ISSN: 1692-7257 - Volumen 2 – Número 22 - 2013
Revista Colombiana de Tecnologías de Avanzada

Recibido: 14 de julio de 2012
Aceptado: 28 de septiembre de 2012

# STATE OF THE ART OF EMBEDDED SYSTEMS FROM AN INTEGRATED PERSPECTIVE BETWEEN HARDWARE AND SOFTWARE

# ESTADO DEL ARTE DEL DESARROLLO DE SISTEMAS EMBEBIDOS DESDE UNA PERSPECTIVA INTEGRADA ENTRE EL HARDWARE Y SOFTWARE

**Ing. Rubén D. Sanchez Dams**

**Universidad de la Costa,** Líder del grupo de investigación GIACUC.
Profesor de tiempo completo programa de Ingeniería Electrónica.
Tel.: +(57) (5) 336 2273.
E-mail: rsanchez5@cuc.edu.co

**Abstract:** This paper deals with the studies in the research program "Development Methods for electronics", which aims to help companies in the electronics sector to more efficiently conduct their projects. This is a summary of the analysis of the state of art and theories applicable to the development of technological solutions that involve electronics. It exposes the tools originally developed for software engineering have begun to implement the development of embedded systems, and hardware systems, presenting proposals for problem formulation and solution methods applied to the domain of hardware projects and embedded systems..

**Keywords:** SysML, UML, SoC, embedded systems, system requirements, systems engineering.

**Resumen:** Este artículo trata sobre los estudios realizados en el programa de investigación “Métodos de desarrollo para electrónica”, el cual tiene como fin ayudar a empresas del sector de la electrónica a que realicen de forma más eficiente sus proyectos. Se muestra una síntesis del análisis del estado del arte y teorías aplicables al desarrollo de soluciones tecnológicas que involucren a la electrónica. Se expone como las herramientas desarrolladas originalmente para la Ingeniería del software se han comenzado a aplicar al desarrollo de sistemas embebidos, y sistemas hardware, presentando propuestas de formulación de problemas y métodos de solución aplicados al dominio de proyectos de hardware y sistemas embebidos.

**Palabras clave:** SysML, UML, SoC, sistemas embebidos, requisitos del sistema, ingeniería de sistemas.

## 1. INTRODUCCIÓN

En la actualidad, con la continua generación de nuevas aplicaciones software con mayores potenciales, se ha aumentado constantemente la exigencia del hardware en el cual se ejecutan tales aplicaciones. Lo que ha permitido el advenimiento de sistemas embebidos y hardware cada vez más potente y reducido (Vicente *et al.*, 2005). Este desarrollo ha sido motivado por la aparición de nuevas exigencias en los algoritmos, lenguajes de programación y aplicaciones con mayores potencialidades, orientando los esfuerzos de la industria hacia el diseño e implementación de Hardware capaz de ejecutar operaciones lógicas cada vez más rápida (Sánchez Dams, 2009a), con diseños reducidos y fabricaciones de menor costo (Sánchez Vítores, 2004). Cumplir con estos

objetivos requiere de herramientas y procesos adecuados que garanticen los resultados esperados.

En los últimos años la academia e industria ha dedicado recursos, tiempo y esfuerzo a la búsqueda de mejores métodos y procesos para el diseño e implementación de sistemas tecnológicos que involucran hardware. El desarrollo de las tecnologías de hardware ha sido paralelo al desarrollo de las tecnologías de Software, que en conjunto se han desarrollado inter-relacionadas y dependientes entre sí. En sus inicios ambos aspectos de la tecnología de la información se desarrollaban indistintamente como un solo dispositivo (Posadas Cobos, 2011), pero a medida que se comenzó a establecer el conocimiento nacieron dos nuevas disciplinas a partir de la ingeniería Eléctrica: la Electrónica y la Informática. La ingeniería electrónica se convirtió en el soporte físico (hardware), genérico que brinda la plataforma en donde se manifiesta, procesa y se trasmite la información. A su vez la informática se encargo de organizar lógicamente (software) la información, estableciendo las formas, pasos, y estructuras de la información. Solas las dos nuevas disciplinas aportarían muy poco, por un lado el hardware sólo brindaría la electrónica análoga, y por otro el software necesita de un hardware para funcionar.

En su desarrollo, la Ingeniería del software se encontró con retos que ha ido solucionando, generando lenguajes que a partir de las representaciones binarias ha ido abstrayéndose hasta las conceptualizaciones que se utilizan hoy en día a nivel de diseño y análisis. A su vez se han desarrollado procesos que describen y organizan el desarrollo de este tipo de productos. Actualmente esta forma de desarrollo de software se ha asentado, siendo común su estudio sistémico mediante asignaturas en la formación de un ingeniero de sistemas y computación (Booch *et al.*, 2005a), (Jacobson *et al.*, 2007), (Pressman, 2002). Por otro lado el desarrollo de la ingeniería electrónica se ha dado desde una perspectiva diferente, concretamente en la electrónica digital se han establecido fundamentos de la tecnología (basados en las compuertas lógicas), los cuales han servido de base de dispositivos más complejos que se encapsulan y a su vez sirven de base para generar nuevos componentes, y así en continua evolución (Null & Lobur, 2006). Los productos de hardware han venido incrementando su complejidad, para abordarlo se han desarrollado nuevas herramientas software que ayudan a describir este hardware, los llamados Lenguajes de Descripción de Hardware HDL, que facilitan el manejo de la complejidad actual (Posadas Cobos, 2011); en esta evolución el desarrollo de hardware ha venido incorporando procedimientos, métodos y estrategias de su disciplina hermana el software, en donde se han generado lenguajes de descripción y especificaciones, en conjunto con procesos que describen como organizar los esfuerzos y con qué fines, constituyéndose en una referencia para organizar la búsqueda de soluciones similares en el desarrollo de hardware.

Actualmente, estas dos disciplinas convergen también en muchos desarrollos tecnológicos aportando soluciones específicas desde sus perspectivas, de los cuales el ejemplo más común es el computador, en donde se desarrollo un hardware de disposición general, al que se le suministra por aparte un sistema operativo y un conjunto de programas de usuario. A pesar de lo anterior existen otros sistemas tecnológicos en donde la línea que separa el hardware y software es difusa, a esta categoría pertenecen los denominados sistemas embebidos. El presente artículo hizo un acercamiento al desarrollo conjunto de hardware y software (co-diseño), dentro de sistemas embebidos tomando los métodos y procesos de desarrollo de la ingeniería del Software.

## 2. FUNDAMENTOS DE UN DESARROLLO INTEGRADO DE HARDWARE Y SOFTWARE

Como punto de partida se toma el concepto: sistemas embebidos, que como “su denominación indica, es un sistema que encapsula en un dispositivo todo el hardware y software, normalmente en una sola tarjeta, para realizar un propósito específico. Estos están orientados a resolver problemas de tiempo real, por lo que su sistema operativo va encaminado a este fin. Además, usualmente su hardware es sencillo y compacto y cuenta sólo con lo necesario para realizar la finalidad para la cual se diseña” (Sánchez Dams, 2009b). Ellos funcionan con un programa que se considera un "Firmware", este se constituye de instrucciones de programa para propósitos específicos, grabado en una memoria de tipo no volátil (ROM, EEPROM, flash, etc.), que establece la lógica de más bajo nivel que controla los circuitos electrónicos de este tipo de dispositivo. Al estar integrado en la electrónica del dispositivo, el firmware es en parte hardware, pero también es software. Funcionalmente el firmware es el intermediario (interfaz) entre las ordenes que

recibe el dispositivo entre la electrónica de bajo nivel y el software de capas superiores (Sánchez Dams, 2009b).

La complejidad de estos sistemas ha venido aumentando generando nuevos desafíos, por lo que para abordarlos, se han reformulado procesos de desarrollo, adaptándose y combinando diferentes disciplinas (Debbabi at el., 2010). Como se mencionó el desarrollo de estos dispositivos se encuentra en una zona gris entre la electrónica (hardware) y la programación (software), últimamente (Vanderperren at el., 2008) los proyectos que afrontan su fabricación utilizan lenguajes empleados para el desarrollo de Software, como el Lenguaje de Modelado Unificado (UML). UML es un lenguaje estandarizado por la OMG para el análisis y diseño de aplicaciones, especificando la estructura y el comportamiento del sistema, definiendo una sintaxis abstracta subyacente, con una representación gráfica concreta (OMG, s. f.-a).

## 3. UNA NUEVA PERSPECTIVA PARA EL DISEÑO DE HARDWARE

Las exigencias actuales en el diseño de hardware requieren no sólo de técnicas y tecnologías de punta, sino también, de la aplicación de procesos organizados y lenguajes que permitan describir los objetivos comprometidos, y el diseño propuesto. Esta necesidad ha despertado un creciente interés en la aplicación de lenguajes de modelado y procesos de desarrollo en el diseño e implementación de hardware, tema sobre el cual algunos investigadores han abordado resultados innegables (Vanderperren *et al.*, 2008) y desarrollado aplicaciones específicas implementando métodos de desarrollo y lenguajes de modelado (Huang, Hsiung, & Shen, 2010).

Se postula que los métodos de desarrollos en la ingeniería del Software son en gran medida aplicables a la ingeniería de sistemas[1], entendida esta última como la aplicación de diferentes disciplinas para el estudio de la realidad y la implementación de sistemas complejos. Esto se sustenta en principalmente en dos aspectos: 1) Actualmente el principal lenguaje de descripción de sistemas SysML, parte de la especificación del lenguaje de modelado para desarrollo de software UML (Weilkiens, 2008), (Friedenthal *et al.*, 2011), (OMG SysML, s.f.), 2) Uno de los autores más influyentes en la teorización de la ingeniería del software, Ivar Jacobson, planteó sus fundamentos inicialmente a partir de las soluciones propuestas para sistemas embebidos o sistemas de sistemas en telecomunicaciones cuando él trabajaba en Ericson (*Interview with Ivar Jacobson «Ivar Jacobson on UML, MDA, and the future of methodologies»*, 2006).

Como resultado ha estas investigaciones, la industria y la academia, e incluso los propios creadores de entornos de desarrollo han enfocado esfuerzos a la consecución de buenas prácticas, metodologías y procesos para el diseño e implementación de hardware.

Esto ha dado como resultado que las herramientas se adapten cada vez mejor al desarrollo de hardware; los lenguajes de modelado UML y SysML actualmente tienen la posibilidad de integrarse junto a plataformas de modelado de hardware comúnmente usadas como MATLAB[2]/Simulink[3] y LabVIEW[4] («Artisan Studio - Products - Atego», s.f.).

Esta nueva perspectiva direcciona hacia futuro, la integración de las herramientas de diseño hardware y software, con los lenguajes de modelado y los entornos de trabajo. Que propenderán por la obtención de productos de mayor calidad en materia de hardware y programación de sistemas embebidos.

En los ítems siguientes se describió la trayectoria de estas herramientas, lenguajes, y como se ha dado comienzo hacia una nueva perspectiva de co-diseño e implementación de hardware o sistemas de hardware/software.

[1] Sistemas en este contexto y en general en el artículo se entiende como sistemas tecnológicos, que involucran como parte integral al hardware.

[2] MATLAB es un lenguaje de alto nivel y entorno interactivo que permite al usuario realizar tareas computacionales intensivas más rápido que con de programación tradicionales como C, C++ o Fortran (Mathworks, s. f.-a).

[3] Simulink es un entorno para simulación multi-dominio y diseño basado en modelado para sistemas dinámicos y embebidos. Este provee un entorno gráfico interactivo y un conjunto de bloques de librerías personalizable que permite diseñar, simular, implementar y probar una variedad de sistemas variantes en tiempo, incluyendo sistemas de comunicación, control, procesamiento de señales, procesamieto de video y procesamiento de imagen (Mathworks, s.f.-b).

[4] Herramienta de programación grafica, para ciencias e ingeniería, para desarrollos rápidos, de forma económica, que sirve de interface con elementos de medida, hardware de control, o analizadores de datos, permitiendo compartir y distribuir datos a través de un sistema («NI LabVIEW - Improving the Productivity of Engineers and Scientists», s. f.).

## 4. UML Y PERFILES ORIENTADOS A DESARROLLO DE HARDWARE

UML ha sido en mayor medida un lenguaje de modelado aplicado comúnmente al desarrollo de software, sin embargo, la OMG (sigla para Object Management Group) quienes desarrollaron y mantienen actualizado este estándar, ha dirigido la notación de UML para hacer de él un lenguaje de modelado de propósito general, usable en múltiples campos de aplicación. Aún así, en algunos contextos como el diseño de hardware, UML posee una semántica algo distante para la descripción adecuada de estos sistemas. Para tales contextos, la OMG desarrolla adecuaciones del estándar UML que adaptan el lenguaje a campos de aplicación especiales. A estos UML "personalizados" se les conoce como perfiles de UML.

Para el ámbito del desarrollo de hardware se han establecidos ciertos perfiles, entre los cuales se destacan el perfil de UML para sistemas en un solo Chip (SoC) y el Lenguaje para modelado de sistemas (SysML) (OMG, s. f.-b). En primer lugar, se define la especificación de UML 2.0 para Sistemas en un solo Chip (SoC). Este perfil introduce el diagrama estructurado dando importantes capacidades como la representación jerarquizada de módulos interfaces y puertos, elementos fundamentales de SoC. Así mismo específica roles de módulos y la representación de transferencia de información entre los módulos usando un único diagrama. En segundo lugar, pero no menos importante, se encuentra SysML, que es definido como un lenguaje de modelado grafico de propósito general para la especificación, análisis, diseño y verificación de sistemas complejos que deben incluir hardware, software, información, procedimientos, recursos, etc. En particular SysML, provee una representación gráfica para el modelado de requisitos del sistema, de comportamiento, de estructura y parámetros, los cuales son usados para integrarse con otros modelos de análisis de ingeniería (OMG, s. f.-b).

SysML es un lenguaje que evoluciona de UML 2.0 para ser aplicado a sistemas que puedan llegar a incluir hardware, software, información, procesos y persona y apoya la especificación, el análisis, el diseño, la verificación y la validación de sistemas complejos. A su vez SysML se hizo compatible con UML, el cual puede facilitar la integración de las disciplinas de la ingeniería de software y la ingeniería de sistemas (Sánchez Dams & Amaya Tejera, 2012). Este facilita la comunicación entre equipos heterogéneos (por ejemplo ingenieros mecánicos, eléctricos e ingenieros de software) que trabajan de forma conjunta para desarrollar un sistema. El lenguaje es efectivo en requisitos específicos, estructura, comportamiento, localización de elementos sobre modelos y restringe las propiedades del sistema al apoyo del análisis ingenieril. SysML es un subconjunto de UML, esto es entendido en el sentido que para el desarrollo de hardware no son necesarias algunas características del UML, que solo aplican al software (por ejemplo los diagramas de objetos), y hay otros que son redundantes para el hardware (diagramas de tiempos y diagramas de comunicaciones) (*Interview with Ivar Jacobson «Ivar Jacobson on UML, MDA, and the future of methodologies»*, 2006) (Rosenberg & Stephens, 2007). Algunos diagramas son derivados desde UML sin cambios significativos (diagramas de secuencia, máquinas de estado, casos de uso y paquetes), y otros son derivados con algunos cambios (diagramas de actividades, diagramas de bloques). A su vez para suplir necesidades específicas surgen dos nuevos diagramas, los diagramas de requisitos de los cuales se habla en la siguiente sección y los paramétricos descritos a continuación en conjunto con ciertas particularidades del lenguaje (Holt & Perry, 2007) (Herzogat el., 2005).

El diagrama paramétrico permite la captura de restricciones y características del sistema (rendimiento y fiabilidad), al igual que las relaciones entre las propiedades de los diferentes elementos del modelo. Estos introducen en el modelo de componente de UML la posibilidad de modelar restricciones lógicas, muestra restricciones paramétricas entre elementos estructurales, y es útil para análisis cuantitativo y de rendimiento. A partir del diagrama de componentes de UML 2.0 surge una adaptación denominada diagrama de ensamble, que tiene como propósito capturar los componentes del sistema, las interfaces y sus iteraciones. El diagrama de actividades también sufrió cambios incluyendo el soporte para la representación de sistemas continuos y discretos y para el control del tamaño y comportamiento de buffers en flujos de datos entrantes.

## 5. MODELADO DIRIGIDO POR LAS ESPECIFICACIONES DEL USUARIO UTILIZANDO SYSML

Los diagramas de casos de uso (CU) de SysML son una excelente forma de capturar las especificaciones del usuario. Estos diagramas en

SysML son derivados de UML 2.0 sin extensiones importantes, pero amplían el enfoque permitiendo modelar adicionalmente con un CU a otros sistemas, personas y hardware. Por lo anterior a continuación se centrará el análisis solo en los requisitos. Los requisitos en ingeniería, son parte fundamental tanto en el ciclo de vida de un sistema, como en la estructuración y elaboración de éstos. Cuando un proyecto de este estilo es ejecutado de manera ineficiente, se pueden llegar a presentar una serie de problemas.

Esta sección recomienda el enfoque dirigido por modelos propuesto en (Soares & Vrancken, 2008), para la ingeniería de requisitos, basado en casos de uso y requisitos de SysML. Los requisitos para un sistema, son la colección de necesidades que expresan los clientes, o las personas interesadas en algún negocio. Estas necesidades pueden ser divididas en categorías, siendo una clasificación la hecha de acuerdo a requisitos del usuario, del sistema, de infraestructura, etc.

La captura de estos es una actividad crítica durante el desarrollo de sistemas complejos, como lo son los sistemas embebidos y de telecomunicaciones, debido a que estos sistemas deben interactuar con otros software, sistemas, sensores y con personas. Requisitos de usuario bien escritos son fundamentales para las fases posteriores de un proyecto, lo son durante todo el ciclo de vida del sistema. En este escenario SysML lo que propone es representar gráficamente los requisitos del usuario, con la intención de que sean modelados mucho antes de que empiece la construcción de esquemáticos o la codificación, así es deseable en un proyecto abordar estas actividades de descomposición del sistema de manera prematura. SysML define una semántica que relaciona los requisitos con otros modelos creados durante el diseño del sistema. Los diagramas ayudan a organizarlos y muestran de manera explícita los tipos de relaciones, estandarizando la forma de especificarlos a través de una semántica definida que establece relaciones explícitamente mapeadas.

## 6. PROPUESTA DE DESARROLLO DE REQUISITOS HACIENDO USO DE SYSML

Desde una perspectiva comercial, implementar todos los requisitos sobre un solo sistema puede no ser atractivo, debido a una mala relación costo/beneficio, por las limitaciones de personal, tiempo, e incluso la presión del cliente y del mercado. Estas dificultades hacen que priorizar sea una actividad fundamental durante el proceso de la ingeniería de requisitos (RE). Priorizarlos es dar el orden en la cual se debería considerar su implementación, para lograr esto y el seguimiento de los mismos el estándar SysML dispone de relaciones de herencia, derivados, satisfacción, verificación, y refinamiento.

Otra herramienta muy útil es la tabla de requisitos, esta busca la trazabilidad de los mismos, la cual es definida en (Gotel & Finkelstein, 1994)[5] como la capacidad de describir y rastrear un requisito, en ambas direcciones, es decir, desde su formulación, a través de su desarrollo y especificación, hasta su implementación y uso subsecuente y viceversa, mediante su refinamiento en iteraciones de acuerdo a las fases del proyecto. Con las tablas, los requisitos son presentados de una manera ordenada y organizada con lo cual se facilita su trazabilidad durante todo el ciclo de vida.

En las siguientes figuras adaptadas de (Soares & Vrancken, 2008), se muestran un diagrama de requisitos y otro conceptual, que con ayuda de las herramientas que brinda SysML permiten estructurar los requisitos de un sistema.

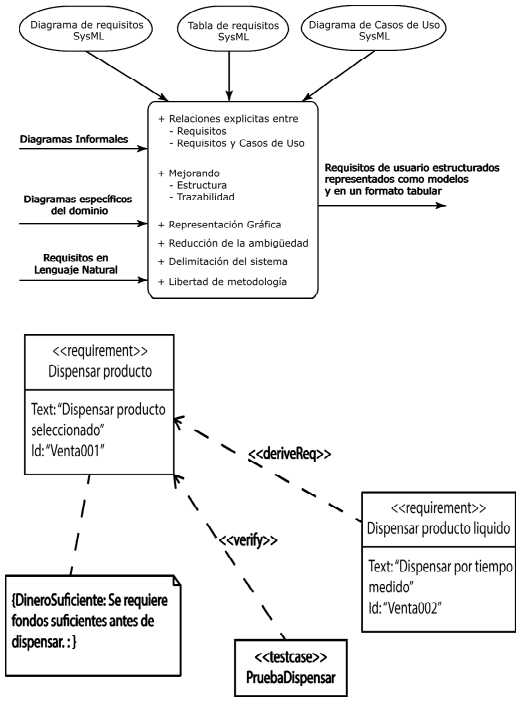


*Fig.1: Diagrama de requisitos, y Enfoque del modelo dirigido por la Ingeniería de Requisitos con SysML, Adaptada de (Soares & Vrancken, 2008)*

[5] Referencia secundaria tomado de (Soares & Vrancken, 2008)

Al hacer uso del modelo de requisito de SysML en un proceso, es posible direccionar la complejidad del sistema mediante actividades tempranas. Descomponer, es una tarea vital para lidiar con la complejidad, así los requisitos son descompuestos en otros más pequeños para organizarlos y agruparlos con la finalidad de explorar las características de un sistema definiéndolo adecuadamente. Para finalizar este apartado se enuncia un caso muy interesante para el desarrollo sistemas embebidos presentado por Doug Rosenberg, en la sección dedicada a los sistemas embebidos de su libro Iconix Process Roadmaps (Rosenberg, 2011), se resalta debido a que muestra de forma simplificada en el marco de un proceso la aplicación del estándar SysML en el desarrollo de un proyecto, partiendo de los requisitos del sistema.

## 7. CONCLUSIONES

Hay falta de consenso por metodologías reconocidas de amplia práctica, para el desarrollo de los sistemas embebidos[6], como lo hay para el software, o ingeniería Industrial con sus tendencias principales y procesos de desarrollos. En este sentido en el estado del arte se encuentran procesos de conocimiento explícito como Seis Sigma y RUP, o procesos de desarrollos con conocimiento tácito agrupados comúnmente como Ágiles, o cualquiera de los procesos a medio camino entre ellos como ICONIX o EUP, o marcos de trabajo para el mejoramiento continuo como CMMI (Jacobson, 2006) y PMI (Project Management Institute, 2009), no siendo así para los sistemas embebidos.

No hay una metodología ampliamente aceptada para el desarrollo integral de sistemas embebidos, por lo que es deseable una iniciativa, en lo posible incluyente y sintética, que aborde el desarrollo de sistemas electrónicos unificando posturas y métodos de desarrollo dispersos con un adecuado soporte de herramientas, y que esté disponible ampliamente en el estado del arte. A partir de las ideas tomadas desde la gestión de proyectos (Project Management Institute, 2009), esta iniciativa debería abarcar todo el ciclo de vida, y debido a sus similitudes en el proceso creativo en la construcción de soluciones hardware, software, o soluciones tecnológicas que integren ambas; es posible pensar en un núcleo comunes o desarrollar un proceso o metodología desde una perspectiva conjunta o complementaria desde el Software y hardware, que aborde de forma holística el desarrollo de sistemas embebidos.

Haciendo un análisis del trabajo presentado se concluye que todavía hay mucho por hacer y proponer sobre la temática estando en una etapa temprana las teorías sobre la formalización de los procesos de desarrollo para sistemas embebidos o soluciones electrónicas (Kuhn, 1996), afirmación que se sustenta en lo disperso de las referencias encontradas en cuanto a la temática, herramientas de soporte, proyectos realizado al respecto, falta de consenso (Sánchez Dams, 2012), entre otras, y al hecho de que las tecnologías estandarizadas derivadas de la Ingeniería del software como SysML y SoC son aun recientes (Sep 2007 y Jun 2006 respectivamente), si se comparan con el lenguaje UML que lo precede en aproximadamente 10 años (1997 OMG v1.0) (Booch at el., 2005b). De acuerdo a lo anterior se planteó que es factible el proponer a la comunidad académica y productiva la realización de estudios sobre la temática.

El autor adicionalmente ha realizado una caracterización de empresas MIPyMEs[7] del sector electrónico de la ciudad de Barranquilla, de donde se espera realizar divulgación sobre las actividades que vienen realizando estas organizaciones para realizar sus proyectos. Para trabajos futuros en la temática se pretende generar una propuesta metodológica extensible con base en las buenas prácticas encontradas en el estado del arte y a al diagnostico hecho a las empresas del sector en el que se ha realizado el estudio, ajustadas a la realidad comercial de ellas. En una segunda etapa de este mismo trabajo se pretende avanzar en el planteamiento y adecuación posterior de la mencionada metodología desde la perspectiva SEMAT (Bertrand at el., 2009), describiendo la metodología mediante el uso de prácticas separadas y auto contenidas, que faciliten su asimilación por una organización.

## 8. RECONOCIMIENTOS

Gracias a la empresa Bermit en donde se hizo el primer análisis de la temática y la identificación de su proyección como tema de investigación, y a los pupilos, Zhoe Comas y Ángel Mejía, por la ayuda en la búsqueda bibliográfica, y en especial a Heyder Páez, y Nazhir Amaya quienes adicionalmente ayudaron en la revisión final.

[6] Análisis registrado en (Rubén Darío Sánchez Dams, 2012) de la cual se concluye la afirmación.

[7] Acrónimo de "micro, pequeñas y medianas empresas"

## REFERENCIAS


Artisan Studio - Products - Atego. (s. f.). Recuperado octubre 28, 2010, a partir de http://www.atego.com/products/artisan-studio/

Bertrand, M., Jacobson, I., & Soley, R. (2009, 2010/01). *Software engineering method and theory – a vision statement*. Recuperado a partir de http://blog.paluno.uni-due.de/semat.org/wp-content/uploads/2012/03/SEMAT-vision.pdf

Booch, G., Rumbaugh, J., & Jacobson, I. (2005a). *Unified Modeling Language User Guide, The (Addison-Wesley Object Technology Series)* (2.ª ed.). Addison-Wesley Professional.

Booch, G., Rumbaugh, J., & Jacobson, I. (2005b). *Unified Modeling Language User Guide, The (Addison-Wesley Object Technology Series)* (2.ª ed.). Addison-Wesley Professional.

Debbabi, M., Hassaïne, F., Jarraya, Y., Soeanu, A., & Alawneh, L. (2010). *Verification and Validation in Systems Engineering: Assessing UML/SysML Design Models*. Springer-Verlag New York Inc.

Friedenthal, S., Moore, A., & Steiner, R. (2011). *A Practical Guide to SysML, Second Edition: The Systems Modeling Language* (2.ª ed.). Morgan Kaufmann.

Gotel, O. C. Z., & Finkelstein, A. C. W. (1994). An analysis of the requirements traceability problem. *Requirements Engineering, 1994., Proceedings of the First International Conference on* (pp. 94–101). IEEE.

Herzog, E., Pandikow, A., & Syntell, A. B. (2005). SysML–an Assessment. *Syntell AB, SE*, *100*, 55.

Holt, J., & Perry, S. (2007). *SysML for Systems Engineering*. IET.

Huang, C. H., Hsiung, P. A., & Shen, J. S. (2010). UML-based hardware/software co-design platform for dynamically partially reconfigurable network security systems. *Journal of Systems Architecture*, *56*(2-3), 88–102.

*Interview with Ivar Jacobson «Ivar Jacobson on UML, MDA, and the future of methodologies»*. (2006). Recuperado de http://www.infoq.com/interviews/Ivar_Jacobson

Jacobson, I. (2006, octubre 24). Ivar Jacobson on UML, MDA, and the future of methodologies. Recuperado a partir de http://www.infoq.com/interviews/Ivar_Jacobson#

Jacobson, I., Rumbaugh, J., & Booch, G. (2007). *El Lenguaje Unificado de Modelado, Manual de Referencia, 2/ed.* (2nd. ed.). Pearson Education.

Kuhn, T. S. (1996). *The structure of scientific revolutions Third Edition* (3.ª ed., Vol. 2). University of Chicago press.

Mathworks. (s. f.-a). MATLAB - The Language Of Technical Computing. Recuperado octubre 28, 2010, a partir de http://www.mathworks.com/products/matlab/

Mathworks. (s. f.-b). Simulink - Simulation and Model-Based Design. Recuperado octubre 28, 2010, a partir de http://www.mathworks.com/products/simulink/

NI LabVIEW - Improving the Productivity of Engineers and Scientists. (s. f.). Recuperado noviembre 4, 2010, a partir de http://www.ni.com/labview/

Null, L., & Lobur, J. (2006). *The essentials of computer organization and architecture* (Second.). Jones and Bartlett Publishers, Inc.

OMG. (s. f.-a). Object Management GroupTerms and Acronyms. Recuperado octubre 28, 2010, a partir de http://www.omg.org/gettingstarted/terms_and_acronyms.htm

OMG. (s. f.-b). UML Profile Specifications. Recuperado octubre 28, 2010, http://www.omg.org/technology/documents/profile_catalog.htm

OMG SysML. (s. f.). Recuperado octubre 28, 2010, a partir de http://www.omgsysml.org/

Posadas Cobos, H. (2011). *Estimación de prestaciones para Exploración de Diseño en Sistemas Embebidos Complejos HW/SW*. Universidad de Cantabria. Recuperado de: www.tdx.cat/bitstream/10803/32204/1/TesisHPC.pdf

Pressman, R. S. (2002). *Ingeniería del Software - Un Enfoque Practico 5 Edición*. McGraw-Hill Companies.

Project Management Institute. (2009). *Guia De Los Fundamentos Para La Direccion De Proyectos/A Guide to the Project Management Body of Knowledge (PMBOK Guide): Official Spanish Translation* (4.ª ed.). Project Management Inst.

Rosenberg, D. (2011). *Iconix Process Roadmaps: Step-by-step Guidance for SOA, Embedded, and Algorithm-intensive Systems*. Fingerpress.

Rosenberg, D., & Stephens, M. (2007). *Use Case Driven Object Modeling with UMLTheory and Practice*. Apress.

Sánchez Dams, Rubén D., & Amaya Tejera, N. J. (2012, octubre). Un acercamiento a la Ingeniería de requisitos. *CUCtrónica*, *Segundo*.

Sánchez Dams, Rubén Darío. (2009a). Entornos de Desarrollo Para Tecnologías Informáticas y Electrónicas, Vistas Desde el Microcontrolador. *Revista Institucional Alejandria, Colombia*, *2*, 33–38.

Sánchez Dams, Rubén Darío. (2009b). *Controlador lógico programable. Una mirada interna* (Primera ed.). Barranquilla, Colombia: Editorial Universitaria de la Costa. EDUCOSTA.

Sánchez Dams, Rubén Darío. (2012). *Metodología ágil estandarizada para el desarrollo o ejecución de proyectos de sistemas embebidos, propuesta para empresas de la ciudad de Barranquilla*. Monografía, Barranquilla, Colombia.

Sánchez Vítores, R. (2004). Aplicaciones de los sistemas embebidos. *Técnica industrial*, (1), 24–27.

Soares, M. S., & Vrancken, J. (2008). Model-driven user requirements specification using SysML. *Journal of Software*, *3*(6), 57.

Vanderperren, Y., Mueller, W., & Dehaene, W. (2008). UML for electronic systems design: a comprehensive overview. *Design Automation for Embedded Systems*, *12*(4), 261–292.

Vicente, C., Toledo, A., Fernández, C., & Sánchez, P. (2005). Generación Automática de Aplicaciones Mixtas Sw/Hw mediante la Integración de Componentes COTS. Recuperado a partir de repositorio.bib.upct.es/dspace/bitstream/10317/.../gaa.pdf

Weilkiens, T. (2008). *Systems Engineering with SysML/UML: Modeling, Analysis, Design* (1.ª ed.). Morgan Kaufmann.