Hacia Una Metodología de Desarrollo Para la Construcción de Entornos Virtuales

Gonzalo Méndez Pozo[1], Mª Isabel Sánchez Segura[2], Angélica de Antonio Jiménez[1]

[1]Facultad de Informática
Universidad Politécnica de Madrid
Campus de Montegancedo s/n
28660 Boadilla del Monte, Madrid
(+34) 91 336 69 25
gonzalo@gordini.ls.fi.upm.es
angelica@fi.upm.es

[2]Escuela Politécnica Superior
Universidad Carlos III de Madrid
Departamento de Informática
Avda. Universidad, 30, Leganés 28911
Madrid
(+34) 91 624 91 04
misanche@inf.uc3m.es

Abstract. La última tendencia en la construcción de Entornos Virtuales (EVs) es el desarrollo orientado a objetos, incluso si no está basado en ninguna metodología de desarrollo formal. En el presente artículo se pretende dar apoyo a esta tendencia y clarificar por qué las metodologías orientadas a objetos son una buena elección. Usando una de las notaciones más extendidas en el desarrollo orientado a objetos, UML, y el método de desarrollo de software expuesto por Craig Larman, se exponen algunas modificaciones al mismo para hacer que este método se ajuste mejor a las necesidades del desarrollo de este tipo de sistema software. Esto no constituye un método definitivo para el desarrollo de EVs, sino una primera aproximación que debe aún ser completada y refinada. También sirve como medio para analizar las deficiencias del citado método, y de otras metodologías, de manera que se pueda tomar como punto de partida para desarrollar las técnicas que sean necesarias para dar un apoyo formal a la construcción de EVs.

1. Introducción

Es innegable que, durante los últimos años, Internet ha experimentado un enorme crecimiento que ha causado la creación y desarrollo de nuevas tecnologías. Uno de los campos que está sacando mayor partido de este desarrollo es el de los Entornos Virtuales (EVs), que están evolucionando hacia aplicaciones multiusuario que se ejecutan a través de una red de ordenadores, lo que permite a personas de todo el mundo relacionarse como si estuviesen en el mismo lugar.

Sin embargo, este crecimiento tan rápido también está provocando que la construcción de este nuevo tipo de software se realice de una manera carente de cualquier rigor formal, mientras que la tendencia actual en la construcción de software apunta hacia la formalización de los procesos con el objetivo de garantizar su madurez [10]. Actualmente se está usando muy poco conocimiento del existente en el campo de la Ingeniería del Software para construir EVs [1], [2], [6], [9].

Se está haciendo patente la necesidad de adoptar o adaptar una de las metodologías de desarrollo de software existentes o crear una nueva, si es que ninguna de las existentes se adecua a las necesidades del sistema en construcción. En el caso de los EVs, no obstante, su construcción no es fácilmente adaptable a los paradigmas de desarrollo de software más extendidos actualmente, como el estructurado o el orientado a objetos.

Las metodologías orientadas a objetos son las más adecuadas para construir EVs, y pueden ser la clave para establecer una base real para el desarrollo de EVs [6]. Hay una razón para este hecho, y es que la parte visual del EV consiste en una serie de objetos que interactúan con otros. La identificación de las propiedades del objeto virtual con los atributos del objeto del diseño es inmediata, y las acciones y reacciones del objeto virtual se pueden equiparar con los métodos de los objetos del diseño [3].

Por encima de todo, uno de los principales objetivos de la Ingeniería del Software debe ser la búsqueda de soluciones simples a los problemas, y el concepto del emparejamiento que se menciona anteriormente es tan simple e intuitivo que es conveniente dedicarle un poco más de tiempo para ver si es una solución válida a uno de los problemas subyacentes a los EVs.

Una de las principales diferencias entre los EVs y las aplicaciones tradicionales es el hecho de que los EVs son eminentemente visuales, y es esta característica de su visualización la que aporta una indicación inicial sobre la dirección que deberían seguir nuestros esfuerzos. Además, una de las ventajas del paradigma orientado a objetos es el hecho de que un buen diseño es reutilizable, lo que es fundamental a la hora de construir EVs, ya que uno de los principales objetivos es construirlos de forma que puedan ser ampliados o que se puedan intercambiar objetos con otros EVs.

La razón por la cual se han elegido UML y el método de Larman en este trabajo ha sido para evaluar su utilidad para el diseño de EVs, incluso sabiendo de antemano que no iban a dar soporte a todas las características de los EVs, pues, por ejemplo, no contemplan el desarrollo de los componentes tridimensionales de los mismos. Como veremos más adelante, ésta no es la única carencia.

En los sucesivos apartados se dará una breve descripción de uno de los sistemas utilizados para realizar el presente estudio, al cual nos referiremos para explicar las diferentes técnicas utilizadas. Tras esta descripción se explicará cuál ha sido el proceso de desarrollo utilizado, haciendo hincapié en las carencias que se han encontrado en cada momento y cómo se ha intentado suplirlas. Después se presentará el proceso que se ha diseñado a partir de éste y otros proyectos y que se adapta mucho mejor a las necesidades que surgen a la hora de construir un Entorno Virtual. Y por último, se dará una visión de hacia dónde se dirigirán posteriores esfuerzos en este campo.

2. Descripción del Sistema

El EV que se ha tomado como ejemplo fue construido inicialmente como parte del proyecto Amusement (ESPRIT IV 25197). Uno de los objetivos de este proyecto era experimentar con la interacción social en escenarios cuyo principal propósito era servir como recintos de juego. Por esta razón se decidió implementar el juego del escondite inglés, juego elegido por la simplicidad de sus reglas y por las posibilidades que ofrecía para el estudio de la interacción social, la movilidad de los participantes, el sentimiento de inmersión, etc.

En este juego existen dos tipos de jugadores: uno, que es el que se la liga, se coloca ante una pared; el resto de los participantes se coloca tras una línea de salida y deben tratar de alcanzar esa pared lo antes posible sin que el jugador que se la liga los vea moverse. Para ello, el jugador que se la liga deberá volverse hacia la pared y decir una frase en voz alta, y mientras tanto el resto de los jugadores deberán ir acercándose a la pared. El ganador es aquel jugador que alcanza la pared en primer lugar, en cuyo caso pasa a ligársela y el anterior toma ahora el papel de jugador. Todos los participantes tienen una representación virtual que recibe el nombre de avatar.

El propósito era implementar este EV de forma que varios jugadores participasen a la vez. Además, para proporcionarles un avatar con muchas posibilidades sin complicar su manejo, decidimos dotarles de ciertos rasgos de humor y personalidad, de manera que pudiesen decidir la mejor forma de realizar ciertas acciones.

3. El Proceso de Desarrollo

Hemos utilizado el método de Larman [8], basado en UML, para desarrollar el EV que se ha descrito. Una de las razones por las que se ha elegido este método ha sido que es suficientemente flexible como para introducir o eliminar elementos de la notación UML. Además, Larman propone un ciclo de vida iterativo e incremental, lo cual es muy apropiado para el desarrollo de EVs. El proceso seguido y las técnicas aplicadas han sido los siguientes:

- Planificación y especificación de requisitos:
  - Especificación de Requisitos.
  - Definición de los casos de uso de alto nivel.
- Análisis:
  - Definición de los casos de uso expandidos.
  - Modelo Conceptual.
  - Diagramas de secuencia del sistema.
  - Contratos de Operaciones.
  - Diagramas de estado.
- Diseño
  - Elaboración de la interfaz de usuario.
  - Diagramas de Interacción.
  - Diagrama de clases del diseño.

  - Modelo de datos.
  - Arquitectura del sistema.

- Implementación y pruebas.

Además, se admitió la posibilidad de incluir, de ser necesario, procesos adicionales que no aparecen mencionados dentro del método de Larman ni en la notación UML.

3.1. Planificación y Especificación de Requisitos

En esta primera fase se ha encontrado ya la primera carencia del método, ya que aunque se sugiere la extracción de los requisitos a partir de los casos de uso, esto sólo funciona en el caso de la extracción de requisitos funcionales, pero no hay forma de obtener una especificación completa, ya que no se pueden obtener requisitos hardware o de otro tipo, los cuales, en este tipo de aplicación, pueden afectar profundamente al proceso de desarrollo y deben ser identificados lo antes posible.

3.1.1. Especificación de Requisitos

Como modelo para el documento de especificación de requisitos se ha utilizado el estándar IEEE 830 [4], que permite definir todas las características que debe cumplir la aplicación a desarrollar.

La experiencia nos ha mostrado que no es necesario especificar todos los requisitos desde el principio para poder comenzar el análisis, ya que en esta etapa tan temprana el desarrollo de software y de los modelos 3D es prácticamente independiente. Como se verá más adelante, es necesaria alguna coordinación entre el diseñador del sistema y el diseñador gráfico, pero el diseño y la construcción de la parte gráfica del EV puede comenzar en cualquier punto del análisis o el diseño del sistema, en tanto no retrase el desarrollo del resto del proceso de construcción del software.

3.1.2. Casos de Uso de Alto Nivel

Durante la construcción de un EV, un método como el de Larman, que está dirigido por casos de uso, tiene un inconveniente bastante importante: no considera ni el hecho de que en un EV suelen existir agentes autónomos ni el que puede haber acciones que realice el avatar que no estén controladas por el usuario.

Esto provoca que algunas acciones, que pueden ser claves para el funcionamiento del sistema, no se puedan expresar como casos de uso. Es más, a estas alturas del proceso, es difícil definirlas de alguna otra manera ajustándose al método de Larman, por lo que si no se hace algo estas acciones no se tendrían en cuenta en el resto del proceso de desarrollo. Si esto ocurriera, el sistema podría carecer de ciertas características que sin embargo aparecen en los requisitos iniciales.

En el caso de tener agentes, se podría hacer un pequeño 'truco', que es considerar a estos agentes como actores que interactuarían con el sistema, y estos agentes serían diseñados de una manera apropiada y separada del propio EV. Después, al igual que un usuario humano, interactuarían con el resto de los habitantes del EV a través de un avatar.

Este truco no sería válido, sin embargo, a la hora de diseñar avatares controlados por el usuario pero que puedan realizar acciones de manera autónoma. El uso de diagramas de estado se podría considerar como una alternativa para representar el comportamiento del avatar en estos casos, pero nos encontramos aún en una fase muy preliminar como para tener toda la información necesaria para poder hacer esto.

Existe aún otro caso que podría presentarse: si le damos al usuario la posibilidad de decidir qué acciones quiere controlar él y cuáles deben estar a cargo de su avatar, entonces deberíamos tener en cuenta que la misma acción puede ser ordenada por el usuario o por el propio avatar. Puesto que comienzan de manera distinta, deberían ser contempladas como dos acciones diferentes. La principal ventaja de este acercamiento es que en un primer ciclo de desarrollo podemos desarrollar las acciones que debe realizar el usuario, de manera que en un segundo ciclo ya tendremos información suficiente para definir las acciones controladas por el avatar, bien usando diagramas de estados, bien usando otra notación que exprese adecuadamente lo que se debe realizar.

En cualquier caso, lo que parece estar claro es el hecho de que los casos de uso no son el medio más adecuado para definir acciones que no vayan a ser iniciadas por actores. Sin embargo, deben ser descritas de alguna manera para que se puedan considerar dentro del plan de desarrollo del sistema y se puedan asignar a alguno de los ciclos de desarrollo. Por esta razón se ha decidido definir estas acciones con una técnica nueva a la que hemos dado el nombre de ***concepto de uso***, y ha sido definida en [11]. Los conceptos de uso no aparecen en la especificación de UML 1.3 [12]. Básicamente, lo que hace un concepto de uso es describir someramente el propósito de una acción, cómo funciona y, puesto que es automática, cuándo debe realizarse. Esto es un buen punto de partida si luego vamos a usar diagramas de estado para modelarlas.

Sin los conceptos de uso no es posible especificar todas las funcionalidades del EV que queremos construir, por lo que tendremos que incluir esta técnica dentro del proceso tratando de provocar el menor número de cambios posible.

El criterio que se ha usado para planificar los ciclos de desarrollo se ha basado en la cantidad de información disponible. Como resultado, se desarrollaron las funcionalidades descritas en los casos de uso en primer lugar, dejando las descritas en los conceptos de uso para el segundo ciclo de desarrollo, para así poder sacar partido de la información obtenida en el primer ciclo de desarrollo.

3.2. Primer Ciclo de Desarrollo

Los elementos a desarrollar en este ciclo corresponden a los casos de uso obtenidos a partir de los requisitos iniciales.

3.2.1. Fase de Análisis

En este ciclo no tiene sentido intentar diseñar diagramas de estados, ya que no aportan ninguna información de relevancia. Para el resto de las actividades, el resultado es el que se muestra.

3.2.1.1. Casos de Uso Expandidos

El primer paso en esta fase es detallar los casos de uso descritos anteriormente. Un caso de uso es un documento narrativo que describe una secuencia de eventos de un actor que usa el sistema para completar un proceso [5], es decir, una funcionalidad del sistema. Tomando esta definición al pie de la letra, resultaría que, en nuestro ejemplo, tendríamos un gran caso de uso que sería "*Jugar al escondite Inglés*", lo que no permitiría expresar la interacción existente de una manera clara, ya que habría un pequeño curso típico de eventos y un gran número de cursos alternativos y secciones.

Por tanto, la primera decisión que hubo que tomar fue interpretar los casos de uso de una manera más amplia, considerando cada acción que el avatar pudiese realizar como un caso de uso distinto, lo cual posibilita que se obtenga un número de casos de uso más razonable. También es cierto que en esta ocasión los casos de uso resultaron ser demasiado simples, pero es sobre todo debido a la naturaleza de la aplicación implementada.

El resto de las actividades de esta fase no presenta ningún aspecto destacable.

3.2.2. Fase de Diseño

Es en esta fase donde se ha encontrado más falta de apoyo formal a la hora de construir un EV. Se ha considerado necesario incorporar un nuevo proceso que incluya el diseño 3D del EV. Es más, es de extrema importancia realizar este proceso antes que ningún otro en la fase de diseño. La razón es bastante simple: si consideramos los avatares, su estructura será completamente distinta dependiendo de las acciones que se tengan que realizar. Si se quiere construir un EV en el que los usuarios puedan jugar a las cartas, sería importante tener caras con un alto grado de detalle de manera que los jugadores tengan la oportunidad de hacer trampas haciendo señas a su compañero a través de la boca, los ojos, la nariz, etc., pero no será necesario que estos avatares tengan cuerpo. En el caso del escondite inglés, en cambio, se necesitan avatares que puedan andar, correr, saltar, saludar, etc., por lo que es importante que tengan piernas y brazos articulados, pero la expresión facial no es tan importante, y de hecho puede ser preferible que carezcan de elementos como ojos, boca y nariz.

Una vez que están claras las diferentes necesidades en la estructura de los avatares, es fácil ver que esta estructura implicará algunas diferencias en el diagrama de clases resultante, ya que la idea es dar soporte a cada objeto gráfico con un objeto de una de las clases que aparezcan en el diagrama de clases, para poder manejar así sus propiedades y sus acciones. Cualquier cambio en la estructura de un avatar implicará un cambio equivalente en el diagrama de clases que lo representa. Además, la estructura jerárquica de las piezas del avatar puede ser una buena guía en cuanto a cómo crear la estructura del diagrama de clases que representen al avatar.

Por tanto, es de gran importancia disponer de un proceso específico en el que definir la estructura de la componente tridimensional del EV. En este caso se ha usado el método descrito en [11].

3.2.2.1. Diseño 3D

En el proceso de diseño 3D se deben considerar dos aspectos distintos. En primer lugar, se deben identificar todas las dependencias que existen entre los distintos

objetos para poder manejar fácilmente grupos de objetos. En segundo lugar, debe haber una manera clara de decirle a los diseñadores gráficos cómo deben construir los modelos 3D. Los diseñadores gráficos intentarán hacer siempre unos modelos con la mejor apariencia posible, y puesto que normalmente no tienen ninguna relación previa con el desarrollo de EVs, puede resultar difícil que comprendan las limitaciones a las que estos están sometidos (número de polígonos, uso de texturas, etc.).

Por tanto, es necesario proporcionarle al diseñador gráfico dos entradas básicas:

- Formularios de modelado donde se le indiquen las características de los modelos 3D.
- Bocetos de los objetos a diseñar:
  - Para los avatares, la estructura corporal debe estar muy detallada, así como la estructura jerárquica de todas sus partes.
  - Para los escenarios, mapas de vistas donde se muestre la localización de los objetos.
  - Para los objetos, su forma y las partes que no pueden simularse con texturas.

3.2.2.2. Diagrama de Clases

La construcción de los diagramas de colaboración a partir de los contratos de operación es bastante sencilla, y tras ella nos queda la realización del diagrama de clases, que es uno de los productos clave del diseño. Puede existir una ligera diferencia entre la estructura del avatar que se ha realizado en los bocetos del diseño 3D y la que aparece en el diagrama de clases, pero ello es debido a que esta última resulta mucho más flexible y más fácil de implementar, ya que ofrece un mejor manejo del nivel de detalle con el que se construye el avatar.

3.2.2.3. Arquitectura del Sistema

Aunque Larman no sugiere su uso de manera explícita, hay algunos diagramas, como el de despliegue y el de componentes, que pueden ser muy útiles en un EV a la hora de aclarar algunos aspectos del sistema, como la forma en que se almacenarán y relacionarán entre sí los diferentes elementos usados para construir un avatar.

Mientras que las herramientas de modelado permiten realizar una serie de acciones bastante complejas, las librerías gráficas limitan en gran medida las operaciones que se pueden realizar en un EV que se debe ejecutar en tiempo real, como el manejo de objetos compuestos o el uso de texturas. Para resolver este problema, una solución suele ser separar un objeto complejo en varios simples y tratarlos de forma separada. Los diagramas de componentes permitirán expresar las relaciones entre distintos ficheros con objetos y sus correspondientes texturas.

Si, además, se utiliza hardware específico de Realidad Virtual (guantes, cascos, etc.), los diagramas de despliegue también ayudarán a dar una visión de su relación con el sistema.

3.3. Segundo Ciclo de Desarrollo

Tras la finalización del primer ciclo de desarrollo, es necesario ampliar el sistema para incluir las funcionalidades que los usuarios pueden delegar en los avatares, de forma que se puedan centrar más en lo que sucede en el EV que en el manejo de su avatar.

3.3.1. Análisis

Como ya se ha comentado anteriormente, se añadirán los conceptos de uso al método de Larman para tener un apoyo formal para la construcción de esta parte del EV.

A partir de los conceptos de uso se realizarán una serie de escenarios, de manera similar a como se extraían los diagramas de secuencia del sistema a partir de los casos de uso. También se realizarán, para dotar de más claridad al análisis, diagramas de estados para las acciones representadas en los escenarios, pues aunque no aparecen entre los diagramas que recomienda Larman para esta fase, hemos podido comprobar su utilidad para no dejar escapar ningún tipo de información necesaria.

Por último, siguiendo otra vez con el método de Larman, se extenderá el modelo conceptual obtenido en el primer ciclo del presente desarrollo y, para concluir con el análisis de este segundo ciclo, se especificarán los contratos de las nuevas operaciones, tras lo cual se iniciará la fase de diseño.

3.3.2. Diseño

Para realizar el diseño de este segundo ciclo, se van a seguir también las recomendaciones definidas por Larman, pero, además, se va a completar con la introducción de un nuevo apartado en el que se desarrollarán unos diagramas de estados más detallados a partir de los que se realizaron en la fase de análisis. A pesar de que Larman desaconseja su utilización, creemos que en el caso del diseño es la mejor manera para modelar el comportamiento de los avatares, ya que aunque todos los aspectos que engloban aparecen dispersos por otros diagramas, estos diagramas nos ayudan a ver esta información de forma condensada.

3.4. Conclusiones sobre la Aplicación del Método de Larman

Haciendo un recorrido por todas las fases del método de Larman, las conclusiones a las que se puede llegar son las siguientes:

- Planificación y especificación de requisitos: en esta fase, donde se describen por primera vez los casos de uso, se echa de menos alguna referencia a la manera en que se debería construir el documento de especificación de requisitos para poder sacarle partido a la hora de describir los casos de uso. Por otro lado, también sería necesario especificar algún método para describir de manera más concreta las acciones que no realice directamente el usuario, ya que, si no se hace de manera correcta y lo más exhaustiva posible, ocasiona defectos que se van arrastrando a lo largo de todo el desarrollo.

- Análisis: de igual manera que en la etapa de planificación, hace falta alguna herramienta que permita desarrollar las funcionalidades que no corresponden directamente al usuario. De igual forma a como existe un camino a seguir desde los casos de uso en formato de alto nivel hasta los contratos de operaciones, debería existir un método para llegar desde la descripción de tareas realizadas automáticamente por el sistema hasta los mismos contratos de operaciones o alguna otra herramienta similar a ellos.
- Diseño: probablemente sea la etapa en la que se han encontrado más carencias a la hora de realizar el desarrollo. En primer lugar, ha habido que introducir una fase completamente nueva para poder contemplar el diseño 3D del EV. Siendo como es una de las partes constituyentes del sistema más importantes, es un hecho bastante desafortunado el no poder encontrar una fase en el método de desarrollo que al menos dé la oportunidad de incluir en ella este proceso. Por otro lado, a pesar de permitir su utilización, el método de Larman deja prácticamente de lado la estructura física del sistema, y eso es algo que, en un sistema de este tipo, puede dar lugar a que se dejen cabos sueltos, ya que es necesario, en una aplicación que depende tanto de dispositivos físicos y componentes software, especificar cómo se relacionan entre ellos. También hay que destacar que, aunque probablemente se les ha sacado menos partido del que hubiese sido posible, habría que darle mayor importancia a la utilización de los diagramas de transición de estados, pues parece a todas luces la herramienta más potente para reflejar el complejo comportamiento que tendrán elementos del sistema como los avatares.

Como se puede ver, el método de Larman puede funcionar bien en la construcción de aplicaciones de corte más o menos tradicional, pero presenta grandes lagunas con sistemas que se salen de lo habitual. Aún así, no es posible negar que para las partes del sistema que se asimilan a un sistema clásico, el método de Larman permite ir haciendo un desarrollo minucioso en el que se contempla toda la información necesaria para que el sistema haga todo lo que tiene que hacer, y el hecho de que se trate de un método orientado a objetos posibilita que se adapte muy bien a la estructura que tiene un EV.

4. Procesos Necesarios para Desarrollar un EV

A través de las conclusiones extraídas de la aplicación de la orientación a objetos al desarrollo de EVs, se puede ver que la aplicación de los procesos de desarrollo tal cual los proponen las diferentes metodologías existentes presenta problemas y carencias en la forma de definir los requisitos, problemas a la hora de gestionar los proyectos, sobre todo en la tarea de estimación, así como problemas de verificación de algunas partes del trabajo que debe realizarse.

En la Figura 1 y la Figura 2 aparecen los procesos seleccionados, tanto de ISO 12207 [14] como de IEEE 1074 [13], respectivamente, como procesos clave, y en los cuales hemos centrado especialmente nuestra atención. Dichos procesos son los que requieren un tratamiento especial o aquellos en los que se proponen técnicas nuevas. El resto de procesos no se han seleccionado porque no requieren una modificación

específica, es decir, no son tan dependientes del tipo de aplicaciones objeto de estudio, de modo que se pueden utilizar técnicas convencionales, tal cual están definidas actualmente.

En la Figura 3 aparece el Modelo de Procesos sobre el que se fundamenta nuestro trabajo. Para cada uno de los procesos incluidos en él será necesario dar indicaciones sobre cuáles son las cuestiones que las técnicas convencionales no son capaces de resolver, así como los motivos que llevan a demandar una mejora de estos.

Debido a las características especiales de estos desarrollos basados en EVs, los procesos de *Diseño* e *Implementación*, se van a ver descompuestos en otros.

El proceso de Diseño se descompone en: Diseño 3D del EV, Diseño de Elementos Multimedia, Diseño de la Arquitectura Interna de los Objetos y Diseño del Sistema. El proceso de Implementación se divide en dos procesos: Implementación de Componentes de Soporte e Implementación del Módulo Principal.

El motivo de esta división en procesos se debe a la diversidad e importancia de las tareas que en cada uno de ellos se realizan y a la relación existente entre procesos del modelo global. Gracias a esta descomposición se consigue un mejor seguimiento y visibilidad sobre los desarrollos que se lleven a cabo con este modelo como guía.

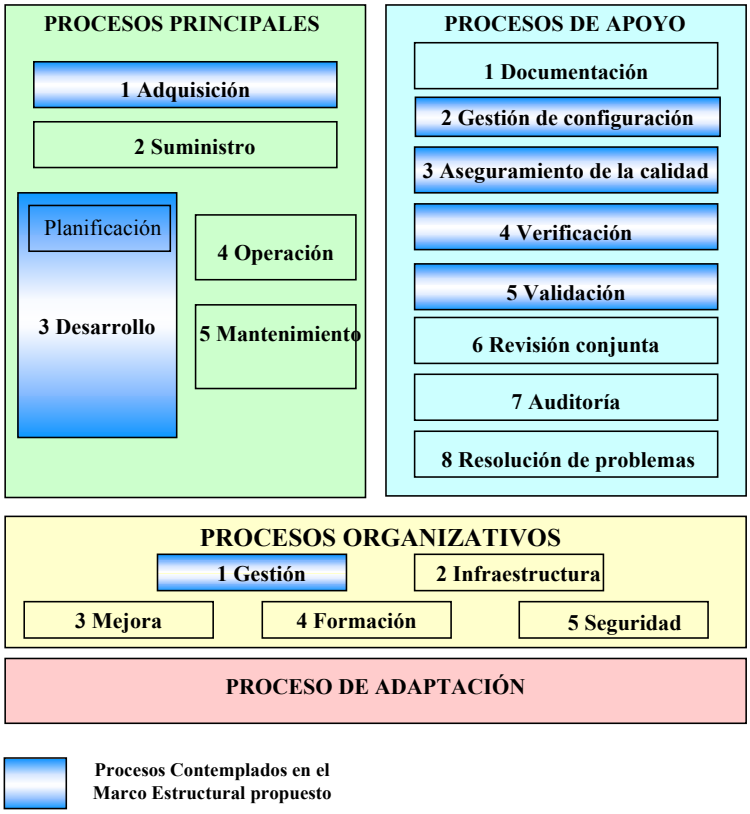

Figura 1: Procesos seleccionados de ISO 12207

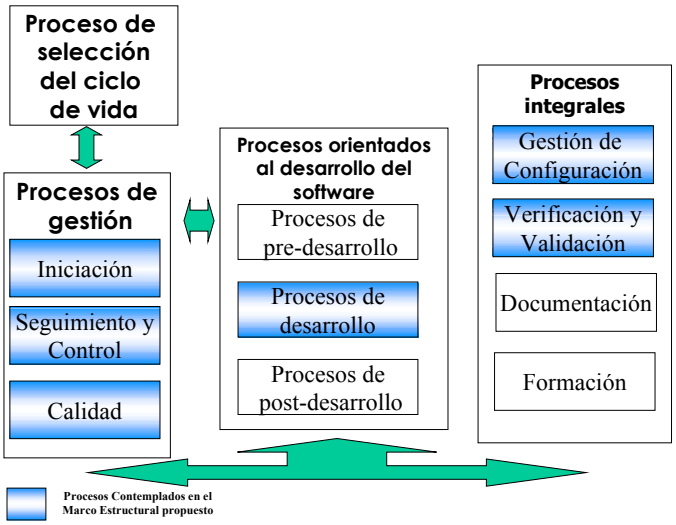

Figura 2: Procesos seleccionados de IEEE 1074

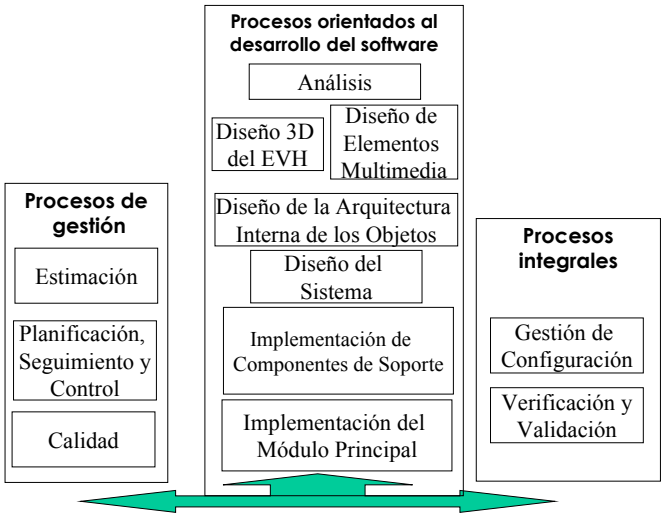

Figura 3: Modelo de Procesos Propuesto Detallado

A continuación, vamos a describir de forma general la necesidad de los procesos que sufren cambios o que son propuestos para la construcción de EVs.

Como en todo sistema software que se quiera desarrollar, el primer paso en el desarrollo debe ser la extracción de requisitos. Concretamente, en los EVs, es preciso tomar una serie de decisiones tecnológicas ya desde el principio, que van a marcar el resto del desarrollo.

Lo antes posible se debe decidir el tipo de dispositivos de realidad virtual a utilizar, el software de desarrollo, el hardware, etc., con el fin de comprobar la compatibilidad entre dichos elementos. Por esto se ha decidido incluir una tarea centrada en los Requisitos Específicos, dentro del proceso de análisis. Así se puede puntualizar sobre la toma de dichas decisiones, que son específicas para EVs.

Además, como en todos los sistemas software, se debe hace la extracción de requisitos funcionales. Una vez que se tenga la lista de requisitos del sistema, y tomando como axioma que la orientación a objetos es la que mejor se adapta a los EVs, se deben representar los casos de uso para dar satisfacción a todos aquellos requisitos que impliquen una interacción del usuario con el sistema, tal como se propone en [5].

Dichos casos de uso se quedan cortos al intentar representar determinadas funciones que no son demandadas por el usuario directamente. Para resolver esta falta de formalismos se propone una nueva técnica, los conceptos de uso, que resolverán este problema.

Tanto los casos como los conceptos de uso se deberán entonces catalogar basándonos en un conjunto de categorías, como son la percepción, el razonamiento, la animación, la visualización, etc. Así, cada una de estas categorías será tratada de forma aislada a través del proceso de Diseño de la arquitectura interna de los elementos.

Además de requisitos específicos y funcionales, existe otro conjunto de requisitos asociados al aspecto del EV. Dichos requisitos, como no son relativos a la funcionalidad del sistema, suelen quedar sin describir.

Para poder contemplarlos adecuadamente, se ha propuesto un proceso ubicado dentro de los de diseño llamado proceso de Diseño 3D.

Este proceso es necesario para poder alcanzar dos objetivos:

1. Extraer información sobre el aspecto del EV.
2. Poder comunicarse con el modelador, en un lenguaje común, que permita al diseñador del sistema y al modelador entenderse. En estos casos el lenguaje natural falla, debido a que la formación del diseñador del sistema (informático), y el modelador (experto en artes gráficas), es muy diferente.

Lo mismo ocurre con los elementos multimedia. Aunque son elementos que tienen menos riesgo que los elementos tridimensionales, también son importantes y muy diferentes del software convencional; y por eso se ha creado un proceso específico para el Diseño de Elementos Multimedia.

Una vez que se tiene el diseño de los elementos relacionados con el aspecto del EV, se debe comprobar que estos satisfacen al cliente. Para esto se deben construir maquetas, vídeos, etc. Esto debe realizarse como parte del proceso de Diseño del sistema, y concretamente en la tarea de Diseño de la Interfaz.

En dicha tarea se aprovechará para mostrar ejemplos de distintas alternativas de navegación por el EV. Esto es algo común a muchos sistemas software, pero es especialmente relevante en los EVs. La diferencia esencial entre la interfaz de un sistema tradicional y un EV es que en el primero la interfaz sirve para ofrecer funcionalidad al usuario, mientras que en el segundo sirve además para conseguir que el usuario se sienta parte del EV.

Las técnicas de diseño participativo o centradas en el usuario son útiles para llevar a cabo esta tarea.

A partir de todos los requisitos funcionales del EV se extraerán clases para construir el modelo de estructura estática. Dicho modelo incluirá, además, clases provenientes del proceso de Diseño 3D y del Diseño de Elementos Multimedia.

Al igual que los casos y conceptos de uso estaban catalogados, también cada método pertenecerá a una cierta categoría. Por ejemplo, si un caso de uso está catalogado dentro de la categoría de percepción, entonces seguro que dará lugar en alguna clase a un método de detección.

Dentro de las categorías contempladas, son especialmente críticas las que se refieren a lo que los EVs pueden percibir, las que representan características internas de los elementos, como rasgos de personalidad, humor, sociales, etc., las que se refieren al modo en que razonarán los elementos del EV y las que representan la reacción ante una interacción. Dichas reacciones pueden suponer una modificación en una variable o la representación de una escena muy compleja en el EV.

Para poder diseñar todos estos aspectos tan complejos, se propone un proceso de Diseño de la arquitectura interna de los componentes. En dicho proceso se diseñará:

1. Cómo deben detectar o percibir los elementos del EV.
2. Cómo afectarán las características internas de los elementos al comportamiento de estos.
3. Cómo funcionarán los mecanismos de razonamiento partiendo de la detección de una interacción.
4. Cómo se representarán externamente las acciones de los elementos del EV.

Los cuatro elementos anteriores se diseñarán para cada clase. Obviamente, si una clase no tiene mecanismos de percepción no se diseñarán métodos de detección.

Por otro lado, dado que los EVs tienen muchos módulos o partes, como por ejemplo el software del dispositivo de realidad virtual, los modelos 3D, los elementos multimedia, etc., que conforman todos ellos el EV, pero que pueden ir implementándose de forma separada, se ha propuesto un proceso de Implementación de los Componentes de Soporte, que organiza la implementación de todos estos módulos o partes de forma independiente.

A medida que dichos módulos vayan terminándose, se integrarán en el sistema, con el fin de ir mostrándole al cliente, lo antes posible, el EV definitivo, de modo que si no es lo que esperaba se puedan hacer modificaciones a la parte que se acaba de incorporar. Para ello se ha creado un proceso de Implementación del Módulo Principal que permite, de forma incremental, ir añadiendo pequeños módulos al EV.

5. Conclusiones y Trabajo Futuro

A lo largo del presente trabajo se ha podido observar que las metodologías tradicionales ofrecen poco soporte a la construcción de EVs, especialmente para aquellas características que se separan de lo que son los sistemas tradicionales, como puede ser toda la parte gráfica asociada a estos sistemas o la existencia de acciones no

controladas ni iniciadas por el usuario. Además, dada la relativa novedad de este tipo de software, es aún bastante escaso el trabajo que se ha llevado a cabo en el campo de la Ingeniería de Software aplicada a la construcción de EVs, por lo que el soporte formal para la construcción de los mismos es prácticamente inexistente.

El objetivo final de este trabajo es la definición de una metodología especialmente diseñada para la construcción de EVs, pero tomando como punto de partida las metodologías y técnicas ya existentes, de forma que su utilización no suponga una ruptura radical con las experiencias previas de los desarrolladores. Una primera aproximación a esta metodología puede ser consultada actualmente en [11], donde ya aparecen definidas herramientas como los conceptos de uso o los formularios de modelado 3D.

En cualquier caso, estos sistemas no se encuentran lo suficientemente evolucionados como para poder decir que nuestra aproximación al proceso de desarrollo dará soporte a cualquier tipo de EV que se quiera construir, por lo que en un futuro cercano se proseguirá este trabajo estudiando nuevas características que puedan surgir en los EVs que se encuentran actualmente en construcción. En concreto, se prestará especial atención a los llamados Entornos Virtuales Colaborativos, cuyas especiales características los están haciendo cada vez más populares y que no encuentran un soporte completo en el proceso que se ha presentado.

6. Referencias

1. Brown,J., Encarnaçao, J., Shniderman., B.: Human-Centered computing, online communities, and virtual environments. IEEE Computer Graphics and Applications. Vol 19, Nº6, Pg 70-74. 1999.
2. Fencott, C.: Towards a design methodology for Virtual Environments. Workshop on User Centered Design and Implementation of Virtual Environments. University of Teeside. 1999.
3. Hubbold, R., Cook, J., Keates, M., Gibson, S., Howard, T., Murta, A., West, A., Pettifer, S.: GNU/MAVERIK A micro-kernel for large-scale virtual environments. Proceedings of the ACM symposium on Virtual Reality Software and Technology (VRST'99). Pag. 66-73. December 1999. London, UK.
4. IEEE Recommended Practice for Software Requirements Specifications. Institute of Electrical and Electronics Engineers, Inc. 1998.
5. Jacobson, I.: Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley, 1992.
6. Kulwinder, K.: Interacting with virtual environments: an evaluation of a model of interaction. Interacting with computers. Vol. Nº 11, Pg. 403-426. 1999.
7. Larijani, C.: Realidad Virtual. McGrawHill, 1994.
8. Larman, C.: Applying UML and Patterns. Prentice-Hall, 1999.
9. Maher, M.L., Skow, B.: Designing the virtual campus. Design Studies. Vol 20, nº 4, pp 319-324. 1999.
10. Paulk, M.C., García, S.M., Chrissis, M.B., and Bush, M.: Capability Maturity Model for Software, Version 1.1. CMU/SEI-93-TR-24., CMU/SEI-93-TR-25. Technical Report. Software Engineering Institute. Carnegie Mellon University, 1993.

11. Sánchez Segura, M.I.: Modelado de Entornos Virtuales Basados en la Interacción Social “MEVBIS”. Master of Science Thesis in Software Engineering. Facultad de Informática, UPM, 1999.
12. OMG Unified Modelling Language Specification. Version 1.3, June 1999.
13. IEEE, Standard for Developing Software Life Cycle Processes. IEEE Std. 1074-1991. Nueva York (EE.UU.), IEEE Computer Society.
14. ISO/IEC Standard 12207:1995. (1995). Software Life Cycle Processes. Ginebra (Suiza), International Organization for Standardization.