UNIVERSIDAD NACIONAL DE SANTIAGO DEL ESTERO
FACULTAD DE CIENCIAS EXACTAS Y TECNOLOGÍAS

LICENCIATURA EN SISTEMAS DE INFORMACION

TRABAJO FINAL DE GRADUACION

PROPUESTA DE UN MARCO PARA LA EVALUACIÓN ESTRUCTURAL DE METODOLOGÍAS ORIENTADAS A AGENTES

Autora:
MARIELA SUSANA SANDEZ

Profesora Guía:
ING. DIANA PALLIOTTO

Diciembre de 2015

UNIVERSIDAD NACIONAL DE SANTIAGO DEL ESTERO
FACULTAD DE CIENCIAS EXACTAS Y TECNOLOGÍAS

TRABAJO FINAL DE GRADUACIÓN DE LA LICENCIATURA EN SISTEMAS DE INFORMACIÓN

PROPUESTA DE UN MARCO PARA LA EVALUACIÓN ESTRUCTURAL DE METODOLOGÍAS ORIENTADAS A AGENTES

Autora:

……
MARIELA SUSANA SANDEZ

Profesora Guía:

……..
ING. DIANA PALLIOTTO

* ——————————————— * ———————————————*

Aprobado el día ..... del mes de ..... del año 20.....
por el Tribunal integrado por

....

....

Santiago del Estero – Argentina

A mi hijo Rodrigo, luz y bendición que Dios me dio.

A mis padres, por su ejemplo y acompañamiento incondicional.

Agradecimientos

A mis padres, María Elena y Roberto, por cuidarme y acompañarme siempre.

A mi madrina Lily, mi abuela Martina, mi prima María Rosa y mis tíos, Gaby y Tati por estar siempre, ante cada necesidad que surge.

A la Ing. Diana Palliotto por aceptar guiarme en la realización de este trabajo y brindarme su tiempo y conocimientos para la concreción del mismo.

A mis compañeros de carrera, con los cuales, a lo largo de estos años, me unió estudio, diversión, buenos y malos momentos. Entre ellos me permito nombrar a Fabio Gallo, Romina Soria, Cristian Di Natale, Noelia Yolde, Matías Loto, Mario Crespo, Pio Montenegro, Oscar Gallardo, Karina Noriega, Maxi Campos, Mariza Rodríguez y Cynthia Gallardo.

A las autoridades de la FCEyT, Decano: Ing. Héctor Paz, Vicedecano: Ing. Pedro Basualdo, Secretarios: Ing. Rosa Kairuz, Dra. Fernanda Mellano, Ing. Teresita Pilan, Ing. Ricardo Cordero, entre otros, por interesarse y alentarme a finalizar este trabajo final de graduación.

A mis compañeros de trabajo de la FCEyT, por su aliento para concretar este trabajo.

A mis compañeros del MIEC (Movimiento Integrador Estudiantil por el Cambio) por permitirme soñar y planificar juntos un cambio superador e integrador.

M.S.S.
Santiago del Estero, Argentina
Diciembre de 2015

CONTENIDO

RESUMEN

En los últimos años se desarrollaron una gran cantidad de aplicaciones de agentes exitosas que, entre otras cosas, sirvieron para validar la viabilidad y la utilidad del modelo de agentes. Debido a esto, existe un gran interés en su utilización en el contexto industrial, es decir en aplicaciones de gran tamaño.

Se han propuesto numerosas metodologías para el desarrollo de sistemas basados en agentes; sin embargo, no existe un estándar para definir una "metodología" orientada a agentes debido a un abuso del término; en consecuencia, su aplicación es todavía limitada por su falta de madurez. Surge así la necesidad de contar con metodologías adecuadas para abordar la solución de problemas complejos, coordinar el trabajo en equipo, conducir el proceso de desarrollo y extender la utilización del paradigma de agentes.

No hay muchos trabajos que se ocupen de la evaluación de metodologías orientadas a agentes. Por ello, este trabajo propone un marco (conjunto de criterios) para la evaluación estructural de metodologías orientadas a agentes. Dicho marco está basado en cuatro aspectos principales de una metodología: conceptos y propiedades; notaciones y técnicas de modelado; procesos; pragmática. La utilización de este marco implica un examen detallado de los modelos y los procesos, y de las fortalezas y debilidades de una metodología. Todo este análisis juega un rol importante en la evolución y en el desarrollo de una "nueva generación" de metodologías orientadas a agentes estandarizadas.

PALABRAS CLAVES: Agentes; Ingeniería de Software Orientada a Agentes; Metodologías Orientadas a Agentes; Evaluación de Metodologías Orientadas a Agentes.

CAPÍTULO I
PRESENTACIÓN

I.1. INTRODUCCIÓN

Debido a la gran evolución que tiene la tecnología de agentes, se hace necesario contar con metodologías adecuadas para abordar la solución de problemas complejos, coordinar el trabajo en equipo, conducir el proceso de desarrollo y extender la utilización del paradigma de agentes.

En la actualidad, existen más de dos docenas de Metodologías Orientadas a Agentes (MOA); sin embargo, ninguna de ellas puede considerarse madura puesto que todavía necesitan que se las use reiteradamente para que evolucionen. Además, ante la falta de estandarización, la evaluación de las metodologías presenta varias dificultades:

- La diferencia en la terminología de los conceptos que utilizan.
- La influencia que reciben de otros paradigmas.
- La diversificación en los elementos que la integran; por ejemplo, algunas proporcionan sólo notaciones gráficas, mientras que otras integran varios aspectos (notaciones, directrices, etc.).

Un paso fundamental es entender las fortalezas, las debilidades y los ámbitos de aplicabilidad de cada metodología. Este trabajo pretende ofrecer un *marco* bien definido y estructurado de los aspectos que una MOA debería incluir. Se basa en el marco propuesto por Sturm & Shehory [1], el cual trata Criterios de Evaluación de la Ingeniería de Software y Características de los Sistemas basados en Agentes.

I.2. PLANTEAMIENTO Y FORMULACIÓN DEL PROBLEMA

Existe una gran cantidad de aplicaciones de agentes que se desarrollaron exitosamente, mostrando la viabilidad y la utilidad del modelo, y su aplicabilidad al contexto industrial. Así, entra en juego la *Ingeniería de Software Orientada a Agentes* (ISOA), definida como "la aplicación de un método sistemático, disciplinado y cuantificable al desarrollo, implantación y mantenimiento de software con agentes" [2]. La ISOA utiliza el enfoque de agentes para la implementación de sistemas software complejos (distribuidos), mediante la descomposición del problema en un conjunto de entidades independientes (agentes) que interactúan entre sí utilizando modelos de comunicación de alto nivel y mecanismos organizativos y sociales para interactuar.

Según Graham [3], una metodología debe proporcionar: un proceso de ciclo de vida completo, un amplio conjunto de conceptos y modelos, un conjunto completo de técnicas (normas, directrices, heurística), un determinado conjunto de entregables de cada fase, un lenguaje de modelado, un conjunto de métricas de calidad, normas de codificación (y otras), asesoramiento, reutilización y directrices para la gestión de proyectos. Estos elementos están asociados con cada una de las cuatro principales divisiones siguientes:

- **Conceptos y propiedades.** Se refiere a si una metodología utiliza las nociones básicas de un agente o de un sistema multiagente. Son palabras claves como autonomía, reactividad, proactividad, metas, tareas, mensajes, protocolos, sociedades, etc.
- **Notaciones y técnicas de modelado.** Las notaciones se refieren a símbolos técnicos de un paradigma, y las técnicas de modelado a un conjunto de elementos de descripción con varias capas de abstracción que permiten definir bien una parte, en este caso, los agentes, los roles, las organizaciones, etc.
- **Proceso de desarrollo.** Se ocupa de definir adecuadamente una serie de acciones que, precisamente cuando se realizan, obtienen como resultado un sistema con determinada funcionalidad.
- **Pragmática.** Trata los aspectos prácticos de las metodologías, es decir, aspectos que hacen referencia a la manera en que una metodología se introduce, y a su proceso de evolución.

En este sentido, el presente trabajo propone un marco para realizar una evaluación estructural de las metodologías. Esto implicará examinar sus procesos y modelos en detalle, a través de un conjunto de propiedades a las que deberían adherirse las *notaciones y técnicas de modelado* de una metodología, por ejemplo: *accesibilidad, analizabilidad, expresividad, aplicabilidad, modularidad, ejecución y pruebas*, etc. Además, el análisis debe incluir un conjunto de *aspectos del proceso de desarrollo*, como son el *contexto de desarrollo*, las *fases del ciclo de vida y sus actividades*, los *mecanismos de validación y de verificación*, entre otros. Todo con el fin de contribuir un paso más hacia el desarrollo de la "próxima generación" de MOA.

I.3. ESTADO DEL ARTE

No existen muchos trabajos de evaluación de MOA. Shehory & Sturm [4] realizaron una evaluación basada en las características de las metodologías de la ISOA, en

la que incluyen criterios relacionados con la ingeniería de software y criterios relativos a los conceptos de agentes.

Gómez Alvarez, en su tesis doctoral [5], realiza modificaciones al marco de Sturm & Shehory [1], al añadir propiedades y desagregar conceptos, buscando obtener un marco aplicable y completo al problema de la evaluación de metodologías.

En su tesis de magister, Zohreh Akbari [6], presenta un marco de evaluación con diferentes niveles de criterios basado en Sturm & Shehory [1]. Muestra los aspectos cubiertos por [1] sumado a criterios para evaluar el soporte a la ingeniería del software y los criterios de comercialización del mismo.

O'Malley y DeLoach [7] proponen una serie de criterios para evaluar las metodologías con vistas a permitir que las organizaciones decidan si adoptan MOA o utilizan las Metodologías Orientadas a Objetos (MOO). A pesar de realizar un estudio para validar sus criterios, no proporcionan directrices detalladas o un método de evaluación de dichos criterios. Su ejemplo de comparación entre MaSE y Booch da calificaciones a las mismas, pero sin justificarlas.

I.4. JUSTIFICACIÓN

El área de agentes autónomos y de sistemas multiagentes (SMA) se ha convertido en una tecnología prometedora, ofreciendo alternativas sensatas para el diseño de sistemas distribuidos inteligentes. Se han realizado varios esfuerzos (académicos, industriales y de consorcios de estandarización) para proporcionar nuevos métodos y herramientas, con la intención de establecer las normas necesarias para lograr un uso extenso de las técnicas de los SMA. Poseer un agente de calidad es de importancia para una organización ya que se logra personalización de los servicios, flexibilidad de la distribución y delegación de las tareas; y, además, al asegurar una infraestructura integrada, robusta, escalable y compatible se disminuye el costo de mantenimiento y permite compartir información de manera confiable. Para esto, es esencial que tal tecnología pueda incorporarse en las prácticas existentes en la industria del software, y no que se vea simplemente como un nuevo paradigma prometedor.

Con este trabajo se intentará demostrar que el marco de evaluación resultante puede ser utilizado: por los investigadores para examinar similitudes y diferencias entre metodologías existentes; por las organizaciones para seleccionar una metodología para el desarrollo de aplicaciones basadas en agentes. También puede dar lugar a una selección de

las mejores metodologías, reduciendo gradualmente su número, lo que significaría un paso más hacia la estandarización.

I.5. OBJETIVOS DE LA INVESTIGACIÓN

I.5.1. OBJETIVO GENERAL

Contribuir al mejoramiento de la calidad del proceso de desarrollo de los sistemas basados en agentes.

I.5.2. OBJETIVOS ESPECÍFICOS

- Reconocer las características principales de las distintas MOA, a partir de su revisión y estudio.
- Plantear un marco para evaluar MOA.
- Aplicar el marco obtenido a casos de estudio (es decir, evaluar una o más metodologías).
- Obtener una evaluación de los aspectos cubiertos por las metodologías analizadas.

I.6. ESTRUCTURA DEL TRABAJO

El presente trabajo de investigación se encuentra dividido en siete capítulos. En el primer capítulo se presenta el trabajo de investigación a realizar. En el segundo, tercero y cuarto capítulo se presentan los marcos referenciales en los que se fundamenta el trabajo. En el quinto capítulo se revisa y modifica el marco de evaluación propuesto por Sturm & Shehory [1]. El sexto capítulo describe la operacionalización del marco resultante, y el séptimo capítulo presenta la evaluación realizada sobre el mismo. Finalmente, se presentan las conclusiones del trabajo en donde se incluyen las líneas de investigación futuras derivadas de éste.

CAPÍTULO II
TECNOLOGÍA DE AGENTES: ORÍGENES, CONCEPTOS Y EVOLUCIÓN

II.1. INTRODUCCIÓN

En la historia de la Informática han sucedido constantes cambios de paradigmas: desde la electrónica al ensamblador, de aquí a la programación simbólica y la descomposición funcional, y más recientemente a la orientación a objetos y servicios. En esta evolución, el esfuerzo de desarrollo no puede aumentar de forma proporcional a la complejidad del software producido. Es por ello que la Ingeniería del Software (IS) se enfrenta al problema de desarrollar software cada vez más complejo con unos recursos productivos limitados, mediante la utilización de paradigmas cada vez más abstractos.

Ante sistemas complejos, entornos abiertos y una necesidad creciente de abstracción para hacer software a un costo razonable, la Ingeniería de Software Orientada a Agentes (y las plataformas o los lenguajes de desarrollo de agentes relacionados) es un enfoque que ha demostrado ser muy eficiente para construir sistemas complejos (en promedio, el tiempo de desarrollo es 350% más rápido) en comparación con el desarrollo estándar, mediante un modelo donde el control y la información están distribuidos en entidades autónomas, proactivas y deliberativas, capaces de colaborar y de organizarse [8].

Los agentes son, en definitiva, una solución adecuada para aplicaciones descentralizadas, mal estructuradas, cambiantes y complejas [9].

II.2. FUNDAMENTOS E INICIOS DE LA TECNOLOGÍA DE AGENTES

II.2.1. INTELIGENCIA ARTIFICIAL

La Asociación Americana de Inteligencia Artificial, define la *Inteligencia Artificial* (IA) como *"*el conocimiento científico de los mecanismos que subyacen al pensamiento y al comportamiento inteligente y su incorporación a las máquinas" [10].

El problema de este tipo de definiciones viene dado por la interpretación misma del concepto *inteligencia* y así, a lo largo de la historia, la IA ha estudiado disciplinas o comportamientos "inteligentes" que, con el avance de la Informática, una vez madurados e incorporados a los sistemas dentro de la normalidad, han dejado de ser considerados propios de este ámbito.

Una obra clásica de referencia a este tema es el artículo "*Computing Machinery and Intelligence"* de Alan Turing [11], donde se propone el *test de Turing* como método para determinar si una máquina es inteligente o no, basándose en su capacidad para emular una conversación con una persona real.

II.2.2. INTELIGENCIA ARTIFICIAL DISTRIBUIDA

La Inteligencia Artificial Distribuida (IAD) se define como "el campo de la Inteligencia Artificial que se centra en los comportamientos inteligentes colectivos que son producto de la cooperación de diversas entidades denominadas agentes" [12].

A pesar de ser una disciplina joven, se pueden distinguir tres periodos cronológicos bien diferenciados [13]:

- **IAD "clásica"**, estudia la conducta colectiva, a diferencia de la IA que estudia la conducta individual.

  Se pueden distinguir dos áreas de investigación: la *Resolución (Cooperativa) de Problemas Distribuidos* (DPS - Distributed Problem Solving) y los *Sistemas Multiagentes* (SMA). La diferencia entre ambas es la flexibilidad, en DPS las tareas e interacciones están prefijadas según un plan centralizado, mientras que en los SMA, los agentes tienen un grado de autonomía mayor y pueden decidir dinámicamente qué interacciones son adecuadas, qué tareas deben realizar, quién realiza cada tarea y, además, es posible mantener conocimiento que no es globalmente consistente, incluso los agentes pueden mantener objetivos globales diferentes.

  Aborda los temas de:

  - Descripción, descomposición y asignación de tareas.
  - Comunicación.
  - Coordinación.
  - Resolución de conflictos.

- **IAD "autónoma"**, estudia los agentes individuales en un mundo social; se ocupa de la perspectiva interna del agente y engloba:
  - Teoría de agentes; propiedades, caracterización, taxonomía.
  - Arquitecturas de agentes.
  - Lenguajes de agentes.
- **IAD "comercial"**, se centra en la aplicación de la IAD clásica y autónoma, desarrollando agentes software con características muy diferenciadas (agentes móviles, personales, etc.) que están siendo explotados de forma comercial.

Para presentar las diferentes áreas de trabajo de la IAD, se puede partir de la clasificación realizada por Moulin y Chaib-draa [14] [15], que distingue cuatro áreas (Figura II.1.):

1) La *perspectiva de grupo*, que estudia los elementos que caracterizan a un grupo de agentes, su comunicación e interacciones. Los elementos característicos pueden descomponerse a su vez en tres aspectos: organización del grupo, comunicación y coordinación. Esta perspectiva está ligada a los temas abordados por la *IAD "clásica"*.
2) La *perspectiva de agente*, que incluye todos los elementos que caracterizan a un agente. Esta perspectiva ha sido desarrollada siguiendo las divisiones de la "IAD autónoma y comercial", combinando la división de los congresos ATAL (*Agent Theories, Architectures and Languages*) [16] con la tipología de agentes software de Nwana [13].
3) *Perspectivas específicas*, que recogen las relaciones de otros campos con la IAD. Como por ejemplo, los sistemas de información abiertos de Hewitt [17][18], o los agentes *heterogéneos* de Genesereth [19] [20].
4) La *perspectiva del diseñador*, que constituyen el estudio de metodologías, plataformas y técnicas disponibles para realizar aplicaciones basadas en agentes.

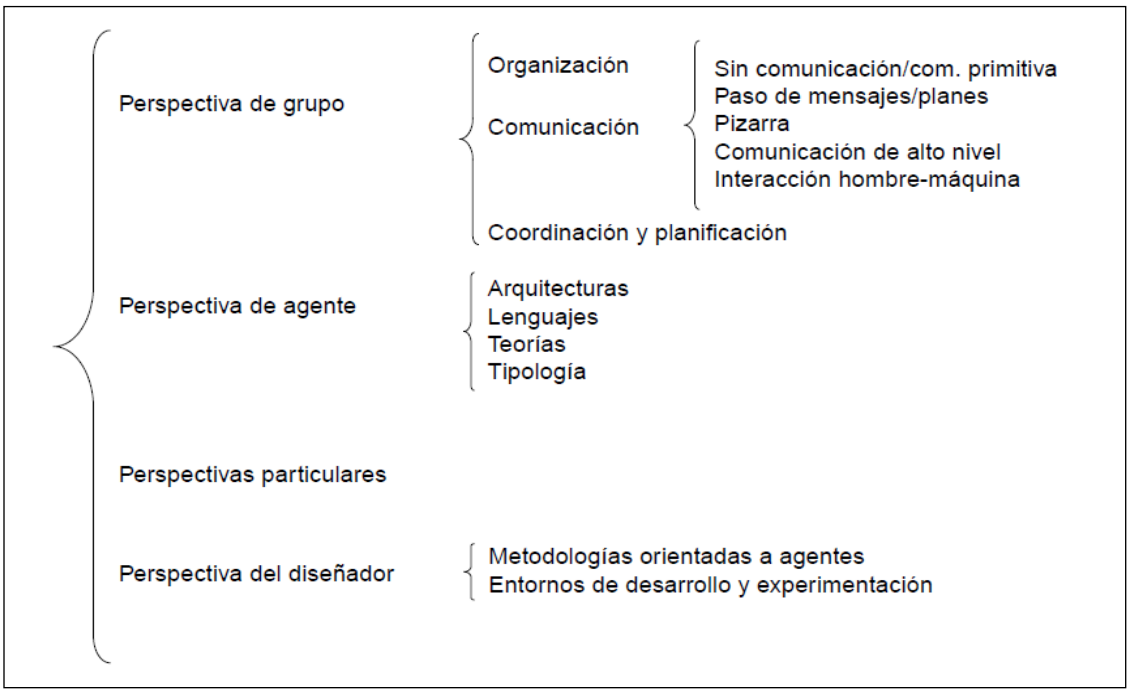

Figura II.1. Perspectivas para analizar y clasificar la IAD (extraída de [12])

II.3. AGENTE

El paradigma de Ingeniería de Software Orientada a Agentes (ISOA) se basa en la noción de ***agente*** como *elemento fundamental del sistema* [21] [22].

Newell describe que un programa convencional maneja expresiones, variables en el nivel de símbolos, mientras que un agente se sitúa en el nivel del conocimiento, por lo tanto maneja únicamente conocimiento en forma de entidades mentales como objetivos, planes o creencias y se comporta de acuerdo con el *principio de racionalidad*, es decir, realiza acciones para satisfacer unos objetivos [21].

"*Un agente software es un sistema software situado en un determinado* ***entorno*** *capaz de realizar acciones flexibles de manera* ***autónoma*** *en pro de unos* ***objetivos*** *para los que ha sido diseñado*" [23] (Figura II.2.). Esta definición, una de las más usadas, identifica una de las características de un agente, como la *autonomía* para realizar acciones que resuelvan los objetivos de diseño, pero no alcanza para distinguirlo claramente de un sistema distribuido convencional.

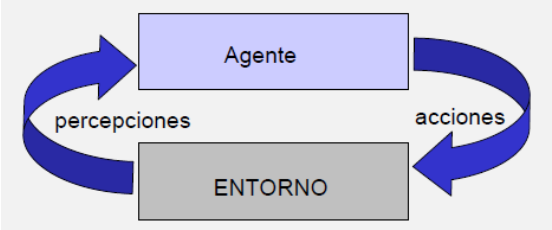

Figura II.2. Definición de agente (extraída de [24]).

Demazeau, define el concepto de agente desde el punto de vista externo e interno. Desde el punto de vista externo, "*Agente es una entidad real o virtual que* ***evoluciona*** *en un* ***entorno****, es capaz de* ***percibir*** *y sobre ello* ***actuar*** *sobre este entorno, capaz de comunicarse con otros agentes y que* ***exhibe un comportamiento autónomo***", mientras que desde el punto de vista interno es "*una entidad real o virtual con* ***control local*** *en algunos de sus procesos de* ***percepción, comunicación, adquisición de conocimiento, razonamiento, decisión, ejecución y acción***" [25].

El concepto de "*agente*" caracteriza a una entidad software con una arquitectura robusta y adaptable que puede funcionar en distintos entornos o plataformas computacionales y es capaz de realizar en forma "*inteligente*" y autónoma distintos objetivos intercambiando información con el entorno, o con otros agentes humanos o computacionales [26].

Ante el debate de qué es un agente, en Franklin & Graesser [27] se discuten cuatro nociones claves que distinguen a los agentes de los programas: *reacción con el medio ambiente, la autonomía, la meta de la orientación y la persistencia*.

Este trabajo concluye que el concepto de *agente* ofrece una conveniente y potente manera de describir una entidad computacional compleja la cual es capaz de actuar con un cierto grado de autonomía con el fin de completar tareas en nombre del entorno. A diferencia de los objetos, los cuales son definidos en términos de *métodos* y *atributos*, un agente es definido en términos de su *comportamiento*.

Los atributos básicos de un *agente software* son que dichos agentes:

- no son invocados estrictamente por una tarea, sino que se activan ellos mismos;
- pueden estar en un servidor en estado de espera, preservando el contexto;
- pueden hacer que se ejecute un estado en un servidor sobre condiciones de inicio;
- no requieren acción de los usuarios;
- pueden invocar otras tareas incluyendo comunicación.

II.3.1. CARACTERÍSTICAS DE LOS AGENTES

Algunas de las características que se suelen atribuir a los agentes en mayor o menor grado para resolver problemas particulares y que han sido descritas por autores como Franklin & Graesser [27], y Nwana [13], son:

- **Autonomía**: un agente es completamente autónomo si es capaz de actuar basándose en su experiencia. Una vez creado y provisto de la información necesaria sobre sus objetivos, tareas y limitaciones, el agente debería poder operar independientemente de su usuario, esto es, autónomamente en background [28] [29]. Para ello, es necesario que el agente sepa qué debe hacer según cada caso y mantener un estado interno.
- **Reactividad**: un agente actúa como resultado de cambios en su entorno. En este caso, un agente percibe el entorno y esos cambios dirigen el comportamiento del agente.
- **Proactividad**: Tiene que tener un carácter emprendedor y tomar la iniciativa para actuar guiado por los objetivos que debe satisfacer. En cada momento el agente decide qué acción llevar a cabo. No sólo actúa en

función de los estímulos que percibe sino que realiza acciones como resultado de sus decisiones.

- **Sociabilidad**: Para poder interactuar con su entorno, el agente típicamente deberá interactuar con el mundo exterior. Esta interacción puede darse a distintos niveles, pero en general, el agente necesitará comunicarse con el entorno, con otros agentes, y con sus usuarios.
- **Racionalidad**: el agente siempre realiza «lo correcto» a partir de los datos que percibe del entorno. Nunca actuará en contra de sus objetivos y sí a favor de ellos.
- **Adaptabilidad**: está relacionado con el aprendizaje que un agente es capaz de realizar y si puede cambiar su comportamiento basándose en ese aprendizaje.
- **Movilidad**: capacidad de un agente de trasladarse a través de una red para acometer sus objetivos.
- **Veracidad**: asunción de que un agente no comunica información falsa a propósito.
- **Benevolencia**: El agente no tendrá objetivos que lo fuercen a transmitir falsa información o realizar acciones que vayan contra sus propios objetivos.
- **Continuidad Temporal**: se considera a un agente como un proceso sin fin, ejecutándose continuamente y desarrollando su función.

II.3.1.1. CARACTERÍSTICAS PROPIAS DE LOS AGENTES

Luego de observar las diversas propiedades propuestas por la bibliografía analizada y teniendo en cuenta las definiciones de *agente* presentadas en este mismo capítulo, para los propósitos de la presente investigación:

- No se tendrán en cuanta las propiedades: Veracidad y Benevolencia, de forma individual, debido a que están presentes en la propiedad Racionalidad.
- No se tendrá en cuanta la propiedad: Continuidad Temporal, debido a que se asume que todo agente es un proceso en ejecución.

La Tabla II.1 muestra las características resultantes.

Tabla II.1. Características propias de los agentes

Característica	*Significado*	*Justificación*
Reactividad	Responde de manera oportuna a los cambios en el medio ambiente.	El agente percibe el entorno en el que está inmerso y responde de manera oportuna a cambios que tienen lugar en él.
Autonomía	Ejerce control sobre sus propias acciones.	Una vez creado y provisto de la información necesaria sobre sus objetivos, tareas y limitaciones, el agente debería poder operar independientemente de su usuario.
Proactividad	No simplemente actúa en respuesta al medio ambiente, sino que es orientado a los objetivos.	En cada momento el agente decide qué acción llevar a cabo. No sólo actúa en función de los estímulos que percibe sino que realiza acciones como resultado de sus decisiones.
Sociabilidad	Es socialmente capaz. Se comunica con otros agentes, tal vez incluyendo a las personas.	Para poder interactuar con su entorno, el agente deberá interactuar con el mundo exterior. Esta interacción puede darse a distintos niveles: con el entorno, con otros agentes, y con sus usuarios.
Adaptabilidad	Cambia su comportamiento en función de su experiencia previa.	El agente mejora su rendimiento con el tiempo.
Movilidad	Es capaz de transportarse de una máquina a otra.	Permite a los agentes moverse a través de la red para acometer sus objetivos.
Racionalidad	Posee "personalidad" creíble y estado emocional.	Es el principio por el cual un agente nunca actuará de modo que vaya contra sus objetivos y sí a favor de ellos.

II.3.2. CLASIFICACIÓN DE LOS AGENTES

Se puede clasificar a los agentes desde distintos puntos de vista. Demazeau propuso un modelo de clasificación llamado “de las Vocales” (**AEIOU**) [30].

Según **A**, pueden distinguirse dos categorías de agentes atendiendo a sus **características individuales:**

1. **Agentes Reactivos**. Su modelo computacional está basado en un ciclo de *recepción de eventos externos/reacción*. La reacción consiste en la ejecución de procedimientos o rutinas sencillas según el estado interno del agente. El comportamiento reactivo puede consistir en cambiar el estado interno, o en ejecutar funciones internas o externas sobre el entorno. Un agente de este tipo *no realiza procesos de razonamiento*, ni tiene ningún mecanismo explicito de representación del conocimiento.

   Internamente, la arquitectura de los agentes reactivos está basada en capas especializadas en cada tipo de estímulos que se pueden estructurar en paralelo, si todas las capas tienen acceso a los sensores y actuadores,

(arquitecturas horizontales) o en cadena si los resultados de una capa son la entrada de la siguiente (arquitecturas verticales).

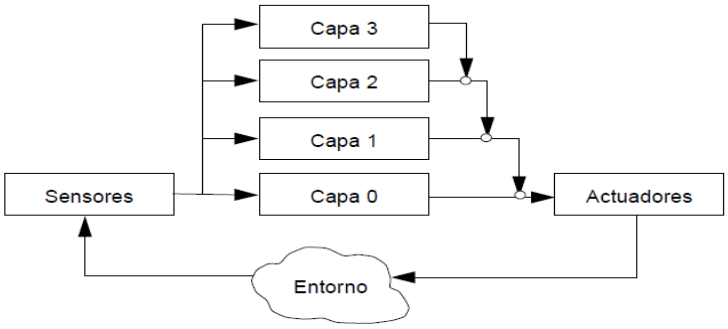

Figura II.3. Arquitectura de subsunción de un agente reactivo (extraída de [12])

2. **Agentes Cognitivos (o Deliberativos)**. Utilizan algún tipo de representación explicita (simbólica) del conocimiento. Para realizar tareas complejas necesitan llevar a cabo procesos de razonamiento y otros procesos cognitivos como la planificación y el aprendizaje. Su modelo computacional se basa en un ciclo *percepción-asimilación-razonamiento-actuación*. Su arquitectura interna tiene como núcleo central algún tipo de procesador de conocimiento que integra la información procedente del entorno y controla el comportamiento interno y las acciones del agente de acuerdo con el conocimiento de éste. (Figura II.4.)

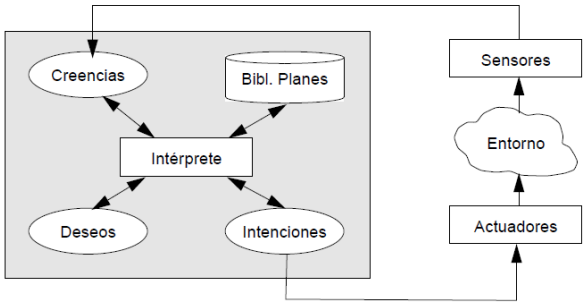

Figura II.4. Arquitectura de un agente deliberativo (extraída de [12])

También, existen **Agentes Híbridos,** los cuales pretenden combinar aspectos deliberativos y reactivos. Los más claros exponentes son *PRS*

(Procedural Reasoning System – sistema de razonamiento procesal) [31], *TouringMachines* [32,33] e *Interrap* [34,35,36]. Estas arquitecturas combinan módulos reactivos con módulos deliberativos. Los módulos reactivos se encargan de procesar los estímulos que no necesitan deliberación, mientras que los módulos deliberativos determinan qué acciones deben realizarse para satisfacer los objetivos locales y cooperativos de los agentes. Las arquitecturas híbridas pueden ser horizontales (*TouringMachines*) o verticales (*PRS* e *Interrap*). (Figura II.5.)

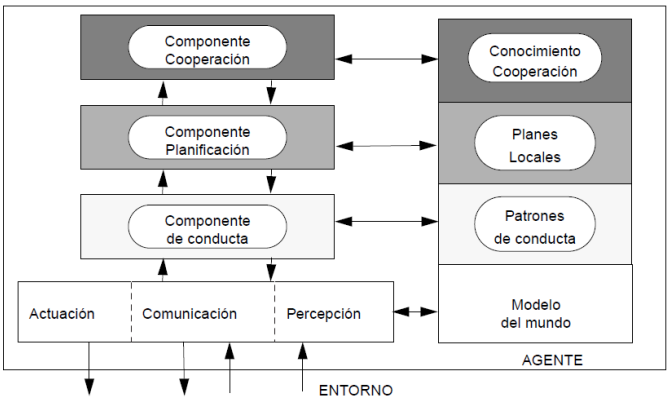

Figura II.5. Arquitectura híbrida (vertical) de *Interrap* (extraída de [12])

También podemos clasificar los agentes en función de **E**, es decir según las propiedades del **entorno** en el que operan [37]:

a. **Entornos Observables o Parcialmente Observables**. Según la información disponible sobre el entorno, un entorno es observable si los sensores detectan todos los aspectos relevantes en la toma de decisiones. Un entorno puede ser parcialmente observable debido al ruido y a la existencia de sensores poco exactos o por que los sensores no reciben información de parte del sistema.
b. **Entornos Deterministas o Estocásticos**. Si el siguiente estado del medio está totalmente determinado por el estado actual y la acción ejecutada por el agente, entonces el entorno es determinista; de otra forma es estocástico.
c. **Entornos Estáticos o Dinámicos**. Si el entorno puede cambiar cuando el agente esta deliberando, entonces el entorno es dinámico para el agente; caso contrario es estático.

d. **Entornos Discretos o Continuos**. Según sean los cambios y propiedades observadas en el entorno. Tiene en cuenta la forma en que se maneja el *tiempo* y las *percepciones* y *acciones* del agente.

Considerando al **entorno** del agente como toda la **infraestructura computacional** que lo rodea (SO, protocolos, etc.) podemos considerar dos tipos de agentes [30]:

1. Los que requieren un *entorno especial*, es decir una plataforma software específica para su funcionamiento. Por ejemplo, los agentes móviles, donde cada plataforma proporciona los mecanismos para gestionar su ciclo de vida y para que se desplacen de un nodo a otro.
2. Los que se ejecutan en *plataformas computacionales existentes*. Estos se crean mediante los recursos del sistema operativo y siguen el ciclo de vida de cualquier aplicación.

Los tipos de agentes según **I**, modo de interacción, se distinguen de **tres tipos de interacciones** [30]:

1. **Interacciones *agente – agente***. Pueden llevarse a cabo mediante lenguajes estándares de comunicación entre agentes como ACL y KQML. También pueden implicar el uso de protocolo de comunicación para sistemas distribuidos (RMI, CORBA, HTML). A veces requieren el uso de protocolos propietarios.
2. **Interacciones *agente – entorno***. Comprende la comunicación del agente con distintos elementos del entorno de los cuales tiene que obtener información o recibirla de forma espontanea: SO, protocolos, librerías, interfaces de usuario, etc.
3. **Interacciones *agente – persona***. Los agentes especializados en este tipo de interacción se han llamado *Agentes de interfaz*. Sirven como interfaz hombre-máquina (HCI), asistiendo al usuario en la utilización de un sistema informático. Algunas funciones típicas son el *aprendizaje* (por imitación o mediante técnicas de minería de datos), *proactividad* (realización de sugerencias al usuario) y *procesamiento de lenguaje natural*.

Según **O**, su **dimensión social**, se puede clasificar a los agentes en:

a. **Agentes Autónomos** (A+E). Los agentes son autónomos y perciben su entorno.
b. **Agentes Interactivos** (A+E+I). Los agentes pueden interactuar entre sí.
c. **Agentes Sociales** (A+E+I+O). Los agentes perciben el entorno, interactúan y son capaces de gestionar su relación con los demás.

La perspectiva **U** de utilidad permite clasificar los agentes de acuerdo con la **finalidad o propósito** con el que fueron construidos. Para ello, tendremos en cuenta [30]:

a. El **dominio de aplicación**. Las áreas donde se ha aplicado la tecnología de agentes ha ido creciendo progresivamente. Además del comercio electrónico y las telecomunicaciones, se utilizan en gestión de procesos de negocios, administración, defensa, procesos de fabricación, diseño colaborativo, entrenamiento, etc.
b. Tipos de **tareas que realizan**. Comprenden actividades especializadas como monitorización, diagnostico, control de sistemas, búsqueda de información, etc.; tareas de mediación, como bases de datos o interfaces de usuario; y tareas de coordinación entre agentes y de resolución de conflictos.

II.4. SISTEMAS MULTIAGENTES

II.4.1. DEFINICIÓN

De la propiedad social de los agentes se derivan entonces sistemas formados por diferentes agentes que colaboran entre sí. Una definición de **Sistema Multiagente** (**SMA**) es la siguiente: "Un *sistema multiagente* está formado por una red de agentes software que interactúan para resolver problemas que están más allá de las capacidades individuales o del conocimiento de cada resolvedor de problemas" [38]. Es decir, que un SMA es un sistema compuesto por agentes software donde el control del sistema está distribuido.

También, los SMA pueden definirse como un conjunto de entidades inteligentes llamadas agentes que coordinan sus habilidades para la resolución de problemas individuales o globales [39].

Una definición simple de SMA sería que es sistema informático formado por un grupo de agentes que interactúan entre sí, y que para interactuar satisfactoriamente, los

agentes necesitan la *habilidad de cooperar, coordinar y negociar* (las cuales se comentan en secciones siguientes).

En la caracterización de Ferber [40], un SMA está compuesto por los siguientes elementos:

1) Entorno
2) Objetos pasivos, gestionados por los agentes.
3) Objetos especiales, activos (Agentes)
4) Relaciones entre los objetos (y por ende, entre los agentes).
5) Conjunto de operaciones que permiten a los agentes percibir y manipular los objetos.
6) Operadores que representan la aplicación de operaciones sobre el mundo y la reacción de éste al ser alterado.

II.4.2. CARACTERISTICAS

Se puede caracterizar a los SMA [41] según tres principios:

1) *Principio Declarativo*: Un sistema multiagente se compone de los **A**gentes, **E**ntorno, **I**nteracciones y **O**rganización.
2) *Principio Funcional:* La funcionalidad de un SMA es la suma de las funcionalidades de los agentes que lo componen *más* la función de Emergencia. Este principio se aplica fundamentalmente a las arquitecturas reactivas, tipo "colonia de hormigas" donde la respuesta o solución que ofrece el sistema viene dada por el comportamiento global observado y no atendiendo a ningún agente en concreto.
3) *Principio de Recursividad*: Una entidad (que forma parte de un SMA) puede ser básica o a su vez puede ser un SMA. Este principio permite la organización jerárquica de sistemas multiagentes.

II. 4.3. RETOS

En un SMA, los agentes comparten un modelo de comunicación, a la vez que existen mecanismos de coordinación y cooperación. Sykara [38] identifica seis retos de los sistemas multiagentes:

1) Cómo descomponer problemas y asignar tareas a los agentes individuales
2) Cómo coordinar el control de agentes y las comunicaciones
3) Cómo hacer que múltiples agentes actúan de manera coordinada

4) Cómo conseguir que los agentes razonen sobre otros agentes y el estado de coordinación
5) Cómo reconciliar los conflictos de objetivos entre agentes que se coordinan entre sí
6) Cómo hacer ingeniería de sistemas multiagentes prácticos.

II. 4.4. DESARROLLO DE LOS SMA

Se pueden distinguir dos aproximaciones a la hora de desarrollar SMA. Por un lado, mediante la utilización de *lenguajes de programación de agentes*, que permiten generar el SMA utilizando un lenguaje de definición de agentes, partiendo en general de un modelo operacional o formal del SMA (enfoque Micro). La segunda aproximación parte de una *plataforma* cuyos agentes están preconcebidos y se centra en la configuración externa de los tipos de agente (se los dota de capacidades, base de conocimiento: objetivos, planes, etc. y funcionalidad o servicios) obviando el funcionamiento interno y la definición de las relaciones entre los mismos (enfoque Macro).

Con respecto a los lenguajes de agentes, entre los más populares cabe mencionar a **Agent0** [42] donde un agente es una entidad cuyo estado se ve como un conjunto de componentes mentales tales como *creencias, habilidades, elecciones y compromisos.*
Por ejemplo, ConGOLOG [43] modela la ejecución de tareas asignadas a varios agentes y su efecto en el entorno. Define estados mentales (*fluents*) modificables por las tareas, axiomas de ejecución de tareas, de precondición de tareas y da el marco para especificar a qué *fluents* afecta la ejecución de tareas. La ejecución del SMA consiste en descubrir qué acciones se pueden ejecutar en el SMA que no creen inconsistencias en el conjunto de axiomas acumulados. Si se descubren estas acciones, se puede hablar de un estado siguiente al actual.

Actualmente, entre los lenguajes de programación el más usado es Java, con sus versiones: Javalog (programación lógica) y Jess (sistemas basados en reglas de producción), mientras que las plataformas de desarrollo son:

- JADE (Java Agent DEvelopment Framework)
- FIPA-OS (FIPA Open Source)
- ABLE (Agent Building and Learning Environment)
- Jackal
- OAA (Open Agent Architecture)

II. 5. COMUNICACIÓN

Los agentes pueden realizar tareas de forma coordinada, lo que les permite poder conseguir objetivos que autónomamente no podrían conseguir, o no del mismo modo. Para la ***coordinación*** es necesaria la ***comunicación***. Se distinguen varias formas posibles [12]:

- **Sin comunicación**. Los agentes pueden interactuar sin comunicarse, infiriendo las intenciones de otros agentes. Esto puede deberse a la imposibilidad técnica o al deseo de una mayor autonomía por parte de los agentes.
- **Comunicación primitiva**. Se restringe la comunicación a un conjunto de señales con interpretación fija.
- **Arquitecturas de pizarra**. Los agentes se comunican mediante un esquema que permite publicar mensajes al resto de agentes. Consta de:

- *La pizarra*. Es una base de datos global donde acceden los agentes para publicar los mensajes. La información se suele estructurar en niveles de hipótesis jerárquicos.
- *Fuentes de conocimiento*. Módulos especializados que intercambian mensajes.
- *Mecanismos de control*. Para gestionar la concurrencia.

- **Paso de mensajes**. Los agentes dirigen mensajes a otros agentes, identificados por una dirección, nombre o *alias*. Este mecanismo se ha importado de la programación orientada a objetos concurrente.
- **Comunicación de alto nivel**. Consiste en posibilitar la comunicación en el nivel simbólico que requieren los agentes BDI para comunicar sus deseos, intenciones y objetivos. Para ello se definen *actos comunicativos* y conversaciones que incluyen una descripción semántica convenida de antemano. El lenguaje KQML (*Knowledge Query Manipulation Language*) aplica la teoría de *actos de habla*, y los estándares de comunicación de agentes de la FIPA.
- **Interacciones Hombre-Máquina**. La comunicación entre un agente artificial y un usuario se realiza mediante dos estrategias, básicamente. Convertir los mensajes del usuario a un lenguaje de comunicación de agentes y aprovechar la tecnología multiagente para simplificar la comunicación con el usuario.

II. 6. COORDINACIÓN

La ***coordinación*** se puede definir como la propiedad de interacción entre un conjunto de agentes que realizan una actividad colectiva [44]. La coordinación implica un cierto grado de predecibilidad mutua y falta de conflictos.

Se pueden distinguir distintos mecanismos para la coordinación [12]:

- **Negociación.** Puede definirse como el proceso de mejorar el acuerdo (reduciendo inconsistencias e incertidumbre) sobre puntos de vista comunes a través del intercambio estructurado de información relevante [45].

  Müller [46] distingue las siguientes categorías dentro del proceso de negociación:

  • *Lenguaje***:** trata las primitivas de comunicación para negociar (proponer, refinar, confirmar, etc.), su semántica, el objeto de la negociación (plan, oferta de tarea, etc.) y los protocolos de negociación.

  • *Decisión***:** presenta qué aspectos tienen en cuenta los agentes para decidir en la negociación como maximizar una función de utilidad, preferencias, estrategias de negociación (ser cooperativos, competitivos, etc.).

  • *Proceso***:** estudio de modelos generales del proceso de negociación y de la conducta global de los participantes.

- **Cooperación Funcionalmente Precisa** (FA/C). Consiste en el intercambio de soluciones tentativas para resolver errores y converger en las soluciones del problema.
- **Estructuración organizativa**. Se basa en el uso de conocimiento común sobre los papeles generales de resolución del problema y patrones de comunicación para reducir la incertidumbre saber cómo deben cooperar.
- **Planificación multiagente**. Los agentes comparten información para construir un plan que define cómo cooperar para la resolución del problema. La planificación puede ser centralizada, si un agente decide cómo se cooperará, o distribuida, cuando los agentes deciden cómo deben actuar en base a su visión (parcial) sobre los planes del resto de agentes. Se destaca dentro de este segundo tipo el algoritmo PGP (*Partial Global Planning*) que se basa en que cada nodo construye un plan que comparte con otros agentes para identificar planes globales parciales. Después se planifican las acciones locales y las comunicaciones con el resto de agentes involucrados en el plan. Por último se

ejecuta el plan, revisando si es necesario replanificar las acciones en base a las desviaciones observadas sobre lo esperado.

II. 7. CONCLUSIÒN

En este capítulo se han establecido los inicios y fundamentos de la tecnología de agentes. También, se han señalado definiciones, características, clasificaciones, entre otros aspectos, sobre los conceptos de *agente* y *SMA*, sumados a la presentación de conceptos propios de la tecnología de agentes como lo son la comunicación y coordinación.

CAPÍTULO III
INGENIERÍA DE SOFTWARE Y METODOLOGÍAS ORIENTADAS A AGENTES

III.1. INTRODUCCIÓN

La construcción de SMA integra tecnologías de distintas áreas de conocimiento: técnicas de la *Ingeniería de Software* (IS) para estructurar el proceso de desarrollo; técnicas de la *Inteligencia Artificial* para dotar a los programas de capacidad para tratar situaciones imprevistas y tomar decisiones; y técnicas de la *Programación Concurrente y Distribuida* para tratar la coordinación de tareas ejecutadas en diferentes máquinas bajo diferentes políticas de planificación [30].

Debido a esta combinación de diversas tecnologías, el desarrollo de SMA no es tarea fácil. Aunque existen plataformas que facilitan el proceso, éstas quedan incompletas sin un proceso de desarrollo de software especializado para agentes que haga la producción de SMA similar a la producción de software convencional. Las técnicas convencionales de ingeniería no tienen en cuenta las especificaciones de los SMA derivadas de su carácter distribuido (planificación de tareas, intercambio de información, etc.), por ello se propusieron, y se siguen proponiendo metodologías basadas en agentes, las cuales parten de un modelo de cómo debe ser un SMA y describen métodos y guías de construcción.

Se podría decir que las *Metodologías Orientadas a Agentes* (MOA) son el resultado de una transferencia de conocimiento desde la IS. Ahora bien, como resultado de sus investigaciones, muchos grupos de trabajo presentan una metodología pero, en realidad, sólo introducen un lenguaje de modelado o enumeran unas pocas actividades de desarrollo [47].

Los investigadores de agentes son los expertos para decir si una metodología captura lo esencial del concepto de agente; sin embargo, para evaluar la capacidad de una metodología, los expertos son los ingenieros de software.

III.2. INGENIERÍA DE SOFTWARE

El Glosario Estándar de Terminología de Ingeniería del Software de IEEE [48] define a la **Ingeniería del Software** como: "(1) La aplicación sistemática de conocimientos científicos y tecnológicos, métodos y experiencia para el diseño, implementación, prueba y documentación de software. (2) La aplicación de un enfoque

sistemático, disciplinado y cuantificable al desarrollo, la operación y el mantenimiento del software; es decir, la aplicación de la ingeniería al software".

Se puede decir, que la IS es la definición de técnicas y procesos software robustos y repetibles que conducen el proceso de desarrollo y permiten coordinar el trabajo en equipo, a través de:

- Lenguajes de modelado para la especificación;
- Técnicas de obtención de requisitos;
- Modelos de análisis;
- Métodos de diseño;
- Plataformas de implementación y despliegue;
- Métodos de validación y verificación, etc.

La IS incluye procesos, métodos y herramientas que permiten elaborar a tiempo y con calidad sistemas complejos basados en computadora [49].

III.2.1. CAPAS DE LA INGENIERÍA DE SOFTWARE

La IS es una tecnología con varias capas. Como se muestra en la Figura III.1, la IS debe basarse en un compromiso organizacional con la *calidad*. Luego, el fundamento para la IS es la capa *proceso*. El *proceso* de software forma la base para el control de la administración de proyectos de software, y establece el contexto en el que se aplican los métodos, se generan los productos del trabajo, se establecen puntos de referencia, se asegura la calidad y se administra el cambio de mantera apropiada. Los *métodos* incluyen un amplio conjunto de tareas como comunicación, análisis, diseño, etc., que proporcionan la experiencia técnica para elaborar software. Las *herramientas* proporcionan un apoyo automatizado o semi-automatizado para el proceso y los métodos.

Figura III.1. Capas de la Ingeniería del Software [49]

III.2.2. INGENIERÍA DE SOFTWARE ORIENTADA A AGENTES

Las arquitecturas de software que contienen muchos componentes interactuando dinámicamente, cada uno con su propio hilo de control, e involucrándose en protocolos de interacción complejos, son mucho más complejas para construirlas correctamente y eficientemente, que aquellas que simplemente computan una función de alguna entrada en un hilo de control. En muchas aplicaciones reales esta es la característica, por lo cual se han desarrollado herramientas y técnicas para modelar, entender, e implementar sistemas en los que las interacciones son la norma.

Desde la década de 1980 los sistemas de software de agentes y SMA han crecido en lo que es una de las áreas más activas de investigación y desarrollo de la computación. Una de las razones más importantes es la que los agentes como sistemas autónomos, son capaces de interactuar con otros agentes para satisfacer sus objetivos de diseño. De ello, se deriva que *la Ingeniería del Software Orientada a Agentes* (ISOA) *es cuando se realizan las tareas de la IS usando como abstracción clave el concepto de agente* [21] [50].

III.2.3. PROCESOS DEL SOFTWARE

III.2.3.1. *Definición*

Un *proceso de software* es un conjunto de actividades y resultados asociados que producen un *producto* software [51]. Para Pressman [49], el *proceso del software* define una estructura para las actividades, acciones y tareas que se requieren a fin de construir software de alta calidad.

Un proceso del software define el enfoque adoptado mientras se hace ingeniería sobre el software. Pero, la IS también incluye tecnologías para el proceso: métodos técnicos y herramientas automatizadas.

III.2.3.2. *Características*

Los procesos del software son inherentemente complejos y comprenden un gran número de actividades como también de características [51] (Véase Tabla III.1).

III.2.3.3. *Modelo de Proceso*

Un *modelo de proceso del software* es una descripción de un proceso del software que se presenta desde una perspectiva particular. Por naturaleza, los modelos son simplicaciones, por lo tanto un modelo de proceso del software es una abstracción de un proceso real [51].

Tabla III.1. Características del proceso

Características del Proceso	*Descripción*
Comprensión	¿Hasta qué punto se define completamente el proceso y qué tan fácil es comprender su definición?
Visibilidad	¿Las actividades del proceso culminan en resultados claros de tal forma que el progreso del proceso es visible externamente?
Apoyo	¿Hasta qué punto las actividades del proceso pueden apoyarse en herramientas CASE?
Aceptación	¿El proceso definido es aceptable y utilizable por los ingenieros responsables de producir el producto software?
Fiabilidad	¿El proceso diseñado es de tal forma que los errores del proceso se evitan o identifican antes de que se conviertan en errores del producto?
Robustez	¿El proceso puede continuar a pesar de los problemas inesperados?
Mantenibilidad	¿El proceso puede evolucionar para reflejar los requerimientos organizacionales cambiantes o las mejoras identificadas del proceso?
Rapidez	¿Qué tan rápido se puede completar el proceso de construcción de un sistema a partir de una especificación dada?

Debido a que esta investigación hace hincapié en la evaluación estructural de las MOA, se toma como válida la *estructura del proceso* (modelo genérico del proceso) para la IS presentada por Pressman [49], la cual define las siguientes cinco actividades estructurales:

1. **Comunicación.** Antes de cualquier trabajo técnico, tiene importancia crítica comunicarse y colaborar con todos los participantes del proyecto. Se busca entender sus objetivos respecto del proyecto, y reunir los requerimientos que ayuden a definir las características y funciones del software.
2. **Planeación.** Consiste en crear el plan del proyecto software (especie de mapa) que guía al equipo de trabajo. En dicho plan se describen las tareas a realizar, los riesgos probables, los recursos que se requieren, los productos que se obtendrán y una programación de las actividades.
3. **Modelado.** Se crean modelos a fin de entender mejor los requerimientos del software y el diseño que los satisfará.

4. **Construcción.** Combina la generación de código (ya sea manual o automática) y las pruebas que se requieren para descubrir errores en éste.
5. **Despliegue.** El software (como entidad completa o como un incremento parcial) se entrega al usuario que lo evalúa y que le da retroalimentación basada en dicha evaluación.

En la Figura III.2, se presenta el proceso de software de manera esquemática. La misma muestra que cada *actividad estructural* está formada por un conjunto de acciones de IS y cada una de éstas se encuentra definida por un *conjunto de tareas* que identifica las tareas del trabajo que deben realizarse, los productos del trabajo que se producirán, los puntos de aseguramiento de la calidad que se requieren y los puntos de referencia que se utilizarán para evaluar el avance.

Otro aspecto importante del proceso del software es el *flujo del proceso* que describe la manera en que están organizadas las actividades estructurales y las acciones y tareas que ocurren dentro de cada una con respecto de la secuencia y el tiempo [49].
En la Figura III.3, se muestran los distintos tipos de flujo del proceso que describen el proceso del software.

Un *flujo de proceso lineal* ejecuta cada una de las actividades estructurales en secuencia. Un *flujo de proceso iterativo* repite una o más de las actividades antes de pasar a la siguiente. Un *flujo de proceso evolutivo* realiza las actividades en forma circular. A través de las cinco actividades, cada circuito lleva a una versión más completa del software. Un *flujo de proceso paralelo* ejecuta una o más actividades en paralelo con otras.

Resultan importantes para esta investigación, las conclusiones sobre la integración de MOA tomadas de Sanz & Mestras [52], en las cuales se concluye que *el proceso de desarrollo que se sigue es iterativo e incremental, basado en casos de uso*.

III.2.4. MÉTODOS Y METODOLOGÍAS

III.2.4.1. *Método*

Según SWEBOK (*Software Engineering Body of Knowledge*) [53], *un método puede ser heurístico, formal o de prototipado*.

- Los métodos heurísticos pueden ser orientados a funciones, orientados a datos u orientados a objetos.
- Los métodos formales se basan en la matemática y se enfocan principalmente en los lenguajes de especificación, el refinamiento de la especificación y la verificación de las propiedades.

- Los métodos de prototipado distinguen entre el estilo de prototipado, el objetivo de prototipado y las técnicas de evaluación de prototipado.

De esta categorización, un **método orientado a agentes** podría rotularse como un método heurístico donde el sistema se ve como una colección de agentes. Es importante remarcar que un método debe proporcionar notación y vocabulario, procedimientos para realizar tareas identificables y pautas para verificar tanto el proceso como el producto.

III.2.4.2. *Componentes de un método*

Todos los métodos se basan en la idea de modelos gráficos de desarrollo de un sistema y en el uso de estos modelos como sistema de especificación o diseño. Los métodos incluyen una variedad de componentes diferentes [51] (Véase Tabla III.2).

Tabla III.2. Componentes del método

Componente	*Descripción*	*Ejemplo*
Descripciones del modelo del sistema	Descripciones de los modelos del sistema que se desarrollará y la notación utilizada para definir estos modelos.	Modelos de objetos, de flujo de datos, de máquina de estado, etc.
Reglas	Restricciones que siempre se aplican a los modelos del sistema.	Cada entidad de un modelo debe tener un único nombre.
Recomendaciones	Heurística que caracteriza una buena práctica de diseño para obtener un modelo bien organizado.	Ningún objeto debe tener más de siete subobjetos asociados a él.
Guías en el proceso	Descripciones de las actividades a seguirse para desarrollar los modelos del sistema y la organización de las mismas.	Los atributos de los objetos deben documentarse antes de definir las operaciones asociadas a un objeto.

III.2.4.3. *Método y Metodología*

Algunos puristas opinan que no es correcto utilizar los términos *método* y *metodología* de manera indistinta, ya que consideran que metodología es únicamente la disciplina que estudia los métodos.

Desde la perspectiva de organismos como FIPA y AgentLink, y para los propósitos de este trabajo, se utiliza el término ***metodología*** para abarcar la definición de procesos, actividades, resultados a producir, lenguajes para describir dichos resultados, guías,

métricas y herramientas asociadas, siguiendo un paradigma de ingeniería de software, en este caso el paradigma de agentes [30].

Además, Dorfman y Thayer [54], indican que una metodología adecuada debe cumplir los siguientes requisitos:

- **Documentada**. El procedimiento debe estar documentado.
- **Repetible**. Su aplicación debe ser repetible.
- **Enseñable**. Los procedimientos descritos deben ser detallados y con ejemplos.
- **Probada**. La metodología está basada en técnicas probadas, es decir, implementa procedimientos fundamentales probados u otras metodologías más simples.
- **Validada**. La metodología ha funcionado correctamente en un gran número de aplicaciones.
- **Apropiada**. La metodología es apropiada al problema que quiere resolverse.

Las metodologías permiten mejorar la calidad de los procesos mediante la trazabilidad y repetibilidad de los mismos, y suelen proporcionar también directrices y métricas para medir la calidad de los productos obtenidos.

III.2.5. METODOLOGÍAS ORIENTADAS A AGENTES

Cuando comenzó a establecerse la tecnología de agentes, y se desarrollaron diversas plataformas y lenguajes para emplear SMA en variadas aplicaciones, empezaron a surgir metodologías que tratan de asistir en el ciclo de vida de la construcción de los SMA para obtener las ventajas de reciclaje de los sistemas y mantenimiento, entre otras. Una *Metodología Orientada a Agentes* (MOA) no es una metodología en sí misma construida sobre los principios de la tecnología de agentes, sino una metodología que se orienta hacia la creación de *software basado en agentes*.

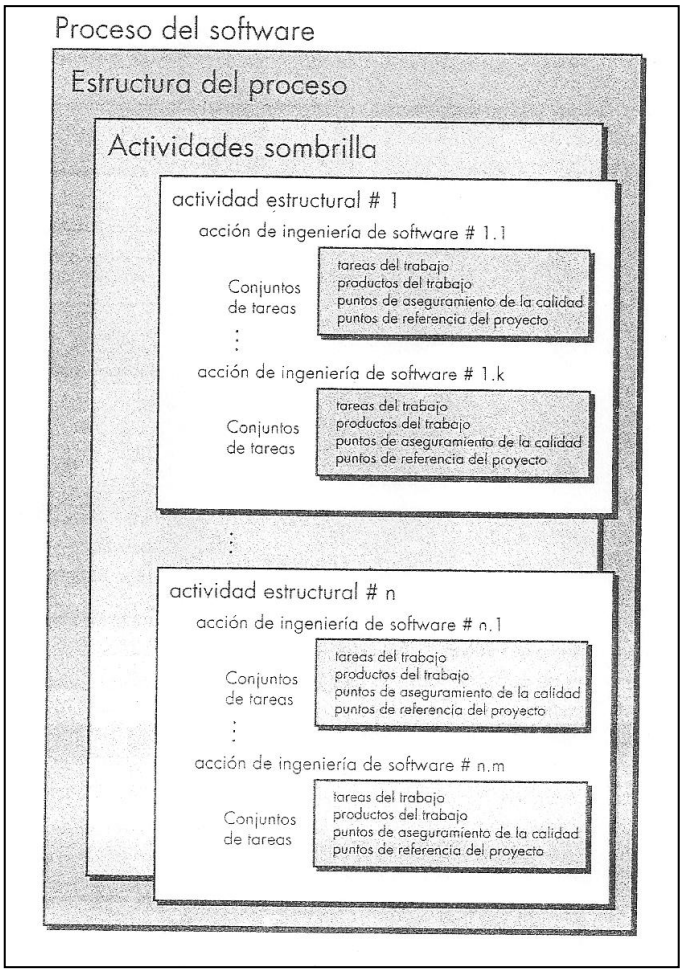

Figura III.2. Estructura de un proceso de software [49]

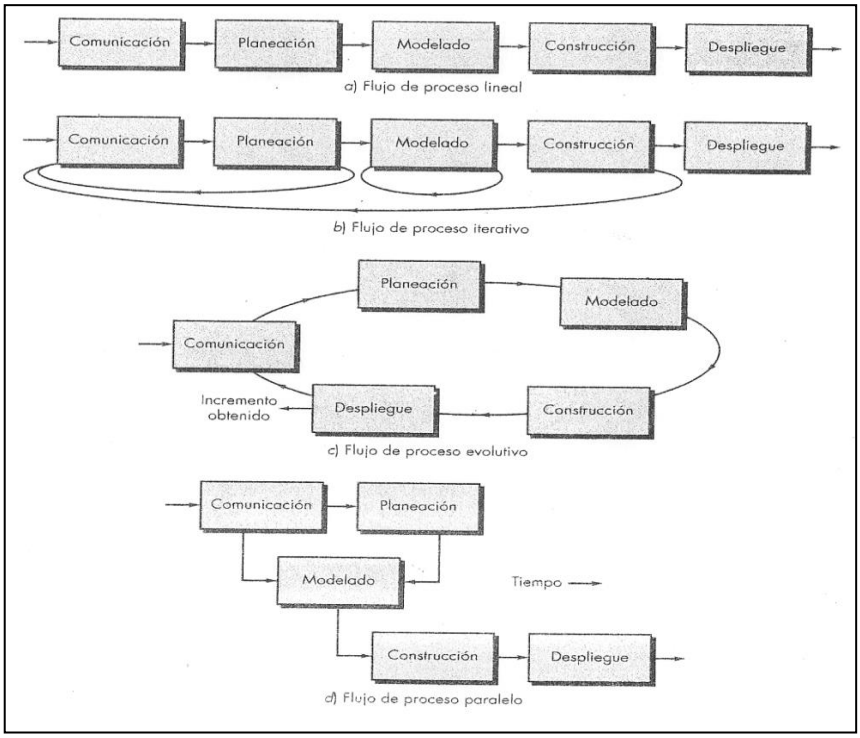

Figura III.3. Flujo del proceso [49]

III.2.5.1. *Desarrollo con agentes*

El primer elemento a considerar ante el desarrollo de un SMA es la *distribución de las entidades que componen el sistema en un entorno abierto*. Un considerable número de componentes y una gran cantidad de información que cambia con el tiempo, sumado a que las relaciones entre las entidades son de cooperación o de competición por los recursos, y que no es posible determinar el efecto que tendrá la realización de una acción en el entorno, contribuyen a un indeterminismo en la dinámica del sistema, para lo cual los agentes tienen capacidades de adaptación y planificación.
Los agentes pueden determinar si una tarea no se está llevando a cabo como estaba previsto, y pueden decidir cambiarla por otra tarea o colaborar con otros agentes [30].

Otra característica es la *heterogeneidad*. La diversidad de dispositivos con distintas capacidades de proceso y memoria, conectados a redes fijas o móviles, implica mayores grados de distribución en la gestión de las entidades del sistema, en la localización del control y en las interacciones. Los agentes se conciben como entidades autónomas que residen en un nodo de una red, e incluso, pueden migrar a otros nodos durante sus existencia (es el caso de los agentes móviles).

Desde el punto de vista del procesamiento y gestión del conocimiento, se necesitan nuevos mecanismos para procesar la información, y la interacción entre componentes del sistema puede requerir mayor abstracción, con más soporte semántico de la información. Ante ello, el uso de ontologías y lenguajes de comunicación de agentes supone un avance para tratar estas cuestiones.

Por último, hay que mencionar la *usabilidad del software*. Ante sistemas altamente configurables, donde hay que realizar un proceso especial para cada usuario, la definición de un agente con capacidad de aprender puede adaptarse a los cambios en el perfil del usuario.

Por ello, para decidir si una aplicación se va a desarrollar como un agente software, hay que preguntarse si tiene características que hagan difícil su construcción con paradigmas tradicionales.

III.2.5.2. *Metodologías orientadas a agentes basadas en metodologías orientadas a objetos*

Las MOA parten de las Metodologías Orientadas a Objetos (MOO) añadiendo las peculiaridades de los agentes: creencias, objetivos, planes, cómo identificar agentes, relaciones e interacciones entre agentes, etc. [55] [56] [57] [58] [59]

Pero, los agentes no son simples objetos por lo que las MOO no abordan los siguientes aspectos diferenciadores:

1. Los objetos tienen una estructura simple (atributos y métodos) mientras que los agentes tienen una estructura compleja.
2. En los objetos el pasaje de mensajes se traduce en invocación de métodos, en los agentes este paso de mensajes se suele modelar como el intercambio de un conjunto predeterminado de actos de habla, con protocolos asociados para negociar o responder a cada acto comunicativo. Además, los agentes realizan un procesado de los mensajes, y pueden decidir ejecutar, o no, la acción correspondiente al mensaje recibido.
3. Los agentes se pueden caracterizar por su estado mental, y las MOO no proporcionan técnicas para modelar cómo los agentes llevan a cabo sus inferencias, su proceso de planificación, etc.
4. Los agentes se caracterizan por su dimensión social. Es necesario definir procedimientos para modelar las relaciones "sociales" de los agentes.

III.2.5.3. *Metodologías orientadas a agentes basadas en metodologías de ingeniería del conocimiento*

Las metodologías de ingeniería del conocimiento pueden proporcionar una buena base para modelar SMA ya que tratan el desarrollo de sistemas basados en conocimiento. Dado que los agentes tienen características cognitivas, estas metodologías pueden proporcionar las técnicas de modelado de la base de conocimiento de los agentes.
Además se pueden reutilizar las bibliotecas de métodos de resolución de problemas y ontologías así como las herramientas desarrolladas en estas metodologías.

Pero, la mayor parte de las metodologías de ingeniería del conocimiento conciben a los SMA como un sistema centralizado. Por tanto, no abordan los aspectos distribuidos o sociales de los agentes, ni en general su conducta proactiva, dirigida por objetivos.

Las soluciones propuestas a este problema son varias extensiones para desarrollar SMA basándose en estas metodologías, añadiendo principalmente el modelado de las interacciones y la conducta proactiva de los agentes, como por ejemplo: *CoMoMAS* [60] y *MAS-CommonKADS.* [61] [62] 63]

III.2.5.4. *Genealogía de las metodologías orientadas a agentes*

Varias metodologías reconocen una descendencia directa de los métodos orientados a objetos (OO). En particular, **MaSE** [64][65] reconoce influencias de Kendall [59], así como una herencia de **AAII** [55], que a su vez estaba fuertemente influenciada por la MOO de Rumbaugh y sus colegas llamada **OMT** [66]. De manera similar, la MOO denominada **Fusión** [67], se dice que fue muy influyente en el diseño de **Gaia** [68][69].

Dos otros enfoques también se han utilizado como base para las MOA: **RUP** [70] sirvió de base para **ADELFE** [71] y también para **MESSAGE** [26], que a su vez, es la base para **INGENIAS** [72]. Más recientemente, **RUP** también se ha utilizado como una de las entradas (junto con **AOR** [73] para **RAP** [74]). El enfoque abierto para el desarrollo de software orientado a objetos ha ampliado de manera significativa su apoyo a los agentes a través de **OPEN** [75]. Por último, otras dos metodologías presentan influencias de programación orientada a objetos: **Prometheus** [76][77], por ejemplo, sugiere el uso de diagramas y conceptos orientados a objetos cuando sean compatibles con el paradigma orientado a agentes. Del mismo modo, **PASSI** se funde en ideas MOO, usando UML como notación principal.

Algo diferente es la metodología **MAS-CommonKADS** [61][62]. Esta es la única metodología sólidamente basada en IA; sin embargo, afirma también haber sido fuertemente influenciada por las MOO, especialmente OMT.

Luego están las metodologías que no reconocen ningún vínculo directo genealógico a otros enfoques, OO o OA: **Tropos** [78][79][80] cuenta con un aporte significativ0 de **i *** [81] y una fuerza distinta en el modelado de requisitos tempranos, centrándose en la descripción de los objetivos de las partes interesadas que describen el "por qué", así como el apoyo más estándar para el "qué" y el "cómo". Este uso del lenguaje de modelado i * (sobre todo en fases de análisis y diseño) le da a Tropos un aspecto diferente de los que utiliza Agent UML (AUML) [82] como notación.

La Figura III.3 ilustra los aspectos mencionados sobre las influencias y evolución de las distintas MOA.

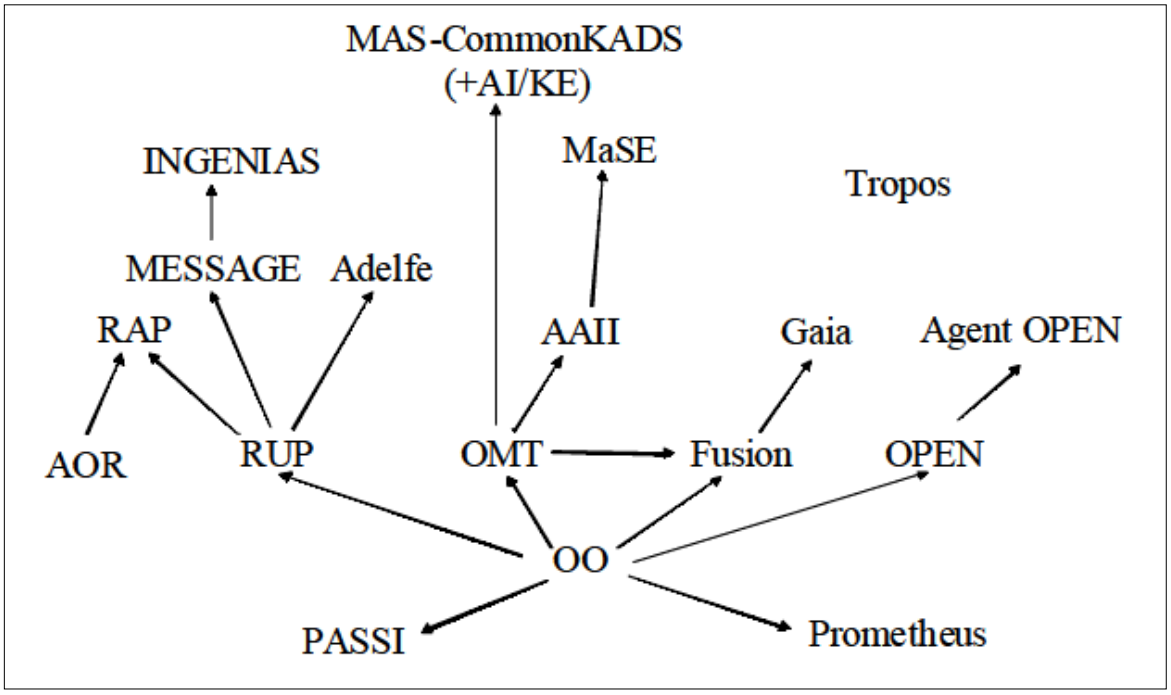

Figura III.3. Linajes e influencias de las metodologías orientadas a agentes más reconocidas (extraída de [83]).

III.2.5.5. *Estado actual*

Actualmente existe un gran número de metodologías orientadas a agentes propuestas, lo que dificulta su evaluación y elección. Además, debido a que las metodologías suelen ir acompañadas de herramientas de desarrollo y notaciones específicas, los sistemas desarrollados siguiendo distintas metodologías seguramente diferirían en el concepto y las propiedades de los agentes diseñados, pues el concepto mismo de "agente" varía según la metodología que utilicemos. Por otro lado, Sturm & Shehory [1] añaden que el excesivo esfuerzo investigador, volcado hacia diversas metodologías, produce resultados que a menudo se solapan entre sí, esfuerzos que si fueran concentrados en unas pocas permitirían abordar un mayor número de aspectos.

Por consiguiente, sería deseable y previsible que la evolución de las MOA tienda hacia unas pocas metodologías más ampliamente aceptadas, ya que esto se unificaría el concepto sobre lo que es un agente y sus propiedades, favoreciendo la elaboración de estándares, herramientas, y el lenguaje común de la comunidad de desarrolladores.

III.2.5.6. *Breve descripción de las MOA más utilizadas*

En base al estudio de Sturm & Shehory [1], a continuación se presenta un resumen de las metodologías que actualmente son más representativas para desarrollar agentes.

- 🕮 ***GAIA**:* propone dos fases: análisis y diseño [84]. En la fase de análisis se identifican los roles del sistema y se modelan sus interacciones, para cada rol se define un esquema basado en términos de permisos, responsabilidades, actividades y protocolos. En la fase de diseño se mapean los roles identificados en la fase de análisis a estructuras concretas, asignando roles a tipos de agentes. Esta metodología tiene dos grandes limitaciones, la primera consiste en que es adecuada solo para el diseño de SMA cerrados, la segunda corresponde a que su notación algunas veces no representa por completo la complejidad de un sistema real. Para sobreponer estas limitaciones se han propuesto versiones y extensiones que se basan en hacer adecuada la metodología para SMA abiertos e implementar técnicas de notación estándar.

- 🕮 ***TROPOS**:* se basa en dos ideas fundamentales: nociones mentales (planes y metas) y la trazabilidad en los requisitos [80]. Esta se divide en cuatro fases, un análisis de requerimientos temprano en el cual se trata el entendimiento del problema que se trabaja, otro análisis de requerimientos posterior en el cual se maneja el ambiente en el que funcionará el sistema y sus necesidades, diseño de arquitectura que define el sistema en términos de conexión a datos, subsistemas y dependencias, y diseño detallado donde se define el comportamiento de cada componente. Tropos soporta técnicas de análisis formales para la verificación de la especificación de requerimientos, que están basadas en un lenguaje propio (*Formal Tropos*) el cual cubre todas las nociones de la metodología. Lo que resalta esta metodología de las demás es su énfasis en los requerimientos y el papel fundamental en los actores y las metas.

- 🕮 ***MaSE**:* según DeLoach [64], esta metodología considera un SMA como una extensión de sistemas orientados a objetos, e igualmente utiliza modelos gráficos que son derivados del estándar UML. Es una especialización de las metodologías de ingeniería de software tradicionales. MaSE brinda una herramienta gratuita que soporta cada una de sus fases, que son análisis y diseño. En la fase de análisis se busca definir el sistema (requerimientos) y

se usan diagramas como el de casos de uso y el de secuencias. En la fase de diseño se define la arquitectura interna de los agentes (roles, clases y conversaciones). MaSE en comparación con GAIA especifica en mayor detalle los procesos de análisis y diseño. Su utilizacion puede ser complementada con Tropos, ya que MaSE no trata la parte de requisitos, que es el fuerte de Tropos.

- ***INGENIAS**:* trae consigo un set de herramientas complementarias para su desarrollo. Esta metodología [72], integra los aspectos más relevantes de otras metodologías desde cinco puntos de vista (Figura III.4). La *Organización* que define la estructura de los agentes, recursos, tareas y metas; *Agente* que trata con la funcionalidad de cada agente (responsabilidades y capacidades); *Metas y Tareas* que definen las tareas desde varios niveles de abstracción; *Interacción* que define los intercambios de información y *Entorno* que describe como el SMA encajara en el sistema donde se implantará. Con estos puntos de vista y una notación estándar, se construyen unos meta-modelos que expresan el SMA. INGENIAS es una metodología derivada de MESSAGE con una definición más completa de los meta-modelos que describen cada punto de vista.

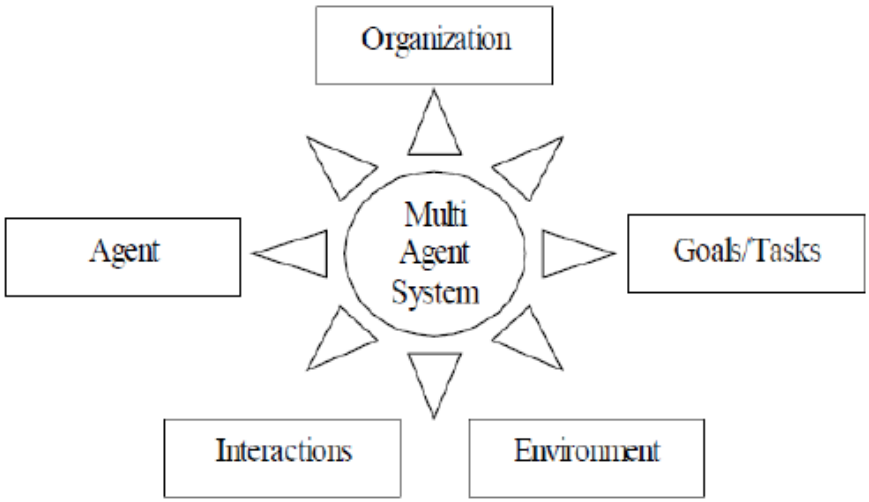

Figura III. 4. Aspectos para el modelado de un SMA basado en INGENIAS [72]

III.3. CONCLUSIÓN

Debido a que el software desempeña un papel importante en las organizaciones y en la sociedad en general, el hecho de que la IS desarrolle mecanismos de abstracción cada vez más potentes y de alto nivel, permite dar soluciones a los cambios contextuales, y los agentes son un claro ejemplo de ello. Por ello, es importante el proceso de desarrollo de los mismos.

En este capítulo se han señalado definiciones, características, clasificaciones, entre otros aspectos, de los conceptos: IS, métodos y metodologías de la IS y proceso software. También se han establecido definiciones, características, clasificaciones, entre otros aspectos, sobre las MOA existentes. Resaltando su estado actual y el deseo de que la evolución de las mismas tienda hacia la estandarización.

CAPÍTULO IV
EVALUACÍON DE LA CALIDAD DEL SOFTWARE: CONCEPTOS, MODELOS Y CRITERIOS

IV.1. INTRODUCCIÓN

La ***calidad*** ha adquirido gran relevancia con el paso del tiempo, ya que es considerada como uno de los principales activos con los que cuenta una organización para mejorar su posición competitiva global [85].

IV.2. CALIDAD DEL SOFTWARE

El American Heritage Dictionary, define a la **calidad** como "*una característica o atributo de algo*". Como atributo de un elemento, la calidad se refiere a las características mensurables, es decir, cosas que se pueden comparar con estándares conocidos, por ejemplo: longitud, color, etc.

En la IS, la mayoría de los métodos, herramientas y procedimientos están orientados hacia un único fin: producir software de alta calidad. Por ello, la **Calidad del Software** se define como: "*la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, y con las características implícitas que se espera de todo software desarrollado profesionalmente"* [49].

La definición anterior sirve para hacer hincapié en tres puntos importantes:

1. Los requisitos del software son la base de las medidas de la calidad. La falta de concordancia con los requisitos es una falta de calidad.
2. Los estándares especificados definen un conjunto de criterios de desarrollo que guían la forma en que se aplica la IS. Si no se siguen esos criterios, casi siempre habrá falta de calidad.
3. Existe un conjunto de requisitos implícitos que a menudo no se mencionan (facilidad de uso, buen mantenimiento, etc.). Si el software se ajusta a sus requisitos explícitos pero falla en alcanzar los requisitos implícitos, la calidad del mismo queda en duda.

Particularmente, para conseguir software de buena calidad es esencial establecer un conjunto de medidas a tomar con respecto al producto, y utilizar modelos y métodos apropiados para controlar el proceso de su desarrollo. La ausencia de defectos, la

capacidad de uso, la seguridad, la confiabilidad y demás especificaciones son elementos que están involucrados en el concepto de calidad del software. Sin embargo, esa calidad debe construirse desde el comienzo, debido a que no es algo que pueda añadir después [86].

En definitiva, la calidad del software es una mezcla compleja de ciertos factores que varían de acuerdo a las aplicaciones y los clientes [87].

IV.2.1. CALIDAD DEL PROCESO SOFTWARE Y DEL PRODUCTO SOFTWARE

Olsina [88] presenta un resumen del documento ISO/IEC 9126 y permite observar un marco conceptual de la calidad. Galván [87] esquematiza gráficamente lo presentado en [88] considerando las diferentes visiones de la misma: calidad del proceso, calidad del producto y calidad en uso (Véase Figura IV.1).

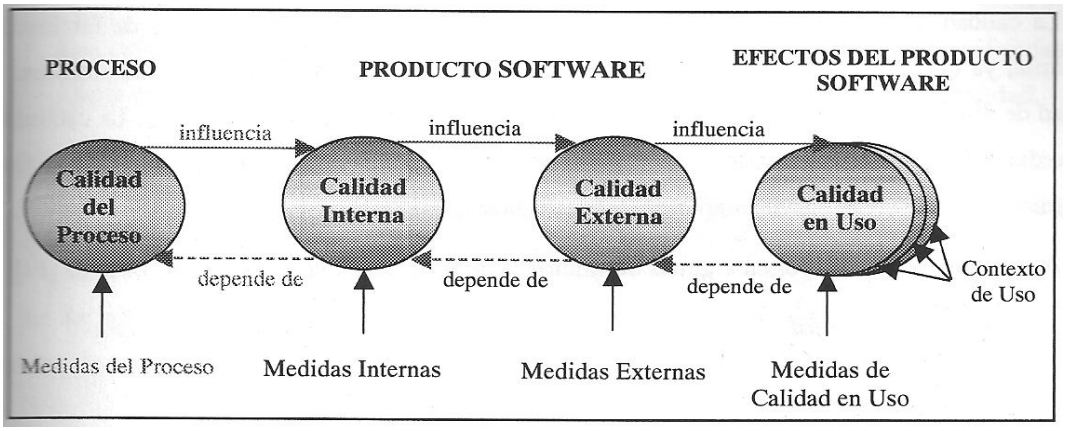

Figura IV.1. Marco conceptual para el modelo de calidad ISO/IEC 9126 (extraída de [87])

La calidad del proceso contribuye a mejorar la calidad del producto, y a su vez, la calidad del producto contribuye a mejorar la calidad en uso. Por lo tanto, las metas que se establezcan para la calidad del producto van a determinar las metas a establecer para la calidad del proceso de desarrollo, ya que la calidad del producto va a estar en función de la calidad del proceso de desarrollo. Sin un buen proceso de desarrollo es casi imposible obtener un buen producto.

IV.3. MODELOS DE CALIDAD

Dado que el concepto general de **calidad** es un concepto que se deriva de un conjunto de subconceptos, cada uno de los cuales se puede evaluar a través de indicadores

o métricas, los ***Modelos de Calidad*** ayudan en la puesta en práctica de ese concepto general [89].

En los modelos de calidad, ésta se define en forma jerárquica (Véase Figura IV.2).

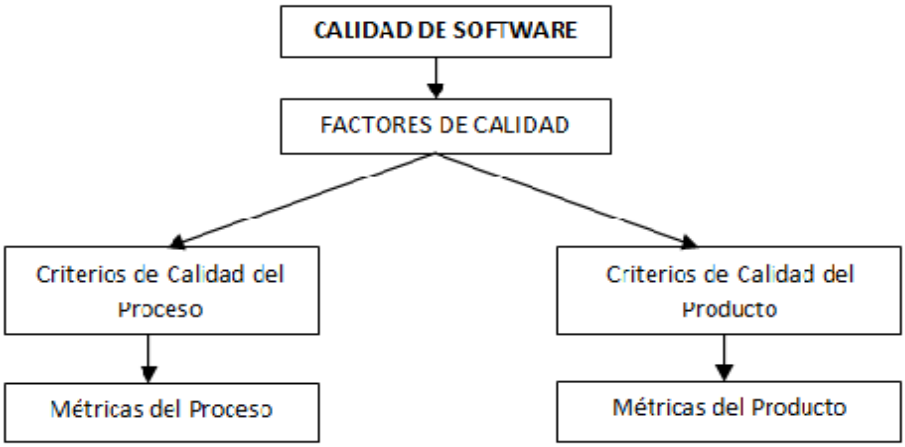

Figura IV.2. Modelo jerárquico de la calidad (extraída de [89])

En el nivel más alto de la jerarquía se encuentran los *Factores de Calidad*, que representan la calidad desde el punto de vista del usuario. Son las características que componen la calidad. Cada uno de estos factores se descompone en un conjunto de *Criterios o atributos de calidad*. Se trata de una visión de la calidad desde el punto de vista del proceso y del producto software.

IV.3.1. MÉTRICAS E INDICADORES DE CALIDAD

La medición es fundamental para cualquier disciplina de ingeniería, y la IS no es una excepción.

IV.3.1.1. *Definición de Métrica*

Se pueden definir las *métricas del software* como *la aplicación continua de técnicas basadas en las medidas de los procesos de desarrollo del software y sus productos, para producir una información de gestión significativa y a tiempo. Esta información se utilizará para mejorar esos procesos y los productos que se obtienen de ellos* [89].

Las métricas del software pretenden mejorar los procesos de desarrollo de software y todos los aspectos de la gestión de aquellos procesos. Son medidas aplicables a todo el ciclo de vida del desarrollo, desde la iniciación, la estimación de costos, seguimiento y control de la fiabilidad de los productos finales, y a la forma en que los productos cambian

a través del tiempo debido a la aplicación de las mejoras. Incluyen el uso de técnicas para detectar y corregir anticipadamente los errores de los distintos componentes de los productos, antes de llegar a la codificación, y controlan el progreso del proyecto, de tal manera de que lo que pueda ocurrir dentro de unos meses se pueda identificar lo más pronto posible.

IV.3.1.2. *Características de las métricas de software*

Una Medida proporciona una indicación cuantitativa de la extensión, cantidad, dimensiones, capacidad y tamaño de algunos atributos de un producto o proceso [49].

La calidad de las medidas debería facilitar el desarrollo de modelos que sean capaces de predecir el comportamiento de determinados parámetros que afectan al desarrollo de productos o de procesos.

Una medida ideal debería ser:

- *Objetiva.*
- *Sencilla*: definible con precisión para que pueda ser evaluada.
- *Fácilmente obtenible* (a un costo razonable).
- *Valida*: debería medir exactamente lo que se quiere medir y no otra cosa.
- *Robusta*: relativamente insensible a cambios poco significativos en el proceso o en el producto.

Además, para una mejor utilización de estas medidas, deberían tener unos valores que se ajusten a una cierta escala de medida.

IV.3.1.3. *Clasificación de las métricas de software*

De manera general, las métricas de software se pueden clasificar en:

- ***Métricas de Producto***: son medidas del producto software durante cualquier fase de su desarrollo, desde los requisitos hasta la instalación. Pueden medir la complejidad del diseño, el tamaño del producto final (fuente u objeto) o el número de páginas de documentación producida.
- ***Métricas de Proceso***: son medidas del proceso de desarrollo del software tales como tiempo de desarrollo total, esfuerzo en días/hombre o meses/hombre de desarrollo del producto, tipo de metodología utilizada o nivel medio de experiencia de los programadores.

IV.4. EVALUACIÓN DE LA CALIDAD DEL SOFTWARE

Al planificar la calidad de un producto software hay que seleccionar cuáles de los factores de calidad van a ser requisitos de calidad del sistema. Para ello hay que tener en cuenta las siguientes cuestiones:

- *La relación con las características peculiares del producto*. Por ejemplo, si el sistema se desarrolla para un entorno en el que hardware cambia rápido, la potabilidad será importante.
- *El costo del factor de calidad frente al beneficio que proporciona*.
- *Las interrelaciones entre factores*. Algunos factores pueden ser conflictivos entre sí. La eficiencia, por ejemplo, está en conflicto con prácticamente todos los demás factores de calidad.

Por ello, la calidad siempre va a depender de los requisitos o necesidades que se deseen satisfacer. Y la *evaluación de la calidad* de un producto va a implicar: *Una comparación entre requisitos preestablecidos y el producto realmente desarrollado* [90].

IV.4.1. TIPOS DE EVALUACIÓN DE LA CALIDAD

En el área de proyectos de evaluación, se utilizan con frecuencia, dos tipos de evaluación: ***cualitativa*** *y* ***cuantitativa***.

Según Olsina [88], los *métodos y técnicas de evaluación cualitativa* se basan generalmente en una lista de características a ser analizadas para un producto o productos competitivos. La lista puede contener características de distinto tipos (procesos, productos o procesos del sistema a evaluar). Luego del proceso de evaluación, durante el análisis y conclusiones, los tomadores de decisión crean para cada sistema a comparar, una lista de ventajas y desventajas. Finalmente, mediante un mecanismo que compara las ventajas y desventajas listadas, arriban a un ranking final.

Este enfoque es conveniente cuando el objeto de la evaluación y el proceso de decisión son suficientemente simples. Pero, en un proceso de evaluación, comparación y selección en donde intervienen varias características y atributos para cada sistema seleccionado, sumado a las relaciones entre los mismos, el enfoque carece de propiedades de precisión y justificación objetiva. Esta dificultad puede minimizarse mediante el uso de un enfoque cuantitativo.

Un *método o una técnica de evaluación cuantitativa* proveen un proceso de evaluación flexible, estructurado, preciso, y basado en principios ingenieriles para brindar

indicadores cuantitativos elementales, parciales y globales, los cuales son usados como base y justificación de mejores decisiones.

IV.4.2. OPERACIONALIZACIÓN DE VARIABLES

Una de las actividades claves de un proceso de evaluación es el proceso de operacionalizacion de variables.

La operacionalizacion es el proceso por el cual transformamos o traducimos una variable teórica en variables empíricas, directamente observables, con la finalidad de poder medirlas [91].

Así, una vez operacionalizada una variable y reducida a aspectos observables, puede ser dividida en dimensiones y elementos; estos últimos, deben ser observables y concretos, y se los denomina *indicadores*.

IV.4.3. MARCO METODOLÓGICO DE EVALUACIÓN DE LA CALIDAD

Existen varios estándares, enfoques, procesos y marcos de evaluación de la calidad. Para esta investigación se utilizará el ***Enfoque orientado a metas*** **GQM** (Goal-Questions-Metrics) tomado de Basili [92], debido a su similitud con el modelo jerárquico de presentar a la calidad visto anteriormente (Véase apartado IV.3.).

IV.4.3.1. *Enfoque orientado a metas GQM*

Para que las medidas sean efectivas deben estar focalizadas en metas especificas, aplicadas a todo o parte del ciclo de vida de los objetos e interpretadas en función da la comprensión del contexto organizacional [88]. Esto implica definir las mediciones bajo una estrategia "top-down" (de arriba hacia abajo).

El enfoque para mediciones en proyectos de software GQM responde a esa estrategia, es decir, deriva mediciones a partir de metas (goals). Dado un conjunto de metas del proyecto, teniendo en cuenta las características y atributos deseables de los objetos, se construye y refina un conjunto de preguntas (questions) para cada meta, y en función de cada pregunta, se eligen las medidas (metrics) apropiadas (Véase Figura IV.3).

El siguiente, es un ejemplo de aplicación de GQM para expresar el objetivo de un estudio de IS: *El* ***propósito*** *de este estudio* ***es*** *caracterizar* ***el*** *efecto de la programación en parejas* ***sobre*** *los esfuerzos del programador y la calidad del programa* ***desde el punto de vista*** *de los administradores de software* ***en el contexto*** *de una pequeña empresa de desarrollo web.*

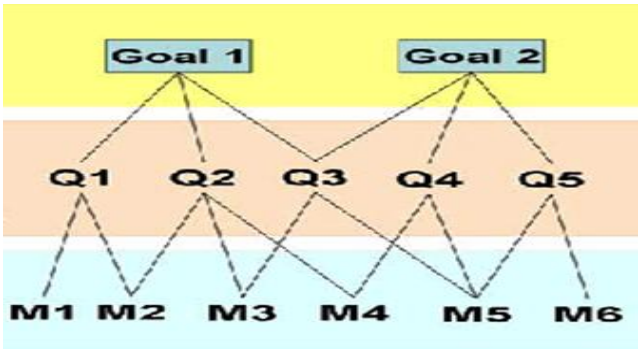

Figura IV.3. Enfoque para mediciones GQM (extraída de [92])

IV.5. EVALUACION DE METODOLOGIAS ORIENTADAS A AGENTES

En Sturm & Shehory [1] se observa que las metodologías proporcionan un conjunto de componentes que se puede resumir en:

- Conjunto de Técnicas (Métricas, Calidad, Estándares, Herramientas)
- Lenguaje de Modelado, compuesto de una notación y un metamodelo, es decir, el modelo para modelar el sistema. El metamodelo se corresponde y se define a partir del concepto que tiene una metodología concreta de las distintas abstracciones (agente, mensaje, rol, etc.)
- El proceso del ciclo de vida con sus aspectos relacionados de: gestión, roles, procedimientos y entregables.

Ante la cantidad de metodologías existentes y la falta de estandarización de las mismas, la *evaluación de metodologías* presenta varias dificultades:

- La variación en los conceptos utilizados en su terminología.
- La influencia que reciben de otros (distintos) paradigmas.
- La variación en los elementos que la integran. Por ejemplo, algunas proporcionan sólo notaciones gráficas, mientras que otras integran varios aspectos (notaciones, directrices, etc.).

IV.5.1. CRITERIOS DE EVALUACION DE MOA

Los distintos componentes de una metodología, nombrados anteriormente (Véase apartado IV.5.), están asociados con cada una de las cuatro principales divisiones siguientes:

- **Conceptos y propiedades:** Se refiere a sí una metodología utiliza las nociones básicas de un agente o un SMA. Son palabras claves como autonomía, reactividad, proactividad, metas, tareas, mensajes, protocolos, sociedades, etc.
- **Notaciones y técnicas de modelado:** Las notaciones se refieren a símbolos técnicos de un paradigma, y las técnicas de modelado a un conjunto de elementos de descripción con varias capas de abstracción que permiten definir bien una parte, en este caso, los agentes, los roles, las organizaciones, etc.
- **Proceso de desarrollo:** Se ocupa de definir adecuadamente una serie de acciones que, precisamente cuando se realizan, traen como resultado un sistema con cierta funcionalidad.
- **Pragmática:** trata los aspectos prácticos de las metodologías, es decir, aspectos que hacen referencia a la manera en que cierta metodología ha sido introducida y a su proceso de evolución.

IV.5.2. MARCO DE EVALUACION DE MOA PROPUESTO POR STURM & SHEHORY

El marco de evaluación propuesto por Sturm & Shehory [1] intenta resolver las dificultades de evaluar metodologías siguiendo los apartados que se enumeran a continuación.

IV.5.2.1. *Evaluación de Conceptos y Propiedades*

Un *concepto* es una abstracción o noción inferida o derivada de instancias específicas en un dominio del problema.

Una *propiedad* es una capacidad especial o una característica.

En este punto se evalúa el soporte a los conceptos de los SMA (Véase Tabla IV.1.).

Tabla IV.1. Conceptos de los SMA

Concepto	*Definición*
Agente	Programa de computadora que puede aceptar tareas y averiguar qué acciones realizar con el fin de ejecutar estas tareas sin supervisión. Es capaz de realizar un conjunto de tareas y proporcionar un conjunto de servicios.
Creencia	Hecho que se cree que es verdadero acerca del mundo.
Deseo	Hecho cuyo valor actual es falso y el agente (que posee el deseo) preferiría que fuera cierto. Un deseo puede ser utilizado como un objetivo. Los deseos de una entidad puedan ser contradictorios y el conjunto de objetivos dentro de un agente debe ser coherente.

Tabla IV.1. Conceptos de los SMA (continuación)

Concepto	*Definición*
Intención	Hecho que representa la forma de realizar una deseo. Algunas veces conocido como un plan.
Mensaje	Medio de intercambio de datos u objetos entre entidades.
Norma	Pauta que caracteriza a una sociedad. Un agente que desea ser un miembro de una sociedad es necesario que sigua todas sus normas. Una norma puede ser referida como una regla.
Organización	Grupo de agentes que trabajan en conjunto para lograr un objetivo común. Una organización se compone de funciones que caracterizan a los agentes, que son miembros de la organización.
Protocolo	Conjunto ordenado de mensajes que definen los patrones admisibles de un tipo particular de interacción entre entidades.
Rol	Representación abstracta de la función de un agente, servicio, o de identificación dentro de un grupo.
Servicio	Interfaz suministrada por un agente para el mundo exterior. Se trata de un conjunto de tareas que ofrecen algunos agentes Un servicio puede consistir en otro servicio.
Sociedad	Conjunto de agentes y organizaciones que colaborar para promover sus metas individuales.
Tarea	Parte del trabajo que se puede asignar a un agente o que puede ser realizado por éste. Puede ser una función y tener limitaciones de tiempo. También conocida como acción.

También, se evalúa si la metodología recoge las propiedades de los SMA (Véase Tabla IV.2.).

Tabla IV.2. Propiedades de los SMA

Propiedad	*Definición*
Autonomía	Capacidad de un agente para operar sin supervisión.
Reactividad	Capacidad de un agente para responder oportunamente a los cambios en el medio ambiente.
Proactividad	Capacidad de un agente para perseguir nuevas metas.
Sociabilidad	Capacidad de un agente para interactuar con otro agente, mediante el envío y recepción de mensajes, encaminamiento estos mensajes, y la comprensión de los mismos.

IV.5.2.2. *Evaluación de Notaciones y Técnicas de modelado*

Las *notaciones* son un sistema técnico de símbolos utilizados para representar elementos dentro de un sistema.

Una *técnica de modelado* es un conjunto de modelos que describen un sistema en diferentes niveles de abstracción y en diferentes aspectos del mismo.

En la Tabla IV.3, se evalúan los aspectos relacionados a las notaciones y técnicas de modelado.

Tabla IV.3. Aspectos de las notaciones y técnicas de modelado

Aspecto	*Definición*
Accesibilidad	Atributo que se refiere a la facilidad o sencillez de la comprensión y el uso de un método.
Analizabilidad	Capacidad para comprobar la consistencia o implicaciones de los modelos, o para identificar aspectos que parecen ser poco claros, como interrelaciones entre operaciones aparentemente no relacionadas. Es generalmente apoyada por herramientas automáticas.
Abstracción	Capacidad para hacer frente a varios niveles de abstracción (es decir, varios niveles de detalle). A veces, se necesitan requisitos de alto nivel, mientras que en otras situaciones se necesitan más detalles.
Ejecutabilidad	Capacidad de realizar una simulación o generar un prototipo de al menos algunos aspectos de una especificación.
Expresividad	Capacidad de presentar conceptos y propiedades del sistema.
Modularidad	Capacidad de especificar un sistema de forma iterativa incremental. Es decir, cuando nuevos requisitos se añaden no deberían afectar las especificaciones existentes, pero si se puede usar.
Precisión	Atributo de desambiguación. Permite a los usuarios evitar la mala interpretación de los modelos existentes.

IV.5.2.3. *Evaluación del Proceso*

Un *proceso de desarrollo* consiste en una serie de acciones, cambios y funciones que, cuando se realizan, derivan en un producto software.

En este punto, se evalúan aspectos del proceso de desarrollo (Véase Tabla IV.4.).

Tabla IV.4. Aspectos del proceso de desarrollo

Aspecto	*Definición*
Contexto de desarrollo	Se debe evaluar si la metodología se adapta a la creación de nuevo software, reingeniería, ingeniería inversa, reutilización, etc.
Fases del ciclo de vida	Requisitos, análisis, diseño, implementación, pruebas, gestión de proyectos, etc.
Actividades en cada fase	Descripción de las actividades.
Entregables	Se refiere a la documentación, modelos, código u otro tipo de entregables como planes de prueba y prototipos.

Tabla IV.4. Aspectos del proceso de desarrollo (continuación)

Aspecto	*Definición*
Mecanismos de validación y verificación	Comprueban si un método tiene reglas para la verificación de que sus entregas se ajustan a los requisitos. Comprueban si un método tiene reglas de validación para que las prestaciones de una etapa sean consistentes con su anterior etapa.
Aseguramiento de la calidad	Guías para el control de la calidad.
Gestión del proyecto	Directrices para la gestión del proyecto.

Se definen ***fases del ciclo de vida*** para el proceso de desarrollo y se muestran en la Tabla IV.5.

Tabla IV.5. Fases del ciclo de vida

Requisitos	Consiste en la especificación (por lo general, en forma de texto libre) de las necesidades del sistema.
Análisis	Describe las características externamente visibles del sistema, por ejemplo, funcionalidad, rendimiento.
Diseño	Define la forma en que el sistema llevará a cabo sus requisitos. Los modelos definidos en la etapa de análisis son refinados y/o transformados para que representen la lógica y la física de la naturaleza del producto de software.
Implementación	Convierte los modelos desarrollados en la etapa de diseño en software ejecutable en el ambiente del sistema. Esto implica codificación manual de las unidades de programa, generación automatizada de tales códigos, o utilización de bibliotecas reutilizables.
Pruebas	Se centran en garantizar que cada entregable de cada etapa se ajusta a las direcciones y requisitos que declaró el o los usuarios.

IV.5.2.4. *Evaluación de Aspectos Pragmáticos*

La *pragmática* se refiere a los aspectos prácticos del uso y de implementación de una metodología.

En particular, el marco sugiere examinar las cuestiones presentadas en Tabla IV.6.

Tabla IV.6. Aspectos pragmáticos

Aspecto	*Definición*
Recursos de capacitación y costos	Qué recursos hay disponibles para apoyar el uso de la metodología con respecto a la capacitación y el presupuesto, y las alternativas de elección de los mismos.

Tabla IV.6. Aspectos pragmáticos (continuación)

Aspecto	*Definición*
Experiencia necesaria	Comprobar si los usuarios candidatos a utilizar la metodología cumplen las condiciones requeridas para el uso de la misma.
Idoneidad del lenguaje (paradigma y arquitectura)	Permite verificar si una metodología se adhiere a la infraestructura y el conocimiento del proyecto u organización.
Aplicabilidad del dominio	Hasta qué punto la metodología se adhiere al dominio del problema.
Escalabilidad	Si la metodología es apropiada para el manejo de distintos tamaños de la aplicación.

IV.5.2.5. *Métrica*

Se evalúan las propiedades anteriores (desde V.5.2.1. a V.5.2.4.) del 1 al 7 de la siguiente manera:

1. No se trata la propiedad

2. Se trata la propiedad sin dar detalles

3. Se trata la propiedad de forma limitada, no se tratan temas relacionados con la propiedad.

4. Se trata la propiedad, aunque faltan algunos temas importantes relacionados.

5. Se tratan la propiedad y la mayor parte de temas relacionados.

6. Se trata la propiedad y los temas relacionados, con menores deficiencias.

7. Se trata la propiedad y los temas relacionados de forma completa.

IV.6. CONCLUSIÓN

Debido a la importancia que hoy en día tiene el concepto de calidad, la calidad de los productos software y la calidad de los procesos de desarrollo corren la misma suerte. Por ello, es necesario definir metas para la calidad del producto, las cuales van a determinar los objetivos del proceso de desarrollo, porque la calidad del producto va a depender, entre otros aspectos, de estos.

En este capítulo se han señalado definiciones, características, clasificaciones, entre otros aspectos, de los conceptos: Calidad, Calidad de software, Modelos de calidad y Evaluación de la calidad, entre otros.

CAPÍTULO V
REVISIÓN Y MODIFICACIÓN DEL MARCO DE EVALUACIÓN PROPUESTO POR STURM & SHEHORY

V.1. INTRODUCCIÓN

No existen muchos trabajos de evaluación de MOA. Sturm & Shehory [1] presentan una evaluación basada en los componentes de las metodologías de Ingeniería del Software Orientadas a Agentes (ISOA) (Véase Cap.IV.5.), en la que incluyen criterios relacionados con la IS y criterios relativos a los conceptos de agentes.

Partiendo de los conceptos y características analizados en los capítulos anteriores de esta investigación, se revisan los aspectos cubiertos por el marco de Sturm & Shehory [1] para realizar agregaciones y/o modificaciones al mismo, buscando definir los aspectos que una MOA debería incluir.

V.2. REVISIÓN DE LOS CONCEPTOS Y PROPIEDADES ESTABLECIDOS EN EL MARCO DE STURM & SHEHORY

Gómez Álvarez, en su tesis doctoral [5], realiza modificaciones al marco de Sturm & Shehory [1] buscando obtener un marco aplicable y completo al problema de la evaluación de metodologías. Tales modificaciones son las siguientes:

- Desagrega la propiedad de Reactividad en la clasificación de agentes reactivos: simple respuesta a estímulo, coordinación (mecanismos de inhibición, arquitecturas de subsunción), cooperación entre agentes, reproducción y auto-organización.
- Asocia a la Proactividad el concepto de Racionalidad propio de Agentes Cognitivos/Deliberativos.
- En Sociabilidad distingue las capacidades de Cooperación, Interacción, Negociación y Organización.
- Añade las propiedades: Movilidad, Veracidad, Benevolencia.
- Agrupa los conceptos Creencias / Deseos / Intenciones bajo la sigla en ingles "**BDI".**
- Añade el concepto de Emergencia.
- Agrega el concepto de Entorno y sus propiedades (Accesible/Inaccesible, Estático/Dinámico, Determinista/No Determinista, Discreto/Continuo).
- Añade el concepto de Interacción al de Protocolo, por entender que el concepto de interacción es más amplio que el concepto de protocolo,

pudiendo existir interacciones no protocolarias (por ejemplo en SMA abiertos, heterogéneos, adaptativos, o agentes de interfaz con usuarios humanos).

En su tesis de magister, Zohreh Akbari [6], presenta un marco de evaluación con diferentes niveles de criterios basado en Sturm & Shehory [1]. Muestra los cuatro aspectos cubiertos por Sturm & Shehory [1] sumado a criterios para evaluar el soporte a la IS y los criterios de comercialización del mismo.

Recordando las características propias de los agentes que se utilizan en este trabajo (Véase Cap. II.3.1.1.) y teniendo en cuenta las propiedades y conceptos brindados por el marco de Sturm & Shehory [1] más las agregaciones y/o modificaciones vistas en [5] y [6], se obtienen las tablas comparativas de conceptos propios de los agentes y propiedades (Tablas V.1 y V.2).

Tabla V.1. Conceptos propios de los agentes según diversos autores

Conceptos		
Sturm & Shehory	Gómez Álvarez	Akbari
Agente		
Creencia	BDI	Actitudes mentales
Deseo		
Intención		
Mensaje		Mensaje
Norma		Norma
Organización		Organización
Protocolo	Protocolo - Interacción	Protocolo
Rol		Rol
Servicio		Servicio
Sociedad		Sociedad
Tarea		Tarea
	Emergencia	
	Entorno • Accesible/Inaccesible, • Estático/Dinámico, • Determinista/No Determinista, • Discreto/Continuo.	

Tabla V.2. Propiedades de los agentes según diversos autores

Propiedades			
Sturm & Shehory	Características propias	Gómez Álvarez	Akbari
Autonomía	Autonomía		Autonomía
Reactividad	Reactividad	Reactividad • simple respuesta a estímulo, • coordinación (mecanismos de inhibición, arquitecturas de subsunción), • cooperación, • reproducción, • auto-organización.	Reactividad
Proactividad	Proactividad	Proactividad- Racionalidad	Proactividad
Sociabilidad	Sociabilidad	Sociabilidad • Cooperación, • Interacción, • Negociación, • Organización	Sociabilidad
	Adaptabilidad		
	Movilidad	Movilidad	
	Racionalidad	Proactividad- Racionalidad	
		Veracidad	
		Benevolencia.	

Como observación, se comenta que las celdas en blanco significan que los autores citados no hacen referencia a los conceptos y propiedades nombrados.

V.2.1. CONCEPTOS Y PROPIEDADES RESULTANTES DE LA REVISIÓN

A modo de análisis de las tablas comparativas presentadas (véase Tablas V.1 y V.2), se realizan las siguientes consideraciones:

- No se tendrán en cuentan en cuentan los conceptos individuales de Creencia, Deseo e Intención, ni su agrupación bajo BDI o actitudes mentales, debido a que para esta investigación están considerados en la asociación Proactividad-Racionalidad.
- Se considera la asociación Protocolo - Interacción, debido a la importancia de ampliar el concepto de Protocolo.

- Se tendrán en cuenta las propiedades definidas para Entorno, ya que permiten presentar los distintos tipos de entornos en los cuales puede desempeñarse un agente.
- Se toma como valido el concepto de Agente para definir que es un agente software.
- Se recogen los conceptos de Mensaje e Interacción-protocolo bajo el criterio de Comunicación; los de Tarea y Servicio bajo el criterio de Operación y los conceptos: Norma, Mensaje, Organización, Roles, Sociedad bajo Socializacion, debido a que son propios de la tecnología de agentes y permiten la modelización física de los mismos.
- No se tendrá en cuenta la desagregación de Reactividad, debido a que la presente investigación pretende analizar las MOA de manera general y no particularizadas para un tipo de agente (en el caso citado, los agentes reactivos).
- Se toma como válida la asociación Proactividad-Racionalidad, debido a la importancia que tiene el hecho de que un agente persiga nuevas metas siempre con un comportamiento apropiado.
- No se tendrán en cuenta las capacidades definidas por Gómez-Sanz para Sociabilidad debido a que la propiedad nombrada por si misma ya define la interacción y la comunicación que un agente debe tener con otros agentes, con su entorno y sus usuarios.
- Se toman las propiedades de Autonomía, Adaptabilidad y Movilidad.
- Se considera como propiedad la agregación del concepto Emergencia.
- No se tendrán en cuanta las propiedades: Veracidad y Benevolencia, de forma individual, debido a que para esta investigación están presentes en la propiedad Racionalidad.

Teniendo en cuenta las consideraciones anteriores y la agrupación de conceptos propuesta en [6], se obtiene el siguiente conjunto de conceptos extendido presentados en la Tabla V.3.

También, se obtiene el conjunto de propiedades extendido mostrado por la Tabla V.4.

Tabla V.3. Conceptos propios de los agentes resultantes de la revisión

Conceptos		
Agente	Programa de computadora que puede aceptar tareas y averiguar qué acciones realizar con el fin de ejecutar estas tareas y proporcionar servicios sin supervisión.	
Entorno - Accesible/Inaccesible - Estático/Dinámico, - Determinista/No Determinista - Discreto/Continuo	Ámbito o medio de los agentes. Puede proporcionar datos y eventos a los agentes y es modificado por las actuaciones de los agentes sobre él.	
Comunicación	***Mensaje***	Medio de intercambio de datos u objetos entre entidades.
	Interacciones - Protocolos	Interacción es un intercambio de mensajes en entidades. Protocolo es la formalización de los patrones validos de interacción entre entidades.
Operación	***Tarea***	Parte del trabajo que se puede asignar a un agente o que puede ser realizado por éste. Puede ser una función y tener limitaciones de tiempo. También conocida como acción.
	Servicio	Interfaz suministrada por un agente para el mundo exterior. Conjunto de tareas que ofrecen algunos agentes al consumidor del servicio. Un servicio puede consistir en otro servicio.
Socialización	***Roles***	Representación abstracta de la función de un agente, servicio, o de identificación dentro de un grupo (perfil).
	Norma	Pauta que caracteriza a una sociedad. Un agente que desea ser un miembro de la sociedad es necesario que sigua todas sus normas. Una norma puede ser referida como una regla.
	Organización	Grupo de agentes que trabajan en conjunto para lograr un objetivo común. Una organización se compone de funciones que caracterizan a los agentes, que son miembros de la organización.
	Sociedad	Conjunto de agentes y organizaciones que colaboran para conseguir sus metas individuales.

Tabla V.4. Propiedades de los agentes resultantes de la revisión

Propiedades	
Autonomía	Capacidad de un agente para operar sin supervisión.
Reactividad	Capacidad de un agente para responder oportunamente a los cambios en el medio ambiente.
Proactividad - Racionalidad	Capacidad de un agente para perseguir nuevas metas. Es el principio por el cual un agente nunca actuará de modo que vaya en contra de sus objetivos y sí a favor de ellos.
Emergencia	El comportamiento del sistema emerge de las relaciones entre agentes y con el entorno.
Adaptabilidad	Cambia su comportamiento en función de su experiencia previa.
Movilidad	Es capaz de transportarse de una máquina a otra.
Sociabilidad	Para poder interactuar con su entorno, el agente típicamente deberá interactuar con el mundo exterior. Esta interacción puede darse a distintos niveles, pero en general, el agente necesitará comunicarse con el entorno, con otros agentes, y con sus usuarios.

V.3. REVISIÓN DE LAS NOTACIONES Y TÉCNICAS DE MODELADO ESTABLECIDAS EN STURM & SHEHORY

Teniendo en cuenta las notaciones y técnicas de modelado brindadas por el marco de Sturm & Shehory [1], más las características vistas en la bibliografía revisada**,** se obtiene la siguiente tabla comparativa:

Tabla V.5. Características de las notaciones y técnicas de modelado

Notaciones y técnicas de modelado		
Sturm & Shehory	Akbari	Conceptos estáticos y dinámicos expresados por una notación
Accesibilidad	Facilidad de uso	
Analizabilidad		
Abstracción		
Ejecutabilidad		Módulo de Interfaz (implementaciones físicas de los objetos representados)

Tabla V.5. Características de las notaciones y técnicas de modelado (continuación)

Notaciones y técnicas de modelado		
Sturm & Shehory	Akbari	Conceptos estáticos y dinámicos expresados por una notación
Expresividad	Expresividad	Agregación (objetos componentes que construyen un objeto compuesto)
		Especialización (representación como generalización, o especialización de otro objeto)
		Comunicación
		Cambios de estado
		Sincronización
		Interacción
Modularidad	Modularidad	
Precisión	Precisión	
	Trazabilidad	
		Reutilización

V.3.1. NOTACIONES Y TÉCNICAS DE MODELADO RESULTANTES DE LA REVISIÓN

La IS conduce el proceso de desarrollo y permite coordinar el trabajo en equipo, a través de: lenguajes de modelado para la especificación, técnicas de obtención de requisitos, modelos de análisis, métodos de diseño, etc.

Luego de analizar la Tabla V.5 se realizan las siguientes consideraciones:

- Se toma como válido el concepto de Expresividad que hace referencia a la descripción de los componentes y funciones del sistema.
- Se integran los conceptos de Agregación, Especialización, Comunicación, Cambios de estados, Sincronización e Interacción a la característica de Expresividad.
- Se toman como válidas las propiedades: Facilidad de uso, Modularidad, Precisión y Reutilización.

Se define el conjunto de propiedades para evaluar si una metodología se adhiere a las técnicas de modelado en la Tabla V.6.

Tabla V.6. Notaciones y técnicas de modelado resultantes de la revisión

Notación y técnicas de modelado	
Facilidad de uso	Facilidad o sencillez de comprensión y uso de un método.
Expresividad - Agregación - Especialización - Comunicación - Cambios de estado - Sincronización - Interacción	Capacidad de presentar conceptos y propiedades del sistema.
Modularidad - Mecanismos de partición - Consistencia incremental	Capacidad de especificar un sistema de forma iterativa incremental y de comprobar la consistencia o implicaciones de los modelos.
Precisión - Simbólica - Sintáctica - Semántica	Permite a los usuarios evitar la mala interpretación de los modelos existentes.
Reutilización	Índica el grado en que un modelo puede ser utilizado por otro proceso.
Extensibilidad	Capacidad para ampliar los conceptos básicos y bloques de construcción.

V.4. REVISIÓN DE LOS PROCESOS DE DESARROLLO ESTABLECIDOS EN EL MARCO DE STURM & SHEHORY

Revisando los aspectos del proceso de desarrollo brindados por Sturm & Shehory [1], las características del proceso software de Sommerville [51] y los criterios de Akbari [6], se obtiene la tabla comparativa Tabla V.7.

Tabla V.7. Aspectos del proceso de desarrollo

Proceso de desarrollo		
Sturm & Shehory	Características del proceso software	Akbari
Contexto de desarrollo		Contexto de desarrollo
Fases del ciclo de vida		Ciclo de vida
Actividades en cada fase		
Entregables	Visibilidad (resultados)	

Tabla V.7. Aspectos del proceso de desarrollo (continuación)

Proceso de desarrollo		
Sturm & Shehory	Características del proceso software	Akbari
Mecanismos de validación y verificación		Planes de administración
Aseguramiento de la calidad		
Gestión del proyecto		
	Mantenibilidad	
	Rapidez	
	Herramientas de soporte	
	Fiabilidad	
	Robustez	
	Aceptación	
		Perspectiva de desarrollo

También es importante definir las fases que integraran el ciclo de vida a seguir. Para ello se toman las fases brindadas por Sturm & Shehory [1], las actividades estructurales de Pressman [93] y otras visiones de desarrollo de software para obtener la siguiente tabla comparativa TablavV.8.

Tabla V.8. Diversas visiones para el desarrollo de sistemas

Sturm & Shehory	Actividades Estructurales	Instituto Tecnológico de las Américas	Ciclo de vida tradicional
	Comunicación	Investigación preliminar	Definición
Requisitos		Requerimientos	
		Levantamiento de información	
	Planeación		
Análisis	Modelado	Análisis	Análisis
Diseño		Diseño	Diseño
Implementación	Construcción	Implementación	Programación
Pruebas		Prueba	Instalación
	Despliegue	Documentación	
		Mantenimiento	Mantenimiento

V.4.1. PROCESOS DE DESARROLLO RESULTANTES DE LA REVISIÓN

A modo de análisis de la Tabla V.7, se realizan las siguientes consideraciones:

- Se toma como válida la similitud de los conceptos: Entregables - Visibilidad debido a que hacen igual referencia a los resultados de cada fase de desarrollo (documentación, código, etc.).
- Se integran los conceptos de Plan de proyecto, Generación de especificación, Validación de diseño, Generación de código y Generación de documentación bajo el criterio de Planes de administración.
- Se toma como válida la característica Mantenibilidad debido a la importancia que tiene la posibilidad de que un método evolucione.

De dicho análisis, se definen los *Procesos de desarrollo* que se presentan en la Tabla V.9.

Tabla V.9. Definición de los procesos de desarrollo

Contexto de desarrollo
Ciclo de vida - Fases cubiertas - Definición de tareas en cada fase - Entregables
Plan de administración - Plan de proyecto - Generación de especificación - Validación de diseño - Generación de código - Generación de documentación
Mantenibilidad

Luego de analizar la Tabla V.8, se obtienen las siguientes consideraciones:

- Las fases de Investigación preliminar, Requerimientos y Levantamiento de la información se integraran a los aspectos cubiertos por la actividad estructural *Comunicación*.
- Los aspectos cubiertos por la fase Definición referidos al proyecto software pasaran a formar parte de la actividad estructural *Planeación.*
- Las fases de Análisis y Diseño serán absorbidas por la actividad estructural *Modelado*.

- Las fases Implementación y/o Programación, Prueba e Instalación serán absorbidas por la actividad estructural *Construcción*, debido a que son tares involucradas en la creación e implementación de un sistema.
- No se tendrá en cuanta a la Documentación como una fase individual debido a que cada fase del ciclo de vida resultante deberá entregar documentación de las tareas realizadas en las mismas.
- Se toma como válida la actividad estructural *Despliegue*, junto a las tareas que la misma implica, debido a que el sistema desarrollado se despliega en varios sitios.
- Se renombra como *Evolución* a la fase Mantenimiento, debido a que el software desarrollado e instalado, se mejora y optimiza para alcanzar nuevos requerimientos emergentes.

Se definen para esta investigación, seis ***actividades estructurales***, las cuales se muestran en la Tabla V.10.

Tabla V.10. Definición de actividades estructurales

Comunicación	Implica una intensa colaboración y comunicación con los clientes. También una investigación de requerimientos y actividades relacionadas a los mismos como alcance y restricciones del proyecto.
Planeación	Describe tareas criticas, riesgos probables, recursos requeridos, los productos a obtener y un programa de trabajo.
Modelado	Crea modelos que permiten al desarrollador y al cliente entender mejor los requerimientos y el diseño que lograra satisfacerlos.
Construcción	Genera código (manual o automático) y realiza las pruebas necesarias para descubrir errores en el código.
Despliegue	El software, completo o como incremento parcial, se entrega al cliente, quien lo evalúa y proporciona información basada en su evaluación.
Evolución	Es el proceso de control, mejora y optimización del software ya desarrollado e instalado.

Pero, presentar y definir las actividades de desarrollo no es suficiente para el uso de una metodología. Una metodología más elaborada presenta en detalle sus acciones dentro del ciclo de vida de desarrollo, con el fin de proporcionar al usuario los medios para un uso correcto y eficiente de la misma.

Luego, se definen las siguientes *tareas* para cada una de las *actividades estructurales* especificadas anteriormente (Tablas V.11, V.12, V.13, V.14, V.15, V.16 y V.17):

Tabla V.11. Tareas para la actividad estructural: Comunicación

COMUNICACIÓN
Tareas del trabajo: • Establecer comunicación con el cliente. • Definir un equipo de trabajo. • Hacer contacto con el cliente y demás personas involucradas en el proyecto. • Definición de alcances. • Recabar requerimientos. • Descripción de restricciones.
Productos del trabajo: ➢ Especificación en forma de texto libre y breve de los requerimientos.

Tabla V.12. Tareas para la actividad estructural: Planeación

PLANEACIÓN
Tareas del trabajo: • Comprender el proyecto para definir el problema. • Definir el alcance, los objetivos y las restricciones del sistema. • Identificar los beneficios del sistema propuesto a través de un estudio de factibilidad. • Especificar un estimado de tiempo y costo para las próximas fases de desarrollo.
Productos del trabajo: ➢ Informe al cliente describiendo el problema y detallando si se recomienda continuar con el desarrollo. ➢ Plan de proyecto.

Tabla V.13. Tareas para la actividad estructural: Modelado

MODELADO
Tareas del trabajo: • Definir las necesidades del usuario y las condiciones informáticas bajo las que debe funcionar la aplicación. • Capturar requisitos funcionales a través de casos de uso. • Agrupar las distintas funcionalidades en roles y servicios asociados a un agente o grupo de agentes. • Diseñar los roles que definen servicios (responsabilidades y funcionalidad esperada). • Diseñar las tareas que definen los procedimientos para alcanzar los objetivos (requisitos) • Diseñar la arquitectura (estructura y normas de organización). • Diseñar las relaciones entre tareas, roles y recursos. • Realizar revisiones técnicas a los modelos de análisis y diseño obtenidos.
Productos del trabajo: ➢ Documento de especificación de requerimientos. ➢ Prototipo del manual de usuario. ➢ Estructuración de la aplicación según la arquitectura diseñada. ➢ Documento de especificación de diseño. ➢ Documentación y datos de las revisiones técnicas. ➢ Estimaciones económicas finales.

Tabla V.14. Tareas para la actividad estructural: Construcción

CONSTRUCCIÓN
Tareas del trabajo: • Generar el código de la aplicación de forma manual o automática. • Validar el comportamiento del SMA a través de la simulación. • Realizar pruebas de clases, de integración y de validación del sistema. • Realizar e implementar el sistema ejecutable.
Productos del trabajo: ➢ Código ejecutable del sistema. ➢ Documentación en medios magnéticos del sistema. ➢ Datos y documentación de las pruebas.

Tabla V.15. Tareas para la actividad estructural: Despliegue

DESPLIEGUE
Tareas del trabajo: • Preparación del entorno operativo. • Instalación del sistema. • Realización de pruebas y ajustes.
Productos del trabajo: ➢ Documentación de pruebas realizadas. ➢ Resultados de la evaluación realizada por el cliente.

Tabla V.16. Tareas para la actividad estructural: Evolución

EVOLUCIÓN
Tareas del trabajo: • Depuración del sistema, mediante corrección de errores o defectos. • Revisión y nuevas versiones para mejorar el sistema. • Extender el sistema para adaptarlo a cambios del entorno.
Productos del trabajo: ➢ Incidentes y nuevos requerimientos. ➢ Documentación de la especificación inicial y las modificaciones solicitadas y realizadas.

V.5. REVISION DE LOS ASPECTOS PRAGMATICOS ESTABLECIDOS EN EL MARCO DE STURM & SHEHORY

La *pragmática* se refiere a los aspectos prácticos del uso y de implementación de una metodología [1].

Recordando los aspectos pragmáticos brindados por Sturm & Shehory [1] y los criterios de Akbari [6], mas cuestiones observables en la práctica diaria del desarrollo de software, se procede a enunciar las siguientes consideraciones:

- ✓ La importancia de los recursos existentes de la metodología a través de herramientas CASE. Por ejemplo, si es gratis o paga la disponibilidad de dichas herramientas, la cantidad de etapas de desarrollo que cubren las herramientas, cantidad de ejemplos del uso de la metodología para desarrollar SMA, cuan completos son y el éxito de los mismos.
- ✓ Los recursos necesarios para el desarrollo de los agentes propiamente dichos.
- ✓ El conocimiento del dominio de aplicación y de la escabilidad del proyecto.

Para el uso de esta investigación, se define el siguiente conjunto de aspectos pragmáticos a examinar y se nuestra en la Tabla V.17.

Tabla V.17. Definición de los aspectos pragmáticos

Aspectos pragmáticos	
Recursos existentes	Herramientas
	Información
Recursos necesarios	Adaptabilidad
	Plataforma idónea
Adaptabilidad al proyecto	Dominio de aplicación
	Escalabilidad

V.6. CONCLUSIÓN

En este capítulo, se realizó una revisión al marco para la evaluación de metodologías orientadas a agentes desarrollado por Sturm & Shehory.

Luego, se presentaron diversas tablas comparativas de conceptos, características, aspectos, etc., recopilados por la investigación, para obtener un marco extendido dividido en cuatro sesiones: *conceptos y propiedades*, *notaciones y las técnicas de modelado*, *procesos de desarrollo* y *aspectos pragmáticos*.

CAPÍTULO VI
OPERACIONALIZACIÓN DEL MARCO DE EVALUACIÓN EXTENDIDO

VI.1. INTRODUCCIÓN

El marco de evaluación resultante de la revisión, agregaciones y/o modificaciones realizadas al marco de Sturm & Shehory [1], para propósito de esta investigación, pasa a denominarse *marco extendido*.

Como pudo observarse (Véase Cap. V), los aspectos cubiertos por el marco extendido tienen un carácter cualitativo, lo que impide que sean medidos de manera directa, por lo cual deberán ser sometidos a un proceso de operacionalización. Dicho proceso, permite descomponer cada aspecto en varios niveles hasta obtener indicadores observables y concretos, susceptibles de medición.

Para determinar estos indicadores, se utilizará el *Enfoque orientado a metas* **GQM** (**Goals- Questions-Metrics**) tomado de Basili [92].

VI.2. ENFOQUE GQM

Basili describió el proceso GQM [92] en seis pasos. Los primeros tres son llamados, a menudo, *fase de definición de GQM* (Véase Figura VI.1) ya que proveen la estructura para pasar del concepto a métricas significativas que cuantifican los objetivos y proveen datos significativos para la toma de decisión.

Las **metas u objetivos (Goals)** identifican lo que queremos lograr; **las preguntas (Questions),** nos dicen si se están satisfaciendo los objetivos o ayudan a comprender cómo interpretarlos; y **las métricas (Metrics)** identifican las mediciones que son necesarias para responder a las preguntas y cuantificar el objetivo.

Los restantes pasos son para recolectar y usar los resultados de las medidas para mejorar la toma de decisiones.

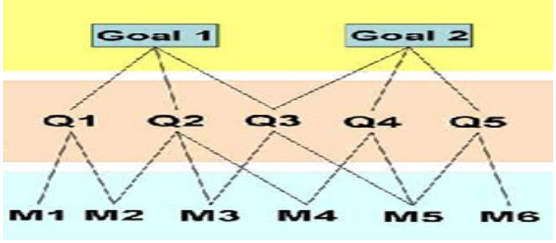

Figura VI.1. Fase de definición de GQM (extraída de [92])

Los objetivos (Goals, en el tope del árbol GQM) se expresan usando cinco facetas de información para definir lo que la medida debe lograr en términos precisos. Cada declaración de objetivo debe contener explícitamente:

- *Objeto*: El producto o el proceso bajo estudio.
- *Propósito*: Motivación detrás del objetivo (Por qué?)
- *Enfoque*: Los atributos de calidad del objeto bajo estudio
- *Punto de Vista*: Perspectiva de las metas (el punto de vista de quien?)
- *Entorno*: Alcance o contexto del programa de medidas.

La Figura VI.2 ilustra una plantilla o formato para la definición de objetivos GQM.

Analizar	El objeto bajo medición
Con el propósito de	Entender, controlar o mejorar el objeto
Con respecto a	El enfoque de calidad del objeto en que se centra la medición
Desde el punto de vista de	Las personas que miden el objeto
En el contexto de	El ambiente en el cual la medición tiene lugar

Figura VI.2. Plantilla de objetivos GQM

La generación de preguntas (Q1, Q2, …, Qn) ayuda a identificar las diversas interpretaciones del objetivo que pueden existir entre los stakeholders, así como también las restricciones impuestas por el entorno.

La especificación de las métricas (M1, M2, …, Mn) es la revisión de cómo deben ser respondidas las preguntas de forma que provean toda la información cuantitativa para responder las mismas de manera satisfactoria.

VI.3. PROCESO DE OPERACIONALIZACIÓN UTILIZANDO GQM

Para el proceso de operacionalización del marco extendido mediante el Enfoque GQM, se utilizará la plantilla definida en la tesis de magister de Olsina [88], la cual registra objetivos, preguntas, métricas y comentarios (en caso de ser necesarios), de forma más resumida.

También, se define la siguiente notación para los conceptos establecidos en dicha plantilla:

Concepto	*Notación*
OBJETIVO	Obj i
PREGUNTA	P i.j
METRICA	Me j.k

Donde i, j, y k son índices para enumerar los conceptos definidos.

VI.3.1. DEFINICIÓN DE LOS OBJETIVOS DE MEDIDA PARA EVALUAR EL MARCO EXTENDIDO

Las plantillas de las Tablas VI.1, VI.2, VI.3, VI.4 y VI.5, muestran la definición de los objetivos de medición necesarios para evaluar el grado en que una MOA cubre los conceptos y propiedades propios de la tecnología de agentes, las notaciones y las técnicas de modelado, el proceso de desarrollo y los aspectos pragmáticos.

VI.4. CONCLUSIÓN

En este capítulo, se realizó el proceso de operacionalización de los criterios definidos en el marco extendido resultante. Dicho proceso cuenta con la definición de cinco objetivos de medición para evaluar el grado en que una MOA cubre los conceptos y propiedades propios de la tecnología de agentes, las notaciones y las técnicas de modelado, el proceso de desarrollo y los aspectos pragmáticos.

Tabla VI.1. Objetivo de medida para los conceptos

<table>
<tr><td colspan="3">OBJETIVO 1</td></tr>
<tr><td colspan="2">Objeto</td><td>MOA (recurso)</td></tr>
<tr><td colspan="2">Propósito</td><td>Evaluar</td></tr>
<tr><td colspan="2">Característica o atributo</td><td>Conceptos propios de la tecnología de agentes</td></tr>
<tr><td colspan="2">Agente asignado a un rol</td><td>Desarrolladores de agentes</td></tr>
<tr><td colspan="2">Contexto</td><td>Desarrollo de agentes</td></tr>
<tr><td colspan="3">PREGUNTA/S</td></tr>
<tr><td>P1.1</td><td colspan="2">Permite representar las tareas a realizar por el agente?</td></tr>
<tr><td>P1.2</td><td colspan="2">Permite describir las acciones que llevara a cabo el agente con el fin de realizar las tareas?</td></tr>
<tr><td>P1.3</td><td colspan="2">Permite describir los servicios que proporcionara el agente?</td></tr>
<tr><td>P1.4</td><td colspan="2">Permite captar datos del entorno del agente?</td></tr>
<tr><td>P1.5</td><td colspan="2">Permite representar el paso de mensajes entre agentes?</td></tr>
<tr><td>P1.6</td><td colspan="2">Brinda un protocolo para la comunicación entre agentes?</td></tr>
<tr><td>P1.7</td><td colspan="2">Permite representar de manera abstracta la función del agente’</td></tr>
<tr><td>P1.8</td><td colspan="2">Permite describir las normas de actuación que deben seguir los agentes?</td></tr>
<tr><td>P1.9</td><td colspan="2">Permite agrupar los agentes con un fin común?</td></tr>
<tr><td>P1.10</td><td colspan="2">Permite representar la colaboración entre agentes y grupos de agentes?</td></tr>
<tr><td colspan="3">MÉTRICA/S</td></tr>
<tr><td>Me1.1</td><td colspan="2">Se propone la siguiente escala para valorar la respuestas obtenidas:
<table>
<tr><th>RESPUESTA</th><th>VALORACIÓN</th></tr>
<tr><td>La MOA no hace referencia al concepto.</td><td>0</td></tr>
<tr><td>La MOA hace referencia al concepto de forma limitada, es decir, no aborda muchas cuestiones del mismo.</td><td>1</td></tr>
<tr><td>La MOA se refiere al concepto con deficiencias menores, es decir, cubre la mayoría de las cuestiones que describe el mismo.</td><td>2</td></tr>
<tr><td>La MOA contempla plenamente el concepto.</td><td>3</td></tr>
</table></td></tr>
<tr><td>Me1.2</td><td colspan="2">Evaluación cualitativa de la MOA en cuanto a la cobertura general de los conceptos de la tecnología de agentes definidos en el marco extendido.
<table>
<tr><th>Nula</th><th>Suficiente</th><th>Parcial</th><th>Total</th><th>Observaciones / sugerencias</th></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
</table></td></tr>
<tr><td colspan="3">COMENTARIOS</td></tr>
<tr><td>Me1.1</td><td colspan="2">La escala definida se utilizará para valorar las respuestas a todas las preguntas planteadas en este objetivo.</td></tr>
<tr><td>Me1.2</td><td colspan="2">Se marca con una X la respuesta seleccionada.
Se toma al campo observaciones/sugerencias como obligatorio.</td></tr>
</table>

Tabla VI.2. Objetivo de medida para las propiedades

<table>
<tr><td colspan="3">OBJETIVO 2</td></tr>
<tr><td colspan="2">Objeto</td><td>MOA (recurso)</td></tr>
<tr><td colspan="2">Propósito u objetivo</td><td>Evaluar</td></tr>
<tr><td colspan="2">Característica o atributo</td><td>Propiedades de los agentes</td></tr>
<tr><td colspan="2">Agente asignado a un rol</td><td>Desarrolladores de agentes</td></tr>
<tr><td colspan="2">Contexto</td><td>Desarrollo de agentes</td></tr>
<tr><td colspan="3">PREGUNTA/S</td></tr>
<tr><td>P2.1</td><td colspan="2">¿Cuál es el grado en que la MOA permite representar la autonomía de un agente?</td></tr>
<tr><td>P2.2</td><td colspan="2">¿La MOA permite desarrollar agentes reactivos?</td></tr>
<tr><td>P2.3</td><td colspan="2">¿Permite representar los objetivos y lo planes de un agente?</td></tr>
<tr><td>P2.4</td><td colspan="2">¿Trata la iniciativa para la persecución de objetivos de forma autónoma?</td></tr>
<tr><td>P2.5</td><td colspan="2">¿Contempla la racionalidad de un agente?</td></tr>
<tr><td>P2.6</td><td colspan="2">¿Trata las relaciones que emergen de los agentes entre sí y con el entorno?</td></tr>
<tr><td>P2.7</td><td colspan="2">¿Contempla que el funcionamiento del agente cambie en función de su experiencia previa?</td></tr>
<tr><td>P2.8</td><td colspan="2">¿Trata la movilidad de un agente de una maquina a otra?</td></tr>
<tr><td>P2.9</td><td colspan="2">Permite representar la sociabilidad (cooperación, interacción, negociación y organización) entre agentes, con el entorno y con sus usuarios?</td></tr>
<tr><td colspan="3">MÉTRICA/S</td></tr>
<tr><td>Me2.1</td><td colspan="2">Se propone la siguiente escala para valorar la respuestas obtenidas:
<table>
<tr><th>RESPUESTA</th><th>VALORACIÓN</th></tr>
<tr><td>La MOA no hace referencia a la propiedad.</td><td>0</td></tr>
<tr><td>La MOA hace referencia a la propiedad, pero no proporciona detalles de la misma.</td><td>1</td></tr>
<tr><td>La MOA hace referencia a la propiedad de forma limitada, es decir, no aborda muchas cuestiones de la misma.</td><td>2</td></tr>
<tr><td>La MOA se refiere a la propiedad con deficiencias menores, es decir, cubre la mayoría de las cuestiones que describe la misma.</td><td>3</td></tr>
<tr><td>La MOA contempla plenamente la propiedad.</td><td>4</td></tr>
</table></td></tr>
<tr><td>Me2.2</td><td colspan="2">Evaluación cualitativa de la MOA en cuanto a la cobertura general de las propiedades de los agentes definidas en el marco extendido.
<table>
<tr><th>Nula</th><th>Suficiente</th><th>Parcial</th><th>Total</th><th>Observaciones / sugerencias</th></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
</table></td></tr>
<tr><td colspan="3">COMENTARIOS</td></tr>
<tr><td>Me2.1</td><td colspan="2">La escala definida se utilizará para valorar las respuestas a todas las preguntas planteadas en este objetivo.</td></tr>
<tr><td>Me2.2</td><td colspan="2">Se marca con una X la respuesta seleccionada.
Se toma al campo observaciones/sugerencias como obligatorio.</td></tr>
</table>

Tabla VI.3. Objetivo de medida para las notaciones y técnicas de modelado

<table>
<tr><td colspan="3">OBJETIVO 3</td></tr>
<tr><td colspan="2">Objeto</td><td>MOA (recurso)</td></tr>
<tr><td colspan="2">Propósito u objetivo</td><td>Evaluar</td></tr>
<tr><td colspan="2">Característica o atributo</td><td>Propiedades de las notaciones y técnicas de modelado</td></tr>
<tr><td colspan="2">Agente asignado a un rol</td><td>Desarrolladores de agentes</td></tr>
<tr><td colspan="2">Contexto</td><td>Desarrollo de agentes</td></tr>
<tr><td colspan="3">PREGUNTA/S</td></tr>
<tr><td>P3.1</td><td colspan="2">Permite una fácil comprensión del método a utilizar?</td></tr>
<tr><td>P3.2</td><td colspan="2">Permite una facilidad de uso del método?</td></tr>
<tr><td>P3.3</td><td colspan="2">Brinda la capacidad de representar los conceptos del agente?</td></tr>
<tr><td>P3.4</td><td colspan="2">Permite describir las propiedades del agente?</td></tr>
<tr><td>P3.5</td><td colspan="2">Permite especificar el agente de forma iterativa e incremental?</td></tr>
<tr><td>P3.6</td><td colspan="2">Permite comprobar la consistencia o implicaciones de los modelos?</td></tr>
<tr><td>P3.7</td><td colspan="2">Permite interpretar los modelos existentes de manera simbólica?</td></tr>
<tr><td>P3.8</td><td colspan="2">Permite interpretar los modelos existentes de manera sintáctica?</td></tr>
<tr><td>P3.9</td><td colspan="2">Permite interpretar los modelos existentes de manera semántica?</td></tr>
<tr><td>P3.10</td><td colspan="2">Permite la reutilización de los modelos?</td></tr>
<tr><td>P3.11</td><td colspan="2">Permite ampliar los conceptos básicos?</td></tr>
<tr><td>P3.12</td><td colspan="2">Permite ampliar los bloques de construcción?</td></tr>
<tr><td colspan="3">MÉTRICA/S</td></tr>
<tr><td>Me3.1</td><td colspan="2">Se propone la siguiente escala para valorar la respuestas obtenidas:
<table>
<tr><th>RESPUESTA</th><th>VALORACIÓN</th></tr>
<tr><td>La MOA no hace referencia a la propiedad.</td><td>0</td></tr>
<tr><td>La MOA hace referencia a la propiedad, pero no proporciona detalles de la misma.</td><td>1</td></tr>
<tr><td>La MOA hace referencia a la propiedad de forma limitada, no aborda muchas cuestiones de la misma.</td><td>2</td></tr>
<tr><td>La MOA se refiere a la propiedad con deficiencias menores, cubre la mayoría de las cuestiones que describe la misma.</td><td>3</td></tr>
<tr><td>La MOA contempla plenamente la propiedad.</td><td>4</td></tr>
</table></td></tr>
<tr><td>Me3.2</td><td colspan="2">Evaluación cualitativa de la MOA en cuanto a la cobertura general de propiedades de las notaciones y técnicas de modelado definidas en el marco extendido.
<table>
<tr><th>Nula</th><th>Suficiente</th><th>Parcial</th><th>Total</th><th>Observaciones / sugerencias</th></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
</table></td></tr>
<tr><td colspan="3">COMENTARIOS</td></tr>
<tr><td>Me3.1</td><td colspan="2">La escala definida se utilizará para valorar las respuestas a todas las preguntas planteadas en este objetivo.</td></tr>
<tr><td>Me3.2</td><td colspan="2">Se marca con una X la respuesta seleccionada. Se toma al campo observaciones/sugerencias como obligatorio.</td></tr>
</table>

Tabla VI.4. Objetivo de medida para el proceso de desarrollo

<table>
<tr><td colspan="3">OBJETIVO 4</td></tr>
<tr><td colspan="2">Objeto</td><td>MOA (recurso)</td></tr>
<tr><td colspan="2">Propósito u objetivo</td><td>Evaluar</td></tr>
<tr><td colspan="2">Característica o atributo</td><td>El proceso de desarrollo</td></tr>
<tr><td colspan="2">Agente asignado a un rol</td><td>Desarrolladores de agentes</td></tr>
<tr><td colspan="2">Contexto</td><td>Desarrollo de agentes</td></tr>
<tr><td colspan="3">PREGUNTA/S</td></tr>
<tr><td>P4.1</td><td colspan="2">La MOA se adapta a la creación a nuevo software?</td></tr>
<tr><td>P4.2</td><td colspan="2">Se adapta al prototipado y reutilización de componentes?</td></tr>
<tr><td>P4.3</td><td colspan="2">Representa las fases del ciclo de vida?</td></tr>
<tr><td>P4.4</td><td colspan="2">Permite definir las tareas propias de cada fase?</td></tr>
<tr><td>P4.5</td><td colspan="2">Permite la obtención de entregables?</td></tr>
<tr><td>P4.6</td><td colspan="2">Permite definir el plan del proyecto?</td></tr>
<tr><td>P4.7</td><td colspan="2">Genera especificaciones?</td></tr>
<tr><td>P4.8</td><td colspan="2">Valida el diseño?</td></tr>
<tr><td>P4.9</td><td colspan="2">Genera código?</td></tr>
<tr><td>P4.10</td><td colspan="2">Genera documentación?</td></tr>
<tr><td>P4.11</td><td colspan="2">Permite la evolución del método?</td></tr>
<tr><td colspan="3">MÉTRICA/S</td></tr>
<tr><td>Me4.1</td><td colspan="2">Se propone la siguiente escala para valorar la respuestas obtenidas:
<table>
<tr><th>RESPUESTA</th><th>VALORACIÓN</th></tr>
<tr><td>La MOA no hace referencia al aspecto.</td><td>0</td></tr>
<tr><td>La MOA hace referencia al aspecto de forma limitada, es decir, no aborda muchas cuestiones del mismo.</td><td>1</td></tr>
<tr><td>La MOA se refiere al aspecto con deficiencias menores, es decir, cubre la mayoría de las cuestiones que describe el mismo.</td><td>2</td></tr>
<tr><td>La MOA contempla plenamente el aspecto.</td><td>3</td></tr>
</table></td></tr>
<tr><td>Me4.2</td><td colspan="2">Evaluación cualitativa de la MOA en cuanto a la cobertura del proceso de desarrollo definido en el marco extendido.
<table>
<tr><th>Nula</th><th>Suficiente</th><th>Parcial</th><th>Total</th><th>Observaciones / sugerencias</th></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
</table></td></tr>
<tr><td colspan="3">COMENTARIOS</td></tr>
<tr><td>Me4.1</td><td colspan="2">La escala definida se utilizará para valorar las respuestas a todas las preguntas planteadas en este objetivo.</td></tr>
<tr><td>Me4.2</td><td colspan="2">Se marca con una X la respuesta seleccionada.
Se toma al campo observaciones/sugerencias como obligatorio.</td></tr>
</table>

Tabla VI.5. Objetivo de medida para los aspectos pragmáticos

<table>
<tr><td colspan="3">OBJETIVO 5</td></tr>
<tr><td colspan="2">Objeto</td><td>MOA (recurso)</td></tr>
<tr><td colspan="2">Propósito u objetivo</td><td>Evaluar</td></tr>
<tr><td colspan="2">Característica o atributo</td><td>Los aspectos pragmáticos</td></tr>
<tr><td colspan="2">Agente asignado a un rol</td><td>Desarrolladores de agentes</td></tr>
<tr><td colspan="2">Contexto</td><td>Desarrollo de agentes</td></tr>
<tr><td colspan="3">PREGUNTA/S</td></tr>
<tr><td>P5.1</td><td colspan="2">Brinda herramientas de desarrollo gratuitas?</td></tr>
<tr><td>P5.2</td><td colspan="2">Brinda herramientas de desarrollo pagas?</td></tr>
<tr><td>P5.3</td><td colspan="2">Cuenta con información actualizada y en línea?</td></tr>
<tr><td>P5.4</td><td colspan="2">Brinda grupos de apoyo?</td></tr>
<tr><td>P5.5</td><td colspan="2">Los recursos que brinda se adaptan al proyecto?</td></tr>
<tr><td>P5.6</td><td colspan="2">El entregable se adapta a la plataforma idónea?</td></tr>
<tr><td>P5.7</td><td colspan="2">Permite manejar el crecimiento continuo?</td></tr>
<tr><td colspan="3">METRICA/S</td></tr>
<tr><td>Me5.1</td><td colspan="2">Se propone la siguiente escala para valorar la respuestas obtenidas:
<table>
<tr><th>RESPUESTA</th><th>VALORACIÓN</th></tr>
<tr><td>La MOA no hace referencia al aspecto.</td><td>0</td></tr>
<tr><td>La MOA hace referencia al aspecto de forma limitada, es decir, no aborda muchas cuestiones del mismo.</td><td>1</td></tr>
<tr><td>La MOA se refiere al aspecto con deficiencias menores, es decir, cubre la mayoría de las cuestiones que describe el mismo.</td><td>2</td></tr>
<tr><td>La MOA contempla plenamente el aspecto.</td><td>3</td></tr>
</table></td></tr>
<tr><td>Me5.2</td><td colspan="2">Evaluación cualitativa de la MOA en cuanto a la cobertura general de los aspectos pragmáticos definidos en el marco extendido.
<table>
<tr><th>Nula</th><th>Suficiente</th><th>Parcial</th><th>Total</th><th>Observaciones / sugerencias</th></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
</table></td></tr>
<tr><td colspan="3">COMENTARIOS</td></tr>
<tr><td>Me5.1</td><td colspan="2">La escala definida se utilizará para valorar las respuestas a todas las preguntas planteadas en este objetivo.</td></tr>
<tr><td>Me5.2</td><td colspan="2">Se marca con una X la respuesta seleccionada. Se toma al campo observaciones/sugerencias como obligatorio.</td></tr>
</table>

CAPÍTULO VII
EVALUACIÓN DEL MARCO EXTENDIDO Y ANÁLISIS DE RESULTADOS

VII.1. INTRODUCCIÓN

La gestión de procesos de software identifica cuatro responsabilidades claves: definir, medir, controlar y mejorar el proceso [94].

Sin embargo, sobre la responsabilidad de medir el proceso se puede afirmar que, en general, las mediciones se realizan sobre los productos y la existencia de mediciones sobre los procesos son escasas. Por ello, es importante recordar que sólo midiendo es posible conocer el estado de un proceso de manera objetiva, y sólo gracias a esto se pueden planificar estrategias y soluciones acerca de las mejoras a realizar, siguiendo los objetivos de la organización [95].

VII.2. EVALUACIÓN DEL MARCO EXTENDIDO

Para realizar la evaluación del marco extendido se tendrán en cuenta los objetivos de mediciones definidos en el capitulo anterior (Véase Cap. VI.3.1), sus preguntas y métricas, y se definirán actividades para obtener indicadores y así, poder medir el grado del logro de cada objetivo.

VII.2.1. ACTIVIDAD 1: IDENTIFICACIÓN DE LOS OBJETIVOS DE MEDIDA

A continuación, se listan los objetivos de medida, que se buscan alcanzar a través de la evaluación del marco extendido.

- **Obj.1.** Analizar la MOA con el propósito de evaluar el alcance de la misma, respecto a los conceptos propios de la tecnología de agentes, desde el punto de vista de los desarrolladores en el contexto del desarrollo de agentes.
- **Obj.2.** Analizar la MOA con el propósito de evaluar el alcance de la misma, respecto a las propiedades de los agentes, desde el punto de vista de los desarrolladores en el contexto del desarrollo de agentes.
- **Obj.3.** Analizar la MOA con el propósito de evaluar el alcance de la misma, respecto a las propiedades de las notaciones y técnicas de modelado, desde el punto de vista de los desarrolladores en el contexto del desarrollo de agentes.

- **Obj.4.** Analizar la MOA con el propósito de evaluar el alcance de la misma, respecto al proceso de desarrollo, desde el punto de vista de los desarrolladores en el contexto del desarrollo de agentes.
- **Obj.5.** Analizar la MOA con el propósito de evaluar el alcance de la misma, respecto a los aspectos pragmáticos, desde el punto de vista de los desarrolladores en el contexto del desarrollo de agentes.

VII.2.2.ACTIVIDAD 2: CREACIÓN DE LOS FORMULARIOS DE RECOLECCIÓN DE INFORMACIÓN

Debido a que cada objetivo de medida está descompuesto en una serie de preguntas (Véase Cap. VI.3.1), con el fin de obtener valores que permitan medir el grado del logro de cada uno de ellos, para esta investigación se define un formulario para recolectar esos valores y se muestra en el ***Anexo A***.

Cada formulario será distribuido vía mail a un grupo de egresados de la carrera de Lic. en Sistemas de Información que han utilizado alguna de las MOA para el desarrollo de sus trabajos de graduación.

VII.2.3. ACTIVIDAD 3: OBTENCIÓN DE INDICADORES

El valor del grado de satisfacción de una pregunta es su valor numérico asociado. Por ejemplo, la pregunta P1.1 puede tomar valores en un rango discreto de 0 (cero) a 3 (tres). (Véase Cap. VI.3.1).

El valor del grado del logro de un objetivo de medida se obtiene en base a los valores obtenidos de las preguntas, a través de la siguiente ecuación:

$$ValorObj_i = \frac{\sum_j^m \boldsymbol{P}ij}{m}$$

Donde ***i*** va de 1 (uno) a 5 (cinco), por ser cinco los objetivos y ***j*** va de 1 (uno) a ***m*** (eme) según la cantidad de preguntas definidas para el objetivo ***i***.

Si se aplica el formulario a varias personas, el valor del logro del objetivo debe ponderarse utilizando la siguiente ecuación:

$$ValTotObj_i = \frac{\sum_1^n ValorObj}{n}$$

Donde ***i*** corresponde al valor obtenido para cada formulario y ***n*** la cantidad de personas que respondieron dicho formulario.

Luego, cada *ValTotObj* obtenido se ubicara en la escala de valores definida para cada objetivo y de esta manera se determinará el grado del logro para cada objetivo.

VII.2.4. ACTIVIDAD 4: ANÁLISIS DEL LOGRO DE CADA OBJETIVO

La Tabla VII.1 muestra los valores para el objetivo de medida 1 obtenidos a través de cuatro encuestas. Como puede observarse, los encuestados coinciden en que la MOA utilizada por todos ellos, como es INGENIAS, cubre la mayoría de las cuestiones referidas a la descripción y representación de tareas y servicios que realiza el agente, pero falla en la comunicación entre agentes al no brindar un protocolo de comunicación entre ellos.

El grado del logro del objetivo 1 (VALOR TOTAL) es 2, por lo que se concluye que la MOA cubre la mayoría de las cuestiones referidas a los conceptos propios de la tecnología de agentes.

Tabla VII.1. Valores obtenidos para el objetivo de medida 1

Obj1				
Preguntas	*Encuesta 1*	*Encuesta 2*	*Encuesta 3*	*Encuesta 4*
P1.1	3	3	3	3
P1.2	3	3	2	2
P1.3	2	3	3	3
P1.4	2	2	1	1
P1.5	1	1	1	1
P1.6	1	1	0	1
P1.7	2	3	2	2
P1.8	0	3	3	2
P1.9	1	3	2	2
P1.10	1	3	0	2
VALOR del **Obj 1**	2	3	2	2
VALOR TOTAL				**2**

La Tabla VII.2 muestra los valores para el objetivo de medida 2. Para los encuestados, la MOA utilizada (INGENIAS), permite representar agentes autónomos, reactivos, racionales y sociables. Lo que le falta es representar los objetivos y planes del agente.

Aunque la mayoría de las preguntas pertenecientes a este objetivo tiene buenos valores como 3 o 4, debido a la formula de ponderación, el grado del logro del objetivo 2 (VALOR TOTAL) es 2, por lo tanto la MOA cubre las propiedades de los agentes de forma limitada.

Tabla VII.2. Valores obtenidos para el objetivo de medida 2

Obj2				
Preguntas	*Encuesta 1*	*Encuesta 2*	*Encuesta 3*	*Encuesta 4*
P2.1	3	3	4	3
P2.2	0	3	3	3
P2.3	0	0	0	0
P2.4	0	3	3	3
P2.5	1	4	3	3
P2.6	2	3	3	3
P2.7	1	3	4	3
P2.8	0	2	2	2
P2.9	1	4	2	2
VALOR del **Obj 2**	1	3	3	2
VALOR TOTAL				**2**

La Tabla VII.3 muestra los valores para el objetivo de medida 3. Según las respuestas recibidas, la MOA utilizada (INGENIAS), es de fácil comprensión y permite representar los conceptos y propiedades de los agentes. También, permite especificar el agente de forma iterativa y comprobar la consistencia del modelo. Pero falla en la evolución y reutilización de lo diseñado.

Nuevamente, debido a la formula de ponderación, se diluyen los buenos valores y el grado del logro del objetivo 3 (VALOR TOTAL) es 2, por lo tanto la MOA cubre de forma limitada las propiedades de las notaciones y técnicas de modelado definidas en el marco extendido.

La Tabla VII.4 muestra los valores para el objetivo de medida 4. Para los encuestados, la MOA utilizada (INGENIAS), no se adapta a la creación de nuevo software, ni al prototipado y no representa todas las fases del ciclo de vida definidas en el marco extendido. Se obtienen pocos entregables, no contempla las especificaciones, y todo lo referido al código se realiza a través de una herramienta que lo genera de forma automática.

Tabla VII.3. Valores obtenidos para el objetivo de medida 3

Obj3				
Preguntas	*Encuesta 1*	*Encuesta 2*	*Encuesta 3*	*Encuesta 4*
P3.1	4	2	3	3
P3.2	4	3	2	2
P3.3	4	3	2	2
P3.4	3	3	3	3
P3.5	2	2	3	3
P3.6	1	3	3	3
P3.7	3	3	3	3
P3.8	0	3	3	3
P3.9	0	3	3	3
P3.10	0	2	0	0
P3.11	2	2	0	1
P3.12	0	2	0	1
VALOR del **Obj 3**	2	3	2	2
VALOR TOTAL				**2**

Por ello, el grado del logro del objetivo 4 (VALOR TOTAL) es 1, la MOA cubre de forma limitada el proceso de desarrollo, sin abordar muchas cuestiones del mismo.

Tabla VII.4. Valores obtenidos para el objetivo de medida 4

Obj4				
Preguntas	*Encuesta 1*	*Encuesta 2*	*Encuesta 3*	*Encuesta 4*
P4.1	0	1	1	1
P4.2	0	2	1	1
P4.3	0	2	1	1
P4.4	3	2	3	3
P4.5	0	2	1	1
P4.6	0	2	3	3
P4.7	2	1	1	2
P4.8	2	1	0	2
P4.9	3	2	0	2
P4.10	3	1	1	2
P4.11	0	1	1	1
VALOR del **Obj 4**	1	2	1	2
VALOR TOTAL				**1**

La Tabla VII.5 muestra los valores para el objetivo de medida 5. Según las respuestas recibidas,, la MOA utilizada (INGENIAS), es de acceso gratuito pero brinda muy pocos grupos de apoyo. Lo beneficioso de su uso es la adaptabilidad al proyecto y que brinda información actualizada y en línea. Lo que queda poco claro es si el entregable se adapta a la plataforma idónea o no. En este punto, las respuestas son dispares al respecto.

Luego, el grado del logro del objetivo 5 (VALOR TOTAL) es 2, por lo tanto la MOA cubre la mayoría de las cuestiones referidas a los aspectos pragmáticos.

Tabla VII.5. Valores obtenidos para el objetivo de medida 5

Obj5				
Preguntas	*Encuesta 1*	*Encuesta 2*	*Encuesta 3*	*Encuesta 4*
P5.1	3	1	1	2
P5.2	0	0	0	0
P5.3	2	3	3	2
P5.4	0	1	1	1
P5.5	3	3	3	2
P5.6	1	3	3	1
P5.7	0	3	3	2
VALOR del **Obj 5**	1	2	2	1
VALOR TOTAL				2

VII.2.5. ACTIVIDAD 5: ANÁLISIS GENERAL DE LOS RESULTADOS

Luego de analizar y comparar las encuestas recibidas, se puede concluir que la MOA utilizada por este grupo de personas, en este caso INGENIAS, brinda pocas opciones para representar las funciones de los agentes. Sería conveniente que brinde una herramienta para diseñar los agentes, por ejemplo: un compilador, de modo que la representación de estos no sea muy abstracta, dando márgenes a muchas interpretaciones de un mismo modelo; y una herramienta para generar especificaciones para así obtener gráficos y documentación de lo modelado.

Aunque incluye una herramienta para la generación automática de código (INGENIAS Development KIT). Dicho código se obtiene en lenguaje Java y muchas veces necesita ser adaptado al lenguaje PHP.

VII.3. CONCLUSIÓN

En este capítulo, se realizó la evaluación del marco extendido. Para ello, se listaron los objetivos de medidas y se definieron los formularios para recolectar los datos necesarios para medirlos. Luego se presentaron, a través de tablas, las respuestas obtenidas y se procedió a obtener los valores del grado alcanzado por cada uno de los objetivos. En base a los valores obtenidos se realizo un análisis de cuanto cubre una MOA el proceso de desarrollo de agentes software.

CONCLUSIONES FINALES

Con respecto a los objetivos planteados al inicio del trabajo se puede decir que se alcanzaron considerablemente. La revisión de las MOA existentes permitió conocer orígenes, influencias y particularidades de cada una. La revisión y análisis del marco de evaluación propuesto por Sturm & Shehory, base de este trabajo, sumado a las agregaciones de otros trabajos relacionados con criterios de evaluación de MOA y la bibliografía utilizada, permitió obtener una marco extendido para evaluar las MOA a través de la definición de objetivos de medidas y sus respectivas preguntas y métricas. el marco se aplico al estudio de las MOA, a través de la realización de una encuesta para recolectar los datos necesarios para medir dichos objetivos y de esta forma se evaluó los aspectos referidos al desarrollo de agentes cubiertos por la MOA utilizada. Puntualmente, se analizo la metodología INGENIAS debido a que fue la utilizada por todos los encuestados para desarrollar módulos o SMA completos.

El aporte del marco extendido es la presentación de conceptos, propiedades y aspectos propios de la tecnología de agentes de una forma unificada buscando estandarizar las características que una MOA debería poseer para cubrir fehacientemente el proceso de desarrollo. Por supuesto, que al ser un paradigma que evoluciona constantemente, dicho marco necesita ser revisado de forma periódica para no perder su propósito

Se requiere que las MOA asistan a todas las fases del ciclo de vida del agente software. Sin técnicas adecuadas para soportar este proceso, tales sistemas probablemente no serán lo suficientemente confiables y mantenibles; serán difíciles de comprender y sus elementos más difíciles de reusar.

Línea futura de investigación: con respecto al marco extendido, podrían tomarse las fases del ciclo de vida como objetivos de medidas y de esta manera, definir preguntas y métricas para evaluar el grado en una determinada MOA cubre las tareas y productos propios de cada fase.

BIBLIOGRAFÍA

[1] Sturm, A., & Shehory, O. (2004, January). A framework for evaluating agent-oriented methodologies. In *Agent-Oriented Information Systems* (pp. 94-109). Springer Berlin Heidelberg.

[2] Bergenti, F., Gleizes, M. P., & Zambonelli, F. (Eds.). (2006). *Methodologies and software engineering for agent systems: the agent-oriented software engineering handbook* (Vol. 11). Springer Science & Business Media.

[3] Graham, I., Henderson-Sellers, B., & Younessi, H. (1997). *The OPEN process specification*. ACM Press/Addison-Wesley Publishing Co..

[4] Shehory, O., & Sturm, A. (2001, May). Evaluation of modeling techniques for agent-based systems. In *Proceedings of the fifth international conference on Autonomous agents* (pp. 624-631). ACM.

[5] Gómez Álvarez, J. M. (2005). Programa de Doctorado "Retos científicos de la computación". *Sistemas de Información Multiagente*.

[6] Akbari, O. Z. (2010). A survey of agent-oriented software engineering paradigm: Towards its industrial acceptance. *Journal of Computer Engineering Research*,*1*(2), 14-28.

[7] O'Malley, S. A., & DeLoach, S. A. (2002). *Determining when to use an agent-oriented software engineering paradigm* (pp. 188-205). Springer Berlin Heidelberg.

[8] Palliotto, Diana (2012-2015). Proyecto de investigación**:** MÉTODOS, TÉCNICAS Y HERRAMIENTAS PARA LA INGENIERÍA DE SOFTWARE ORIENTADA A AGENTES. CICyT-UNSE.

[9] Parunak, H. V. D. (1998). What can agents do in industry, and why? An overview of industrially-oriented R&D at CEC. In *Cooperative Information Agents II Learning, Mobility and Electronic Commerce for Information Discovery on the Internet* (pp. 1-18). Springer Berlin Heidelberg.

[10] American Association for Artificial Intelligence
http://www.aaai.org
http://www.aaai.org/AITopics/index.html

[11] Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 433-460.

[12] Iglesias, C. (1998). Definición de una metodología para el desarrollo de sistemas multiagente. *Tesis doctoral*. Departamento de Ingeniería Telemática. Universidad Politécnica de Madrid. España.

[13] Nwana, H. S. (1996). Software agents: An overview. *The knowledge engineering review*, *11*(03), 205-244.

[14] Moulin, B., & Brassard, M. (1996). A scenario-based design method and an environment for the development of multiagent systems. In *Distributed Artificial Intelligence Architecture and Modelling* (pp. 216-232). Springer Berlin Heidelberg.

[15] Chaib-Draa, B., Moulin, B., Mandiau, R., & Millot, P. (1992). Trends in distributed artificial intelligence. *Artificial Intelligence Review*, *6*(1), 35-66.

[16] Wooldridge, M., & Jennings, N. R. (1995). Agent theories, architectures, and languages: a survey. In *Intelligent agents* (pp. 1-39). Springer Berlin Heidelberg.

[17] Hewitt, C. (1986). Offices are open systems. *ACM Transactions on Information Systems (TOIS)*, *4*(3), 271-287.

[18] Hewitt, C. (1991). Open information systems semantics for distributed artificial intelligence. *Artificial intelligence*, *47*(1), 79-106.

[19] Genesereth, M., & Singh, N. P. (1994). A knowledge sharing approach to software interoperation. *Computer Science Department, Stanford University*.

[20] Singh, N., Genesereth, M., & Syed, M. (1995). A distributed and anonymous knowledge sharing approach to software interoperation. *International Journal of Cooperative Information Systems*, *4*(04), 339-367.

[21] Newell, A. (1982). The knowledge level. *Artificial intelligence*, *18*(1), 87-127.

[22] Shoham, Y. (1993). Agent-oriented programming. *Artificial intelligence*, *60*(1), 51-92.

[23] Wooldridge, M. J. (1998). *Agent technology: foundations, applications, and markets*. Springer Science & Business Media.

[24] http://gwai.ei.uvigo.es

[25] Demazeau, Y. (1996). *Vowels*. In Invited lecture, 1st Ibero-American Workshop on Distributed AI and Multi-Agent Systems (IWDAIMAS`96), Xalapa, Mexico.

[26] Caire, G., Coulier, W., Garijo, F., Gomez, J., Pavón, J., Leal, F., ... & Massonet, P. (2002). Agent oriented analysis using MESSAGE/UML. In *Agent-oriented software engineering II* (pp. 119-135). Springer Berlin Heidelberg.

[27] Franklin, S., & Graesser, A. (1997). Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In *Intelligent agents III agent theories, architectures, and languages* (pp. 21-35). Springer Berlin Heidelberg.

[28] Castelfranchi, C. (1995). Guarantees for autonomy in cognitive agent architecture. In *Intelligent agents* (pp. 56-70). Springer Berlin Heidelberg.

[29] Maes, P. (1997). Pattie Maes on software agents: humanizing the global computer. *IEEE Internet Computing*, *1*(4), 10-19.

[30] ANA MAS. (2005). *Software de Agentes y Sistemas Multiagente: Conceptos, arquitecturas y Aplicaciones*. Prentice Hall.

[31] George, M. P., & Ingrand, F. F. (1989, November). Monitoring and control of spacecraft systems using procedural reasoning. In *Proceedings of the Space Operations Automation and Robotics Workshop*.

[32] Ferguson, I. A. (1995). Integrated control and coordinated behavior: A case for agent models. In *Intelligent Agents* (pp. 203-218). Springer Berlin Heidelberg.

[33] Ferguson, I. (1995). On the role of BDI Modelling for integrated control and coordinated behaviour in autonomous agents. *Journal of Applied Artificial Intelligence*, *9*(4).

[34] Fischer, K., Müller, J. P., & Pischel, M. (1996). *Agenda: A general testbed for distributed artificial intelligence applications* (Vol. 19). John Wiley & Sons, Inc., New York.

[35] Müller, J. (1996). *An Architecture for Dynamically Interacting Agents*. PhD thesis. Germain AI Research Center (DFKI GmbH).

[36] Muller, J., Pischel, M., & Thiel, M. (1995). A pragmatic approach to modelling autonomous interacting systems: Preliminary report. *Intelligent Agents: Theories, Architectures, and Languages. Lecture Notes in Artificial Intelligence LNAI, 890*.

[37] Russell, S. J., & Norvig, P. (1996). *Inteligencia Artificial: un enfoque moderno*.

[38] Sycara, K. P. (1998). Multiagent systems. *AI magazine*, *19*(2), 79.

[39] Wooldridge, M. (2009). *An introduction to multiagent systems*. John Wiley & Sons.

[40] Ferber, J. (1999). *Multi-agent systems: an introduction to distributed artificial intelligence* (Vol. 1). Reading: Addison-Wesley.

[41] Demazeau, Y. (2004). *Multiagent Systems: Methodology, Practice and Challenges*. Curso de Temas Avanzados. DIA-UPM.

[42] Shoham, Y. (1993). Agent-oriented programming. *Artificial intelligence*, *60*(1), 51-92.

[43] De Giacomo, G., Lespérance, Y., & Levesque, H. J. (2000). ConGolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, *121*(1), 109-169.

[44] Bond, A. H., & Gasser, L. (1988). An analysis of problems and research in DAI.

[45] Durfee, E. H., Lesser, V. R., & Corkill, D. D. (1989). Trends in cooperative distributed problem solving. *Knowledge and Data Engineering, IEEE Transactions on*, *1*(1), 63-83.

[46] Müller, H. J. (1996). Negotiation principles. In G. M. P. O'Hare and J. N. R, editors, *Foundations of Distributed Artificial Intelligence*, pages 211–229. John Wiley & Sons

[47] Gomez-Sanz, J. J., Fuentes-Fernández, R., & Pavón, J. (2011). Understanding agent oriented software engineering methodologies. In *12th International Workshop on AgentOriented Software Engineering* (pp. 81-91).

[48] INTERNATIONAL STANDARD ISO/IEC/ IEEE 24765. (2010). Systems and software engineering — Vocabulary.

[49] Pressman, R. S. (2005). Ingeniería de Software, Sexta Edición, Ed.

[50] Wooldridgey, M., & Ciancarini, P. (2001, January). Agent-oriented software engineering: The state of the art. In *Agent-oriented software engineering* (pp. 1-28). Springer Berlin Heidelberg.

[51] Sommerville, I. (2002). Ingeniería de Software, 6ª. *Edición. Addison Wesley. México*.

[52] Sanz, J. G., & Mestras, J. P. (2001). Análisis y Diseño de Sistemas Multi-Agente. *Dep. Sistemas Informáticos y Programación. Universidad Complutense Madrid. España*.

[53] Abran, A., Moore, J. W., Bourque, P., Dupuis, R., & Tripp, L. (2004). Guide to the software engineering body of knowledge, 2004 version. *IEEE Computer Society*, *1*.

[54] Thayer, R. H., & Dorfman, M. (1995). *System and software requirements engineering*. IEEE computer society press.

[55] Kinny, D., Georgeff, M., & Rao, A. (1996). A methodology and modelling technique for

systems of BDI agents. In *Agents breaking away* (pp. 56-71). Springer Berlin Heidelberg.

[56] Burmeister, B. (1996). Models and methodology for agent-oriented analysis and design. *Working Notes of the KI*, *96*(96-06), 52.

[57] Moulin, B., & Cloutier, L. (1994, July). Collaborative work based on multiagent architectures: A methodological perspective. In *Soft computing* (pp. 261-296). Prentice-Hall, Inc.

[58] Moulin, B., & Brassard, M. (1996). A scenario-based design method and an environment for the development of multiagent systems. In *Distributed Artificial Intelligence Architecture and Modelling* (pp. 216-232). Springer Berlin Heidelberg.

[59] Kendall, E. A., Malkoun, M. T., & Jiang, C. H. (1996). A methodology for developing agent based systems for enterprise integration. In *Modelling and Methodologies for Enterprise Integration* (pp. 333-344). Springer US.

[60] Glaser, N. (1996). Contribution to knowledge modelling in a multi-agent framework (the CoMoMAS approach). *PhDthesis, Lniverstit Henri Poincar, Nancy I, France*.

[61] Iglesias, C. A., Garijo, M., González, J. C., & Velasco, J. R. (1996, November). A methodological proposal for multiagent systems development extending CommonKADS. In *Proceedings of the 10th Banff knowledge acquisition for knowledge-based systems workshop* (Vol. 1, pp. 25-1).

[62] Iglesias, C. A., Garijo, M., González, J. C., & Velasco, J. R. (1997). MAS-CommonKADS: A comprehensive agent-oriented methodology. In *Proceedings of the 11th International Conference On Mathematical and Computer Modelling and Scientific Computing* (Vol. 1).

[63] Iglesias, C. A., Garijo, M., González, J. C., & Velasco, J. R. (1998). Analysis and design of multiagent systems using MAS-CommonKADS. In *Intelligent Agents IV Agent Theories, Architectures, and Languages* (pp. 313-327). Springer Berlin Heidelberg.

[64] DeLoach, S. A. (1999). *Multiagent systems engineering: a methodology and language for designing agent systems*. AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH DEPT OF ELECTRICAL AND COMPUTER ENGINEERING.

[65] Wood, M. F., & DeLoach, S. A. (2001, January). An overview of the multiagent systems engineering methodology. In *Agent-Oriented Software Engineering*(pp. 207-221). Springer Berlin Heidelberg.

[66] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., & Lorensen, W. E. (1991).*Object-oriented modeling and design* (Vol. 199, No. 1). Englewood Cliffs: Prentice-hall.

[67] Coleman, D., Arnold, P., Bodoff, S., Dollin, C., Gilchrist, H., Hayes, F., & Jeremaes, P. (1994). *Object-oriented development: the fusion method*. Prentice-Hall, Inc.

[68] Wooldridge, M., Jennings, N. R., & Kinny, D. (2000). The Gaia methodology for agent-oriented analysis and design. *Autonomous Agents and multi-agent systems*, *3*(3), 285-312.

[69] Zambonelli, F., Jennings, N. R., & Wooldridge, M. (2003). Developing multiagent systems: The Gaia methodology. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, *12*(3), 317-370.

[70] Kruchten, P. (2004). *The rational unified process: an introduction*. Addison-Wesley Professional.

[71] Bernon, C., Gleizes, M. P., Picard, G., & Glize, P. (2002). The Adelfe Methodology For an Intranet System Design. *AOIS@ CAiSE*, *57*.

[72] Pavón, J., Gómez-Sanz, J. J., & Fuentes, R. (2005). The INGENIAS methodology and tools. *Agent-oriented methodologies*, *9*, 236-276.

[73] Wagner, G. (2003). The Agent–Object-Relationship metamodel: towards a unified view of state and behavior. *Information Systems*, *28*(5), 475-504.

[74] Taveter, K., & Wagner, G. (2005). Towards radical agent-oriented software engineering processes based on AOR modelling. *Agent-oriented methodologies*, *10*, 277-316.

[75] Debenham, J., & Henderson-Sellers, B. (2001). Designing agent-based process systems-extending the OPEN Process Framework. *Intelligent agent software engineering*, *8*, 160-190.

[76] Padgham, L., & Winikoff, M. (2003). Prometheus: A methodology for developing intelligent agents. In *Agent-oriented software engineering III* (pp. 174-185). Springer Berlin Heidelberg.

[77] Padgham, L., & Winikoff, M. (2002, November). Prometheus: A pragmatic methodology for engineering intelligent agents. In *Proceedings of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies* (pp. 97-108).

[78] Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., & Mylopoulos, J. (2004). Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, *8*(3), 203-236.

[79] Castro, J., Kolp, M., & Mylopoulos, J. (2002). Towards requirements-driven information systems engineering: the Tropos project. *Information systems*,*27*(6), 365-389.

[80] Giorgini, P., Kolp, M., Mylopoulos, J., & Pistore, M. (2003). The tropos methodology: An overview. *Methodologies And Software Engineering For Agent Systems, Kluwer Academic Publishing (New York)*.

[81] Yu, E. (2011). Modelling strategic relationships for process reengineering.*Social Modeling for Requirements Engineering*, *11*, 2011.

[82] Odell, J., Parunak, H. V. D., & Bauer, B. (2000). Extending UML for agents.*Ann Arbor*, *1001*, 48103.

[83] Henderson-Sellers, B. (Ed.). (2005). *Agent-oriented methodologies*. IGI Global.

[84] Cernuzzi, L., Juan, T., Sterling, L., & Zambonelli, F. (2004). The gaia methodology. In *Methodologies and Software Engineering for Agent Systems*(pp. 69-88). Springer US.

[85] Ivanisevich, J., P. Lorenzi, S. Skinner, and P. Crosby. (1997). *Management Quality and Competitiveness,* Second Edition. IrWin/MsGraw-Hill, New York.

[86] Humphrey, W. S. (1997). *Introduction to the personal software process*. Addison-Wesley Professional.

[87] Galván, I. M. L. (2005). *Evaluación de los sistemas de información desde una perspectiva blanda. Propuesta metodológica* (Tesis de grado).

[88] Olsina Santos, L. A. (1999). *Metodología cuantitativa para la evaluación y comparación de la calidad de sitios web* (Tesis Doctoral).

[89] Apuntes de Cátedra (2010). *Administración de los Sistemas de Información*. Departamento de Informática. Facultad de Cs. Exactas y Tecnologías. UNSE.

[90] De Antonio, A. (1999). GESTIÓN, CONTROL Y GARANTÍA DE LA CALIDAD DEL SOFTWARE.

[91] EYSSAUTIER De la Mora, M. (2002). *Metodología de la investigación: desarrollo de la inteligencia/. México, Ecafsa*.

[92] Basili, V., Caldiera, G., & Rombach, H. D. (1994). *Goal question metric (gqm) approach*. Encyclopedia of Software Engineering.

[93] Pressman, R. S. (2010). Ingeniería de Software, Séptima Edición, Ed.

[94] Derniame, J. C., Kaba, B. A., & Wastell, D. (1999). Software Process: Modelling and Technology (Lecture Notes in Computer Science).

[95] Vásquez, D., Pardo, C., Collazos, C. A., & Pino, F. J. (2010). Modelo liviano de medidas para evaluar la mejora de procesos de desarrollo de software MLM-PDS. *Ingeniería y Ciencia*, *6*(12), 171-202.

ANEXO A

ENCUESTA "OBJETIVOS DE MEDIDA"

Encuesta "Objetivos de medidas"

Esta encuesta está dirigida a personas que hayan utilizado alguna/s de las Metodologías Orientadas a Agentes (MOA) existentes para desarrollar agentes software completos o partes de ellos o Sistemas multiagentes. La finalidad es recolectar información sobre aspectos que la MOA utilizada cubre del proceso de desarrollo y obtener observaciones/sugerencias que podrían emplearse para mejorar el desempeño de dicha metodología. La información obtenida será empleada en el TFG titulado "Propuesta de un marco para la evaluación estructural de Metodologías Orientadas a Agentes".

Lugar y Fecha:

Nombre encuestado/a:

Trabajo desarrollado utilizando tecnología de agentes (módulos, sistemas varios, sistemas multiagentes, u otros):

Metodología Orientada a Agentes (MOA) utilizada para tal desarrollo:

Grado de conocimiento sobre la tecnología de agentes (marca con X el grado seleccionado):

Bajo	Intermedio	Alto	Experto

Desde ya, muchas gracias por su colaboración!

Para asignar valores a cada una de las preguntas definidas para cada objetivo de medida, remitirse a la Tabla anexa que contiene las escalas de valoración con su correspondiente descripción. Luego, se pide una evaluación general y observaciones/sugerencias.

Obj.1. Analizar la MOA utilizada con el propósito de evaluar el alcance de la misma, respecto a los conceptos propios de la tecnología de agentes, desde el punto de vista de los desarrolladores en el contexto del desarrollo de agentes.		
PREGUNTA/S		**VALORACIÓN**
P1.1	Permite representar las tareas a realizar por el agente?	
P1.2	Permite describir las acciones que llevara a cabo el agente con el fin de realizar las tareas?	
P1.3	Permite describir los servicios que proporcionara el agente?	
P1.4	Permite captar datos del entorno del agente?	
P1.5	Permite representar el paso de mensaje entre agentes?	
P1.6	Brinda un protocolo para la comunicación entre agentes?	
P1.7	Permite representar de manera abstracta la función del agente?	
P1.8	Permite describir las normas de actuación que deben seguir los agentes?	
P1.9	Permite agrupar los agentes con un fin común?	
P1.10	Permite representar la colaboración entre agentes y grupos de agentes?	

RESPUESTA	***VALORACIÓN***
La MOA no hace referencia al concepto.	**0**
La MOA hace referencia al concepto de forma limitada, no aborda muchas cuestiones del mismo.	**1**
La MOA se refiere al concepto con deficiencias menores, cubre la mayoría de las cuestiones que describe el mismo.	**2**
La MOA contempla plenamente el concepto.	**3**

La MOA cubre de manera general los conceptos de la tecnología de agentes. (Marcar con X la respuesta seleccionada. Campo Observaciones / sugerencias obligatorio)

Nula	Suficiente	Parcial	Total	Observaciones / sugerencias

Obj.2. Analizar la MOA con el propósito de evaluar el alcance de la misma, respecto a las propiedades de los agentes, desde el punto de vista de los desarrolladores en el contexto del desarrollo de agentes.		
PREGUNTA/S		**VALORACIÓN**
P2.1	¿Cuál es el grado en que la MOA permite representar la autonomía de un agente?	
P2.2	¿La MOA permite desarrollar agentes reactivos?	
P2.3	¿Permite representar los objetivos y lo planes de un agente?	
P2.4	¿Trata la iniciativa para la persecución de objetivos de forma autónoma?	
P2.5	¿Contempla la racionalidad de un agente?	
P2.6	¿Trata las relaciones que emergen de los agentes entre sí y con el entorno?	
P2.7	¿Contempla que el funcionamiento del agente cambie en función de su experiencia previa?	
P2.8	¿Trata la movilidad de un agente de una maquina a otra?	
P2.9	Permite representar la sociabilidad (cooperación, interacción, negociación y organización) entre agentes, con el entorno y con sus usuarios?	

RESPUESTA	***VALORACIÓN***
La MOA no hace referencia a la propiedad.	**0**
La MOA hace referencia a la propiedad, pero no proporciona detalles de la misma.	**1**
La MOA hace referencia a la propiedad de forma limitada, no aborda muchas cuestiones de la misma.	**2**
La MOA se refiere a la propiedad con deficiencias menores, cubre la mayoría de las cuestiones que describe la misma.	**3**
La MOA contempla plenamente la propiedad.	**4**

La MOA cubre de manera general las propiedades de los agentes. (Marcar con X la respuesta seleccionada. Campo Observaciones / sugerencias obligatorio)

Nula	Suficiente	Parcial	Total	Observaciones / sugerencias

Obj.3. Analizar la MOA con el propósito de evaluar el alcance de la misma, respecto a las propiedades de las notaciones y técnicas de modelado, desde el punto de vista de los desarrolladores en el contexto del desarrollo de agentes.

PREGUNTA/S		**VALORACIÓN**
P3.1	Permite una fácil comprensión del método a utilizar?	
P3.2	Permite una facilidad de uso del método?	
P3.3	Brinda la capacidad de representar los conceptos del agente?	
P3.4	Permite describir las propiedades del agente?	
P3.5	Permite especificar el agente de forma iterativa e incremental?	
P3.6	Permite comprobar la consistencia o implicaciones de los modelos?	
P3.7	Permite interpretar los modelos existentes de manera simbólica?	
P3.8	Permite interpretar los modelos existentes de manera sintáctica?	
P3.9	Permite interpretar los modelos existentes de manera semántica?	
P3.10	Permite la reutilización de los modelos?	
P3.11	Permite ampliar los conceptos básicos?	
P3.12	Permite ampliar los bloques de construcción?	

RESPUESTA	***VALORACIÓN***
La MOA no hace referencia a la propiedad.	**0**
La MOA hace referencia a la propiedad, pero no proporciona detalles de la misma.	**1**
La MOA hace referencia a la propiedad de forma limitada, no aborda muchas cuestiones de la misma.	**2**
La MOA se refiere a la propiedad con deficiencias menores, cubre la mayoría de las cuestiones que describe la misma.	**3**
La MOA contempla plenamente la propiedad.	**4**

La MOA cubre de manera general las propiedades de las notaciones y técnicas de modelado. (Marcar con X la respuesta seleccionada. Campo Observaciones / sugerencias obligatorio)

Nula	Suficiente	Parcial	Total	Observaciones / sugerencias

Obj.4. Analizar la MOA con el propósito de evaluar el alcance de la misma, respecto al proceso de desarrollo, desde el punto de vista de los desarrolladores en el contexto del desarrollo de agentes.		
PREGUNTA/S		**VALORACIÓN**
P4.1	La MOA se adapta a la creación a nuevo software?	
P4.2	Se adapta al prototipado y reutilización de componentes?	
P4.3	Representa las fases del ciclo de vida?	
P4.4	Permite definir las tareas propias de cada fase?	
P4.5	Permite la obtención de entregables?	
P4.6	Permite definir el plan del proyecto?	
P4.7	Genera especificaciones?	
P4.8	Valida el diseño?	
P4.9	Genera código?	
P4.10	Genera documentación?	
P4.11	Permite la evolución del método?	

RESPUESTA	***VALORACIÓN***
La MOA no hace referencia al concepto.	**0**
La MOA hace referencia al concepto de forma limitada, no aborda muchas cuestiones del mismo.	**1**
La MOA se refiere al concepto con deficiencias menores, cubre la mayoría de las cuestiones que describe el mismo.	**2**
La MOA contempla plenamente el concepto.	**3**

La MOA cubre de manera general al proceso de desarrollo. (Marcar con X la respuesta seleccionada. Campo Observaciones / sugerencias obligatorio)

Nula	Suficiente	Parcial	Total	Observaciones / sugerencias

Obj.5. Analizar la MOA con el propósito de evaluar el alcance de la misma, respecto a los aspectos pragmáticos, desde el punto de vista de los desarrolladores en el contexto del desarrollo de agentes.		
PREGUNTA/S		**VALORACIÓN**
P5.1	Brinda herramientas de desarrollo gratuitas?	
P5.2	Brinda herramientas de desarrollo pagas?	
P5.3	Cuenta con información actualizada y en línea?	
P5.4	Brinda grupos de apoyo?	
P5.5	Los recursos que brinda se adaptan al proyecto?	
P5.6	El entregable se adapta a la plataforma idónea?	
P5.7	Permite manejar el crecimiento continuo?	

RESPUESTA	***VALORACIÓN***
La MOA no hace referencia al concepto.	**0**
La MOA hace referencia al concepto de forma limitada, no aborda muchas cuestiones del mismo.	**1**
La MOA se refiere al concepto con deficiencias menores, cubre la mayoría de las cuestiones que describe el mismo.	**2**
La MOA contempla plenamente el concepto.	**3**

La MOA cubre de manera general los aspectos pragmáticos. (Marcar con X la respuesta seleccionada. Campo Observaciones / sugerencias obligatorio)

Nula	Suficiente	Parcial	Total	Observaciones / sugerencias