Una Propuesta de Evaluación de Herramientas CASE para la Enseñanza

Nicolás Battaglia, Roxana Martínez, Carlos Neil, Marcelo De Vincenzi
Facultad de Tecnología Informática - Universidad Abierta Interamericana
{nicolas.battaglia, roxana.martinez, carlos.neil, medevincenzi}@uai.edu.ar

Abstract. Este trabajo presenta un marco teórico de las Herramientas CASE y UML en base al sustento de análisis realizado por varios autores y el propio del equipo. Por lo que, en base a dicho estudio, se propone un método de evaluación para conocer en qué grado una herramienta CASE se adapta con la funcionalidad requerida. Además, se sugieren criterios de evaluación a tener en cuenta con su importancia correspondiente para cada uno de los casos. Presentamos en base a esto, las características deseables de una herramienta CASE destinada a la enseñanza de la Ingeniería de Software.

Keywords: Herramientas CASE, UML, Evaluación, Ingeniería del Software.

1 Introducción

Actualmente en Argentina, el sector de software ofrece un potencial considerable en términos de crecimiento económico para las empresas de software. Este sector puede ser visto mediante dos perspectivas, por un lado, como avance a futuro sobre la modernización tecnológica en el país y, por otro lado, como un sector capaz de desarrollar procesos de innovación y aprendizaje. Partiendo de este último concepto, en el desarrollo de sistemas de información, es notorio el interés constante por mejorar la productividad y la calidad en los productos desarrollados. El avance en el software, impacta directamente en los costos de desarrollo y mantenimiento, por lo que es vital, establecer un marco de trabajo disciplinado y estructurado el cual permita identificar en forma clara y puntual, las tareas a realizar por un equipo de trabajo, siendo un punto fundamental para evitar resultados inesperados.

La inclusión de las herramientas CASE (*Computer Aided Software Engineering*), están tomando cada vez más relevancia en la planeación y ejecución de proyectos que involucran sistemas informáticos, debido a que permiten aumentar la productividad en el desarrollo de software. Estas herramientas, suelen inducir a sus usuarios a la correcta utilización de metodologías que implican una reducción tanto en el costo del proyecto, como en el tiempo de desarrollo del producto de software final. Para esto, es importante considerar que tipo de herramienta CASE es necesaria al momento de llevar adelante una actividad relacionada con la Ingeniería de Software (IS), por lo que es importante conocer y analizar las distintas taxonomías y la usabilidad relacionada en cada fase del ciclo de vida del desarrollo de software a construir.

Si bien, la importancia de la utilización de las herramientas CASE se relaciona con el mercado laboral, también debe ser considerada en el sector educativo, porque

impacta directamente en los alumnos que serán los que se encuentren a fututo, insertados en dicho mercado laboral. Por lo que, es necesario conocer y estudiar las diferentes limitaciones, ventajas y desventajas de utilizar herramientas CASE en el proceso de enseñanza en una carrera de ingeniería en sistemas, donde es fundamental hacer hincapié en el concepto de modelado, durante el análisis, diseño e implementación.

1.1 Herramientas CASE

A lo largo de los años, la introducción de nuevos componentes y circuitos integrados en la IS, permitió que varias aplicaciones comenzaran a ser propuestas válidas y a materializarse en productos mucho más amplios y complejos, dejando como experiencia que el enfoque informal en la construcción de software no era bueno, y que se generaban productos más caros de lo presupuestado en los proyectos, los cuales eran difíciles de mantener, con desempeños muy pobres y de mala calidad.

Los costos de software fueron aumentando considerablemente, mientras que los de hardware iban por el camino opuesto, por lo que surgió la necesidad de nuevas técnicas y métodos que permita controlar la creciente complejidad y así también, controlar los costos [1]. Fue notaria la gran demanda de calidad en el desarrollo de software en el mercado laboral, por lo que se buscaba una manera confiable y económica de administrar los recursos y así cumplir con la entrega del producto de software en tiempo y forma. Es por ello que, surge la necesidad de obtener productos más fáciles de mantener, con más capacidad de adaptación y con costos más accesibles, y, en consecuencia, comenzó a hacerse necesario automatizar las diferentes actividades que propone la IS En base a esto, IEEE (*Institute of Electrical and Electronics Engineers*) desarrollo diferentes estándares y una definición completa, la cual apunta a la aplicación de un enfoque sistemático, disciplinado y cuantificable en el desarrollo y mantenimiento del software.

Pressman asegura que “las herramientas CASE ayudan a garantizar que la calidad se diseñe, antes de llegar a construir el producto” [2]. Y propone entender la IS como una tecnología con diferentes capas, apoyándose en el compromiso con la calidad. Luego el proceso, como base de control; los métodos, que brindan la experiencia necesaria para elaborar software por medio de un conjunto de tareas (como el análisis, diseño, implementación y pruebas). Por otro lado, las primitivas herramientas CASE, se dirigieron principalmente a la automatización de la documentación y la comunicación como una mejora clave de la productividad del software [3].

1.2 Características adicionales

La utilización de modelos para crear software utilizando lenguajes, técnicas y herramientas dentro de la IS es una práctica común durante las etapas de análisis y diseño. Esto conlleva al desarrollo de un producto con calidad de Ingeniería para la resolución de un problema [4].

Todas las herramientas CASE prestan soporte a un lenguaje de modelado para acompañar la metodología que, en su gran mayoría, soportan UML (*Unified Modeling*

Language) el cual permite realizar una notación orientada a objetos, mediante diferentes diagramas, de los cuales, se representan las distintas etapas del desarrollo de un proyecto. UML es de amplia aceptación en el ambiente de IS, ya que su valor conceptual y visual, proporciona facilidad para representar elementos particulares de aplicaciones a desarrollar. Si una herramienta CASE soporta UML, la funcionalidad de intercambio de modelos con otras herramientas CASE se convierte en una meta alcanzable, teniendo en cuenta la estandarización sintáctica y semántica que persigue dicho lenguaje. En esta línea de estandarización la OMG (*Object Management Group*) que es un consorcio dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos, como ser: UML, definió un estándar de intercambio llamado XMI (*Metadata Intercharguer*), que está basado en el meta modelo de UML, el cual sirve para el intercambio de metadatos utilizando XML.

Existen distintos objetivos de las herramientas CASE, de acuerdo al trabajo de Mc Clure [3] algunos de los objetivos principales de las herramientas CASE son: Aumentar la productividad en el desarrollo; Dar calidad a los productos desarrollados; Reducir el costo del software; Automatizar los chequeos de errores; Acelerar el desarrollo de las aplicaciones; entre otros.

Cabe destacar que la carencia de la enseñanza de la IS como característica es recurrente, por lo que, en un entorno académico, los alumnos deben utilizar herramientas que estén pensadas para utilizar en el ámbito profesional y no sólo disponer de aquellas funcionalidades requeridas en el ámbito académico [5] [6][7].

1.3 Modelado UML

Un modelo es una representación de algo en un medio determinado. Los modelos capturan los aspectos de lo que se desea modelar con un punto de vista en particular y provee un nivel de abstracción sobre los aspectos menos importantes [8].

Un modelo posee dos aspectos importantes, información semántica y presentación visual por un lado y el contexto por otro. Los aspectos semánticos capturan el significado de las construcciones que son descriptos de forma separada, pero están relacionadas como parte de un modelo único y coherente. La presentación visual muestra la información semántica para que pueda ser usada por humanos. El contexto incluye información interna para que los modelos no carezcan de sentido. Es decir, que un modelo tiene un sentido distinto según el contexto en el que se utilice.

Un modelo de software se utiliza para diversos propósitos, entre ellos destacamos los más importantes: a) Para que un equipo interdisciplinario pueda entenderse; b) Para capturar y precisar requerimientos funcionales; c) Para Diseñar un sistema; d) Para crear productos de trabajo; y e) Para organizar grandes sistemas.

UML surge de la necesidad de tratar de entender, diseñar, navegar, configurar y mantener toda la información relativa a los sistemas de software. No está pensado como un método completo de desarrollo ni un marco metodológico de trabajo, sino como una herramienta que, en concurrencia con una metodología, brinde soporte al proceso de desarrollo en cada una de sus etapas. Por esto es que, los autores del estándar UML describen en sus libros que UML se crea con el objetivo de unificar experiencias anteriores sobre técnicas de modelado y así, incorporar las mejores

prácticas en un único estándar. Hoy por hoy, UML es soportado por la mayoría de las herramientas comerciales que se utilizan para modelar software [9].

2 Propuesta de una Herramienta CASE para Modelado

Toda herramienta CASE de Modelado, debería soportar algún lenguaje de modelado estándar que permita simplificar tanto la comunicación como la documentación del sistema objeto de estudio [10]. Es por lo que las tendencias metodológicas de los últimos años están altamente influenciadas por el enfoque de Orientación a Objetos (OO), permitiendo identificar las abstracciones necesarias para proponer una solución consistente con los elementos del dominio del problema. Para lograr esto, es necesario disponer de una herramienta de modelado que brinde soporte a la comunicación entre participantes y representar estos conceptos a lo largo proceso propuesto por la IS, y que brinde una semántica común [2] [11].

La propuesta de este trabajo, pretende optar por una herramienta CASE relacionada con la enseñanza de modelado UML en el marco de la IS, y que los posibles usuarios son alumnos de nivel universitario con, posiblemente, poca o nula experiencia en modelado de software y el uso de herramientas CASE.

El primer elemento a considerar como característica es la posibilidad de brindar soporte completo a UML y sus características a nivel notación, según propone González Génova [12]. En el mismo trabajo, los autores proponen 2 tipos de herramientas CASE para modelado según su notación: a) Las herramientas sintácticas, que solo permiten dibujar diagramas correctos en base a las reglas notación de UML, beneficiando a los usuarios con asistentes que evitan errores de diseño y, b) Las herramientas Semánticas que proveen mecanismos de validación para garantizar construcciones con sentido y coherencia.

2.1 Método de Evaluación

Si bien es sencillo encontrar información cualitativa sobre las herramientas objeto de análisis, muchas veces resultan insuficientes debido a que están atadas a promesas de sus fabricantes y hasta campañas publicitarias que no caracterizan las herramientas como el cliente espera. Es por esto que, es necesario definir una metodología que permita valorar de forma cuantificable, con el fin de reducir la ambigüedad inherente a la interpretación humana.

Existen numerosos trabajos que proponen métodos para evaluar herramientas CASE acorde a un proyecto o empresa. Entre ellos, Rojas et. al, propone considerar aspectos técnicos y organizacionales (cómo la empresa se prepara para asimilar el nuevo producto). Para considerar aspectos técnicos se utilizan un conjunto de métricas definidas que refieren a la cobertura de la herramienta respecto a etapas del ciclo de vida, metodologías soportadas, trabajo concurrente, plataformas, manejo de bases de datos, generación de código, ingeniería de reverso, horario de atención del soporte técnico, material de apoyo, y temas relativo al costo [13]. Todos estos puntos que se mencionan, sugieren un ranking para comenzar con la selección de la herramienta. Se considera que este punto propone una división general sin tener en

cuenta aspectos puntuales como, por ejemplo, tipos de bases de datos que utiliza o si posee soporte para generar código a un lenguaje en particular. Creemos que la primera división es interesante para luego especificar según criterios a definir en base al uso en particular. En otro trabajo, los autores definen un conjunto de características deseables agrupados en cuatro enfoques: Enfoque procedimental, definiendo como las herramientas utilizan las metodologías para guiar al usuario a través de un proceso de IS, el segundo aspecto es sobre el soporte al modelado que proveen las herramientas, evaluando la capacidad de modelado UML para apoyar la definición de un sistema. El tercer enfoque lo denominan apoyo al repositorio, refiriéndose a la capacidad de la herramienta de mantener trabajo colaborativo permitiendo niveles de robustez y consistencia de los datos y, por último, un enfoque funcional en base a como las herramientas ayudan a los usuarios a desarrollar un sistema de información con mayor o menor facilidad, utilizando criterios de versionado, navegación de diagramas, trazabilidad, entre otros [11].

Para poder evaluar con objetividad una solución informática es importante centrarse en aquellas características deseables que describen el tipo de herramienta, por ejemplo, para una herramienta CASE de modelado es de suma importancia que posea soporte para UML no así para una herramienta CASE para análisis de riesgos o de gestión de proyectos. En este sentido, la selección de una herramienta CASE puede verse afectada por la cantidad de aspectos a considerar [13].

Otro trabajo relacionado a la selección de Herramientas CASE de modelado utiliza criterios para definir el soporte UML desde el punto de vista de notación, es decir, que la herramienta provea de todos los métodos gráficos, sintácticos y semánticos para poder dibujar diagramas, darles significado a los componentes y darles sentido a las relaciones existentes, utilizando como base el estándar UML [12]. En este trabajo, los autores proponen definir un conjunto de características y contabilizar los números de respuesta afirmativas y negativas de cada herramienta. Si bien, se plantean la dificultad de poder catalogar todas las posibles características, proponen un método dinámico para ir confeccionando las preguntas solo cuando alguna herramienta objeto de análisis no la cubre. Un último punto a considerar para definir los criterios de evaluación, se basa en la selección de un enfoque metodológico y que soporte propone UML para tal. Para ello, se tomó en cuenta el trabajo de Kruchten quien propone utilizar la metodología denominada RUP (Rational Unified Process) [14].

Es importante generar valor para el cliente porque es una meta obvia en todo desarrollo de software. Por este motivo, RUP propone como característica que el desarrollo esté guiado por casos de uso (CU). Un caso de uso describe requisitos funcionales, en termino de como los actores interactúan con el sistema. Este enfoque "*guiado por casos de uso*" permite describir como los usuarios y la interacción, van a existir para conseguir un objetivo funcional, de forma temporal y mostrando las características para una funcionalidad determinada. En RUP está actividad es la más importante para formalizar la elicitación resultante en requisitos funcionales.

3 Propuesta de Evaluación

En base al estudio anterior proponemos el siguiente método de evaluación de herramientas CASE, el mismo está centrado en estas características deseables mencionadas en la sección anterior.

El objetivo principal es enmarcar y valorizar la herramienta ideal como punto de comparación definiendo como criterio de evaluación a un conjunto de características que a su vez dispondrán de ítems denominados *funcionalidades*, según si son aspectos funcionales a la IS o funcionales al usuario. Cada ítem identificado, podrá valorarse como *verdadero* o *falso* lo que dará un número finito que identifique en qué medida cumple las expectativas de las funcionalidades requeridas.

El total de puntos posibles estará dado por el total de funcionalidades posibles que tiene la herramienta ideal para el trabajo en cuestión, pudiendo utilizar un porcentual para identificar en qué grado cumple con las funcionalidades esperadas de una herramienta ideal, la cual es necesaria para una determinada actividad.

Por ejemplo, si la herramienta ideal pose 200 funcionalidades aceptadas y la herramienta arroja un total de 150, nos indicará que el acercamiento hacia el ideal en un 75% (aplicamos una regla de tres simples). La puntuación total posible, denominado *nivel de aceptación*, está dada por la cantidad de funcionalidades o especialidades requeridas para la evaluación.

3.1 Definición de los Criterios de Evaluación

La herramienta a evaluar será una herramienta CASE para modelado UML de tipo semántica, con utilidades para la enseñanza y aprendizaje de la IS. Para desarrollar el correcto análisis de las herramientas propuestas, se definieron criterios que consideramos acordes en base a las necesidades y dificultades detectadas con el objetivo de calificar aquellas herramientas que sirvan de forma referencial para obtener una puntuación objetiva. Escoger una herramienta CASE acorde a la aplicación que se requiere, supone mejoras durante el proceso de la IS. Para esto es necesario conocer ciertas características fundamentales. Para ello se proponen dos enfoques, uno funcional a la IS y otro funcional a la experiencia del usuario.

En base a los trabajos estudiados en el apartado anterior y desde el enfoque funcional a la IS proponemos las siguientes características a considerar para la definición de los criterios de evaluación: a) Enfoque procedimental; b) Apoyo metodológico; c) Soporte completo UML; d) Especificación de casos de uso; e) Facilidad de extensión del lenguaje; f) Modelado de Datos; g) Autogeneración de código; h) Ingeniería Inversa; i) Métricas; j) Apoyo a lenguajes formales; k) Soporte al modelado arquitectónico; l) Apoyo al modelado por capas; m) Apoyo a la enseñanza y aprendizaje de la IS. Desde el punto de vista de la experiencia del usuario, las características a considerar deberán ser las siguientes: a) Control de concurrencia; b) Versionado; c) Navegación; d) Manejo y compatibilización de diagramas; e) Visualización; f) Trabajo Online; g) Colaboración entre usuarios.

4 Criterios de Evaluación

En base a las características enumeradas en la sección anterior, se definen los siguientes conjuntos de *criterios de evaluación*. A cada criterio de evaluación se le definió un conjunto de *funcionalidades* esperadas en base al tema tratado. 1) Extensivo al proceso completo de desarrollo; 2) Adaptabilidad y flexibilidad; 3) Amigable para el usuario; 4) Producir documentación sólida; 5) Trabajo online y colaborativo; 6) Apoyo al proceso de enseñanza y aprendizaje; 7) Ingeniería Inversa y generación de código; 8) Apoyo metodológico e interpretación de invariantes OCL.
A continuación, se explica brevemente cada una de éstas.

4.1 Extensivo al Proceso Completo de Desarrollo

Muchas de las herramientas actualmente vigentes en el mercado no contemplan el proceso de ingeniería de software de forma integral, sino que se especializan en una parte específica del proceso. “La herramienta de desarrollo deberá soportar todas las etapas del ciclo de vida del desarrollo de un Sistema de Información, a saber: Análisis, Diseño, Implementación y Prueba” [15].

RUP propone un conjunto de fases y disciplinas que deberán recibir soporte UML en consecuencia. También es importante que la herramienta pueda dar un soporte completo de UML, adaptador a la norma, es decir la capacidad de las herramientas de construir todos los diagramas que propone este modelo, o por lo menos los más relevantes: el diagrama de casos de uso, que representa la funcionalidad o alcance del sistema; el diagrama de clases, que escribe la estructura de objetos y sus relaciones; el diagrama de interacción (secuencia o colaboración), diagramas de actividad para describir un proceso de negocio y en algunos casos, el diagrama de estados, que visualiza el ciclo de vida de un objeto.

Las características a evaluar son las siguientes: 1) Diagramas de casos de uso: Según RUP, centrar el desarrollo de software en casos de uso permite acercarse a la validez funcional propuesta por los requisitos descriptos durante el análisis; 2) Especificación de casos de uso: describir la interacción entre el actor y el sistema de forma detallada para promover una corrección funcional; 3) Diagrama de dominio: Permite describir por medio de un diagrama de clases UML simplificado la estructura abstracta del sistema, conociendo solo entidades y relaciones más importantes. Es un modelo de análisis que será tratado independiente al diagrama de clases; 4) Diagrama de Actividades: permite modelar procesos de negocio; 5) Diagrama de clases: al refinar el diagrama de dominio obtenemos un diagrama de clases UML que permite especificar relaciones, estado y comportamiento de las entidades que se transformaran en objetos software. Es un modelo de diseño; 6) Diagrama de secuencia: Es un modelo de diseño que permite relacionar las clases software con la arquitectura y el comportamiento con el objetivo de darle una vista dinámica al sistema que se está diseñando; 7) Aspectos notacionales UML; 8) Validación de modelo UML: Para mantener la semántica de los modelos, es necesario que se puedan realizar validaciones.

4.2 Adaptabilidad y Flexibilidad

Se valora la posibilidad de que la información de distintos proyectos, guardados por una herramienta en un formato específico, pueda ser leída y compatibilizada por otras herramientas CASE. Por lo que las características a evaluar son: 1) Compatibilidad XMI: Es una especificación que permite compatibilizar entre herramientas CASE que interpreten modelos UML; 2) Exportación de imágenes: Permite portar los diagramas a otras plataformas como por ejemplo editores de texto.

4.3 Amigable para el Usuario

Según los estándares ISO dedicadas a la calidad del Software (ISO 9241 e ISO 14598) la usabilidad es "grado en el que un producto puede ser utilizado por usuarios específicos para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un determinado contexto de uso". Es por este motivo que resulta sumamente subjetiva la evaluación de usabilidad. Por lo que las características a evaluar son las siguientes: 1) Ayuda en Línea: El usuario puede acceder a una ayuda indexada, actualizada e idioma propio; 2) Manual de Usuario: La herramienta tiene un manual de uso disponible desde el momento de su instalación; 3) Soporte Técnico Flexible: Disponibilidad de acceso al soporte técnico de forma.

4.4 Producir Documentación

Se deberá considerar un repositorio que permita centralizar, administrar y gestionar las versiones o estados de un proyecto en el que se requieren revisiones frecuentes. La documentación deberá ser personalizable y estar basada en diagramas, modelos.

4.5 Trabajo Online y Colaborativo

Consideramos para nuestro análisis la herramienta de Google denominada *Trends* [16] que muestra un resumen de tendencias sobre un determinado criterio de búsqueda en base a un plazo de tiempo determinado. *Google Trends* es una herramienta que proporciona información sobre las búsquedas que los usuarios ingresan en Google durante un periodo de tiempo y en un área geográfica determinada, es decir el volumen total de consultas para el término de búsqueda. Los datos de consulta se analizan con un método de muestreo ya que considera solo aquellas consultas realizadas un número significativo de veces. Esta herramienta sirve para predecir el futuro, pero se puede utilizar como herramienta para "Predecir el Presente". En estos términos, se puede lograr una predicción inmediata sobre temas de particular interés [17]. En base a tendencias analizadas, se deduce que durante los últimos años hubo una merma significativa en la cantidad de veces que se realizaron consultas relacionadas a estos conceptos de colaboración y trabajo on-line para con Herramientas CASE.

4.6 Apoyo al Proceso de Enseñanza y Aprendizaje

Para que una herramienta se adapte al proceso de enseñanza y aprendizaje debe contar con espacios específicos para el docente y para el alumno, con la posibilidad de realizar no solo trabajo de forma individual, sino también de forma colaborativa. Deberá tener también, el espacio necesario para realizar evaluación y seguimiento de trabajos [6][7].

4.7 Ingeniería Inversa y Generación de Código

Las herramientas que asisten al proceso de implementación, pueden generar código fuente. Si bien UML no es un lenguaje de programación, se puede usar para escribir programas, pero no posee la sintaxis y ni la semántica necesaria. Diferentes herramientas pueden generar código de tal manera que puedan importarse en un lenguaje de programación para luego utilizarse durante el proceso de implementación,

Generar e interpretar código existente hace que una herramienta CASE pueda participar activamente del proceso de Implementación. Para esto la herramienta debe lograr analizar diferentes lenguajes y debe ser escalable en cuanto a la diversidad de lenguajes existente. Las características a evaluar son las siguientes: 1) Generación de Código; 2) Ingeniería Inversa: Permite importar el código fuente para realizar el diagrama de clases correspondiente.

4.8 Apoyo Metodológico e Interpretación de Invariantes OCL.

Es importante que la herramienta provea soporte para alguna metodología tradicional de desarrollo de software para realizar trabajos de modelado en todas las fases principales: Análisis, Diseño e Implementación. De esta manera, el usuario deberá conocer en todo momento que rol está cumpliendo al momento de operar la herramienta. Las características a evaluar son las siguientes: 1) Soporte para identificación de Roles: deberá identificar en todo momento que rol está cumpliendo el usuario. Por ejemplo, si se está haciendo un diagrama de clases, podría indicarse que está trabajando en el diseño.

7 Conclusiones y Trabajos Futuros

En este trabajo se presentó un método de evaluación para las herramientas CASE teniendo en cuenta determinados criterios considerados en base a diversos análisis y estudios de investigación por parte de varios autores. Cada uno de los criterios que se propusieron, representan un conjunto preciso de funcionalidades o especialidades esperadas por un desarrollador de software. Se espera continuar en el estudio comparativo de diferentes herramientas CASE que se utilizó en el proceso de selección propuesto por Topper et al [18]. Este proceso consta de un conjunto de pasos bien definidos: 1) Realizar una revisión exhaustiva de las herramientas disponibles. 2) probar un pequeño grupo de una selección del paso anterior, probarlas

a través de un proyecto piloto o de una evaluación más detallada y 3) presentar una puntuación de las herramientas y seleccionar la de puntuación más alta. Mediante ello, se analizará el nivel de aceptación de las herramientas CASE de modelado más comunes en el mercado. Por último, se utilizará esta metodología para evaluar la integración propuesta en trabajos anteriores [6] [7].

References

1. Sommerville, I., & Galipienso, M. I. A. (2005). Ingeniería del software. Pearson Educación.
2. Pressman, R. S. (2010). Software engineering: a practitioner's approach. Palgrave Macmillan.
3. Mc Clure, C. O., & Ortega, J. M. (1993). Case: la automatización del software.
4. Battaglia, N., Neil C., Cardacci, D., De Vincenzi M., Martínez R. (2016, Septiembre). Evaluación y Seguimiento Durante el Proceso de Enseñanza y Aprendizaje del Modelado UML en Entornos Colaborativos. In V Workshop de Innovación en Educación en Informática (WIEI), Congreso Argentino de Ciencias de la Computación (CACIC).
5. Neil C., De Vincenzi M., Battaglia N., Martínez R. (2016). Herramientas Colaborativas Multiplataforma en la Enseñanza de la Ingeniería de Software. In XVIII Workshop de Investigadores en Ciencias de la Computación
6. Battaglia, N., Neil, C., De Vincenzi, M., & Martinez, R. (2016, Junio). UAICase: integración de un entorno académico con una herramienta CASE en una plataforma virtual colaborativa. In XI Congreso de Tecnología en Educación y Educación en Tecnología (TE&ET 2016).
7. Battaglia, N., Neil, C., De Vincenzi, M., Martínez, R., González, Dana. (2017, Junio). uCASE-CL: Aprendizaje Colaborativo de la Ingeniería de Software en Entornos Virtuales Ubicuos. In XII Congreso de Tecnología en Educación y Educación en Tecnología (TE&ET 2017).
8. Rumbaugh, J., Jacobson, I., & Booch, G. (2005). Unified Modeling Language Reference Manual, Second Edition, Addison-Wesley.
9. Watson, A. (2008). Visual Modelling: past, present and future. Online, Object Management Group, http://www. uml. org.
10. Larman, C. (1999). UML y Patrones. Introducción al análisis y diseño orientado a objetos. Ed. Pearson
11. Quintero, J. B., de Páez, R. A., Marín, J. C., & López, A. B. (2012). Un estudio comparativo de herramientas para el modelado con UML. revista universidad eafit, 41(137), 60-76.
12. Génova, G., Fuentes, J., & Valiente, M. (2006). Evaluación comparativa de herramientas CASE para UML desde el punto de vista notacional. Novática, 181, 59-64.
13. Rojas, T., Pérez, M., Grimán, A., Ortega, M., & Diaz, A. (2000). Modelo de decisión para soportar la selección de herramientas CASE. Revista de la Facultad de Ingeniería, UCV, 15(2), 117-144.
14. Kruchten, P. (2004). The rational unified process: an introduction. Addison-Wesley Professional.
15. Rojas, T., Pérez, M., Grimán, A., Ortega, M., & Diaz, A. (2000). Modelo de decisión para soportar la selección de herramientas CASE. Revista de la Facultad de Ingeniería, UCV, 15(2), 117-144.
16. Google Trends. (2017). Tendencias de Google. [online] Available at: https://www.google.com.ar/trends [Accessed 6 Jan. 2017].
17. Choi, H., & Varian, H. (2012). Predicting the present with Google Trends. Economic Record, 88(s1), 2-9.
18. Topper, A., Ouellette, D., & Jorgensen, P. (1994). Structure Methods Mergin Models. Techniques an CASE Mc Graw Hill.