JAIIO Jornadas Argentinas de Informática e Investigación Operativa. Setiembre 2015 Rosario, Santa Fe.**

Resumen La arquitectura de software de un sistema es un activo importante para una organización que desarrolla software. Para maximizar los beneficios que provee una arquitectura, ésta debe estar en correspondencia con la implementación del sistema. En muchos proyectos existe cierta documentación de la arquitectura, pero sin embargo, la información de mapeos entre los elementos de dicha arquitectura y su implementación en código es escasa o inexistente. Este problema trae aparejadas dificultades de entendimiento de los elementos de código en relación a la arquitectura originalmente diseñada, lo que repercute negativamente sobre el aseguramiento de la calidad y los esfuerzos de mantenimiento del sistema. Si bien la provisión manual de estos mapeos es factible, es una tarea compleja y proclive a errores, particularmente a medida que la implementación del sistema evoluciona en el tiempo. En este contexto, las técnicas de alineación de ontologías se presentan como una alternativa para producir mapeos en forma automática. Por esta razón, el presente trabajo propone un enfoque automatizado y basado en ontologías para la generación de mapeos entre la arquitectura de un sistema y su implementación.

Grado de participación: Alto, ya que estuvo directamente involucrada la definición de la ontología y en la escritura del paper. El trabajo son los primeros resultados de la investigación de Hernán Ceferino Vazquez donde Claudia Marcos es co-directora.

10. ***Conformidad Estructural de Arquitecturas combinado con Análisis de Impacto de Cambios*****. Miguel Armentano, Luis Soldavini, J. Andrés Díaz Pace, Santiago Vidal, and Claudia Marcos. Proceedings of ASSE 2015 Argentine Symposium on Software Engineering. JAIIO - Jornadas Argentinas de Informática e Investigación Operativa. ISSN: 1850-2792 En el marco de las JAIIO Jornadas Argentinas de Informática e Investigación Operativa. Setiembre 2015 Rosario, Santa Fe.**

Abstract. La conformidad de arquitecturas de software es una práctica que permite mantener la estructura arquitectónica alineada y consistente con su implementación en código. Entre otros beneficios, este alineamiento permite a los arquitectos y desarrolladores realizar distintos análisis de la solución desde etapas tempranas (por ej., de performance, o de modificabilidad, entre otros). Para esto, deben verificarse periódicamente las relaciones entre los elementos arquitectónicos y sus contrapartes en el código fuente, a fin de detectar posibles violaciones de la arquitectura. Las técnicas y herramientas existentes para conformidad arquitectónica proveen un buen soporte para verificar relaciones de tipo estructural. Sin embargo, ciertos análisis que son útiles a nivel arqui-ectónico, como es el caso del impacto de cambios (CIA, Change Impact Analysis), son difíciles de verificar en relación al código fuente, lo cúal genera una brecha entre las suposiciones y conclusiones que se hacen a nivel arquitectónica y la implementación actual del sistema. Este trabajo presenta un enfoque que extiende las reglas básicas de conformidad estructural con reglas que

permiten contemplar suposiciones de CIA con el fin de validarlas en la implementacióon de la arquitectura. En particular, se propone una herramienta que integra un CIA en base a escenarios de modificabilidad con el modelo de reflexión de Murphy & Notkin. Los resultados, si bien son preliminares, indican que este enfoque permite identificar distintos tipos de violaciones arquitectónicas.

Grado de participación: Medio, ya que estuvo involucrada en el proceso análisis del enfoque de conformidad y en la escritura del paper.

11. ***Improving Requirements with NLP Techniques.*** **Alejandro Rago - Advisors: Claudia Marcos & J. Andrés Díaz-Pace 2nd. IJCAI School on Artifitial Inteligence. En el marco de las JAIIO Jornadas Argentinas de Informática e Investigación Operativa. Pp 7-8 Publisher: SADIO | ISSN: 1850-2784 Setiembre 2014 Buenos Aires.**

Abstract. Elaborating "good" requirements specifications is an important factor for the success of a software project. Requirements are normally expressed using textual descriptions in natural language, but not without problems. Some requirements documentation techniques, such as use cases specifications, often focus on functionality and leave many concerns understated in the text and scattered through several documents. These concerns, commonly known as crosscutting or architecturally-relevant concerns, often come from business goals or quality attributes that must be clearly identified by analysts and developers, as these concerns can have a far-reaching effect in the development process. Not treating these concerns at early development stages can lead to poor design solutions that become difficult (and costly) to fix afterwards. Unfortunately, searching for concerns in textual requirements is a difficult and time-consuming task for analysts, because requirements are often poorly modularized and there is text duplicated across documents. Automated techniques able to interpret textual documents and extract information about concerns can be of great help for the analyst's work, allowing her/him to get a quick view of duplicated functionalities and potential concerns at the outset of development and thus make informed decisions. Several approaches to analyze textual requirements have relied on Natural Language Processing (NLP) techniques in order to inspect the text and extract "hidden" information. Some techniques have been proposed for analyzing the modularity of requirements, although only a few researchers have addressed duplicate functionality in textual specifications [1]. Likewise, there are several techniques for discovering architecturally-relevant concerns scattered in textual requirements [5]. Yet, such techniques do not work well in real-world specifications because of their poor semantic support (in the context of software development). The main objective of our work is to assist analysts for improving the quality of requirements documentation and fostering the inspection of architecturally-significant concerns. The hypothesis is that by leveraging on NLP techniques and domain knowledge (about use cases) is possible to develop techniques that reveal duplicate functionality and concern-related information. The proposed approach makes use of NLP modules to process textual requirements, labeling meta-information such as time tenses, syntactical functions, and morphological lemmas. To improve the modularity of use cases, we feed the information out-putted by NLP-based modules into a sequence alignment algorithm for exposing duplicate requirements and suggesting candidate refactorings for the use cases. Sequence alignment originated in bioinformatics for finding duplicated chains

in DNA strings. To reveal information about latent architectural concerns, we complement the analysis of textual use cases with semantic NLP modules (e.g,dependency parsing, semantic role labeling) and specific concepts derived from use cases, called domain actions. Then, analysts can query this information via concern-specific rules for identifying a given concern. We have started to build tools to test the techniques, emphasizing on the analysts' user experience. Technologically speaking, the tools are built on top of the Unstructured Information Management Architecture (UIMA) and Rule-based Text Annotation (RUTA), two frameworks that enable the configuration of state-of-the-art NLP modules in different arrangements. We currently have publications reporting on one particular technique for discovering architectural concerns [2,3] and have submitted another one about the technique for improving the modularity of use cases [4]. We have made considerable progress so far in the development of the approach. We built a core engine for executing UIMA modules in an Eclipse environment and investigated a number of NLP packages publicly available (e.g., OpenNLP and MateTools). These packages were wrapped as UIMA annotators and arranged into a customizable NLP pipeline. In addition, we developed UIMA modules able to recognize requirements-specific concepts using machine learning algorithms. We also adapted a sequence alignment algorithm to work on textual use cases, making the necessary adjustments to perform well in this domain, and implemented a rule-based module on top of RUTA that allows analysts to run queries to reveal architectural concerns. We have performed preliminary experiments of the techniques in several case-studies, obtaining promising results. Nonetheless, we plan to test the techniques with more case-studies to further corroborate our findings. Particularly, we are interested in analyzing the influence of discovering latent architectural concerns on poorly-modularized requirements versus discovering them on well- modularized requirements. Some of the next steps in our research are: (i) investigate querying specialized concerns, such as quality-attribute concerns or business goals, directly from stakeholders workshops or business/mission statements; (ii) if the concern identification is performed on multiple text documents, explore the traceability links between the concerns extracted from each document; and (iii) analyze the performance of the techniques when used in combination with the criteria of human analysts.

Grado de participación: Alto, ya que estuvo involucrada en el análisis de las necesidades de mejoras de las técnicas de NLP y en la escritura del paper. El trabajo es una parte del resultado de investigación de la tesis de doctorado de Alejandro Rago donde Claudia Marcos fue la directora.

12. ***Analyzing the History of Software Systems to Predict Class Changes*. Santiago A. Vidal · Claudia Marcos · J. Andrés Díaz-Pace. Argencon 2014. Junio 2014. San Carlos de Bariloche. ISBN 97B-1-4799-4270-1.**

Abstract—Determining the critical parts of a system is key to effectively conduct preventive software maintenance. To accomplish this task, information from the system history (i.e., past versions) can be helpful for identifying those software elements that are more likely to receive modifications in the near future. However, interpreting the large amount of data usually present in the history can be difficult. In this work, we propose an approach adapted from financial markets for analyzing the history of an object-oriented system and predicting the classes that might change. We have evaluated our approach by comparing it with existing approaches. The results,

although preliminary, show that our approach makes more accurate predictions for class changes.

Grado de participación: Alto, ya que estuvo en el análisis de cómo la historia de un sistema podría llegar a ayudar en la identificación de problemas y en la escritura del paper. El trabajo es una parte del resultado de investigación de la tesis de doctorado de Santiago Vidal donde Claudia Marcos fue la directora.

13. ***Una Comparación de Técnicas de NLP Semánticas para Analizar Casos de Uso.*** **Alejandro Rago, Claudia Marcos, Andrés Díaz-Pace. In Proceedings of the 2nd IEEE Biennial Congress of Argentina (ARGENCON'14) | San Carlos de Bariloche, Argentina, June 2014 | Pages 479-484 | Publisher: IEEE Argentina | ISSN: 1850-0870 | ISBN: 978-1-4799-4270-1 | DOI: 10.1109/ARGENCON.2014.6868539**

Resumen—La inspección computacional de documentos escritos en lenguaje natural es una tarea viable gracias a los avances en técnicas de Procesamiento de Lenguaje Natural (NLP). Sin embargo, ciertas aplicaciones requieren un análisis semántico más profundo para obtener buenos resultados. En este artículo, se presenta un estudio exploratorio de técnicas de NLP semánticas para descubrir concerns ocultos en especificaciones de casos de uso. Con este motivo, se proponen dos técnicas de NLP: clustering semántico y reglas enriquecidas semánticamente. Los resultados de la evaluación experimental de estas dos técnicas, y su comparación contra una tercera técnica desarrollada por otros investigadores, muestran que las técnicas de NLP semánticas tienen un gran potencial para detectar concerns candidatos. Particularmente, si estas técnicas son configuradas correctamente, pueden ayudar a reducir el esfuerzo de los analistas de requerimientos y promover la construcción de software de mejor calidad.

Grado de participación: Alto, ya que estuvo involucrada en la comparación de las técnicas de NLP y en la escritura del paper. El trabajo es una parte del resultado de investigación de la tesis de doctorado de Alejandro Rago donde Claudia Marcos fue la directora.

8.2 TRABAJOS EN PRENSA Y/O ACEPTADOS PARA SU PUBLICACIÓN*. Debe hacer referencia exclusivamente a aquellos trabajos en los que haya hecho explícita mención de su calidad de Investigador de la CIC (Ver instructivo para la publicación de trabajos, comunicaciones, tesis, etc.). Todo trabajo donde no figure dicha mención no debe ser adjuntado porque no será tomado en consideración. A cada trabajo, asignarle un número e indicar el nombre de los autores en el mismo orden en que figurarán en la publicación y el lugar donde será publicado. A continuación, transcribir el resumen (abstract) tal como aparecerá en la publicación. La versión completa de cada trabajo se presentará en papel, por separado, juntamente con la constancia de aceptación. En cada trabajo, el investigador deberá aclarar el tipo o grado de participación que le cupo en el desarrollo del mismo y, para aquellos en los que considere que ha hecho una contribución de importancia, deber á escribir una breve justificación.*

14. ***Opportunities for Analyzing Hardware Specifications with NLP Techniques*** **Alejandro Rago, Claudia Marcos, J. Andrés Diaz-Pace. DUHDe 2016 — 3rd**

Workshop on Design Automation for Understanding Hardware Designs. March 18, 2016 — Friday Workshop at DATE 2016, Dresden, Germany.

Abstract—Hardware design is a mature discipline that heavily relies on complex models to sketch the blueprints of a system and special notations to describe the expected behavior of its components. However, hardware engineers frequently have to go through multiple specifications written in natural languageAbstract—Hardware design is a mature discipline that heavily relies on complex models to sketch the blueprints of a system and special notations to describe the expected behavior of its components. However, hardware engineers frequently have to go through multiple specifications written in natural language to identify components, constraints and assertions and translate them to more formal expressions in order to make automated verifications and consistency checks possible. For this reason, computer-assisted tools capable of processing and understanding hardware documentation can be of great help to assist and guide engineers in difficult and otherwise error-prone activities. In previous works, we have explored several Natural Language Processing (NLP) techniques for the analysis of requirements and architecture specifications with promising results. In this article, we report on some interesting tools we developed for inspecting Software Engineering documentation by leveraging in advanced NLP techniques and discuss their potential applications to automated hardware design. to identify components, constraints and assertions and translate them to more formal expressions in order to make automated verifications and consistency checks possible. For this reason, computer-assisted tools capable of processing and understanding hardware documentation can be of great help to assist and guide engineers in difficult and otherwise error-prone activities. In previous works, we have explored several Natural Language Processing (NLP) techniques for the analysis of requirements and architecture specifications with promising results. In this article, we report on some interesting tools we developed for inspecting Software Engineering documentation by leveraging in advanced NLP techniques and discuss their potential applications to automated hardware design.

Grado de participación: Alto, ya que estuvo involucrada en la identificación de técnicas de NLP para hardware y en la escritura del paper. El trabajo es una extensión de la tesis de doctorado de Alejandro Rago donde Claudia Marcos fue la directora.

15. ***Chasing Critical Code Anomalies with JSpIRIT*. Santiago Vidal, Andres Diaz Pace, Claudia Marcos Software Engineering Institute (SEI) Architecture Technology User Network Conference SATURN 2016. San Diego, CA, May 2-5, 2016**

As a software system evolves, its design structure often degrades and accumulates technical debt. The emergence of code smells, such as a God Class, is a well-known symptom of such problems. Although several tools exist for detecting code smells, the number of smells returned by current tools generally exceeds the number of problems developers can deal with. This is particularly evident when a team should focus on customer-visible features, and thus the time available for system restructuring is limited. Furthermore, not all smells require urgent attention, as they might not be related to architectural problems or business goals. In this context, having a tool that can prioritize critical smells is of great help for architects and developers.To this end, we developed JSpIRIT (Java Smart Identification of Refactoring opportunITies) as a recommender system for ranking code smells

according to multiple criteria. JSpIRIT performs a scanning of the system code, but its analysis is flexible enough to include information from past system versions, modifiability scenarios, and architectural components, among other assets. In the past few years, we have applied JSpIRIT to several Java projects with satisfactory results. Consequently, we have continued to improve the tool with more features. For instance, since smells often appear interrelated in the code, JSpIRIT provides insights to the developer about smell groupings. In addition, it offers visualizations for different smell configurations. We will present the key tool features and discuss project experiences in which JSpIRIT was useful for diagnosing the system "health" and planning for refactorings.

Grado de participación: Alto, ya que estuvo en el análisis de los problemas de código que puede tener un sistema y en la escritura del paper. El trabajo es una parte del resultado de investigación de la tesis de doctorado de Santiago Vidal donde Claudia Marcos fue la directora.

16. ***On the Criteria for Prioritizing Code Anomalies to Identify Architectural Problems.*** **Santiago Vidal, Everton Guimaraes, Willian Oizumi, Alessandro Garcia, Andrés Díaz Pace, Claudia Marcos The 31st ACM Symposium On Applied Computing. SAC 2016 Pisa, Italy April 4 - 8, 2016 to appear at the conference as a poster.**

Abstract - Architectural problems constantly affect evolving software projects. When not properly addressed, those problems can hinder the maintenance of a system. Some studies revealed that a wide range of architectural problems are reflected in source code through code anomalies. However, a software project often contains thousands of code anomalies and many of them have no relation to architectural problems. Unfortunately, state-of-the-art techniques fail short in assisting developers on the prioritization of architectural problems realized in the source code. As a consequence, developers struggle to effectively determine which (groups of) anomalies are architecturally relevant. This work proposes an innovative suite of criteria for prioritizing groups of code anomalies as indicators of architectural problems in evolving systems. These criteria are supported by a tool called JSpIRIT. We have assessed the prioritization criteria in the context of more than 20 versions of 3 systems, analyzing their effectiveness for detecting symptoms of architectural problems. The results provide evidence that the proposed criteria helped to correctly prioritize more than 80 architectural problems in our top-7 rankings, alleviating tedious manual inspections of the source code vis-a-vis with the architecture. Our prioritization criteria would help developers to discard at least 1000 code anomalies that has no relation to architectural problems in the systems analyzed.

Grado de participación: Alto, ya que estuvo en la definición de la estrategia de priorización de los problemas de código y en la escritura del paper. El trabajo es una parte del resultado de investigación del proyecto de cooperación entre Brasil y Argentina donde Claudia Marcos forma parte del equipo.

***8.3 TRABAJOS ENVIADOS Y AUN NO ACEPTADOS PARA SU PUBLICACION**. Incluir un resumen de no más de 200 palabras de cada trabajo, indicando el lugar al que han sido enviados. Adjuntar copia de los manuscritos.*

17. *Over-exposed Classes in Java: An Empirical Study*. S. Vidal, A. Bergel b , J. A. Díaz-Pace, C. Marcos. Computer Languages, Systems and Structures. An International Journal. Elsevier.

Estado: Envío de las correcciones de la primer revisión, con cambios menores.

Abstract. Java access modifiers regulate interactions among software components. In particular, class modifiers specify which classes from a component are publicly exposed and therefore belong to the component public interface. Restricting the accessibility as specified by a programmer is key to ensure a proper software modularity. It has been said that failing to do so is likely to produce maintenance problems, poor system quality, and architecture decay. However, how developers uses class access modifiers or how inadequate access modifiers affect software systems has not been investigated yet in the literature. In this work, we empirically analyze the use of class access modifiers across a collection of 15 Java libraries and 15 applications, totaling over 3.6M lines of code. We have found that an average of 25% of classes are over-exposed, i.e., classes defined with an accessibility that is broader than necessary. A number of code patterns involving over-exposed classes have been formalized, characterizing programmers' habits. Furthermore, we propose an Eclipse plugin to make component public interfaces match with the programmer's intent.

***8.4 TRABAJOS TERMINADOS Y AUN NO ENVIADOS PARA SU PUBLICACION**. Incluir un resumen de no más de 200 palabras de cada trabajo.*

***8.5 COMUNICACIONES**. Incluir únicamente un listado y acompañar copia en papel de cada una. (No consignar los trabajos anotados en los subtítulos anteriores).*

***8.6 INFORMES Y MEMORIAS TECNICAS**. Incluir un listado y acompañar copia en papel de cada uno o referencia de la labor y del lugar de consulta cuando corresponda.*

9. TRABAJOS DE DESARROLLO DE TECNOLOGÍAS.

***9.1 DESARROLLOS TECNOLÓGICOS.** Describir la naturaleza de la innovación o mejora alcanzada, si se trata de una innovación a nivel regional, nacional o internacional, con qué financiamiento se ha realizado, su utilización potencial o actual por parte de empresas u otras entidades, incidencia en el mercado y niveles de facturación del respectivo producto o servicio y toda otra información conducente a demostrar la relevancia de la tecnología desarrollada.*

***9.2 PATENTES O EQUIVALENTES**. Indicar los datos del registro, si han sido vendidos o licenciados los derechos y todo otro dato que permita evaluar su relevancia.*

***9.3 PROYECTOS POTENCIALMENTE TRASNFERIBLES, NO CONCLUIDOS Y QUE ESTAN EN DESARROLLO**. Describir objetivos perseguidos, breve reseña de la*

labor realizada y grado de avance. Detallar instituciones, empresas y/o organismos solicitantes.

Se continúa la relación y el trabajo con Temperies S.A., sin embargo la relación y las actividades no han sido tan estrechas como en el período anterior.

El proyecto con Q4TEch S.A. y Lider File S.A., tenía como objetivo identificar los problemas de código que tienen en sus sistemas y analizarlos para construir un orden de prioridad. Para realizar esta actividad se pensó en utilizar la herramienta JSpIRIT (Identificación Inteligente de Oportunidades de Refactorización) que prioriza los code smells más críticos para un sistema la cual fue construida como tesis de doctorado de Santiago Vidal. Dado un sistema orientado a objetos con code smells, JSpIRIT ayuda al desarrollador priorizando los code smells. La identificación de los code smells se basa en el catálogo de Marinescu. El aspecto novedoso de este enfoque es la priorización de code smells basado en la evaluación de sus relaciones con cuestiones de modificabilidad. La evaluación de una instancia de code smells es determinada por varios criterios: *historial de cambio*, que analiza cuán frecuente una clase ha sido modificada desde su creación; *impacto de escenarios de modificabilidad*, que analiza si los componentes de un code smell también están afectados por uno o más escenario; y *relevancia del code smell*, que analiza la relevancia de cada code smell según las necesidades del desarrollador. Se comenzó a analizar un sistema destinado a brindar soluciones para la gestión integral de la visita médica y ficheros médicos de los laboratorios farmacéuticos. Se identificaron los code smells del sistema y se los ordenó de acuerdo a los criterios establecidos por el cliente. Una vez definido el ranking se realizó una reunión donde se presentó una descripción detallada de la situación, obteniendo interesantes resultados. Sin embargo, el sistema no pudo ser refactorizado debido a cambios en las reglas del negocio del cliente y algunos aspectos específicos de la empresa. Igualmente, la relación con Q4TEch sigue en vigencia.

Se ha presentado a aprobado el PDTS – CIN (Consejo Interuniversitario Nacional): *Un método centrado en arquitecturas de software para líneas de producto con soporte de herramientas.* Código PDTS217 – CIN (Consejo Interuniversitario Nacional). Cuyo director es Andrés Díaz Pace y donde Claudia Marcos forma parte del grupo responsable (Resolución CE 1055/15). El proyecto tiene como objetivo general el de establecer una cooperación de I+D entre LIVEWARE IS e investigadores de la UNICEN (ISISTAN-CONICET) y UTN-FRC (LIDICalSo), con el fin de realizar transferencia de conocimientos sobre Arquitecturas de Software, Mejora de Procesos y Desarrollo Ágil, y aplicar estos conocimientos en la construcción de un método concreto para Arquitecturas de Línea de Producto (SPL). Estas tecnologías permiten un reuso sistemático de activos, facilitan el desarrollo de sistemas, y pueden integrarse también con principios ágiles de desarrollo.

El principal producto a obtener del proyecto es una tecnología de software, que comprende:

i. el método formalizado en términos de sus artefactos, procesos, y guías de adaptación,
ii. un conjunto de herramientas prototipo para facilitar su aplicación en proyectos.

Se espera que esta tecnología incremente las capacidades de LIVEWARE IS para llevar adelante proyectos centrados en arquitecturas de software, y le brinde ventajas competitivas en el sector.

El proyecto ha sido aprobado pero aún no se ha comenzado a trabajar en el mismo, debido a que los recursos no han sido liberados por el momento.

9.4 OTRAS ACTIVIDADES TECNOLÓGICAS CUYOS RESULTADOS NO SEAN PUBLICABLES (desarrollo de equipamientos, montajes de laboratorios, etc.).

9.5 *Sugiera nombres (e informe las direcciones) de las personas de la actividad privada y/o pública que conocen su trabajo y que pueden opinar sobre la relevancia y el impacto económico y/o social de la/s tecnología/s desarrollada/s.*

- Vanesa Dell Acqua, Temperies S. A. vdellacqua@temperies.com

10. SERVICIOS TECNOLÓGICOS*. Indicar qué tipo de servicios ha realizado, el grado de complejidad de los mismos, qué porcentaje aproximado de su tiempo le demandan y los montos de facturación.*

11. PUBLICACIONES Y DESARROLLOS EN:

11.1 DOCENCIA

- Material didáctico para la materia Metodologias de Desarrollo de Software I http://metodologias.alumnos.exa.unicen.edu.ar/Home/apuntes
- Material didáctico para la materia Métodos Agiles para el Desarrollo de Software http://metagiles.alumnos.exa.unicen.edu.ar/Home/apuntes
- Material didáctico para la materia Estrategias para mejorar la Separación de Concerns http://ccc.alumnos.exa.unicen.edu.ar/apuntes-2011

11.2 DIVULGACIÓN

12. DIRECCION DE BECARIOS Y/O INVESTIGADORES. *Indicar nombres de los dirigidos, Instituciones de dependencia, temas de investigación y períodos.*

Dirección de Investigadores

- Codirectora en conjunto con el Dr. Andrés Diaz Pace del Dr. Santiago Vidal como Investigador Asistente de CONICET. Nro. Resolución designación 955/15 Fecha: 01/04/2015.

Dirección de Becarios

- Directora de la beca DeltaG "PROYECTO DE ESTÍMULO A LA GRADUACIÓN DE ESTUDIANTES DE CARRERAS DE INGENIERÍA" de la alumna Alejandra Massillo. 2015.
- Directora de la beca DeltaG "PROYECTO DE ESTÍMULO A LA GRADUACIÓN DE ESTUDIANTES DE CARRERAS DE INGENIERÍA" del alumno Martin Pavletic. 2014 - 2015.
- Co-directora en conjunto con el Dr. Andres Díaz Pace de la Beca de Formación de Postgrado Tipo II del Mg. Alejandro Rago otorgada por CONICET - Abril 2013 a Marzo 2015. Tema: Asistencia para la Identificación de Concerns en Etapas Tempranas del Desarrollo de Software. Resolución D N 4103 23/11/2012.
- Co-directora en conjunto con la Dra. Sandra Casas de Fabiana Miranda Análisis de desarrollo de aplicaciones iTVD con features. Beca de investigación para alumnos de posgrado, otorgada por la Universidad Nacional Patagonia Austral. Exp Nro. 50048-UNPA-2013. Resolución Nro. 0016/14-R-UNPA.
- Directora de la beca de entrenamiento de CIC junto con el Mg. Santiago Vidal de fecha 26/07/2013 del alumno Lucas Ditz. Tema: Investigación de técnicas para el análisis de un conjunto de refactoring y su impacto en el sistema. Octubre del 2013 a Octubre del 2014.

13. DIRECCION DE TESIS. *Indicar nombres de los dirigidos y temas desarrollados y aclarar si las tesis son de maestría o de doctorado y si están en ejecución o han sido defendidas; en este último caso citar fecha.*

En curso

- Co-directora en conjunto con el Dr. Andrés Díaz Pace del Ing. Hernán Ceferino Vazquez de la tesis de Doctorado en Ciencias de la Computación, *Técnicas de Asistencia para la Conformidad de Arquitecturas,* Fac. de Cs. Exactas UNCPBA. Resolución 180/14, Tandil 15/08/2014.
- Directora en conjunto con la Dra. Sandra Casas del Lic. Hector Reinaga de la tesis de de Maestría en Ingeniería de Sistemas, Conexión de Reglas de Negocio con DSAL (Lenguaje de Aspectos de Dominio Específico). Fac. de Cs. Exactas UNCPBA. Resolución 129/11.
- Tutora de la Ing. Varas, Valeria de la Maestría en Informática y Sistemas. Tema: Trazabilidad de requerimientos para el desarrollo con métodos ágiles. Universidad Nacional Patagonia Austral Expediente N 09053-UNPA-2012.

Finalizado

- Directora en conjunto con el Dr. Andres Diaz Pace del Mg. Alejandro Rago de la tesis de Doctorado en Ciencias de la Computación, *Un Enfoque Automatizado para Asistir el Análisis de Requerimientos Textuales,* Fac. de Cs. Exactas UNCPBA. Resolución 208/09. Defendida Marzo 2015.

14. PARTICIPACION EN REUNIONES CIENTIFICAS. *Indicar la denominación, lugar y fecha de realización, tipo de participación que le cupo, títulos de los trabajos o comunicaciones presentadas y autores de los mismos.*

15. CURSOS DE PERFECCIONAMIENTO, VIAJES DE ESTUDIO, ETC. *Señalar características del curso o motivo del viaje, período, instituciones visitadas, etc.*

16. SUBSIDIOS RECIBIDOS EN EL PERIODO. *Indicar institución otorgante, fines de los mismos y montos recibidos.*

- Integrante del grupo responsable de *Un MÉTODO CENTRADO EN ARQUITECTURAS DE SOFTWARE PARA LÍNEAS DE PRODUCTO CON SOPORTE DE HERRAMIENTAS*. Código PDTS217. PDTS – CIN (Consejo Interuniversitario Nacional). Director: DIAZ Pace, Jorge Andres. Idea proyecto aprobada Resolución CE 1055/15. Monto total 200.000$.
- Integrante del grupo colaborador del proyecto PICT "Refactoring de Sistemas de Software Conducido por la Arquitectura" (PICT 2014-0855). Director: Santiago Vidal. Categoría: temas abiertos. Tipo: B. Total subsidio: $100000. Otorgado por Agencia Nacional de promoción científica y tecnológica.
- Subsidio Institucional para investigadores CIC (Comisión de Investigaciones Científicas de la provincia de Buenos Aires) 2015. Resolución 1266/14, 23 de Septiembre 2015. Suma 8.750$.
- Subsidio Institucional para investigadores CIC (Comisión de Investigaciones Científicas de la provincia de Buenos Aires) 2014. Resolución 833/14, 23 de octubre 2014. Suma 7000$.
- Integrante del grupo responsable de Recomendación de Anomalías de Código relevantes a nivel Arquitectural. Codigo BR/13/09 Programa de Cooperación

Científico-Tecnológica entre MINCYT y CAPES, Brasil (2014-2015) Director: Diaz Pace, Jorge Andres. 1/2014 a 12/2015. Monto $45.000.

- Directora del proyecto Estrategias para mejorar a evolucion y el mantenimiento de sistemas Programa de Subsidios Proyectos de Investigacion Cientifica y Tecnologica - Convocatoria 2013, otorgado por CIC Comisión de Investigaciones Científicas de la provincia de Buenos Aires), Resolución Nro 813/13 del 24 de Febrero 2014, 2014-2015. Suma de $50.000.
- Co-directora del proyecto de investigación tipo 1 Código 29/A273-1 "Técnicas para anticipar y analizar la evolución del software orientado a aspectos" Director: Dra. Sandra Casas. Integrantes: Docentes: Graciela Vidal, Fernanda Oyarzo, Mirtha Miranda, Franco Herrera, Juan Henriquez, Daniel Gonzalez, Marcela Constanzo, Hector Reinaga y la alumna Cecilia Fuentes Zamorano. 2012-2014. Universidad Nacional Patagonia Austral (UNPA), Unidad Académica Río Gallegos (UARG) "http://sites.google.com/site/profeprog/proyecto4"http://sites.google.com/site/profeprog/proyecto4 "http://sites.google.com/site/profeprog/proyecto2")

17. OTRAS FUENTES DE FINANCIAMIENTO. *Describir la naturaleza de los contratos con empresas y/o organismos públicos.*

18. DISTINCIONES O PREMIOS OBTENIDOS EN EL PERIODO.

- Best Paper Award. *Identifying Duplicate Functionality in Textual Use Cases by Aligning Semantic Actions* (SoSyM abstract) Alejandro Rago, Claudia Marcos, Andrés Díaz-Pace. In Proceedings of the ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS'15) | Ottawa, Ontario, Canada, September 2015 | Pages 442 | Publisher: ACM/IEEE | ISBN: 978-1-4673-6908-4/15 | DOI: 10.1109/MODELS.2015.7338276

19. ACTUACION EN ORGANISMOS DE PLANEAMIENTO, PROMOCION O EJECUCION CIENTIFICA Y TECNOLÓGICA. *Indicar las principales gestiones realizadas durante el período y porcentaje aproximado de su tiempo que ha utilizado.*

- Evaluadora externa de proyectos de investigacion 2016-2017. Secretaría de Investigación y Desarrollo de la Universidad Nacional de Tres de Febrero. Diciembre 2015. (3 días)
- Evaluadora externa del Programa Nacional de Incentivos a Docentes-Investigadores correspondientes a la convocatoria 2014, de la Facultad de Ciencias Exactas y Naturales. UNLPam. Octubre 2014. (3 días)

20. TAREAS DOCENTES DESARROLLADAS EN EL PERIODO. *Indicar el porcentaje aproximado de su tiempo que le han demandado.*

- Metodologias de Desarrollo de Software I. Facultad de Ciencias Exactas, UNCPBA, Tandil, (3 hrs. semanales durante el 1er cuatrimestre).
- Estrategias para mejorar la separación de concerns curso de Grado, Maestría en Ingeniería de Sistemas y Doctorado en Ciencias de la Computación. UNICEN, Fac. de Ciencias Exactas, Dpto. de Computación y Sistemas. 2014 Resolución 201/14; 2015 Resolución 194/15. (2 hrs. semanales durante el 2do. Cuatrimestre).
- Métodos Agiles (MA) para el Desarrollo de Software curso de Grado, Maestría en Ingeniería de Sistemas y Doctorado en Ciencias de la Computación. UNICEN, Fac. de

Ciencias Exactas, Dpto. de Computación y Sistemas. 2014 Resolución 201/14; 2015 Resolución 194/15 .(2 hrs. semanales durante el 2do. Cuatrimestre).

21. OTROS ELEMENTOS DE JUICIO NO CONTEMPLADOS EN LOS TITULOS ANTERIORES. *Bajo este punto se indicará todo lo que se considere de interés para la evaluación de la tarea cumplida en el período.*

Actividades de Transferencia

- Acuerdo Específico de Colaboración Científica entre la Empresa Lider Life S.A y la U.N.C.P.B.A. Unidad Ejecutora: Facultad de Ciencias Exactas. Resolución 5365 2/07/14.
- Asesora de la Empresa it-Mentor. "http://www.it-mentor.com.ar/" Desde 2004.
- Colaboradora en la definición de los alcances de las Calificaciones Profesionales de IBM desde 2004.

Actualización Docente de la Provincia de Buenos Aires

- Integrante del equipo de trabajo del Program.AR. del Estado Nacional a través del "PROGRAMA PARA EL DESARROLLO DE LA INFRAESTRUCTURA DESTINADA A PROMOVER LA CAPACIDAD EMPRENDEDORA" destinado a maestros del nivel primario de la ciudad de Tandil. Mg. Carmen Leonardi Responsable Técnica, integrantes Mg. Virginia Mauco, Dra. Claudia Marcos, Mg. Laura Felice, Dr. Santiago Vidal e Ing. Federico Améndola TANDIL: 03/07/2015. RESOLUCIÓN: 171/15 y 177/15.
- Integrante del equipo de trabajo del Program.AR. del Estado Nacional a través del "PROGRAMA PARA EL DESARROLLO DE LA INFRAESTRUCTURA DESTINADA A PROMOVER LA CAPACIDAD EMPRENDEDORA" destinado a quipos de la Dirección General de Cultura y Educación de la Provincia de Buenos Aires. Mg. Carmen Leonardi Responsable Técnica, integrantes Mg. Virginia Mauco, Dra. Claudia Marcos, Mg. Laura Felice, Dr. Santiago Vidal e Ing. Federico Améndola TANDIL: 12/04/2016. RESOLUCIÓN: 090/16.

Actividades de Gestión

- Miembro Suplente del Consejo Editorial de la Universidad. 2013-2014. Resolución: 118/13. Tandil 26/04/13. Resolución: Nº5260, Tandil 26/03/2014.
- Integrante del Consejo de Representantes de la Red Institucional de Modelación de Sistemas Agropecuarios de la Región de Buenos Aires (MODASUR), por la Facultad de Ciencias Exactas. Desde 2009 a la fecha, Resolución 259/09 (23/10/09), Resolución 233/12 (24/08/2012).

Integrante de Comités de Programa y Evaluación de Artículos

- Miembro del Comité de Programa de la Conferencia IADIS Ibero-Americana WWW/Internet 2015. 30 Noviembre 1 Diciembre, Santa Catarina Brazil.
- Miembro del Comité de Programas del ASSE (Argentine Simposium on Software Engineering). En el marco de las 44 JAIIO (Jornadas Argentinas de Informática e Investigación Operativa. Córdoba Argentina. Agosto 2015.

- Miembro del Comité de Programa de la Conferencia IADIS Ibero-Americana WWW/Internet 2014. 25-27 Octubre, Porto, Portugal.
- Evaluadora de International Journal of Cooperative Information Systems, 2014. Print ISSN: 0218-8430 Online ISSN: 1793-6365
- Evaluadora de ESWA Expert System With Applications. 2014.

Actividades de Evaluación

- Miembro del comité técnico de LACCIR (Latin American and Caribbean Collaborative ICT Research): http://www.laccir.org/ desde 2007.
- Miembro del Registro de Expertos de la CONEAU desde 2005.

Miembro de Comisiones de Postgrado

- Miembro suplente de la Comisión de Posgrado en Doctorado en Ciencias de la Computación (CPCC) 2015 - 2018. Resolución 155/15.
- Miembro de la Comisión de Postgrado en Maestría en Ingeniería de Sistemas (CPMIS). 2012 – 2015. Resolución 051/12.
- Miembro suplente de la Comisión de Posgrado en Doctorado en Ciencias de la Computación (CPCC) 2011 - 2014. Resolución 112/11.
- Miembro suplente del Comité Académico de la Carrera Maestría en Informática y Sistemas de la Universidad Nacional de la Patagonia Austral. Resolución 032/10-R-UNPA. 23/04/10.

Actividades de Evaluación en Postgrado

- Jurado suplente de la tesis para el Doctorado en Ciencias de la Computación del Ing. Alejandro Corbellini *Mecanismos de soporte para procesamiento distribuido de algoritmos de recomendación en redes sociales.* Diciembre 2015. Resolución Consejo Académica 323/15.
- Jurado de la tesis para el Doctorado en Ciencias de la Computación del Ing. Martin Garriga *Selección, Prueba y Adaptación de Servicios para Integración en Aplicaciones Orientadas a Servicios.* Diciembre 2015. Resolución Consejo Académica 319/15.
- Jurado de la tesis para el Doctorado en Ciencias de la Computación del Ing. Sergio Salinas *ProDG: un midlleware para Desktop Grids basado en la optimización de la productividad del sistema.* Abril 2015. Resolución Consejo Académica 037/15.

Dirección de Trabajos Finales de Grado

- Directora en conjunto con el Dr. Alejandro Rago del trabajo final *Recuperación de Trazabilidad entre Documentos de Diseño y Requerimientos mediante Técnicas semi-automáticas* de los alumnos Attanasio Ruiz German, Gonzalez Rodrigo Damian correspondiente a la carrera Ingeniería de Sistemas. Diciembre 2015.
- Directora en conjunto con el Dr. Santiago Vidal del trabajo final *REFACTORIZACIÓN AUTOMATIZADA PARA LA ELIMINACIÓN DE BRAIN METHODS* de los alumnos Santiago Zuliani Held e Iñaki Berra correspondiente a la carrera Ingeniería de Sistemas. Junio 2015.
- Directora en conjunto con la Ing. Vanesa Dell Acqua del trabajo final *TRAZABILIDAD ENTRE REQUERIMIENTO, COMPONENTES Y CASOS DE PRUEBA* ASOCIADOS del alumno Martin Pavletic correspondiente a la carrera Ingeniería de Sistemas. Octubre 2014.

Miembro de Jurado de Concursos Docentes

- Integrante del Tribunal Evaluador para cubrir el cargo de Profesor Adjunto dedicación parcial en la catedra Programación Orientada a Objetos de la carrera Ingeniería de Sistemas. Facultad de Ciencias Exactas. Universidad Nacional del Centro de la Provincia de Buenos Aires. Resolución de Consejo Académico Nº 122/15. 22 de Mayo 2015.
- Integrante del Tribunal Evaluador para cubrir el cargo de Profesor Adjunto dedicación parcial en la catedra Taller de Sistemas Multi-Agentes de la carrera Ingeniería de Sistemas. Facultad de Ciencias Exactas. Universidad Nacional del Centro de la Provincia de Buenos Aires. Resolución de Consejo Académico Nº 122/15. 22 de Mayo 2015.
- Integrante del Tribunal Evaluador para cubrir el cargo de Profesor Adjunto dedicación parcial en la catedra Introducción a la Programación Evolutiva de la carrera Ingeniería de Sistemas. Facultad de Ciencias Exactas. Universidad Nacional del Centro de la Provincia de Buenos Aires. Resolución de Consejo Académico Nº 122/15. 22 de Mayo 2015.
- Integrante del Tribunal Evaluador para cubrir el cargo de Profesor Adjunto dedicación parcial en la catedra Ingenierïa de Software de la carrera Ingeniería de Sistemas. Facultad de Ciencias Exactas. Universidad Nacional del Centro de la Provincia de Buenos Aires. Resolución de Consejo Académico Nº 122/15. 22 de Mayo 2015.
- Integrante del Tribunal Evaluador para cubrir el cargo de Profesor Adjunto dedicación parcial en la catedra Sistemas Operativos de la carrera Ingeniería de Sistemas. Facultad de Ciencias Exactas. Universidad Nacional del Centro de la Provincia de Buenos Aires. Resolución de Consejo Académico Nº 122/15. 22 de Mayo 2015.
- Integrante del Tribunal Evaluador para cubrir el cargo de Profesor Adjunto dedicacion parcial en la catedra Análisis y Diseño de Sistemas II de la carrera Ingeniería Informatica. Facultad de Ingenieria. Universidad Nacional de Mar del Plata. Resolución de Consejo Académico Nº 1231 Diciembre 2014.
- Integrante del Tribunal Evaluador para cubrir el cargo de Jefe de Trabajos Practicos dedicacion simple en la catedra Análisis y Diseño de Sistemas I de la carrera Ingeniería Informatica. Facultad de Ingenieria. Universidad Nacional de Mar del Plata. Resolución de Consejo Académico Nº 1231 Diciembre 2014.

22. TITULO Y PLAN DE TRABAJO A REALIZAR EN EL PROXIMO PERIODO. *Desarrollar en no más de 3 páginas. Si corresponde, explicite la importancia de sus trabajos con relación a los intereses de la Provincia.*

Estrategias para Mejorar el Mantenimiento de Sistemas

1. Calidad del Software

La calidad de un software hace referencia a "la totalidad de aspectos y características de un producto o servicio que tienen que ver con su habilidad de satisfacer las necesidades explícitas o implícitas" [7]. Para lograr esta calidad existen, el ciclo de vida del software, metodologías de desarrollo, modelos de mejora continua de procesos, crosscutting concerns, atributos de calidad arquitectónicos, entre otros [11]. Por lo tanto, es muy importante centrar el desarrollo de un software en su calidad [3,12].

La calidad del software no se refiere únicamente a obtener un producto sin errores, sino también a que su código fuente se pueda extender y modificar de manera simple, y entender sin que resulte un problema con el paso del tiempo, ya que el día después de entregar el software importa incluso más que cuando se entrega [4]. Por lo tanto, el mantenimiento del software, es muy importante para mantener en alto el nivel de calidad de un software.

Es importante señalar que la calidad del software no se refiere únicamente a obtener un producto sin errores, sino también a que su código fuente se pueda extender, modificar de manera simple y entender sin que ésto resulte un problema con el paso del tiempo, ya que el día después de desarrollar software importa incluso más que cuando se entrega [6]. Al abordar la tarea de mantenimiento de un sistema se encuentra que el código tiende a ser extenso y complejo, y por ende difícil de adaptar a los nuevos requerimientos [9]. En general, el principal problema es el cambio de los requerimientos del sistema que se modificaron desde que éste se diseñó y produjeron un cambio en el código. Estos problemas de diseño son generalmente materializados como code smells. Un code smell, se refiere a cualquier síntoma que puede indicar que el código fuente presenta problemas [5].

2. El problema de los Code Smells

Los code smells no son errores, pero tampoco son técnicamente correctos y no impiden el funcionamiento del sistema. Indican debilidades en el diseño que pueden incrementar el costo de mantenimiento y evolución o el riesgo de errores o fallos en el futuro.

La existencia y aumento de code smells en el código genera una degradación en la calidad del software, impactando en la comprensión, mantenibilidad y evolución del código fuente [8]. Un desarrollador puede intuir que su sistema contiene code smells cuando hay [7]:

● Código duplicado: Este problema suceder cuando se repite el mimo código en una clase. El código duplicado lleva a programas difíciles de modificar. Para solucionarlo se deben eliminar las líneas de código que son exactamente iguales y se repiten en varias ocasiones.

● Métodos muy largos: Cuando los métodos son extremadamente largos se dificulta mucho su entendimiento, volviéndose difícil agregar funcionalidad o encontrar errores. Las clases que contienen a estos métodos se vuelven complejas de mantener. En este caso hay que dividir el código en porciones y extraerlas para crear métodos más pequeños, que sean más fáciles de mantener, reusar y comprender.

● Clases muy grandes: Este resulta un caso similar al anterior, pero puede significar una mala división de responsabilidad de los objetos. Las clases de gran tamaño llevan a un sistema que es complejo de entender y extender. Se debe tratar de identificar qué actividades realiza esa clase y ver si realmente esas actividades se encuentran relacionadas. Si no es así, hacer clases más pequeñas, con una correcta división de la funcionalidad.

● Métodos que necesitan muchos parámetros: Cuando hay alto acoplamiento entre métodos se requieren pasar como parámetro muchas variables. Este problema tiene como resultado un método poco mantenible. Se puede solucionar realizando una clase que contenga esos parámetros y utilizar esa clase en el pasaje por parámetros. Esto se realiza, en especial, cuando los parámetros están fuertemente relacionados o suelen utilizarse juntos.

Para solucionar los code smells se utiliza la técnica de refactoring [5]. El refactoring se describe como el proceso de realizar un cambio en la estructura interna del software para que éste sea más fácil de entender y más simple de modificar sin cambiar su comportamiento. El refactoring es realizado por los desarrolladores como parte del proceso de desarrollo del

software, ya que incrementa la legibilidad y mantenibilidad. Este proceso, además, permite mejorar el diseño del software mejorando su modularización y mantenimiento.

A pesar de la importancia del proceso de refactoring como medio para mantener la calidad del software, éste no siempre es realizado con la frecuencia ni con el tiempo que requiere. La falta de refactorings, en general, se debe a lo costoso que puede resultar el proceso de detección de code smells, la forma de solucionarlos, y el riesgo de generar errores durante el proceso. También, en la industria, la frecuencia y tiempo que se le dedica a este proceso se ve afectada por los cortos tiempos de entrega que se disponen y el costo adicional producido por el refactoring.

El autor K. Beck, en uno de sus libros sobre la metodología ágil eXtreme Programming, afirma que el refactoring ahorra tiempo en desarrollo y mejora la calidad [1], instando a los desarrolladores a incluir al refactoring como una etapa que sea parte del ciclo de desarrollo del software. Por otro lado, existe la idea de que los ingenieros de software a menudo evitan el refactoring cuando se encuentran limitados con respecto a recursos y tiempos de entrega [2]. En general, muy pocos desarrolladores se acercan a un equilibrio entre la adición de nueva funcionalidad y la utilización de refactoring [2]. En cualquiera de los casos, la falta de herramientas para determinar el impacto real de refactoring y sus efectos secundarios contribuyen con la decisión de posponer el refactoring reiteradas veces, y en la mayoría de los casos nunca es aplicado [10].

4. Plan de Trabajo para el Período 2016-2018

El objetivo principal de la investigación para el próximo período está orientado a completar el ciclo de refactoring de un sistema comenzado en el período anterior. JSpIRIT realiza un ranking de code smells según su prioridad, para ello tiene en cuenta escenarios de modificabilidad, historia de las clases y la complejidad de cada code smell. Luego, se proveen diferentes alternativas para poder solucionar el code smell Brain Method. Lsa actividades de investigaciónstarán centradas en:

- Analizar la incorporación de nuevas estrategias a JSpIRIT orientadas a realizar un análisis de la calidad de la arquitectura del sistema. Una de las estrategias que se está estudiando para incorporar es el análisis de problemas arquitectónicos que posea el sistema y la degradación de la arquitectura.
- Analizar las estrategias de resolución de los code smells presentados en los catálogos. Los catálogos de code smells proveen diferentes estrategias de solución, cada uno será analizado para identificar las actividades que puedan ser automatizadas.
- Definir una estrategia de análisis de la aplicación de un refactoring. Actualmente, la herramienta provee alternativas de solución del code smell Brain Method. El objetivo es proveer un análisis de las diferentes alternativas de refactoring del resto de los code smells, proponiendo diferentes soluciones y proveyendo una análisis de costo/beneficio de cada una de dichas alternativas de resolución. Para ello, se tendrá en cuenta el análisis realizado para la resolución del code smell Brain Method.
- Identificación de casos de estudio que permitan validar la propuesta.

Cabe aclarar que a medida que se vayan obteniendo resultados, los mismos serán publicados en congresos y revistas nacionales e internacionales para divulgar la investigación realizada.

Adicionalmente, se seguirán analizando estrategias para mejorar la modularización de un sistema en todas las etapas del ciclo de vida, pricipalmente en la etapa de diseño.

5. Referencias

[1] Beck, Kent. Extreme programming explained: embrace change Addison-Wesley Professional, 2000.

[2] Belady, Laszlo A., and Meir M. Lehman. A model of large program development IBM Systems journal 15.3 (1976): 225-252.

[3] Ben Menachem, Mordechai, and G. S. Marliss. Software Quality: Producing Practical, Consistent Software, Slaying the Software Dragon Series. Boston, MA: International Thomson Computer Press, 1997.

[4] Černý, Vladimír. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of optimization theory and applications 45.1 (1985): 41-51.

[5] Fowler, Martin. Refactoring: improving the design of existing code Pearson Education India, 2002.

[6] Herranz López, José Ignacio. Desarrollo de software para la autodiagnosis del grado de madurez de una empresa conforme a la ISO 90003, 2011.

[7] International Organization for Standardization. ISO 8402: 1994: Quality Management and Quality Assurance Vocabulary. International Organization for Standardization, 1994.

[8] Kirkpatrick, Scott, and M. P. Vecchi. Optimization by simmulated annealing. Science 220.4598 (1983): 671-680.

[9] Lehman M.M. and Ramil J.F. . An approach to a theory of software evolution. In Proceedings of the 4th international Workshop on Principles of Software Evolution (IWPSE '01), pages 70-74, Vienna, Austria. ACM, 2001.

[10] Murphy Hill, Emerson, and Andrew P. Black. Breaking the barriers to successful refactoring. Software Engineering, 2008. ICSE'08. ACM/IEEE 30th International Conference on. IEEE, 2008.

[11] Opdyke W. and Don R. Refactoring Object-Oriented Software to Support Evolution and Reuse OOPSLA 96: 11th Annual Conference on Object-Oriented Program Systems, Languages and Applications, San Jose, California, 1996.

[12] OTI: JDT Programmer's GUIDE. IBM Corporation (2005).

Condiciones de la presentación:

A. El Informe Científico deberá presentarse dentro de una carpeta, con la documentación abrochada y en cuyo rótulo figure el Apellido y Nombre del Investigador, la que deberá incluir:
   a. Una copia en papel A-4 (puntos 1 al 21).

b. Las copias de publicaciones y toda otra documentación respaldatoria, en otra carpeta o caja, en cuyo rótulo se consignará el apellido y nombres del investigador y la leyenda "Informe Científico Período .....".

c. Informe del Director de tareas (en los casos que corresponda), en sobre cerrado.

B. Envío por correo electrónico:

a. Se deberá remitir por correo electrónico a la siguiente dirección: HYPERLINK "mailto:infinvest@cic.gba.gob.ar"infinvest@cic.gba.gob.ar (puntos 1 al 21), en formato .doc zipeado, configurado para papel A-4 y libre de virus.

b. En el mismo correo electrónico referido en el punto a), se deberá incluir como un segundo documento un currículum resumido (no más de dos páginas A4), consignando apellido y nombres, disciplina de investigación, trabajos publicados en el período informado (con las direcciones de Internet de las respectivas revistas) y un resumen del proyecto de investigación en no más de 250 palabras, incluyendo palabras clave.

C. Sistema SIBIPA:

a. Se deberá peticionar el informe en la modalidad on line, desde el sitio web de la CIC, sistema SIBIPA (ver instructivo).

Nota: El Investigador que desee ser considerado a los fines de una promoción, deberá solicitarlo en el formulario correspondiente, en los períodos que se establezcan en los cronogramas anuales.