METODOLOGÍAS DE DESARROLLO DE SOFTWARE, UN ENFOQUE A ROBOTS MÓVILES

Nelson Londoño Ospina[1]

[1] Ing. Electrónico, PhD (c) en Automática e Informática Industrial, Universidad del Valle, Colombia. Profesor Tiempo completo vinculado al Departamento de Ing. Eléctrica, y miembro del grupo de investigación en manejo eficiente de la energía GIMEL, categoría A Colciencias. Facultad de Ingeniería, Universidad de Antioquia, Medellín-Colombia, nlondono@udea.edu.co.

RESUMEN

Se presenta una visión general de las disciplinas relacionadas con el estudio de las arquitecturas software para robots y se evidencia la interdisciplinariedad y complejidad que exige. Se hace una revisión de los métodos aplicados a los sistemas robots móviles como motivación para proponer un proceso sistemático que propicie salvar la brecha entre el ingeniero en robótica (quien concibe la arquitectura) y el ingeniero de software (quien desarrolla el sistema).

Palabras clave: robótica, arquitecturas software, metodologías software.

Recibido: 15 de Abril de febrero de 2009. Aceptado: 19 de Junio de 2009
Received: April 15, 2009 Accepted: June 19, 2009

SOFTWARE DEVELOPMENT METHODOLOGIES, AN APPROACH TO MOBILE ROBOTS

ABSTRACT

A general overview of the fields related to study of robot software architecture is presented, making noticeable its demanding complexity and interdisciplinary. A general assessment of the methods applied to mobile robot systems is made in order to encourage a systematic process that enables the gap shortening between the robotics engineer (who conceives the architecture) and the software engineer (who develops the system).

Keywords: *robotics, software architectures, software methodology*

1. INTRODUCCIÓN

Tradicionalmente, los sistemas de control, y específicamente los sistemas robot, han sido concebidos y diseñados con base en criterios propios de sus creadores y en cada caso particular, se opta por la representación y documentación de la arquitectura diseñada para robots, de acuerdo a criterios del autor o grupo de diseñadores que la propone.

La importancia que cobra cada día el tema de la robótica en general y la robótica móvil en particular, no sólo como dispositivo de solución de problemas en diferentes áreas del conocimiento a nivel de investigación, docencia y aplicaciones industriales, sino como laboratorio de experimentación por excelencia, ha motivado el estudio de la problemática asociada a las arquitecturas software para robot y sus métodos de desarrollo.

Al hacer una evaluación detallada de las metodologías aplicadas a los sistemas robots, se evidenció la necesidad de clarificar los alcances y problemáticas ligadas a las arquitecturas software para robots. Se propone, en este artículo, una visión general que llevó, en el marco de una tesis doctoral, a proponer las estrategias de diseño, aplicable a sistemas robots móviles, considerando los elementos importantes de una arquitectura software.

2. CONCEPTOS GENERALES

Los sistemas robots y afines, se conciben como competencia de un área del conocimiento que requiere de la participación de muchas disciplinas. Actualmente, para cubrir los diferentes aspectos y temáticas propias de los sistemas robots, se acude a disciplinas tan diversas como: electrónica, eléctrica, mecánica, informática, sociología, biología, psicología, y disciplinas que incluyen los sistemas multi-robots [1]. No obstante, en primera instancia, los sistemas robots son tradicionalmente competencia directa de los ingenieros mecánicos, electrónicos y de sistemas. En la Figura 1, se ilustra un esquema general, en donde se presenta los temas y sus derivaciones. Como se observa, por ejemplo, la concepción metodológica de una arquitectura software para robots móviles requiere, fundamentalmente, de la participación de dos áreas del conocimiento: arquitecturas software para robots y metodología orientadas a objetos, las cuales a su vez son derivadas de disciplinas más generales (robot, sistemas software, metodologías y formalismos).

Cada uno de los bloques presentado en la Figura 1, obedece a una disciplina básica, cuyos objetivos y orientación esta claramente definidos, pero que al fusionarse con otras, resuelven problemáticas propias de otras disciplinas, o crean una nueva línea de investigación y desarrollo hibrido de dicha fusión. A continuación se explica, de forma general, cada uno de estos bloques.

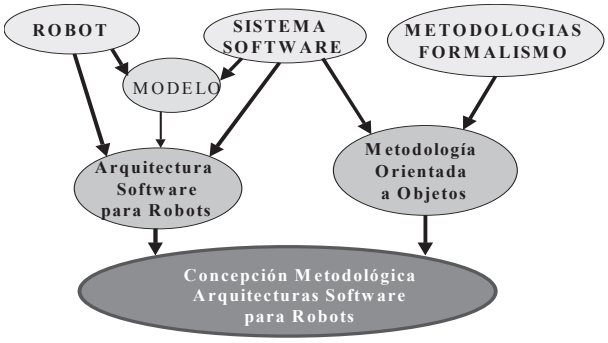

Figura 1. Esquema general de los aspectos más relevantes ligados la concepción metodológica de arquitecturas software para robot.

3. SISTEMAS ROBOTS: CONCEPTOS GENERALES

Definiciones: Son muchas las interpretaciones y definiciones que se han dado a los sistemas robots, bajo las cuales caben desde una aspiradora, en algunos contextos, hasta sistemas altamente sofisticados e inteligentes. En general, es un sistema electromecánico, dotado de sensores y actuadores para realizar actividades similares a las realizadas por seres vivos, con un sistema de control y capacidad de decisión que le permite realizar tareas automáticas o autónomas, dependiendo de su grado de versatilidad y complejidad.

Propósitos generales: Al desarrollar un sistema robot, se espera alcanzar los siguientes objetivos: "Se pretende conseguir un robot que sea capaz de comportarse de forma adecuada, inmerso en un entorno no estructurado, sin intervención humana directa (autosuficiencia) y realizando una tarea para la cual ha sido construido. El robot debe ser capaz de generar sus propias leyes de control (autonomía), siendo capaz de producir y regenerar

sus propios componentes software y/o hardware" [3]

Interdisciplinariedad: La estrecha relación que une la robótica con el estudio y explicación de los comportamientos humano y animal [4], [5], [6], [7], y su aplicación en la creación de mecanismos artificiales que imiten dichos comportamiento, ha exigido que la robótica trabaje estrechamente con especialistas de muchas áreas del conocimiento, propiciando un trabajo interdisciplinario y motivando la participación y cooperación de especialistas en diversas ramas de la ciencia como electrónica, sistemas, mecánica, psicología, biología, etc.

Clasificación: La robótica actualmente se clasifica en dos grandes líneas de trabajo diferenciadas claramente de acuerdo a sus características de aplicación: robótica manipuladora y robótica móvil [8], [9], [53]. A su vez, como se sabe, cada una de las anteriores se subdivide en otras tantas clasificaciones de acuerdo a sus características físicas, de aplicación, de control, etc. [10], [11], [12], [13], [3], [14].

En la Figura 2. Se lustran las líneas y características más generales de los robots, donde se evidencia la interdisciplinariedad y complejidad de estos sistemas.

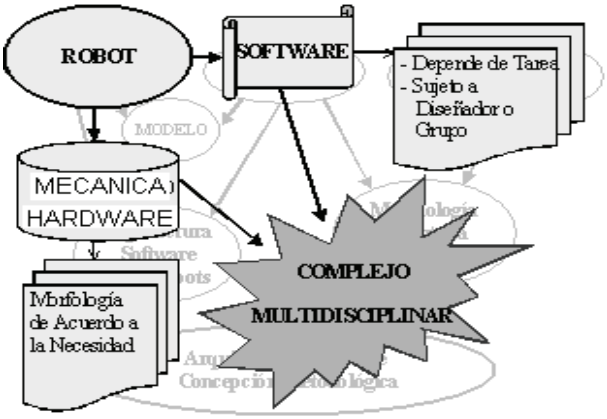

Figura 2. Sistema Robot.

Arquitecturas de Control para Robots: Una arquitectura para robots es un conjunto de módulos y elementos de software y hardware involucrados en su control. El desarrollo e implementación de los módulos de hardware y la interconexión entre ellos y los elementos de software definen una arquitectura [15], [16].

Cuando el hardware de un sistema robot está completamente definido, el problema de las arquitecturas se restringe al diseño, análisis, validación y chequeo del sistema software. En [3], se plantea que: "En realidad una arquitectura queda establecida cuando se consigue realizar una abstracción del hardware subyacente en términos de definición de una 'máquina virtual' o modelo de programación".

La arquitectura de los sistemas robots ha sido tema de investigación permanente y ha obedecido a diferentes enfoques: desde concepciones eminentemente conexionistas [17], pasando por la que se ha orientado al estudio y emulación de criaturas [5], hasta la emulación de la inteligencia humana y sus tendencias [17], [18]. Igualmente, ha estado sujeto a estudios bajo concepciones psicológicas, neurofisiológicas y etimológicas [4] [19]. A pesar que, en general, las arquitecturas se han planteado desde una concepción software, hay algunas propuestas que cubren tanto el aspecto hardware como el software [20], [21].

4. SISTEMAS SOFTWARE – INGENIERIA DE SOFTWARE

En la ingeniería del software el propósito consiste construir productos software o mejorar los existentes, para ello se han propuesto *"procesos de desarrollo*" los cuales deben proveer un enfoque para la asignación de tareas y responsabilidades que aseguren la producción de software de alta calidad y que satisfaga las necesidades de los usuarios finales. Los procesos de desarrollo son un conjunto de normas y actividades necesarias para transformar los requisitos de usuario en un sistema de software. Según [22] "Un proceso define quién está haciendo qué, cuándo, y cómo alcanzar un determinado objetivo"

Son muchos los enfoques y desarrollos que se han documentado, tendientes a proponer un proceso de desarrollo de software estándar [23], [24], [25], [26], [27]. Algunos de estos han sido ampliamente adoptados por la comunidad científica.

En general, una propuesta de desarrollo de software, proporciona una guía para ordenar las actividades de un equipo, es un proceso que dirige las tareas de cada desarrollador por separado y del equipo como un todo, especifica además los artefactos (piezas de información tangible) que

deben desarrollarse y ofrece criterios para el control y la medición para una gran variedad de sistemas de software [28] (Figura 3.)

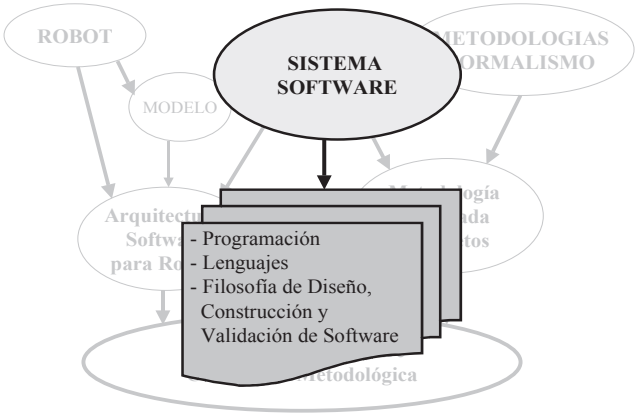

Figura 3. Sistema Software

5. MODELOS

El concepto de modelación esta relacionado con la utilización de herramientas que permitan validar un proceso o sistema, mediante la utilización de mímicos, ecuaciones o sistemas virtuales que cumplan con mayor o menor nivel de complejidad, las características del original. Para el caso de los sistemas robots (Figura 4) y, particularmente de las arquitecturas software para robots, es fundamental contar con elementos que representen el comportamiento de sus dispositivos, su interacción, y su funcionalidad. La relación entre estos elementos conforman una arquitectura software que, como se ilustra en la Figura 1, se deriva de disciplinas más generales.

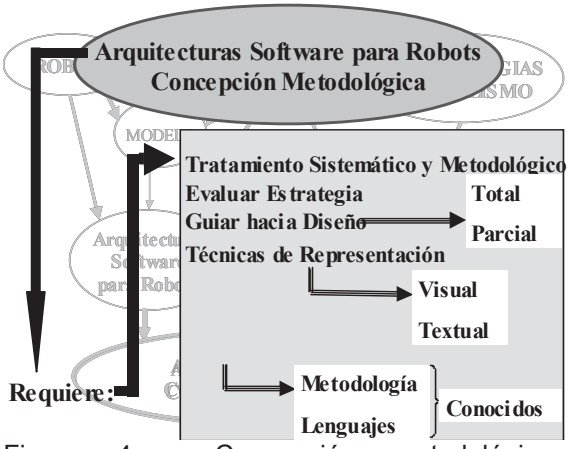

Figura 4. Concepción metodológica de arquitecturas software para robots

6. METODOLOGÍAS

"Se define metodología como un conjunto de métodos. A su vez, método se define como un grupo de reglas a seguir para solucionar un problema" [2].

Una metodología en general, consta de las siguientes partes fundamentales [29]:

Una estructura semántica y un esquema de notación. Esto es a lo que se le denomina "lenguaje de modelado".
Un proceso de desarrollo. Un conjunto de actividades de trabajo secuenciales. Describe las actividades que gobiernan el uso de los elementos del lenguaje.
Un conjunto de objetos de trabajo. Resulta de la aplicación de estos en una secuencia definida de actividades.

A su vez, como se ilustra en la Figura 5, las Fases de desarrollo de un proceso son fundamentalmente cuatro [29]**:**

Análisis**:** Identificación de las características esenciales de todas las posibles soluciones correctas.
Diseño**:** Adicionar elementos al análisis que define una solución particular con base en la optimización de algún criterio.
Implementación (Translación)**:** Crear un ejecutable. Es la parte más destacable del diseño.
Validación (Chequeo o prueba)*:* Verifica que la translación sea equivalente al diseño.

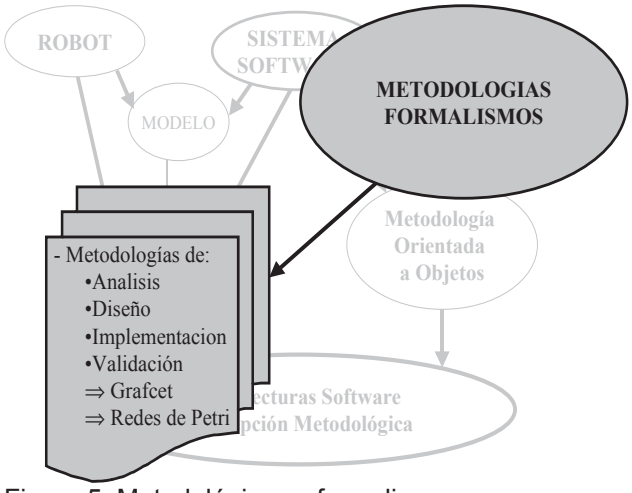

Figura 5. Metodológicas y formalismos

6.1 Metodologías orientadas a objetos

En el dominio de los sistemas software, no se concibe hoy en día tratar un proyecto de gran envergadura sin optar por una metodología para su concepción. Son muchas las propuestas que tratan los temas propios de una metodología encaminada al análisis, diseño, especificación y validación de sistemas, las cuales, por sus características, son válidas en muy diversos campos de aplicación. Muchas de estas metodologías se clasificadas como Orientadas a Objetos [54], [30], [31] que las hace mas confiables, flexibles, mantenibles y capaces de evolucionar, para satisfacer requerimientos de cambio. Adicionalmente, gran parte de ellas han sido concebidas para cubrir una o varias etapas del ciclo de desarrollo, lo que ha motivado la gran cantidad de literatura y enfoques a este respecto[1]. La Figura 6, ilustra de forma general, las opciones que ofrecen diferentes metodologías y métodos con este enfoque.

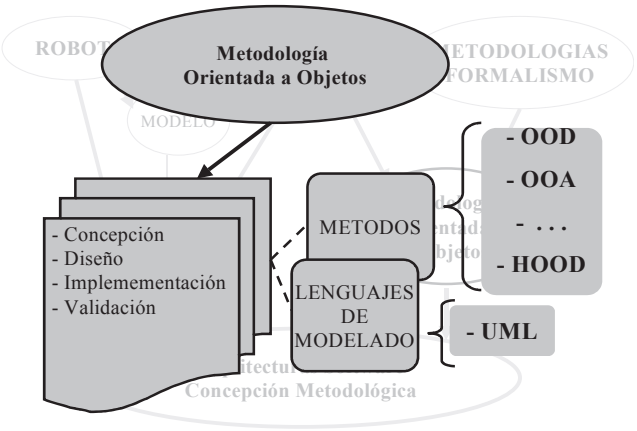

Figura 6. Metodológicas Orientadas a Objetos

Mediante las metodologías orientadas a objetos, se pretende crear sistemas mas confiables, flexibles, mantenibles y capaces de evolucionar que satisfagan requerimientos de cambio. Para ello, se han propuesto métodos que cubre las diferentes etapas, lo que a propiciado que en la literatura, sean comunes los acrónimos: OOD (Object Oriented Design), OOSA (Object Oriented System Análisis), OMT (Object Oriented Technique), OOA (Object Oriented Analisis), HOOD (Hierarchical Object Oriented Design), OOSD (Object Oriented Structured Design), RDD (Responsability-Driven Design), OOSE (Object Oriented Software Engineerin), UML (Unified Modeling Languaje), etc. [30], [54], [32], Sistemas multiagentes [33] [34].

Para este estudio, se han evaluado un poco más en detalle tres métodos que a nuestro juicio pueden cubrir gran parte de las etapas de Análisis, Diseño, Implementación y Validación y que no excluyen la evaluación de otros métodos o técnica durante el desarrollo del proyecto. Estos son: HOOD, UML y PUD, cada uno de los cuales ofrece características importantes para tratar el problema que nos ocupa.

6.2 HOOD

HOOD (Hierarchical Object Oriented Design) [35], [36], Es el resultado de la fusión de los métodos conocidos como “Máquinas Abstractas” y “Diseño Orientado a Objetos” y ha sido adaptado para integrar la orientación a objetos y los conceptos y notaciones de software avanzado.

HOOD permite la identificación de arquitecturas de software y guía naturalmente los diseños detallados, donde la operación de los objetos son diseñados usando un Lenguaje de Descripción de Programa (PDL), inspirado en lenguaje de programación ADA. Esta descripción de diseño detallado puede ser refinada sucesivamente en descripción de lenguaje objetivo, hasta un punto donde se puede generar el código. Cada paso produce piezas de texto que refleja la actividad del diseño [36], [37].

HOOD es un método de diseño arquitectónico que ayuda al desarrollador a particionar el software en módulos con interfaces definidas, implementando directamente o dividiendo en módulos de más baja complejidad. Soporta diseño basado en objetos y diseño Orientado a Objetos. Fue concebido para desarrollar actividades que ocurren en el mismo momento como: Integración con el análisis de requerimientos, desarrollo concurrente de partes independientes, generación y chequeo de código automático, soporte cliente-servidor y pos-particionado [36].

Uno de los atractivos más importantes de HOOD es que permite describir el sistema software mediante una representación gráfica, como se ilustra en la figuras 7 y 8, y ofrece una serie de pasos como, diagrama jerárquico y descripción textual, que facilitan la especificación y subdivisión clara del sistema.

[1] [31] presenta estudio, clasificación y comparación de muchas de estas metodologías que pueden complementar este tema.

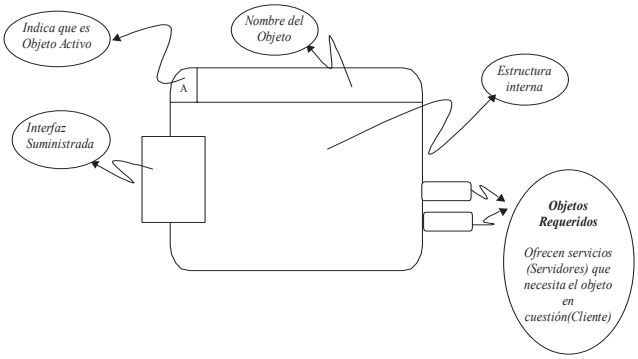

Figura 7. Objeto típico HOOD

6.3 UML (Unified Modeling Language)

UML es un lenguaje de propósito general para el modelado orientado a objetos, formal, estándar y con gran nivel de aceptación actualmente. Ofrece alto nivel de abstracción y permite la posibilidad de visualizar, especificar, diseñar, validar, construir y documentar los sistemas que involucren complejidad y gran cantidad de software [38], [39], [40].

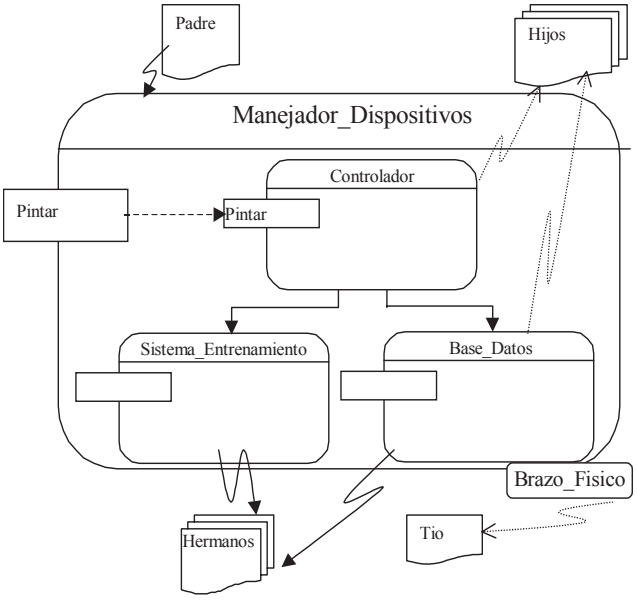

Figura 8. Diagrama de Jerarquías HOOD

"UML es un lenguaje gráfico con una semántica y sintaxis bien definida, combina notaciones provenientes de: Modelado Orientado a Objetos, Modelado de Datos, Modelado de Componentes, Modelado de Flujos de Trabajo (Workflows)". "El objetivo de UML es describir cualquier tipo de sistema en términos de diagramas orientados a objetos" [41].

Siendo un lenguaje de modelado, con componente gráfica, ofrece una gran variedad de diagramas que llevan fácilmente el proceso de diseño y representación de un proyecto y que permiten, entender el sistema aún cuando su nivel de complejidad sea alto.

Una muy buena representación gráfica de UML se ofrece en [41] y que, para efectos de claridad, se ilustra en la Figura 9.

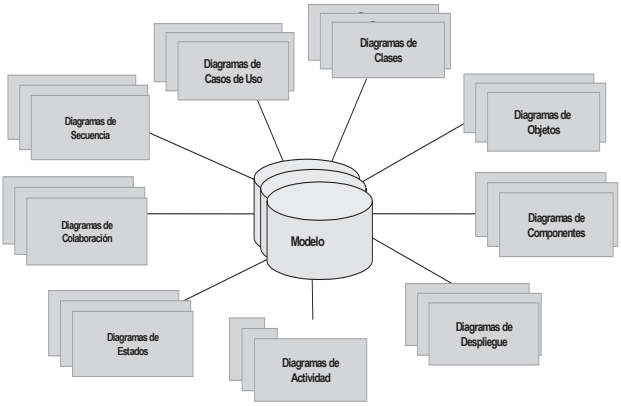

Figura 9. Representaciones gráficas en UML

6.4 PUD (Proceso Unificado de Desarrollo)

El PUDS [42], [43], escrito por los desarrolladores de UML. Surgió de la unificación de las tres metodologías basadas en el paradigma orientado a objetos, aunque ha recibido aportaciones de muchas fuentes como:

OOSE: Object Oriented Software Engineering (Casos de Uso) creado por Ivar Jacobson.
Booch (Diseño): Creado por Grady Booch.
OMT: Object Modeling Technique (Análisis) creado por James Rumbaugh [55].

PUDS se apoya en el estándar UML y es la norma de referencia en materia de métodos de desarrollo orientado a objetos por lo cual ha cobrado mucha importancia pues integra las múltiples facetas del desarrollo de software. Ello ha convertido a PUDS en una guía obligada de quienes pretenden proponer metodologías y procesos de desarrollo de software de diferentes niveles de complejidad [44].

El proceso Unificado se define como: un proceso de desarrollo de software que se adapta a proyectos que varían en tamaño y complejidad; adopta a UML como uno de los componentes clave que propicia la construcción de modelos, promueve

la creación de una base de conocimiento a la que pueden acceder los diferentes actores del proyecto y contiene guías, patrones, y herramientas para el desarrollo de las actividades más importantes.

Según sus creadores, "El PUDS es un proceso dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental. Está basado en componentes y utiliza el nuevo estándar de modelado visual UML" [28], [45].

Una de las herramientas que más ha ayudado a la consolidación y mayor aceptación del paradigma OO, es precisamente su versatilidad para desarrollarse apoyado en modelos [46]. La figura 10 *pr*esenta la relación de los modelos típicos que apoyan cada una de las fases del proceso de desarrollo.

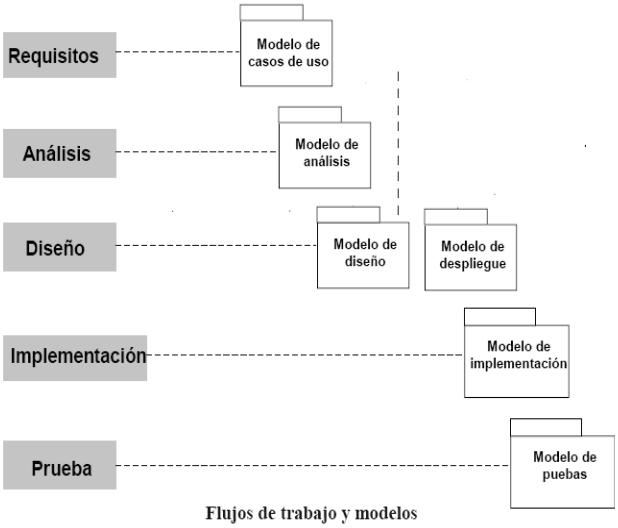

Figura 10. Relación de los modelos típicos - las fases del proceso de desarrollo

7. METODOLOGÍAS PARA ROBOT

Específicamente para el diseño de sistema de control (arquitecturas) para robots, son pocas las referencias, una de las primeras, se encuentra en [47], quien propone un procedimiento sistemático para la implementación de una arquitectura destinada al control autónomo de una excavadora:

Definición del problema. Define los objetivos principales del sistema, y se indican los recursos necesarios.
Adquisición del conocimiento. Aprendizaje y consecución de objetivos tomados de un operador del sistema.
Definición de los requisitos del sistema y estudio de viabilidad. Descripción de alto nivel y definición de viabilidad del proyecto.
Descomposición del sistema. En módulos independientemente más manejables (subsistemas).
Especificaciones detalladas de los subsistemas. Se hace uso de modelos apropiados de los procesos.
Diseño e implementación de prototipo. Diseño de un sistema que tenga las mismas funcionalidades que el definitivo.
Diseño e implementación Definitiva.

En [48] se establece una metodología que cubre las etapas de desarrollo, en este caso, específicamente para un sistema robot:

Describir la anatomía del robot. Estado inicial y el objetivo principal.
Analizar el objetivo principal y descomponerlo en estructuras de comportamientos simples.
Especificar completamente los componentes del robot. Se debe especificar: La comunicación entre sensores y actuadores del robot con el entorno, La arquitectura del controlador, Estrategia de entrenamiento cuando se requiera, diseñar, implementar y verificar. Antes del entrenamiento, Ejecutar tarea de aprendizaje y Evaluar comportamiento final.

Por su parte en [2] se propone una metodología, para el diseño e implementación de una arquitectura software de control de robots móviles, que considera las diferentes etapas del problema:

Definición del problema a solucionar y el objetivo principal del sistema.
Especificación de las condiciones de funcionamiento y los requisitos del sistema.
Análisis del problema y determinación de cómo debería ser resuelto por el sistema, descomponiéndolo en tareas más simples y especificando los mecanismos (hardware y software) necesarios para su desarrollo.
Definición de los componentes software del sistema.
Implementación de los componentes anteriores.
Comprobación del funcionamiento del sistema desarrollado.
Ajuste y entrenamiento del sistema para satisfacer las especificaciones definidas en la segunda etapa.

En [49] se presenta un sistema de diseño automático que produce robots complejos explorando los principios de regularidad, modularidad, jerarquía y reuso, usando 4 niveles de computación (proceso evolutivo, proceso de ensamblaje, constructor y simulador).

En [50] se utiliza las Redes De Petri (RDP) como formalismo para: especificar, validar y generar código, a la vez que considera restricciones de tiempo real duro.

En [51] se presenta la especificación formal del sistema de mecanismo robótico, modelado usando el Unifield Modeling Language (UML). De esta manera es posible soportar la representación del modelo matemático en forma que permita cálculo eficiente de cantidades del robot, así como una optimización del modelo con respecto al número de operaciones calculadas.

En [*52*] se propone la utilizar HOOD para diseño y especificación de una arquitectura particular, AFREB, para un sistema robot móvil. Presenta de forma sistemática los procedimientos para el análisis y concepción de la arquitectura tratada.

8. CONCLUSIONES

Luego de una revisión respecto de las concepciones metodológicas específicamente orientadas a diseño de sistemas robots moviles, se observa que:

- Hay metodologías, pero que no se aplican métodos formales en el desarrollo del proyecto.

- Se aplica métodos y formalismos, pero no bajo una concepción metodológica.

De lo anterior, se podría concluir que es necesario abordar el problema de los sistemas robots, desde una concepción metodológica, incluyendo todas las etapas de un proyecto, y aplicando los métodos mas apropiados para el problema particular que se espera abordar. En concreto, como se plantea en la Figura 10, se requiere:

- Aplicar un tratamiento sistemático y metodológico en todo el desarrollo del proyecto.

- Evaluar las estrategias y métodos más apropiadas que faciliten el diseño y comprensión en la realización del proyecto.

- Guiar hacia el diseño aplicando, a todo o parte del sistema a diseñar, procedimientos fáciles, sistemáticos y estándares.

- Utilizar técnicas de representación que ofrezcan la posibilidad de interpretar y seguir fluidamente todo el proceso del proyecto. Para ello, es necesario contar con representaciones tanto visuales como textuales.

- La metodología y lenguajes que se implementen y apliquen, deben ser conocidos y aceptados por la comunidad científica, con el fin de garantizar la legibilidad y seguimiento de todas las etapas del proyecto.

Teniendo en cuenta, los conceptos anteriores e integrando los elementos más importantes, en la Figura 11 se presenta un esquema metodológico que se espera seguir en el desarrollo del proyecto, en el se consideran los aspectos más relevantes tratados en la bibliografía consultada.

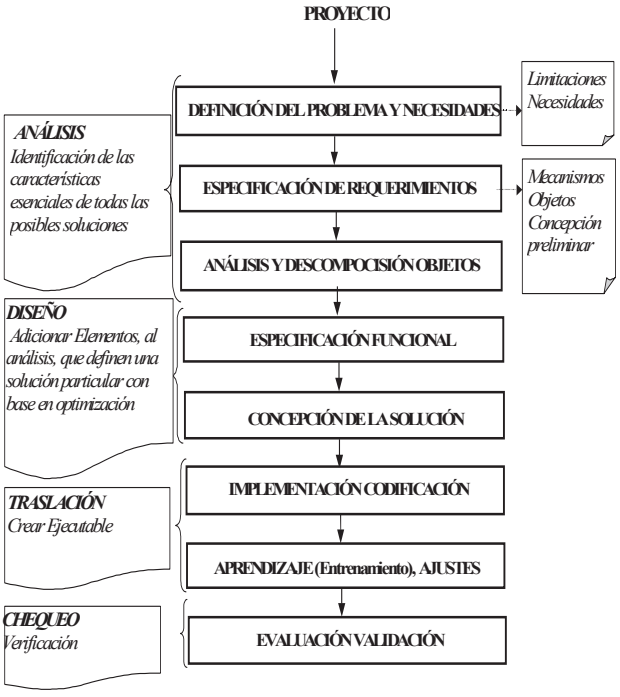

Figura 11. Diagrama de Metodología Propuesta.

9. REFERENCIAS

[1] Arai, T.; Pagello, E.; Parker, L.E. "Guest editorial advances in multirobot systems". IEEE Transactions on Robotics and Automation, Volume:18, Issue: 5, Year: Oct 2002.

[2] Muñoz L., José L. "Control en Robótica Móvil, Arquitectura y Metodología". Tesis Doctoral. Universidad de Murcia. Departamento de Automática, Electricidad y Electrónica Industrial.1998.

[3] Simó, Jose E. "Una Arquitectura Basada en Motivaciones para el Control de Robots Móviles".Tesis Doctoral, Universidad Politécnica de Valencia. 1997.

[4] Arkin, Ronald C. "The impact of Cybernetics on the Design of a Mobile Robot System: A Case Study". IEEE Transactions on System, Man and Cybernetics. Vol. 20. Nº 6. 1990.

[5] Brooks, Rodney A. "New Approaches to Robotics". Science, Vol. 253, sep. 1991, pp. 1227 1232 "MIT Artificial Intelligence Laboratory".

[6] Steel, Luc. "The Biology and Thechnology of Intelligent Autonomous Agents". Proceedings of the NATO Advanced Study Institute on The Biology and Technology of Intelligent Autonomous Agents, Held in Castel Ivano, Trento, Italy, March 1-12, 1993. Ed. Springer, 1995.

[7] Weitzendeld, Alfredo; Arkin, Ronald; Cervantes, F.; Olivares, R; Corebachof, F. "A Neural Schema Architecture for Autonomous Robots", 1999.

[8] Lewis, F.L.; et. al. "Robotics". Mechanical Engineering Handbook. Ed. Frank Kreith. CRC Press LLC, 1999.

[9] Lasky, T.A.; Hsia, T.C.; Tummala, R.L.; Odrey, N.G. "Robotics". The Electrical Engineering Handbook. Ed. Richard C. Dorf. Boca Raton: CRC Press LLC, 2000.

[10] Fu, K. S.; Gonzalez, R. C. y Lee, C. S. G. "Robótica: control, detección, visión e inteligencia". Ed. McGraw Hill. 1990.

[11] Jones., Joseph; Seiger, Bruce and Flynn, Anita. "Mobile Robots, Inspiration to Implementation". Ed. A.K.Peters. 1998.

[12] Lunt, Karl. "Build Your Oun Robot". Ed. A. K. Peters. 2000.

[13] McFaland, David. "The Biology of Behavior-Criteria for success in Animal and Robots". Proceeding of the NATO Advanced Study Institute on Biology Technoligy of Intelligent and Autonomous Agents. Trento. March 1993.

[14] Sandin, Paul E. "Robot mechanism and mechanical devices". McGraw-Hill. 2003.

[15] "What is a Robot Architecture?". http://www.frc.ri.cmu.edu/robotics.faq.

[16] "Robot Architecture" http://www.techeak_uni bielefeld.de/ag/ai.

[17] Braitenverg, Valentino. "Vehicles: Experiments in Synthetic Psycology". Cambridge. MITPress. 1984.

[18] Brooks, Rodney A. "From Earwings to Humans". Robotic and Autonomous Systems, Vol.20, Jun1997, pp. 291-304

[19] Arkin, Ronald C. *"Behavior-Based Robotics".* 2ª ed. The MIT Press, Cambridge, Massachusetts, London, England, 1999.

[20] Mautner, Craig; Belew, Richard K. "Evolving Robot Morphology and Control". Computer Science and Engineering, University of California, San Diego La Jolla, CA. 92093-0114.

[21] Câmara,Bruno; Lopez, José; Marquez, Carlos; Lima, Pedro; Cardeira, Carlos. *"IQ99*- A mobile Autonomous Vehicle" Instituto Superior técnico, Lisboa, Portugal, 1999.

[22] Booch, Grady. "Object solutions: managing the object-oriented project". Addison Wesley Longman Publishing. 1995.

[23] Larman, Craig. "Agile and iterative development: a manager's guide" A.Wesley, 2004.

[24] Bátíz P., Juan de Dios. "Desarrollo Orientado a Objetos con UML" http://www.willydev.net/descargas/Articulos/General/umlTotal.pdf.

[25] http://www.rational.com

[26] Travassos, Guilherme; Shull, Forrest; Caver, Jefferey. "Working with UML: A Software Design Process Based on Inspections for the Unified Modeling Language." Advances in Computers BookSeries, vol. 54. Academic Press, 2001.

[27] Kroll, Per; Kruchten, Philoppe. "The Rational Unified Process Made Easy":A Practitioner's Guide to the RUP". Addison Wesley. 2003.

[28] Jiménez L., Rafael. "Análisis y Diseño Orientado a Objetos de un Framework para el Modelado Estadístico con MLG". Tesis doctoral, Facultat de Psicología, Universitat de les Illes Balears. Palma de Mallorca, 2003

[29] Douglass, Bruce Poewl. "DOING HARD TIME. Developing Real-Time Systems with UML, Objects, Frameworks, and Patterns". Addison-Wesley, 1999.

[30] Booch, Grady. "Object Oriented Analysis and Design". 2ª ed. Benjamin Cummings 1994.

[31] Cruzado Nuño, Ignacio; García D., Jorge; Portugal A. Javier. "Metodologías Orientadas a Objeto. http://pisuerga.inf.ubu.es/icruzado/tfc/Metodologias.pdf.
[32] Rumbaugh, J. Et al. "Object Oriented Modeling and Design". Prentice Hall. 1991.
[33] Julián, Vicente J. ; Botti, Vicente J. "Estudio de métodos de desarrollo de sistemas multiagente". Departamento de Sistemas Informáticos y Computación Universidad Politécnica de Valencia. 2002.
[34] Julian, V.; y Botti, V, "Estudio de métodos de desarrollo de sistemas multiagente". Revista Iberoamericana de Inteligencia Artificial No. 18. 2003
[35] HOOD, User Group. "HOOD User ". Issue 1.0, 1996.
[36] Agency European Space, HOOD Working Group, "HOOD Reference Manual". Issue 4.0, 1995.
[37] Rosen, Jean Pierre. "HOOD: an Industrial Approach for Software Design". HOOD Technical Group. 1997.
[38] Craig, L. "UML y Patrones: Introducción al Análisis y Diseño Orientado a Objetos". Ed. Prentice Hall, may, 1999.
[39] Booch, G; Jacobson,, I y J. Runbaugh. "El Lenguaje Unificado de Modelado". México: Addison Wesley, 1999.
[40] Rumbaugh, James; Jacobson, Ivar; Booch, Grady. "El lenguaje Unificado de Modelado, Manual de Referencia". Racional Software Corporation. 2000.
[41] Letelier T., Patricio; Sánchez P., Pedro. "Análisis y Diseño Orientado a Objetos usando la notación UML". Departamento Sistemas Informáticos y Computación Universidad Politécnica de Valencia (España). 2000. www.dsic.upv.es/~uml LLC, 1999.
[42] Jacobson, Ivar; Booch, Grady; Rumbaugh, James. "El Proceso Unificado de Desarrollo de Software". Adison Westley. 2000.
[43] Booch, Grady; Rumbaugh, James; Jacobson, Ivar "The Unified Modeling Language User Guide" 2a Ed. Addison Wesley Professional. 2005.
[44] Saroka, Raúl Horacio. "Sistemas de Información en la Era Digital". Programa Avanzado de Perfeccionamiento en Management de la Fundación OSDE y Universidad Nacional de San Martín. 2002.
[45] Castro G., Robin Alberto "Estructura básica del proceso unificado de desarrollo de software". Universidad Icesi. 2004. www.icesi.edu.co/esn/contenido/pdfs/rcastro_estructura-bas-puds.pdf
[46] Kruchten, P. "The 4+1 View Model of Architecture". IEEE Software, 12(6). November 1995
[47] Seward, D. W.; Garman, A. "The software development process for an intelligent robot". Computing & Control Engineering Journal. Abril 96
[48] Colombetti, Marco; Dorigo, Marco and G. Borghi. "Behavior Analysis and Training- A methodology for Behavior Engineering". IEEE Transactions on System, Man and Cybernetics. Vol. 26. Nº 3. 1996.
[49] Hornby, G.S.; Lipson, H.; Pollack, J.B. "Generative representations for the automated design of modular physical robots" IEEE Transactions on Robotics and Automation. Volume: 19, Issue: 4. Aug. 2003.
[50] Montano, Luis; Garcia, F. José; Villa, José. "Usingthe time Petri Net Formalism for specification, Validationand Code Generationin Robot Control Applications". Universidad Zalamanca Universidad la Rioja, Jul, 1999.
[51] Surla, Dusan; Konjovic, Zora; Rackovic, Milo. "UML Specification of the System for Gernerating Simbolic Models of Robotic Mechanism Dynamics". 13th ISPE/IEE International Conference on CAD/CAM, Robotics / Fabricatories on the Future – Pereira 1997.
[52] Londoño O, Nelson; Tornero, Josep,"Análisis de la Metodología HOOD para la Implementación de Arquitecturas Aplicadas a Robots". VIII Congreso Latinoamericano de Control Automático, Viña del Mar, Chile, Nov. 1988.
[53] Kurfess, Thomas R. "Robotics and Automation Handbook". CRC Press LLC. 2005. http://www.crcpress.com
[54] Meyer, Bertrand. "Construcción de Software Orientado a Objetos". 2ª ed. Prentice Hall. 1999.
[55] J. Rumbaugh. "OMT: The development model". JOOP Journal of Object Oriented Programming, pages 8–16, 76, May 1995.