Revista Digital del Departamento de Ingeniería e Investigaciones Tecnológicas de la Universidad Nacional de La Matanza

ISSN: 2525-1333
Vol.:4-Nro.2 (Diciembre-2019)

artículo original

Similitudes y Diferencias entre el Análisis de Sistemas y la Ingeniería de Requisitos

Similarities and Differences between Systems Analysis and Requirements Engineering

Gladys Noemí Kaplan[(1)] ***y*** *Jorge Horacio Doorn*[(2,3)]

[(1)]Universidad Nacional de La Matanza, Departamento de Ingeniería e Innovación Tecnológica
gkaplan@unlam.edu.ar

[(2)]Universidad Nacional del Oeste
[(3)]Universidad Nacional de Tres de Febrero
jdoorn@uno.edu.ar

Resumen:

Pese a la gran cantidad de años de uso del Análisis de Sistemas y de los años transcurridos desde la introducción de la Ingeniería de Requisitos, las diferencias y similitudes entre ambas permanecen confusas. Varios autores han presentado a la Ingeniería de Requisitos como una evolución natural del Análisis de Sistemas, sin indicar en qué consiste esa evolución y cuáles son sus consecuencias. Quizás parte de la

ISSN: 2525-1333. Vol.:4-Nro.2 (Diciembre-2019)

confusión está originada en que tanto el Análisis de Sistemas como la Ingeniería de Requisitos abordan la misma problemática y comparten algunos métodos, herramientas y técnicas. Estas similitudes han llevado a considerar que pueden ser utilizadas indistintamente sin consecuencias. En el presente artículo se describen someramente ambas disciplinas y se analizan los efectos que se producen por la elección de una en lugar de la otra. La relevancia de la presente comparación reside en el hecho que una parte importante de las fallas de los sistemas de software son originadas por defectos en los requisitos, y estos suelen provenir justamente de la falta de percepción de la importancia de las diferencias entre ambas visiones.

Abstract:

In despite of the number of years of use System Analysis and since the introduction of the Requirements Engineering, the differences and similarities between them remain confuses. Several authors have presented Requirements Engineering as a natural evolution of Systems Analysis, with no indication about the aims of such evolution and its consequences. Perhaps, part of the confusion comes from the fact that both Systems Analysis and Requirements Engineering address the same problem and share some methods, tools and techniques. These similarities have led to consider that they can be used interchangeably without inconvenience. In this article both disciplines are briefly described and the effects that are produced by choosing one instead of the other are analyzed. The relevance of the present comparison lies in the fact that an important part of the failures of software systems are caused by defects in the requirements and they usually come precisely from the lack of perception of the importance of the differences between both approaches.

Palabras Clave: *Análisis de Sistemas, Ingeniería de Requisitos, Requisitos del Software.*

Key Words: *System Analysis, Requirements Engineering, Software Requirements.*

I. CONTEXTO

Es notable que pese al tiempo transcurrido y de toda la actividad académica y profesional referida al desarrollo de sistemas de software y particularmente a la etapa de construcción de los requisitos del software, todavía resultan confusas las diferencias entre el Análisis de Sistemas (AS) [1] [2] [3] [4] y la Ingeniería de Requisitos (IR) [5] [6] [7] [8]. Si bien ambas disciplinas comparten el mismo objeto de

estudio y el mismo objetivo, llegan a ellos de diferente manera. Por un lado, el AS atiende datos, entidades involucradas, etc. Por el otro, la IR presta atención a objetivos, conductas y necesidades. Además de atender diferentes aspectos del contexto, también utilizan diferentes formas de representación, ya que el AS utiliza modelos propios a su disciplina mientras que la IR utiliza modelos cercanos a los clientes y usuarios.

El presente artículo se enmarca en el proyecto de investigación que se lleva a cabo en la UNLaM denominado "Aspectos No Funcionales de los Procesos de Requisitos".

II. INTRODUCCIÓN

Desde una mirada más metodológica que estratégica, se puede observar que el AS ha tomado poca distancia del diseño y del código, provocando una suerte de simbiosis metodológica en pos de una mayor calidad en el producto software a construir; calidad entendida en el contexto del sistema en si propio, es decir ponderando la robustez y mantenibilidad del producto.

Dado que el análisis aspira a facilitar el diseño y la programación, entonces esta última ha ido influyendo sobre aquel en forma progresiva. Esto ha sucedido desde épocas muy tempranas, haciendo que la programación presione metodológicamente a las etapas más tempranas del desarrollo. Tal es el caso del paradigma Estructurado que conceptualiza el sistema en términos de funciones y procesos, input y output y en el OO el desarrollador piensa en términos de objetos, clases, métodos, herencia, etc. En ambos existe una distancia conceptual muy importante en el cómo se ve el contexto y en el cómo se lo modela [9] y ambos con la forma que ve el contexto la IR.

> "El **análisis de sistemas** es el proceso de clasificación e interpretación de hechos, diagnóstico de problemas y empleo de la información para recomendar mejoras al sistema. Este es el trabajo del analista de sistemas"
>
> **James Senn**
> *Análisis y Diseño de Sistemas de Información*

La IR se estableció a mediados de los 70, donde se comenzó a investigar como una práctica individual [7]. Mucho se ha avanzado en cuanto a los métodos, técnicas y herramientas, pero aún queda mucho por hacer.

La IR hipotetiza que la importancia de los inconvenientes que se detallan más abajo para el AS, referidos a no asegurar una clara comprensión de las necesidades de los clientes-usuarios, es notablemente superior a las ventajas enunciadas y que lo que se debe hacer es facilitar la comprensión por parte de los clientes usuarios de las características del software que van a recibir. Que esta hipótesis sea efectivamente correcta no ha sido completa ni extensivamente probado. Tampoco es el objeto de este trabajo avanzar sobre este punto en gran profundidad. Sí, se pretende clarificar las diferencias entre el AS y la IR y todas sus implicancias.

ISSN: 2525-1333. Vol.:4-Nro.2 (Diciembre-2019)

> "La **ingeniería de requisitos** puede ser definida como un proceso sistemático de desarrollo de los requisitos a través de un proceso cooperativo-interactivo de análisis del problema, documentando los resultados observados en una variedad de formatos de representación, y chequeando la comprensión obtenida. Las actividades envueltas en la ingeniería de requisitos ayudan a comprender las exactas necesidades de los usuarios de un sistema de software y trasladar esas necesidades en sentencias precisas y sin ambigüedad las que luego serán utilizadas en el desarrollo de software."
>
> ***Loucopoulos and Karakostas***
> *System Requirements Engineering*

La IR privilegia la comprensión por parte del cliente-usuario de los servicios que brindará el sistema de software y las consecuencias que los mismos tendrán sobre el contexto en que se lo inserte. Para logarlo, utiliza representaciones comprensibles para ellos y utiliza, en muchos casos, el lenguaje natural y el vocabulario propio del dominio donde el sistema de software se pondrá en servicio.

III. LENGUAJE NATURAL

Durante muchos años la frase "una imagen vale más que mil palabras" justificó la existencia de modelos gráficos en los procesos de construcción del software. Esta frase fue dejándose de lado al vislumbrar que muchos proyectos de software fracasaban debido a Especificaciones de Requisitos de Software (ERS) incorrectas e incompletas. Muchos de estos defectos tenían su origen en la utilización de representaciones que los clientes-usuarios no podían comprender. Requisitos incompletos e incorrectos, significa que se dejaban de atender necesidades o aspiraciones de los clientes-usuarios o que se las atendía de una manera inapropiada en términos de los deseos los mismos. Una de estas postulaciones a favor de lo gráfico se dio en el paradigma estructurado haciendo referencia a "la vaguedad y embrollo del lenguaje corriente" [10], justificando el uso de herramientas gráficas durante el análisis. Esto es simultáneamente, cierto y nocivo. En esa simple afirmación hay actores en off, que se deben identificar. Para los desarrolladores, el uso de gráficos es lisa y llanamente mejor. Para los clientes-usuarios se está reemplazando la *vaguedad y el embrollo del lenguaje corriente* por gráficos u otras representaciones incomprensibles, esto es peor.

El uso del lenguaje natural (LN) para describir los procesos del negocio facilita el intercambio de conocimiento entre los clientes-usuarios y el ingeniero/a de requisitos. Facilitar el intercambio de conocimiento mejora la comunicación en todas las direcciones posibles dentro del proceso de requisitos y asegura la comprensión de todos los artefactos generados. El lenguaje natural es un denominador común entre el equipo de desarrollo y los clientes-usuarios que puede ser utilizado con un mínimo esfuerzo de aprendizaje. Algunos procesos utilizan el LN con un cierto grado de estructuración, como sucede con las secciones pre-establecidas de la IEEE Std. 830-1998 o el uso de un vocabulario reducido.

El reemplazo de modelos basados en gráficos por modelos basados en lenguaje natural implica un

mayor esfuerzo por parte del ingeniero/a de requisitos y un menor esfuerzo por parte de los clientes-usuarios. Esto es muy apropiado, el esfuerzo recae en donde debe recaer, ya que los ingenieros de requisitos y los restantes miembros del equipo de desarrollo deben prestar un servicio importunando lo menos posible a los clientes-usuarios.

Que los principales problemas asociados al uso del LN sean la ambigüedad y la falta de precisión [11], no implica que se lo deba descartar en forma automática, ya que estas desventajas pueden ser mitigadas. Estos aspectos han sido una de las justificaciones para no utilizarlo, pero se ha comprobado que pueden ser controladas sin que esto afecte la legibilidad y simplicidad del texto, retomando la ventaja más significativa del uso del LN que es asegurar una buena comunicación.

El uso del LN en los documentos de la Ingeniería de Requisitos está largamente difundido, llegando ya en el año 2004 al 79% del total de todos los documentos existentes, según un estudio realizado en la Università di Trento de Italia [11]. En la Fig.1 se muestra otra distribución del mismo año, donde ese total asciende al 87,70% incluyendo el Lenguaje Corriente y el Lenguaje Estructurado [12].

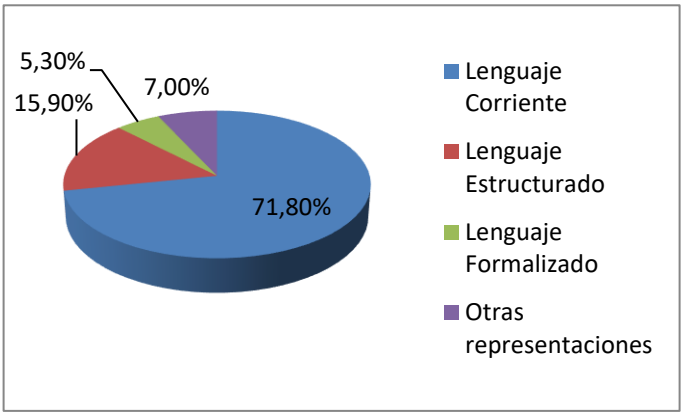

Fig.1 – Uso del LN en los documentos de Ingeniería de Requisitos

El LN es utilizado en modelos como los Casos de Uso [13] que son descripciones narrativas de la interacción entre un actor y el sistema, en los Escenarios [14] [15] [16] que describen situaciones del contexto, en los glosarios [17] [18] [19] [20] [21] [22], en modelo de Reglas de Negocio [23] [24] [25], en Historias de Usuarios (User-Stories) [26] y en otros.

IV. DIFERENCIAS ENTRE EL AS Y LA IR

El AS produce una descripción del sistema de software en términos que le son propios, esencialmente computacionales, mientras que la IR lo hace en términos de los servicios perceptibles por los clientes y usuarios. En la Tabla 1 se describen estas características que marcan y definen el comportamiento de cada uno, ya que para la discusión acerca de las ventajas relativas del AS sobre la IR o viceversa, se debe ponderar la importancia relativa de la calidad de la definición de

los servicios y la facilidad para el diseño del sistema.

	Calidad de la definición de los servicios del sistema	Facilidad para el diseño del sistema
AS	Pobre	Buena
IR	Buena	Pobre

Tabla 1 –Propiedades del AS y de la IR

Las ventajas de la IR, no son gratuitas, disponer de la especificación del sistema a ser desarrollado en base a modelos y construcciones que faciliten la comprensión por parte de los clientes y usuarios, dificulta en forma sensible la actividad de diseño. Naturalmente que son posibles infinidad de situaciones intermedias, pero siempre existirá la inexorable tensión entre facilitar un punto de vista o el otro.

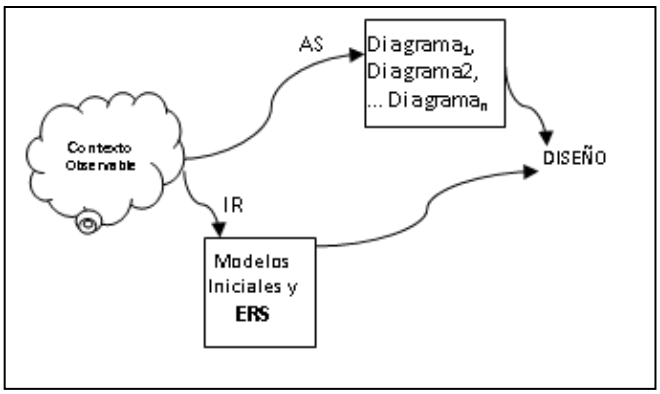

Fig. 2 – Distancias cognitivas

En la Fig.2 se identifican dos distancias cognitivas. Cabe aclarar que esta distancia está medida en términos de la comprensión y el esfuerzo tanto del cliente como del desarrollador. La primera distancia corresponde al camino entre el mundo observable y la obtención de los requisitos del software. La segunda, es el salto hacia el diseño. Se puede observar que las distancias son proporcionalmente inversas. Obviamente, esta es una apreciación puramente conceptual. La distancia cognitiva del AS se incrementa en la primera parte y disminuye significativamente cuando se acerca al diseño. Esto se debe a la conveniencia de los modelos utilizados por el AS. Por otro lado, la IR se mantiene lo más cerca posible del contexto observable, teniendo una distancia más pronunciada cuando debe avanzar hacia el diseño, necesitando de un esfuerzo adicional para lograrlo con éxito. Es justamente, la corta distancia entre el contexto observable y la IR donde radica el cambio, ya que la IR procura mantenerse dentro de los límites cognitivos del cliente.

Analizar estas distancias cognitivas antes de decidir cuál de las dos disciplinas es la adecuada, puede determinar la satisfacción o insatisfacción del cliente y puede depender de la forma de manifestarse que tenga el contexto observable.

V. CONCLUSIONES

En síntesis, el AS y la IR procuran hacer lo mismo por caminos diferentes y presentan sus resultados, también de manera diferente. Esta bifurcación produce otra mirada en cuanto a la calidad esperada por esta primera etapa del proceso de construcción del software, ya que ambas maximizan la calidad de los requisitos, pero miden esa calidad de forma diferente. Para

la IR, la calidad está entendida como satisfacción del cliente-usuario. Para el AS, la calidad está entendida como la facilidad para construir el software.

Así como es sencillo encontrar un ejemplo en el que el enfoque del AS es más apropiado, también es sencillo encontrar un ejemplo en que es mejor utilizar la IR. Posiblemente, la balanza se incline a favor de la IR cuando crece la complejidad del sistema.

VI. REFERENCIAS

[1] Tom DeMarco, "Structured Analysis", Prentice Hall; Edición: 1 (21 de mayo de 1979), ISBN-10: 0138543801, ISBN-13: 978-0138543808, 1978.

[2] Edward Yourdon & Larry L. Constantine, "Structured Design: Fundamentals of la Discipline of Computer Program and Systems Design", Yourdon Press, N.Y., 1978.

[3] James Senn, "Analisis y Diseño de Sistemas de Informacion", Segunda Edicion, McGraw Hill, 1991

[4] Kendal y Kendal, "Analisis y Diseño de Sistemas", Tercera Edicion, Prentice Hall, México, 1997.

[5] Zave, P., The Operational Versus the Conventional Approach to Software Development, Communications of the ACM, 27, 104-118, 1984.

[6] Leite, J.C.S.P., "Engenharia de Requisitos", Notas Tutoriales, material de enseñanza en el curso Requirements Engineering, Computer Science Department of PUC-Rio, Brasil, 1994.

[7] Loucopoulos, P., Karakostas, V., "System Requirements Engineering", McGraw-Hill, Londres, 1995.

[8] Faulk, S.R., "Software Requirements: A Tutorial", en Software Engineering, editores M. Dorfman y R.H. Thayer, IEEE Computer Society Press, 1996, pp.82-103. Reimpreso en "Software Requirements Engineering", editores Richard H. Thayer y Merlin Dorfman, IEEE Computer Society Press, 2º edición, Los Alamitos, CA, 1997, pp.128- 149.

[9] Jaelson Castro, Manuel Kolp,and John Mylopoulos, "A Requirements-Driven Development Methodology", K.R. Dittrich, A. Geppert, M.C. Norrie (Eds.): CaiSE 2001, LNCS 2068, pp. 108–123, 2001 © Springer-Verlag Berlin Heidelberg 2001.

[10] Gane & Sarson, "Structured Systems Analysis: Tools and Techniques", Editor: McDonnell Douglas Infomation; ISBN-10: 0930196007, ISBN-13: 978-0930196004, Edition: 1. 1977.

[11] Nadzeya Kiyavitskaya , Nicola Zeni , Luisa Mich , John Mylopoulos, "NLP-Based Requirements Modeling: Experiments on the Quality of the models", http://eprints.biblio.unitn.it/520", 2004.

[12] Berry Daniel and Erik Kamsties, "Ambiguity in Requirements Specification" in "Perspectives on Software Requirements", Kluwer Academic Publishers, EEUU, capítulo 2, 2004, pp.7-44.

[13] Jacobson, I., Christerson, M., Jonsson, P., Overgaard, G., "Object-Oriented Software Engineering - A Use Case Driven Approach", Reading, MA: Addison Wesley, Nueva York: ACM Press, 1992.

[14] Booch, G., "Object-Oriented Analysis and Design", The Benjamin Cummings Publishing Company, Redwood City, CA, 1992.

[15] "Inquiry-Based Requirements Analysis", Potts, C., Takahashi, K., Antón, A. I., IEEE Software, Vol. 11, No.2, pp 21-32, 1994.

[16] Wirfs-Brock, R., "Designing Objects and Their Interactions: A Brief Look at Responsibility-Driven Design", Scenario-Based Design: Envisioning Work and Technology in System Development, editor J. Carroll, John Wiley, Nueva York, 1995, pp. 337-359.

[17] Leite, J.C.S.P., "Eliciting Requirements Using a Natural Language Based Approach: The Case of the Meeting Scheduler", Monografía en Ciência da Computaçao Nº 13/93, Computer Science Department of PUC-Rio, Brasil, 1993.

[18] Whitenack, B.G. Jr., "RAPPeL: A Requirements Analysis Process Pattern Language for Object Oriented Development", Knowledge Systems Corp., 1994.

[19] Oberg, R., Probasco, L, Ericsson, M., "Applying Requirements Management with Use Cases", Rational Software Corporation, 1998.

[20] Kovitz, B.L., "Practical Software Requirements: A manual of Content and Style", Greenwich, CT: Manning Publications Co., Octubre 1998.

[21] Rolland, C., Ben Achour, C., Cauvet, C., Ralyté, J., Sutcliffe, A., Maiden, M., Jarke, M., Haumer, P., Pohl, K., Dubois, E., Heymans, P., "A Proposal for a Scenario Classification Framework", Requirements Engineering Journal, Springer-Verlag London Ltd., Vol.3, Nº1, 1998, pp.23-47.

ISSN: 2525-1333. Vol.:4-Nro.2 (Diciembre-2019)

[22] Alspaugh, T.A., Antón, A.I., Barnes, T., Mott, B.W.,"An Integrated Scenario Management Strategy", International Symposium On Requirements Engineering (RE´99), Limerick, Irlanda, IEEE Computer Society Press, 1999, pp.142-149.
[23] Ross, R., "The Business Rule Book: Classifying, Defining and Modeling Rules", Business Rule Solutions, LLC, 2º edición, 1997.
[24] Leite J.C.S.P, Leonardi, M.C., "Business rules as organizational Policies", IEEE Ninth International Workshop on Software Specification and Design, IEEE Computer Society Press, 1998, pp.68-76.
[25] Gottesdiener, E., "Business Rules as Requirements", Software Development, Vol.7, Nº12, Diciembre 1999.
[26] Beck, K., "Extreme Programming Explained: Embrace Change", Addison-Wesley, 2000.

ISSN: 2525-1333. Vol.:4-Nro.2 (Diciembre-2019)

Recibido: 2019-12-12
Aprobado: 2019-12-23
Hipervínculo Permanente: https://reddi.unlam.edu.ar
Datos de edición: Vol. 4 - Nro. 2 -Art. 3
Fecha de edición: Formato: 2020-01-31