Investigación en Progreso: Estudio del Comportamiento Dinámico del Diseño de Sistemas de Información basado en Redes Complejas

Jorge Benjamín Pagani[1,2]
1. Programa de Maestría en Ingeniería de Sistemas de Información.
Escuela de Posgrado, Facultad Regional de Buenos Aires. Universidad Tecnológica Nacional. Argentina.
2. Laboratorio de Investigación y Desarrollo en Arquitecturas Complejas.
Grupo de Investigación en Sistemas de Información. Universidad Nacional de Lanús. Argentina.
jbpagani@gmail.com

***Resumen* — La medición de sistemas de información es una de las tareas más importantes dentro de la Ingeniería del Software ya que permite caracterizarlos y estudiarlos eficientemente. Se han desarrollado decenas de métricas aunque principalmente orientadas a aspectos estáticos. Este proyecto tiene como objetivo desarrollar un proceso de análisis que permita estudiar y evaluar el comportamiento dinámico de un sistema software desde el punto de vista del diseño de sistemas.**

***Palabras claves* — ingeniería del software, análisis dinámico de sistemas de información, diseño de sistemas de información, métricas de software, redes complejas.**

I. Justificación de la propuesta

La medición de sistemas de información es uno de los aspectos más importantes de la Ingeniería del Software ya que permite la comprensión de las características cuantitativas y cualitativas de los mismos. El objetivo de la misma es proveer información de valor para los procesos de toma de decisiones del ciclo de vida de un proyecto de software, tanto en lo que respecta a aspectos técnicos como a aspectos gerenciales [7].

Las métricas dinámicas de diseño, entendidas como aquellas que se enfocan en medir el comportamiento en ejecución de aspectos del diseño de sistemas, no han recibido suficiente atención hasta el momento ya que los desarrollos se han orientado históricamente a las métricas estáticas y, en los últimos años, a las métricas dinámicas pero a nivel de código [32].

Se parte de la hipótesis que un conjunto de métricas dinámicas, definidas a nivel de diseño, podrían ser utilizadas para obtener una mejor interpretación del funcionamiento del sistema en una fase temprana de su desarrollo.

II. Estado actual del conocimiento sobre el tema

Se presenta a continuación una breve introducción a las métricas de software (sección A), en donde se pondrá especial énfasis en demostrar las ventajas de las métricas dinámicas por sobre las estáticas (sección B) y los beneficios que presentan las métricas dinámicas a nivel de diseño con respecto a las métricas dinámicas a nivel de código (sección C). Así mismo, se realizará una descripción de los avances en modelado y medición de sistemas software mediante redes complejas (sección D).

A. Métricas de software

A medida que los sistemas software crecen en tamaño y complejidad se necesitan mejores métodos y herramientas para poder caracterizarlos y estudiarlos, de forma tal de tener la capacidad de entenderlos y gestionarlos eficientemente [33]. Las métricas de software cumplen un papel esencial a la hora de entender los sistemas y administrarlos a lo largo de todo su ciclo de vida [7].

Una primera clasificación de las métricas de software [19] permite agruparlas como:

- estáticas, que estudian que hace el software
- dinámicas, que se enfocan en como lo hace

Diversos factores han llevado a que las métricas estáticas hayan recibido mayor desarrollo y atención hasta el momento [32], destacándose entre los principales el hecho que en las primeras etapas del ciclo de vida de un proyecto de software, los esfuerzos están centrados principalmente en lo que debe hacer el software y no en cómo debe hacerlo.

Una segunda clasificación de las métricas permite agruparlas de acuerdo a la etapa del desarrollo de sistemas en la cual se aplican, limitándose el estudio del aspecto dinámico a las etapas de diseño y codificación [6] [18] [25].

B. Métricas estáticas y métricas dinámicas

En la literatura de métricas de software ha quedado demostrado [7] [26] [28] que, en comparación con las métricas estáticas, las métricas dinámicas proveen información precisa sobre el funcionamiento del software. Esta limitación de las métricas estáticas surge del hecho de que las mismas sólo pueden inferir el comportamiento del software a partir de su estructura, mientras que las métricas dinámicas lo analizan intrínsecamente. Esta diferenciación se hace aún más evidente en el paradigma orientado a objetos, cuyas principales virtudes, como el polimorfismo, sólo pueden estudiarse en un ambiente dinámico.

A diferencia de las métricas estáticas, las métricas dinámicas habilitan:

- El análisis del comportamiento del sistema
- La medición de aspectos cuantitativos y cualitativos del funcionamiento del software
- La ponderación de la precisión en sistemas reales
- El estudio de técnicas propias de la orientación a objetos

C. Métricas dinámicas a nivel de diseño y métricas dinámicas a nivel de código

Existe un desarrollo desbalanceado entre las métricas dinámicas a nivel de diseño y las métricas dinámicas a nivel de código. En lo que respecta a las métricas dinámicas a nivel de

código hay una nutrida literatura sobre las mismas, habiéndose desarrollado métricas de:

- acoplamiento [4] [8] [11] [14] [21]
- cohesión [8] [9] [12] [13] [31]
- complejidad estructural [8] [20] [22] [35] [36]

Mientras que en lo que respecta a métricas dinámicas de diseño sólo se ha avanzado en el conjunto de métricas propuestas en Dynamic Metrics for Object Oriented Designs [36], las cuales no tienen en consideración algunas propiedades clave de la orientación a objetos como la herencia y el polimorfismo, tal como se indica en el trabajo de Chhabra y Gupta [7].

D. Sistemas software como redes complejas

Las redes complejas han demostrado su capacidad para modelar sistemas software considerando a los mismos como redes funcionales que interconectan un elevado número de módulos (clases u objetos) que colaboran entre sí [20] [23] [33] [37]. Del cuerpo de conocimiento desarrollado en el área se concluye que los sistemas software se comportan como redes libres de escala (scale-free), de forma análoga a la mayoría de sistemas creados por el hombre [1] [33].

El análisis de sistemas software mediante redes complejas ha sido limitado a la comprensión de los mismos a partir de su estructura y a la predicción de su evolución mediante mecánica estadística sobre redes complejas [22] [33] [34]. Un camino alternativo de análisis se propone en [20] en el cual se desarrolla una métrica de complejidad estructural de un sistema software basada en redes complejas.etc.

III. Objetivo del Proyecto de Investigación

El objetivo principal de este trabajo es desarrollar un proceso de análisis que permita estudiar y evaluar el comportamiento dinámico de un sistema software desde el punto de vista del diseño de sistemas. El proceso de análisis propuesto deberá permitir identificar falencias y oportunidades de mejora en el diseño del sistema para así proveer una mejor entrada a las siguientes etapas del ciclo de vida de desarrollo de software. Se proponen además los siguientes objetivos específicos:

A. Objetivos específicos

- Desarrollar un proceso de modelado de la dinámica de un sistema software basado en redes complejas, el cual deberá ser aplicable a partir de diagramas de interacción de UML v2
- Desarrollar un conjunto de métricas, aplicables sobre los modelos generados, que permitan analizar y evaluar el comportamiento del software a partir de los principios del diseño de sistemas

B. Alcance de los objetivos planteados

- El proceso a desarrollar debe ser escalable a fin de permitir la incorporación de nuevas métricas
- El proceso a desarrollar debe brindar una concepción integral del software pero a la vez permitir que sea aplicado a una sola parte del mismo
- Las métricas a desarrollar deben cumplir con los criterios de validación de métricas de software definidos en el estándar IEEE 1061-1998

IV. Metodología de Desarrollo

Para construir el conocimiento asociado al presente proyecto de investigación, se seguirá un enfoque de investigación clásico [10] [27] con énfasis en la producción de tecnologías [30]; identificando métodos y materiales necesarios para desarrollar el mismo.

A. Métodos

En esta sección se definen los métodos que se aplicarán para el desarrollo de esta tesis.

1) Revisiones Sistemáticas

Las revisiones sistemáticas de artículos científicos [3] siguen un método explícito para resumir la información sobre determinado tema o problema. Se diferencian de las revisiones narrativas en que se originan a partir de una pregunta estructurada y de un protocolo previamente establecido.

2) Prototipado Evolutivo Experimental (Método de la Ingeniería)

El prototipado evolutivo experimental [5] consiste en desarrollar una solución inicial para un determinado problema, generando su refinamiento de manera evolutiva por prueba de aplicación de dicha solución a casos de estudio (problemáticas) de complejidad creciente. El proceso de refinamiento concluye al estabilizarse el prototipo en evolución.

B. Materiales

Los materiales que se utilizarán para el desarrollo de esta tesis son:

- Formalismos de modelado conceptual usuales en la Ingeniería de Software [29] y enfoques recientes [15]
- Modelos de Proceso usuales en Ingeniería de Software [2] [16] [24]

C. Metodología

Para alcanzar los objetivos se propone: (i) realizar una investigación documental sobre métricas de software, particularmente en lo que respecta a métricas dinámicas y métricas a nivel de diseño, identificando casos de estudio, prueba y validación, (ii) desarrollar una versión inicial del proceso de análisis (incluyendo el proceso de modelado de la dinámica del sistema como red compleja y el conjunto de métricas dinámicas a nivel de diseño aplicables sobre el modelo), aplicando posteriormente dicho proceso de análisis sobre los casos de estudio identificados, perfeccionándolo mediante casos de complejidad creciente (prototipado evolutivo experimental) y (iii) aplicar el proceso desarrollado a los casos de prueba y validación, analizando los resultados obtenidos de las métricas para posteriormente compararlos con las métricas existentes.

V. Condiciones institucionales para el desarrollo del proyecto de investigación

Se prevé desarrollar las investigaciones vinculadas al Proyecto de Tesis de Magister en el Laboratorio de Investigación y Desarrollo en Arquitecturas Complejas (LIDAC GISI UNLa) a cargo del Director Propuesto. El LIDAC pertenece al Grupo de Investigación en Sistemas de Información (GISI) de la Universidad Nacional de Lanús. Las líneas de investigación del GISI cuentan con financiamiento de la Secretaria de Ciencia y Técnica de la misma Universidad. El GISI tiene vínculos con Grupos de Investigación que trabajan en áreas vinculadas a las temáticas de tesis entre los que cabe

mencionar: el Grupo de Ingeniería de Software Empírica de la Universidad Politécnica de Madrid que dirige la Dra. Natalia Juristo y el Instituto de Investigación en Informática de la Universidad Nacional de La Plata que dirige el Ing. Armando De Giusti, Estas vinculaciones permitirán al candidato intercambiar opiniones con tesistas de posgrado e investigadores que trabajen en líneas de investigación afines.

VI. BIBLIOGRAFÍA

[1] Albert, R., & Barabási, A. L. (2002). Statistical mechanics of complex networks. Reviews of modern physics, 74(1), 47.

[2] ANSI/IEEE (2007). "Draft IEEE Standard for software and system test documentation". ANSI/IEEE Std P829-2007.

[3] Argimón J. (2004). "Métodos de Investigación Clínica y Epidemiológica". Elsevier España, S.A. ISBN 9788481747096.

[4] Arisholm, E., Briand, L. C., & Foyen, A. (2004). Dynamic coupling measurement for object-oriented software. Software Engineering, IEEE Transactions on, 30(8), 491-506.

[5] Basili, V. (1993). "The Experimental Paradigm in Software Engineering". En Experimental Software Engineering Issues: Critical Assessment and Future Directions (Ed. Rombach, H., Basili, V., Selby, R.). Lecture Notes in Computer Science, Vol. 706. ISBN 978-3-540-57092-9.

[6] Boehm, B. W. (1988). A spiral model of software development and enhancement. Computer, 21(5), 61-72.

[7] Chhabra, J. K., & Gupta, V. (2010). A survey of dynamic software metrics. Journal of computer science and technology, 25(5), 1016-1029.

[8] Chidamber, S. R., & Kemerer, C. F. (1991). Towards a metrics suite for object oriented design (Vol. 26, No. 11, pp. 197-211). ACM.

[9] Cho, E. S., Kim, C. J., Kim, D. D., & Rhew, S. Y. (1998, December). Static and dynamic metrics for effective object clustering. In Software Engineering Conference, 1998. Proceedings. 1998 Asia Pacific (pp. 78-85). IEEE.

[10] Creswell, J. W. (2002). Educational research: Planning, conducting, and evaluating quantitative and.

[11] Gunnalan, R., Shereshevsky, M., & Ammar, H. H. (2005). Pseudo dynamic metrics [software metrics]. In Computer Systems and Applications, 2005. The 3rd ACS/IEEE International Conference on (p. 117). IEEE.

[12] Gupta, V., & Chhabra, J. K. (2011). Dynamic cohesion measures for object-oriented software. Journal of Systems Architecture, 57(4), 452-462.

[13] Gupta, N., & Rao, P. (2001, November). Program execution based module cohesion measurement. In Automated Software Engineering, 2001.(ASE 2001). Proceedings. 16th Annual International Conference on (pp. 144-153). IEEE.

[14] Hassoun, Y., Counsell, S., & Johnson, R. (2005, December). Dynamic coupling metric: proof of concept. In Software, IEE Proceedings- (Vol. 152, No. 6, pp. 273-279). IET.

[15] Hossian, A. (2012). "Modelo de Proceso de Conceptualización de Requisitos" (Tesis Doctoral en Ciencias informáticas). Facultad de Informática. Universidad Nacional de La Plata.

[16] IEEE (1997). "IEEE Standard for Developing Software Life Cycle Processes. IEEE Std 1074-1997" Revision of IEEE Std 1074-1995; Replaces IEEE Std 1074.1-1995.

[17] IEEE (1998). "IEEE Std. 1061-1998, Standard for a Software Quality Metrics Methodology, revision." Piscataway, NJ,: IEEE Standards Dept., 1998.

[18] Jiang, Y., Cuki, B., Menzies, T., & Bartlow, N. (2008, May). Comparing design and code metrics for software quality prediction. In Proceedings of the 4th international workshop on Predictor models in software engineering (pp. 11-18). ACM.

[19] Kaur, K., Minhas, K., Mehan, N., & Kakkar, N. (2009). Static and Dynamic Complexity Analysis of Software Metrics. World Academy of Science, Engineering and Technology, 56, 2009.

[20] Ma, Y., He, K., & Du, D. (2005). A qualitative method for measuring the structural complexity of software systems based on complex networks. In Software Engineering Conference, 2005. APSEC'05. 12th Asia-Pacific (pp. 7-pp). IEEE.

[21] Mitchell, A., & Power, J. F. (2004, January). An approach to quantifying the run-time behaviour of Java GUI applications. In Proceedings of the winter international synposium on Information and communication technologies (pp. 1-6). Trinity College Dublin.

[22] Munson, J. C., & Khoshgoftaar, T. M. (1992). Measuring dynamic program complexity. Software, IEEE, 9(6), 48-55.

[23] Myers, C. R. (2003). Software systems as complex networks: Structure, function, and evolvability of software collaboration graphs. Physical Review E, 68(4), 046116.

[24] Oktaba, H., Garcia, F., Piattini, M., Ruiz, F., Pino, F., Alquicira, C. (2007). "Software Process Improvement: The Competisoft Project". IEEE Computer, 40(10): 21-28. ISSN 0018-9162.

[25] Pressman, R. S., & Ince, D. (1992). Software engineering: a practitioner's approach (Vol. 5). New York: McGraw-hill.

[26] Quynh, P. T., & Thang, H. Q. (2009). Dynamic coupling metrics for service–oriented software. International Journal of Computer Science and Engineering, 3(1), 46-46.

[27] Rosas, L., & Riveros, H. G. (1985). Iniciación al método científico. 1ª.

[28] Rothlisberger, D., Harry, M., Villazón, A., Ansaloni, D., Binder, W., Nierstrasz, O., & Moret, P. (2009, September). Augmenting static source views in IDEs with dynamic metrics. In Software Maintenance, 2009. ICSM 2009. IEEE International Conference on (pp. 253-262). IEEE.

[29] Rumbaugh, J., Jacobson, I., & Booch, G. (1999). The unified modeling language reference manual.

[30] Sabato, J. A., & Mackenzie, M. (1982). La producción de tecnología: autónoma o transnacional. Instituto Latinoamericano de Estudios Transnacionales.

[31] Safari-Sharifabadi, E., & Constantinides, C. (2008). Dynamic analysis of Ada programs for comprehension and quality measurement. ACM SIGAda Ada Letters, 28(3), 15-38.

[32] Tahir, A., & MacDonell, S. G. (2012, September). A systematic mapping study on dynamic metrics and software quality. In Software Maintenance (ICSM), 2012 28th IEEE International Conference on (pp. 326-335). IEEE.

[33] Wen, L., Kirk, D., & Dromey, R. G. (2007). Software systems as complex networks. In Cognitive Informatics, 6th IEEE International Conference on (pp. 106-115). IEEE.

[34] Wen, L., Dromey, R. G., & Kirk, D. (2009). Software Engineering and Scale-Free Networks. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 39(4), 845-854.

[35] Yacoub, S. M., & Ammar, H. H. (2002). A methodology for architecture-level reliability risk analysis. Software Engineering, IEEE Transactions on, 28(6), 529-547.

[36] Yacoub, S. M., Ammar, H. H., & Robinson, T. (1999). Dynamic metrics for object oriented designs. In Software Metrics Symposium, 1999. Proceedings. Sixth International (pp. 50-61). IEEE.

[37] Zheng, X., Zeng, D., Li, H., & Wang, F. (2008). Analyzing open-source software systems as complex networks. Physica A: Statistical Mechanics and its Applications, 387(24), 6190-6200.

Benjamín Pagani. Es Ingeniero en Sistemas por la Facultad Regional Buenos Aires de la Universidad Tecnológica Nacional. Es Candidato del Programa de Magister en Ingeniería de Sistemas de Información de la Escuela de Postgrado de la Facultad Regional Buenos Aires de la Universidad Tecnológica Nacional. Es Investigador Tesista del Laboratorio de Investigación y Desarrollo en Arquitecturas Complejas del Grupo de Investigación en Sistemas de Información de la Universidad Nacional de Lanús. Es Consultor en Inteligencia de Negocios. Sus áreas de interés son: inteligencia de negocios, ingeniería de software, infraestructura IT y bases de datos.