Desarrollo de sistemas interactivos en base a modelos de usuario

Francisco Luis Gutiérrez[1], Miguel Gea[1], José Luis Garrido[1], Nicolás Padilla[2]

[1] Dpto. Lenguajes y Sistemas Informáticos. Universidad de Granada
ETSI Informática, Av. Andalucía, 38. Granada. España
{fgutierr,mgea,jgarrido}@ugr.es

[2] Dpto. de Lenguajes y Computación. Universidad de Almería.
Ctra. Sacramento s/n. 04120 Almería.
Npadilla@ualm.es

Resumen

La complejidad inherente que poseen los sistemas interactivos actuales obliga a hacer un gran esfuerzo en el proceso de especificación para garantizar su corrección. Las fases iniciales del desarrollo de un producto tienen una gran importancia ya que los errores que no se detecten en estas fases pueden provocar grandes aumentos en el costos de desarrollo del software. Los métodos formales son una gran ayuda la hora de modelar, de forma precisa, tanto el comportamiento como la estructura del sistema. La constante evolución de los modelos sociales y de la organización de trabajo obligan a afrontar nuevos retos y estrategias en el desarrollo de sistemas interactivos. En esta dirección se observa una demanda creciente hacia sistemas basados en el trabajo en grupo. Es necesario encontrar métodos de desarrollo de software que tengan en cuenta las características particulares tanto de los sistemas interactivos como del contexto del trabajo. En este articulo vamos a presentar una metodología de desarrollo de sistemas interactivos basándonos en el modelo de los usuarios como centro del proceso.

Palabras Claves: Modelos de tareas, Modelos Arquitectónicos, Especificación formal. CSCW.

1. Introducción

Cuando nos enfrentamos a la especificación de un sistema interactivo, necesitamos realizar un gran esfuerzo en gran parte debido a la excesiva complejidad que poseen estos sistemas. Una de las características principales de este tipo de sistemas es el importante papel que en ellos desempeña el usuario. Si tenemos en cuenta el punto de vista del usuario, nos vemos obligados a considerar aspectos tan importantes como la usabilidad, el rendimiento, la adaptabilidad o el proceso de aprendizaje [1] durante las fases de análisis del sistema.

Todos estos factores hacen que la realización de una especificación completa del sistema pueda ser una labor extremadamente complicada y larga. La utilización de modelos formales permite abordar con cierta seguridad el proceso de especificación facilitando en gran medida la realización de actividades como el prototipado, la validación, la verificación de propiedades o el estudio de la usabilidad del sistema obtenido [2,3].

A la hora de utilizar técnicas de especificación formales es necesario partir de un modelo conceptual del sistema en el que se describan los elementos importantes que en él aparecen. Los modelos que tradicionalmente se han usado pueden

clasificarse en dos grandes grupos: **modelos arquitectónicos** y **modelos centrados en el usuario** (o modelos de tareas).

En las especificaciones basadas en modelos arquitectónicos el sistema se modela como un conjunto de elementos constructivos y de las relaciones entre ellos. Algunos ejemplos de estos modelos son: el modelo de los Interadores [4], el modelo Arch, PAC[5], o el MVC[6]. Este tipo de especificaciones permite descomponer el sistema en un pequeño conjunto de componentes de los que se que describen las funciones que pueden realizar. Usando estos modelos podemos realizar de manera sencilla una traducción a un diseño orientado a objetos. La mayor desventaja de este modelo es que la descripción se centra en el diseño del sistema con la perdida de abstracción e independencia que esto genera.

El segundo tipo de modelos se centra en la percepción que el usuario posee del sistema. La especificación se va a centrar en la representación de las distintas tareas que el usuario puede realizar, describiendo un conjunto de metas de usuario (*goals*) que se pueden descomponer en secuencias de tareas para alcanzar los objetivos deseados. Dentro de los distintos métodos que siguen esta aproximación podemos observar que cada uno refleja distintos aspectos del problema. Por ejemplo, CCT [7] describe el proceso de aprendizaje, mientras que TAG [8] se usa más para validar el diseño del sistema trabajando en términos de consistencia. Otras notaciones como GOMS y NGOMSL [9] se usan para medir el rendimiento del sistema y son buenas para expresar el conocimiento que posee el usuario del mismo.

En los sistemas cooperativos (CSCW) donde varios usuarios comparten el mismo escenario, estos modelos basados en tareas tienen que extenderse para incluir aspectos sociales y organizativos de las actividades que realizan los usuarios (GTA [10]). El objetivo de esta extensión permite analizar la participación de diferentes usuarios y las interrelaciones que van a aparecer entre ellos a la hora de poder realizar estas tareas. En un escenario cooperativo podemos encontrar elementos tales como objetos (compartidos), agentes, roles, tareas y relaciones (usa, responsable de, autor de..)[11].

Esta aproximación cognitiva ha sido ampliamente utilizada pero con algunas limitaciones debido sobre todo al nivel de abstracción que se usa. Las tareas se describen usando especificaciones semi-formales y el nivel de detalle utilizado varia de forma arbitraria de una especificación a otra.

Otro aspecto que no se tiene en cuenta en estas notaciones es que los distintos roles que pueden realizar los usuarios no tienen por que ser estáticos. Una aproximación centrada en el usuario debe tener en cuenta el comportamiento tanto actual como futuro de cualquier usuario. Un usuario puede cambiar sus objetivos o intenciones durante su ciclo de vida, de forma que el rol que representa puede cambiar de manera dinámica produciéndose un cambio a otra personalidad distinta de la actual. Por otro lado, todo esto debe poder describirse de una manera lo mas formal posible de cara a facilitar la verificación y validación del sistema.

2. Metodología de Desarrollo

Con este trabajo presentamos una metodología de desarrollo de sistemas interactivos que pretende abordar las fases mas importantes del desarrollo, prestando especial importancia a las fases de modelización del sistema y de validación de la Especificación. Estas dos fases son las más importantes para la obtención de un modelo que represente al sistema y que pueda ser utilizado como base para el resto del desarrollo del software. El esfuerzo que se realice en obtener un "buen modelo" del sistema se va a ver recompensado por la reducción, de errores, decisiones de diseño y mal entendidos que se van a producir en el resto del desarrollo.

En la figura 1 podemos ver un esquema en el que aparecen las principales actividades que se van a realizar durante el desarrollo de un Sistema Interactivo.

Como base para la representación del sistema vamos a utilizar dos modelos conceptuales, por un lado un "**Modelo Interactivo**" que va a representar todos los aspectos relacionados con la interacción persona-ordenador y por otro lado un "**Modelo Cooperativo**" que se va a encargar de la representación de los aspectos de interacción entre personas a la hora de modelar sistemas en los que aparece algún tipo de cooperación.

Durante la primera fase del desarrollo vamos a obtener los requisitos del sistema. Esto nos permitirá comprender mejor el funcionamiento del mismo. Para ello vamos a partir de tres fuentes de información:

a) **Descripción de propiedades**. Partimos de un modelo abstracto del sistema sobre el que vamos poder realizar una descripción y un estudio de propiedades relevantes que satisface el sistema,

sin entrar en detalles internos y de forma previa a cualquier decisión de diseño que se realice.

b) **Análisis del Usuario**. Los usuario juegan un papel muy importante a la hora de especificar un sistema. La forma de representar el comportamiento que va a poseer cada usuario va a ser mediante alguna de las técnicas existentes para el Análisis de Tareas (Nosotros en los ejemplos realizados utilizamos el método de análisis GOMS).

c) **Analisis de Grupos**. Cuando describimos sistemas en los que distintos usuario cooperan para realizar actividades, es muy importante extender los modelos tradicionales para permitir la descripción de elementos como tarea cooperativa, grupo, actor o rol.

La información que se va a representar en estos tres modelos va a permitir obtener un modelo formal del sistema que será utilizado como base para la realización de una posible propuesta de diseño.
El modelo Interactivo va a estar formado por un conjunto de tareas que el usuario puede realizar, junto con la descripción del escenario en el que se realizan las actividades asociadas a las tareas. La descripción del escenario estará formada por los distintos objetos que son utilizados por el sistema en cada una de las tareas.

Partiendo del modelo de tareas vamos a realizar una especificación formal del sistema en base a objetos. Para ello utilizaremos el lenguaje de especificación Algebraico GRALPLA[12,13] que nos va a permitir describir los elementos interactivos que aparecen en nuestro sistema . Utilizando esta especificación y usando las herramientas que se han desarrollado, podemos obtener un prototipo del sistema que pueda ser utilizado como instrumento de validación del sistema y así poder probar propiedades sobre la propia especificación [14].

El objetivo final del modelo cooperativo es la descripción de todos aquellos aspectos [15] acerca de cómo colaboran los usuarios para llevar a cabo trabajo en grupo (tareas cooperativas). Así, el modelo cooperativo proporciona un nivel de abstracción mayor sobre los modelos de roles y tareas del modelo interactivo, siendo capaz de modelar, además de tareas cooperativas, otros conceptos adicionales como grupo, restricciones de la organización, capacidades, etc.
La notación que se utiliza en el modelo cooperativo, es una extensión a *Unified Modelling Language* (UML) [16] (denominada Extended-UML) que hemos elaborado a raíz de la adaptación de UML al modelado de sistemas cooperativos. Extended-UML, al igual que el propio lenguaje UML, es una notación básicamente gráfica y combina partes declarativas y operacionales. Es posible validar y

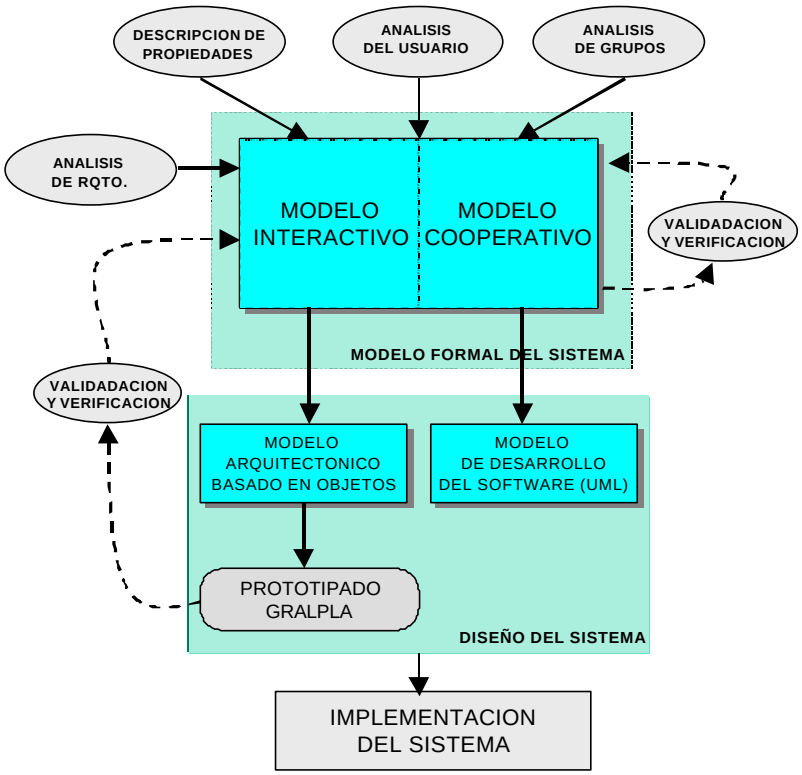


Figura 1. Etapas en el desarrollo de un Sistema Interactivo

verificar propiedades del modelo resultante bien directamente o traduciéndolo a Redes de Petri Coloreadas (CPN).

3. Descripción de propiedades. Modelo Abstracto.

El diseño de un sistema cooperativo es una tarea compleja por la variedad de matices y propiedades que debe preservar relativas a los usuarios (percepción del grupo), al sistema (sincronización de actividades) y a la propia organización del trabajo (definición de tareas). En este sentido, nos vamos a centrar en el estudio de este tipo de sistemas mediante la ayuda de un modelo formal abstracto que nos permita razonar sobre las propiedades del mismo como paso previo al proceso de diseño. Existe una larga tradición el área de HCI en el uso de métodos formales para el diseño de sistemas interactivos ya que permiten analizar propiedades que conforman los requisitos del sistema. Estos estudios formales se han extendido para el análisis de sistemas cooperativos centrándose principalmente en el diseño de tareas en grupo [17], modelos de coordinación [18], y en la conexión con arquitectura de diseño de software cooperativo. Sin embargo, estos modelos se centran en aspectos parciales del sistema (relaciones entre usuarios y conexión con diseño). Nuestra intención es el análisis de las propiedades inherentes de este tipo de sistemas, por lo que necesitamos un modelo abstracto que nos permita describir el comportamiento externo del sistema sin tener en cuenta su estructura interna. Para ello partimos de un modelo simple (el modelo PIE) de caja negra que describe un sistema interactivo en términos de posibles entradas del usuario y el efecto que éstas producen en el mismo, y vamos a proponer una extensión para recoger la noción de usuarios, tareas y entorno cooperativo.

En la figura 2 se muestra el modelo PIE-extendido, en el cual la entrada $\tilde{O} = (C \times U)^*$ es una secuencia de tuplas donde C representa el dominio de órdenes y U el dominio de usuarios, I es la función de interpretación de esas entradas en el dominio de efectos (E), y que a su vez se pueden descomponer en un resultado alcanzado en el sistema (R) y la observación de los efectos en el conjunto de displays $D= \{d_0, d_1,..,d_i\}$ donde el subíndice denota la relación entre el display y usuario.

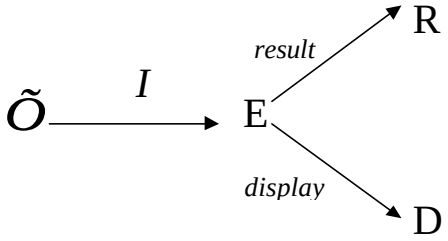


Figura 2. Modelo PIE extendido

En este modelo podemos recoger la noción de usuario (cada entrada que se produce en el sistema puede ser realizada por un usuario diferente) y la noción de entorno común compartido, ya que el estado que alcanza el sistema es único. Sin embargo podemos comprobar que cada usuario puede tener una visión diferente del sistema ya que cada usuario tiene asignado un modo de visualización diferente para la observación del efecto alcanzado. Con esta extensión, podemos considerar el modelo PIE como un caso particular restringido a un único usuario con un único display. Sobre la secuencia de entrada $\tilde{O}$ se define el operador de composición ($p_i, p_j \in \prod$) como la concatenación de tuplas en orden temporal ($p_i \cdot p_j \in \prod$).

3.1 Propiedades

La ventaja de los modelos abstractos es que permiten realizar un estudio de las propiedades relevantes al sistema sin entrar en detalles internos, observando el comportamiento del sistema desde una visión externa (modelo de caja negra). Esta percepción nos permite inferir propiedades relacionadas con el punto de vista de los participantes (acerca de cómo ve el usuario el sistema). Por ejemplo, una de las características que se han comentado de este tipo de sistemas es que varios participantes comparten un entorno común. Podríamos por tanto analizar la *visibilidad* que poseen unos participantes de otros, es decir, si cualquier usuario puede observar el efecto producido por otro participante. Esto lo podríamos enunciar formalmente del siguiente modo:

$$\text{display}_i\, I(p_o \cdot (c_m, u_j) \cdot p_k) \neq \text{display}_i\, I(p_o \cdot p_k),$$
$$\forall p_o, p_k \in \prod, \forall c_m \in C, \forall u_i, u_j \in U$$

Es decir, el efecto de las acciones que realiza otro participante son reflejadas y notificadas al resto de participantes. Por ejemplo, pueden actuar simultáneamente un decorador y un pintor para la recreación del piso piloto, y la mutua interacción mejora el resultado final.

Otra característica es la colaboración entre participantes. Podríamos caracterizar el proceso de colaboración en este tipo de sistemas si se satisface que existe un cierto objetivo ($r \in R$) y una secuencia de entrada ($\tau \in \prod$) tal que se verifica lo siguiente:

$$\exists r \in R \,/\, result\,(I(\tau)) = r \wedge$$
$$\tau = (p_o \cdot (c_m, u_i) \cdot .. \cdot (c_n, u_j) \cdot p_k\,),$$
$$\forall p_o, p_k \in \prod, \forall c_m, c_n \in C, u_i, u_j \in U$$

Para lograr un objetivo, existirá una secuencia de entrada en la cual interviene al menos dos participantes para su realización. Podemos considerar que un sistema es colaborativo si al menos posee una tarea que necesite de la participación de más de un usuario. Por ejemplo, el arquitecto modifica la posición de las cocinas a instancias del aparejador para mejorar la distribución general.

Otra propiedad relativa al modelo mental de cada usuario es la *predecibilidad*, es decir, el predecir el comportamiento futuro del sistema a partir del estado actual. Si bien esta propiedad es clara en un sistema interactivo, puede prestar a confusión en un sistema donde hay más de un participante realizando tareas sobre el sistema. Se puede ver desde dos puntos de vista complementarios: el usuario puede predecir el comportamiento futuro sin ser afectado por la participación de otros usuarios en el sistema, o bien, cualquier modificación que ocurra en el sistema será notificada al usuario para que tenga constancia del cambio. Esta situación se puede expresar del siguiente modo. Dado $r \in R$, representando el efecto de la tarea del usuario u_i, se debe satisfacer la siguiente relación:

$$\forall u_j \in U, c_k \in C$$
$$r = result(\, I\,(p_1 \cdot p_m)\,) = result\,(\, I\,(p_1 \cdot (c_k, u_j) \cdot p_m)\,)$$

Esto indica que el efecto de realizar la tarea por el usuario u_i no se ve afectada por el resto de participantes. En el segundo caso, la propiedad de visibilidad garantizaría la observación de las modificaciones del resto de usuarios sobre el sistema. Por ejemplo, un arquitecto puede modificar la ubicación de un tabique independientemente de la navegación de los usuarios, y por su parte, los usuario deberán ser notificados del cambio de estructura (ver la nueva colocación del tabique) para conocer y explorar el edificio.

Podemos ir más allá y estudiar incluso el comportamiento de los usuarios en el entorno. Podemos definir la *equivalencia de usuarios* si el efecto de las acciones que pueden llevar a cabo son iguales, y por tanto:

$$u_i \sim u_j \equiv \forall\,(c_k, u_i), \exists\,(c_q, u_j)\,/$$
$$I(s \cdot (c_k, u_i) \cdot r) = I(s \cdot (c_q, u_j) \cdot r) \qquad s, r \in \Pi$$

siendo c_k, c_q entradas realizadas por los usuarios u_i y u_j respectivamente. En el ejemplo, las acciones que pueden llevar a cabo por los clientes son las mismas, independientemente de su características y conocimientos personales (pueden ser a su vez arquitectos que quieren "comprar" un piso).

4. Descripción del Usuario. Análisis de Tareas

Los usuarios juegan un papel importante en la especificación del sistema ya que son los actores que aparecen en el escenario. El usuario tiene que comprender el sistema de forma correcta para poder utilizarlo de una manera adecuada. La especificación del sistema realizada desde el punto de vista del usuario tiene que tener en cuenta distintos aspectos:

- El proceso de aprendizaje: El usuario necesita aprender como puede usar el sistema. Este conocimiento puede expresarse usando métodos formales que permitan especificar las tareas del usuario. Estos métodos intentan buscar una respuesta a la pregunta "¿Cómo puedo hacerlo?"
- El comportamiento del usuario: Una vez que el usuario conoce el sistema puede explorar las posibilidades funcionales que le ofrece. En este caso estamos interesados en el comportamiento dinámico del sistema. Algunas veces aparecen cosas que no pueden realizarse debido a errores u omisiones cometidas durante la especificación. Estos métodos intentan buscar una respuesta a la pregunta "¿Puedo llegar a hacerlo?"
- Los usuarios juegan un papel determinado dentro del sistema. Cada uno de los usuarios sigue una serie de reglas para cada una de las posibles tareas a realizar. Este papel va a determinar su actividad en el sistema. Estos métodos intentan buscar una respuesta a la pregunta "¿Puedo/debo hacerlo?"

El análisis de tareas ayuda al diseñador a comprender y validar el sistema, pero además es muy útil para el usuario, ya que puede aprender qué actividades va a poder realizar y cómo puede llevarlas a cabo. Este tipo de información nos va a ayudar a explicar como hay que realizar ciertas tareas, pero no nos da información sobre la intención del usuario (permiso u obligación de realizar ciertas tareas). Esta información está mas relacionada con

los diferentes roles que el usuario puede tomar en el sistema que con las tareas que puede realizar.

La mayor parte de las técnicas de especificación asumen que existen ciertos tipos de usuarios, pero no se describe ningún tipo de información sobre ellos. Nosotros proponemos centrar la especificación en la descripciión del usuario. Esta alternativa trata de modelar (caracterizar) al usuario haciendo una taxonomía de usuarios mediante la inspección de su naturaleza (tipología), capacidad, conocimiento o habilidad. El término *estereotipo* se usa para definir un patrón de usuario, así, el sistema determina quién es el usuario y adaptará su interacción para aumentar la comunicación con el mismo. Nos vamos a centrar en el usuario y su participación en el sistema interactivo, teniendo en cuenta su relación con los objetos, roles y tareas que puede desempeñar.

4.1. Modelo de los Roles

Un rol es un patrón de comportamiento que podemos asignar a un usuario. Este patrón determina como puede actuar el usuario en el sistema y está formado por un conjunto de posibles tareas y de relaciones con otros roles. Estas relaciones van a reflejar los diferentes modelos de trabajo que el usuario puede realizar dentro del escenario.

Bajo este modelo de descripción deberemos tener en cuenta algunas consideraciones tales como que el **dominio de los roles** debe cubrir todas las posibilidades de actuación dentro del sistema. Así cualquier acción que el usuario quiera realizar estará incluida en alguno de los roles existentes. El **dominio de las tareas** debe estar relacionado con el dominio de los roles, una tarea que actúa sobre los objetos del sistema podría ser diferente en función del rol actual del usuario que la realiza.

El **dominio de los Objetos** describe el conjunto de elementos que comparten los usuarios en el escenario. La descripción del propio **sistema** consiste en la definición de las relaciones entre usuarios y las dependencias existentes entre los distintos dominios.

Un buen ejemplo de aproximación centrada en el usuario puede ser un juego de ajedrez distribuido sobre la Red Internet. En este sistema varios usuarios pueden acceder al servidor de ajedrez para jugar partidas o para observar el estado de otras partidas que se estén realizando en ese momento. Se pueden jugar múltiples juegos al mismo tiempo y cualquier participante puede retar a cualquier otro para la realización de una partida. Las partidas pueden jugarse de manera interactiva (on-line) o de forma asíncrona (off-line). Los posibles usuarios del sistema puede registrarse en él para poder participar en las partidas o para observar partidas ya comenzadas.

En este escenario se observan aspectos interesantes de los distintos usuarios respecto a los roles que realizan. Cada usuario puede ir cambiando el Rol que esta realizando de forma dinámica. Por ejemplo, un invitado al sistema puede ser un potencial usuario registrado del mismo y puede ser jugador de una partida y/o observador de otra. Podemos ver también ejemplos en los que una misma actividad es distinta en función del rol que tenga el usuario en el momento. Cuando un usuario actúa de observador la tarea de salir le permite dejar de mirar una partida determinada, mientras que si actúa de jugador la tarea salir indica que va a abandonar una partida.

4.2. Definición de los Roles.

La definición de cada uno de los roles que pueden aparecer en nuestro sistema implica la asignación de las distintas tareas a cada uno de ellos. La evolución del usuario hace que se tenga que realizar una nueva configuración de las tareas distinta a la que se obtiene en un modelo de tareas tradicional. Algunas de las actividades que puede realizar son innecesarias (por ejemplo, registrarse mas de una vez) y pueden ser necesarias nuevas tareas (por ejemplo, puede observar un juego, inspeccionar la historia de movimientos de todos los usuarios, ...) Este tipo de cambios se resumen en los siguientes tipos de tareas, que se utilizaran para describir cada uno de los Roles.

- **Nuevas Tareas**. El nuevo rol le da acceso al usuario para la realización de nuevas tareas que antes no podía realizar.
- **Tareas Redefinidas**. En el nuevo rol una tarea puede tener un significado distinto.
- **Tareas con acceso controlado**. Algunas actividades pueden necesitar algún tipo de condición adicional para poder ser realizadas bajo este rol.

Figura 3. Notación para definir un Rol.

La siguiente configuración (Fig 4) muestra la evolución que un usuario puede realizar en el sistema y las *restricciones de acceso* que aparecen para poder desempeñar un rol determinado.

Cualquier invitado no puede jugar una partida ya que no tiene privilegios suficientes para poder activar ese rol. En el siguiente diagrama podemos ver como podrían reflejarse estos posibles cambios de rol. En el se pueden ver las actividades y pasos que el usuario debe realizar para llegar a una meta determinada .

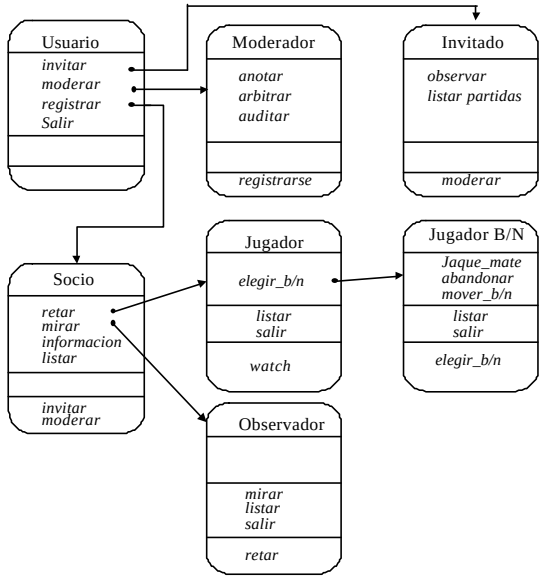


Figura 4. Diagrama Tareas-Roles

Cada rol va a representar un comportamiento que puede ser realizado por un usuario. Esto es similar a lo que ocurre en el teatro donde un actor interpreta a un personaje determinado. En cualquier momento, un usuario puede cambiar el rol que está realizando en ese momento. Esto es necesario ya que un usuario puede cambiar sus intenciones en el sistema y comportarse como otra "persona". El cambio de roles puede deberse a un proceso de aprendizaje (desde usuario novel a usuario experto), a una concepción mental (desde pasivo a activo) o a un cambio de objetivos (desde observador a actor).

4.3. Definición del Escenario

El usuario es una de las partes mas importantes del sistema pero en el caso de sistemas complejos existen toda una serie de elementos que son usados por el usuario para realizar las distintas tareas que deberían ser descritos. En el ejemplo que estamos siguiendo vemos que aparecen objetos como el tablero, en un juego determinado o el marcador utilizado por el moderador de la partida.

La estructura de los distintos objetos se puede describir usando una especificación algebraica. De esta forma representamos las estructuras de datos necesarias formalizadas mediante tipos de datos abstractos. Las posibles funciones que van a aparecer en los distintos objetos se corresponden con las actividades que pueden aparecer en las descripciones de las tareas del usuario.

Para obtener una descripción completa del sistema necesitamos describir las relaciones entre tareas, roles y objetos. En este punto tenemos que decidir para cada tarea cuáles son los objetos que se usan, quienes son los participantes que realizan las tareas así como una descripción detallada de la propia tarea. La descripción de las tareas se puede hacer con cualquiera de las técnicas de análisis de tareas existentes. En este ejemplo se muestra parte de la descripción del movimiento de una pieza de ajedrez mediante GOMS:

GOAL: MOVER-B/N
. GOAL: DETERMINAR-TURNO
. . [select GOAL: CONOCER-ULTIMO-MOVIMIENTO
. . . MOVER-A-LA-HISTORIA-DE-MOVIMIENTOS
. . . DETERMINAR-ULTIMO-MOVIMIENTO
. . . COMPROBAR-SI-NO-ERES-TU
. . GOAL: CONOCER-MOVIMIENTO-SIGUIENTE
. . . MOVERSE-AL-TABLERO
. . . IDENTIFICAR-POSICION-DEL-RELOJ
. . . COMPROBAR-SI -RELOJ-ESTA-EN-TU-POSICION]
. GOAL: ELEGIR -MEJOR-ESTRATEGIA
. GOAL: REALIZAR -MOVIMIENTO
. . GOAL: SELECCIONAR -PIEZA-ADECUADA
. . . [select GOAL: IDENTIFICAR -PIEZA
. . . . SELECCIONAR-TECLADO
. . . . ESCRIBIR-IDENTIFICACION-PIEZA
. . . . CONFIRMAR
. . . GOAL: COGER-PIEZA
. . . . MOVER-CURSOR-A-PIEZA
. . . . PULSAR-BOTON-RATON]
. . GOAL: ELEGIR-DESTINO
. . . [select GOAL: IDENTIFICAR-DESTINO
. . . . MOVER-CURSO-ARRASTRANDO-PIEZA
. . . . ESCRIBIR-IDENTIFICACION-POSICION
. . . . CONFIRMAR
. . . GOAL: SOLTAR-PIEZA
. . . . MOVER-CURSOR-ARRASTRANDO-PIEZA
. . . . SOLTAR-BOTON-RATON]
. GOAL: CONFIRMAR-MOVIMIENTO
. . . [select GOAL: TECLA-CONFIRMACION
. . . . PULSAR-ENTER
. . . GOAL: PARAR-RELOJ
. . . . MOVER-CURSOR-RELOJ
. . . . PULSAR-BOTON-RATON]
Selection Rule for GOAL: DETERMINAR-TURNO
Si es una visualización gráfica, usar el método CONOCER-MOVIMIENTO-SIGUIENTE en otro caso usar el CONOCER-ULTIMO-MOVIMIENTO
Selection Rule for GOAL: SELECCIONAR-PIEZA-APROPIADA
Si no tienes ratón usar el método IDENTIFICAR-PIEZA, en otro caso usar el método COGER-PIEZA
Selection Rule for GOAL: ELEGIR-DESTINO
Si no tienes ratón usar el método IDENTIFICAR-DESTINO, en otro caso usar SOLTAR-PIEZA
Selection Rule for GOAL: CONFIRMAR-MOVIMIENTO

Si estas usando el teclado usar el método TECLA-CONFIRMACION, en otro caso usar el método PARAR-RELOJ

4.4. Descripción de los Roles bajo un Modelo Arquitectónico.

Utilizando la información representada en el modelo anterior podemos realizar una traducción de cada uno de los roles a objetos usando especificación Algebraica. Como ejemplo veamos la descripción del objeto que encapsula el comportamiento del Rol Jugador.

```
object Jugador: Socio;
  import JuegoAjedrez, Color, ListaDeMovimientos;
  Jugador: JuegoAjedrez, Socio -> Jugador;
Functions
 Elegir_b/n: Jugador, Color  -> Jugador_B/N;
 Salir:       Jugador -> Socio;
 Listar:      Jugador -> ListaDeMovimientos;
axioms
 var p: Jugador; g: JuegoAjedrez; c: Color; m: Socio;
     b: JuegoAjedrez;
 listar(Jugador(g,m)) = listar(g);
 Salir(Jugador(g,m) = m;
 Elegir_b/n(Jugador(g,m),c) = if(NumJugadores(g)=0)
    Jugador_B/N(Elegir_color(AñadirJugador(g,Jugador(g,m)),c),
      Jugador(g,m))
  else
    Jugador_B/N(Elegir_color(AñadirJugador(g,Jugador(g,m)),
     color_opuesto(g)), Jugador(g,m));
synchronisation
       do not mirar(p,g);
       do not retar(p,g);
```

La información necesaria para realizar esta especificación se obtiene de la especificación anterior. Algunos de los elementos expresivos que contempla GRALPLA permiten especificar mecanismos como la evolución de los roles (herencia entre especificaciones), la compartición de objetos (importación), etc. Los axiomas pueden obtenerse directamente a partir de la descripción de tareas.

De esta forma se realiza un posible diseño arquitectónico del sistema en el que se traducen a objetos tanto los roles como los elementos que describen el escenario en el que se realizan las tareas.

En nuestra aproximación hemos utilizado GRALPLA como un lenguaje formal con un nivel de abstracción alto permitiéndonos realizar especificaciones sin tener que realizar demasiadas decisiones de diseño. Una característica importante del lenguaje es que permite la traducción automática de una especificación a un lenguaje orientado a objetos. Esta herramienta nos permite obtener de manera automatica, un prototipo del sistema a partir de la propia especificación. Este prototipo se puede utilizar para la realización de validaciones y verificaciones formales de propiedades importantes del Sistema [14].

5. Modelo Cooperativo.

Esta sección presenta un modelo para describir organizaciones formadas por grupos de personas. La principal característica de estos sistemas es los cambios de comportamiento que se producen como consecuencia de las interacciones entre los miembros del grupo. Así, el grupo se describe a través de su comportamiento dinámico resultado de la interacción que tiene lugar cuando actores interpretan roles. Por tanto, la descripción es en términos de roles y sus relaciones que vienen impuestos por restricciones de la organización, actividades del grupo orientadas a la colaboración, y dinámicas de grupos gobernadas mediante coordinación.

Para realizar la especificación del modelo cooperativo se sigue un método [20] que consiste en describir, en una serie de pasos, diferentes niveles del dominio del problema en base a identificar y separar conceptos tales como grupo, rol, tarea cooperativa, etc, pero manteniendo y describiendo sus relaciones. Como notación se propone una extensión a UML denominada Extended-UML, capaz de modelar las bases y peculiaridades que conforman los grupos.

5.1. Notación

UML proporciona un enfoque técnico al desarrollo de sistemas. Sin embargo, nosotros estamos interesados en modelar la parte de interacción implicada en el comportamiento de grupos de trabajo. Así pues, la notación Extended-UML se ha desarrollado en base a:

1. Utilizar la parte más abstracta de UML (diagramas de actividades y estados), que no cubren aspectos estructurales, para modelar los conceptos presentes en nuestro marco de trabajo.
2. Combinar estos diagramas con cierta notación de texto para simplificar la especificación e incrementar su expresividad.

Los diagramas de estados de UML se utilizan para modelar como un actor puede dinámicamente cambiar su comportamiento e influir en el comportamiento del grupo. Por otro lado, los diagramas de actividades de UML se utilizan para describir como un grupo se coordina para llevar a cabo tareas cooperativas. Los dos tipos de diagramas

se combinan anidando los últimos en los primeros. Esta combinación no plantea problemas de integración ya que los diagramas de actividades son un caso especial de los diagramas de estados.

5.2. Proceso de Modelado Cooperativo

Como ejemplo se ha escogido un caso extremo, en el sentido de que no incluye sistema informático alguno. Con ello, se pretende mostrar el poder de modelado de la notación. El ejemplo en cuestión presenta los miembros de la plantilla de profesores/investigadores de una Facultad. Este escenario presenta una serie de participantes, cada uno tiene asignado un rol (*Lecturer*, *Director*, *Director Assistant* o *Secretary*), que a su vez tiene asignado un modelo de tareas. Varias tareas cooperativas pueden ser contempladas como por ejemplo celebrar una junta de facultad (*AttendFacultyAssembly*). Además, un participante puede interpretar un rol diferente en el futuro o incluso interpretar dos roles diferentes de manera alternativa. Por ejemplo, un profesor/investigador elegido como director de la facultad puede interpretar ambos roles pero en diferentes momentos, observe que el director no deja de ser profesor/investigador. También, se tienen que preservar ciertos protocolos y políticas impuestos por la organización, por ejemplo, la obligación de asistir a una junta de facultad.

5.3. Evolución del Comportamiento

El conjunto de posibles comportamientos de un grupo de actores se modela con un diagrama de estados (Figura 5) que resulta de conectar roles por medio de transiciones. Cada estado se corresponde a un rol (p.e. *Lecturer*) y los cambios dinámicos entre dos roles se representan por una transición y su correspondiente capacidad y/o restricción que aparece descrita como una condición de guarda (p.e. *Director*). Así pues, un actor interpretando el rol *Lecturer* puede interpretar el de *Director* si satisface la capacidad con el mismo nombre. La ocurrencia de eventos o la realización de actividades puede habilitar o deshabilitar capacidades/restricciones, determinando éstas el conjunto de posibles comportamientos que un actor puede llevar a cabo en un instante dado.

5.4. Especificación de Roles

La Figura 6 muestra la especificación de dos roles. En cada uno se identifican tareas cooperativas que son responsabilidad de dicho rol o que opcionalmente puede llevar a cabo, junto con:

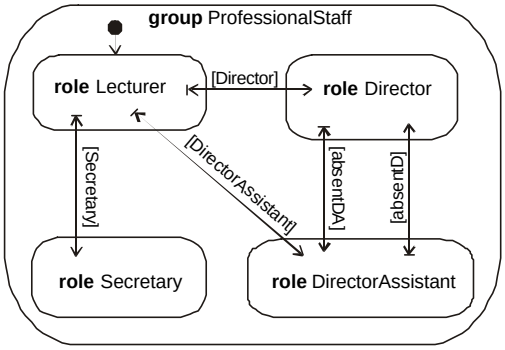


Figura 5. Diagrama de estados.

- eventos que disparan su ejecución, por ejemplo es la hora para celebrar una junta de facultad (*TimeFA* para la tarea *AttendFacultyAssembly*),
- capacidades/restricciones que habilitan su ejecución, por ejemplo, un profesor/investigador (rol *Lecturer*) tiene que ser miembro electo de la junta de facultad (capacidad *FAM*), y
- otras tareas que pueden interrumpir su ejecución, por ejemplo la asistencia a una junta de facultad puede ser interrrumpida por la tarea correspondiente a impartir un clase (*GiveALecture*). Por defecto, una tarea no puede ser interrrumpida.

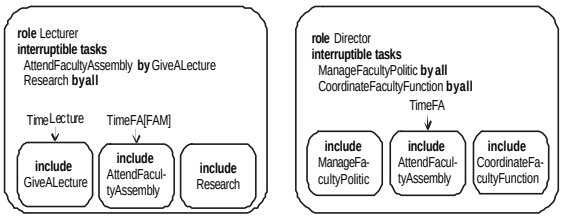
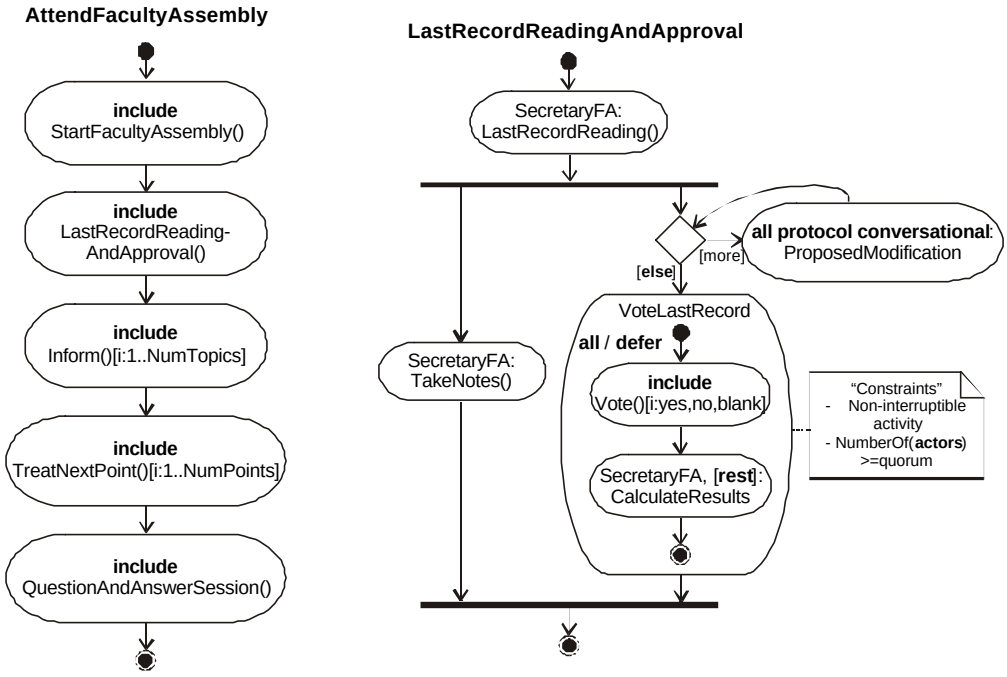


Figura 6. Especificación de dos Roles.

Comparando la especificación de estos dos roles se observa que comparten una misma tarea (*AttendFacultyAssembly*), lo cual claramente revela que es una tarea cooperativa. De esta forma es posible especificar diferentes eventos para comenzar su realización además de diferentes capacidades/restricciones y tareas que la pueden interrumpir.

5.5. Definición de Tareas Cooperativas

Un diagrama de actividades de UML es especialmente conveniente para modelar sistemas donde no existen transiciones disparadas por eventos. Los eventos ya han sido capturados en los dos pasos previos. Durante todo el proceso de



(a) (b)

Figura 7. Definición de una Tarea Cooperativa

modelado se especifica coordinación, pero aquí es donde aparece de forma más explícita con el fin de

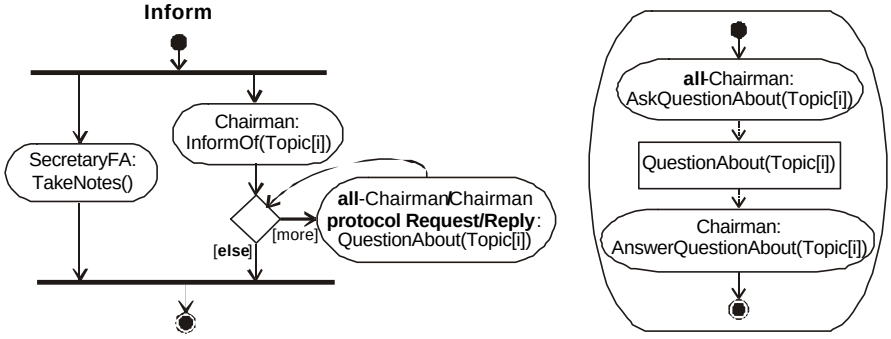
lograr la colaboración deseada. La coordinación necesaria para llevar a cabo las tareas implica directamente coordinación entre actores.

En la figura 7 se muestra la definición de la tarea cooperativa . En primer lugar (figura 7(a)), la tarea se divide en estados generales de actividad describiendo los pasos necesarios para su realización, en este caso es una simple secuencia de actividades. Después, cada actividad se define por la composición de otras actividades/acciones como se muestra para la actividad de lectura y aprobación del acta de la junta anterior en la figura 7(b). Y así sucesivamente, se continúa el proceso hasta llegar a

especificar todos los estados del diagrama como estados de acción, esto indica que ya no es posible continuar con la descomposición. La figura 7(b) también muestra como es posible modelar la composición concurrente/paralela de actividades (barras gruesas horizontales), condicionales (rombos), especificar restricciones adicionales como comentarios estereotipados, etc.

5.6. Protocolos de Interacción

Hasta ahora no se ha tenido en cuenta explícitamente la interacción directa entre actores. Es posible



(a) (b)

Figura 8. Descripción del protocolo de comunicación.

especificar protocolos de interacción entre actores para las actividades que lo requieran. La especificación de uno de estos protocolos consta de:

1. La descripción de los participantes identificando fuente y destino de la interacción. En el ejemplo de la figura d(a), la fuente en la interacción es cualquier participante en la junta a excepción del presidente y el destino es el presidente (*all-Chairman/Chairman*).
2. Tipo de reglas que gobiernan la interacción, en este caso un protocolo del tipo petición-respuesta (*Request-Reply*).
3. Contenido de la interacción, es decir, los objetos implicados en la comunicación (*QuestionAbout(Topic[i])*).

La figura 8(b) describe con detalle la actividad con el protocolo de interacción que aparece en la parte (a). Se han identificado otros protocolos estándar, y cualesquiera otros pueden ser incorporados según sean requeridos. Según se ha podido observar anteriormente, estos protocolos de interacción pretenden ser lo más abstractos posible, lo que nos permite desacoplar interacción de medios de comunicación concretos. Así, la ligadura entre ellos puede resolverse dinámicamente durante el uso del sistema.

6. Conclusiones y Trabajos Futuros

En este artículo hemos presentado una metodología de desarrollo de sistemas interactivos basándonos en la descripción del usuario como centro del proceso.

Esta aproximación centrada en el usuario permite conocer sus responsabilidades en el sistema y los posibles cambios de rol que pueda ir realizando en el futuro. Podemos describir de manera formal la evolución dinámica del usuario en el sistema y expresarlo mediante diagramas. Esta evolución dinámica en el comportamiento del usuario determina los cambios en las tareas que puede realizar el usuario así como la forma de llevarlas a cabo.

Hemos abordado la importancia de los sistemas cooperativos en la sociedad actual, y hemos analizado las propiedades que caracterizan a este tipo de entornos. En base a este análisis se ha desarrollado un modelo que permite describir sistemas que contienen organizaciones de grupos de personas que realizan actividades de forma cooperativa.

Se ha desarrollado una notación basada en UML que nos permite modelar las bases y peculiaridades que conforman los grupos.

Los trabajos futuros se centran en el diseño y construcción de herramientas de apoyo al modelado y que permitan automatizar algunas de las fases del desarrollo.

Referencias

1. S. Card, T. Moran, A. Newell: “The psychology of Human-Computer Interaction”. Lawrence Erlbaum, Hillsdale, N.J., (1983).
2. A. Dix: “Formal Methods for Interactive Systems”. Academic Press, (1991).
3. M. Harrison, H. Thimbleby (eds.) “Formal Methods in Human-Computer Interaction”. Cambridge University Press, (1990).
4. D. Duke, M. Harrison: “Abstract Interaction Objects”. Computer Graphics Forum, 12 (3): 25-36. (1993).
5. J. Coutaz: “PAC, and object oriented model for dialogue design”. En proc. Interact’87. North-Holland, (1987).
6. A. Golberg: “Smalltalk 80: The interactive programming environment”. Addison Wesley, (1984).
7. D. E. Kieras, P. G. Polson: “An approach to the formal analysis of user complexity”. International Journal Man-Machine Studies, 22. (1985).
8. S. J. Paine, T. Green: “Task-Action Grammars: A model of the mental representation of task languages”. Human-Computer Interaction, 2. (1986)
9. B. E. John, D. E. Kieras: “The GOMS Family of user Interface Analysis Techniques: Comparison and Contrast”. ACM Transactions on Human-Computer Interaction, vol 3, nº 4, (1996).
10. G. Van der Veer, B. Lenting, B. Bergevoet: “GTA: Groupware Task Analysis - Modelling Complexity”. Acta Psycologica, 91. (1996)
11. M. Van Welie, G.C. van der Veer: “An Ontology for Task World Models”. En Design, Specification and Verification of Interactive System’98. Springer Computer Science (1998).
12. J.C.Torres, M. Gea, F.L. Gutiérrez, M. Cabrera, M.L. Rodríguez: “GRALPLA: An algebraic Specification Language for Interactive graphic Systems”. In Design, Specification and Verification of Interactive System’96. Springer Computer Science (1996).
13. M.Gea, “Especificación formal de sistemas gráficos.” Tesis doctoral, Universidad de Granada, (1997).
14. F.L. Gutiérrez, M. Gea, J.C. Torres: “Verification of Interactive System Using Algebraic Specification”. In Proc. of Design,

Specification and Verification of Interactive System'98. (1998).
15. J.L, Garrido., M. Gea, F.L. Gutiérrez, N. Padilla, : "Designing Cooperative Systems for Human Collaboration". In: Dieng, R., Giboin, A., Karsenty, L., De Michelis, G. (eds.): Designing Cooperative Systems - The Use of Theories and Models. IOS Press-Ohmsha (2000)
16. J., Jacobson, I., Booch, G.: "The Unified Modeling Language - Reference Manual". Addison-Wesley (1999)
17. F. Paterno, C. Mancini, S. Meniconni, "ConcurTaskTrees: A Diagrammatic Notation for Specifyng Task Models", Proceeding of Interact'97, (1997)
18. G. Calvary, J. Coutaz, L. Nigay, "From Single User Architecture to Design to PAC*", Proceeding of CHI'97. (1997).
19. F.L. Gutiérrez, M. Cabrera, J.C. Torres, M. Gea: "Modelo de construcción de Sistemas Interactivos basado en técnicas formales". I Jornadas de Interacción Persona Ordenador, Interacción'2000. Granada, (2000).
20. J.L. Garrido, M. Gea, "Modelling Dynamic Group Behaviours". In: Johnson, C. (ed.): Interactive Systems - Design, Specification and Verification. LNCS 2220. Springer (2001)