Universidad de las Ciencias Informáticas

“Facultad de Entornos Virtuales”

Título:

Propuesta de Proceso de Software para sistemas de Realidad Virtual

Trabajo de Diploma para optar por el título de Ingeniería en Informática.

Autor:

Ignais La Paz Trujillo

Tutor:

Jandrich Domínguez Fortún

Co- Tutor:

Maikel Pérez Javier

Ciudad de la Habana, Cuba

Junio, 2007

“Miremos más que somos padres de nuestro porvenir que no hijos de nuestro pasado."

Miguel de Unamuno

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____	_____
Nombre del Autor	Nombre del Tutor
_____	_____
Firma del Autor	Firma del Tutor

Datos de contacto.

Autor.

Nombre: Ignais La Paz Trujillo.

Correo electrónico: ilapaz@estudiantes.uci.cu

Tutor.

Nombre: Jandrich Domínguez Fortín.

Correo electrónico: jandrich@uci.cu

Agradecimientos.

A cada una de esas personas que hicieron posible la terminación de esta tesis, a mis padres y mi hermano, a toda mi familia por su apoyo y confianza, a mi tutor, a Aracelis que sin su ayuda no hubiese llegado a donde he llegado, a Fidel Castro por haber ingeniado este proyecto, a mis amigos.

Dedicatoria.

A mis padres que bajo cualquier circunstancia hubiesen hecho lo imposible para que yo alcanzara este sueño.

A mi hermano por tener confianza en mí.

A toda mi familia por el apoyo que siempre me brindaron.

A la gente de mi barrio que me vio crecer, en especial a Ronal y Yordanis.

A mis compañeros de estudio de 5 años, a mis compañeros de cuarto.

A Raúl (Luly) por ser mi amigo, mi hermano, mi padre.

A todos los profesores que me ayudaron tanto, a todo aquel que me regalo un consejo, que me tendió la mano.

A la vida que me ha dado tanto.

Dedicado a todos.

Ignais

Resumen:

A partir de la necesidad de definir un proceso de software para proyectos de realidad virtual en la facultad 5 de la Universidad de las Ciencias Informáticas se realiza este trabajo, el cual propone una solución a la problemática planteada. Para elaborar la propuesta fue necesario realizar un estudio de diferentes metodologías de software utilizadas en el desarrollo de entornos virtuales dentro las que se encuentran RUP, XP y la metodología de Larman, se entrevistaron y encuestaron además a los lideres de proyectos con el objetivo de encontrar deficiencias en el modo de producción llevado a cabo hasta el momento y obtener sus opiniones para la perfección de un proceso futuro.

Después del estudio se elaboró un flujo de trabajo, que contiene 30 actividades y se separan de la siguiente manera: 17 actividades fundamentales, 8 de retroalimentación con el grupo de trabajo y 5 de retroalimentación con el cliente. El proceso está dividido fundamentalmente es dos partes; una primera parte de concepción y planificación hasta la firme del contrato, y posteriormente la etapa de construcción del proyecto hasta la entrega al cliente.

Palabras claves:

Proceso, realidad virtual.

Introducción.

Introducción:

Innegablemente el avance de la computación, así como de sus diferentes esferas ha facilitado el trabajo y desarrollo del hombre. Internet comparte información inimaginable, se ha convertido en la bola de cristal en la cual la información es accesible por cualquier persona dotada de una computadora personal. Una de las modalidades que más ventajas ha sacado a la red de redes son precisamente los Entornos Virtuales, donde a través del mismo cualquier ínter nauta podría conocer partes del mundo casi de la misma forma que estando presente en tiempo y espacio.

Pero su principal ventaja no radica solamente en esto. Probablemente sería más seguro antes de una operación quirúrgica, que un cirujano inexperto experimentara de forma virtual, las incisiones que se le realizarán a un paciente. De la misma forma aumentaría la confianza de un chofer si practicara frente a un ordenador especializado su examen de conducción automovilístico antes de la presentación oficial. O en una escuela especializada en clases de aviación, sus estudiantes ejercitarían el vuelo en tierra de manera que se evitarían accidentes indeseados. Así podemos concluir otra de las importancias de los Entornos Virtuales (EVs) en la construcción de simuladores, tan fundamentales en la rama de la educación.

Los juegos también forman parte de la representación virtual de la realidad, estos empleados de forma racional y centrados en la formación, incluyen de igual manera una importante modalidad de los entornos virtuales.

En la actualidad todo productor de software debido a sus magnitudes y complejidad, para lograr un sistema de calidad, en costo aceptable y que satisfaga las necesidades y requerimientos del cliente necesita de una metodología que guíe el proceso desde la misma conformación del equipo de trabajo (definición de roles), hasta el momento de prueba, entrega y mantenimiento del sistema automatizado. En la realidad esto no ocurre siempre, empresas y grupos de producción no emplean ninguna metodología lo que implica que a la larga sus proyectos obtienen perdidas tanto materiales como de clientes y esto los conduce a un fracaso total.

Situación Problémica: Actualmente la facultad 5 no tiene definido el proceso de producción de software de Realidad Virtual para la producción de simuladores y juegos, se usan diferentes metodologías de

desarrollo, y en el peor de los casos, algunos proyectos trabajan empíricamente, sin seguir ninguna de estas metodologías. Por lo que el trabajo es desorganizado e ineficiente, los proyectos se demoran y algunos no llegan ni a su fase final.

Podemos partir también de que no se tiene conocimiento de algún proceso encaminado al software de realidad virtual, y las metodologías existentes ninguna se ajusta exactamente a nuestro interés, algunas resultan complejas y otras muy simples, además de no adaptarse a los posibles equipos de trabajo y roles que requiere un sistema de realidad virtual en la universidad. Podemos poner el ejemplo de los diseñadores del sistema y los diseñadores gráficos, los cuales aunque no tienen la misma tarea deben trabajar muy unidos en la elaboración de los requerimientos. Se encuentran lagunas que de cierta forma restan calidad o seguridad al proceso de software para Entornos Virtuales. Un análisis más profundo lo podemos encontrar en el artículo “Hacia una metodología para el desarrollo de Entornos Virtuales” del Ingeniero Gonzalo Méndez Pozo (2001).

Por lo antes expuesto el **problema científico** de este trabajo es: ¿Cómo mejorar el proceso de producción de software de entornos virtuales en la Facultad 5 de la universidad de las Ciencias Informáticas?

En los **Objetivos Generales** se traza como meta **diseñar el proceso de desarrollo de software para Realidad Virtual según los intereses y necesidades de trabajo de la Facultad 5 y la UCI** basado en la experiencia de los desarrolladores de la facultad vinculados a los diferentes proyectos de la misma, sean estos tantos simuladores como juegos.

El siguiente trabajo investigativo, tiene como **objeto de estudio** un muestreo de las diferentes metodologías existentes en el mundo aplicadas a sistemas de EV, lo que construye el estado del arte. Se analizan exhaustivamente las diferentes caracterizaciones de las Metodologías Ágiles, como principal exponente tenemos la Programación Extrema (XP), y Metodologías Tradicionales tal como Proceso Unificado de Software (RUP). Se hará énfasis en sus principios y conformación de los equipos de desarrollo, así como su colaboración con el cliente.

Dentro de las **Principales Tareas de la Investigación** está el estudio del modo de producción de los diferentes proyectos desarrollados por la Facultad 5 así como la experiencia de desarrolladores de nuestro país para entornos virtuales al igual que el estado del arte a nivel internacional referente al interés investigativo, sean estos sistemas basados en Realidad Virtual.

Introducción.

En el siguiente trabajo se hará uso de diferentes **métodos científicos de la investigación** con el objetivo de lograr una mayor profundidad y organización. Trabajaremos a partir de Métodos Teóricos como el Analítico-Sintético y Métodos Empíricos como Entrevistas y Encuestas.

En el capítulo 1 se realiza un estudio del estado del arte, analizando cada una de las metodologías encontradas que de alguna forma han guiado procesos de software de entornos virtuales en lugares ajenos y a la universidad. Se profundiza en temas como sus principios y flujos de trabajos, valorando en algunos casos sus ventajas y desventajas respecto a Entornos Virtuales.

El capítulo 2 trata el tema de realidad en la facultad, como se realiza hasta el momento el proceso. Para este estudio se emplearan métodos de investigación tales como entrevistas y encuestas a los líderes de proyecto de la facultad.

Finalmente en el capítulo 3 se detalla el proceso propuesto, teniendo en cuenta los resultados de las investigaciones de los dos capítulos anteriores. La propuesta está basada en un flujo de actividades organizadas con el fin de asegurar la etapa de elaboración teniendo como premisa una etapa inicial de concepción y planificación del proyecto, donde cada actividad cuenta con responsables, participantes y artefactos o documentos de entrada y salida.

Capítulo 1

Introducción:

La Realidad Virtual (RV) es una de las áreas de investigación y desarrollo más reciente en la industria de la computación. Sus aplicaciones van desde el diseño de interiores hasta simulación de vuelos aeronáuticos y juegos. Existen diversas formas de emplear la tecnología de realidad virtual, teniendo como premisa, crear medios más intuitivos para que humanos y computadores trabajen juntos. En este capítulo se realizar un estudio de los antecedentes, estado del arte de las metodologías y procesos de software empleados para guiar los sistemas de Realidad Virtual. Para una mejor comprensión se incluyen los principales conceptos asociados relacionados con el tema.

1.1 Descripción de los principales conceptos asociados:

¿Qué es la Realidad Virtual?

Puede ser descrita como un conjunto de tecnologías que, apoyadas en el uso de la computadora, simulan nuestra realidad u otras realidades creadas por nuestra fantasía. Esta nueva herramienta permite a los usuarios de computadoras el PARTICIPAR de ambientes tridimensionales permitiéndoles interactuar con objetos virtuales. En resumen integra elementos de video, audio, y gráficos tridimensionales para crear una impresión de realidad, inicialmente mediante periféricos especialmente diseñados (cascos, guantes) y hoy día a través de la WWW en la Internet. Se atribuye la creación del termino Realidad Virtual al empresario-filosofo Jaron LANIER en la década de los ochenta.(GONZALO MÉNDEZ POSO; JAHN)

¿Qué es un sistema?

Un sistema es un conjunto de elementos organizados que interactúan entre sí y con su ambiente, para lograr objetivos comunes, operando sobre información, sobre energía o materia u organismos para producir como salida información o energía o materia u organismos. Un sistema aislado no intercambia ni materia ni energía con el medio ambiente.(*Wikipedia* 2007)

¿Qué es una Metodología de Desarrollo de Software?

Es aquella rama dentro de la Ingeniería de Software que se encarga de elaborar estrategias de desarrollo de software que promuevan prácticas adoptativas en vez de predictivas; centradas en las personas o los equipos, orientadas hacia la funcionalidad y la entrega, de comunicación intensiva y que requieren implicación directa del cliente.(*Wikipedia* 2007)

¿Qué es un Proceso de Desarrollo de Software?

Un proceso define quien está haciendo que, cuando y como alcanzar un determinado objetivo. En la ingeniería de software el objetivo es lograr un producto de calidad o mejorar uno existente. Un proceso efectivo proporciona normas para el desarrollo eficiente de software de calidad. Captura y presenta las mejores prácticas que el estado actual de la tecnología permite. En consecuencia reduce riesgos y hace el proyecto más predecible.(JAMES RUMBAUGH 2000)

1.2 Antecedentes:

El desarrollo de los Entornos Virtuales (EVs) es uno de los campos más novedosos de la computación, por lo que respecta a las metodologías usadas no podemos realizar un estudio profundo del tema, mucho menos a procesos de desarrollo que guíe la producción de software de este tipo.

Para entender los antecedentes debemos partir de la base del concepto de Realidad Virtual la cual se describe como: “un conjunto de tecnologías que, apoyadas en el uso de la computadora, simulan nuestra realidad u otras realidades creadas por nuestra fantasía. Esta nueva herramienta permite a los usuarios de computadoras el participar de ambientes tridimensionales permitiéndoles interactuar con objetos virtuales. En resumen integra elementos de video, audio, y gráficos tridimensionales para crear una impresión de realidad, inicialmente mediante periféricos especialmente diseñados (cascos, guantes) y hoy día a través de la WWW en la Internet. Se atribuye la creación del termino Realidad Virtual al empresario-filosofo Jaron Lanier en la década de los ochenta.(*VRML y Realidad Virtual*)

Su origen es debido a simuladores de vuelo desarrollados hace cincuenta años por el ejército norteamericano. También aporta a esta visión inicial el Cinerama, de múltiples cámaras de cine, y el Sensorama, un juego de arcada desarrollado en 1962 para simular un paseo multi-sensorial en bicicleta. Un precursor de la Realidad Virtual fue Ivan SUTHERLAND (MIT, 1965) quien introduce el primer casco de inmersión tridimensional, posteriormente divulgado en la industria de periféricos con el

nombre de HMD(Head.Mounted Display).Otro precursor, Nicholas NEGROPONTE y su grupo de MIT produce en los 70 un mapa virtual de rutas visitables de un modelo de la ciudad de Aspen, Colorado.(*VRML y Realidad Virtual*)

Tanto el desarrollo de simuladores como juegos en los últimos años ha sido vertiginoso y de mucho éxito, muy usado en la ciencia, la educación así como en el esparcimiento, se han apoderado de un importante espacio en el ámbito informático. Lamentablemente el brazo ejecutor de mayor auge en el mundo de los sistemas basados en Realidad Virtual lo controlan empresas privadas, por lo que el conocimiento de sus metodologías es casi inaccesible, obligándonos así a realizar un estudio comparativo entre los diferentes métodos existentes para así desarrollar un proceso de producción de software que se adapte a las condiciones y necesidades de la facultad 5 de la Universidad de las Ciencias Informáticas (UCI). Para ello nos apoyamos en el estudio de la ingeniería de software, su surgimiento, así como de diferentes metodologías que de alguna forma han sido utilizadas para el proceso de Entornos Virtuales.

1.2.1 Ingeniería de Software

ISW es una tecnología multicapa en la que según Pressman se pueden identificar los métodos (que indican como construir técnicamente el software), el proceso (es el fundamento de ISW, es la unión que mantiene junta las capas de la tecnología), y las herramientas (soporte automático y semiautomático para el proceso y los métodos)(UCI Curso 2005- 2006)

La ingeniería del Software nace como una disciplina para aplicar los principios técnicas y herramientas de desarrollo de software, surgió porque todos los desarrolladores en la década de los 80's, realizaban el software de forma artística, donde la experiencia (el ensayo-error) era el camino a seguir. Este enfoque produjo grandes y exitosos productos de programación pero conforme los proyectos se volvieron más complejos debido al avance del hardware y software y la penetración cada vez mayor de la informática en todos los ámbitos de la sociedad, llevó a que se produjera software sin calidad, se incumplieran los presupuestos y se incrementara dramáticamente los costos de mantenimiento.

A partir de la década de los 70 surgieron un grupo de métodos basados en la programación estructurada. Desde del surgimiento de la programación orientada a objetos en la década de los 80 y 90, se produce un incremento de las metodologías existente de aproximadamente 10 a más de 50. Mediante este desarrollo se llega a implementar el Proceso Unificado del Software (RUP).

1.2.2 El surgimiento de RUP:

En 1967 con la Metodología Ericsson (*Ericsson Approach*) elaborada por Ivar Jacobson, una aproximación de desarrollo basada en componentes, que introdujo el concepto de Caso de Uso. Entre los años de 1987 a 1995 Jacobson fundó la compañía *Objectory* AB y lanza el proceso de desarrollo *Objectory* (abreviación de *Object Factory*). Posteriormente en 1995 *Rational Software Corporation* adquiere *Objectory AB* y entre 1995 y 1997 se desarrolla *Rational Objectory Process* (ROP) a partir de *Objectory* 3.8 y del Enfoque Rational (*Rational Approach*) adoptando UML como lenguaje de modelado.

UML oficialmente se presentó cuando Rumbaugh, Booch y Jacobson unifican sus estudios con una semántica y notación, para lograr compatibilidad en el análisis y diseño orientado a objetos, permitiendo que los proyectos se asentaran en un lenguaje de modelado, ayudándole a los constructores de herramientas enfocarse en producir características más útiles.

Desde ese entonces y a la cabeza de Grady Booch, Ivar Jacobson y James Rumbaugh, Rational Software desarrolló e incorporó diversos elementos para expandir ROP, destacándose especialmente el flujo de trabajo conocido como modelado del negocio. En junio de1998 se lanza *Rational Unified Process*. (VALENCIA 2006)

1.2.3 Surgimiento de las Metodologías Ágiles:

En febrero de 2001, tras una reunión celebrada en Utah-EEUU, nace el término “ágil” aplicado al desarrollo de software. En esta reunión participan un grupo de 17 expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas.

Tras esta reunión se creó “The Agile Alliance”, una organización, sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida es fue el Manifiesto Ágil, un documento que resume la filosofía “ágil”.(JOSÉ H. CANÓS)

Según el Manifiesto se valora:

· **Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas.** La gente es el principal factor de éxito de un proyecto software. Es más importante construir un buen equipo que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte automáticamente. Es mejor crear el equipo y que éste configure su propio entorno de desarrollo en base a sus necesidades.

Desarrollar software que funciona más que conseguir una buena documentación. La regla a seguir es .no producir documentos a menos que sean necesarios de forma inmediata para tomar un decisión importante.. Estos documentos deben ser cortos y centrarse en lo fundamental.

La colaboración con el cliente más que la negociación de un contrato. Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.

Responder a los cambios más que seguir estrictamente un plan. La habilidad de responder a los cambios que puedan surgir a los largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta.

Los valores anteriores inspiran los doce principios del manifiesto, los cuales serán tratados más adelante en este capítulo. Son características que diferencian un proceso ágil de uno tradicional.(JOSÉ H. CANÓS)

Aunque los creadores e impulsores de las metodologías ágiles más populares han suscrito el manifiesto ágil y coinciden con los principios enunciados anteriormente, cada metodología tiene características propias y hace hincapié en algunos aspectos más específicos. A continuación se resumen otras metodologías ágiles. La mayoría de ellas ya estaban siendo utilizadas con éxito en proyectos reales pero les faltaba una mayor difusión y reconocimiento.

- **SCRUM.** Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las

reuniones a lo largo proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración.(JOSÉ H. CANÓS)

- **Crystal Methodologies.** Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. Han sido desarrolladas por Alistair Cockburn. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros).(JOSÉ H. CANÓS)
- **Dynamic Systems Development Method (DSDM).** Define el marco para desarrollar un proceso de producción de software. Nace en 1994 con el objetivo de crear una metodología RAD unificada. Sus principales características son: es un proceso iterativo e incremental y el equipo de desarrollo y el usuario trabajan juntos. Propone cinco fases: estudio viabilidad, estudio del negocio, modelado funcional, diseño y construcción, y finalmente implementación. Las tres últimas son iterativas, además de existir realimentación a todas las fases.(JOSÉ H. CANÓS)
- **Adaptive Software Development (ASD).** Su impulsor es Jim Highsmith. Sus principales características son: iterativo, orientado a los componentes software más que a las tareas y tolerante a los cambios. El ciclo de vida que propone tiene tres fases esenciales: especulación, colaboración y aprendizaje. En la primera de ellas se inicia el proyecto y se planifican las características del software; en la segunda desarrollan las características y finalmente en la tercera se revisa su calidad, y se entrega al cliente. La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo.(JOSÉ H. CANÓS)
- **Feature -Driven Development (FDD).** Define un proceso iterativo que consta de 5 pasos.(JOSÉ H. CANÓS)

- Las iteraciones son cortas (hasta 2 semanas). Se centra en las fases de diseño e implementación del sistema partiendo de una lista de características que debe reunir el software. Sus impulsores son Jeff De Luca y Peter Coad.(JOSÉ H. CANÓS)
- **Lean Development (LD).** Definida por Bob Charette.s a partir de su experiencia en proyectos con la industria japonesa del automóvil en los años 80 y utilizada en numerosos proyectos de telecomunicaciones en Europa. En LD, los cambios se consideran riesgos, pero si se manejan adecuadamente se pueden convertir en oportunidades que mejoren la productividad del cliente. Su principal característica es introducir un mecanismo para implementar dichos cambios.(JOSÉ H. CANÓS)

En general esto solo sería el surgimiento de estas metodologías, las cuales como se explica, han sido empleadas mayormente en proyectos sencillos y de corta duración principalmente sujetos a cambios constantes que no afecten al equipo ni al propio sistema en si. Por lo cual basados en sus características estos métodos han sido usados mayormente, en cuanto a nuestro interés, en juegos cortos y simuladores pequeños, ya que estos requieren de una rápida elaboración y al cliente le importa más el producto que la documentación que este pueda generar. No obstante no se le puede restar importancia a metodologías ágiles como XP, la cual en estos momentos cuenta con un importante prestigio dentro del mundo de la ingeniería de software, y si tomamos en cuenta la filosofía de: “divide y vencerás” podíamos tener buenos resultados en sistemas de grandes magnitudes.

1.3 Estudio de las principales metodologías usadas en el mundo para guiar el proceso de desarrollo de software para entornos virtuales:

Actualmente las metodologías de software se encuentran divididas en dos grandes grupos imperantes: “Metodologías Ágiles” siendo este el grupo más joven nacido a partir del 2001 a través del “Manifiesto Ágil” y teniendo como principales exponente a eXtreme Programming (XP), Cristal Methods, SCRUM, entre otras. Y por otro lado se cuenta con las “Metodologías Tradicionales”, siendo las pioneras de la ingeniería de software, se puede enumerar varías de ellas pero sería en vano si no se menciona a RUP, siendo esta una de las más usadas en el mundo y en Cuba.

1.3.1 Extreme Programming (XP)

Dentro de las “Metodologías Ágiles” es XP la que juega el papel más importante, ya que el resto parten de la misma, además de ser la más exitosa y difundida. Esto se debe al énfasis que realiza en la satisfacción del cliente y el trabajo en equipo. Otra característica que se destaca en XP es su fácil adaptación a los cambios bruscos del proyecto ya sea por problemas del propio equipo de trabajo o por modificaciones en los requisitos propuestos por el cliente, y es precisamente esta cualidad la que hace de XP una metodología difundida dentro del campo de los Entornos Virtuales (EVs), principalmente en el ámbito de los juegos. Entre las compañías productoras de videojuegos más importantes y que guían su proceso a través de XP se encuentra la “EA Games”, la cual se destaca en juegos de tres dimensiones tratando temas mayormente deportivo.

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico; ajustándose esta característica principalmente a los juegos

El desarrollo bajo XP tiene características que lo distinguen claramente de otras metodologías:

- Los diseñadores y programadores se comunican efectivamente con el cliente y entre ellos mismos
- Los diseños del software se mantienen sencillos y libres de complejidad o pretensiones excesivas.
- Se obtiene retroalimentación de usuarios y clientes desde el primer día gracias a las baterías de pruebas
- El software es liberado en entregas frecuentes tan pronto como sea posible
- Los cambios se implementan rápidamente tal y como fueron sugeridos.

- Las metas en características, tiempos y costos son reajustadas permanentemente en función del avance real obtenido.(FRANKLIN RENE RIVERO GARCÍA 2006)

Entre los principales valores que promueve XP se encuentran:

- **Comunicación**

  El eXtreme Programming se nutre del ancho de banda más grande que se puede obtener cuando existe algún tipo de comunicación: la comunicación directa entre personas. Cuando dos (o más) personas se comunican directamente pueden no solo consumir las palabras formuladas por la otra persona, sino que también aprecian los gestos, miradas, etc. que hace su compañero. Sin embargo, en una conversación mediante el correo electrónico, hay muchos factores que hacen de esta una comunicación, por así decirlo, mucho menos efectiva.

- **Coraje**

  El coraje es un valor muy importante dentro de la programación extrema. Un miembro de un equipo de desarrollo extremo debe de tener el coraje de exponer sus dudas, miedos, experiencias sin "embellecer" éstas de ninguna de las maneras. Esto es muy importante ya que un equipo de desarrollo extremo se basa en la confianza para con sus miembros. Faltar a esta confianza es una falta más que grave.

- **Simplicidad**

  Dado que no se puede predecir como va a ser en el futuro, el software que se esta desarrollando; un equipo de programación extrema intenta mantener el software lo más sencillo posible. Esto quiere decir que no se va a invertir ningún esfuerzo en hacer un desarrollo que en un futuro pueda llegar a tener valor. En el XP frases como "...en un futuro vamos a necesitar..." o "Haz un sistema genérico de..." no tienen ningún sentido ya que no aportan ningún valor en el momento.

Este valor es justo de analizar ya que puede ser interpretado de disímiles formas. El caso más flexible sería, no detenerse en funcionalidades del sistema que en el presente no son verdaderamente importantes, o incluir algunas que el cliente no ha pedido, lo que no implica que en un futuro podrían jugar un papel importante en la funcionalidad del software.

- **Feedback (Retroalimentación)**

La agilidad se define (entre otras cosas) por la capacidad de respuesta ante los cambios que se van haciendo necesarios a lo largo del camino. Por este motivo uno de los valores que nos hace más ágiles es el continuo seguimiento o feedback que recibimos a la hora de desarrollar en un entorno ágil de desarrollo. Este feedback se toma del cliente, de los miembros del equipo, en cuestión de todo el entorno en el que se mueve un equipo de desarrollo ágil.

Los valores anteriores inspiran los doce principios siguientes:

1. **Planificación incremental**

   La Programación Extrema asume que la planificación nunca será perfecta, y que variará en función de cómo varíen las necesidades del negocio. Por tanto, el valor real reside en obtener rápidamente un plan inicial, y contar con mecanismos de feedback que permitan conocer con precisión dónde estamos. Como es lógico, la planificación es iterativa: un representante del negocio decide al comienzo de cada iteración qué características concretas se van a implementar.

   El objetivo de la XP es generar versiones de la aplicación tan pequeñas como sea posible, pero que proporcionen un valor adicional, desde el punto de vista del negocio. A estas versiones se las denomina releases.

   Una release cuenta con un cierto número de historias. La historia es la unidad de funcionalidad en un proyecto XP, y corresponde a la mínima funcionalidad posible que tiene valor desde el punto de vista del negocio. Durante cada iteración se cierran varias historias, lo que hace que toda iteración añada un valor tangible para el cliente.

   Es fundamental en toda esta planificación la presencia de un representante del cliente, que forma parte del equipo y que decide cuáles son las historias más valiosas. Estas historias son las que se desarrollarán en la iteración actual.

   La obtención de feedback que permita llevar a cabo estimaciones precisas es fundamental. Se hacen estimaciones para cada historia, de modo que en cuanto se comienzan a tener datos históricos, éstos se utilizan para hacer que las siguientes estimaciones sean más precisas.

   Como se puede ver, y como siempre ocurre con la Programación Extrema, el enfoque utilizado para llevar a cabo la planificación es eminentemente pragmático. Gran parte de la eficacia de este modelo de planificación deriva de una división clara de responsabilidades, que tiene en cuenta las

necesidades del negocio en todo momento. Dentro de esta división, el representante del cliente tiene las siguientes responsabilidades:

- Decidir qué se implementa en cada release o iteración.
- Fijar las fechas de fin de la release, recortando unas características o añadiendo otras.
- Priorizar el orden de implementación, en función del valor de negocio.

Las responsabilidades del equipo de desarrollo son las siguientes:

- Estimar cuánto tiempo llevará una historia: este feedback es fundamental para el cliente, y puede llevarle a reconsiderar qué historias se deben incluir en una iteración.
- Proporcionar información sobre el coste de utilizar distintas opciones tecnológicas.
- Organizar el equipo.
- Estimar el riesgo de cada historia.
- Decidir el orden de desarrollo de historias dentro de la iteración.

2. Testing

La ejecución automatizada de tests es un elemento clave de la XP. Existen tanto tests internos (o tests de unidad), para garantizar que el mismo es correcto, como tests de aceptación, para garantizar que el código hace lo que debe hacer. El cliente es el responsable de definir los tests de aceptación, no necesariamente de implementarlos. Él es la persona mejor cualificada para decidir cuál es la funcionalidad más valiosa.

3. Programación en parejas

La XP incluye, como una de sus prácticas estándar, la programación en parejas. Nadie programa en solitario, siempre hay dos personas delante del ordenador. Ésta es una de las características que más se cuestiona al comienzo de la adopción de la XP dentro de un equipo, pero en la práctica se acepta rápidamente y de forma entusiasta.

4. Refactorización

A la hora de la verdad, el código de la mayor parte de las aplicaciones empieza en un razonable buen estado, para luego deteriorarse de forma progresiva. El coste desorbitado del mantenimiento, modificación y ampliación de aplicaciones ya existente se debe en gran parte a este hecho.

Uno de los objetivos de la XP es mantener la curva de costes tan plana como sea posible, por lo que existen una serie de mecanismos destinados a mantener el código en buen estado, modificándolo activamente para que conserve claridad y sencillez. A este proceso básico para mantener el código en buena forma se le llama refactorización.

La refactorización no sólo sirve para mantener el código legible y sencillo: también se utiliza cuando resulta conveniente modificar código existente para hacer más fácil implementar nueva funcionalidad.

5. Diseño simple

Otra práctica fundamental de la Programación Extrema es utilizar diseños tan simples como sea posible. El principio es "utilizar el diseño más sencillo que consiga que todo funcione". Se evita diseñar características extra porque a la hora de la verdad la experiencia indica que raramente se puede anticipar qué necesidades se convertirán en reales y cuáles no. La XP nos pide que no vivamos bajo la ilusión de que un diseño puede resolver todas o gran parte de las situaciones futuras: lo que parece necesario cambia con frecuencia, es difícil acertar a priori.

Es obvio que, si no se va a anticipar futuras necesidades, se debe poder modificar el diseño si alguna de estas se materializa. XP soporta estas modificaciones gracias a los tests automatizados. Estos permiten hacer cambios importantes gracias a la red de protección que proporcionan. La refactorización, que hace que el código existente sea claro y sencillo, también ayuda a hacer factibles las modificaciones.

La XP define un "diseño tan simple como sea posible" como aquél que:

- Pasa todos los tests.
- No contiene código duplicado.
- Deja clara la intención de los programadores (enfatiza el qué, no el cómo) en cada línea de código.

- Contiene el menor número posible de clases y métodos.

6. Propiedad colectiva del código

La XP aboga por la propiedad colectiva del código. En otras palabras, todo el mundo tiene autoridad para hacer cambios a cualquier código, y es responsable de ellos. Esto permite no tener que estar esperando a otros cuando todo lo que hace falta es algún pequeño cambio.

Por supuesto, cada cuál es responsable de las modificaciones que haga. El principio básico es "tú lo rompes, tú lo arreglas, no importa si está en el código propio o en el de otros".

Por último, vale la pena tener en cuenta que la existencia de tests automatizados impide que se produzca un desarrollo anárquico, al ser cada persona responsable de que todos los tests se ejecuten con éxito al incorporar los cambios que ha introducido al programa.

7. Integración continua

En muchos casos la integración de código produce efectos colaterales imprevistos, y en ocasiones la integración puede llegar a ser realmente traumática, cuando dejan de funcionar cosas por motivos desconocidos. La Programación Extrema hace que la integración sea permanente, con lo que todos los problemas se manifiestan de forma inmediata, en lugar de durante una fase de integración más o menos remota.

La existencia de una fase de integración separada tiene dos efectos colaterales indeseables: se empieza a hacer codificación "yo-yo", en la que todo el mundo modifica código "sólo para que funcione, ya lo ajustaremos", y hace que se acumulen defectos. Evitar que se acumulen defectos es muy importante para la XP, como lo es el conseguir que los defectos que cada programador inyecta los elimine él mismo.

8. Cliente en el equipo

Algunos de los problemas más graves en el desarrollo son los que se originan cuando el equipo de desarrollo toma decisiones de negocio críticas. Esto no debería ocurrir, pero a la hora de la verdad con frecuencia no se obtiene feedback del cliente con la fluidez necesaria: el resultado es que se ha de optar por detener el avance de los proyectos, o por que desarrollo tome una decisión de

negocio. Por otra parte, los representantes del negocio también suelen encontrarse con problemas inesperados debido a que tampoco reciben el feedback adecuado por parte de los desarrolladores.

9. **Releases pequeñas**

   Siguiendo la política de la XP de dar el máximo valor posible en cada momento, se intenta liberar nuevas versiones de las aplicaciones con frecuencia. Éstas deben ser tan pequeñas como sea posible, aunque deben añadir suficiente valor como para que resulten valiosas para el cliente.

10. **Semanas de 40 horas**

    La Programación Extrema lleva a un modo de trabajo en que el equipo está siempre al 100%. Una semana de 40 horas en las que se dedica la mayor parte del tiempo a tareas que suponen un avance puede dar mucho de sí, y hace innecesario recurrir a sobreesfuerzos -excepto en casos extremos.

    Además, el sobreesfuerzo continuado pronto lleva a un rendimiento menor y a un deterioro de la moral de todo el equipo.

11. **Estándares de codificación**

    Para conseguir que el código se encuentre en buen estado y que cualquier persona del equipo pueda modificar cualquier parte del código es imprescindible que el estilo de codificación sea consistente. Un estándar de codificación es necesario para soportar otras prácticas de la XP.

    Sin embargo, la XP también es pragmática en esto, y apuesta por establecer un número mínimo de reglas: el resto se irán pactando de-facto. Esto evita un ejercicio inicial más o menos estéril.

12. **Uso de Metáforas**

    La comunicación fluida es uno de los valores más importantes de la Programación Extrema: la programación en parejas, el hecho de incorporar al equipo una persona que represente los intereses del negocio y otras prácticas son valiosas entre otras cosas porque potencian enormemente la comunicación.

    Para conseguir que la comunicación sea fluida es imprescindible, entre otras cosas, utilizar el vocabulario del negocio. También es fundamental huir de definiciones abstractas. Dicho de otro modo, la XP no pretende seguir la letra de la ley, sino su espíritu. Dentro de este enfoque es

fundamental buscar continuamente metáforas que comuniquen intenciones y resulten descriptivas, enfatizando el qué por delante del cómo.(FRANKLIN RENE RIVERO GARCÍA 2006)

Dentro de XP los roles son:

- **Programador.** El programador elabora las pruebas unitarias y produce el código del sistema.
- **Cliente.** Escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.
- **Encargado de pruebas (Tester).** Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.
- **Encargado de seguimiento (Tracker).** Proporciona retroalimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración.
- **Entrenador (Coach).** Es responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.
- **Consultor.** Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas.
- **Gestor (Big boss).** Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.(VALENCIA 2006)

El proceso como tal en XP se desarrolla de la siguiente forma:

El ciclo de desarrollo consiste (a grandes rasgos) en los siguientes pasos:

1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación.
3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.

4. El programador construye ese valor de negocio.
5. Vuelve al paso 1.(VALENCIA 2006)

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad en el software o no se cumplirán los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurarse que el sistema tenga el mayor valor de negocio posible con cada iteración.

El ciclo de vida ideal de XP consiste de seis fases: Exploración, Planificación de la Entrega (Release), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto.

XP tiene una forma especial de tratar las funcionalidades del software a través de los requisitos los cuales son escritos por el cliente en las Historias de Usuarios.

Historia de Usuario:

Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas.

Las tarjetas de usuario tienen en su contenido: fecha, tipo de actividad (nueva, corrección, mejora), prueba funcional, número de historia, prioridad técnica y del cliente, referencia a otra historia previa, riesgo, estimación técnica, descripción, notas y una lista de seguimiento con la fecha, estado, cosas por terminar y comentarios. A efectos de planificación, las historias pueden ser de una a tres semanas de tiempo de programación (para no superar el tamaño de una iteración). Las historias de usuario son descompuestas en tareas de programación (task card) y asignadas a los programadores para ser implementadas durante una iteración.(VALENCIA 2006)

En la práctica, la principal suposición que se realiza en XP es la posibilidad de disminuir la mítica curva exponencial del costo del cambio a lo largo del proyecto, lo suficiente para que el diseño evolutivo funcione. Esto se consigue gracias a las tecnologías disponibles para ayudar en el desarrollo de software y a la aplicación disciplinada de las siguientes prácticas.

La mayoría de las prácticas propuestas por XP no son novedosas sino que en alguna forma ya habían sido propuestas en ingeniería del software e incluso demostrado su valor en la práctica. El mérito de XP es integrarlas de una forma efectiva y complementarlas con otras ideas desde la perspectiva del negocio, los valores humanos y el trabajo en equipo.

Para los diseñadores de juegos, XP trae consigo numerosas ventajas. Entre las que se encuentran su flexibilidad para el cambio constante de requisitos y funcionalidades del sistema; se alega que no se sabe en el transcurso de la elaboración del videojuego en si, como se comportará en los diferentes niveles, así como el nivel de dificultad. En caso que se realizara el diseño mediante una metodología tradicional sería bastante complicado cambiar una funcionalidad, pues implicaría modificar toda la documentación que requiere el caso de uso, y al estar en constante movimiento la fase de implementación, el proyecto sería interminable. Con esta ventaja colabora directamente otra de las características nombradas de XP: la comunicación y participación constante del cliente, el cual forma parte del equipo de trabajo y tendría al final del producto pocas posibilidades de quedar insatisfecho, pues en la terminación del sistema influyó notablemente su propio esfuerzo.

Otra ventaja planteada por los desarrolladores de Entornos Virtuales es la escasez de documentación, esto agilizaría el trabajo, ya que este tipo de sistema lleva consigo una gran parte de implementación y algoritmización, dado por su importante estudio físico-matemático, elementos de inteligencia artificial y otros, por lo que la burocracia le robaría mucho tiempo, y dentro de los logros que se desea alcanzar con las metodologías ágiles es precisamente, no detenerse en elementos innecesarios.

Entrega constante de entregables, lo que de igual forma ameniza la interacción con el cliente y el mismo tiene una noción, a tiempo entero, del desarrollo del proyecto, y al mismo tiempo en cada pequeña iteración va obteniendo un entregable.

Mediante la especialización de los roles permite que cada trabajador se sienta cómodo en el proyecto, permitiendo, por ejemplo, que si el equipo de trabajo cuenta con un programador que es mejor programando un tipo de estructura de datos, realizará este trabajo para todos los diferentes módulos. Esto lo permite la política de la sencillez del código, por lo que todos los programadores pueden laborar en todos los módulos, incluso sin haberlo hecho en la iteración anterior.

Defiende el trabajo en equipo, valora la importancia del hombre, no de la herramienta, por lo que requiere de desarrolladores bien preparados, vota por la integración del equipo, haciendo así el trabajo

más ameno y confiado. Mediante la programación en pareja, saca mejor provecho a la computadora y al hombre: “dos cerebros piensan mejor que uno”.

- El hecho de que todas las decisiones las tomen al menos dos personas proporciona un mecanismo de seguridad enormemente valioso.
- Con dos personas responsabilizándose del código en cada momento, es menos probable que se caiga en la tentación de dejar de escribir tests, etc., algo fundamental para mantener el código en buena forma. Es muy difícil que dos personas se salten tareas por descuido o negligencia.
- El hecho de programar en parejas permite la dispersión de know-how[1] por todo el equipo. Este efecto es difícil de conseguir de otro modo, y hace que la incorporación de nuevos miembros al equipo sea mucho más rápida y eficaz.
- El código siempre está siendo revisado por otra persona. La revisión de código es el método más eficaz de conseguir código de calidad, algo corroborado por numerosos estudios, muchos de los cuáles son anteriores a la Programación Extrema.
- En contra de lo que pueda parecer, los dos desarrolladores no hacen lo mismo: mientras el que tiene el teclado adopta un rol más táctico, el otro adopta un rol más estratégico, preguntándose constantemente si lo que se está haciendo tiene sentido desde un punto de vista global.

Los datos indican que la programación en parejas es realmente más eficiente. Si bien se sacrifica un poco de velocidad al comienzo, luego se obtiene una velocidad de crucero muy superior. Esto contrasta con lo que ocurre en la mayor parte de los proyectos, en los que se arranca con una velocidad enorme pero rápidamente se llega a un estado muy parecido a la parálisis, en el que progresos cada vez más pequeños consumen cantidades de tiempo cada vez más grandes. Todos conocemos proyectos que se pasan el 50% del tiempo en el estado de "finalizado al 90%".

Pero como todo, XP no es siempre color de rosas, al igual que todo lo creado por el hombre presenta baches, lo que implicaría para desarrolladores inexpertos un caos total de no ser lo suficientemente cuidadosos, y son precisamente sus ventajas las que en forma de boomerang podrían realizar una mala jugada. Es el caso de la escasa documentación, en un proyecto guiado por XP no se tiene control de casi

[1] Se traduce literalmente por "saber-cómo", mejor dicho sería "Saber hacer"

nada, pues solo se tiene constancia de las tarjetas de usuario, por lo que al final el sistema se entrega casi sin documentación.

En su uso común, el diseño evolutivo es un desastre. El diseño acaba siendo la agregación de una sarta de decisiones tácticas ad-hoc[2], cada una de las cuales hace el código más difícil de modificar. Se podría alegar que eso no es diseño, ciertamente suele llevar a un diseño pobre. El diseño está para permitir cambiar el software fácilmente a largo plazo. Conforme el diseño se deteriora, igualmente se deteriora la capacidad de cambio.

No hay lugar para los patrones de diseño en XP. La relación entre patrones de diseño y la XP es interesante y es una cuestión común. Son muchos los expertos en desarrollo de software que consideran que los patrones son subestimados en la XP. La idea de XP es que los patrones son frecuentemente sobre utilizados. Una teoría es que las fuerzas del diseño simple te llevarán a los patrones. Muchas refactorizaciones hacen esto explícitamente, pero aún sin ellas siguiendo las reglas del diseño simple llegarás a los patrones aun si no los conoces previamente. Lo que se propone es el uso de los patrones de manera planeada cuando se hace el diseño planeado, muy bien, dejarse llevar por el diseño simple, pero si desde la planeación se ve venir un patrón determinado no temer usarlo al final nos ahorra correr por caminos alternativos hasta llegar a él, una premisa muy importante es que los diseñadores tengan un profundo conocimiento de los patrones y experiencia en el trabajo con estos. El objetivo no es usar 10 patrones en 30 líneas de código, sino que si usamos 5 patrones estos estén bien justificados y sigan la premisa del diseño simple. Cuatro cosas a tener en cuenta cuando se trabaja con XP:

1. Invertir tiempo en aprender sobre patrones
2. Concentrarse en cuándo aplicarlos
3. Concentrarse en cómo implementar el patrón en su forma más simple primero, y añadir complejidad luego.
4. No temer eliminar un patrón si no está valiendo su peso.

XP ignora la arquitectura, la ruta de XP es ir hacia la codificación rápida y confiar en que la refactoración resolverá todos los problemas de diseño. No pongas una base de datos hasta que realmente sepas que la

[2] "para esto", herramienta elaborada específicamente para una determinada ocasión o situación.

necesitas. Trabaja con archivos primero y refactora a la base de datos en una iteración posterior. Este es un punto en el que se debe reflexionar, siempre hay espacio para una arquitectura inicial, temas como la estructura del software en términos generales, la organización, la comunicación entre las partes, son cosas que no se pueden dejar en el aire para que surjan en el camino. Definir una arquitectura base a un nivel bien alto y sin mucha especificación guiará el proceso por un mejor camino y dará mejores resultados. Un aspecto importante en este punto es el papel del arquitecto en XP, ¿para que un arquitecto si no se tiene arquitectura? En realidad en XP el arquitecto seria el Coach, ese desarrollador de gran experiencia que está todo el tiempo junto a los programadores de menor nivel ayudando en lo que se necesite, aclarando dudas, pasando la experiencia adquirida durante años de trabajo a los desarrolladores que recién se unen al proyecto.

"Los XPeros no diagraman", frase común entre los adeptos mas acérrimos de la XP. Si se va a utilizar algo de diseño planeado, se van a usar patrones y se va a concebir una arquitectura inicial, se van a necesitar diagramas, como tener un mapa si no se sabe como leerlo, los diagramas ayudaran en gran medida a la comunicación entre los miembros del equipo, ya no será solamente el código el que hable. En este punto es donde entra el UML, lenguaje mas que difundido y usado en el mundo. No es tampoco abusar de los diagramas y llenar nuestro proceso de documentación inútil, es simplemente señalar en el mapa los detalles más significativos del camino a seguir.

En forma de conclusiones podríamos resumir que XP independientemente de sus lagunas es una importante metodología a tener en cuenta para el desarrollo de entornos virtuales, compañías importantes como EA-Games guían su proceso a través de la programación extrema. En la Universidad de Ciencias Informáticas el proyecto "Paseo Virtual" de la facultad 5, trabaja en la elaboración del juego "Laberinto del Saber 2" y guía su proceso a partir de XP, siendo esta una experiencia experimental, aunque van teniendo buenos resultados.

Otras Metodologías:

Varias son las personas que se han dado cuenta de la necesidad de implementar una nueva metodología para los Entornos Virtuales (EVs), o en caso contrario, adaptar una de las existentes, soluciones que en ninguno de los dos casos es tarea fácil. Algunos desarrolladores alegan que las metodologías orientadas a objetos serían las más adecuadas para adaptar, ya que la realidad virtual

consiste en un grupo de objetos interactuando entre si, los cuales contienen atributos y los métodos serían las acciones y reacciones con el resto del entorno.

Por encima de todo, uno de los principales objetivos de la Ingeniería de software debe ser la búsqueda de soluciones simples a los problemas, por lo que sería conveniente dedicarle un poco más de tiempo para ver si se encuentra una solución adecuada al problema de la Realidad Virtual.

Dentro de las diferencias notablemente evidentes entre los Entornos Virtuales y el resto de las aplicaciones es el hecho de que los EVs son eminentemente visuales, así que es precisamente hacía esta línea a donde debe ir dedicado la mayor parte del esfuerzo.

Una de las ventajas encontradas al paradigma de la programación orientada a objetos es el hecho de que un buen código es reutilizable, y en los EV un objeto es construido con el objetivo de que pueda ser reutilizado o para mejorar otro.

A continuación se realiza un estudio del método de Larman el cual emplea el UML por ser el mismo un lenguaje de modelado orientado a objeto. Con esta metodología también se da inicio al estudio de las "Metodologías Tradicionales". Los desarrolladores que hacen uso de este método están concientes que no cumple con todas sus necesidades como por ejemplo pasar por alto el desarrollo de los elementos tridimensionales.

1.3.2 Metodología de Larman:

Al igual que otras metodologías tradicionales la Metodología de Larman es iterativo e incremental, además de usar UML por su gran flexibilidad a la hora de agregar y quitar elementos del diseño. Teniendo en cuenta que ha sido utilizada para procesos de software implicados en el desarrollo de entornos virtuales, formando esta parte del estado del arte, se valora la necesidad de realizar un estudio de sus características, enfatizando sus ventajas y desventajas para software de tres dimensiones (3D). Entre sus características consta, durante su proceso, con un ciclo de vida de cuatro flujos de trabajo:

- **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.
- Diagrama de secuencia del sistema.
- Contratos de operaciones.
- Diagrama 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 prueba.** (GONZALO MÉNDEZ POSO)

En esta etapa no se cuenta con una documentación definida.

Planificación y Especificación de requisitos:

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

Especificación de Requisitos:

La experiencia ha mostrado que no es necesario especificar todos los requisitos desde el principio para poder comenzar el análisis, debido a 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.

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[3] 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

[3] Imagen o icono que representa a una persona en relaciones por Internet o entornos virtuales compartidos.

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*. Los *conceptos de uso* no aparecen en la especificación de UML 1.3. 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 basa en la cantidad de información disponible. Como resultado, se desarrollan 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.

Primer Ciclo de Desarrollo:

Los elementos a desarrollar en este ciclo corresponden a los casos de uso obtenidos a partir de los requisitos iniciales. (GONZALO MÉNDEZ POSO)

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. (GONZALO MÉNDEZ POSO)

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, 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”, 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.(GONZALO MÉNDEZ POSO)

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.(GONZALO MÉNDEZ POSO)

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 decirles 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 objetos.

- Para los objetos, su forma y las partes que no pueden simularse con texturas. (GONZALO MÉNDEZ POSO)

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. (GONZALO MÉNDEZ POSO)

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.

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.

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 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.

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.

Conclusión 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. (GONZALO MÉNDEZ POSO)

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.

1.3.3 Proceso Unificado del Racional (RUP):

Dentro del mundo de la ingeniería del software El Proceso Unificado del Rational ha ocupado un lugar importante, siendo esta, una de las metodologías más usadas para la construcción de sistemas. No se puede pasar por alto a pesar de conocerse de pocas experiencias de su uso para el desarrollo de EVs.

El Proceso Unificado de Rational (RUP, el original inglés *Rational Unified Process*) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a

objetos. RUP es en realidad un refinamiento realizado por Rational Software del más genérico Proceso Unificado.

El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

Características esenciales

Los autores de RUP destacan que el proceso de software propuesto por RUP tiene tres características esenciales: está dirigido por los Casos de Uso, centrado en la arquitectura y es iterativo e incremental.

Proceso dirigido por Casos de Uso

Se define un Caso de Uso como un fragmento de funcionalidad del sistema que proporciona al usuario un valor añadido. Los Casos de Uso representan los requisitos funcionales del sistema. Guían su diseño, implementación y prueba. Constituyen un elemento integrador y una guía del trabajo. Los Casos de Uso no sólo inician el proceso de desarrollo sino que proporcionan un hilo conductor, permitiendo establecer trazabilidad entre los artefactos que son generados en las diferentes actividades del proceso de desarrollo. (JAMES RUMBAUGH 2000)

Proceso centrado en la arquitectura

La arquitectura de un sistema es la organización o estructura de sus partes más relevantes, lo que permite tener una visión común entre todos los involucrados (desarrolladores y usuarios) y una perspectiva clara del sistema completo, necesaria para controlar el desarrollo. Involucra los aspectos estáticos y dinámicos más significativos del sistema, está relacionada con la toma de decisiones que indican cómo tiene que ser construido el sistema y ayuda a determinar en qué orden. Además la definición de la arquitectura debe tomar en consideración elementos de calidad del sistema, rendimiento, reutilización y capacidad de evolución por lo que debe ser flexible durante todo el proceso de desarrollo. La arquitectura se ve influenciada por la plataforma software, sistema operativo, gestor de bases de datos, protocolos, consideraciones de desarrollo como sistemas heredados. Muchas de estas restricciones constituyen requisitos no funcionales del sistema. Cada producto tiene tanto una función como una forma. La función corresponde a la funcionalidad reflejada en los Casos de Uso y la forma la proporciona la arquitectura. La arquitectura se representa mediante varias vistas que se centran en aspectos concretos del sistema, abstrayéndose de los demás. Para RUP, todas las vistas juntas forman el llamado modelo 4+1 de la

arquitectura el cual recibe este nombre porque lo forman las vistas lógica, de implementación, de proceso y de despliegue, más la de Casos de Uso que es la que da cohesión a todas. Al final de la fase de elaboración se obtiene una *línea base* de la arquitectura donde fueron seleccionados una serie de Casos de Uso arquitectónicamente relevantes (aquellos que ayudan a mitigar los riesgos más importantes, aquellos que son los más importantes para el usuario y aquellos que cubran las funcionalidades significativas)(JAMES RUMBAUGH 2000)

Proceso iterativo e incremental

RUP es un proceso iterativo e incremental, donde el trabajo se divide en partes más pequeñas o mini proyectos. Permitiendo que el equilibrio entre Casos de Uso y arquitectura se vaya logrando durante cada mini proyecto, así durante todo el proceso de desarrollo. Cada mini proyecto se puede ver como una iteración (un recorrido más o menos completo a lo largo de todos los flujos de trabajo fundamentales) del cual se obtiene un incremento que produce un crecimiento en el producto. Una iteración puede realizarse por medio de una cascada. Se pasa por los flujos fundamentales (Requisitos, Análisis, Diseño, Implementación y Pruebas), también existe una planificación de la iteración, un análisis de la iteración y algunas actividades específicas de la iteración. Al finalizar se realiza una integración de los resultados con lo obtenido de las iteraciones anteriores.

El proceso iterativo e incremental consta de una secuencia de iteraciones. Cada iteración se analiza cuando termina. Se puede determinar si han aparecido nuevos requisitos o han cambiado los existentes, afectando a las iteraciones siguientes. Durante la planificación de los detalles de la siguiente iteración, el equipo también examina cómo afectarán los riesgos que aún quedan al trabajo en curso. Toda la retroalimentación de la iteración pasada permite reajustar los objetivos para las siguientes iteraciones. Se continúa con esta dinámica hasta que se haya finalizado por completo con la versión actual del producto. (JAMES RUMBAUGH 2000)

Principios de desarrollo

El RUP está basado en 6 principios clave:

Adaptar el proceso

El proceso deberá adaptarse a las características propias del proyecto u organización. El tamaño del mismo, así como su tipo o las regulaciones que lo condicionen, influirán en su diseño específico.

Balancear prioridades

Los requerimientos de los diversos inversores pueden ser diferentes, contradictorios o disputarse recursos limitados. Debe encontrarse un balance que satisfaga los deseos de todos.

Colaboración entre equipos

El desarrollo de software no lo hace una única persona sino múltiples equipos. Debe haber una comunicación fluida para coordinar requerimientos, desarrollo, evaluaciones, planes, resultados, etc.

Demostrar valor iterativamente.

Los proyectos se entregan, aunque sea de un modo interno, en etapas iteradas. En cada iteración se analiza la opinión de los clientes, la estabilidad y calidad del producto, y se refina la dirección del proyecto.

Elevar el nivel de abstracción.

Este principio dominante motiva el uso de conceptos reutilizables tales como patrón del software, lenguajes 4GL o esquemas (frameworks) por nombrar algunos. Esto previene a los ingenieros de software ir directamente de los requisitos a la codificación de software a la medida del cliente. Un nivel alto de abstracción también permite discusiones sobre diversos niveles arquitectónicos. Éstos se pueden acompañar por las representaciones visuales de la arquitectura, por ejemplo con UML.

Enfocarse en la calidad.

El aseguramiento de calidad no debe realizarse al final de cada iteración, sino en todos los aspectos de la producción

El RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al final de cada ciclo, cada ciclo se divide en fases que finalizan con un hito donde se debe tomar una decisión importante:

Fases del RUP

RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en los distintas actividades.(VALENCIA 2006)

Inicio

En las fases de Inicio las primeras iteraciones se enfocan hacia la comprensión del problema y la tecnología, la delimitación del ámbito del proyecto, la eliminación de los riesgos críticos, y al establecimiento de una *línea base* de la arquitectura. Durante la fase de inicio las iteraciones ponen mayor énfasis en actividades de modelado del negocio y de requisitos.

Durante la fase de inicio se define el modelo del negocio y el alcance del proyecto. Se identifican todos los actores y Casos de Uso, y se diseñan los Casos de Uso más esenciales (aproximadamente el 20% del modelo completo). Se desarrolla, un plan de negocio para determinar que recursos deben ser asignados al proyecto.

Los objetivos de esta fase son:

- Establecer el ámbito del proyecto y sus límites.
- Encontrar los Casos de Uso críticos del sistema, los escenarios básicos que definen la funcionalidad.
- Mostrar al menos una arquitectura candidata para los escenarios principales.
- Estimar el coste en recursos y tiempo de todo el proyecto.
- Estimar los riesgos, las fuentes de incertidumbre.

Los resultados de la fase de inicio deben ser:

- Un documento de visión: Una visión general de los requerimientos del proyecto, características clave y restricciones principales.
- Modelo inicial de Casos de Uso (10-20% completado).
- Un glosario inicial: Terminología clave del dominio.
- El caso de negocio.
- Lista de riesgos y plan de contingencia.
- Plan del proyecto, mostrando fases e iteraciones.
- Modelo de negocio, si es necesario

- Prototipos exploratorios para probar conceptos o la arquitectura candidata.

Al terminar la fase de inicio se deben comprobar los criterios de evaluación para continuar:

- Todos los interesados en el proyecto coinciden en la definición del ámbito del sistema y las estimaciones de agenda.
- Entendimiento de los requisitos, como evidencia de la fidelidad de los Casos de Uso principales.
- Las estimaciones de tiempo, coste y riesgo son creíbles.
- Comprensión total de cualquier prototipo de la arquitectura desarrollado.
- Los gastos hasta el momento se asemejan a los planeados.

Si el proyecto no pasa estos criterios hay que plantearse abandonarlo o repensarlo profundamente.

Elaboración

En la fase de elaboración, las iteraciones se orientan al desarrollo de la *línea base* de la arquitectura, abarcan más los flujos de trabajo de requerimientos, modelo de negocios (refinamiento), análisis, diseño y una parte de implementación orientado a la *línea base* de la arquitectura. El propósito de esta fase es analizar el dominio del problema, establecer los cimientos de la arquitectura, desarrollar el plan del proyecto y eliminar los mayores riesgos.

Se construye un prototipo de la arquitectura, que debe evolucionar en iteraciones sucesivas hasta convertirse en el sistema final. Este prototipo debe contener los Casos de Uso críticos identificados en la fase de inicio. También debe demostrarse que se han evitado los riesgos más graves.

Los objetivos de esta fase son:

- Definir, validar y cimentar la arquitectura.
- Completar la visión.
- Crear un plan fiable para la fase de construcción. Este plan puede evolucionar en sucesivas iteraciones. Debe incluir los costes si procede.
- Demostrar que la arquitectura propuesta soportará la visión con un coste razonable y en un tiempo razonable.

Al terminar deben obtenerse los siguientes resultados:

- Un modelo de Casos de Uso completa al menos hasta el 80%: todos los casos y actores identificados, la mayoría de los casos desarrollados.
- Requisitos adicionales que capturan los requisitos no funcionales y cualquier requisito no asociado con un Caso de Uso específico.
- Descripción de la arquitectura software.
- Un prototipo ejecutable de la arquitectura.
- Lista de riesgos y caso de negocio revisados.
- Plan de desarrollo para el proyecto.
- Un caso de desarrollo actualizado que especifica el proceso a seguir.
- Un manual de usuario preliminar (opcional).

En la fase de elaboración se actualizan todos los productos de la fase de inicio.

Los criterios de evaluación de esta fase son los siguientes:

- La visión del producto es estable.
- La arquitectura es estable.
- Se ha demostrado mediante la ejecución del prototipo que los principales elementos de riesgo han sido abordados y resueltos.
- El plan para la fase de construcción es detallado y preciso. Las estimaciones son creíbles.
- Todos los interesados coinciden en que la visión actual será alcanzada si se siguen los planes actuales en el contexto de la arquitectura actual.
- Los gastos hasta ahora son aceptables, comparados con los previstos.

Construcción

En la fase de construcción, se lleva a cabo la construcción del producto por medio de una serie de iteraciones. Para cada iteración se selecciona algunos Casos de Uso, se refina su análisis y diseño y se

procede a su implementación y pruebas. Se realiza una pequeña cascada para cada ciclo. Se realizan tantas iteraciones hasta que se termine la implementación de la nueva versión del producto. La finalidad principal de esta fase es alcanzar la capacidad operacional del producto.

Los objetivos concretos son:

- Minimizar los costes de desarrollo mediante la optimización de recursos y evitando el tener que rehacer un trabajo o incluso desecharlo.
- Conseguir una calidad adecuada tan rápido como sea práctico.
- Conseguir versiones funcionales (alfa, beta, y otras versiones de prueba) tan rápido como sea práctico.

Los resultados de la fase de construcción deben ser:

- Modelos Completos (Casos de Uso, Análisis, Diseño, Despliegue e Implementación)
- Arquitectura íntegra (mantenida y mínimamente actualizada)
- Riesgos Presentados Mitigados
- Plan del Proyecto para la fase de Transición.
- Manual Inicial de Usuario (con suficiente detalle)
- Prototipo Operacional – beta
- Caso del Negocio Actualizado

Los criterios de evaluación de esta fase son los siguientes:

- El producto es estable y maduro como para ser entregado a la comunidad de usuario para ser probado.
- Todos los usuarios expertos están listos para la transición en la comunidad de usuarios.
- Son aceptables los gastos actuales versus los gastos planeados.

Transición

En la fase de transición se pretende garantizar que se tiene un producto preparado para su entrega a la comunidad de usuarios, para lo que se requiere desarrollar nuevas versiones actualizadas del producto, completar la documentación, entrenar al usuario en el manejo del producto, y en general tareas relacionadas con el ajuste, configuración, instalación y facilidad de uso del producto.

Se puede incluir esta fase:

- Prueba de la versión Beta para validar el nuevo sistema frente a las expectativas de los usuarios
- Funcionamiento paralelo con los sistemas legados que están siendo sustituidos por nuestro proyecto.
- Conversión de las bases de datos operacionales.
- Entrenamiento de los usuarios y técnicos de mantenimiento.
- Traspaso del producto a los equipos de marketing, distribución y venta.

Los principales objetivos de esta fase son:

- Conseguir que el usuario se valga por si mismo.
- Un producto final que cumpla los requisitos esperados, que funcione y satisfaga suficientemente al usuario.

Los resultados de la fase de transición son:

- Prototipo Operacional
- Documentos Legales
- Caso del Negocio Completo
- Línea de Base del Producto completa y corregida que incluye todos los modelos del sistema
- Descripción de la Arquitectura completa y corregida
- Las iteraciones de esta fase irán dirigidas normalmente a conseguir una nueva versión.

Los criterios de evaluación de esta fase son los siguientes:

- El usuario se encuentra satisfecho.
- Son aceptables los gastos actuales versus los gastos planificados.

RUP también es identificado por sus 6 principales prácticas la cual define de forma efectiva el trabajo del equipo de desarrollo de Software:

Gestión de requisitos

RUP brinda una guía para encontrar, organizar, documentar, y seguir los cambios de los requisitos funcionales y restricciones. Utiliza una notación de Caso de Uso y escenarios para representar los requisitos.

Desarrollo de software iterativo

Desarrollo del producto mediante iteraciones con hitos bien definidos, en las cuales se repiten las actividades pero con distinto énfasis, según la fase del proyecto.

Desarrollo basado en componentes

La creación de sistemas intensivos en software requiere dividir el sistema en componentes con interfaces bien definidas, que posteriormente serán ensamblados para generar el sistema. Esta característica en un proceso de desarrollo permite que el sistema se vaya creando a medida que se obtienen o se desarrollan sus componentes.

Modelado visual (usando UML)

UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema software. Es un estándar del Grupo de Administración de Objetos, con siglas en ingles OMG (http://www.omg.org). Utilizar herramientas de modelado visual facilita la gestión de dichos modelos, permitiendo ocultar o exponer detalles cuando sea necesario. El modelado visual también ayuda a mantener la consistencia entre los artefactos del sistema: requisitos, diseños e implementaciones. En resumen, el modelado visual ayuda a mejorar la capacidad del equipo para gestionar la complejidad del software.

Verificación continúa de la calidad

Es importante que la calidad de todos los artefactos se evalúe en varios puntos durante el proceso de desarrollo, especialmente al final de cada iteración. En esta verificación las pruebas juegan un papel fundamental y se integran a lo largo de todo el proceso. Para todos los artefactos no ejecutables las revisiones e inspecciones también deben ser continuas.

Gestión de los cambios

El cambio es un factor de riesgo crítico en los proyectos de software. Los artefactos software cambian no sólo debido a acciones de mantenimiento posteriores a la entrega del producto, sino que durante el proceso de desarrollo, especialmente importantes por su posible impacto son los cambios en los requisitos. Por otra parte, otro gran desafío que debe abordarse es la construcción de software con la participación de múltiples desarrolladores, posiblemente distribuidos geográficamente, trabajando a la vez en una *release*, y quizás en distintas plataformas. La ausencia de disciplina rápidamente conduciría al caos. La Gestión de Cambios y de Configuración es la disciplina de RUP encargada de este aspecto.

Roles:

RUP define grupos de roles, agrupados por participación en actividades relacionadas. Estos grupos son:

Analistas:

- Analista de procesos de negocio.
- Diseñador del negocio.
- Analista de sistema.
- Especificador de requisitos.

Desarrolladores:

- Arquitecto de software.
- Diseñador
- Diseñador de interfaz de usuario
- Diseñador de cápsulas.

- Diseñador de base de datos.
- Implementador.
- Integrador.

Gestores:

- Jefe de proyecto
- Jefe de control de cambios.
- Jefe de configuración.
- Jefe de pruebas
- Jefe de despliegue
- Ingeniero de procesos
- Revisor de gestión del proyecto
- Gestor de pruebas.

Apoyo:

- Documentador técnico
- Administrador de sistema
- Especialista en herramientas
- Desarrollador de cursos
- Artista gráfico

Especialista en pruebas:

- Especialista en Pruebas (*tester*)

- Analista de pruebas
- Diseñador de pruebas

Otros roles:

- *Stakeholders (mantenedores de estado)*.
- Revisor
- Coordinación de revisiones
- Revisor técnico
- Cualquier rol (VALENCIA 2006)

Flujos de trabajo

Con la enumeración de roles, actividades y artefactos no se define un proceso, se necesita contar con una secuencia de actividades realizadas por los diferentes roles, así como la relación entre los mismos. Un flujo de trabajo es una relación de actividades que nos producen unos resultados observables. A continuación se dará una explicación de cada flujo de trabajo.

Modelado del negocio

Con este flujo de trabajo se pretende llegar a un mejor entendimiento de la organización donde se va a implantar el producto.

Los objetivos del modelado de negocio son:

- Entender la estructura y la dinámica de la organización para la cual el sistema va ser desarrollado (organización objetivo).
- Entender el problema actual en la organización objetivo e identificar potenciales mejoras.
- Asegurar que clientes, usuarios finales y desarrolladores tengan un entendimiento común de la organización objetivo.

- Derivar los requisitos del sistema necesarios para apoyar a la organización objetivo.

Para lograr estos objetivos, el modelo de negocio describe como desarrollar una visión de la nueva organización, basado en esta visión se definen procesos, roles y responsabilidades de la organización por medio de un modelo de Casos de Uso del negocio y un Modelo de Objetos del Negocio. Complementario a estos modelos, se desarrollan otras especificaciones tales como un Glosario.

Requisitos:

Este es uno de los flujos de trabajo más importantes, porque en él se establece qué tiene que hacer exactamente el sistema que se desea construir. En esta línea los requisitos son el contrato que se debe cumplir, de modo que los usuarios finales tienen que comprender y aceptar los requisitos que especifiquemos.

Los objetivos del flujo de datos Requisitos son:

- Establecer y mantener un acuerdo entre clientes y otros *stakeholders* sobre lo que el sistema podría hacer.
- Proveer a los desarrolladores un mejor entendimiento de los requisitos del sistema.
- Definir el ámbito del sistema.
- Proveer una base para la planeación de los contenidos técnicos de las iteraciones.
- Proveer una base para estimar costos y tiempo de desarrollo del sistema.
- Definir una interfaz de usuarios para el sistema, enfocada a las necesidades y metas del usuario.

Los requisitos se dividen en dos grupos. Los requisitos funcionales representan la funcionalidad del sistema. Se modelan mediante diagramas de Casos de Uso. Los requisitos no funcionales representan aquellos atributos que debe exhibir el sistema, pero que no son una funcionalidad específica. Por ejemplo requisitos de facilidad de uso, fiabilidad, eficiencia, portabilidad, etc.

Para capturar los requisitos es preciso entrevistar a todos los interesados en el proyecto, no sólo a los usuarios finales, y anotar todas sus peticiones. A partir de ellas hay que descubrir lo que necesitan y expresarlo en forma de requisitos.

En este flujo de trabajo, y como parte de los requisitos de facilidad de uso, se diseña la interfaz gráfica de usuario. Para ello habitualmente se construyen prototipos de la interfaz gráfica de usuario que se contrastan con el usuario final.

Análisis y Diseño

El objetivo de este flujo de trabajo es traducir los requisitos a una especificación que describa cómo implementar el sistema.

Los objetivos del análisis y diseño son:

- Transformar los requisitos al diseño del futuro sistema.
- Desarrollar una arquitectura para el sistema.
- Adaptar el diseño para que sea consistente con el entorno de implementación, diseñando para el rendimiento.

El análisis consiste en obtener una visión del sistema que se preocupa de ver qué hace, de modo que sólo se interesa por los requisitos funcionales. Por otro lado el diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva cómo cumple el sistema sus objetivos.

Al principio de la fase de elaboración hay que definir una arquitectura candidata: crear un esquema inicial de la arquitectura del sistema, identificar clases de análisis y actualizar las realizaciones de los Casos de Uso con las interacciones de las clases de análisis. Durante la fase de elaboración se va refinando esta arquitectura hasta llegar a su forma definitiva. En cada iteración hay que analizar el comportamiento para diseñar componentes. Además si el sistema usará una base de datos, habrá que diseñarla también, obteniendo un modelo de datos.

El resultado final más importante de este flujo de trabajo será el modelo de diseño. Consiste en colaboraciones de clases, que pueden ser agregadas en paquetes y subsistemas.

Otro producto importante de este flujo es la documentación de la arquitectura de software, que captura varias vistas arquitectónicas del sistema.

Implementación

En este flujo de trabajo se implementan las clases y objetos en ficheros fuente, binarios, ejecutables y demás. También se deben hacer las pruebas de unidad: cada programador es responsable de probar las unidades que produzca. El resultado final de este flujo de trabajo es un sistema ejecutable.

En cada iteración habrá que hacer lo siguiente:

- Planificar qué subsistemas deben ser implementados y en que orden deben ser integrados, formando el Plan de Integración.
- Cada programador decide en que orden implementa los elementos del subsistema.
- Si encuentra errores de diseño, los notifica.
- Se prueban los subsistemas individualmente.
- Se integra el sistema siguiendo el plan.

La estructura de todos los elementos implementados forma el modelo de implementación. La integración debe ser incremental, es decir, en cada momento sólo se añade un elemento. De este modo es más fácil localizar fallos y los componentes se prueban más a fondo. En fases tempranas del proceso se pueden implementar prototipos para reducir el riesgo. Su utilidad puede ir desde ver si el sistema es viable desde el principio, probar tecnologías o diseñar la interfaz de usuario. Los prototipos pueden ser exploratorios (desechables) o evolutivos. Estos últimos llegan a transformarse en el sistema final.

Pruebas

Este flujo de trabajo es el encargado de evaluar la calidad del producto que estamos desarrollando, pero no para aceptar o rechazar el producto al final del proceso de desarrollo, sino que debe ir integrado en todo el ciclo de vida.

Esta disciplina brinda soporte a las otras disciplinas. Sus objetivos son:

- Encontrar y documentar defectos en la calidad del software.
- Generalmente asesora sobre la calidad del software percibida.
- Provee la validación de los supuestos realizados en el diseño y especificación de requisitos por medio de demostraciones concretas.

- Verificar las funciones del producto de software según lo diseñado.
- Verificar que los requisitos tengan su apropiada implementación.

Las actividades de este flujo comienzan pronto en el proyecto con el plan de prueba (el cual contiene información sobre los objetivos generales y específicos de las prueba en el proyecto, así como las estrategias y recursos con que se dotará a esta tarea), o incluso antes con alguna evaluación durante la fase de inicio, y continuará durante todo el proyecto.

El desarrollo del flujo de trabajo consistirá en planificar que es lo que hay que probar, diseñar cómo se va a hacer, implementar lo necesario para llevarlos a cabo, ejecutarlos en los niveles necesarios y obtener los resultados, de forma que la información obtenida nos sirva para ir refinando el producto a desarrollar.

Despliegue

El objetivo de este flujo de trabajo es producir con éxito distribuciones del producto y distribuirlo a los usuarios. Las actividades implicadas incluyen:

- Probar el producto en su entorno de ejecución final.
- Empaquetar el software para su distribución.
- Distribuir el software.
- Instalar el software.
- Proveer asistencia y ayuda a los usuarios.
- Formar a los usuarios y al cuerpo de ventas.
- Migrar el software existente o convertir bases de datos.

Este flujo de trabajo se desarrolla con mayor intensidad en la fase de transición, ya que el propósito del flujo es asegurar una aceptación y adaptación sin complicaciones del software por parte de los usuarios. Su ejecución inicia en fases anteriores, para preparar el camino, sobre todo con actividades de planificación, en la elaboración del manual de usuario y tutoriales.

Gestión del proyecto

La Gestión del proyecto es el arte de lograr un balance al gestionar objetivos, riesgos y restricciones para desarrollar un producto que sea acorde a los requisitos de los clientes y los usuarios.

Los objetivos de este flujo de trabajo son:

- Proveer un marco de trabajo para la gestión de proyectos de software intensivos.
- Proveer guías prácticas realizar planeación, contratar personal, ejecutar y monitorear el proyecto.
- Proveer un marco de trabajo para gestionar riesgos.

La planeación de un proyecto posee dos niveles de abstracción: un plan para las fases y un plan para cada iteración.

Configuración y control de cambios

La finalidad de este flujo de trabajo es mantener la integridad de todos los artefactos que se crean en el proceso, así como de mantener información del proceso evolutivo que han seguido.

Entorno

La finalidad de este flujo de trabajo es dar soporte al proyecto con las adecuadas herramientas, procesos y métodos. Brinda una especificación de las herramientas que se van a necesitar en cada momento, así como definir la instancia concreta del proceso que se va a seguir.

En concreto las responsabilidades de este flujo de trabajo incluyen:

- Selección y adquisición de herramientas
- Establecer y configurar las herramientas para que se ajusten a la organización.
- Configuración del proceso.
- Mejora del proceso.
- Servicios técnicos.

No por gusto es RUP la metodología de software más usada en el mundo, su aplicación llega incluso a resolver problemas empresariales inherentes a la informática. Es robusta y flexible. Muchas personas

asocian al Proceso Unificado de Software al excesivo uso de documentación, al trabajo innecesario y a la redundancia de información. No se debe ver de esta forma, si es cierto que cuanta con un gran número de artefactos, también es de es de razonar que no todos son aplicables en todos los casos, por lo que es de valorar en que momento se utiliza cada documento. Se puede señalar como ventaja el control con que cuenta RUP en cada paso del proyecto.

Varios desarrolladores tras darse cuenta de lo fallido de sus antiguos métodos han emigrado, por así decirlo, a RUP. Se debe mencionar el caso de Kevin Flood, que tras insatisfacciones en proyectos de elaboración de juegos virtuales, donde usaba el método de cascada, se dio cuenta de que el mismo le presentaba disímiles problemas desventajas como son:

- Se trabaja en equipos por separado según roles.(VALENCIA 2006)
- Tanto el cliente como los jefes de proyecto muchas veces no pueden ver el producto hasta que las versiones alfas no están terminadas.(VALENCIA 2006)
- Cada vez que se le realiza un cambio a los requisitos se le tiene que realizar todo el proceso desde el principio al juego. (VALENCIA 2006)

Por lo que su solución fue emplear un híbrido, mezclando las principales preeminencias que le podrían traer el uso de XP con RUP. Dentro de las ventajas que encontró en el Proceso Unificado de Rational se hallan la mejoría que provee cada artefacto a la unidad del equipo y de los diferentes módulos, ayudando estos a forzar la comunicación entre los diferentes grupos de roles, así como al gerente del proyecto y el usuario, por lo que traería consigo una mayor idea del estado del proyecto por parte de todos los implicados. Otras de sus superioridades es el estímulo constante a la reutilización del código, lo que no solo sería de beneficio para el proyecto desarrollado en el presente sino que agilizaría el vencimiento de un sistema futuro. La documentación constante de cada paso contribuye a menos iteraciones ya que esto ayuda a una disminución de los cambios.

Partiendo de esto también se mezclan ventajas de XP ya mencionadas anteriormente en este trabajo, como son las pruebas constante entre otras, esto da al equipo de trabajo una mejor noción del proyecto y una versión casi diario de lo que sería el producto.

Conclusiones parciales:

Cada una de las metodologías estudiadas por si solas son un potencial a tener en cuenta para guiar un proceso de software de Entornos Virtuales. Con las particularidades de XP no es sorprendente que sea la metodología más apropiada para un entorno caracterizado por requerimientos cambiantes originados por un mercado fluctuante y los propios avances de la tecnología y los negocios. De la misma forma RUP a opinión de los desarrolladores se encuentra entra las metodologías más completas y flexible con que se cuenta, con el empleo del UML para la representación de objetos es elemental que vaya más allá de los software de gestión. La Metodología de Larman aunque muy poco conocida, sin dudas se ha logrado ajustar también al desarrollo de Entornos Virtuales, su similitud, como otras metodologías tradicionales, como RUP hace que también se deba tenerse en cuenta en estudios más profundos.

A través del estudio de RUP y XP se conocieron aspectos importantes como sus principios, los cuales no solo serían bien aplicados en entornos virtuales, constituirían un método de producción de gran provecho para la Universidad de las Ciencias Informáticas. Tanto el Proceso Unificado del Racional como Programación Extrema incitan al trabajo en equipo, organizado, valoran al hombre por encima del entorno de trabajo.

Capítulo 2

Introducción:

La Universidad de las Ciencias Informáticas está compuesta por 10 facultades y aunque en todas se estudia la carrera de Ingeniería Informática, cada una de las facultades se especializa en un segundo perfil con el objetivo de lograr graduados, especialistas en diferentes temas. La facultad 5 se desarrolla dentro del campo de los Entornos Virtuales (EVs.), en este campo ha desarrollado diferentes proyectos como: simuladores de conducción, juegos de mesa, paseos virtuales y otros. Su trabajo es amplio, aunque lamentablemente su proceso no se encuentra bien definido. En el siguiente capitulo se realizará un estudio del modo de producción actual, mediante entrevistas y encuestas realizadas a los líderes de proyecto, seguidamente se valorarán las características de las metodologías XP y RUP que podrían resultar desfavorable o ventajosas para la producción de EVs en la UCI.

2.1 Encuesta:

La encuesta es un diseño o estrategia general no experimental, que permite contrastar las hipótesis de investigación con información sobre características de una población determinada. En efecto, constituye una de las formas más utilizadas para indagar sobre las características principales de un segmento poblacional y/o para conocer las opiniones, experiencias y expectativas de grupos de personas. Esta técnica permite recabar una serie de datos en un amplio número de casos.

Se realiza, cuando la información que se necesita puede ser obtenida a partir de la respuesta que una persona o varias puedan dar a un cuestionario preelaborado, y las mismas están dispuestas a colaborar con la investigación.

2.1.1 Ventajas y Desventajas de las Encuestas:

Entre las ventajas y desventajas de la utilización de encuestas como método de consulta se encuentran:

Ventajas

a) Proporciona un conocimiento de primera mano de la realidad y contribuye a realizar mediciones precisas.

b) Permite investigar grandes poblaciones, obteniendo datos –como por ejemplo, habitantes de una localidad, segmentos de consumidores, empleados de grandes organismos- que pueden ser estudiados

accediendo a pequeñas muestras representativas con excelentes niveles de exactitud, dentro de los márgenes propios del error de muestreo.

c) Para explicitar información recurre al lenguaje sobre temas abstractos o complejos que no se desprenden solamente de la observación de actitudes. En tal sentido, se pueden obtener por este medio, declaraciones sobre los fines, razones y propósitos –aún en el perfil cerrado que ofrece un estilo cuantitativo- el cual es de imposible acceso por observación de conductas.(GUIDO 2007)

Desventajas

a) No permite profundizar el conocimiento, en tanto, ofrece únicamente información extensiva, lo cual a fuerza de ser su principal virtud, constituye a la par su mayor desventaja para quienes persiguen la búsqueda de información intensiva (como es el caso de los grupos focales o las entrevistas en profundidad).

b) Tiene un costo importante en recursos humanos y económicos.

De acuerdo con la información que se quiere obtener los tipos de preguntas a obtener se pueden clasificar de la forma siguiente:

Cerradas: Se limita su respuesta a varias posibilidades previstas, donde la respuesta esta estructurada por comparaciones.

Abiertas: Son preguntas para ser respondidas libremente, no permiten obtener con exactitud la información deseada, sólo se logra conocer la opinión del encuestado.

Semi-cerradas: Limita la respuesta pero deja espacio libre para emitir opiniones sobre el tema.

Directas: Cuando el objetivo de la pregunta coincide con el objeto de interés del investigador.

Indirectas: Cuando de la respuesta se infiere la verdadera información que se quiere obtener. La formulación de este tipo de pregunta es un de las tareas mas difíciles que se enfrenta en la elaboración de un cuestionario.

De contenido: Por el contenido pueden ser objetivas cuando se refieren a hechos concretos o subjetivos cuando se buscan opiniones, actitudes del encuestado, etc.

De filtro: Permiten acceder a preguntas para las cuales se necesita cierta información.

De colchón: Para relajar tensiones que se producen por preguntas complejas o controvertidas.

De control: Se usan para valorar la consistencia de las respuestas dadas a determinadas preguntas. (GUIDO 2007)

En las preguntas formuladas en la encuesta realizada a la mayoría de los proyectos de producción de la facultad 5, principalmente a los que se desarrollan sobre el área de entornos virtuales, fueron previstas de formas semi-cerradas y directas. Los jefes de proyectos escogían entre diferentes opciones una o más respuestas, comentándolas según el interés des cuestionario.

2.1.2 Objetivos:

La facultad 5 no tiene un proceso de software definido, por lo que no cuenta con una metodología única para todos sus proyectos que guíen sus pasos. Con este estudio queremos lograr:

- Conocer cuales son las metodologías usadas en la facultad 5 para software de entornos virtuales.
- Conocer con que precisión se maneja cada metodología de forma independiente en los proyectos de entornos virtuales.
-
- Conocer que documentación se lleva a cabo en cada proyecto según tipo de metodología RUP o XP.
- Tener una idea de la efectividad y cumplimiento de los proyectos según el tipo de metodología.
- Establecer segmentos de población según edad con relación a los jefes de proyectos.

2.1.3 Diseño de muestra:

La facultad 5 cuenta con 15 proyectos que se desarrollan en diferentes áreas, como son: software de gestión, diseño 3D, SCADA (Control Supervisor y Adquisición de Datos) y software de entornos virtuales. Estos son:

Laboratorios Virtuales.

Paseo Virtual.

Diseño 3D.

Simulador Quirúrgico.

Juegos con Neurociencia.

Juegos de Consola.

Juegos de Mesa.

SCADA.

Auto Teórico-Práctico.

Sistema de Gestión.

Herramientas.

Tiro.

Tiro I+D.

Calidad.

Elementos Virtuales Inteligentes.

La muestra estuvo compuesta por 6 proyectos de la facultad 5, los mismos son desarrolladores de software de entornos virtuales tales como: juegos, simuladores y herramientas.

Los proyectos escogidos fueron:

Tiro.

Elementos Virtuales Inteligentes**.**

Paseo Virtual.

Laboratorios Virtuales.

Auto Teórico-Práctico.

Simulador Quirúrgico.

2.1.4 Recolección de Datos:

Para realizar esta encuesta se recurrió principalmente a los métodos: ***Encuestas auto administradas*** (esta metodología supone el llenado del cuestionario por parte de los propios ciudadanos sin asistencia de

encuestadores) debido a la muestra tan pequeña a la que se le realizaría el cuestionario y la sencillez de las preguntas, estas se realizaron de manera personal; y ***Encuestas vía e-mail*** aprovechando la tecnología con que cuenta la universidad lo cual sería de mucha comodidad tanto para el entrevistado como para el entrevistador. Según la regularidad de su implementación, por ser el llenado de forma voluntario por parte de los encuestados su clasificación es ***Encuestas ad-hoc***.

2.1.5 Análisis y Presentación de los datos:

En la encuesta, la cual se puede encontrar en el anexo 1, se realizaron preguntas para conocer que metodologías usaban generalmente los proyectos, así como la documentación generada, para tener una idea del manejo de las mismas. Excepto Paseo Virtual el resto coincide en el empleo de RUP, aunque no de una forma adecuada.

El proceso de software de estos proyectos como tal, es llevado a cabo mediante RUP, pero en su totalidad no se usa a profundidad la metodología. Según entrevistas realizadas a jefes de proyecto pos encuestas, estos fundamentan el uso de la misma por ser la única que conocen, a pesar de su poco domino de otras la encuentran como la más idónea a desarrollarse en la UCI por su versatilidad y flexibilidad. El uso de UML les permite una mejor comprensión del negocio y dominio del sistema. Independientemente de lo planteado, su empleo no es correcto debido a la falta de conocimiento práctico por parte de los desarrolladores. Sus etapas son atropelladas y no cumplidas a cabalidad. Falta disciplina de trabajo por parte del equipo, ya que no se despliega ni se almacena toda la documentación necesaria. Existe además un mal manejo de los artefactos definidos en cada flujo de trabajo. A pesar de esto continúan trabajando bajo la base teórica de RUP: guiado por casos de uso, centrado en la arquitectura, e iterativo e incremental.

Paseo Virtual es el único proyecto de los encuestados que guía su proceso bajo XP. De igual forma que al resto de los proyectos, se le realizó una entrevista a la dirección del proyecto para indagar las causas del uso de la metodología. Como principal respuesta a la interrogante se encuentra que este grupo de trabajo desarrolla proyectos cortos con fechas de entregas cortas, RUP sería una perdida de tiempo producto a su abundante documentación y métodos. Como otra de las características favorables se encontró que XP es fácilmente adaptable a un equipo pequeño de desarrollo, no se excede en la utilización de roles. Pero tampoco XP es utilizada de forma correcta, esto es debido a que no cuenta con las condiciones máximas para su desarrollo, el equipo de trabajo por ser de diferentes años es disperso y

cuesta trabajo reunirlo con frecuencia como plantea la metodología, por lo que las tareas se reparten de forma individual por el jefe de proyecto. A pesar de ser una metodología ágil en ocasiones el proyecto emplea el UML para la comprensión de determinada funcionalidad, siempre y cuando lo requiera; por lo que se valora como un aspecto positivo del proyecto el cual ha sabido adaptar XP a su régimen de trabajo.

A través de la encuesta y las entrevistas realizadas a los proyectos se puede concluir que aunque no se trabaja de forma artesanal, la facultad 5 no cuenta con un proceso de desarrollo de software definido ni implementado, debido a que no se realizan actividades encaminadas a establecer un orden entre los pasos que se deberían cumplir para entregar el producto en tiempo y forma, no trabajan de forma única y organizada. Falta profundidad en el conocimiento de las metodologías usadas por los proyectos, así como una adaptación de las mismas al proceso de Entornos Virtuales.

2.2 Análisis XP y RUP con respecto a la facultad 5 y la UCI..

Teniendo en cuenta la situación actual de la UCI, en especial de la facultad 5, referente a su proceso de producción de software, el cual no está definido, valorando las características propias de cada una de las metodologías estudiadas anteriormente, XP, Larman y RUP y sabiendo además que ellas forman parte de las más usadas en el desarrollo de software de entornos virtuales, se realiza un estudio valorativo de sus principales características, enfocando el análisis a las particularidades de la Universidad de las Ciencias Informáticas. Se obviará Larman, pues se considera que RUP dentro de las metodologías tradicionales es más completa y actual.

En muchos aspectos de cada metodología de las estudiadas cabe señalar algunos puntos en los cuales se tendría que valorar el llevarlos a cabo o no en el proceso de producción a proponer para Entornos Virtuales. Se podría comenzar analizando el tema de la Programación Extrema (XP).

Como problema fundamental en esta metodología se encuentra que está diseñada prácticamente para un equipo de trabajadores expertos y, en cierta forma, bien fusionados. En la UCI, la fuerza productiva está principalmente en estudiantes de distintos años y de profesores jóvenes, en la mayoría de los casos recién graduados, por lo que el término de expertos desaparece prácticamente. Esto trae consigo que se pierdan algunas de las ventajas que plantea la metodología, debido a que sería muy difícil el no respetar la propiedad del código, ya que solo un programador de experiencia podría modificar el trabajo de su homólogo sin dañar el programa, por lo que sería de mucha dificultad aplicar el principio #6 de

XP:"Propiedad colectiva del código", por lo que se recomienda mayormente un correcto control de versiones y que cada cual se responsabilice con su trabajo. La magnitud del proyecto o la cantidad de trabajadores también serían una dificultad relacionada con la cohesión del equipo; en la UCI se pretende una gran participación de estudiantes por proyecto, lo que dificultaría la comunicación entre los miembros del mismo, como ejemplo de la vida real, es mucho más fácil llegar a un acuerdo entre un grupo reducido de personas que entre una multitud, por lo que tal vez una forma de solución sería trabajar en pequeños equipos por roles.

Es cierto que en los juegos virtuales generalmente los requisitos cambian con mucha facilidad, el cliente según avanza el desarrollo se va dando cuenta de las funcionalidades necesarias, esto sin dudas no es problema para XP debido a la importancia del rol del cliente dentro del proyecto y a que las iteraciones son relativamente cortas. Cabe señalar que estos cambios en los requisitos, de no ser manejados cuidadosamente se pueden convertir en un caos. Si el cliente no es lo suficientemente capaz de calcular el impacto del cambio, como plantea XP, esto implicaría altos costos además de convertirse el trabajo en tedioso e interminable. Por lo que también se necesita de un cliente experto, debido a que es el que toma la mayor parte de las decisiones en el proyecto, lo que sería algo difícil para la UCI si el inversor es extranjero o ajeno a la universidad, lo que sucede generalmente.

El principio # 10 de XP: "Semanas de 40 horas" es prácticamente inaplicable al modo de producción de la UCI, no se puede tener un estudiante implicado en una jornada de 8 horas diarias, ni siquiera a un profesor sin ser liberado de sus obligaciones. Quizás esto podría cumplirse más bien para los medios de producción, se lograría un mejor aprovechamiento de las PC si al menos se trabajaran 40 horas a la semana, incluso más tiempo ubicando de 3 a 4 estudiantes por PC.

Como se mencionó anteriormente el cliente consta con gran responsabilidad dentro del proyecto, según XP. Pero no sería nada ventajoso para un polo productivo que fuera este el que estimara los costos del negocio a implementar como plantea el proceso de desarrollo de la metodología. En este caso sería de mayor seguridad un especialista de la universidad trabajando en conjunto con el inversor.

Otro problema a señalar es la especialización de los roles por preferencia, otra vez la UCI se encuentra en desventaja debido a su fuerza de producción inexperta, al estar los trabajadores de proyectos de la universidad en pleno desarrollo como profesionales, es casi imposible lograr que un estudiante realice la función que desea dentro del grupo. Tema que tendría como posible propuesta de solución la

especialización por año cursado, un ejemplo sería: dentro del programa de estudio de un estudiante de segundo año está la asignatura Base de Datos, por lo que este estudiante podría ejercer este rol perfectamente, en tercer año se incluye ingeniería de software, por lo que el rol de diseñador o analista no podría desarrollarlo un estudiante de menor año.

Lo que tal vez para los desarrolladores de XP es una ventaja (la escasez de documentos que se usan), no es verdaderamente provechoso para un sistema que transite por etapas de análisis y diseño; las historias de usuarios podrían funcionar perfectamente como la descripción de un caso de uso, pero se estima que no se debe pasar por alto la diagramación en UML que tanto obvia XP. Debe tomarse con más seriedad el caso de la arquitectura, ya que sería una ventaja la integración constante que tanto se defiende si se cuenta con un arquitecto encargado de integrar las principales funcionalidades del sistema.

Para llegar a la conclusión final no solo debe detenerse el estudio en las desventajas, XP cuenta con muchos aspectos que deben tomarse como referencia para guiar un proceso de software para sistemas de realidad virtual y juegos, no por gusto se incluye en este estudio, además de ser una de las metodologías más utilizadas a nivel internacional, e incluso en la facultad 5 de la UCI.

Se podría citar como una de sus principales ventajas la comunicación constante entre el cliente y los restantes miembros del proyecto, esto para la UCI mejoraría las relaciones interpersonales y consigo se lograría una mejor cohesión del grupo de trabajo en conjunto con los inversionistas. Como habíamos mencionado anteriormente los requisitos de un software de realidad virtual son muy cambiantes, teniendo al cliente como un miembro del equipo este tomaría los cambios con mayor cautela. La retroalimentación nutre la verdadera comprensión del negocio.

Los roles en la UCI varían constantemente, debido al cambio de años y de materias de los estudiantes, por lo que si alguien es nuevo como programador le sería muy difícil comprender el trabajo de su antecesor si este no aplica la política de sencillez del código planteada en XP, por lo que este método también podría ser paradigma de los desarrolladores de realidad virtual.

Para cualquier grupo de desarrolladores es elemental la agilidad y el culminar rápidamente y de forma organizada un sistema; si se toma como referencia lo planteado por XP en cuanto a la planificación de iteraciones cortas e integraciones constantes del software podría ser adquirido como una ventaja práctica a aplicar en la UCI. Esto traería consigo no solo una mejoría para el proyecto, influiría también en la satisfacción del cliente porque en cada release obtendría una versión del sistema. Esto trae consigo

además la implementación constante de pruebas, lo que demostraría la corrección del código y daría a los desarrolladores una idea de cuan adelantado se encuentra el trabajo.

Una de las mayores ventajas de la Programación Extrema que conseguiría la universidad, y por consiguiente la facultad 5, sería el aprovechamiento de la programación en pareja, principio # 3 de XP. De esta manera se aprovecharían aun más los medios de producción, dígase las computadoras y laboratorios, además de plantearse como principal ventaja que dos cabezas piensan mejor que una; entre dos personas se lograría una mejor perfección del código, se encontrarían con mayor agilidad los errores mostrados por el compilador, se llegaría a una solución más fluida y rápida, no se perdería la idea del trabajo ni se dejarían tareas sin hacer, el código siempre estará siendo revisado por una persona a la misma vez que lo escribe otro. Este principio podría ser extendido no solo a la implementación, también sería válido para el análisis y diseño, siendo la diagramación en UML un tema a veces complicado para una sola persona. Esto ayudaría además al flujo de la información en el equipo.

Una de los procesos más complicados dentro de la producción de software es la integración de los módulos por los efectos colaterales que esto pueda tener, como el dejar de funcionar partes del software por motivos desconocidos. Precisamente debido a la entrega constantes de release, XP proporciona una integración menos complicada y de manera rápida, realizándose esta el final de cada pequeña iteración.

XP como una de sus principales políticas, valora la importancia de las personas y el equipo por encima del entorno de trabajo, siendo este último las herramientas: software y hardware. Este planteamiento podría ser de vital importancia para la UCI si se implementa como un valor en sus estudiantes. No sería de mucho provecho, contar con una tecnología de punta, con el mejor software y hardware, si no se tiene un trabajador comprometido con el proyecto, no a un estudiante de 5 puntos, sino a uno preparado y dispuesto a desempeñar su rol. De igual forma tampoco se obtendrían buenos resultados, si los desarrolladores no trabajaran de forma comunicativa y en unidad, no se trata de lograr un equipo “todos estrellas”, sino uno bien compacto y encaminado a lograr de forma unánime un sistema de calidad.

Ahora, si bien la Programación Extrema se limita en el uso de la documentación, el Proceso Unificado de Software en ocasiones se excede en el uso de artefactos que solo retrasarían el trabajo, además de volverse tedioso y muchas veces provocar el descontento en el equipo de desarrolladores, aunque se ha demostrado que se puede desarrollar un buen sistema, incluso complejo, omitiendo gran parte de la

documentación propuesta por RUP. Es recomendable tener especial cuidado con esto, porque podría atropellarse la metodología y no lograr los objetivos deseados durante la elaboración del proyecto.

Contradictorio en ambas metodologías- XP y RUP- es el uso de la arquitectura, si bien XP no abusa de la misma, RUP le da al rol de arquitecto una vital importancia dentro del proyecto; en esta metodología el encargado de la arquitectura es unos de los principales trabajadores en la toma de decisiones, es el encargado de la integración y de la lógica de las principales funcionalidades del sistema. A los sistemas de entornos virtuales les sería de gran utilidad dentro del grupo de trabajo el arquitecto, pues muchas veces los simuladores llevan un soporte en software y hardware especial, este sería el encargado de la distribución de las partes del sistema.

De la misma forma que se encuentran diferencias irrefutables entre ambas metodologías, dígase estas XP y RUP, las dos coinciden en una serie de aspectos importantes a tomar en cuenta. RUP es dirigido por casos de usos que son las funcionalidades del sistema las cuales le proporcionan un valor añadido al usuario, y dan respuestas a un grupo de requisitos. XP garantiza describir las funcionalidades del software mediante las tarjetas de usuarios, estas describen de forma clara pero breve un grupo pequeño de requisitos que puedan ser implementados por el programador en no más de una semana de trabajo, en este caso la plantilla de la tarjeta de usuario sería el homólogo del artefacto en RUP: Descripción del Caso de Uso.

Al igual que RUP, XP es iterativo e incremental. Cabe señalar además que ambas coinciden en la entrega constante de mini proyectos al cliente, solo que las iteraciones en XP serían mucho más pequeñas ya que aunque de igual forma transita el proyecto por todos los flujos de trabajo, RUP cuenta con más flujos de trabajo (Requisitos, Análisis, Diseño, Implementación y Pruebas, como fundamentales) por lo que el recorrido se hace más complejo.

Los 6 principios en que está basado el proceso de RUP, podrían ayudar de mucho a la producción en los proyectos de realidad virtual en la facultad 5. Estos plantean la flexibilidad de los procesos de la metodología para cualquier tipo de sistemas. Transitar el proyecto por un grupo de procesos antes de pasar directamente a la codificación del sistema. Valorar las herramientas a utilizar, así como los patrones de diseño, son métodos que harían más factible la solución y elaboración del sistema.

RUP cuenta con un proceso más definido y la misma vez más complejo que XP. Consta de nueve flujos de trabajo y cuatro fases bien definidas por donde transita el futuro software.

Conclusiones parciales.

A partir de la encuesta realizada en este capítulo se pudieron apreciar algunas características del modo de producción llevado a cabo hasta el momento por los proyectos de la facultad 5, viendo sus principales deficiencias expuestas principalmente por los líderes de proyecto. Se realizó además un análisis de las ventajas que podrían traer al modo de producción de la facultad y los principios estudiados en el capitulo 1 de las metodologías RUP y XP

Capítulo 3

Introducción:

Generalmente las acciones para comenzar un proyecto parten de la necesidad del cliente. Este solicita a la entidad productora de software la automatización de un conjunto de requisitos obteniéndose al final un sistema capaz de cumplir con los requerimientos del proyecto. Visto de forma tal parece tarea fácil, de pensar así seguramente el fracaso es evidente.

En la actualidad los sistemas informáticos son cada vez más complejos, por lo que si en el pasado su producción era de forma artesanal y sencilla, hoy trae consigo algo más que implementación y código. Para realizar un proyecto es necesario transitar por una serie de actividades de forma organizada antes de sentarse a programar. Se necesita tener bien definidas las tareas que se llevarán a cabo con el cliente, se debe realizar una eficiente elección del grupo de desarrolladores y la planificación de cada uno de los procesos ha de ser primordial para una adecuada prevención de los tiempos de entrega y cumplimiento de las tareas.

En la Facultad 5 de la Universidad de las Ciencias Informáticas, para la producción de entornos virtuales, donde se encuentran las líneas de video juegos o juegos virtuales y simuladores no se cuenta hasta el momento con un proceso que guíe o mejore las tareas a seguir para lograr un producto de calidad y en tiempo para entregar al cliente, es por eso que se propone en este capítulo un proceso constituido por 30 actividades con el objetivo de lograr un trabajo organizado. De estas actividades, 17 son de carácter fundamental, y consisten principalmente en tareas de planificación, gestión, y elaboración del proyecto; 8 son de retroalimentación interna con el grupo de trabajo y especialistas o asesores directamente relacionados con el proyecto, y 5 actividades de retroalimentación con el cliente, el cual dará su valoración y aceptación de los artefactos generados por las actividades fundamentales. La propuesta nace inicialmente a partir de talleres y cursos realizados en la facultad 5 donde los líderes de proyectos proponían actividades que lograran eficiencia en los proyectos de realidad virtual. El proceso cuenta inicialmente con una etapa de concepción y planificación del proyecto, donde se realizan tareas, conjuntamente con el cliente, consistentes en pacto, modelado del negocio, definición de gestión de configuración y requisitos de alto nivel en pos de preparar como tal la etapa de elaboración, la cual comenzará seguidamente después de la firma del contrato legal entre los desarrolladores y cliente. En la etapa de elaboración se procedería e llevar a cabo todo lo planificado en el período de concepción,

capturando requisitos funcionales y no funcionales de manera más específica, realizando el análisis y el diseño para finalmente implementar, integrar y probar el software que se entregará al cliente.

En las actividades propuestas se establecerán los responsables, participantes, la misión, la entrada y la posible salida en forma de artefactos. El proceso además está descrito en forma de diagrama de actividades en la siguiente figura, donde aparecen solo algunos de los artefactos de entrada y salidad.

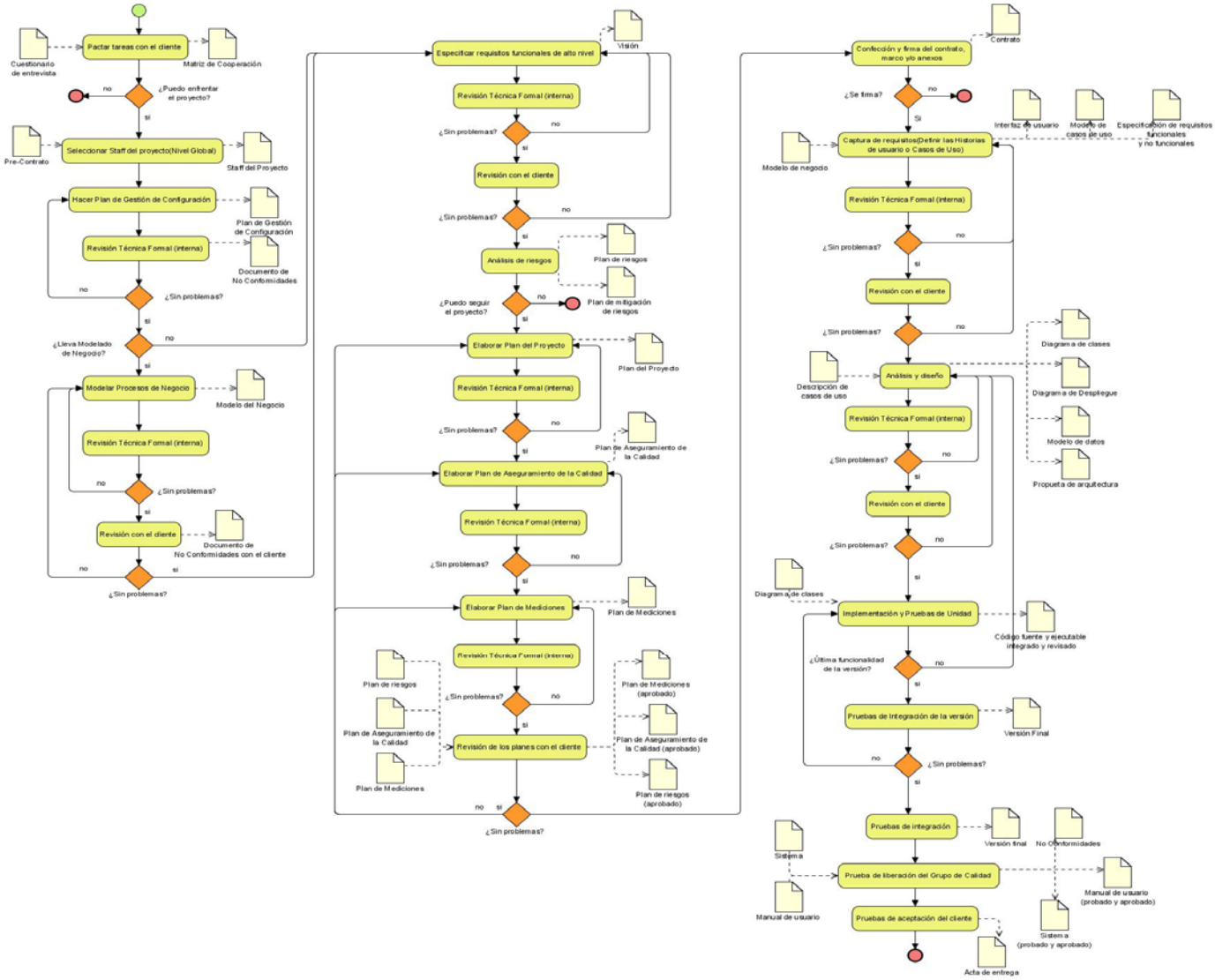

Figura 1: *Diagrama de actividades del Proceso de Producción de Software de Realidad Virtual.*

3.1 Descripción del proceso:

Actividad 1: Pactar tareas con el cliente.

Responsable: Jefe de Proyecto.

Participantes: Jefe de Proyecto; Vicedecano de producción; Jefe polo productivo; Asesor de calidad, Cliente.

Misión: Acordar las tareas que se realizarán hasta la firma del contrato, fijar los momentos de encuentro entre las partes, comprender lo que desea el cliente y explicarle lo que la organización es capaz de hacer.

Entradas:

- Cuestionario de la entrevista.

Salidas:

- Acta de la reunión.

- Matriz de cooperación.

Esta tarea da inicio a todo proyecto, es la reunión inicial donde el cliente muestra sus necesidades y la facultad expone sus riesgos y condiciones para enfrentar el trabajo. Al finalizar esta tarea se tomará la decisión de enfrentar el proyecto o retirarse a partir de las solicitudes del cliente y las viabilidades de cumplir con las mismas. De aceptar las condiciones se procedería a la actividad 2, en caso contrario se abandonarían las negociaciones. En la matriz de cooperación quedará plasmado el cronograma de las tareas que realizarán los desarrolladores con el cliente.

Actividad 2: Seleccionar Staff del proyecto (Nivel Global).

Responsable: Jefe de Proyecto.

Participantes: Jefe de proyecto; Vicedecano de producción; Jefe polo productivo.

Misión: Definir el staff necesario para la concepción y definición del proyecto.

Entradas:

- Precontrato.

- Plantilla de Roles de los proyectos de Realidad Virtual.

Salidas:

- Staff Global del proyecto (Esta salida va para el proceso de selección del personal).

En la selección del Staff del proyecto solo se tendrá en cuenta la elección de los principales roles que integrarán el equipo, por ejemplo: Arquitecto principal, Analista principal, Diseñador principal, Administrador de Gestión de Configuración y Jefe del grupo de calidad, estos serían los encargados de comprender el negocio así como las necesidades del cliente y posteriormente los mismos conformarían el grupo de desarrollo que llevaría a cabo el sistema.

No es recomendable seleccionar otros roles en este momento, ni la cantidad de trabajadores, pues no se conoce la magnitud del proyecto, ni los requerimientos; por ejemplo: no tendría por que solicitar un diseñador se base de datos si no se tiene conocimiento si el sistema a implementar lo necesita.

Actividad 3: Confeccionar Plan de Gestión de configuración.

Responsable: Jefe de Proyecto

Participantes: Jefe de Proyecto, Arquitecto principal, Administrador de Gestión de Configuración.

Misión: Confeccionar el Plan de Gestión de Configuración del proyecto para garantizar la integridad del producto a lo largo del desarrollo. Establecer las medidas de gestión y control de cambios. Definir quienes van a ser los participantes del proyecto, a medida que este vaya avanzando, elegir la plataforma y herramientas a usar, especificar cuales son los Elementos de Configuración de Software, especificar las líneas bases, realizar plan de control de cambios y versiones.

Entradas:

- Staff del proyecto (listado).
- Propuesta de herramientas.
- Plan del proyecto.
- Proceso de Producción (identificación de las líneas base).
- Proceso de gestión y control de cambios,

Salidas:

- Plan de Gestión de Configuración (Listado de roles y equipos, herramientas a usar, elementos de configuración de software, líneas base, plantilla de control de cambios, plantilla de control de versiones).

Es importante tener en cuenta que la gestión de configuración se realiza durante todas las actividades asociadas al desarrollo del sistema, por lo que solo se encuentra en este orden de manera inicial y la misma se irá surtiendo del proceso según su desarrollo.

Actividad 4: Revisión técnica formal (interna).

Responsable: Jefe de Calidad.

Participantes: Jefe de Proyecto, Arquitecto principal, Administrador de Gestión de Configuración, Jefe de Calidad.

Misión: Revisión del Plan de Gestión de Configuración del proyecto con el Jefe de Calidad con el objetivo de hallar alguna deficiencia y ser rectificada por los desarrolladores

Estradas:

- Plan de Gestión de Configuración.

Salidas:

- Acta de la reunión.

- Documento de no conformidades.

Seguidamente después de cada actividad fundamental dentro del flujo se realizaría una revisión técnica internamente del proyecto, esto ayudaría a conocer si la actividad se realizó de forma correcta así como si fueron cumplidos los objetivos, constituiría una parte importante dentro de la retroalimentación interna del grupo de trabajo, sería un indicador del trabajo realizado y contribuiría con el tiempo a erradicar los errores antes cometido. Sería esto poner en práctica el valor de XP de la Retroalimentación, antes analizado. En caso de que esta actividad encontrara algún problema, se procedería a realizar nuevamente la actividad anterior, en este caso sería la 3.

Actividad 5: Modelado del negocio.

Responsable: Analista principal.

Participantes: Analista principal; Jefe de proyecto; Asesor; Arquitecto; Diseñador de base de datos; Planificador del proyecto, Cliente.

Misión: Identificar, describir y mejorar los procesos de la organización.

Entradas:

- Acta de la reunión.

- Metodología a utilizar para el modelado del negocio.

- Guía de las entrevistas a realizar.

- Matriz de cooperación.

Salidas:

- Modelado del negocio.

- Glosario de términos.

Esta actividad forma una parte fundamental dentro del proceso propuesto para cualquiera de los dos tipos de software que implementa Entornos Virtuales en la facultad 5 (simuladores y juegos), es necesario comprender el funcionamiento de la entidad para el correcto desarrollo del sistema. En este caso se propone la utilización del UML por su fácil comprensión incluso por personas ajenas a la informática, por lo que tendría una mejor participación el cliente. Esta tarea no siempre es de carácter obligatorio, pueden existir sistemas que no necesiten de esta actividad, en este caso se pasaría directamente a la actividad # 8 de captura de requisitos de alto nivel.

Actividad 6: Revisión técnica formal (Interna).

Responsable: Jefe de Calidad.

Participantes: Jefe de calidad; Analista principal; Jefe de proyecto; Asesor; Arquitecto; Diseñador de base de datos; Planificador del proyecto.

Misión: Revisión y aprobación de los documentos y artefactos del modelado del negocio

Entradas:

- Modelado del negocio.

- Lista de chequeo para la revisión.

Salidas:

- Acta de la revisión.

- Plantilla de no conformidades (Registros).

Como se explica en la misión esta revisión técnica formal es para revisar la calidad del modelado del negocio mediante una lista de chequeo.

Actividad 7: Revisión con el cliente.

Responsable: Jefe de proyecto.

Participantes: Jefe de proyecto; Planificador de la calidad; Analista principal, Cliente.

Misión: Revisión con el cliente del modelado del negocio así como su aprobación.

Entradas:

- Modelado del negocio.

Salidas:

- Acta de aceptación del cliente.

- Documento de no conformidades (Registros).

Todas las actividades del proceso, después de tener las revisiones internas formales con el grupo de calidad, deben tener una revisión con el cliente donde se le explicará la tarea ya repasada por el equipo de trabajo, esta reunión generará una acta donde quedará plasmado todo lo entregado al cliente el cual emitirá también un documento de no conformidades que deberá ser procesado por los desarrolladores. Esta actividad fue concebida teniendo en cuenta principios de XP, donde el cliente constituye una parte fundamental del equipo de trabajo y de los resultados del producto. En caso de detectar problemas en esta actividad que no puedan ser resueltos en el marco de la reunión se retornaría a la actividad fundamental que la antecede.

Actividad 8: Especificar requisitos funcionales de alto nivel.

Responsable: Jefe de proyecto.

Participantes: Jefe de proyecto; Analista principal; Asesor, Arquitecto, Diseñador de base de datos; Planificar del proyecto.

Misión: Definición de los requisitos de alto nivel, así como los objetivos del sistema.

Entradas:

- Modelado del negocio.

- Glosario de términos.

Salidas:

- Documento visión (Contempla además el análisis económico, estado del arte, etc.)

- Staff del proyecto (Esta salida va para el proceso de selección del personal).

- Glosario de términos.

A partir de esta actividad se realiza un análisis económico inicial del proyecto, se analizaría el estado del arte, ya se conformarían los grupos de trabajo, se definiría la cantidad de trabajadores del sistema así como las herramientas de trabajo, inclúyase hardware. Es la captura de requisitos de alto nivel, la actividad que define verdaderamente los objetivos del sistema, se va teniendo una imagen abstracta de cómo quedará conformado el software, es importante llevar a cabo esta tarea con profundidad debido a su influencia en el análisis de riesgos del proyecto.

Actividad 9: Revisión técnica formal (interna)

Responsable: Jefe de Calidad

Participantes: Jefe de proyecto; Analista principal; Asesor, Arquitecto, Diseñador de base de datos; Planificar del proyecto, Jefe de Calidad

Misión: Revisión interna del proyecto con el asegurador de la calidad de la actividad 8

Entrada:

- Documento visión (Contempla además el análisis económico, estado del arte, etc.)

- Staff del proyecto (Esta salida va para el proceso de selección del personal).

- Glosario de términos.

Salidas:

- Glosario de términos.

- Acta de la revisión.

- Documento de no conformidades (Registros).

Posteriormente se realizaría el análisis con el cliente para asegurar que esté de acuerdo con los objetivos planteados en la actividad 8.

Actividad 10: Revisión con el cliente.

Responsable: Jefe de Proyecto.

Participantes: Jefe de proyecto; Analista principal; Asesor, Arquitecto, Diseñador de base de datos; Planificador del proyecto, Jefe de Calidad, Cliente.

Misión: Revisión y aprobación de los requisitos de alto nivel por el cliente

Entradas:

- Documento visión (Contempla además el análisis económico, estado del arte, etc.)

- Staff del proyecto (Esta salida va para el proceso de selección del personal).

- Glosario de términos.

Salidas:

- Glosario de términos.

- Acta de la revisión.

- Plantilla de no conformidades (Registros).

Actividad 11: Análisis de riesgos.

Responsable: Jefe de calidad.

Participantes: Jefe de calidad, Jefe de proyecto, Analista principal, Asesor, Arquitecto, Diseñador de base de datos, Planificador del proyecto, Cliente.

Misión: Análisis y mitigación de los riesgos que pueda traer el desarrollo del proyecto.

Entradas:

- Documento visión

- Glosario de términos.

Salidas:

- Plan de riesgos.

- Plan de mitigación de riesgos.

- Glosario de términos.

La función del análisis de riesgos del software es identificar, estudiar y eliminar las fuentes de riesgo antes de que empiecen a amenazar la finalización satisfactoria de un proyecto software. Dependiendo del momento en que se encuentra el riesgo existen cinco niveles: Control de crisis, que es controlar los riesgos cuando ya se han convertido en un problema; arreglar cada error, que sería reaccionar rápidamente; mitigación de riesgos, que se trata de planificar con antelación el tiempo que se necesitaría para cubrir un riesgo ocurrido; prevención de riesgos, que se trata de planificar la detención de los riesgos antes que estos se conviertan en problemas y eliminación de las causas principales que se encarga de identificar y eliminar las posibles causas de un riesgo. Aunque esta tarea está enmarcada fundamentalmente en esta etapa por la importancia de prevenir los riesgos se considera que sea constante durante todo el proyecto fundamentalmente cada vez que se realice una actividad de planificación o fundamental.

Actividad 12: Confeccionar plan del proyecto.

Responsable: Jefe de proyecto.

Participantes: Jefe de proyecto, Analista principal, Asesor, Arquitecto, Diseñador de base de datos, Planificador del proyecto.

Misión:

Entradas:

- Documento visión.

- Metodología de desarrollo a utilizar.

- Glosario de términos.

Salidas:

- Plan de desarrollo.

- Glosario de términos.

Dentro de las actividades de concepción del proyecto la planificación se encuentra entre las más importantes. Todas las tareas que se desarrollaran en lo adelante deben tener una fecha de cumplimiento, responsable y participantes bien reflejados en el plan. A partir de una correcta planificación tanto el cliente como los desarrolladores tendrían un estimado de la terminación de cada iteración o versión. En el plan del proyecto irían reflejado además los antecedentes y objetivos del sistema entre otros aspectos.

Actividad 13: Revisión técnica formal (Interna).

Responsable: Planificador de la Calidad.

Participantes: Planificador de la calidad; Analista principal; Jefe de proyecto; Asesor; Arquitecto; Diseñador de base de datos; Planificador del proyecto.

Misión: Revisión y aprobación del plan del proyecto

Entradas:

- Plan de desarrollo.

- Lista de chequeo para la revisión.

Salidas:

- Acta de la revisión.

- Plantilla de no conformidades (Registros).

Actividad 14: Confeccionar plan de aseguramiento de la calidad**.**

Responsable: Planificador de la Calidad.

Participantes: Planificador de la calidad; Jefe de proyecto, Planificador del Proyecto.

Misión: Asegurar la calidad del proyecto y que las tareas de esta actividad se cumplan en tiempo y correctamente, para entregar un producto que satisfaga al cliente.

Entradas:

- Plan de desarrollo.

- Glosario de términos.

Salidas:

- Plan de calidad.

- Glosario de términos.

Actividad 15: Revisión técnica formal (Interna).

Responsable: Asesor de calidad.

Participantes: Asesor de calidad; Planificador de la calidad; Jefe de proyecto;

Misión: Revisión interna con los integrantes de la dirección del proyecto y el Asesor de calidad del Plan de Aseguramiento de la Calidad con el fin corregir el plan en caso que este presente problemas.

Entradas:

- Plan de calidad.

- Lista de chequeo para la revisión.

Salidas:

- Acta de la revisión.

- Plantilla de no conformidades (Registros)

El Asesor de la Calidad es una persona del equipo de desarrolladores pero que además de ser un profesional informático, conoce el funcionamiento del negocio, quizás pueda ser alguien con experiencias en el tipo de proyecto que se lleva a cabo.

Actividad 16: Confeccionar plan de mediciones.

Responsable: Jefe de Calidad

Participantes: Jefe de Proyecto, Jefe de Calidad.

Misión: Describir las mediciones que se deben reportar y los mecanismos que se utilizan. Generar la información necesaria a los líderes del proyecto.

Entradas:

- Documento Visión

- Plan de desarrollo.

- Plan de aseguramiento de la calidad.

- Glosario de términos

Salidas:

- Plan de Mediciones.

- Glosario de Términos.

Concepto de Medición: Es una idea de las entidades que deberían ser medidas para satisfacer una necesidad de información.

Actividad 17: Revisión técnica formal (interna).

Responsable: Asesor de Calidad.

Participantes: Asesor de Calidad, Jefe de Calidad, Jefe de Proyecto

Misión: Revisión interna con el Asesor de Calidad del plan de mediciones del proyecto.

Entradas:

- Plan de Mediciones.

- Glosario de Términos.

Salidas:

- Acta de la reunión.

- Plantilla de no conformidades (Registros).

Actividad 18: Revisión de los planes con el cliente.

Responsable: Jefe de proyecto.

Participantes: Jefe de proyecto, Vicedecano de producción, Asesor de calidad, Cliente.

Misión: Revisión con el cliente de todos los planes mencionados en las actividades anteriores con el fin aceptar su aprobación.

Entradas:

- Plan de desarrollo.

- Plan de de Aseguramiento de la Calidad.

Salidas:

- Documento de no conformidades (Registros)

- Acta de aceptación del cliente.

Debidamente después de esta tarea tanto el cliente como la institución se encuentran en condiciones de firmar el contrato. Por parte del cliente ya este conoce las posibilidades que puede brindarle la facultad 5: tiempo aproximado en que se realizaran las entregas, requisitos que pueden darse cumplimiento y costo estimado del proyecto. Por parte de la institución: se conoce el negocio, los requerimientos del cliente, los objetivos del software, la fuerza y medios de trabajo que se necesita para el desarrollo. Por lo que a continuación se pasa a la elaboración y firma del documento legal que compromete a la facultad con el cliente.

Actividad 19: Confección y firma del contrato marco y/o anexos.

Responsable: Jefe del proyecto.

Participantes: Jefe de proyecto, Vicedecano de producción.; Asesor de calidad, Cliente, Asesor Jurídico.

Misión: Firma del contrato de elaboración del proyecto con el cliente con el fin de un compromiso legal entre las dos entidades.

Entradas:

- Plan de desarrollo.

- Plan de calidad.

Salidas:

- Contrato marco y/o anexos.

Esta tarea da inicio al compromiso legal de la facultad 5 de cumplir con la elaboración del proyecto satisfaciendo de esta forma los requerimientos del cliente en tiempo y con calidad, traduciendo esto en un sistema eficiente y eficaz. Aunque también puede existir el caso contrario y no ser firmado el contrato, lo que conllevaría al abandono del proyecto por los interesados. En caso de ser aprobado el tratado no se cerrarían las negociaciones de inmediato, este en el transcurso del proyecto se irá enriqueciendo con nuevos anexos según corresponda y se procedería al siguiente número de actividades del proyecto.

Actividad 20: Captura de requisitos (confección de las historias de usuario o casos de uso, según metodología).

Responsable: Jefe de Proyecto

Participantes: Analista principal, Jefe de proyecto, Especificador de Casos, Diseñador Gráfico, Arquitecto.

Misión: Descripción y capturar los requerimientos funcionales y no funcionales del software o la versión Confección de los casos de uso, o tarjetas de usuarios

Entradas:

- Plan de captura de requisitos.

- Modelado del negocio.

Salidas:

- Especificación de requisitos funcionales y no funcionales.

- Descripción de casos de uso.

- Modelo de casos de uso.

- Interfaz de Usuario.

- Historias de Usuario en caso de usar XP.

Para esta actividad se deben realizar una serie de entrevistas, de modo personal; donde estarían presentes el Analista principal del proyecto, el Jefe de dicho proyecto y una representación de la entidad cliente. En esta reunión debe aclararse cuales son las características principales y las funcionalidades básicas que debe de poseer el software a realizar, de manera general. Luego se interactúa directamente con el negocio durante el tiempo planificado, en dicho proceso se debe convivir con todos aquellos trabajadores del negocio que de algún modo intervienen en el proceso, para lograr la total comprensión del mismo y el refinamiento de la lista de requisitos obtenida en la entrevista con la representación de la entidad, de modo que se capturen aquellas cualidades o facilidades que debe de aportar el producto a elaborar, el tiempo de esta actividad estará en dependencia de la complejidad de dicho negocio. Esta tarea al igual que el resto del proceso debe realizarse de forma iterativa, no se debe ejecutar una sola vez en el transcurso del proyecto ya que se nos pueden quedar funcionalidades por definir en las primeras iteraciones. Aunque parece sencillo la captura de requisitos no es una labor fácil, además de existir variedad de patrones no se cuenta con un método capaz de ser eficiente al 100%, y siendo más así en los juegos donde los requisitos son cambiantes y muchas veces el cliente solo viene con una idea vaga de lo que desea, además de que los juegos cuentan con requerimientos tan singulares como por ejemplo la claridad de la expresión de una persona o la representación exacta de un mapa o terreno. El trabajador del sistema que realizará esta tarea se recomienda que esté preparada en diseño gráfico, este en ocasiones podría ser capaz de guiar al cliente aconsejándole como debería quedar al juego o el simulador teniendo en cuenta que en muchas ocasiones este no es informático.

Actualmente la Facultad 5 no cuenta con una técnica específica para realizar la Captura de Requisitos para los juegos y los simulador, es esto una actividad que se va mejorando con la práctica debido a que con el tiempo se van conociendo y superando la variedad de riesgos que se encuentra en esta tarea. A pesar de esto, RUP propone pasos, los cuales no convienen llevarse a cabo por separados, que se deben tener en cuenta para cualquier punto de partida y aparecen explicados en El Proceso Unificado de Software, estos son:

- Enumerar los requisitos candidatos.
- Comprender el contexto del sistema.

- Capturar requisitos funcionales.
- Capturar requisitos no funcionales.(JAMES RUMBAUGH 2000)

Debidamente después de tener bien establecidos cuales son los requisitos funcionales y no funcionales del sistema se debe pasar a la construcción de los casos de uso con sus respectivas descripciones y/o historias de usuario. Esto se debe a que son estos artefactos los que describen la solución a los requerimientos del sistema.

En caso de los desarrolladores decidirse por RUP es importante realizar una serie de artefactos que propone la metodología los cuales son fundamentales para la interacción con el cliente y de esta forma llegar a un acuerdo más preciso sobre el sistema. Los artefactos fundamentales son: Modelo de Casos de Uso el cual proporciona la entrada fundamental para el análisis, el diseño y las pruebas, y el Prototipo de Internas de Usuario el cual ayuda a comprender las interacciones entre el usuario y el sistema. Para un mejor conocimiento de estos artefactos se propone el estudio del capitulo 7 del Proceso Unificado de Software.

Como se menciona en el Capítulo 1 XP cuenta con un análisis y diseño bastante escaso, por lo que en caso de decidirse por esta metodología las historias de usuarios deben ir redactada lo más claro posible y de manera que se mencionen pocos requisitos en cada una de ellas. En esta metodología la captura de requisitos tiene una importancia extrema ya que se realiza de forma constante, generalmente es el comienzo de cada iteración.

Actividad 21: Revisión técnica formal (interna).

Responsable: Asesor de Calidad.

Participantes: Asesor de Calidad, Analista principal, Jefe de proyecto, Especificador de Casos, Diseñador Gráfico, Arquitecto, Jefe de Calidad.

Misión: Revisión interna con el asesor de calidad con el objetivo de comprobar la calidad de la Captura de Requisitos y la confección de los Casos de Uso o Historia de Usuario.

Entradas:

- Especificación de requisitos funcionales y no funcionales.

- Descripción de casos de uso.

- Modelo de casos de uso.

- Interfaz de Usuario.

- Historias de Usuario en caso de usar XP.

- Glosario de Términos

Salidas:

- Glosario de Términos.

- Acta de la reunión.

- Documento de no conformidades.

Actividad 22: Revisión con el cliente.

Responsable: Jefe de Proyecto

Participantes: Jefe de Proyecto, Cliente, Analista principal, Especificador de Casos, Diseñador Gráfico, Arquitecto, Jefe de Calidad.

Misión: Revisión con el cliente de la captura de requisitos, confección de los casos de uso, historias de usuarios, modelado de los casos de uso y prototipo de interfaz de usuario.

Entradas:

- Especificación de requisitos funcionales y no funcionales.

- Descripción de casos de uso.

- Modelo de casos de uso.

- Interfaz de Usuario.

- Historias de Usuario en caso de usar XP.

- Glosario de Términos.

Salidas:

- Glosario de Términos.

- Documento de no conformidades (Registros)

- Acta de aceptación del cliente.

Actividad 23: Análisis y Diseño.

Responsable: Analista Principal.

Participantes: Analista Principal, Diseñador, Arquitecto principal, Diseñador de Base de Datos, Jefe de Proyecto

Misión: Se realiza el análisis para investigar el problema al cual se le planteará una solución lógica en el diseño.

Entradas:

- Descripción de Casos de Uso.

Salidas:

- Propuesta de la Arquitectura.

- Diseño de interfaz de usuario.

- Documentos de realización de Casos de Uso.

- Diagrama de Clases (Análisis y diseño).

- Modelo de datos.

- Diagrama de despliegue.

- Documento final de arquitectura.

En esta actividad se realiza el proceso de análisis del sistema. Se define la arquitectura se diseña la interfaz de usuario, se identifican las clases relacionadas con cada caso de uso o historias de usuarios, sus atributos y relaciones. Se procede a elaborar el documento de realización de los casos de uso, con la información de este se construye todo el diagrama de clases orientado a los conceptos y relaciones de estos en el dominio. Se diseña la Base de datos en productos que lo requieran. Las clases se dividen en paquetes, se hace la distribución en nodos físicos del sistema. Posteriormente se elabora el documento final donde se describe la arquitectura del sistema con sus artefactos más significativos.

Actividad 24: Revisión técnica formal (interna).

Responsable: Jefe de Calidad.

Participantes: Jefe de Calidad, Analista Principal, Diseñador, Arquitecto principal, Diseñador de Base de Datos, Jefe de Proyecto

Misión: Revisión interna con los encargados con la actividad de análisis y diseño y el Jefe del grupo de Calidad para comprobar la calidad de la misma.

Entradas:

- Propuesta de la Arquitectura.
- Diseño de interfaz de usuario.
- Documentos de realización de Casos de Uso.
- Diagrama de Clases.
- Modelo de datos.
- Diagrama de despliegue.
- Documento final de arquitectura.
- Glosario de Términos

Salidas:

- Glosario de Términos.
- Acta de la reunión.
- Documento de no conformidades.

Actividad 25: Revisión con el cliente.

Responsable: Jefe de Proyecto.

Participantes: Jefe de Proyecto, Analista principal, Jefe del grupo de Calidad.

Misión: Revisión con el cliente de los documentos y artefactos generados por el análisis y diseño

Entradas:

- Propuesta de la Arquitectura.

- Diseño de interfaz de usuario.

- Documentos de realización de Casos de Uso.

- Diagrama de Clases.

- Modelo de datos.

- Diagrama de despliegue.

- Documento final de arquitectura.

- Glosario de Términos

Salidas:

- Glosario de Términos

- Documento de no conformidades (Registros)

- Acta de aceptación del cliente.

Después de una correcta etapa análisis y diseño se le da paso a la actividad que le dará vida al sistema: la implementación.

Los objetivos de la implementación son:

- Planificar las integraciones de sistemas necesarias en cada iteración, para la cual se debe tener un enfoque incremental. XP sacará mayor provecho se este objetivo debido a su integración a corto plazo y de forma constante.
- Distribuir el sistema asignando componentes ejecutables a los diferentes nodos del diagrama de despliegue. En el caso de los juegos los nodos consistirían mayormente en PC servidoras y PC cliente, sin excluir alguna u otra variante. En el caso de los simuladores es algo más amplia la distribución teniendo en cuenta que existen dispositivos especiales como son los casos de los cascos, guantes, autos especializados entre otros.
- Implementar las clases y subsistemas encontrados durante el diseño.

- Probar los componentes individuales, a continuación integrarlos compilándolos y enlazándolos a uno o más ejecutables.

Actividad 26: Implementación y pruebas de unidad.

Responsable: Jefe del grupo de programadores

Participantes: Jefe del grupo de programadores, programadores, Jefe de Diseño de Base de Datos, diseñadores de base de datos, Jefe grupo de Calidad, grupo de calidad, Arquitectos, Ingeniero de Componentes.

Misión: Convertir los elementos del diseño en elementos de implementación (ficheros fuentes, ejecutables y otros). Realizar pruebas de unidad a los componentes desarrollados. Ejecutar integración a medida que se le adjuntan nuevas funcionalidades al sistema.

Entradas:

- Modelo de Clases del Diseño

- Historias de Usuario (en caso de utilizar XP)

- Plan y Guión de Pruebas.

Salidas:

-Código fuente y ejecutable integrado y revisado.

En esta actividad, el sistema entra en un ciclo, ya que esta acción es la última de la etapa de elaboración del sistema o sus versiones. Después de que en alguna iteración se termine esta actividad, se tendrá que ver si se llegó a la ultima funcionalidad de la versión que se desea entregar, de ser positivo se realizarían las pruebas de integración, en caso contrario, se retornaría a las actividades de análisis y diseño.

Actividad 27: Prueba de integración de la Versión.

Responsable: Jefe de Proyecto.

Participantes: Jefe de Proyecto, Jefe de Programadores, grupos de programadores, Jefe de Calidad, grupo de calidad

Misión: Integrar los resultados producidos por programadores o grupos de programadores individuales en un solo sistema ejecutable Planificar la pruebas necesarias.

Entradas:

- Código fuente y ejecutables.

- Toda la documentación generada hasta el momento durante el desarrollo (casos de uso, historias de usuario etc.)

- Glosario de Términos.

Salidas:

- Sistema probado.

- Documento de no Conformidades.

Las construcciones en las que se detecten defectos son probadas de nuevo y posiblemente devueltas a otras actividades anteriores de implementación.

Actividad 28: Prueba de integración del sistema.

Responsable: Jefe de Calidad.

Participantes: Jefe de Calidad, grupo de calidad

Misión: Probar las Integraciones producidas por programadores o grupos de programadores individuales en un solo sistema ejecutable.

Entradas:

- Código fuente y ejecutables.

- Toda la documentación generada hasta el momento durante el desarrollo (casos de uso, historias de usuario etc.)

- Glosario de Términos.

Salidas:

- Sistema probado.

- Documento de no Conformidades.

La prueba de integración es realizada después de integrar los diferentes módulos con que puede contar el sistema, ya en este caso no se estaría hablando de iteraciones pequeñas, si no de una versión de mayores funcionalidades.

Actividad 29: Prueba de liberación del grupo de calidad.

Responsable: Jefe de Calidad.

Participantes: Jefe de Calidad, grupo de calidad.

Misión: Procurar que el software no llegue a manos del clientes con defectos o falto de funcionalidades.

Entradas:

- Sistema.

- Manual de usuario.

- Glosario de Términos.

Salidas:

- Sistema (probado y aprobado).

- Manual de Usuario (probado y aprobado).

Dentro de los objetivos fundamentales de esta actividad están diseñar e implementar las pruebas creando los casos de pruebas del sistema. Realizar las diferentes pruebas y revisar los resultados de las pruebas sistemáticamente. Realizar las pruebas de liberación del producto final con el objetivo de tenerlo listo para la entrega al cliente.

Actividad 30: Prueba de aceptación del cliente.

Responsable: Jefe de Proyecto

Participantes: Jefe de Proyecto, Cliente, Jefe de Calidad.

Misión: Entrega la Cliente del producto terminado, para ser probado por el mismo y empleado.

Entradas:

- Sistema terminado.

- Manual de Usuario.

- Documentación del Proyecto.

Salidas:

- Acta de entrega.

Con la actividad de prueba de aceptación del cliente concluye el proceso aunque quizás no el software ya que el proceso fue concebido iterativo e incremental, basado en los principios de RUP y XP. Los desarrolladores pueden cargar las diferentes etapas de elaboración según definan por conveniencia en cada iteración y es posible que la primera versión aprobada por el cliente solo sea la interfaz con pequeñas funcionalidades, en el caso de un juego tal vez sea solo un nivel, o de un simulador un demo bastante limitado. Por lo que las posteriores iteraciones se encargaran de enriquecer la siguiente versión.

Seguidamente después de realizar una entrega y ser probada por el cliente, se procedería a recomenzar la etapa de elaboración del sistema para otra versión en la actividad que según los desarrolladores sea necesaria, solo en caso que se haya acordado o falten requisitos a los cuales darles solución, por lo que además se irá surtiendo todo el proceso de planificación y análisis de riesgo del proyecto actualizando las tareas acordadas.

3.2 Validación:

Después de la propuesta del proceso fue preciso realizar una etapa de validación, con el objetivo de conocer la aceptación por los jefes de proyecto y el vicedecano de producción de la facultad 5 de la UCI donde sería aplicado el flujo.

La validación se realizó mediante una encuesta donde se hicieron 4 preguntas de los tipos directas y semi-cerradas. El cuestionario (Anexo 2) fue presentado a tres líderes de proyecto y al vicedecano a los cuales se les realizaron las siguientes interrogantes:

1. ¿Cree UD que la propuesta del proceso está a la altura de las necesidades de los proyectos de Realidad Virtual?

   Si___ No___

La respuesta de los cuatro entrevistado fue positiva, por lo que se considera apta la propuesta.

2. Con la propuesta del proceso definida en el trabajo, ¿cree UD que se puede lograr eficacia en un proyecto de Realidad Virtual?

   Si___ No___

De igual forma la contestación fue Si en las cuatro personas teniendo en cuenta el concepto de eficacia, donde se asegura que con el proceso al menos se va a entregar un producto terminado al cliente con los requerimientos que el definió.

3. ¿Agregaría o eliminaría UD alguna actividad descrita en la propuesta?

   Si___ No___

Ninguno de los compañeros quiso aumentar o eliminar actividades del proceso, aunque alegan que requiere de un estudio mucho más profundo y de experiencia, faltaría poner la propuesta en práctica para poder ver verdaderamente las ventajas y defectos que podría tener el proceso.

4. ¿Qué valoración en general diera UD a la propuesta desarrollada? ¿En que escala del 1 al 3 colocaría UD la propuesta desarrollada?

   - 1: nivel bajo de creatividad y efectividad.
   - 2: nivel medio de creatividad y efectividad.
   - 3: nivel alto de creatividad y efectividad.

En la respuesta de esta pregunta vuelven a coincidir los cuatro encuestados y en este caso con la opción 3, aunque la efectividad es poco palpable en estos momentos, debido a que el proceso no ha sido llevado a cabo hasta el momento por ningún proyecto de la facultad, aunque la propuesta está bien concebida y fundamentada ya que en la misma hay la participación de varias personas que de alguna forma han participado en la dirección de proyectos de realidad virtual.

Otras opiniones de lo líderes de proyecto fueron que consideran que con el proceso propuesto se lograría ganar mayor organización con el cliente, este formaría parte casi del grupo de desarrolladores y se vería más comprometido, de la misma forma el equipo de desarrolladores comprendería mejor los requerimientos.

Conclusiones parciales:

En este capítulo se ha descrito de manera detallada el proceso propuesto para la producción de software de realidad virtual en la facultad 5 de la UCI, explicando la misión de cada una de sus actividades así como sus participantes y artefactos de salida y entrada. Se tuvo en cuenta para la elaboración las opiniones de diferentes líderes de proyectos y el estudio de las metodologías de desarrollo que de alguna forma han guiado la etapa de elaboración de proyectos de entornos virtuales.

Además se validó la propuesta mediante una encuesta realizada al vicedecano de producción de la facultad 5 y a tres jefes de proyectos, donde los cuatro compañeros estuvieron de acuerdo con que sea llevado a cabo el proceso en proyectos de realidad virtual e irlo mejorando según las deficiencias encontradas.

Conclusiones:

Después del estudio de las diferentes metodologías estudiadas, principalmente Programación Extrema (XP), El Proceso Unificado de Software (RUP) y el Método de Larman, las cuales son las que aparecen reflejadas en este documento. Seguidamente de haber analizado el proceso de producción de la facultad 5 en entornos virtuales hasta el momento, de haber entrevistado y encuestado a los líderes de proyectos y encontrar sus principales deficiencias y examinado sus inquietudes, se realiza la propuesta del Proceso de Software para Realidad Virtual en la facultad 5 de la Universidad de las Ciencias Informáticas.

El proceso propuesto estuvo concebido bajo los principios heredados de RUP y XP, siendo el mismo iterativo e incremental, teniendo una retroalimentación constante tanto internamente en el grupo de trabajo como con el cliente, y concibiendo la importancia de las personas y el trabajo en equipo por encima del entorno de trabajo. La propuesta cuenta con un flujo de 30 actividades de ellas 17 fundamentales, 8 reuniones técnicas formales y 5 revisiones con el cliente

Recomendaciones:

1. Emplear el proceso propuesto en los proyectos de Realidad Virtual de la Facultad 5 de la Universidad de las Ciencias Informáticas.
2. Que se profundice en el estudio de cada actividad del proceso enfocado en sistemas de Realidad Virtual.
3. Mejorar el proceso a partir de la retroalimentación obtenida por las experiencias de trabajo.
4. Establecer métricas que midan la eficiencia y eficacia del proceso propuesto.

Referencia Bibliográfica.

FRANKLIN RENE RIVERO GARCÍA, Y. C. V., JENNY INFANTE FRÓMETA XP una buena alternativa para proyectos pequeños, 2006.

GONZALO MÉNDEZ POSO, M. I. S. S., ANGÉLICA DE ANTONIO JIMÉNEZ Hacia una Metodología de Desarrollo para la Construcción de Entronos Virtuales.

GUIDO, L. E. F. *Guía para diseñar y procesar encuestas en organismos públicos*. GOBIERNO, Subsecretaría de la Gestión Pública, Buenos Aires, Argentina 2007.

JAHN, G. V. *CURSO VRML*.

JAMES RUMBAUGH, I. J., GRADY BOOCH. *El Proceso Unificado de Software (RUP)*. 2000. p.

JOSÉ H. CANÓS, P. L., MA. CARMEN PENADÉS Metodologóas Ágiles en el Desarrollo de Software: 8.

UCI. *Conferencia 1: Introducción a la Ingeniería de Software*, Curso 2005- 2006. p.

VALENCIA, U. P. D. Introducción al RUP, 2006: 22.

VRML y Realidad Virtual.

Wikipedia. 2007.

Anexos.

Compañero (a):

El Trabajo de Diploma titulado “Propuesta del Proceso de producción de Software de Realidad Virtual” del autor Ignais La Paz Trujillo, se encuentra haciendo la presente encuesta para conocer criterios sobre los proyectos, específicamente sobre el control y documentación de las informaciones relativas al mismo. No es necesario que ponga su nombre, sólo le pedimos su más sincera respuesta. Lea detenidamente cada una de las preguntas.

Muchas gracias.

1. Necesitamos que nos diga cuál es el nombre del proyecto que usted dirige:

   a. _____

2. Qué metodología ustedes utilizan en el desarrollo de los productos dentro del proyecto productivo.

   1. ( ) RUP
   2. ( ) XP
   3. ( ) Otras. Cuál o cuáles?
      _____
      _____
      _____
      _____

¿Dentro del las producciones en el proyecto al cual pertenecen realizan control de versiones?

1. ( ) Si
2. ( ) No

En el caso de haber dado una respuesta afirmativa anteriormente, pudiera explicarnos el mecanismo que utilizan.

_____

_____

_____

_____

¿Lleva algún tipo de documentación en su proyecto?

1. ( ) Si
2. ( ) No

En caso de ser afirmativo pudiera mencionar algún tipo de documentación que lleve.

_____

_____

_____

_____

Si en su proyecto se aplica la Metodología RUP, mencione la documentación y artefactos que considere necesario.

_____

_____

_____

Si en su proyecto se aplica la Metodología XP, ¿archiva las historias de los usuarios?

1. ( ) Si
2. ¿Cómo lo realiza?

   _____

_____
_____

3. ( ) No

Datos de control

1. ( ) Profesor
2. ( ) Profesor en proyecto
3. ( ) Jefe de proyecto
4. ( ) Líder de proyecto
5. ( ) Estudiante en proyecto
6. ( ) Estudiante jefe o líder de proyecto

Edad

1. ( ) Menos de 20 años
2. ( ) De 21 a 30 años
3. ( ) De 31 a 40 años
4. ( ) De 41 a 50 años
5. ( ) De 51 a 60 años
6. ( ) Más de 60 años

10. Nivel escolar

1. ( ) Universitario
2. ( ) Técnico medio
3. ( ) Curso estudios superiores

11. Sexo

12. Provincia en la que vive

_____

1. ( ) Masculino
2. ( ) Femenino

Anexo 1: Encuesta realizada a los líderes de proyectos.

Compañero (a):

El Trabajo de Diploma titulado “Propuesta del Proceso de producción de Software de Realidad Virtual” del autor Ignais La Paz Trujillo, se encuentra haciendo la presente encuesta para el proceso de validación de la propuesta. No es necesario que ponga su nombre, sólo le pedimos su más sincera respuesta. Lea detenidamente cada una de las preguntas.

1. ¿Cree UD que la Propuesta del proceso está a la altura de las necesidades de los proyectos de Realidad Virtual?

Si___ No____

2. Con la Propuesta del proceso definida en el trabajo, cree UD que se pueda lograr eficacia en un proyecto de Realidad Virtual.

Si___ No____

3. ¿Agregaría o eliminaría UD alguna actividad descrita en la propuesta?

Si____ No___

¿Qué valoración en general diera UD a la propuesta desarrollada?

4. ¿En que escala del 1 al 3 colocaría UD la propuesta desarrollada?

- 1: nivel bajo de creatividad y efectividad.
- 2: nivel medio de creatividad y efectividad.
- 3: nivel alto de creatividad y efectividad.

Anexo 2: Encuesta de validación a los líderes de proyecto.