CARRERA DEL INVESTIGADOR CIENTÍFICO Y TECNOLÓGICO

Informe Científico[1]

PERIODO [2]: 10/05/2012 al 31/12/2013.

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**

Desarrollo Orientado a Aspectos de un Simulador Ganadero con Métodos Agiles

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: 0249 4439681

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. 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.

Desde el comienzo del trabajo con investigadores de la Facultad de Ciencias Veterinarias se desarrolló un simulador ganadero siguiendo un ciclo de vida iterativo e incremental con Crystal Clear con algunas prácticas de otros métodos ágiles. Durante la etapa de mantenimiento y evolución de un sistema varios problemas surgieron debido a problemas de modularidad del mismo provocando que el mantenimiento sea complejo y costoso. Uno de los problemas identificados es que la integración continua no se realizaba de manera correcta produciendo código difícil de mantener. Por esa razón, se definió un proceso que permite identificar problemas en el código de una aplicación, este proceso tiene en cuenta 3 aspectos del código, por ejemplo, problemas en el código, denominados code smells; escenarios de calidad, principalmente aquellos destinados a la modificabilidad; y la historia de cada componente, teniendo en cuenta cuánto ha sido modificada cada clase del sistema. De esta manera, se obtiene un ranking para que el desarrollador pueda resolver los problemas más importantes. Para dar soporte al proceso de identificación de problemas en el código de una aplicación, se implementó la herramienta SpIRIT. SpIRIT (Identificación Inteligente de Oportunidades de Refactorización) prioriza los code smells más críticos para un sistema. Dado un sistema orientado a objetos con code smells, SpIRIT ayuda al desarrollador priorizando los code smells. Además, SpIRIT sugiere refactorizaciones candidatas para cada smell. En cuanto a la sugerencia de refactorizaciones, SpIRIT busca determinar las alternativas de refactorización para cada code smell de su ranking.

Adicionalmente, debido a que el costo de mantenimiento es considerado uno de los más altos del proyecto, se comenzó a trabajar sobre la captura de requerimientos intentando modularizar el sistema desde las primeras etapas del ciclo de vida. La idea es identificar en los documentos de requerimientos, especificamente en los casos de uso, aquellos concerns relevantes del sistema. 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. La herramienta denominada REAssistant REquirements Analysis Assistant) ha sido implementada utilizando UIMA ya que provee componentes para realizar un análisis de las oraciones de los casos de uso. 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.

Por medio de estas dos estrategias, se proveen alternativas para solucionar los problemas de código de los sistemas legados, como desarrollar un sistema teniendo en cuenta desde las primeras etapas del ciclo de vida los concerns más importantes. De esta manera, se intenta mejorar el mantenimiento y evolución de los sistemas obteniendo una buena

modularidad de las componentes. El simulador ganadero con base pastoril ha sido un importante caso de estudio utilizado en los experimentos desarrollados.

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

***7.1 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. ***A Catalog of Aspect Refactorings for Spring/AOP*. Santiago A. Vidal, Claudia A. Marcos, JUCS Journal of Universal Computer Science. Vol 19 issue 1, pp 157-182. March 2013.**

Abstract. The importance of enterprise applications in current organizations makes it necessary to facilitate their maintenance and evolution along their life. These kind of systems are very complex and they have several requirements that orthogonally crosscut the system structure (called crosscutting concerns). Since many of the enterprise systems are developed with the Spring framework, can be taken advantage of the benefit provided by the aspect-oriented module of Spring in order to encapsulate the crosscutting concerns into aspects. In this way, the maintenance and evolution of the enterprise systems will be improved. However, most of the aspect refactorings presented in the literature are not directly applicable to Spring systems. Along this line, in this work we present an adaptation of a catalog of aspect refactorings, initially presented for AspectJ, to be used with Spring/AOP. Also, we conduct a case study in which two enterprise applications developed with the Spring framework are refactored in order to encapsulate their crosscutting concerns into aspects.

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

2. ***Toward automated refactoring of crosscutting concerns into aspects*, Santiago A. Vidal, Claudia A. Marcos, Journal of Systems and Software, ISSN 0164-1212, 2012. pp. 1482-1497 DOI information: 10.1016/j.jss.2012.12.045. Volume 86, Issue 6, June 2013, Pages 1482–1497**

Abstract. Aspect-Oriented Programming (AOP) improves the separation of concerns by encapsulating crosscutting concerns into aspects. Thus, aspect-oriented programming aims to better support the evolution of systems. For this reason, in order to improve the evolution of systems using AOP, we have defined a process that assists the developer to refactor an object-oriented system into an aspect-oriented one. In this paper we propose the use of association rules and Markov models to achieve better assistance of some tasks of this process. Specifically, we use these techniques to help the developer in the task of encapsulating a fragment of aspectizable code into an aspect. This includes the choice of a

fragment of aspectizable code to be encapsulated, the selection of a suitable aspect refactoring, and the analysis and application of additional restructurings when necessary. We demonstrate the advantages of the approach by conducting a case study of the refactoring of a J2EE system.

Grado de participación: Alto, ya que estuvo directamente involucrada en el desarrollo del enfoque propuesto y en la escritura del paper. El trabajo es una parte del resultado de investigación de la tesis de maestría de Santiago Vidal donde Claudia Marcos fue la directora.

3. ***Uncovering Quality-attribute Concerns in Use-case Specifications via Early Aspect Mining*. Alejandro Rago, Claudia Marcos, Andrés Díaz Pace. Requirements Engineering. Springer ISSN 0947-3602. March 2013, Volume 18, Issue 1, pp 67-84 (ISI Index) DOI: 10.1007/s00766-011-0142-z**

Abstract. Quality-attribute requirements describe constraints on the development and behavior of a software system, and their satisfaction is key for the success of a software project. Detecting and analyzing quality attributes in early development stages provides insights for system design, reduces risks, and ultimately improves the developers' understanding of the system. A common problem, however, is that quality-attribute information tends to be understated in requirements specifications, and scattered across several documents. Thus, making the quality attributes first-class citizens becomes usually a time-consuming task for analysts. Recent developments have made it possible to mine concerns semi-automatically from textual documents. Leveraging on these ideas, we present a semi-automated approach to identify latent quality attributes that works in two stages. First, a mining tool extracts early aspects from use cases, and then these aspects are processed to derive candidate quality attributes. This derivation is based on an ontology of quality-attribute scenarios. We have built a prototype tool called QAMiner to implement our approach. The evaluation of this tool in two case-studies from the literature has shown interesting results. As main contribution, we argue that our approach can help analysts to skim requirements documents and quickly produce a list of potential quality attributes for the system.

Grado de participación: Alto, ya que estuvo directamente involucrada en el desarrollo del enfoque propuesto y en la escritura del paper. El trabajo es una parte del resultado de investigación de la tesis de maestría de Alejandro Rago donde Claudia Marcos y Andrés Díaz Pace fueron los directores.

4. ***Rescue of a Whole-Farm System: Crystal Clear in Action*. Pablo Mangudo, Mauricio Arroqui, Claudia Marcos and Claudio Machado. Revista: International Journal of Agile and Extreme Software Development (IJAESD). Inderscience Publishers. ISSN (Online): 1743-5145 - ISSN (Print): 1743-5137. 6-22 pp. Vol 1. No 1. 2012. DOI: 10.1504/IJAESD.2012.048306**

Abstract. This paper presents a case study of an agricultural project. From a desktop research model (stage I), a web-based whole-farm simulator was developed applying a waterfall life cycle (stage II) but several problems were detected and the project failed. The project was continued (stage III) applying Crystal Clear agile method, which suited better the requirements. An efficient team communication and the frequent delivery of usable code increasingly contributed to the sponsor’s satisfaction. It was positively concluded that

Crystal Clear was able to rescue the project and that it could be applied in a short-term period without major difficulties.

Grado de participación: Alto, ya que estuvo directamente involucrada en la definición del método ágil propuesto, en el desarrollo del caso de estudio y en la escritura del paper.

5. ***Building an expert system to assist system refactorization*. Santiago Vidal, Claudia Marcos. Revista: Expert Systems With Applications – Elsevier. ISSN:0957-4174 39 (2012), pp. 3810-3816 DOI information: 10.1016/j.eswa.2011.09.084**

Abstract. The separation of concerns is an important issue in the building of maintenable systems. Aspect oriented programming (AOP) is a software paradigm that allows the encapsulation of those concerns that crosscut a system and can not be modularized using current paradigms such as object-oriented programming. In this way, AOP increases the software modularization and reduces the impact when changes are made in the system. In order to take advantage of the benefits of AOP, the legacy OO systems should be migrated. To migrate object-oriented systems to aspect-oriented ones, specific refactorings for aspects should be used. This is a complex and tedious task for the developer because he/she needs to know how the refactorings should be applied and under what context. Therefore, it is desirable to have tools that help him/her through the process. In this article, we present an expert software agent, named RefactoringRecommender, that assists the developer during a refactorization of a system. The agent uses a Markovian algorithm with the goal of predicting the needed restructurings.

Grado de participación: Alto, ya que estuvo directamente involucrada en el desarrollo del enfoque propuesto y en la escritura del paper. El trabajo es una parte del resultado de investigación de la tesis de maestría de Santiago Vidal donde Claudia Marcos fue la directora.

Artículos en congresos

6. ***An approach to prioritize code smells for refactoring*. Santiago Vidal, Claudia Marcos, Andrés Díaz Pace. In First Latin-American School on Software Engineering (ELA-ES), Rio de Janeiro, Brasil, Julio 2013. Best PhD thesis awards.**

Abstract. Software evolution and maintenance involve high costs in the development process, particularly as systems become larger and complex. A usual concern that makes system maintenance and evolution difficult is the existence of structural design problems, which were not sufficiently taken care of in early development stages. These design problems are often described as code smells. A code smell is a symptom in the source code that helps to identify a design problem. In this way, code smells allow developers to detect fragments of code that should be re-structured, in order to improve the quality of the system. A technique commonly used to fix code smells is refactoring. Different semi-automated tools can be applied to identify code smells in a system. However, a major limitation of existing tools is that they usually find numerous code smells. This is a challenging problem for the developer, for a number of reasons. First, she can get overwhelmed by the amount of information to be analyzed. Second, the effort needed to fix all the code smells usually exceeds the budget that the developer has available for refactoring. Third, in practice, not all code smells are equally important for the goals of the system or its health. Therefore, the developer has to manually peruse the list of code smells and select a set of smells that will be fixed. In this context, the provision of tool support for assisting the developer to quickly identify high-priority code smells becomes essential. In our research, we propose a

semi-automated approach called SpIRIT (Smart Identification of Refactoring opportunITies) that treats refactoring as a cost-effective activity. By cost-effective, we mean that the analysis and re-structuring efforts are driven by a handful of code problems considered as critical for the current system, so that solving those problems will positively contribute to the system quality but with a limited refactoring expenditure. Given an object-oriented system with a number of code smells, SpIRIT assists the developer in two tasks: (i) prioritizing the code smells, and (ii) suggesting candidate refactorings for each smell using a cost-benefit analysis. The novel aspect of our approach is the prioritization of code smells based on assessing their relationships with modifiability issues. Our assessment of a code smell instance is determined by the following factors: past component modifications, important modifiability scenarios for the system, and types of code smells. Regarding the suggestion of refactorings, SpIRIT seeks to determine refactoring alternatives for each code smell of its ranking. After searching through the design space of refactoring alternatives, the tool should assess a set of candidate refactorings in terms of the improvement that their application would produce in the system.

Grado de participación: Alto, ya que estuvo directamente involucrada en el desarrollo del enfoque propuesto 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 y Andrés Díaz Pace fueron los directores.

7. ***Un Enfoque para Automatizar la Refactorización de Casos de Uso*. Alejandro Rago, Paula Frade, Miguel Ruival, Claudia Marcos. Proceedings of ASSE 2013 Argentine Symposium on Software Engineering. JAIIO - Jornadas Argentinas de Informática e Investigación Operativa. Córdoba Argentina Setiembre 2013.**

Resumen. Llevar a cabo las actividades de captura y modelamiento de requerimientos no es una tarea sencilla. Ésta requiere realizar un análisis profundo de las necesidades de los clientes y demanda cierto grado de experiencia de los analistas. Para comunicar satisfactoriamente los requerimientos, se deben aprovechar los instrumentos provistos por las técnicas de especificación (por ejemplo, relaciones entre casos de uso) de forma tal que se evite la redundancia y se promueva el reuso y abstracción de comportamiento. En la práctica, estos instrumentos no son utilizados tanto como deberían y es necesario que los analistas revisen los documentos periódicamente para preservar la calidad de los requerimientos. Desafortunadamente, inspeccionar los documentos manualmente es una tarea compleja y ardua. Por ello, en este artículo presentamos un enfoque semi-automático que permite encontrar defectos en las especificaciones de casos de uso y solucionarlos por medio de refactorizaciones. El enfoque implementa heurísticas avanzadas para localizar los defectos (por ej., comportamientos duplicados) y mecanismos que agilizan su resolución. El enfoque fue evaluado en sistemas reales, obteniendo resultados preliminares prometedores.

Grado de participación: Alto, ya que estuvo directamente involucrada en el desarrollo del enfoque propuesto y en la escritura del paper. El trabajo es una parte del resultado del trabajo de grado de los alumnos Paula Frade y Miguel Ruival donde Claudia Marcos y Alejandro Rago fueron los directores.

8. ***Predicción del cambio a través de la historia del sistema*. Juan F. Hurtado, Franco Sabadini, Santiago Vidal and Claudia Marcos. Proceedings of ASSE 2013 Argentine**

Symposium on Software Engineering. JAIIO - Jornadas Argentinas de Informática e Investigación Operativa. Córdoba Argentina Setiembre 2013.

Resumen. El mantenimiento de aplicaciones es una de las tareas más costosas dentro del desarrollo software. Por esta razón, es importante la realización de mantenimiento preventivo que permita incorporar con menos esfuerzo futuros requerimientos disminuyendo el costo del desarrollo. En esta línea, este trabajo propone un enfoque que permita predecir las clases de un sistema que tienen mayor probabilidad de cambiar en el futuro. De esta forma, podrá dirigirse el esfuerzo de mantenimiento a un subconjunto reducido de las clases de un sistema.

Grado de participación: Alto, ya que estuvo directamente involucrada en el desarrollo del enfoque propuesto y en la escritura del paper. El trabajo es una parte del resultado del trabajo de grado de los alumnos Juan Hurtado y Franco Sabadini donde Santiago Vidal y Claudia Marcos fueron los directores.

9. ***Text analytics for discovering concerns in requirements documents*. Alejandro Rago, Claudia Marcos y Andrés Díaz-Pace. Simposio Argentino de Ingeniería de Software (ASSE), Jornadas Argentinas de Informática e Investigación Operativa (JAIIO '12). ISSN: 1850-2792. La Plata**

Abstract. Recent trends in the software engineering community advocate for the improvement of textual requirements using (semi-)automated tools. In particular, the detection of incomplete or understated concerns at early development stages hold potential, due to the negative efects of untreated concerns on the development. Assistive tools can be of great help for analysts to get a quick picture of the requirements and narrow down the search for latent concerns. In this article, we present a tool called REAssistant that supports the process of discovering concerns in textual specifications. To do so, the tool relies on the UIMA framework and EMF-based technologies to provide an extensible architecture for concern-related analyses. Currently, the tool is configured to process textual use cases by using a number of textual analytics modules that identify lexical, syntactical and semantic entities in the specifications. We have conducted a preliminary evaluation of our tool in two case studies, obtaining promising results when comparing to manual inspections and to another tool.

Grado de participación: Alto, ya que estuvo directamente involucrada en el desarrollo del enfoque propuesto y en la escritura del paper. El trabajo son los primeros resultados de investigación del doctorado de Alejandro Rago donde Claudia Marcos y Andrés Díaz Pace son los directores.

7.2 ***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.

10. ***Analyzing the History of Software Systems to Predict Class Changes*. Santiago A. Vidal · Claudia Marcos · J. Andrés Díaz-Pace. Aceptado para su publicación en Argencon 2014.**

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.

11. ***Comparing Semantic-aware NLP Techniques when Applied to Textual Requirements*. Alejandro Rago, Claudia Marcos, J. Andrés Diaz-Pace. Aceptado para su publicación en Argencon 2014.**

Abstract. The advancements in the field of human language understanding by computers, often referred to as NLP, has brought the attention of many engineering industries. In the context of Software Engineering, there is a growing trend for harnessing NLP power with the end goal of assuring the quality of software requirements and avoiding mistakes that are costly to fix later. However, there is little empirical work showing if these advanced NLP techniques are worth the trouble. In this article, we present an exploratory study of semantic-aware NLP techniques for discovering latent concerns in use case specifications. Two techniques were brought into consideration for evaluation: semantic clustering and semantically-enriched rules. The results showed that semantic NLP techniques hold great potential, and if configured accordingly, can help to diminish the effort of requirement analysts and promote making software of better quality.

***7.3 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.*

12. ***Assisting Requirements Analysts to Find Latent Concerns with REAssistant*. Alejandro Rago, Claudia Marcos, Andrés Díaz Pace. Enviado para su evaluación a Automated Software Engineering.**

Estado: Evaluado con revisiones menores

Abstract. Textual requirements are very common in software projects. However, this format of requirements often keeps relevant concerns (e.g., performance, synchronization, data access, etc.) from the analyst's view, because their semantics are implicit in the text. Thus, analysts must carefully review requirements documents in order to identify key concerns and their effects. Concern mining tools based on NLP techniques can help in this activity. Nonetheless, existing tools cannot always detect all the crosscutting effects of a given concern on different requirements sections, as this detection requires a semantic analysis of the text. In this work, we describe an automated tool called REAssistant that supports the extraction of semantic information from textual use cases in order to reveal latent crosscutting concerns. To enable the analysis of use cases, we apply a tandem of advanced NLP techniques (e.g, dependency parsing, semantic role labeling, and domain actions) built on the UIMA framework, which generates different annotations for the use cases.

Then, REAssistant allows analysts to query these annotations via concern-specific rules in order to identify all the effects of a given concern. The REAssistant tool has been evaluated with several case-studies, showing good results when compared to a manual identification of concerns and a third-party tool. In particular, the tool achieved a remarkable recall regarding the detection of crosscutting concern effects.

13. ***Identifying Duplicate Functionality in Textual Use Cases by Aligning Semantic Actions.*** **Alejandro Rago · Claudia Marcos · J. Andrés Diaz-Pace. Enviado para su evaluación a Journal of Software and Systems Modeling.**

Estado: Evaluado con revisiones menores

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 to avoid redundancy and promote reuse/abstraction (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, which reduces their clarity and under-standability by the system stakeholders. 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.

14. ***Mejora de la especificación de Casos de Uso por medio de refactoring.*** **A. Rago, A. Díaz Pace, C. Marcos. Enviado para su evaluación a Revista IEEE latino.**

Estado: En evaluación

Abstract. This work presents a semi-automatic approach for use case refactoring called RE-USE. RE-USE assess the existing quiality problems in use cases and suggests to the functional analyst a prioritized set of candidate refactorings. The analyst reviews the recommendation list and selects the most important refactoring. The tool applies the chosen refactoring and returns an improved specfication. The tool effectiveness in detecting the existing quality problems and recommending the proper refactorings, was assessed using a set of case of studies related to real-world systems getting encouraging results.

15. ***Gestión de Múltiples Proyectos basado en Kanban*****. Claudia Marcos Julia Arocena, Esteban Roasio, Vanesa Dell Acqua. Enviado para su evaluación al CLEI 2014.**

Estado: En evaluación

Abstract. The multi-project administration is dynamic because of the continuous changes and volatility of the market. For this reason, it is necessary to provide strategies and mechanisms to improve the administration of several projects of a company. In this work we present a reference model for the multi-project administration based on Kanban. The base of this model is to provide some kind of visualization of the process in order to

identify the current state and the phase of a project. In this paper we describe the use of the proposed model for six months in a company by a small developer group.

16. ***Understanding and Addressing Exhibitionism in Java*: Empirical Research about Method Visibility**. Santiago A. Vidal · Alexandre Bergel · Claudia Marcos · J. Andrés Díaz-Pace. Enviado para su evaluación a Empirical Software Engineering.

Estado: 2da ronda de evaluación

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.

17. ***An Approach to Prioritize Code Smells for Refactoring*. Santiago A. Vidal · Claudia Marcos · J. Andrés Díaz-Pace. Enviado para su evaluación a Automated Software Engineering. Solicitud de cambios menores.**

Estado: 2da ronda de evaluación

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 refactoring 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.

18. ***A Lightweight framework for prioritizen technical debt.* Santiago A. Vidal · Claudia Marcos · J. Andrés Díaz-Pace. Enviado para su evaluación a Computing and informatics.**

Estado: 2da ronda de evaluación

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 framework to prioritize technical debt related to code smells based on the

configuration of multiple criteria. An instantiation of our framework has been evaluated in a case-study, and the results show that its usage helps developers to fix code smells.

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

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

7.6 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.*

8. TRABAJOS DE DESARROLLO DE TECNOLOGÍAS.

8.1 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.*

8.2 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.*

8.3 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 pueden incluir dos proyectos interesantes con empresas de software de la provincia de Buenos Aires, la empresa Temperies S.A. el cual ha finalizado, pero se sigue en contacto, y el segundo proyecto con Q4Tech S.A. y Lider File S.A.:

Con respecto a la empresa Temperies S.A. Se ha definido un proceso de desarrollo basado en el método ágil Kanban para poder administrar múltiples proyectos al mismo tiempo. El pilar de este modelo es brindar una clase de visualización del proceso de desarrollo para así identificar el estado y la fase en la cual se encuentra cada tarea. La empresa Temperies S.A. tenía el problema de la administración de varios proyectos al mismo tiempo los que compartían los mismos desarrolladores, donde cada proyecto tenía características particulares y eran de diferentes tipos (mantenimiento, desarrollo, implantación de un sistema existente a un cliente nuevo etc). La administración de proyectos que coexisten de forma simultánea generan permanentemente cambios en las prioridades de cada uno, como así también importantes variaciones en la carga de trabajo, lo que en ocasiones produce picos de trabajo. Debido a las diferencias entre cada uno de los tipos de proyectos descriptos, es que existen metodologías específicas para gestionar cada uno de ellos. Sin embargo, ya que para la empresa no resultaba viable tener un equipo de trabajo asignado a cada proyecto, se optó por una estructura múltiples proyectos y por lo tanto resultaba necesario gestionar este conjunto de proyectos de forma conjunta utilizando una única metodología.

Por estas razones, se ha definido un framework o modelo de referencia para la gestión de múltiples proyectos (GMPK) al mismo tiempo, con un único equipo de desarrollo. Para lograr este objetivo, se define un proceso basado en Kanban, el cual es complementado por un conjunto de prácticas provenientes de otras metodologías ágiles (principalmente Scrum y Xp).

Dicho modelo se describe de forma completa a través de la especificación de la dinámica de trabajo, las prácticas que se deben seguir en el día a día, así como también los roles que deben cumplir los miembros del equipo. Además, se brindan recomendaciones y ejemplos acerca de cómo instanciar el modelo bajo diferentes contextos o circunstancias.

Dicho modelo de referencia fue utilizado durante 6 meses en la empresa para la administración de 5 tipos de sistemas, el proceso se fué ajustando a las necesidades hasta lograr un equilibrio obtiendo muy buenos resultados.

Actualmente, el modelo se sigue utilizando en Temperies S. A. pero ya no es necesaria la interacción con Claudia Marcos. Los resultados de este proyecto se han presentado en Agiles 2011 y se ha enviado un paper al CLEI 2014.

El proyecto con Lider File S.A. y Q4Tech está en sus comienzos, otra empresa que se encuentra indirectamente involucrada en el proyecto es Software del Centro S.A.. El objetivo de este proyecto es identificar los problemas de código que tienen en sus sistemas para identificar los problemas más graves y poder solucionarlos a la brevedad. Para poder identificar dichos problemas se utilizará la herramienta SpIRIT (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, SpIRIT ayuda al desarrollador priorizando los code smells. La identificación de los code smells se basa en los catálogos existentes, como el de Fowler y el de Marinescu. El aspecto novedoso de este enfoque es la priorización de code smells basado en la evaluación de sus relaciones con cuestiones modificabilidad. La evaluación de una instancia de code smells es determinada por los siguientes criterios:

• Historial de cambio: Se proporciona pistas sobre la estabilidad de los componentes que participan en un code smell. Cuanto más seguido los componentes se han modificado en las versiones anteriores del sistema, más inestables estos componentes son ante nuevos cambios.

Por ejemplo, la refactorización del smell en una clase que no se ha modificado desde su implementación (y que no se espera que sea modificada en el futuro) puede tener prioridad baja cuando se compara con un smell en una clase que ha recibido modificaciones en las últimas diez versiones.

• Impacto de escenarios de modificabilidad: Si los componentes de un code smell también están afectados por uno o más escenario, 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.

• Relevancia del code smell: Como no todos los tipos de code smells son igualmente problemáticos, la importancia que el desarrollador da a cada tipo smell debe ser tenida encuenta.

Actualmente, el proyecto se encuentra en desarrollo en la etapa de identificación de los problemas existentes en el sistema seleccionado.

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

***8.5** 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
- Ernesto Mordentti, Software del Centro S.A.

ernesto.mordentti@softwaredelcentro.com.ar

9. *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.*

10. PUBLICACIONES Y DESARROLLOS EN:

10.1 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

10.2 DIVULGACIÓN

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

En curso

• Directora de la beca DeltaG "PROYECTO DE ESTÍMULO A LA GRADUACIÓN DE ESTUDIANTES DE CARRERAS DE INGENIERÍA" del alumno Martin Pavletic. 2014.

• 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.

• 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 junto a la Dra. Sandra Casas de LAlIC. Marcela Alejandra Constanzo. En el proyecto: "Tecnicas para anticipar y analizar la exclusion del software orientado a aspectos" (29/A273). Beca para alumnos de posgrado otorgada por la UNPA. Marzo – Diciembre 2013. Resolucion Nro. 0049/13-R-UNPA.

• Dirección de la Beca de Formación de Postgrado Tipo II del Mg. Santiago Vidal otorgada por CONICET - Abril 2012 – Marzo 2014. Tema: Evolución de Sistemas de Software. Resolución D N 3609 06/12/2011

Finalizada

• Directora de la Beca TICs 2011 Becas de Fin de Carrera para Estudiantes de Grado en Carreras TICs de la Agencia de Martin Pavletic 2011-2014.

• Co-directora junto a la Dra. Sandra Casas de la Lic. Marcela Alejandra Constanzo. Beca de investigación para estudiantes de posgrado de la UNPA, en el marco del proyecto "Tecnicas para anticipar y analizar la evolución del software orientado a aspectos" (29/A273).

• Dirección de la Beca de Formación de Postgrado Tipo I del Ing. Alejandro Rago otorgada por CONICET - Abril 2010 a Marzo 2013. Tema: Asistencia para la Identificación de Aspectos Tempranos en Requerimientos. Resolución D N 204 27/01/2010.

• Co-directora junto a la Dra. Sandra Casas de Cecilia Andrea Fuentes Zamorano. En el proyecto: "Tecnicas para anticipar y analizar la evolución del software orientado a aspectos" (29/A273). Beca para alumnos avanzados otorgada por la UNPA. Marzo – Diciembre 2012. Resolucion Nro. 0091/12-R-UNPA.

• Co-directora junto a la Dra. Sandra Casas de la Lic. Graciela Beatriz Vidal. En el proyecto: "Tecnicas para anticipar y analizar la exclusion del software orientado a aspectos" (29/A273). Beca de investigacion para estudiantes de postgrado, otorgada por la UNPA. Marzo – Diciembre 2012. Resolucion Nro. 0091/12-R-UNPA.

Dirección de la Beca de Formación de Postgrado Tipo I del Ing. Santiago Vidal otorgada por CONICET - Abril 2009 a Marzo 2012. Tema: Migración de Sistemas Orientados a Objetos a Sistemas Orientados a Aspectos. Resolución D N 3105 30/12/08.

12. 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

• 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.

• Directora en conjunto con el Dr. Andres Diaz Pace del Mg. Alejandro Rago de la tesis de Doctorado en Ciencias de la Computación, Asistencia para la identificación de aspectos tempranos en requerimientos, Fac. de Cs. Exactas UNCPBA.

Finalizado

• Directora en conjunto con el Dr. Andres Diaz Pace del Mg. Santiago Vidal de la tesis de Doctorado en Ciencias de la Computación, SpIRIT: Smart Identification of Refactoring opportunITies, Fac. de Cs. Exactas UNCPBA. Resolución 208/09. Defendida Diciembre 2013.

Directora en conjunto con el Dr. Andres Diaz Pace del Ing. Alejandro Rago de la tesis de Maestría en Ingeniería de Sistemas, Tool support for identifying crosscutting concerns in use case specifications, Fac. de Cs. Exactas UNCPBA. Resolución . Defendida Marzo 2013.

13. 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.*

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

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

• 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 $25.000.

• Subsidio anual de CIC (Comisión de Investigaciones Científicas de la provincia de Buenos Aires) para investigadores CIC 2013. Suma de $6.000.
• 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. Monto 15.000 http://sites.google.com/site/profeprog/proyecto4
Co-directora del proyecto de investigación "Estrategias para la Integración y Conexión de Reglas de Negocio con Aspectos". (29/A242). Director: Dra. Sandra Casas. Integrantes: Docentes: Hector H.Reinaga - Juan Gabriel Enriquez - Marcela Costanzo - Daniel Gonzalez - Franco Herrera - Graciela Vidal - Héctor Sóto Peréz - Natalia Trejo. Universidad Nacional Patagonia Austral (UNPA), Unidad Académica Río Gallegos (UARG). 2010-2012. Monto 15.000 http://sites.google.com/site/profeprog/proyecto3

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

17. DISTINCIONES O PREMIOS OBTENIDOS EN EL PERIODO.
• Premio al mejor trabajo de doctorado de Santiago Vidal, bajo la dirección de Claudia Marcos y Andres Díaz Pace, en la First Latin-American School on Software Engineering (ELA-ES), Rio de Janeiro, Brasil, Julio 2013.

18. 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 los Proyectos de Investigación y Desarrollo correspondientes a la convocatoria 2013, de la Secretaria de Ciencia, Tecnología y Posgrado de la Universidad Tecnológica Nacional. Diciembre 2013. (3 días).
- Evaluadora de proyectos de LACCIR (Latin American and Caribbean Collaborative ICT Research): http://www.laccir.org/ 2012. (1 día)

19. 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)
- Métodos Agiles (MA) para el Desarrollo de Software curso de Maestría en Ingeniería de Software. Universidad Nacional Patagonia Austral. Setiembre 2013. (curso dictado por video conferencia 1 hrs. semanal durante el 2do cuatrimestre)
- Estrategias para mejorar la separación de concerns curso de Maestría en Ingeniería de Sistemas y Doctorado en Ciencias de la Computación. UNICEN, Fac. de Ciencias Exactas, Dpto. de Computación y Sistemas. (2 hrs. semanales durante el 1er cuatrimestre)
- Métodos Agiles (MA) para el Desarrollo de Software curso de Maestría en Ingeniería de Sistemas y Doctorado en Ciencias de la Computación. UNICEN, Fac. de Ciencias Exactas, Dpto. de Computación y Sistemas. (2 hrs. semanales durante el 1er cuatrimestre)
- Métodos Agiles (MA) para el Desarrollo de Software curso de Maestría en Ingeniería de Software. Universidad Nacional Patagonia Austral, dependencia Caleta Olivia UACO. Agosto 2012. Acuerdo 229/12. (curso intensivo de una semana)

20. 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

• Asesora de la Empresa it-Mentor. "http://www.it-mentor.com.ar/"Http://www.it-mentor.com.ar Desde 2004.

• Colaboradora en la definición de los alcances de las Calificaciones Profesionales de IBM desde 2004.

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) – UNCPBA - CIC, por la Facultad de Ciencias Exactas. Desde 2009 a la fecha, Resolución 259/09 (23/10/09), Resolución 233/12 (24/08/2012).

• Miembro del Consejo Editorial de la Universidad. 2005-2012. Resolución: 123/05. Tandil 24/06/05. Resolución 014/08, 22/02/2008. Resolucion 133/12, 18/05/2012

Integrante de Comités de Programa y Evaluación de Artículos

• Miembro del Editorial Advisory Board and List of Reviewers "Strategies for a Creative Future with Computer Science, Quality Design and Communicability". HCI Collection Blue Herons. Editor: Francisco V. Cipolla Ficarra. 2013 http://www.blueherons.net/home_en_15.html

• Miembro del Comité de Programa de la Conferencia IADIS Ibero-Americana WWW/Internet 2013. 21-23 Noviembre, Porto Alegre Brazil.

• Miembro del Editorial Advisory Board and List of Reviewers "Computer Engineering and Innovations in Education for Virtual Learning Environments, Intelligent Systems and Communicability: Multimedia Mobile Technologies, Experiences in Research and Quality Educational Trends" Informatics and Emerging Excellence in Education Collection Blue Herons. Editor: Francisco V. Cipolla Ficarra. 2012. http://www.blueherons.net/home_en_6.html

• Miembro del Comité de Programa CoNaIISI 2013 1er COngreso Nacional de Ingenieria Informática/Sistemas de Información, 21 y 22 de Noviembre de 2013. Universidad Tecnológica Nacional, Facultad Regional Córdoba http://conaiisi.frc.utn.edu.ar/.

• Miembro del Scientific Comittee del Third International Workshop Human-Computer Interaction, Tourism and Cultural Heritage (HCITOCH 2012): Strategies for a Creative Future with Computer Science, Quality Design and Communicability. Venice, Italy. September 27 – 28, 2012. http://www.alaipo.com/workshop-2012/workshop_HCITOCH_2012.html

• Miembro del Comité de Programa de la Conferencia IADIS Ibero-Americana WWW/Internet 2012. 12-20 Octubre Madrid. España Octubre 2012.

• Miembro del Editorial Advisory Board and List of Reviewers Emerging Software for Interactive Interfaces, Database, Computer Graphics and Animation: Pixels and the New Excellence in Communicability, Cloud Computing and Augmented Reality" Pixels Collection. Blue Herons. Editor: Francisco V. Cipolla Ficarra. 2012 http://www.blueherons.net/home_en_9.html

• Miembro del Editorial Advisory Board and List of Reviewers New Horizons in Creative Open Software, Multimedia, Human Factors and Software Engineering" Human-Computer Interaction Collection. Blue Herons. Editor: Francisco V. Cipolla Ficarra. 2012. http://www.blueherons.net/home_en_7.html

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 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 2011 - 2014 . Resolución 112/11.

Actividades de Evaluación en Postgrado

• Evaluador del Examen de Calificación del Ing. Alejandro Corbellini titulado Bases de Datos NoSQL distribuidos y persistentes, como requisito previo para la obtención del título de Doctor en Ciencias de la Computación, Diciembre 2013. Resolución Consejo Académico:410/13. Reunión de Consejo Académico 13/12/2013.
• Evaluador del Examen de Calificación de la Ing. Elina Pacini Naumovich titulado Schedulers basados en Swarm Intelligence para Experimentos de Barridos en Entornos Distribuidos, como requisito previo para la obtención del título de Doctor en Ciencias de la Computación, Marzo 2012. Resolución Consejo Académico:052/12. Reunión de Consejo Académico 23/03/2012.

Miembro de Jurado de Tesis de Posgrado

• Jurado suplente de la tesis para el Doctorado en Ciencias de la Computación de la Mg.Ingrid Christensen. SocialGR: Influencia Social en Sistemas de Recomendación a Grupos Heterogéneos. Diciembre 2013. Resolución Consejo Académica 385/13.
• Jurado suplente de la tesis para el Doctorado en Ciencias de la Computación del Ing David Monge. Un enfoque de planificación para la ejecución eficiente de flujes de trabajo científicos en sitios grid. Diciembre 2013. Resolución Consejo Académica 368/13.
• Jurado de la tesis para el Doctorado en Ciencias de la Computación del Ing. Juan Manuel Rodriguez. Malas prácticas en el desarrollo de servicios Web que dificultan su descubirmiento. Agosto 2012. Resolución Consejo Académica 203/2012.

Dirección de Trabajos Finales de Grado
Finalizados

• Directora en conjunto con Vanesa Dell Acqua del trabajo final Gestionando multiples proyectos con Kanban de los alumnos Julian Arocena y Esteban Roasio correspondiente a la carrera Ingeniería de Sistemas. Diciembre 2013.
• Directora en conjunto con Alejandro Rago del trabajo final REUSE: Un enfoque semi-automático para la refactorización de casos de uso de los alumnos Maria Paula Frade y Miguel Tomás Ruival correspondiente a la carrera Ingeniería de Sistemas. Octubre 2013.
• Co-directora en conjunto con Santiago Vidal del trabajo final Análisis de la Evolución de Sistemas Utilizando Algoritmos Financieros de los alumnos Juan Francisco Hurtado y Franco Sabadini correspondiente a la carrera Ingeniería de Sistemas. Setiembre 2013.

- Directora en conjunto con Esteban Abait del trabajo final Análisis de Técnicas de Aspect Mining Aplicadas en un Sistema de Producción de los alumnos Sebastian Bisbal y Juan Pablo Cassano correspondiente a la carrera Ingeniería de Sistemas. Junio 2012.
- Directora en conjunto con Esteban Abait del trabajo final Aspect Mining mediante Técnicas de FCA de los alumnos Cristian Vitale e Ivan Dietz correspondiente a la carrera Ingeniería de Sistemas. Mayo 2012.
- Directora en conjunto con Esteban Abait del trabajo final Una Herramienta de Evaluación de Métricas para los Paradigmas de Aspectos y Objetos (MTOA) de los alumnos José Badino y Juan Bruno correspondiente a la carrera Ingeniería de Sistemas. Abril 2012.

21. 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 la Refactorización de Sistemas

1. Mantenimiento de Sistemas

Al abordar tareas como el mantenimiento de sistemas legados se encuentra el problema de que éstos presentan código orientado a objetos que suele ser demasiado extenso y complejo, y por ende difícil de adaptar a los cambios de requerimientos [5]. En general, debido a que los requerimientos evolucionan [6] el sistema ha sido cambiado y actualizado para poder cumplir con los nuevos requerimientos. La consecuencia de la evolución es que los cambios son inevitables. Estos cambios pueden estar motivados por modificaciones en los requerimientos del usuario (por ejemplo extensiones, modificaciones, etc.), corrección de errores, cambios en los formatos de datos (nuevos tipos de divisas, códigos postales, números telefónicos, nuevos estándares, etc.), cambios en el hardware, mejoras de eficiencia, etc. Por esta razón, ocurre que será necesario un mantenimiento del software para prevenir su "envejecimiento" [1].

Debido a los continuos cambios del sistema, varios problemas pueden aparecer por ejemplo, los documentos quedan obsoletos y no concuerdan con el código actual de la aplicación; debido a los rápidos tiempos de entrega del sistema con la nueva funcionalidad el código de la aplicación no posee los estándares de programación acordados durante el desarrollo y además posee problemas de mala programación, denominados *code smells []*. Los code smells son útiles para identificar problemas estructurales de un sistema que se relacionan con problemas de modificabilidad [2]. De esta manera, un code smell actúa como un anti-patrón que indica el código que debe ser mejorado. Por esta razón, los code smells son una especie de deuda técnica que debe ser tenida en cuenta.

Cada code smell puede afectar a varios elementos (por ejemplo, paquetes, clases, métodos) de un sistema. Algunos de los síntomas utilizados por los code smells son: código duplicado, métodos o clases muy grandes, larga lista de parámetros o violaciones en el encapsulamiento de una clase, entre otros. Los catálogos más populares de code smell son los presentados por Fowler [2] y por Lanza y Marinescu [4]. A través de la utilización de los code smells, varios problemas tales como los métodos con alta complejidad o dependencias excesivas entre las clases se pueden solucionar. Cada code smell puede afectar a diferentes elementos del sistema como métodos, clases o paquetes. De esta manera, es posible identificar el elemento principal en el que el smell se implementa y otros elementos que podrían ser afectados indirectamente por el code smell. Es decir, un análisis del impacto del cambio debe hacerse para comprender las consecuencias del smell. Por ejemplo, se puede identificar la concentración de smells en los elementos del sistema, tales como paquetes. Debido a que un code smell puede ser distribuido en más de una clase, este análisis puede ser realizado mediante el uso de la clase principal del smell.

2. Refactorización de Código

La incorporación de los nuevos requerimientos implica introducir código nuevo o reestructurar código existente, el objetivo es que el código final sea mantenible y esté bien modularizado. Para lograr dicho objetivo se utiliza la técnica de *refactoring* [2]. La refactorización es el proceso mediante el cual se hacen cambios en un sistema de software de forma tal que se mejore la estructura interna del mismo sin alterar el comportamiento externo. Este proceso busca abordar el problema de la evolución del software mejorando el código fuente de tal forma que el esfuerzo y el costo al realizar modificaciones sobre el sistema sea menor (con respecto a un sistema sin estos cambios) cuando cambia el ambiente o los requerimientos.

El concepto de refactorización está íntimamente ligado a la programación orientada a objetos. Utilizando las técnicas propuestas para la reestructuración de código se mejoran la legibilidad y el diseño del código haciéndolo más comprensible y por ende facilitando su mantenimiento. Algunas de las actividades que se realizan con este objetivo son: reducir el código duplicado, mejorar la cohesión, reducir el acoplamiento, mejorar atributos de calidad como la facilidad de comprender el sistema, flexibilidad, mantenimiento, abstracción eficiencia, etc.

La refactorización se puede utilizar para resolver los code smells y, de esta manera, para solucionar la deuda técnica que los smells generan. En los catálogos de code smells, por lo general, se propone para cada smell una refactorización o un grupo de refactorizaciones (de acuerdo a la complejidad del smell) que pueden ayudar a resolver el problema. Por ejemplo, la God Class [2] identifica la situación en la que una clase centraliza la inteligencia del sistema (o un subsistema). En este caso, la estrategia sugerida por la refactorización es extraer los grupos de métodos relacionados en nuevas clases.

3. Problemática de la Refactorización de Code Smells

Un problema importante de refactorización de code smells es que las herramientas existentes que detectan smells, generalmente, listan un gran número de smells. Por ejemplo, después de analizar 9 tipos de código de smells (que se describen en [4]) en SweetHome3D6, una aplicación de código abierto LOC 84K , se encontraron 787 smells.

La refactorización de todo el código de los smells es ideal, pero también es una tarea que consume tiempo y esfuerzo. En estas situaciones, el esfuerzo y costo de resolver todos los code smells es muy alto. Por otra parte, la refactorización de algunos code smells no puede ser urgente. Por ejemplo, la refactorización del code smell en una clase sin cambios desde su implementación inicial (y no se espera que sea modificado en el futuro) puede tener una baja prioridad en comparación con un code smell en una clase que recibió modificaciones en las últimas 10 versiones. Además, dado que los smells pueden conducir a graves consecuencias en términos de mantenimiento y adaptabilidad del sistema, la resolución de los smells relacionados con problemas claves de diseño pueden ser urgente. Relacionado a la prioridad de los elementos del sistema a ser refactorizados, algunos investigadores han sugerido que aquellos elementos cuyo código sufrió muchos cambios en el pasado, es más probable que se modifique en el futuro que los que no se cambiaron [3, 8]. Por ejemplo, en SweetHome3D el 85 % de los code smells detectados fueron modificados sólo en una o dos versiones de las últimas 25 versiones. Es decir, el esfuerzo de refactorización se centrará en el 15% restante de los code smells. Además, para solucionar con éxito la deuda técnica que genera el smell, la información de la arquitectura debe tenerse en cuenta durante el proceso de refactorización. Mediante el análisis de la documentación de la arquitectura el desarrollador podría saber que smell afecta con mayor intensidad a la arquitectura del sistema [6]. De esta manera, refactorizando los code smells que afectan más al diseño ayuda a no degradar la arquitectura del sistema.

Por otro lado, la determinación de la forma en que un code smell debe ser reprogramado no es siempre sencillo. En algunos casos, es necesaria la aplicación de varias refactorizaciones de código para garantizar que el code smell se elimina por completo. Además, para la aplicación de una refactorización, por lo general, es necesario aplicar refactorizaciones adicionales (algunas de las cuales podrían ser refactorizaciones manuales). Por ejemplo, para varios code smells se sugiere la aplicación de Move Method [2]. Sin embargo, si el método a ser movido utiliza una variable de instancia que sólo se utiliza en el método, la variable debe ser primero movida usando Move Data y luego el Move

Method puede aplicarse [2]. Otro ejemplo claro es la resolución de la God Class, los catálogos sugieren construir varias clases, sin embargo no se describe qué técnica se debe seguir para identificar la cantidad ideal de clases que se deben construir y la estrategia para reasignar métodos y atributos a las nuevas clases. Por esta razón, la reprogramación de un code smell depende del contexto en el cual se encuentra dicho problema.

4. Plan de Trabajo para el Período 2014-2016

El objetivo principal de la investigación para el próximo período está orientado a la mejora de las estrategias de identificación de los problemas más críticos de un sistema y a proponer estrategias para la aplicación de un refactorización. Esta nueva funcionalidad será incorporada a la herramienta SpIRIT desarrollada durante el período 2012-2014. Las actividades a desarrollar se detallan a continuación:

- Analizar la incorporación de nuevas estrategias a SpIRIT que permitan mejorar la identificación de los problemas más importantes del sistema. SpIRIT realiza un ranking teniendo en cuenta la hostoria de las componentes, los code smells y los escenarios de modificabilidad. Se está analizando la posibilidad de incorporar heurísticas que permitan analizar la conformidad arquitectónica del sistema. Por otra parte, otro tipo de escenarios podrían ser incorporados.
- 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.
- Definición de estrategias de resolución de los code smells. Se definirán diferentes alternativas de cada code smells teniendo en cuenta los beneficios que se obtendrán en su aplicación y si podría introducir nuevos code smells.
- Integración de las diferentes estrategias a SpIRIT. Integrando las nuevas estrategias de identificación de los problemas más críticos del sistema y las alternativas de solución de cada code smell es posible tener una herramiento con el proceso de refactorización completo.
- 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.

5. Referencias

1. Banker, Rajiv D.; Datar, Srikant M.; Kemerer, Chris F. and Zweig Dani. Software complexity and maintenance costs. Commun. ACM, 36(11):81–94, 1993.
2. Fowler, Martin. Refactoring: improving the design of existing code. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.
3. Gîrba, Tudor; Ducasse, Stéphane and Lanza, Michele. Yesterday"s weather: Guiding early reverse engineering efforts by summarizing the evolution of changes. In ICSM, pages 40–49. IEEE Computer Society, 2004.
4. Lanza, Michele and Marinescu, Radu. Object-Oriented Metrics in Practice - Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems. Springer, 2006.
5. Lehman , M. M. and Belady, L. A. editors. Program evolution: processes of software change. Academic Press Professional, Inc., San Diego, CA, USA, 1985.
6. Macia, Isela; Arcoverde, Roberta; Garcia, Alessandro; Chavez, Christina and von StaaArndt. On the relevance of code anomalies for identifying architecture degradation symptoms. In CSMR, 2012.

7. Parnas, David. Software aging. In ICSE '94: Proceedings of the 16th international conference on Software engineering, pages 279–287, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

8. Mens, Tom and Demeyer, Serge. Future trends in software evolution metrics. In Proceedings of the 4th International Workshop on Principles of Software Evolution, IWPSE '01, pages 83–86, New York, NY, USA, 2001. ACM.

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.