# Una Propuesta Arquitectónica para el Desarrollo de Aplicaciones Colaborativas

J.L. Garrido, M. Gea, M. Noguera, M. González, J.A. Ibáñez
Dpt. Lenguajes y Sistemas Informáticos - Universidad de Granada
E.T.S. Ingeniería Informática, C/ Saucedo Aranda s/n, 18071 Granada, España
(+34) 958 244153
jgarrido@ugr.es, mgea@ugr.es, manoguera@ugr.es, lupin3@fedro.ugr.es

## RESUMEN

Los avances tecnológicos están incrementando las posibilidades de trabajo en grupo, e incluso cambiando la forma en la cual éste se ha realizado tradicionalmente. Así, los usuarios forman parte de un entorno compartido donde sistemas de computación distribuida dan soporte y promueven una nueva forma de interacción (Persona-Ordenador-Persona). La importancia que está cobrando en diferentes ámbitos el estudio y desarrollo de sistemas para proporcionar soporte a grupos de trabajo, requiere de metodologías y procesos que faciliten su estudio y un desarrollo sistemático. Este artículo presenta una propuesta arquitectónica para el desarrollo de aplicaciones colaborativas bajo el marco metodológico que proporciona AMENITIES.

## Palabras Clave

Aplicaciones colaborativas, Metodologías, Modelos Arquitectónicos, UML, internet.

## 1. INTRODUCCIÓN

Las nuevas tecnologías introducen diferentes formas de entender el trabajo, incrementando la colaboración del grupo para alcanzar metas comunes. Tradicionalmente, el trabajo se organiza en equipos donde sus miembros interaccionan para lograr una mayor productividad y rendimiento [14]. Así, el usuario es parte de una comunidad conectada por medio de una red, ampliando los horizontes en investigación hacia la Interacción Persona-Ordenador-Persona con tecnología basada en sistemas distribuidos de computación [27]. El objetivo no es sólo la mejora de la comunicación, sino más bien implicarse en un nuevo entorno que se comparte con otras personas pudiendo llevar a cabo actividades conjuntas bajo el paradigma de denominado Trabajo Cooperativo Asistido por Computadora (CSCW) [17]. El software para sistemas CSCW se denomina groupware definiéndose como "*un sistema basado en computadoras que soporta grupos de personas implicadas en tareas (o metas) comunes y que proporciona un interfaz a un entorno compartido*" [7]. El desarrollo de aplicaciones groupware para soporte de grupos de trabajo es complicado, ya que para un correcto diseño [8], hay que tener en cuenta protocolos sociales, aspectos de la organización (leyes que impone, etc.), actividades (tareas) relacionadas que llevan a cabo los participantes, etc.


Interacción 2004, 3-7 mayo, 2004, Lleida (España).

La ingeniería de requisitos [20], la cual tradicionalmente ha formado parte de la ingeniería del software, se define como "*un proceso sistemático e iterativo de desarrollo de requisitos acerca de lo que se va a diseñar*". La naturaleza del problema debe comprenderse a través de un análisis pormenorizado contemplando todas sus características y peculiaridades. Para ello, el diseñador debe intervenir en todas las actividades relacionadas con requisitos (análisis del problema, viabilidad, modelado, etc.) de manera que pueda decidir qué necesidades han de satisfacerse.

Con carácter general, faltan propuestas metodológicas completas para abordar e integrar todas las fases de estudio, análisis y desarrollo de un sistema cooperativo. Este artículo pretende mostrar cómo una propuesta metodológica concreta para sistemas cooperativos, denominada AMENITIES [10], además de abordar los aspectos más importantes desde el punto de vista de la ingeniería de requisitos (elicitación, especificación y negociación de requisitos) en estos sistemas, también resuelve la parte de desarrollo del software. Para ello, proponemos un diseño arquitectónico para el desarrollo de aplicaciones colaborativas el cual se ha llevado a la práctica con éxito.

En primer lugar, se describen brevemente los trabajos que están relacionados con el que aquí presentamos. A continuación, se proporciona una descripción general de los principales fases y modelos implicados en la metodología AMENITIES con la intención mostrar de situar y relacionar la propuesta que presentamos con su contexto metodológico. Seguidamente describimos los principales objetivos y partes del modelo arquitectónico que se propone como punto de partida para el desarrollo del software. Posteriormente, a través de un caso práctico (diseño de una agenda colaborativa) se describe en detalle la propuesta. Finalmente, se resumen las conclusiones.

## 2. TRABAJOS RELACIONADOS

Se han propuesto varios métodos y modelos para la especificación de sistemas cooperativos. Hasta la fecha, el estudio y desarrollo de estos sistemas se ha llevado a cabo extendiendo métodos y técnicas que tradicionalmente se han aplicado a sistemas interactivos. Las diferentes propuestas existentes pueden clasificarse en *modelos arquitectónicos, modelos basados en tareas* y procesos. En el enfoque arquitectónico, el sistema se modela como un conjunto de componentes y de relaciones entre éstos. Algunos ejemplos de tales modelos son: Interactor Models [16], PAC [5], y MVC [18]. En las extensiones propuestas para sistemas CSCW podemos incluir: PAC* [2] y nuevos modelos como la Arqutectura Dewan. Estas especificaciones del sistema

permiten dividir el sistema en componentes cuya funcionalidad se puede definir y traducir a una implementación utilizando tecnologías de programación orientadas a objetos. Un inconveniente de estos modelos es que están centrados en el diseño del sistema interactivo, y por lo tanto, existe una perdida de abstracción e independencia con respectos a otros aspectos como colaboración, coordinación y comunicación.

Las propuestas basadas en tareas estudian el sistema desde el punto de vista del usuario, centrándose en el análisis y modelado de tareas[22]. En esta línea destacamos aportaciones tales como GTA (Group Task Analysis) [30] que propone el estudio de un sistema basado en una ontología para modelos de mundos de tareas, es decir, en un marco de trabajo comprendiendo participantes, artefactos y situaciones, e identificando las relaciones entre estos componentes. Otra propuesta, CTT (ConcurTaskTrees) [24] es una notación gráfica para especificar tareas individuales y cooperativas de forma jerárquica, la cual utiliza operadores temporales basados en CSP para establecer un orden en los pasos para llevar a cabo estas tareas. Sin embargo, estas técnicas generalmente no consideran cambios dinámicos en el dominio del problema, como por ejemplo, el cambio de responsabilidades de los usuarios. De hecho, para describir organizaciones sociales debería tenerse en cuenta la organización del grupo y su evolución [8,21]. Así, la mayoría de estas propuestas normalmente no integran bajo el mismo modelo de representación muchos de los conceptos relevantes del dominio del problema: grupo, actor, rol social, leyes que impone la organización en la cual está inmersa el grupo, etc. Por otra parte, las propuestas basada en procesos se centran en el modelo de negocio y en el proceso de transformación de la información/producto a través de diferentes canales de producción, siendo estos canales asociados a recursos (tanto humanos como materiales) [25]. En este sentido, se automatiza el trabajo (no existe lapsos ociosos sin trabajo) y por tanto, se puede analizar la productividad y rendimiento general en lugar de la individual. Por tanto, esta visión pierde por completo la naturaleza colaborativa del modelo ya que relega a segundo plano el factor humano (no es lo más importante, sino la eficiencia del proceso a llevar a cabo).

Además, en la mayoría de las propuestas realizadas hasta el momento se presentan de forma aislada ya que ni conectan fácilmente ni se integran bien con procesos, métodos, técnicas y notaciones dentro de la ingeniería del software [12]. La utilización de lenguajes como UML [23], pueden proporcionar claras ventajas y beneficios de cara a desarrollar aplicaciones colaborativas debido a que es considerado un estándar en el desarrollo del software.

## 3. AMENITIES

AMENITIES [10,11] es una metodología basada en modelos de comportamiento y tareas para el análisis, diseño y desarrollo de sistemas cooperativos.

La metodología parte de marcos teóricos cognitivos y metodológicos [1] permitiendo realizar un modelado conceptual del sistema cooperativo. Pretende abordar las carencias que presentan los modelos para análisis y modelado de tareas ya que se centra en el concepto de grupo, cubriendo aspectos relevantes de su comportamiento (dinámicas, evolución, etc.) y estructura (organización, leyes, etc.). El objetivo de la metodología es abordar de forma sistemática el análisis y diseño del sistema cooperativo facilitando el desarrollo posterior del software. AMENITIES proporciona un marco de referencia conceptual y metodológico, pero además propone una metodología concreta. Esta metodología se ha aplicado a diversas áreas como la gestión de recursos humanos en un sistema de control de emergencias[10] o a la representación de la colaboración para la gestión del conocimiento compartido [4].

En la Figura 1 se presenta el esquema general de la metodología AMENITIES. En ella se muestran los principales modelos implicados y las fases generales. Estas fases generales son las siguientes: (1) análisis del sistema y obtención de requisitos; (2) modelado del sistema cooperativo; (3) análisis del modelo cooperativo; (4) diseño del sistema, introduciendo nuevos elementos o cambios en el modelo cooperativo, probablemente como resultado del análisis anterior; y (5) desarrollo del sistema software.

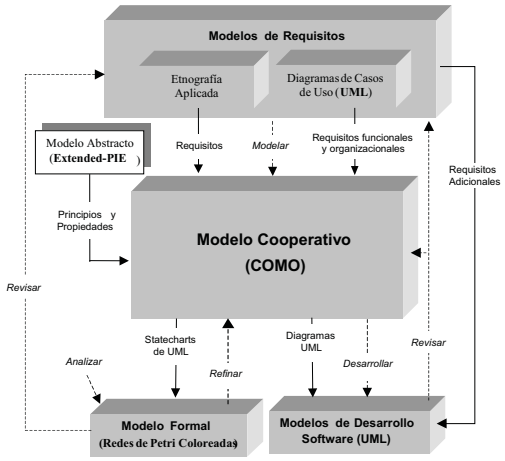


**Figura 1. Esquema general de la metodología AMENITIES**

Como en la mayoría de las metodologías, la aplicación de AMENITIES a un sistema sigue un proceso iterativo simple, permitiendo llevar a cabo un refinado del modelo como consecuencia del análisis de éste, así como una revisión de los requisitos de partida o del modelo cooperativo que podría aportar nueva o diferente información a considerar.

A continuación describimos los principales modelos que comprende la metodología y el objetivo específico de cada uno de ellos dentro de la metodología.

### 3.1 Modelos de Requisitos

El proceso de elicitación de requisitos se va a llevar a cabo mediante la técnica de la etnografía aplicada y los diagramas de casos de uso de UML. En primer lugar, la aplicación de la etnografía permite reunir y documentar informalmente los requisitos del sistema. A continuación, a partir de esta información obtenida, los diagramas de casos de uso se utilizarán

para estructurar, especificar los requisitos funcionales, roles y actores en el sistema.

### *3.1.1 Etnografía Aplicada*

Partiendo de las ideas generales de la etnografía [6], proponemos el uso de los estudios etnográficos en la metodología, ya que pueden aportar los requisitos que justamente suelen escapar a la aplicación de otros métodos estructurados más ampliamente utilizados. Esto es aún más evidente en el caso de los sistemas cooperativos, donde existe la necesidad de obtener requisitos en base a características y/o peculiaridades sociales y culturales dentro de los grupos.

El uso de la etnografía tiene como objetivo el proporcionar una mejor comprensión de las prácticas de trabajo y del comportamiento de los grupos que las llevan a cabo. Así por ejemplo, nos permite identificar:

- La estructura de los grupos en base a ciertos aspectos (comportamiento del participante, etc.).
- Cuestiones socioculturales que pueden afectar a la interacción y comunicación entre participantes ( uso de artefactos tales como videoconfencia por computadora, etc.).
- Requisitos funcionales relacionados con las prácticas de trabajo, recursos, etc.

### *3.1.2 Casos de Uso de UML*

Los casos de uso de UML [23] permiten representar requisitos funcionales acerca del comportamiento del sistema (de forma abstracta los objetivos de los usuarios). Un caso de uso es una unidad coherente de funcionalidad expresada como una transacción entre los usuarios y el sistema. Los casos de uso pueden describirse a diferentes niveles de detalle. Por lo tanto, pueden ser factorizados y descritos en términos de otros casos de uso más simples.

La metodología adopta una visión global del sistema cooperativo. De este modo, con los diagramas de casos de uso representamos toda la funcionalidad presente en el sistema cooperativo, es decir, tanto la que llevan a cabo los propios usuarios/participantes como la que han de proporcionar los subsistemas groupware. Esto implica que un caso de uso puede estar asociado con un usuario/participante por dos motivos:

- porque el usuario hace uso de la funcionalidad que representa el caso de uso, o
- por ser dicho participante el responsable de llevar a cabo esa funcionalidad.

Por lo tanto, en realidad lo que estamos haciendo en este caso es describir transacciones/interacciones funcionales entre usuarios, lo cual es el primer paso para identificar el aspecto de cooperación entre usuarios.

Los diagramas de casos de uso no describen en si mismos la organización pero proporcionan de forma abstracta información relevante acerca de los procesos a llevar a cabo. Estos diagramas nos van a permitir representar (mediante una notación semiformal) parte de la información recabada mediante la etnografía aplicada:

- La funcionalidad del sistema según se percibe por los usuarios, incluyendo relaciones básicas (generalización o especialización, extensión e inclusión) entre estas funciones.
- Identificación y clasificación de los actores que utilizan y/o realizan casos de uso, también incluyendo relaciones básicas (especialización o generalización) entre ellos si las hay.

El beneficio que aporta la utilización de los diagramas de casos de uso de UML lo resumimos en los siguientes puntos:

1. Define características estables o que se mantienen durante un largo periodo de tiempo.
2. Identifica y clasifica los roles que desempeñan los usuarios en el sistema. El concepto de rol es crucial a la hora de estructurar al grupo y especificar su comportamiento.

## 3.2 Modelo Cooperativo

El modelo cooperativo es el núcleo central de AMENITIES. Es un modelo conceptual formado por un conjunto de modelos de comportamiento y tareas organizados jerárquicamente, los cuales son de utilidad para describir y representar cualquier sistema cooperativo desde el punto de vista de su estructura y funcionamiento. Su principal propósito es permitir la negociación de requisitos elicitados en los modelos anteriores. Para ello, proporciona una especificación del sistema independiente de su implementación, proporcionando así una mejor comprensión del dominio del problema.

El método estructurado que se propone para la construcción del modelo cooperativo es el paso intermedio que permite integrar el análisis social con el diseño del sistema. La notación COMO-UML [11] (basada en el lenguaje UML) que proponemos para construir el modelo cooperativo es eminentemente gráfica y combina partes declarativas y operacionales.

## 3.3 Modelo Formal

Este modelo tiene como objetivo proporcionar una semántica formal operacional [9], tomando como base el formalismo de Redes de Petri Coloreadas (CPN), de la notación que se propone (COMO-UML) para construir el sistema cooperativo.

Una Red de Petri Coloreada permite validar requisitos generales (consistencia, completitud, etc.) y evaluar usabilidad del sistema (en términos de tareas) realizando ejecuciones automáticas y/o guiadas. Estas técnicas de simulación también pueden llevar a cabo análisis de rendimiento calculando productividad de transiciones, etc. Además, aplicando otras técnicas de análisis (grafos de alcanzabilidad, invariantes) es posible verificar propiedades de seguridad y vivacidad con la intención de complementar la simulación. En el dominio del problema que nos ocupa es posible, por ejemplo, demostrar que existen tareas que no pueden llevarse a cabo con los recursos humanos actualmente disponibles. Por lo tanto, el modelo permite analizar y negociar con stakeholders tanto requisitos funcionales con no funcionales.

# 4. PROPUESTA ARQUITECTÓNICA PARA EL DESARROLLO DEL SOFTWARE

El modelo cooperativo de AMENITIES está directamente relacionado con el desarrollo de aplicaciones software. EL diseño

arquitectónico constituye el punto de partida para el resto de modelos de desarrollo del software dentro de AMENITIES. Todos los puntos de vista que un modelo arquitectónico debería abordar, es decir, los componentes en que se divide el sistema, la funcionalidad de éstos, metainformación (roles sociales que desempeñan los usuarios, leyes que impone la organización, etc.), así como su comportamiento y despliegue, puede derivarse a partir del modelo cooperativo.

El modelo arquitectónico que proponemos persigue los siguientes objetivos generales cara al desarrollo de aplicaciones colaborativas:

- Simplificación del desarrollo del sistema dividiéndolo en partes más pequeñas (subsistemas).
- Cumplimiento de requisitos funcionales y de comportamiento mediante una asignación adecuada de servicios a subsistemas.
- Cumplimiento de ciertos requisitos generales relativos al desarrollo del software tales como reusabilidad, portabilidad, interoperatividad y mantenimiento.
- Facilidad de aplicación de nuevas tecnologías (Java, Objetos Distribuidos, servicios Web, etc.) haciendo uso de software específico ya desarrollado para implementaciones distribuidas (middlewares tales como Jini, JavaSpaces, CORBA, etc.).

Es importante en nuestro caso obtener la implementación del sistema a partir de un conjunto de subsistemas que se comunican a través de interfaces bien definidas, ya que una aplicación colaborativa es inherentemente distribuida. La arquitectura que proponemos en AMENITIES para el desarrollo de aplicaciones colaborativas se basa en la representación de los diferentes aspectos de dicha arquitectura, a saber, componentes, funcionalidad, comportamiento y despliegue de éstos, mediante vistas especificadas haciendo uso del lenguaje UML de la siguiente manera:

1. Para representar la descomposición de un sistema en subsistemas más pequeños se van a utilizar los *diagramas de componentes*, en concreto *diagramas de paquetes* con los estereotipos *sistema* y *subsistemas* junto con la relación de composición (una forma de la asociación de agregación que considera partes de un todo con tiempos de existencia ligados).
2. Representar la estructura funcional (vista estática) de los subsistemas dentro del diseño arquitectónico mediante *diagramas de clases e interfaces* asociados con subsistemas, definiendo conexiones entre ellos en base a relaciones de uso.
3. Describir el comportamiento (vista dinámica) de los subsistemas que colaboran en base a la estructura funcional descrita en el punto anterior utilizando *diagramas de colaboración*.
4. Y finalmente, describir la distribución de los subsistemas en los nodos del sistema distribuido mediante *diagramas de componentes y despliegue*.

En la siguiente sección se describe con detalle esta propuesta haciendo uso de una aplicación real ya desarrollada.

# 5. DISEÑO ARQUITECTÓNICO DE APLICACIONES COLABORATIVAS

## 5.1 Caso Práctico: Agenda Colaborativa

Para mostrar la propuesta arquitectónica anterior se ha escogido como caso práctico el desarrollo de una agenda colaborativa. Esta aplicación permite que profesores de un departamento de la Universidad de Granada colaboren en la propuesta de reuniones de trabajo. El proceso de colaboración se mejora gracias a la incorporación de mecanismos que refuerzan la conciencia de grupo (*group awareness* [28]) proporcionando un entorno compartido a los usuarios de la aplicación. Un requisito adicional para esta aplicación es permitir de forma alternativa interacciones entre participantes en modos de funcionamiento asíncrono y en tiempo real.

La aplicación consta de (véase Figura 2):

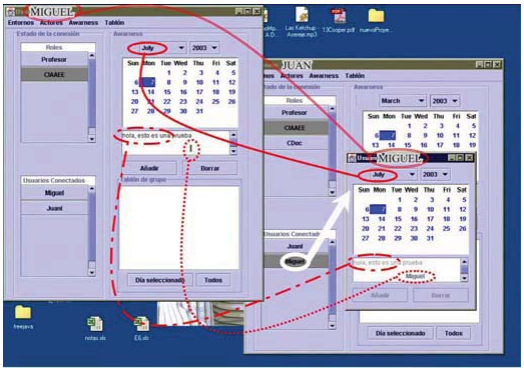

**Figura 2. Vista general de la aplicación de agenda colaborativa**

- Un panel que muestra los posibles roles que un usuario puede desempeñar y el rol que actualmente tiene asumido dicho usuario en el sistema.
- Un panel con un listado de otros participantes que actualmente están desempeñando el mismo rol.
- Un panel que contiene el calendario y mensajes enviados.
- Una ventana de para mostrar las actividades de otro usuario con el que deseamos colaborar en tiempo real en la realización de propuestas de reuniones.

## 5.2 Diseño Arquitectónico

Según los objetivos generales y las vistas descritas para la arquitectura que se propone (véase Sección 4), en las siguientes subsecciones se describen éstas vistas con detalle para el caso práctico que nos ocupa.

### *5.2.1 Diagrama de Componentes*

El sistema se divide en partes con la finalidad de simplificar el desarrollo. Para una aplicación colaborativa básica el sistema se

divide en cuatro subsistemas: *Identificación*, *Metainformación*, *Awareness* y la propia aplicación (*Agenda*).

La Figura 3 nuestra el diagrama de componentes resultante. Para dicho diagrama se ha utilizado la notación de diagramas de paquetes de UML con los estereotipos *system* y *subsystem*.

Los subsistemas *Identificación*, *Metainformación* y *Awareness* están siempre presentes para cualquier aplicacición colaborativa. En cambio, el subsistema de la propia aplicación, en este caso *Agenda*, es específico de cada aplicación concreta. Todos estos subsistemas se describen con más detalle en las siguientes subsecciones según la funcionalidad que proporcionan y el comportamiento que exhiben en su conjunto.

### *5.2.2 Diagrama Funcional*

Cada subsistema proporciona uno más interfaces con el objetivo de ser lo más independiente posible del resto de subsistemas. La Figura 4 muestra, utilizando diagramas de interfaces UML, los cuatro subsistemas y sus relaciones de uso entre ellos en base a la funcionalidad asignada a cada uno.

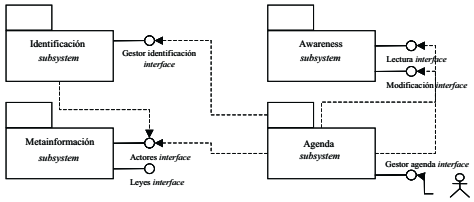


**Figura 4. Diagrama de Interfaces UML (vista funcional)**

El subsistema *Identificación* se utiliza para controlar el acceso de los usuarios al sistema y proporciona un estado de sesión inicial para cada usuario lanzando la infraestructura necesaria para el acceso a la aplicación. El subsistema *Metainformación* soporta toda la funcionalidad relativa a la gestión de metadatos (roles sociales, leyes de la organización, etc.); toda esta información está especificada en el modelo cooperativo AMENITIES para cada sistema (véase Figura 1). El subsistema *Awareness* proporciona la conciencia de grupo que los participantes necesitan para una colaboración efectiva mediante mecanismos tales como telepunteros, listas de participantes que actualmente desempeñan

**Tabla 1. Resumen de interfaces de subsistemas**

| SUBSYSTEMA | OPERACIÓN | ELEMENTO |
|---|---|---|
| **Identificación** | Identify | *Actor* |
| **Awareness (interfaces lectura y modificación)** | Update | *Datos activos del usuarios* |
| | Register | *Presencia del usuario* |
| | Be interested in | *Usuarios bajo un rol dado* |
| | Be interested in | *Estado del usuario* |
| | New coordinates | *Telepuntero* |
| | New generic event | *Evento genérico* |
| | Not interested in | *Usuarios bajo un rol dado* |
| | Not interested in | *Estado del usuario* |
| | Exit system | |
| | Choose | *Entorno distribuido* |
| **Metainformación (interfaces actores y leyes)** | Add | *Rol* |
| | | *Actor* |
| | | *Ley* |
| | Remove | *Rol* |
| | | *Actor* |
| | | *Ley* |
| | Select | *Rol* |
| | | *Actor* |
| | | *Ley* |
| | Get | *Capacidades* |
| | | *Clave* |
| | | *Nombre actor* |
| | | *Roles del Actor* |
| | | *Tareas del Actor* |
| | Update | *Datos del actor* |
| | Check | *Capacidad del actor* |
| **Agenda** | Delete | *Mensaje* |
| | Load | *Tablón de avisos* |
| | Insert | *Mensaje* |
| | Close | *Tablón de avisos* |

un rol concreto, etc. Por ultimo, el subsistema *Agenda* proporciona la misma funcionalidad que correspondería a una aplicación de agenda monousuario.

La Tabla 1 presenta un resumen de los servicios que proporciona cada subsistema por medio de las operaciones que especifican los interfaces y los elementos sobre los que actúan.

### *5.2.3 Diagrama de Comportamiento*

Tomando como punto de partida la especificación funcional de la Figura 4, se especifica el comportamiento del sistema mediante un diagrama de colaboración de UML tal como se muestra en la Figura 5. Dicho diagrama especifica como colaboran los subsistemas para resolver las interacciones entre usuarios y la aplicación, y entre los propios usuarios.

Una interacción de usuario típica comienza a través del subsistema *Agenda*. En primer lugar, el usuario se identifica en el sistema (mensajes 1, 1.1, 1.2). Una vez que la identificación se completa, el usuario puede escoger entre:

- registrar su presencia en el sistema de tal manera que otros usuarios tengan conocimiento de este hecho y puedan decidir observar las interacciones de dicho usuario y el sistema (2), y/o
- cargar el tablón de avisos con el objetivo de leer las últimas propuestas enviadas por otros miembros del grupo (3, 3.d).

De este modo el sistema proporciona la infraestructura necesaria para la conciencia de grupo posibilitando la colaboración asíncrona y en tiempo real de los miembros del grupo (3.a, 3.b, 3.c).

### *5.2.4 Diagrama de Despliegue*

Como consecuencia de la implementación llevada a cabo para esta aplicación de agenda colaborativa, la cual ha utilizado el modelo de coordinación Linda [3] mediante su implementación orientada a objetos JavaSpaces [29], se introduce un nivel de abstracción adicional que permite ocultar detalles simplificando la utilización de los diagramas de despliegue de UML. La Figura 6 muestra dicha simplificación.

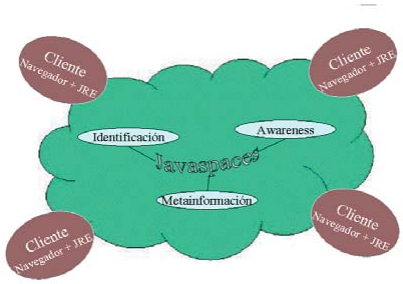


**Figure 6. Configuración del sistema (vista abstracta del despliegue)**

Existe un repositorio distribuido disponible en JavaSpaces, donde el sistema puede insertar y recuperar entradas con la metainformación requerida. Así, no hay necesidad de un repositorio en un nodo concreto del sistema distribuido. El repositorio puede estar distribuido en diferentes máquinas de manera que la información y/o servicios puedan asignarse para mejorar globalmente el rendimiento del sistema dependiendo de las características de uso. También diferentes clientes y servicios pueden asignarse al mismo o a diferentes nodos de la red. Los clientes, cada uno de ellos es una replica del subsistema *Agenda*, interaccionan entre ellos indirectamente a través del subsistema *Awareness*, el cual desempeña una funcionalidad que corresponde a un mecanismo de notificación distribuida de eventos.

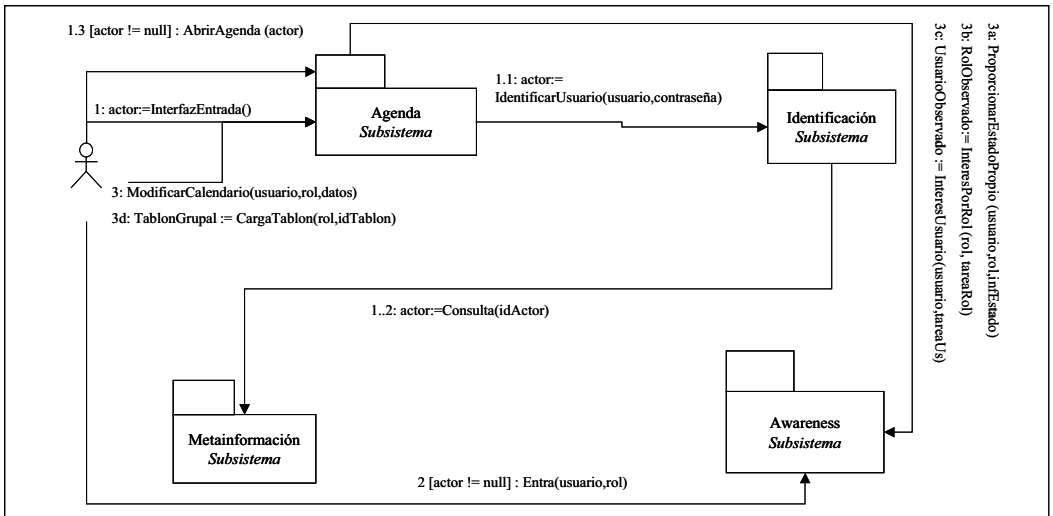


**Figura 5. Diagrama de colaboración de UML (vista dinámica)**

La popularidad del lenguaje Java junto con sus características (multiplataforma, comunicación/sincronización a través de redes, applets, etc.) ha facilitado y permitido instalar, configurar y ejecutar el sistema en internet. El único requisito es la disponibilidad de un navegador con soporte para ejecutar applets (Java Runtime Environment) para cada usuario (cliente) y un servidor web estándar.

## 6. CONCLUSIONES

La metodología AMENITIES aborda actividades en el campo de la ingeniería del software. Así, permite llevar a cabo la elicitación de requisitos, el modelado del sistema y la negociación de dichos requisitos en sistemas basados en computadoras. Además, la propuesta arquitectónica para el desarrollo del software que hemos descrito en este trabajo se realiza bajo el mismo marco metodológico que proporciona AMENITIES conectando con el resto de modelos y sirviendo como punto de partida para el resto de fases relacionadas con el desarrollo del software.

Con el diseño arquitectónico propuesto es posible incrementar la independencia, modularidad, integración y reusabilidad del software. También permite proporcionar una plataforma específica de servicios de alto nivel para la gestión de metainformación (registro de información acerca de usuarios, uso de mecanismos para reforzar la conciencia del grupo, etc.) en esta clase de sistemas. La plataforma se ha construido sobre el middleware JavaSpaces (una implementación del modelo de coordinación Linda) el cual ha sido utilizado como un mecanismo de notificación distribuida de eventos (entrada de usuarios en el sistema, etc.) relacionados con metadatos. Como middleware de coordinación, hace transparente el aspecto de distribución durante todas las fases de desarrollo de las aplicaciones colaborativas.

La comunidad CSCW ha desarrollado en los últimos años varias versiones de sistemas colaborativos experimentales. La mayoría de éstos se han desarrollado específicamente para aplicaciones concretas, lo cual requiere un gran esfuerzo en las implementaciones de otros nuevos desarrollos. En cambio, otros sistemas desarrollados han identificado servicios básicos proporcionando herramientas (toolkits) para construir estas aplicaciones. Por ejemplo, Groupkit [26] proporciona una biblioteca de componentes para construir interfaces multiusuario. Los principales problemas de estas propuestas son:

- los interfaces no pueden interoperar entre ellos, y
- es difícil adaptar los interfaces a las necesidades particulares de cada usuario.

También se han propuesto arquitecturas tales como COCA [19] y Clock [13] ideadas expresamente para aplicaciones colaborativas. El principal problema de estas arquitecturas es que carecen de un marco metodológico y conceptual donde existan, previos al modelo arquitectónico el cual pertenece al dominio de la solución, modelos de especificación y modelado en el dominio del problema.

La propuesta arquitectónica que hemos presentado evita estos problemas ya que parte desde una metodología con conexiones entre los modelos que comprende, y permite un desarrollo como plataforma de servicios (telepunteros, ventanas compartidas, etc.) a un nivel diferente pero que puede integrarse fácilmente con herramientas estándares de desarrollo de interfaces. La plataforma puede extenderse con nuevos servicios cuando éstos se requieran sin necesidad de realizar modificaciones y cambios en los servicios existentes. Además, existen otras ventajas de la arquitectura propuesta que han surgido de la experiencia adquirida en el desarrollo de casos prácticos. Por ejemplo, existen diferentes alternativas reales de implementation siendo muchas de ellas interesantes desde el punto de vista de la abstracción que proporcionan, promoviendo un desarrollo simple y rápido. Adicionalmente, ha sido directo llevar el sistema desarrollado a Internet haciendo uso de tecnologías estándares (lenguaje y utilidades Java, navegadores y servidores web, y applets).

## REFERENCIAS


[1] Cañas, J.J., Waern, Y.: *Ergonomía cognitiva.* Ed. Panamerica (2001)

[2] Calvary, G., Coutaz, J., Nigay, L.: *From Single-User Architectural Design to PAC*: a Generic Software Architecture Model for CSCW*. Proceedings CHI'97. ACM Press (1997) 242-249

[3] Carriero, N., Gelernter, D.: *Linda in context*. Communication ACM 32 (April 1989) 444-458

[4] Cobos, R., Alaman, X., Gea, M., Gutierrez, F.L., Garrido, J.L.: *Modelado del sistema para trabajo colaborativo KnowCat mediante AMENITIES.* Actas del IV Congreso Internacional Interacción Persona-Ordenador 2003 (Interacción'03), Vigo, Spain (Mayo 2003)

[5] Coutaz, J.: *PAC-ing the architecture of your user interface*. In Proceedings of the Fourth Eurographics Workshop on Design, Specification and Verification of Interactive Systems (DSVIS'97). Sringer-Verlag (1997) 15-32

[6] Ehrlich, K.: *Designing Groupware Applications: A Work-Centered Design Approach*. In: Beaudouin-Lafon, M. (ed.): Computer Supported Cooperative Work. Wiley (1999) 1-28

[7] Ellis, C.A., Gibbs, S.J., Rein, G.L.: *Groupware: some issues and experiences*. Communications of the ACM, Vol. 34,No. 1 (January 1991) 38-58

[8] Garrido, J.L., Gea, M.: *Modelling Dynamic Group Behaviours*. In: Johnson, C. (ed.): Interactive Systems - Design, Specification and Verification. Revised papers. LNCS 2220. Springer (2001) 128-143

[9] Garrido, J.L., Gea, M.: *A Coloured Petri Net Formalisation for a UML-Based Notation Applied to Cooperative System Modelling*. In: Forbrig, P. et all (ed.): Interactive Systems - Design, Specification and Verification. Revised papers. LNCS 2545. Springer (2002) 16-28

[10] Garrido, J.L., Gea, M., Padilla, N., Cañas, J.J., Waern, Y.: *AMENITIES: Modelado de Entornos Cooperativos*. In: Aedo, I., Díaz, P., Fernández, C. (eds.):Actas del III Congreso Internacional Interacción Persona-Ordenador 2002 (Interacción'02), Madrid, Spain (Mayo 2002) 97-10

[11] Garrido, J.L. *AMENITIES: una metodología para el desarrollo de sistemas cooperativos basada en modelos de comportamiento y tarea.* Tesis Doctoral. Universidad de Granada, 2003

[12] Gea, M., Gutiérrez, F.L., Garrido, J.L., Cañas J.J.: Teorías y Modelos Conceptuales para un Diseño basado en Grupos. Actas del IV Congreso Internacional Interacción Persona-Ordenador 2003 (Interacción'03), Vigo, Spain (Mayo 2003)

[13] Graham, T.C. Nicholas and Urnes, Tore: *Integrating Support for Temporal Media Into an Architecture for Graphical User Interfaces*. In Proceedings of the International Conference on Software Engineering (ICSE'97) , ACM Press, Boston, USA, pp. 172-183 (May 1997)

[14] Greif, I. (Editor): C*omputerSupported Cooperative Work: A Book of Readings*, Morgan Kaufmann Publishers, San Mateo (1988)

[15] Grudin, J.: *Groupware and Cooperative Work: Problems and Prospects*. Reprinted in Baecker, R.M. (ed.) Readings in Groupware and Computer Supported Cooperative Work, San Mateo, CA, Morgan Kaufman Publishers (1993) 97-105

[16] Harrison, M., Thimbleby, H. (eds.): *Formal Methods in Human-Computer Interaction*. Cambridge University Press (1990)

[17] Jordan, B.: *Ethnographic Workplace Studies and CSCW*. In: Shapiro, D., Tauber, M.J., Traunmueller, R. (eds.): The Design of Computer Supported Cooperative Work and Groupware System. North-Holland, Amsterdam (1996) 17-42

[18] Krasner, G.E., Pope, S.T.: *A cookbook for using the Model-View-Controller user interface paradigm in Smalltalk-80*. Journal of Object-Oriented Programming, 1(3) (August-September 1988) 26-49

[19] Li, D., Muntz, R.: *COCA: Collaborative Objects Coordination Architecture*. In the Preceedings of ACM CSCW'98, Seattle (Nov. 1998)

[20] Macaulay L.A: *Requirements Engineering*. Springer (1996)

[21] McGrath, J.: *Time, Interaction and Performance: a theory of groups*. In: Readings in Groupware and Computer-Supported Cooperative Work. R. Baecker (ed). Morgan Kauffman (1993)

[22] Nardi, B., (ed): *Context and Consciousness: Activity Theory and Human-Computer Interaction,* Cambridge: MIT Press (1996)

[23] Object Management Group: *Unified Modelling Language Specification*. http://www.omg.org (September 2001)

[24] Paternò, F.: Model-based Design and Evaluation of Interactive Applications. Springer-Verlag (2000)

[25] Reijers, H.A.: Design and Control of Workflow Processes. Lectures Notes in Computer Science 2617, Springer, 2002

[26] Roseman, M. and Greenberg, S.: *Building Real Time Groupware with GroupKit, A Groupware Toolkit*. ACM Transactions on Computer Human Interaction 3(1), (March 1996)

[27] *Salvendy, G.: The Human Factors of the Information Society. In Proceeding of the 6th ERCIM Workshop of User Interfaces for All (2000)*

[28] Schlichter J., Koch M., Burger M.: *Workspace Awareness for Distributed Teams*. In: W. Conen, G. Neumann (eds.), Coordination Technology for Collaborative Applications", Springer Verlag, Heidelberg (1998)

[29] Sun Microsystems: *Jini Network Technology*. http://wwws.sun.com/software/jini/index.html

[30] van der Veer, G., Lenting, B., Bergevoet B.: *GTA: Groupware Task Analysis - Modelling Complexity*. Acta Psycologica, 91 (1996) 297-322