ARQUITECTURA Y PROCESO PARA LA CONSTRUCCION DE AMBIENTES CASE INTEGRADOS

Losavio, F. Matteo, A.
Centro Isys
Universidad Central de Venezuela
flosavio@isys.ciens.ucv.ve, amatteo@isys.ciens.ucv.ve
Caracas, Venezuela

Perez, M.
Departamento Procesos y Sistemas
Universidad Simón Bolívar
movalles@usb.ve
Caracas, Venezuela

ABSTRACT
La tendencia del enfoque CASE en términos tecnológicos, es la integración de herramientas que asisten la aplicación de métodos puntuales, con la expectativa de lograr en lo posible el soporte automatizado total a un método de desarrollo. En este sentido, la arquitectura del ambiente, compuesta por la plataforma (el hardware, el software del sistema operativo, incluida la red y la gestión de la base de datos) constituye la base del ambiente CASE. Aún cuando se pueden obtener beneficios a partir de las herramientas CASE, su verdadera potencia se logra a través de la integración. Aunque el problema de integrar herramientas está planteado desde hace dos décadas, las herramientas integradas trabajan bien solo en casos aislados. Los trabajos actuales en integración hacen énfasis en la interoperabilidad de las herramientas especialmente en ambientes distribuidos. La integración sigue siendo un tópico incompleto. Las conversiones de los datos son pasivas y el ordenamiento de las actividades sigue siendo rígido. Los ambientes integrados están bajo la presión de incorporar capacidades para manejar dependencias complejas y seleccionar que herramienta usar. Se propone entonces en este trabajo un proceso para aplicar una arquitectura basada en el modelo multiagentes para sistemas interactivos para construir ambientes CASE integrados. Además, se presentan los resultados de aplicar el proceso e instanciar la arquitectura propuesta para un caso real.

PALABRAS CLAVES: arquitecturas de software, CASE integrados, ambientes CASE orientados a objeto, PAC.

1. INTRODUCCION

Las organizaciones modernas que basan sus procesos en la información necesitan contar con Sistemas de Información que las integren. Los servicios que ellas presten para el logro de sus beneficios deben responder rápidamente ante las presiones del mercado. Estos hechos ejercen una alta presión sobre las organizaciones que desarrollan sistemas. Es por ello que los ambientes Computer Aided Software Engineering (CASE) integrados han pasado a ser un componente estratégico para estas organizaciones (Livari, 1996). Por otro lado, (Tatsuta, 1996) señala que hoy día casi todos los profesionales de las Tecnologías de la Información en el mundo conocen el valor de la Tecnología Objeto. Los obstáculos más grandes ante este paradigma son la falta de experticia, una línea de herramientas de soporte incompleta y la falta de partes o componentes reutilizables. Enfatiza, además que para resolver estos problemas las herramientas que soporten el desarrollo de software bajo el enfoque orientados a objeto (OO) son esenciales. En la actualidad hay una oferta muy variada de estas herramientas. Probablemente la característica más importante buscada por los compradores es el grado en que ellas están integradas. Integración se refiere a la cantidad de información que se comparte entre las funciones del ambiente y los servicios que el ambiente ejecuta automáticamente. Se plantea entonces como objetivo de este trabajo diseñar un "framework" o arquitectura orientada a objetos, para la construcción de ambientes CASE integrados. Para ello se propone una arquitectura de alto nivel (Losavio et al, --) basada en el patrón arquitectural de ambientes interactivos basado en el modelo multiagente Presentation-Abstraction-Control (PAC) (Bass y Coutaz, 1991), la cual garantiza la separación precisa de funciones, un buen soporte para los cambios y extensiones y para la multitarea. Esta es una estructura mantenible con una clara diferenciación de las funcionalidades esperadas para un ambiente CASE integrado. Tambien se propone un proceso orientado a objetos (OO) que permite construir un ambiente CASE integrado utilizando esta arquitectura. Este artículo consta de cuatro secciones, además de la introducción y las conclusiones. En la primera de ella se tratan los conceptos asociados al enfoque CASE, en la segunda sección se propone la arquitectura para un ambiente CASE integrado y se detallan sus ventajas, en la tercera sección se describe el proceso para la construcción de estos ambientes con base a la arquitectura propuesta y las técnicas a utilizar en el proceso y en la cuarta sección se hace una instanciación del método para un caso real.

2. AMBIENTES CASE INTEGRADOS

La arquitectura del ambiente, compuesta por la plataforma, hardware y el software del sistema operativo (incluida la red y la gestión de la base de datos) constituye la base de un ambiente CASE. Sin embargo, aún cuando se pueden obtener beneficios a partir de las herramientas CASE aisladas, su verdadera potencia se logra a través de la integración.

Dos enfoques independientes son los que reflejan el estado del arte en las arquitecturas de los ambientes CASE en la práctica. El primero se fundamenta en la arquitectura propuesta para el soporte completo del proceso de desarrollo del software basado en un conjunto holístico de mecanismos compartidos que facilitan la integración y la coordinación de las herramientas. Este soporte se conoce como ambiente de soporte integrado a proyectos (Integrated Project Support Environment – IPSE). El segundo enfoque toma en cuenta la coordinación entre herramientas independientes con un enfoque "bottom up", llamado frecuentemente tecnología del mercado de herramientas CASE (CASE Tool Market Technology – CTMT). Este enfoque es el seguido por las empresas comerciales de software (Brown et. al, 1994) y consiste en juntar y producir múltiples herramientas, hasta llegar a una coalición de herramientas CASE, cubriendo un alto rango de funcionalidades propias del ambiente CASE.

Brown (Brown et al, 1994) propone un nuevo modelo conceptual para entender la integración de herramientas que conforman un ambiente CASE. El ambiente que resulta de la aplicación del modelo, lo denomina "ambiente federativo" con la idea de indicar que ésta es una arquitectura que presta atención especial a las conexiones necesarias entre los componentes del ambiente. En el modelo propuesto por el autor, se distinguen tres niveles de concentración en un proceso de integración para construir un ambiente (ver figura Nº 1).

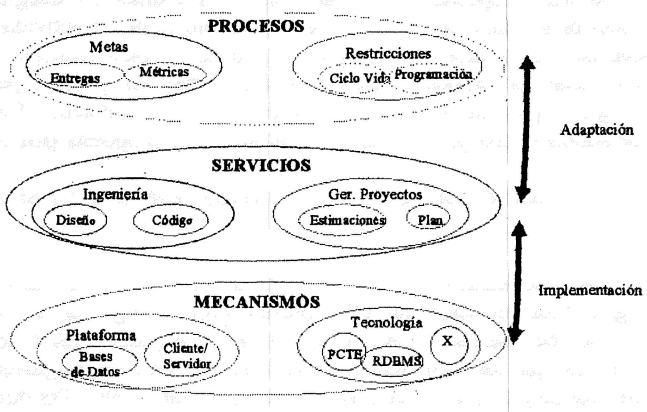

Figura Nº1. Modelo Conceptual de Integración.

El nivel central es el de los servicios disponibles. El tercer nivel consiste de los mecanismos que implementan los servicios para los usuarios finales. El primer nivel se interesa por el proceso que incluye el conjunto de objetivos y restricciones de un proyecto u organización, proveyendo el contexto de reglas en el cual los servicios del usuario final serán establecidos.

3. ARQUITECTURA PARA UN AMBIENTE CASE INTEGRADO

La arquitectura del software se obtiene mediante un proceso de partición, que relaciona los elementos de una solución de software con partes de un problema del mundo real definido implícitamente durante el análisis de los requerimientos (Shaw y Garlan, 1996). La arquitectura del sistema de software define al sistema en términos de elementos computacionales y de las interacciones entre ellos. Estos componentes se pueden especificar desde diferentes vistas para poder mostrar sus propiedades relevantes. En esta sección se hará una breve descripción de las

diferentes arquitecturas de software que hasta ahora se han propuesto para construir ambientes integrados para el desarrollo de aplicaciones (Buschmann et al, 1996).
Un patrón o estilo arquitectural expresa un esquema de organización estructural fundamental para sistemas de software específicos. La selección de un patrón estructural durante la actividad de diseño es fundamental ya que su selección tendrá impacto en la arquitectura de los subsistemas que lo integren (Buschmann et al, 1996). El patrón arquitectural Presentación-Abstracción-Control (PAC) (Losavio y Matteo, 1996) (Bass y Coutaz, 1991) inspirado en el Model View Controller (MVC) (Buschmann et al, 1996) (Gamma et al, 1995) define una estructura para sistemas de software interactivos en la forma de una jerarquía de agentes cooperando. Cada agente es responsable de un aspecto específico de la funcionalidad de la aplicación y consiste de tres componentes: presentación, abstracción y control. Esta subdivisión separa los aspectos de interacción hombre-máquina del agente de su funcionalidad o semántica y de su comunicación con los otros agentes.
Con base al modelo conceptual de Brown (Brown et al, 1994), se propusieron algunas de las posibles funcionalidades esperadas para un Ambiente CASE Orientado a Objeto(OOCE) (Losavio et al, 1996). Definimos un OOCE como un ambiente CASE que soporta y está construido bajo el enfoque de orientación a objetos (OO). Tomando en consideración estas funcionalidades, se propone un "framework" con elementos reutilizables en tres capas (Ver figura N° 2). Estas, a su vez, serán realizadas por uno o varios "frameworks" según amerite.

CAPA	FUNCIONALIDAD
INTERFAZ	• Edición de modelos • Interfaz de acceso a todas las funcionalidades del CASE Integrado
SEMANTICA	• Soportar el análisis, el diseño y la implementación • Gerencia de proyectos
INTEGRACION	• Soporte a la seguridad • Soporte a la integración de herramientas

Figura N° 2. Arquitectura de tres capas para un OOCE

Capa Interfaz: Esta capa se usa para la presentación visual de todos los objetos presentes en la capa semántica y para la interacción con las funcionalidades del OOCE. Capa Semántica: Esta capa proporciona la funcionalidad que garantiza el desarrollo y consistencia de los diferentes modelos que se desarrollan durante el proceso de elaboración del software, en sus diferentes etapas y según las necesidades de la organización. Capa Integración: Para esta capa se proponen dos posibles soluciones. La primera correspondiente a esquemas de seguridad, la cual garantiza aspectos tales como: soporte al desarrollo de software para diferentes tipos de usuarios, soporte a la concurrencia de estos equipos de trabajo, control de acceso a los artefactos de software y la gerencia de versiones. Para la segunda arquitectura, es decir la de integración de herramientas del ambiente, se debe prever dos alternativas de diseño excluyentes. Para la segunda arquitectura, es decir la de integración de herramientas del ambiente, se debe prever dos alternativas de diseño excluyentes. Es decir, una arquitectura que soporte la integración por datos (un sistema de gestión de objetos) o una arquitectura que soporte la integración por mensajes (un servidor de mensajes)

La arquitectura aquí propuesta (Losavio et al, --) se puede considerar como *una arquitectura de alto nivel* para la construcción de un OOCE y está compuesta a su vez, por otros *componentes arquitectónicos*, vistos en capas, y de un conjunto de *orientaciones* para combinarlos de acuerdo al modelo conceptual propuesto por Brown, el cual exige la definición de un proceso y los servicios que prestará el OOCE. Para un diseño de más detalle de la arquitectura de alto nivel se propone utilizar el patrón arquitectural para el desarrollo de sistemas interactivos basados en modelos multiagentes PAC. Se recurre a este patrón para representar la arquitectura porque garantiza la separación de funciones, es decir, cada concepto semántico en el dominio de la aplicación En nuestro caso las herramientas CASE, están representadas por agentes diferentes. Para el caso del OOCE, la arquitectura usando el modelo PAC que se propone, se muestra en la figura N° 3. Esta arquitectura será denominada PAC-OOCE.

Se identifican los siguientes niveles: Nivel de Integración (0), Nivel de Proceso (1), Nivel de Herramientas (2) y Nivel de Interfaz (3). Sus características son las siguientes:

El nivel de Integración(0): en el se encuentra un agente PAC raíz con sus tres formalismos, que se caracteriza por expresar la semántica y control del todo el sistema integrado y se propone: (Ver figura N° 4)

- Que el formalismo abstracción cubra a su vez tres ámbitos: la abstracción referente a los servicios de desarrollo del OOCE (A1). La abstracción referente a los servicios de gerencia del proceso de desarrollo del software (A2). Y la abstracción referente al soporte de la concurrencia y de la gerencia de versiones (A3).
- Así mismo, el formalismo de control, también contempla varios ámbitos: Un primer ámbito que garantice la consistencia entre los modelos de análisis, diseño e implementación garantizando así la difusión de los cambios ocurridos entre las herramientas responsables de estas funcionalidades. Un segundo ámbito que garantice la difusión de los cambios ocurridos entre las herramientas responsables de la gerencia y soporte al proceso de desarrollo del software. Y un tercer ámbito de control que garantice la difusión de los cambios entre las herramientas responsables de la concurrencia y gerencia de versiones.

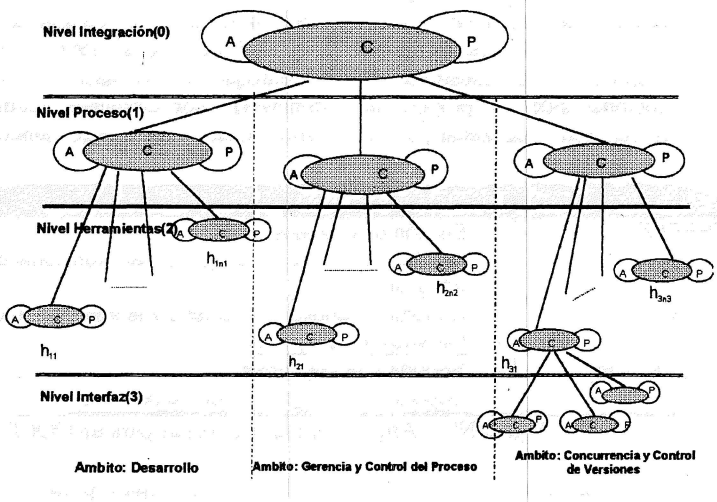

Figura Nº 3. Arquitectura PAC- OOCE

Sin embargo, para garantizar completamente la integración, el formalismo de control deberá contar además con una capa superior de abstracción quien será responsable de la difusión de los cambios entre ámbitos; es decir, cuando estos ocurran en un ámbito y deben ser difundidos a otros. Por ejemplo, una actualización en la herramienta responsable de la planificación del proyecto podría implicar una actualización en algún modelo de análisis (el cual pertenece al ámbito desarrollo).

Para el caso de aplicaciones distribuidas, este formalismo de control también será responsable de satisfacer los requerimientos o servicios de clientes remotos en cuyo caso se puede definir como un *broker*. (Losavio y Matteo, --). En el caso particular de aplicaciones bajo un modelo cliente/servidor en una red local el control de este agente puede verse como un patrón del tipo *mediator* (Levy et al, 1998). Los patrones arquitecturales broker y mediator están definidos en (Buschamnn et al, 1996) y (Gamma et al, 1995) respectivamente. De la misma manera, el formalismo presentación de éste agente raíz, garantizara la integración por presentación del OOCE. Quedando entonces la estructura del agente PAC raíz según la figura Nº 4.

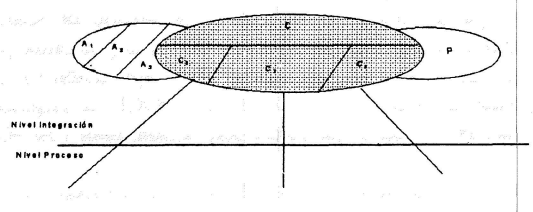

Figura Nº 4. Agente PAC raíz correspondiente al Nivel de Integración de la arquitectura PAC-OOCE

Nivel de Proceso (1): Contiene tres agentes PAC complejos (subsistemas) que tienen por finalidad definir el proceso de desarrollo utilizado en el ambiente OOCE y que se pretende implantar. Estos agentes PAC corresponden a subsistemas que se responsabilizan por definir los servicios esperados para cada ámbito, según sea el proceso de desarrollo de software previsto. Con este nivel se estaría implementando la personalización del proceso de desarrollo de software según el modelo conceptual de Brown (Brown, et al, 1994). Cada uno de estos agentes tiene la responsabilidad de registrar la cantidad y las características de las herramientas que contendrá cada ámbito. A través de su formalismo de control se establecerán el enlace entre la aplicación (agente PAC raíz) y las correspondientes herramientas, esto es siempre de acuerdo a la filosofía del modelo PAC. La presentación de estos agentes permitirá el acceso a las distintas herramientas del ámbito además de las funcionalidades generales referentes al proceso.

Nivel de Herramientas (2): Los agentes PAC ubicados en el nivel 2 representan las herramientas que conforman el OOCE respetándose en cada una de ellas su presentación, su abstracción y su control. Nótese que de todo el conjunto de herramientas, ubicadas en este nivel, se puede hacer una clasificación de las mismas en función del ámbito al cual pertenecen. Es decir, el OOCE, podrá tener un subconjunto de herramientas ($1_1 \ldots 1_{n1}$) (Ver figura Nº 3) dedicadas a prestar los servicios directamente relacionados con el análisis, diseño y generación de software. Un segundo subconjunto de herramientas ($2_1 \ldots 2_{n2}$) dedicadas a prestar los servicios de gerencia y control del proceso de desarrollo del software, tales como: planificación de proyectos y gerencia de los esquemas de seguridad. Y un tercer conjunto de herramientas ($3_1 \ldots 3_{n3}$) dedicadas a prestar los servicios de soporte a la concurrencia y gerencia de versiones. De allí que se han identificado tres ámbitos: Desarrollo, Gerencia y Control del proceso, y Concurrencia y Control de versiones.

Nivel de Interfaz(3): Se puede dar la situación que alguna de las herramientas use, a su vez, el modelo PAC para representar su arquitectura, dándose entonces la situación de que en el tercer nivel cuente con agentes PAC que sean componentes gráficos compuesto o elementales, tal como en el caso de la figura Nº 3.

Ventajas de la arquitectura propuesta

Con base a las características que debe poseer una arquitectura de software definidas por (Sommerville, 1996), (Pressman, 1998), (Pfleeger, 1998), se procede entonces a verificar la presencia de las mismas en la arquitectura propuesta en este trabajo.

Característica	Evaluación
Conformidad Funcional	Se obtiene un método para instanciar la arquitectura propuesta. Partiendo de la especificación del proceso de desarrollo que la arquitectura soportará. Además de definir los servicios (funcionalidades) que este ambiente suministrará. Se está garantizando una interacción con el usuario, un completo entendimiento de las funcionalidades esperadas y la satisfacción de los mismos al distribuirlas según la estructura de agentes PAC prepuesta.
Modularidad	Por definición los agentes PAC son modulares. Cada uno es responsable de su funcionalidad, reflejado en su formalismo abstracción, a su vez es responsable de la presentación y de la comunicación entre ellos a través del formalismo control. Ocultándose para el resto de los componentes sus aspectos internos de diseño e implementación.
Niveles de Abstracción	Al establecer una diferenciacion por capas y por ámbito, se está manejando con precisión los niveles de abstracción. Las capas cumplen un rol de coordinación entre los diferentes niveles. Los ámbitos cumplen un rol de composición.
Cohesión	En la arquitectura propuesta se diferencia claramente la descomposición de funciones.
Acoplamiento	El estilo arquitectural seleccionado (PAC) tiene como ventaja el bajo acoplamiento. Lo mismo sucede con su nivel de refinamiento como es el esquema de objetos.
Adaptabilidad	Como una consecuencia del bajo acoplamiento y de la alta cohesión, algunos agentes PAC son herramientas CASE completas; se consigue alta adaptabilidad. Si se desea incorporar una nueva herramienta, es suficiente con incorporar un nuevo agente PAC a la arquitectura.
Complejidad	Esta es la característica menos presente en la arquitectura propuesta. Por definición los ambientes CASE integrados son complejos. A nivel de componente la complejidad estará limitada por los servicios que preste el componente. Los agentes que implementan las herramientas CASE del ambiente son tan complejos como sean las herramientas. Sin embargo, con esta propuesta se desea simplificar el mecanismo de integración. Esta responsabilidad está localizada en los formalismos de control de pocos agentes, lo cual apunta hacia una simplificación de la complejidad del mecanismo.

4. PROCESO PARA LA CONSTRUCCION DE AMBIENTES CASE INTEGRADOS

En la presente sección se describirán las fases sugeridas como parte de un proceso orientado a objetos, para la aplicación de la arquitectura PAC-OOCE para construir ambientes CASE integrados. Estas son:

- Fase N°1: Definición del proceso de desarrollo de software que será soportado por el ambiente CASE integrado. La frase "definir los procesos de la organización" supone la claridad de dos elementos: 1) el concepto de proceso y 2) a partir del concepto, especificar el lenguaje con el cual se describirá. Su definición permitirá la especificación del formalismo control del agente PAC raíz de la arquitectura PAC-OOCE.

  De acuerdo al Software Productivity Consortium (Software, 1996) un proceso es un conjunto de actividades, productos y roles relacionados. Existen diferentes lenguajes de especificación de procesos, durante esta fase se puede utilizar cualquiera de ellos. (Yourdon, 92), (Logic Works, 1996).

- Fase N° 2: Definición de los servicios que deberá prestar el ambiente CASE integrado. Con base a la visión compartida del proceso de desarrollo de software antes descrita, se procederá a definir los servicios que debería prestar el ambiente CASE integrado. Para ello es necesario, previamente, identificar cuales de estos servicios se pueden automatizar. A su vez deben ser agrupados por ámbito en concordancia con la arquitectura PAC-OOCE.

  De acuerdo al modelo conceptual utilizado en este trabajo, un servicio es un software que persigue facilitar una funcionalidad. La especificación de estos servicios se hará usando el Modelo de Use Case de OOSE (Jacobson et al, 1992). Los servicios se presentan en forma de "*Wizards*" los cuales permitirán guiar a los usuarios (analistas y programadores) para realizar lo necesario de la forma más fácil y rápida posible, asegurando un orden metodológico que permitirá aumentar la calidad de lo que se construye y un mantenimiento sin traumas. Estas funcionalidades, de ahora en adelante denominadas servicios, siempre estarán agrupadas con base a los ámbitos descritos en la arquitectura PAC-OOCE. La figura N° 5 describe el modelo de Use Case.

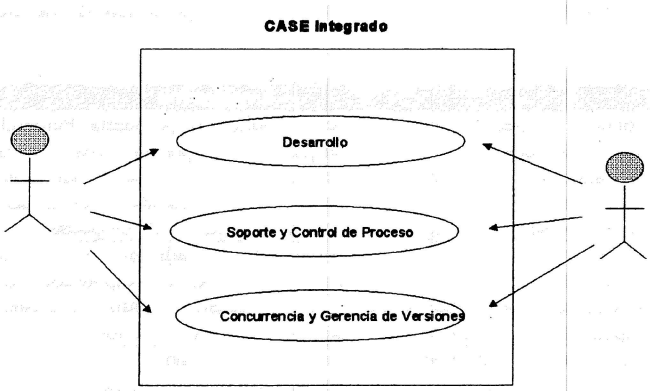

Figura N° 5. Modelo de Use Case del CASE integrado

Al igual que para la definición de los procesos actualmente se cuenta con diferentes técnicas para la definición de los servicios (Rational, 1997) (Whitten y Bentley, 1998). Cualquiera de ellas puede ser utilizada en esta fase. Al identificar los servicios se inicia la definición de los formalismos de control de los agentes PAC de la arquitectura PAC-OOCE para el Nivel Proceso.

- Fase N° 3. Reutilización de las herramientas CASE. Esta fase se puede ejecutar de manera concurrente con la fase anterior. La meta planteada para esta fase es verificar si entre las herramientas CASE adquiridas por la organización, alguna de ellas suple los servicios antes definidos. Con esta actividad se están definiendo cuáles son los agentes PAC que intervendrán en el nivel de Herramientas de la arquitectura PAC-OOCE.

- Fase Nº 4. Definición de los modelo objetos de los agentes PAC. Como una consecuencia de la especificación de los servicios esperados, realizada en la fase Nº 2 se procederá a definir el modelo objeto, por ámbito, el cual corresponderá al formalismo abstracción de los agentes PAC del Nivel Proceso de la arquitectura PAC-OOCE. Este modelo objeto por ámbito, permite la integración del conjunto de herramientas dentro del mismo. Esta fase es una adaptación de uno de los pasos propuestos por Losavio y Matteo en (Losavio y Matteo, 1997) para el desarrollo de Interfaces de Usuarios. La meta es obtener, por ámbito, un esquema de objetos, similar al propuesto en la figura Nº 6.

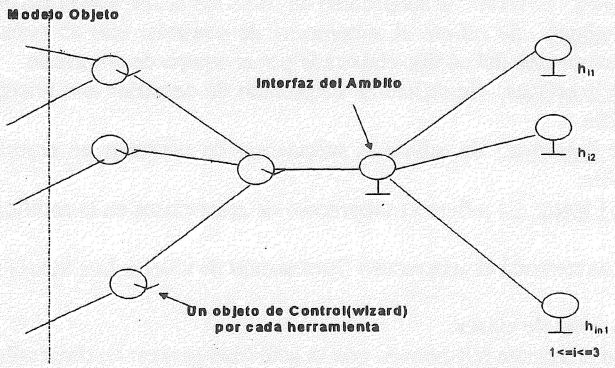

Figura Nº 6. Esquema de objetos por ámbito.

Técnicas propuestas para la aplicación del proceso
Para el caso particular de las Fases Nº 1 y Nº 2, a continuación se describen las técnicas sugeridas para su ejecución.

Fase Nº1: Definición del proceso de desarrollo de software que será soportado por el ambiente CASE integrado. En este sentido, se utilizaron algunas ideas del *Software Productivity Consortium* cuyos miembros pertenecen al *Software Engineering Institute,* (Software, 1996), que van en la misma dirección de Brown, y profundizan cómo especificar un proceso. En (Perez, 1999) se propone un conjunto de pasos para la especificación del proceso de desarrollo de software adaptado del lenguaje propuesto por *Software Productivity Consortium.*

Fase Nº 2: Definición de los servicios que deberá prestar el ambiente CASE integrado que soportara el proceso de desarrollo de software antes definido. Para realizar una especificación de los servicios se ha decidido utilizar la técnica *Use Case,* propuesta por Jacobson et al (Jaconson et al, 1992) pero de manera sistematizada según el enfoque de Whitten y Bantley (Whitten y Bantley, 1998). La razón de tal decisión obedece al alto grado de consistencia en el método propuesto por Whitten y Bantley. Ellos identifican el modelaje con *Use Case* como el proceso de precisar eventos del negocio, quién los inicia, y cómo el sistema responde a estos.

5. CASO DE ESTUDIO: CASE INTEGRADO PARA UNA EMPRESA DE INGENIERÍA CIVIL.

En este caso de estudio se construye un CASE Integrado utilizando la arquitectura y el proceso propuesto por una empresa dedicada a la computación enmarcada por la consultoría gerencial en el área de la Arquitectura y la Ingeniería Civil. Entre sus productos más importantes se encuentran las labores de consultoría y la venta de un software orientado a objetos, llamado SAGA, que permite automatizar el proceso de gerencia de la construcción o reparación de una obra civil. Entre los objetivos del grupo desarrollador de software de la empresa está: **establecer un ambiente de desarrollo que permita lograr el mantenimiento de SAGA con alta calidad, optimizando el recurso necesario y los tiempos de entrega.**

La versión de SAGA en etapa de producción, fue creada sin una planificación técnica adecuada y para ese entonces no se hizo un diseño previo del sistema, ni se utilizó un método específico. Esta situación ha hecho que en la actualidad SAGA esté pasando por un proceso difícil de mantenimiento correctivo y sobre todo adaptativo, por lo

que se hace urgente para la organización con miras a mantener el mercado, la definición de una estrategia que resuelva esta realidad.

Es así como la empresa se aboca a la construcción del ambiente CASE el cual se denominará SAGAOOCE, que soportará su proceso de desarrollo del software. De acuerdo al proceso propuesto se describen los resultados de la aplicación del mismo.

Fase N°1. Definición del proceso de desarrollo de software que será soportado por el ambiente case integrado
Se identificó que el proceso de desarrollo de software para la organización en estudio estará básicamente soportado por cinco (5) subprocesos. Estos son:

1. Control de Cambios. Se refiere al subproceso de controlar la simultaneidad de cambios en un mismo proyecto.
2. Control de Estándares. Se refiere al subproceso de controlar que se estén aplicando los estándares para la creación y documentación del código elaborado por el equipo de desarrollo.
3. Construcción de Interfaces. Se refiere al subproceso de construir una interfaz del sistema desarrollado por el equipo responsable.
4. Construcción de Reportes. Se refiere al subproceso de construir un reporte del sistema desarrollado por el equipo responsable.
5. Construcción de Clases. Se refiere al subproceso de crear clases en la codificación.

A continuación se caracterizará el subproceso Contrucción de Clases. Los demás se realizaron de forma similar.

Subproceso construcción de clases.
Este es un subproceso altamente relacionado con la actividad general de desarrollo de software de la empresa. Tiene que ver con la construcción de una clase cuando se trabaja en el desarrollo de un proyecto utilizando un lenguaje de programación orientado a los objetos. Este subproceso fue especificado utilizando la técnica propuesta para la fase N°1. En la figura N°7 se muestra el esquema del subproceso de construcción de clases.

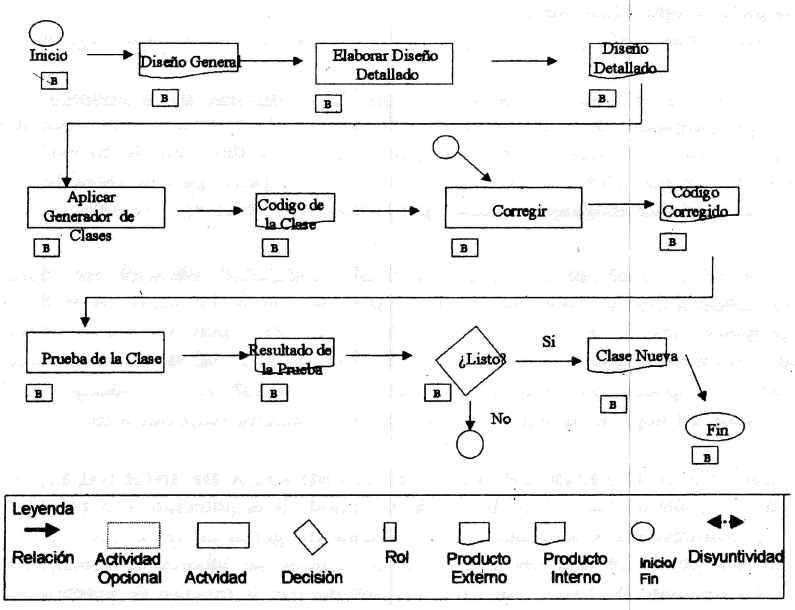

Figura N° 7. Especificación del Subproceso Construcción de Clases

Fase N° 2. Definición de los servicios.
A continuación en la tabla N° 1, se presenta la asociación subproceso-servicio agrupados por ámbito.

Ambito	Subproceso	Servicio
Desarrollo	Construcción de Clases	Wizard Construcción de Clases
	Construcción de Reportes	Wizard para la Construcción de Reportes
	Construcción de Interfaces	Wizard para la Construcción de Interfaces
Soporte y Control del Proceso	Control de Estándares	Wizard para el Control de los Estándares
Concurrencia y Gerencia de Versiones	Control de Cambios	Wizard para el control de Cambios

Tabla N° 1. Servicios del Ambiente CASE para la organización, agrupados por ámbito.

Siguiendo el proceso antes descrito, la especificación de los servicios comienza con la elaboración del modelo de contexto y el modelo de *Use Case* del SAGAOOCE; los cuales se muestran en las figuras N° 8 y N° 9. Nótese que la figura N° 8 se diferencia de la figura N°6 solo en la instanciación de los actores; esto se debe a que los servicios identificados corresponden a los tres ámbitos propuestos en la arquitectura.

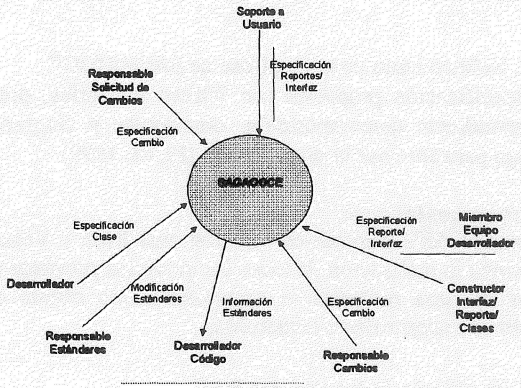

Figura N° 8. Modelo de Contexto del SAGAOOCE

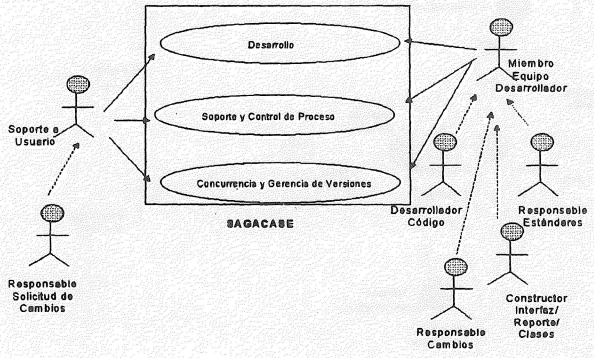

Figura N° 9. Modelo de *Use Case* del SAGAOOCE

A su vez estas tres Use Case, que están representando los servicios esperados para SAGAOOCE según, los ámbitos, pueden ser descompuestas en las Use Case que representan cada uno de los servicios que prestará el ambiente integrado. Este refinamiento se muestra en la figura Nº 10.

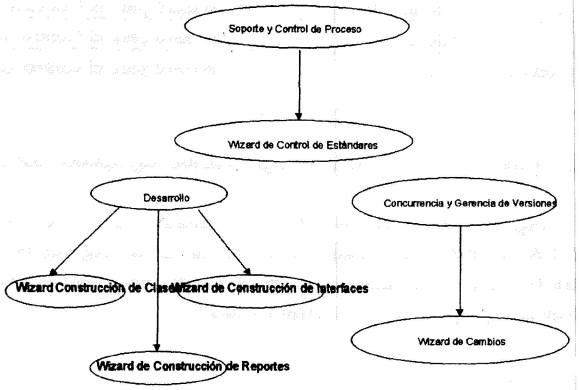

Figura Nº 10. Refinamiento de la Use Case de SAGAOOCE

Continuando con la técnica de especificación propuesta por Whiten y Bentley, por cada servicio identificado se elaboró su correspondiente descomposición, descripción y diagrama objeto. Los cuales no se muestran en este trabajo para abreviar la presentación (Pérez, 1999)

Fase Nº 3. Reutilización de las herramientas case

Para la organización, las herramientas CASE disponibles son: Rational Rose y Visual Age for JAVA. Ambas son reutilizables, en virtud de que permiten la realización de todos los servicios previstos para el ámbito Desarrollo. Sin embargo, con relación a los dos ámbitos restantes la organización no cuenta con herramientas CASE que suministren estos servicios: Tendrán que adquirirlas o construirlas.

Fase Nº 4. Definición de los modelo objetos de los agentes PAC.

Con base a los modelos objetos elaborados en la Fase Nº 2, por cada servicio, se hará a continuación una integración de los mismos, por ámbito. El modelo objeto integrado para el ámbito Desarrollo se muestra en la figura Nº 11. Para la construcción de este modelo objeto se resolvieron los conflictos de estructura, de semántica y de generalización.

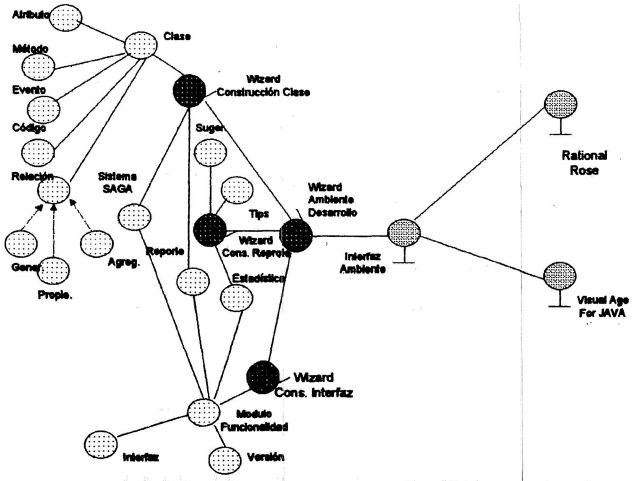

Figura Nº 11. Esquema de Objetos del Ámbito Desarrollo de SAGAOOCE

Luego se hizo la adaptación del esquema de objetos propuesto en el proceso para esta fase. Este esquema de objetos identifica los objetos identidad previamente determinados en la fase Nº2 y los objetos de control y de interfaz

necesarios.

En la figura N°12 se muestra la instanciación de PAC-OOCE para el caso real. El formalismo presentación en el Nivel Herramienta representa la interfaz de cada una de las herramientas del ambiente. A nivel de proceso, este formalismo da las características visuales que debe presentar el ambiente cuando se ingresa en el ámbito Desarrollo, el cual es único para este caso. Para el Nivel Integración, el formalismo abstracción da el acceso a todo el ambiente CASE. Los formalismos de Abstracción a Nivel de Herramientas son los correspondientes modelos información utilizados por cada una de las herramientas. El formalismo abstracción a Nivel de Proceso es el metamodelo que soporta todo el ambiente, el cual se definió en la fase N°4 y esta descrito en la figura N°11. Finalmente, el formalismo abstracción del Nivel Integración contendrá el metamodelo del ambiente completo. Que para este momento coincide con el del Nivel de Proceso.

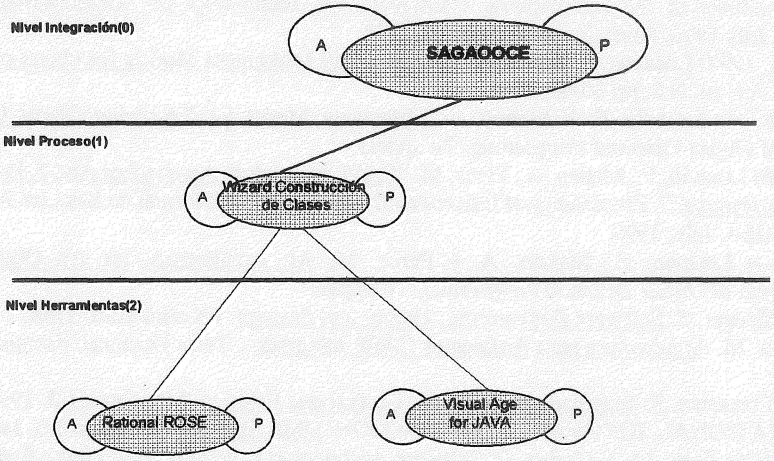

Figura N°12. Arquitectura SAGAOOCE

6. CONCLUSIONES

En este articulo se propuso un proceso que se fundamenta en la arquitectura PAC-OOCE, los cuales permiten la construcción de ambientes CASE integrados. Este proceso es flexible ya que da libertad en la selección de las técnicas a ser aplicadas en sus fases. Esto hace que el proceso sea adaptable a cualquier organización. También se presentaron los resultados de la aplicación de este proceso y de la instanciación de la arquitectura PAC-OOCE para un caso real. Para finalizar, es importante destacar, que esta investigación deja algunos tópicos los cuales serían interesantes investigar. Uno de ellos es lo referente a la propuesta de metamodelos que representen el formalismo abstracción de los tres ámbitos. Igualmente se hace necesario generalizar la arquitectura para aquellas situaciones en las que se desee incorporar al ambiente CASE una herramienta que tenga un comportamiento tipo hipertexto, por ejemplo las asociadas con la tecnología web, ya que estas no son consecuentes con la estructura jerárquica de comunicación que impone el patrón arquitectural PAC. Definitivamente se hace necesario profundizar en el diseño del formalismo de control, si bien el patrón *broker* pareciera ser un estándar, es importante evaluar su desempeño para modelos como los de cliente/servidor en redes locales, por ejemplo.

REFERENCIAS

(Bass y Coutaz, 1991) Bass, L. y Coutaz. Developing Software For The User Interface. Addison – Wesley Pub. Company Reading Masachussets, 1991.

(Brown et al, 1994) Brown A., Carnery D., Morris E., Smith D., Zarrella P. Principles Of CASE Tool Integration. Software Engineering Institute, Oxford University Press, 1994.

(Buschamn et al, 1996) Buschmann, F.; Meunier, R.; Sommerlad, H. Y Stal, M. A System of Pattern. Pattern – Oriented Software Architecture. John Wiley & Sons. 1996.

(Gamma, et al, 1995) Gamma, E.; Hekm, R.; Johnson, R.; Vlissides, J. Design Patterns Elements of Reusable Object Oriented Software. Addison Wesley Proffesional Computing Series. 1995.

(Ghezzi et al, 1991) Ghezzi, C.; Jazayeri, M.; Mandriolo, D. Fundamentals of Software Engineering. Prentice Hall International Editions. 1991.

(Göhner, 1991) Göhner, P. Building IPSE'S By Combining Heterogeneuos CASE Tools.

(Jacobson et al, 1992) Jacobson I., Christerson M., Jonsson P., Övergaard G. Object-Oriented Software Engineering. A Use Case Driven Approach. Addison-Wesley . Fisrt Printed 1992, Revised fourth printing 1993.

(Levy et al, 1998) Levy, N.; Losavio, F. Y Matteo, A. Comparing Architectural Styles: Broker specialiser Mediator. ISAW3. 3erd International Software Architecture Workshop. Orlando. USA. Nov. 1998.

(Livari, 1996) Livari, J. Why Are CASE Tools Not Used?. Communications of The ACM, Vol. 39, N° 10, October, 1996.

(LogicWorks, 1996) Logic Works, INC. BPWin Method. Logic Works, 1996.

(Losavio y Matteo, 1996) Losavio, F. Y Matteo, A. Object-Oriented User-Interface Design based on Agents Framework. Proceedings of the International Conference on Technology of Object-Oriented Languages and Systems. TOOLS, July 1996, Santa Barbara, alifornia, USA.

(Losavio y Matteo, 1997) Losavio, F., Matteo, A. Use case and Multiagent Models for object-orientd design of user interfaces, JOOP, Vol. 10, N 2, pp 30-40, May, 1997.

(Losavio y Matteo, ---) Losavio, F. Y Matteo, A. Multiagent Model for Designing Object Oriented Distributed Systems. Journal of Object Oriented Programing. To appear.

(Losavio et al, 1996) Losavio, F.; Matteo, A.; Perez, M. "CASE Integration Approaches And A Proposal For An Object Oriented CASE Environment", Proceedings of International Conference on Information Systems Analysis and Synthesis ISAS'96, Orlando USA, July, 1996.

(Losavio et al, ---) Losavio, F.; Matteo, A. y Perez, M. An Architecture for An Object Oriented CASE Environment. Journal of Object Oriented Programing. To appear.

(Pfleeger, 1998) Pfleeger, S. Software Engineering, Theory and Practice. Prentice Hall, 1998.

(Perez, 1999) Perez, M. Arquitectura para Ambientes CASE Integrados. Tesis Doctoral, Facultad de Ciencias. UCV. 1999.

(Pressman, 1998) Pressman, R. Ingenieria de Software Un enfoque Práctico. Mc. GrawHill. 1998.

(Rational, 1997) RATIONAL SOFTWARE CORPORATION, UML Summary, Version 1.0, January 1997.

(Shaw y Garlan, 1996) Shaw, M. Y Garlan, D. Software Architectures. Prentice Hall Upper Saddler River, 1996.

(Sommerville, 1996) Sommerville, I. Software Engineering. Fifth Edition. Addison Wesley Publishing Company. 1996.

(Software, 1996) Software Productivity Consortium. Improving the Software Process through Process Definition and Modeling. Software Productivity Consortium. Thomson Computer Press. 1996.

(Tatsuta, 1996) Tatsuta, T. Conference on Computer-Aided Software Engineering Summary Report, Software Engineering Notes, Vol. 21, N° 5, ACM SIGSOFT, September, 1996.

(Whitten y Bentley, 1998) Whitten, J.; Bentley, L. Systems Analysis and Design Methods. Fourth Edition. Irwin McGraw Hill. 1998.

(Yourdon, 92) Yourdon, E. Analisis Estruturado Moderno. Prentice Hall. 1992.