CARRERA DEL INVESTIGADOR CIENTÍFICO Y TECNOLÓGICO

Informe Científico[1]

PERIODO [2]: 01/01/2016 al 31/12/2017

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"):

2. TEMA DE INVESTIGACION

ESTRATEGIAS PARA MEJORAR EL MANTENIMIENTO DE SISTEMAS

PALABRAS CLAVE (HASTA 3) Code Smells Refactoring

Mantenimiento de software

3. DATOS RELATIVOS A INGRESO Y PROMOCIONES EN LA CARRERA

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

ACTUAL: Categoría: INDEPENDIENTE desde fecha: 06/12/2016

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: 4385663

Cargo que ocupa: PROFESOR ASOCIADO

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

Apellido y Nombres:

Dirección Particular: Calle: Nº:

Localidad: CP: Tel:

---

[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 2017 deberá informar sobre la actividad del período 1°-01-2015 al 31-12-206, para las presentaciones bianuales. Para las presentaciones anuales será el año calendario anterior.

Dirección electrónica:

....

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 y el Feature Envy 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 mismos. 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 y el Feature Envy. Esta nueva herramienta, denominada Bandago, toma como entrada el código con problemas y el tipo code smell y genera diferentes soluciones de refactorización para el code smell a resolver. Dependiendo del code smell que se necesita resolver, la herramienta presenta un conjunto de soluciones las cuales son comparadas a

partir de un conjunto de métricas. Las diferentes soluciones comparadas son presentadas al desarrollador el cuál puede hacer un análisis de las diferentes alternativas y seleccionar aquella alternativa que más se adapte a sus necesidades.

JavaScript es uno de los lenguajes de programación más potentes e importantes en la actualidad. Algunas de sus ventajas con respecto a otros lenguajes son su practicidad, utilidad y disponibilidad en cualquier navegador web. JavaScript ha evolucionado en relación a la sintaxis, funcionalidad y características en general, pero hasta el momento no se ha encontrado un proceso o funcionalidad para abordar problemas de calidad de los sistemas de software relacionados con la mantenibilidad. Por lo tanto, la detección de code smells en el lenguaje JavaScript es una problemática que aún no presenta demasiadas soluciones. Por esta razón, se ha comenzado a analizar las características del lenguaje JavaScript para identificar estrategias de separación de concerns y en la identificación de code smells para dicho lenguaje.

Por otro lado, JavaScript (JS) es el lenguaje principal para el desarrollo de aplicaciones Web moderno, ya que da soporte a la codificación del comportamiento de las páginas Web. Para que los navegadores puedan ejecutar y mostrar una página Web, es necesario descargar muchos archivos por lo que afecta negativamente al desempeño de la aplicación. Por esta razón, se ha comenzado a analizar alternativas de optimización al tamaño de las páginas web.

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. Asimismo, para cada publicación deberá indicar si se encuentra depositada en el repositorio institucional CIC-Digital.*

1. ***Using Semantic Roles to Improve Text Classification in the Requirements Domain**.* Alejandro Rago, Claudia Marcos, Andrés Díaz Pace. Language Resources and Evaluation. November 2017. "https://oi.org/10.1007/s10579-017-9406-7 Springer Netherlands. Print ISSN1574-020X, Online ISSN1574-0218

   ***Abstract**:* Abstract Engineering activities often produce considerable documentation as a by-product of the development process. Due to their complexity, technical analysts can benefit from text processing techniques able to identify concepts of interest and analyze deficiencies of the documents in an automated fashion. In practice, text sentences from the documentation are usually transformed to a vector space model, which is suitable for traditional machine learning clasifiers. However, such transformations suffer from problems of synonyms and ambiguity that cause classification mistakes.

For alleviating these problems, there has been a growing interest in the semantic enrichment of text. Unfortunately, using general-purpose thesaurus and encyclopedias to enrich technical documents belonging to a given domain (e.g. requirements engineering) often introduces noise and does not improve classification. In this work, we aim at boosting text classification by exploiting information about semantic roles. We have explored this approach when building a multi-label classifier for identifying special concepts, called domain actions, in textual software requirements. After evaluating various combinations of semantic roles and text classification algorithms, we found that this kind of semantically-enriched data leads to improvements of up to 18% in both precision and recall, when compared to non-enriched data. Our enrichment strategy based on semantic roles also allowed classifiers to reach acceptable accuracy levels with small training sets. Moreover, semantic roles outperformed Wikipedia- and WordNET-based enrichments, which failed to boost requirements classification with several techniques. These results drove the development of two requirements tools, which we successfully applied in the processing of textual use cases

Grado de participación: Alto, ya que estuvo directamente involucrada en definir las estrategias de enriquecimiento semántico 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.

2. ***Una herramienta para priorizar code smells en desarrollo distribuido (A tool to prioritize code smells in distributed development)***. Hernan Vázquez; Claudia Marcos; Santiago Vidal; Jorge Andrés Diaz Pace. Latin American Transaction, (Revista IEEE América Latina) Volume: 15, Issue: 10, Date: Oct. 2017 http://www.ewh.ieee.org/reg/9/etrans/ieee/issues/vol15/vol15issue10Oct.2017/Vol15issue10Oct.2017TLA.htm"http://www.ewh.ieee.org/reg/9/etrans/ieee/issues/vol15/vol15issue10Oct.2017/Vol15issue10Oct.2017TLA.htm Publisher: IEEE Region 9. ISSN: 1548-0992.

Abstract. A code smell is a symptom in the source code that helps to identify a design problem. Several tools for detecting and ranking code smells according to their criticality to the system have been developed. However, existing works assume a centralized development approach, which does not consider systems being developed in a distributed fashion. The main problem in a distributed group of developers is that a tool cannot always ensure a global vision of (smells of) the system, and thus inconsistencies among the rankings provided by each developer are likely to happen. These inconsistencies often cause unnecessary refactorings and might not focus the whole team on the critical smells system-wide. Along this line, this work proposes a multi-agent tool, called D-JSpIRIT, which helps individual developers to reach a consensus on their smell rankings by means of distributed optimization techniques.

Grado de participación: Alto, ya que estuvo directamente involucrada en definir las estrategias de ranking y en el sistema multi-agente y en la escritura del paper. El trabajo es una parte del resultado de investigación de Hernan Vezquez donde Claudia Marcos fue la directora.

3. ***Over-exposed Classes in Java*: *An Empirical Study***. Santiago A. Vidal, Alexandre Bergel, J. Andrés Díaz-Pace and Claudia Marcos. Computer Languages Systems & Structures · Volume 46, November 2016, Pages 1-19 April 2016 Impact Factor: 0.44 · DOI: 10.1016/j.cl.2016.04.001. ISSN: 1477-8424

   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.

   Grado de participación: Medio, colaboró en la definición de problema y en la escritura del paper.

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. Printed version May 2016. Vol 15 Nro 2 pp 579-603; Online Version pp1619-1374 August 2014. DOI 10.1007/s10270-014-0431-3.

   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 definir las estrategias de enriquecimiento semántico 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. ***Assisting Requirements Analysts to Find Latent Concerns with REAssistant***. Alejandro Rago, Claudia Marcos, Andrés Díaz Pace. Automated Software Engineering. ISSN: 0928-8910 (Print) 1573-7535 (Online) Vol 23 Nro 2 pag 219-252 March 2016. (DOI: 10.1007/s10515-014-0156-0)

   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.

   Grado de participación: Alto, ya que estuvo directamente involucrada la extensión a UMIA 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.

6. ***Recuperación de Trazas entre Documentos de Requerimientos y Arquitectura*** A. Rago, C. Marcos, and A. Diaz-Pace. In: Proceedings of the 4th Congreso Argentino de Ingeniería Informatica y Sistemas de Información (CoNaIISI '16). Track: Ingeniería de Sistemas y de Software. RIISIC. Salta, Argentina, Nov. 2016.

   Resumen. Para satisfacer las necesidades de los stakeholders y adecuarse a las demandas del mercado, los desarrolladores de software deben tener en

cuenta diversos atributos de calidad y asegurar su cumplimiento durante el desarrollo de un sistema. En este contexto, mantener relaciones de trazabilidad para verificar que los atributos de calidad asociados a ciertos requerimientos han sido tenidos en cuenta en el diseño arquitectónico (y viceversa) es fundamental. Desafortunadamente, establecer y mantener manualmente las trazas entre los artefactos de un sistema es una tarea compleja y tediosa. Algunos investigadores han desarrollado herramientas para identificar trazas de forma automática, pero las mismas no han sido aplicadas a documentos extensos como son los requerimientos y la arquitectura. En este trabajo se presenta una técnica para identificar trazas entre requerimientos y arquitectura basada en técnicas de procesamiento de lenguaje natural. La técnica filtra información relevante de la documentación para las trazas, haciendo hincapié en los atributos de calidad. Luego, se utiliza un algoritmo de Latent Semantic Analysis (LSA) para detectar trazas al nivel de oraciones. La técnica desarrollada fue evaluada en tres casos de estudio con resultados alenta- dores. Específicamente, se obtuvieron mejoras de desempeño entre 10 % y 40 % al recuperar las trazas.

Grado de participación: Alto, ya que estuvo directamente involucrada las discusiones del trabajo 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.

7. ***Análisis de Dependencias entre Refactorings para Solucionar Code Smells*** Claudia Marcos, Santiago Vidal, J. Andrés Diaz Pace In: Proceedings of the 4th Congreso Argentino de Ingeniería Informatica y Sistemas de Información (CoNaIISI ’16). Track: Ingeniería de Sistemas y de Software. RIISIC. Salta, Argentina, Nov. 2016.

Abstract. Los code smells son síntomas en el código fuente que pueden revelar problemas de diseño. Para poder solucionar un smell deben aplicarse un conjunto de refactorings que permitan restructurar el sistema. Sin embargo, al aplicar un conjunto de refactorings en un orden determinado, pueden surgir problemas que impiden que éstos se apliquen. Por ejemplo, porque un refactoring que depende de una reestructuración realizada por otro refactoring que aún no fue aplicado, o porque un refactoring referencia un artefacto del sistema que fue modificado por un refactoring aplicado anteriormente. Por estos motivos, para aplicar un conjunto de refactorings, se deben analizar las dependencias que existen entre estos para poder establecer el orden de aplicación. En esta línea, este trabajo presenta una herramienta que identifica y soluciona los conflictos originados por dependencias entre refactorings para luego aplicar automáticamente los mismos. Los resultados, si bien son preliminares, indican que este enfoque permite identificar y solucionar un alto porcentaje de conflictos.

***Grado de participación*:** Alto, ya que estuvo directamente involucrada las discusiones del trabajo 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.

8. ***TeXTracT: a Web-based Tool for Building NLP-enabled Applications*.** A. Rago, Facundo M. Ramos, Juan I. Velez, Santiago Vidal, Claudia Marcos, J. Andres Diaz-Pace 1. In: Proceedings of the XII Argentine Symposium on Software Engineering, held at Jornadas Argentinas de Computación e Investigación Operativa (JAIIO '16). Buenos Aires, Argentina: SADIO, Sept. 2016, pp. 123–134. ISSN: 1850-2792.

   Abstract. Over the last few years, the software industry has showed an increasing interest for applications with Natural Language Processing (NLP) capabilities. Several cloud-based solutions have emerged with the purpose of simplifying and streamlining the integration of NLP techniques via Web services. These NLP techniques cover tasks such as language detection, entity recognition, sentiment analysis, classification, among others. However, the services provided are not always as extensible and configurable as a developer may want, preventing their use in industry-grade developments and limiting their adoption in specialized domains (e.g., for analyzing technical documentation). In this context, we have developed a tool called TeXTracT that is designed to be composable, extensible, configurable and accessible. In our tool, NLP techniques can be accessed independently and orchestrated in a pipeline via RESTful Web services. Moreover, the architecture supports the setup and de- ployment of NLP techniques on demand. The NLP infrastructure is built upon the UIMA framework, which defines communication protocols and uniform service interfaces for text analysis modules. TeXTracT has been evaluated in two case-studies to assess its pros and cons.

   ***Grado de participación*:** Alto, ya que el trabajo es el resultado de las actividades desarrolladas por dos de los autores como parte de su trabajo final de grado del cual Claudia Marcos fue una de los directores.

9. ***Opportunities for Analyzing Hardware Specifications with NLP Techniques*.** A. Rago, C. Marcos, and A. Diaz-Pace. In: 3rd Workshop on Design Automation for Understanding Hardware Designs (DUHDe '16). Design, Automation, Test in Europe Conference, and Exhibition (DATE'16). Dresden, Germany, Mar. 2016.

   Abstact. Hardware design is a mature discipline that heavily relies on complex models to create 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 enable automated verifications and consistency checks. 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 applications we developed for inspecting Software Engineering documentation and discuss their potential applications to automated hardware design.

Grado de participación: Alto, ya que estuvo directamente involucrada las discusiones del trabajo 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.

10. ***Identifying Architectural Problems through Prioritization of Code Smells.*** Santiago Vidal, Everton Guimaraes, Willian Oizumi, Alessandro Garcia, Andrés Díaz Pace, and Claudia Marcos. In SBCARS 2016 (X Brazilian Symposium on Components, Architectures, and Reuse). Maringa, Brazil, 2016.

Astract. Architectural problems constantly affect evolving software projects. When not properly addressed, those problems an hinder the longevity of a software system. Studies have revealed that a range of architectural problems are reflected in source code through two or more code smells. However, a software project often contains thousands of code smells and many of them have no relation to architectural problems. Thus, developers may feel discouraged to identify architectural problems if they are not equiped with means to focus their attention in a reduced set of locations in their system to start with. However, state-of-the-art techniques fall short in assisting developers in the prioritization of code smells that are likely to indicate architectural problems in a program. As a consequence, developers struggle to effectively focus on (groups of) smells that are architecturally relevant, i.e., smells that contribute to a critical design problem. This work presents and evaluates a suite of criteria for prioritizing groups of code smells 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 23 versions of 4 systems, analyzing their effectiveness for spoting locations of architectural problems in the source code. The results provide evidence that one of the proposed criteria helped to correctly prioritize more than 80 (locations of) architectural problems, alleviating tedious manual inspections of the source code vis-a-vis with the architecture. This prioritization criteria would have helped developers to

discard at least 500 code smells having no relation to architectural problems in the analyzed systems.

Grado de participación: Alto, ya que estuvo directamente involucrada las discusiones del trabajo y en la escritura del paper. El trabajo es una parte de las actividades de colaboración entre el grupo de investigadores de diversas universidades de Brasil y los investigadores del ISISTAN UNICEN.

11. ***Chasing Critical Code Anomalies with JSpIRIT*** Tool demo. Santiago Vidal, J. Andres Diaz-Pace and Claudia Marcos SATURN 2016. San Diego, CA, May 2-5, 2016.

    Astract. 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 Identi cation 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 exible enough to include information from past system versions, modi ability 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 con gurations. We will present the key tool features and discuss project experiences in which JSpIRIT was useful for diagnosing the system “health” and planning for efactorings.

    Grado de participación: Alto, la demo describe la herramienta desarrollada por Santiago Vidal como parte de su doctorado donde Claudia Marcos fue una de las directoras.

12. ***On the Criteria for Prioritizing Code Anomalies to Identify Architectural Problems.*** Santiago Vidal, Everton Guimaraes, Willian Oizumi, Alessandro Garcia, J. Andrés Díaz Pace and Claudia Marcos The 31st ACM Symposium On Applied Computing SAC 1812-1814. ACM New York 2016 April ISBN: 978-1-4503-3739-7, Pisa Italy.

    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 soft ware 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 directamente involucrada las discusiones del trabajo y en la escritura del paper. El trabajo es una parte de las actividades de colaboración entre el grupo de investigadores de diversas universidades de Brasil y los investigadores del ISISTAN UNICEN.

8.2 TRABAJOS EN PRENSA Y/O ACEPTADOS PARA SU PUBLICACIÓN*. Deb e 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.*

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.*

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. Indicar en cada caso si se encuentra depositado en el repositorio institucional CIC-Digital.

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 TRANSFERIBLES, 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 con Temperies S.A., Q4TEch S.A. y Lider File S.A. sin embargo la relación y las actividades no han sido tan estrechas como en el período anterior.

Se está desarrollando 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 el método formalizado en términos de sus artefactos, procesos, y guías de adaptación, 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 está en desarrollo se ha definido el proceso y se ha comenzado a validar el mismo.

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.*

Nicolas Mosca (BeeReal) nmosca@beerealit.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 HYPERLINK "http://ccc.alumnos.exa.unicen.edu.ar/apuntes-2011"http://ccc.alumnos.exa.unicen.edu.ar/apuntes-2011 HYPERLINK ""

11.2 DIVULGACIÓN

En cada caso indicar si se encuentran depositados en el repositorio institucional CIC-Digital.

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. Andres Diaz Pace del Dr. Santiago Vidal como Investigador Asistente de CONICET. Nro. Resolución designación 955/15 Fecha: 01/04/2015
- Codirectora en conjunto con el Dr. Andres Diaz Pace del Dr. Alejandro Rago como Investigador Asistente de CONICET. Nro. Resolución designación 3143/13-09-2016, Fecha: 1/09/2016

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.*

- Co directora en conjunto con el Dr. Andres Diaz Pace del Ing. Hernan 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.

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.*

- Directora del proyecto Desarrollo de un prototipo de servicio web para la gestión inteligente de módulos y funciones en archivos JavaScript. Fundación Sadosky, proyecto colaboración Universidad-Empresa. 2017-2018. Monto: 100.000$.
- Directora del proyecto Optimización Web JavaScript. Vinculación Tecnológica (SPU). Universidad Agregando Valor VT38-UNICEN10191, 2017-2018. Monto: 90.000
- Subsidio Institucional para investigadores CIC (Comisión de Investigaciones Científicas de la provincia de Buenos Aires) 2017. Suma 16.000 $.
- Subsidio Institucional para investigadores CIC (Comisión de Investigaciones Científicas de la provincia de Buenos Aires) 2016. Suma 11.000 $.
- Co-directora del proyecto de investigación tipo 1 código 29/A359. Estrategias para optimizar / mejorar el desarrollo de frameworks orientado a aspectos. Directora: Casas
- Sandra del 1/1/2016 al 31/12/2017. Universidad Nacional Patagonia Austral (UNPA), Unidad Académica Río Gallegos (UARG).

Participación en Proyectos de Investigación

- Integrante del proyecto de incentivos Plataforma de Servicios para el Desarrollo de Software de Ciudades Inteligentes 03/C288 – Director: Dr. Andrés Diaz Pace – Fecha de acreditación: 01/01/2018, fecha de finalización: 31/12/2020
- Integrante del proyecto “Detection strategies based on Software Metrics for Multitier JavaScript” Regional Program STIC-AmSud 2017 Project Proposal (Research – Innovation). Proyecto entre el MINCYT: Dirección Nacional de Coordinación e Integración Institucional, Argentina; CONICYT: Departamento de Relaciones Internacionales, Chile; INRIA: Direction des Relations Internationales, Francia. Director: Alexandre Bergel, coordinador Alexandre Bergel, Chile; Santiago Vidal, Argentina; Manuel Serrano, Francia. Integrantes: Andrez Díaz Pace, Hernan Ceferino Vazquez, Claudia Marcos, Argentina. Alison Fernandez, Juraj Kubelka, Diego Orellana, Chile; Tamara Rezk, Colin Vidal, Francis Some, Francia. Total subsidio: Euros 28780. Enero 2018/Diciembre 2019.

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.

- Premio al mejor trabajo de investigación *Identifying architectural problems through prioritization of code smells* Santiago Vidal, Andrés Díaz-Pace, Claudia Marcos en 10th Brazilian Symposium on Software Components, Architectures, and Reuse (SBCARS 2016) VII Brazilian Congress on Software en Maringa, Brasil entre los días 19 y 23 de septiembre.
- Premio al mejor paper. *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 Proyecto de Investigación PIDIN Secretaría de Investigación y Posgrado de la Facultad de Ciencia y Tecnología de la Universidad Autónoma de Entre Ríos, 2017.
- Evaluadora de Proyectos de Investigación del área COMPUTACIÓN para la acreditación de los proyectos presentados en la convocatoria 2018 de la Universidad Nacional de la Patagonia Austral. Proyectos: 29/A396 - Evaluación de desempeño del protocolo TCP en topologías mixtas cabledas-inalámbricas. Mg. Carlos Talay, 29/B222 - Diseño y Evaluación de Experiencia de Usuario para Multi-Dispositivos. Dra. Adriana Martin, 29/B225 - Soluciones inteligentes para el desarrollo urbano sostenible. Mg. Norma Villagra, 29/C071 - Las ciencias de la computación en la educación inicial, un enfoque creativo. Ing. Silvia Rivadeneira.
- Evaluadora de Proyectos de Investigación de la Universidad Tecnológica Nacional, 2017.
- Evaluador de proyectos de investigación científica y tecnológica (identificados como PI) y proyectos de desarrollo y transferencia de tecnología (identificados como PI-DTT), de la Universidad Nacional de Río Negro. En el marco de la Convocatoria TRIENAL PI-2016 Título del proyecto: Herramientas Informáticas para el Desarrollo de Servicios Digitales Innovadores para Comunidades Urbanas y Ruralesen el Marco de Ciudades y Regiones Inteligentes (HI-DeSeDI) Código: 80020160900019RN Postulante: Estevez, Elsa Clara
- Evaluadora externa de Proyecto de Investigación PIDIN Secretaría de Investigación y Posgrado de la Facultad de Ciencia y Tecnología de la Universidad Autónoma de Entre Ríos, 2016.
- Evaluadora externa de proyectos de investigación y desarrollo, convocatoria 2016, de la Secretaría de Ciencia, Tecnología y Posgrado de la Universidad Tecnológica Nacional.

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

- Metodologias de Desarrollo de Software I. Carrera Ingeniería de Software, Facultad de Ciencias Exactas, UNCPBA, Tandil, (3 hrs. semanales durante el 1er cuatrimestre).
- Introducción a las Metodologías de Desarrollo de Software. Carrera TUDAI (Tecnicatura Universitaria en Desarrollo de Aplicaciones Informáticas) .Facultad de Ciencias Exactas, UNCPBA, Tandil, (2 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.*

Investigador invitado de ISISTAN-CONICET Instituto de Sistemas Tandil, Fac. Cs. Exactas, UNCPBA – Desde 2016 hasta la fecha.

Actividades de Transferencia

- Acuerdo Específico de Colaboración Científica entre la Empresa Temperies S.A y la U.N.C.P.B.A. Unidad Ejecutora: Facultad de Ciencias Exactas. Resolución 6342 27/10/2016.
- 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. HYPERLINK "http://www.it-mentor.com.ar/"Http://www. HYPERLINK "http://www.it-mentor.com.ar/"it-mentor.com.ar Desde 2004.
- Colaboradora en la definición de los alcances de las Calificaciones Profesionales de IBM desde 2004.

Integrante de Comités de Programa y Evaluación de Artículos

- Integrante del Comité Académico del CONAIISI 6to Congreso Nacional de Ingeniería en Informatica/ Sistemas de Información. Mar del Plata Noviembre 2018.
- Evaluadora del Journal of the Brazilian Computer Society, 2017.
- Integrante del Comité Académico del Eleventh International Conference on Advanced Engineering Computing and Applications in Sciences ADVCOMP 2017 November 12 - 16, 2017 - Barcelona, Spain
- Evaluadora del Journal of the Brazilian Computer Society, 2016.

- Integrante del Comité Académico del CONAIISI 4to Congreso Nacional de Ingeniería en Informatica/ Sistemas de Información. Salta Noviembre 2016.
- Integrante del Comité Académico al 14th Ibero-Americana WWW/Internet 2016 Conference (CIAWI 2016).

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.

Charlas dictadas

- Charla: Mejorando la evolución de software. I Congreso Internacional de Innovación Tecnológica & Desarrollo de Software. Quito 10 y 11 de Noviembre 2016.
- Charla: Desarrolllo de software con métodos ágiles. I Congreso Internacional de Innovación Tecnológica & Desarrollo de Software. Quito 10 y 11 de Noviembre 2016.
- Charla: Improving Requirements Engineering with NLP-enabled Tools. In the Master's course "Requirements Engineering and Software Architecture" at Stuttgart Universität, Germany, Nov. 2016.

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 de Jurado de Tesis de Posgrado

- Jurado de Tesis como miembro titular de la tesis de la Ing. Melina Vidoni Doctorado en Ingeniería mención Sistemas de Información de la Universidad Tecnológica Nacional, Facultad Regional Santa Fé. Santa Fé Argentina 11de Septiembre de 2017.
- Jurado de Tesis como miembro titular de la tesis de Maestría en Informática y Sistemas presentada por el Lic. Juan Gabriel Enriquez, 14 de julio 2016.
- Jurado de Tesis como miembro titular de la tesis de Maestría en Informática y Sistemas presentada por la Lic. Graciela Beatriz Vidal, 14 de julio 2016.

Dirección de trabajos finales de grado

- Directora en conjunto con el Dr. Andres Díaz Pace del trabajo final GENERACIÓN DE ESCENARIOS DE CALIDAD A PARTIR DE REQUERIMIENTOS TEXTUALES de los alumnos Andrés Vicente y Javier Enrique Marsicano Diciembre 2017.
- Directora en conjunto con el Dr. Alejandro Rago del trabajo final UN ASISTENTE INTELIGENTE PARA DERIVAR ESCENARIOS DE

ATRIBUTOS DE CALIDAD EN ARQUITECTURAS DE SOFTWARE del alumno Kevin Ruau Julio 2017.

Evaluaciones Académicas

- Miembro del comité evaluador de incentivos convocatoria 2014. Universidad Nacional de Jujuy Junio 2017.

Actualización Docente

- Integrante del equipo del Plan 111 MIL del Ministerio de Producción. Comienzo Noviembre 2016. Actividades: Dictado de curso, generador de contenido, generador de exámenes.

22. TITULO, 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

Mantenimiento de Software

El mantenimiento de software es una de las tareas más importantes y costosas en el ciclo de desarrollo de software debido a que los sistemas son cada vez más complejos y están sujetos a constantes cambios [3]. El mantenimiento del software se define como:

- El proceso de modificación de un sistema de software o un componente luego de su entrega con el fin de corregir errores, mejorar la performance u otros atributos, o adaptarlo a un ambiente cambiante [8].
- Un producto de software recae en modificaciones de código y documentación asociada debido a un problema o la necesidad de una mejora. El objetivo es modificar el producto de software existente preservando su integridad [9].

Los cambios realizados al software pueden ser cambios sencillos para corregir errores de código, cambios más extensos para corregir errores de diseño o mejoras significativas para corregir errores de especificación o soportar nuevos requerimientos.

Un error que se comete usualmente, es que se piensa al software como un producto al que se le puede definir una etapa de finalización, y que se termina de construir en un momento dado. Se plantea en [5] que una primera versión es solo eso, una "primera versión" dentro de un proceso de evaluación continua.

Cuando el software evoluciona es importante identificar cuáles son los componentes estables y cuáles necesitan que se aplique un mantenimiento. En los componentes que se identifican que es necesario el mantenimiento, es importante analizar la legibilidad, extensibilidad, correctitud y calidad ya que probablemente vuelva a ser modificado en un futuro. Si las sucesivas modificaciones no son efectuadas correctamente, el código se vuelve difícil de mantener, se introducen nuevos bugs y se vuelve más difícil adaptar el sistema a nuevos requerimientos.

Técnicas y estrategias de mantenimiento de software

En la ingeniería de software se sabe que los cambios serán necesarios, ya sea en el presente o a futuro en la vida de los sistemas que se implementen. Actualmente, los enfoques tradicionales de la ingeniería de software en muchas ocasiones carecen de mecanismos que permitan anticipar los cambios.

No importa qué tan bien se diseñe un sistema, siempre pueden surgir modificaciones imprevistas que dificulten la tarea de mantenimiento. Por otro lado, diseñar un sistema desde cero no es algo que se da comúnmente en la ingenieria de software, sino que es algo que se dé muy raramente. La mayoría de los ingenieros son asignados a mantener y/o evolucionar aplicaciones existentes. Tales sistemas sufren de problemas típicos, como documentación obsoleta, diseño complejo, mecanismos de parches, duplicación de código, componentes obsoletas, pérdida de los desarrolladores originales, etc. [7][4].

Una de las técnicas más utilizadas dentro del contexto de la programación orientada a objetos, es la técnica de refactoring. Refactorizar se describe como el proceso de cambio de la estructura interna de un sistema de software, con el objetivo de que sea más fácil de entender y menos costoso de modificar, conservando el comportamiento externo del mismo. Es una técnica disciplinada para optimizar el código, y minimizar las posibilidades de introducir errores [10].

Esta técnica, que incrementa la legibilidad, la mantenibilidad y permite mejorar el diseño de software mediante la modularización, es llevada a cabo como parte del proceso de desarrollo de software. Con la misma, se mejora el código fuente de tal forma que el esfuerzo y costo de realizar modificaciones sobre el sistema sea menor respecto a hacerlo sobre un sistema sin refactorizar.

Una de las estrategias para identificar las componentes a refactorizar es analizando los code smells del código de la aplicación. Un code smell, también llamado bad smell, se refiere a cualquier síntoma que puede indicar que el código fuente presenta problemas [48], o como su nombre lo indica, tiene “mal olor”. Los code smells surgen debido a “malas prácticas” empleadas durante el proceso de desarrollo de software, como por ejemplo, métodos muy largos, código duplicado, clases muy grandes, métodos que engloban funcionalidad en exceso, etc. [13].

Los code smells, generalmente no son técnicamente incorrectos, no son errores, y no interfieren en el funcionamiento normal de un sistema de software. Sin embargo, aumentan el riesgo a fallos, afectan el diseño del sistema y hacen que el sistema se vuelva más lento en cuanto al desarrollo y mantenimiento del mismo [14]. Existen diferentes catálogos de code smells que se utilizan para detectar los lugares en el código a mejorar, como los presentados por Fowler [6] y Lanza y Marinescu [11].

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 tedioso 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 [13].

Mantenimiento de Sistemas en JavaScript

JavaScript es uno de los lenguajes de programación más potentes e importantes en la actualidad. Algunas de sus ventajas con respecto a otros lenguajes son su practicidad, utilidad y disponibilidad en cualquier navegador web. Además, se puede usar para la programación frontend, agregando mayor interactividad a la web, y también en los servidores web. Según la encuesta que realiza el sitio web Stack Overflow, que es el más utilizado por la comunidad de desarrolladores para encontrar soluciones a problemas de programación en diferentes lenguajes, el lenguaje JavaScript es el más utilizado en el mundo [14].

En el lenguaje JavaScript, se ha avanzado en relación a la sintaxis, funcionalidad y características en general, pero hasta el momento no se ha encontrado un proceso o funcionalidad para abordar problemas de calidad de los sistemas de software relacionados con los mantenibilidad. Por lo tanto, la detección de code smells en el lenguaje JavaScript es una problemática que aún no presenta soluciones, o al menos no se han hallado en los trabajos relacionados que se investigaron. A pesar de que es muy importante la detección de code smells para lograr que el código sea mantenible y se eviten errores en el futuro, aún no existe un catálogo de métricas para poder detectar code smells en JavaScript.

Plan de Trabajo para el Período 2019-2020

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 y comenzar a análizar las diferentes estrategias que permitan mejorar la mantenibilidad de sistemas JavaScript. 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 y Feature Envy. La nueva funcionalidad a incorporar en la herramienta está relacionada a las siguientes actividades de investigación:

- Definición de estrategias de resolución de los code smells: Actualmente, la herramienta provee alternativas de solución del code smell Brain Method y Feature Envy. El objetivo es proveer un análisis de las diferentes alternativas de resolución 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 de los code smells ya incorporados a la herramienta.
- Análisis de estrategias para identificar code smells en JavaScript. JavaScript no posee estrategias ni mecanimos para identificar code smells por lo que se analizarán los catálogos de code smells de Java para encontrar su adecuación a JavaScript.
- 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] 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] Computing Curricula Series, “Software Engineering 2004” , Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, August 2004.
[4] Demeyer, Serge, Stephane Ducasse, and Oscar Nierstrasz, “Object-Oriented Reengineering Patterns”, Morgan Kaufmann, 2002.
[5] Favre, Jean Marie, "Languages evolve too! changing the software time scale", Principles of Software Evolution, Eighth International Workshop on. IEEE, 2005.
[6] Fowler, M., “Refactoring: Improving the Design of Existing Code”, Addison Wesley, 1999.
[7] Feathers, Michael C.. Working Effectively with Legacy Code. Prentice Hall, 2005.
[8] IEEE. Standard 610.12 1990: Glossary of Software Engineering Terminology, volume 1. IEEE Press, 1999.
[9] Institute of Electrical and Electronics Engineers and Electronics Industry Association. Ieee/eia 12207 industry implementation of international standard iso/iec 12207 : 1995. Standard, March 1998.
[10]James M. Bieman and Byung K. Kang. “Cohesion and reuse in an Object-Oriented System.” In Proceedings ACM Symposium on Software Reusability, April 1995.
[11]Lanza, Michele, Radu Marinescu, “Object-Oriented. Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems”, 2006.
[12]McCabe, Thomas J., “A complexity Measure”, IEEE Transactions on Software Engineering, Vol. SE-2, NO.4, December 1976.
[13]Murphy Hill, Emerson, and Andrew. Black. Breaking the barriers to successful refactoring. Software Engineering, 2008. ICSE'08. ACM/IEEE 30th International Conference on. IEEE, 2008.
[14]Node Package Manager, https://www.npmjs.com/
[15]Van Emden, Eva and Leon Moonen, “Java quality assurance by detecting code smells” Reverse Engineering, 2002. Proceedings. Ninth Working Conference on. IEEE, 2002.

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 22).
   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: infinvest@cic.gba.gob.ar (puntos 1 al 22), 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.