EL USO DEL UML EN LA FASE DE ANÁLISIS DEL PROCESO DE DESARROLLO DE UN SOFTWARE EDUCATIVO

USE OF UML IN THE ANALYSIS PHASE OF A DEVELOPMENT PROCESS FOR EDUCATIONAL SOFTWARE

Recibido: 18 de febrero del 2011
Aprobado: 30 de marzo del 2011

EDWIN DURAN-BLANDÓN*

Resumen

El Lenguaje Unificado de Modelado (UML) se deriva de una serie de métodos de análisis y diseño orientado a objetos de cualquier software, que ofrecen un marco de trabajo o enfoque necesario para la construcción de software educativo. El autor presenta una reflexión derivada de la investigación "Incidencia de las TIC en los procesos enseñanza-aprendizaje de las ciencias y matemáticas en la Institución Ciudadela Educativa del Magdalena Medio". La investigación está en proceso desde el 2010 y se enfoca en el uso del UML para el desarrollo de software educativo en la institución antes nombrada.

Palabras clave: ingeniería del software, Lenguaje Unificado de Modelado (UML), software educativo.

Abstract

Unified Modelling Language (UML) derives from a series of analysis and object oriented design methods for any kind of software, these provide a framework or approach necesary for the construction of educational software. The author presents a work derived from the research: "Incidence of the TIC in Teaching and Learning Processes for Sciences and Mathematics in the Institución Ciudadela Educativa del Magdalena Medio". The research is in progress since 2010 and focuses in the use of UML for the developpment of educational software in this institution.

Keywords: software engineering, Unified Modelling Language (UML), educational software.

• Cómo citar este artículo: Edwin Duran-Blandón. "El uso del UML en la fase de análisis del proceso de desarrollo de un software educativo". *Revista Ingeniería Solidaria*, vol. 7, núms. 12-13, 2011, pp 83-91.

* Ingeniero de Sistemas de la Universidad Cooperativa de Colombia, sede Barrancabermeja. Especialista en Tecnologías Avanzadas para el Desarrollo de Software de la Universidad Autónoma de Bucaramanga. Candidato a Magíster en Tecnología Educativa de la Universidad Autónoma de Bucaramanga y el Instituto Tecnológico de Monterrey (UNAB-ITESM). Tutor en Ambientes Virtuales de Aprendizaje. Docente Medio Tiempo de la Universidad Cooperativa de Colombia, sede Barrancabermeja. Docente de Educación Media en el Colegio Ciudad Educativa del Medio Magdalena.
Correos electrónicos: edubla01@yahoo.es, edwin.duran@campusucc.edu.co

Introducción

El Lenguaje Unificado de Modelado (Unified Modeling Languaje [UML]) se deriva de una serie de métodos de análisis y diseño, orientada a objetos. Se originó a fines de los ochenta y principios de los noventa, y no fue concebido como un método en sí mismo, sino como la notación básicamente gráfica que cualquier metodología o proceso de software puede utilizar para expresar sus productos de análisis y diseño. Sin embargo, esta notación nació con el famoso proceso de software denominado *proceso unificado*, resultado del trabajo de los llamados "tres amigos". Grady Booch, Jim Rumbaugh e Ivar Jacobson [1]. Booch [1] describe cómo este proceso de software con la notación UML permite el análisis y diseño orientado a objetos de cualquier software.

Por otra parte, el desarrollo de software educativo no está exento del uso de la notación UML, ya que como cualquier otro software, requiere para su construcción de una fase de análisis y diseño. Por ello, el uso del lenguaje de modelado permite al equipo multidisciplinario de desarrollo del software educativo comunicarse entre sí. Al analizar un diseño, lo que el equipo necesita es un lenguaje de modelación más que el proceso seguido para lograr tal diseño, siendo muy conveniente utilizar dicha notación con cualquier proceso o metodología de desarrollo, que para el caso de este artículo se mostrará su uso con la metodología de Ingeniería de Software Educativo, propuesta por Galvis Panqueva [2], describiendo una experiencia en el desarrollo del modulo Software Educativo de la Especialización Multimedia para la Docencia.

El software educativo

Antes de empezar a analizar cómo se desarrolla un software educativo, específicamente cómo la notación UML es utilizada en su análisis y diseño, es conveniente aclarar y comprender el concepto "software educativo", y para ello se presentan dos definiciones importantes:

- El Dr. Pere Marqués [3] utiliza los términos software educativo, programas educativos y programas didácticos como sinónimos. Proporciona la definición siguiente: "Software educativo se denomina a los programas para computadoras creados con la finalidad específica de ser utilizados como medio didáctico, es decir, para facilitar los procesos de enseñanza y de aprendizaje".
- Galvis Panqueva [2] denomina "software educativo a aquellos programas que permiten cumplir o apoyar funciones educativas".

Las anteriores definiciones describen claramente a qué se hace referencia cuando se habla de software educativo, que es motivo de estudio en la ingeniería del software, que ofrece metodologías para su desarrollo.

El UML en el análisis y diseño de software educativo

En este apartado no se pretende explicar en detalle el Lenguaje de Modelación Unificado, pero sí resaltar los aportes de su aplicación, es decir, rescatar principalmente la comunicación que posibilita el UML entre los que diseñan y desarrollan el software educativo, y quienes lo solicitan.

Se tiene, por tanto, que uno de los mayores requerimientos en el proceso de desarrollo de software educativo es el de elaborar un sistema *ad hoc*, que atienda y resuelva las necesidades de los usuarios a un costo bajo y con calidad. Esta labor se vuelve compleja porque nuestro lenguaje formalizado debe serle inteligible a los demás miembros del equipo, docentes, psicólogos en educación, expertos en el dominio del tema, diseñadores gráficos, ingenieros de software, entre otros, que deben comprender las alternativas que se discuten para diseñar un software educativo.

Es así que lograr una buena comunicación, aparte de establecer una adecuada comprensión de los requerimientos del usuario final, es el punto de partida para el desarrollo del software educativo. La técnica que provee para este caso el UML es la conocida como los *casos de uso*.

Un caso de uso es toda funcionalidad que el software ofrecerá a los usuarios. La acumulación de todos los casos de uso constituyen la integridad del sistema, lo que en suma permitirá una explicación de lo que el software educativo podrá hacer.

Los casos de uso son la esencia para comprender lo que quieren los usuarios finales y quienes solicitan el software. Los casos de uso son entes posibles de asumir el desarrollo iterativo, que es en sí mismo una técnica valiosa, puesto que retroalimenta de manera reiterativa a los miembros del equipo sobre el rumbo que va tomando el resultado del desarrollo.

Aparte de que los casos de uso permiten la comunicación de los elementos superficiales, también se convierten en elementos clave para observar las cuestiones más profundas. Esto implica saber cómo entienden su mundo los expertos del dominio, en este caso, los profesores. Una herramienta fundamental que ofrece el UML en esta parte lo constituyen los diagramas de clases, que son muy valiosos en la medida en que se usen de modo conceptual.

En otras palabras, debe tratarse cada clase como si fuese un concepto en la mente del usuario, como parte de su lenguaje. Los diagramas de clase que se diseñan no son, por tanto, diagramas de datos o de clase, sino diagramas del lenguaje de los usuarios.

Otro elemento importante que ofrece el uml lo constituyen los diagramas de actividades, útiles en los casos en que los procesos de flujo de trabajo son una parte importante del mundo de los usuarios. Dado que los diagramas de actividades manejan procesos paralelos, pueden ayudar a deshacerse de secuencias innecesarias en el diseño del software.

Si bien lo descrito anteriormente corresponde a los diversos modelos que se pueden realizar con el uml, a continuación se presentan ciertos objetivos que deben perseguir a la hora de realizar un análisis de un software educativo:

- Comprender el problema, objetivos, contenidos y situaciones de enseñanza-aprendizaje que tendrá que atender la aplicación.
- Suscitar cuestiones relevantes acerca los requerimientos educativos y las respuestas que pueda dar el sistema.
- Proporcionar una base para responder preguntas acerca de propiedades específicas del problema a atender y del sistema.
- Decidir lo que tiene que hacer el sistema.
- Decidir lo que no tiene que hacer el sistema.
- Asegurar que el sistema satisfaga las necesidades de los usuarios y definir los criterios de aceptación.
- Proporcionar una base para el desarrollo del sistema.

Sin lugar a dudas, dichos objetivos pueden ser alcanzados cuando la notación uml es utilizada de forma adecuada en la fase de análisis de cualquier proceso de software.

El proceso de desarrollo de software educativo

La ingeniería del software ofrece las metodologías, herramientas y técnicas para desarrollar software. Estas metodologías son llamadas también modelos de procesos de software [4], y dan las pautas para obtener un software y sus productos asociados como son la documentación, código fuente, etcétera.

Dentro del ámbito del desarrollo de software educativo existen metodologías y procesos de desarrollo de software genéricos como el *proceso unificado de desarrollo de software*, y específicos al dominio educativo, como la metodología de *ingeniería de software educativo*, propuesta por Galvis Panqueva [2] y la metodología para el desarrollo de software educativo, propuesta por S. Gustavo Peláez Camarena [5].

Metodología de *proceso unificado*

La Metodología de Proceso Unificado (pu), de Jacobson, Booch, Rumbaugh [6], usada para preparar todos los esquemas del sistema, requiere de los casos de uso para definir las necesidades del usuario; además, utiliza la arquitectura software para describir el sistema en construcción y, finalmente, estudia paso a paso el incremento y las mejoras que se presentan en la construcción de sistema software. En la figura 1 se pueden apreciar las fases de este proceso de software.

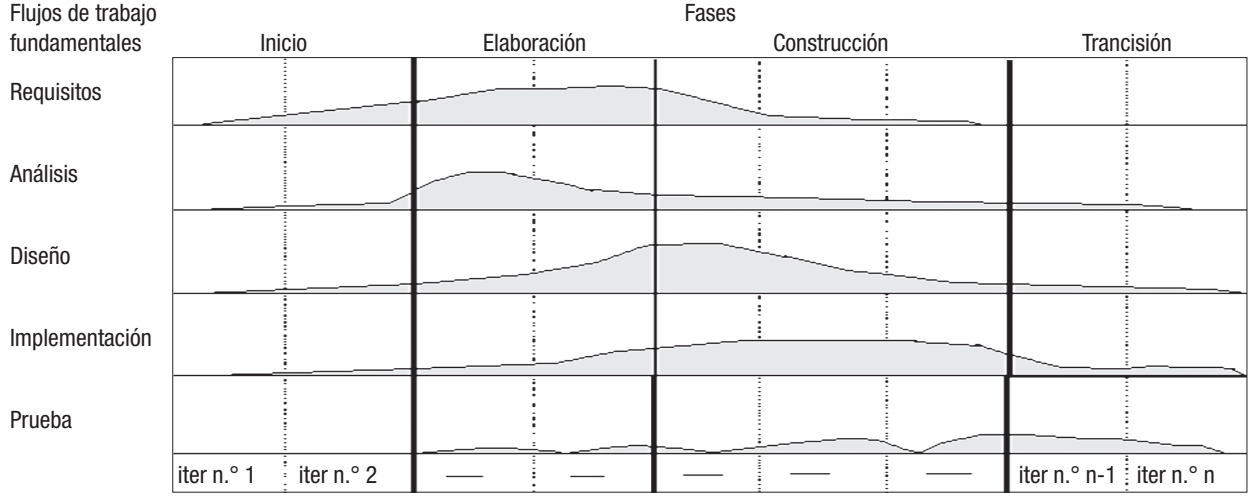

Figura 1. Proceso unificado de desarrollo

Fuente: I. Jacobson, G. Booch y J. Rumbaugh [6]

De acuerdo con la metodología para desarrollar un software, se debe empezar con la fase de inicio en la cual se debe hacer el levantamiento de requisitos, definiendo los requerimientos funcionales y no funcionales, analizando y diseñando los casos de uso del sistema, que para el caso de un software educativo especifique lo que le permitirá hacer el nuevo sistema a los alumnos y estudiantes. En esa fase se crean los diagramas de casos de uso que especifica las funcionalidades del sistema, en el que se definen los actores que participan.

Posteriormente, en la fase de elaboración se analiza y diseña la arquitectura del sistema, la cual, en el caso de un software educativo, puede ser de tres capas. Aquí se definen las herramientas de programación necesarias para implementar el aplicativo, se elabora el diseño conceptual y lógico la base de datos que soportará la herramienta software, así como los diagramas de actividades que describían a fondo los casos de uso definidos y los diagramas de secuencia que mostraban la interacción entre las clases del sistema. En esta fase se deben incluir los aspectos del diseño educativo, los cuales contemplan contenidos, actividades y recursos que utilizará el software.

Luego, en la fase de construcción, se determina cuántos incrementos se desarrollarán con el fin de hacer entregas parciales a los usuarios finales, ya que cada incremento debe ser funcional. También se definen qué herramientas de programación se utilizarán.

A cada incremento se le aplica un ciclo de vida terminado con las pruebas de cada uno y su funcionamiento.

Finalmente, en la fase de transición se realizaron pruebas de funcionamiento, en las que se debe hacer una prueba piloto o de campo. Para ello se escoge una muestra de docentes y estudiantes que utilicen el software para medir su funcionalidad. Esta fase arroja el manual de usuario y el manual de instalación.

Metodología de ingeniería de software educativo propuesta por Galvis Panqueva

Por otra parte, se encuentran metodologías específicas para el desarrollo de software educativo, como es el caso de la metodología de la *ingeniería de software educativo*, propuesta por Panqueva. A continuación (figura 2), se muestran los pasos de la metodología:

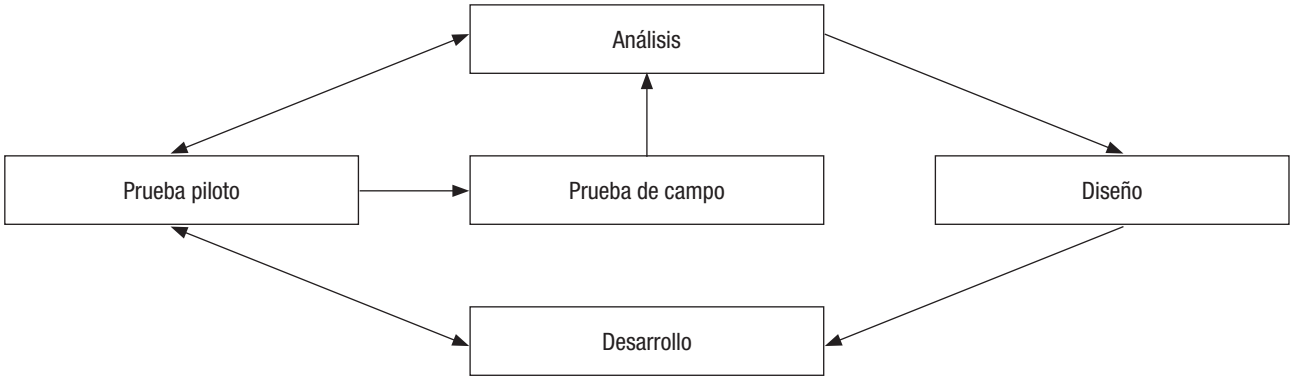

Figura 2. Metodología de desarrollo de Alvaro Galvis Panqueva
Fuente: Galvis Panqueva, Alvaro [2]

Las fases de dicha metodología son las siguientes:

La metodología parte del análisis de necesidades educativas: aquí todo Medio Educativo Computarizado (MEC) debe cumplir un papel relevante en el contexto en el que se utilice. Su incorporación a un proceso de enseñanza/aprendizaje no puede guiarse por criterios no académicos como que el MEC "es chévere" o que "está disponible". Estas y otras razones probablemente lleven a dedicar recursos a labores que no producen los mejores resultados.

A diferencia de las metodologías asistemáticas, en las que se parte de ver de qué soluciones disponemos para luego establecer para qué sirven, de lo que se trata aquí es de favorecer, en primera instancia, el análisis de qué problemas o situaciones problemáticas existen, sus causas y posibles soluciones, para entonces sí determinar cuáles de estas últimas son aplicables y pueden generar los mejores resultados. Aquí, entonces, se pretende responder a las preguntas: ¿cómo identificar las necesidades o los problemas existentes?, ¿qué criterios usar para llegar a decidir si amerita una solución computarizada?, ¿con base en qué decidir si se necesita un MEC y qué tipo de MEC conviene que sea, para satisfacer una necesidad dada?

En esa fase el uso de un lenguaje adecuado para definir las necesidades de los usuarios es de vital importancia, por eso el UML es útil en esta fase mediante el uso de los casos de uso, con lo que cualquier usuario sabría qué debe hacer el software.

Fase de selección o planeación del desarrollo de MEC

El proceso de análisis de necesidades educativas que ameritan ser atendidas con MEC no termina aún. Falta establecer si existe o no una solución computarizada que satisfaga la necesidad que se detecta, en cuyo caso podría estar resuelta, o si es necesario desarrollar un MEC para esto. Cuando se identifican uno o más paquetes que parecen satisfacer las necesidades, es imprescindible someterlos al ciclo de revisión y prueba de MEC que asegure que al menos uno de ellos satisface la necesidad. Para esto, es indispensable tener acceso a una copia documentada de cada MEC, como etapa final de la fase de análisis, y hacerla revisar por expertos en contenido, metodología e informática. Los primeros, para garantizar que efectivamente corresponde al contenido y objetivos de interés; los expertos en metodología para verificar que el tratamiento didáctico es consistente con las estrategias de enseñanza/aprendizaje que son aplicables a la población objeto y al logro de tales objetivos; los expertos en informática para verificar que dicho MEC se puede ejecutar en la clase de equipos de que dispondrán los alumnos y que hace uso eficiente de los recursos computacionales disponibles. Si todo esto se cumple, habrá terminado el análisis con al menos un MEC seleccionado para atender la necesidad.

Cuando no se identifica un MEC con el cual satisfacer la necesidad, la fase de análisis culmina con la formulación de un plan para llevar a cabo el desarrollo del MEC requerido. Esto implica consultar los recursos disponibles y las alternativas de usarlos para cada una de las etapas siguientes. Se debe prever, tanto lo referente a personal y tiempo que se dedicará a cada fase, así como los recursos computacionales que se requieren para cada fase en particular las de desarrollo, y pruebas piloto y de campo.

Ciclos para la selección o el desarrollo de MEC

En la fase anterior se dio a entender la razón de ser del doble ciclo, para selección o desarrollo de MEC. El punto de partida de ambos ciclos es la identificación de necesidades educativas reales que conviene atender con material educativo computarizado. Dependiendo del resultado final de esta etapa, se procede en el sentido contrario al avance de las manecillas del reloj, cuando se trata de seleccionar un MEC; pero en el mismo sentido del avance del avance de las manecillas, si conviene efectuar su desarrollo. En cualquiera de los dos ciclos, una vez que se dispone de un MEC, se requiere evaluarlo con un grupo piloto de alumnos que pertenezca a la población objeto, bajo las condiciones para las cuales está diseñado. Esta es la base para decidir si el MEC debe llevarse a la práctica en gran escala, o para echar pie atrás, rediseñarlo, ajustarlo o desecharlo. Durante su implementación, también es importante que se evalúe el MEC, de modo que se pueda establecer la efectividad real del material: este es el sentido de la prueba de campo.

Diseño de MEC

El diseño de un MEC está en función directa de los resultados de la etapa de análisis. La orientación y contenido de este se deriva de la necesidad educativa o problema que justifica el contenido y habilidades que subyacen en esto, así como de lo que se supone que un usuario del MEC ya sabe sobre el tema. El tipo de software establece, en buena medida, una guía para el tratamiento y funciones educativas que es deseable que el MEC cumpla para satisfacer la necesidad.

Entorno para el diseño del MEC

A partir de los resultados del análisis, es conveniente hacer explícitos los datos que caracterizan el entorno del MEC que se va a diseñar: destinatarios, área del contenido, necesidad educativa, limitaciones y recursos para los usuarios del MEC, equipo y soporte lógico que se va a utilizar.

Entorno del diseño

¿A quiénes se dirige el MEC? ¿Qué características tienen sus destinatarios? ¿Qué área de contenido y unidad de instrucción se beneficia con el estudio del MEC? ¿Qué problemas se pretende resolver con este? ¿Bajo qué condiciones se espera que los destinatarios usen el MEC? ¿Para un equipo con las características físicas y lógicas conviene desarrollarlo? A lo anterior hay que agregarle un diseño educativo del MEC. El diseño educativo debe resolver los interrogantes que se refieren al alcance, contenido y tratamiento que debe ser capaz de apoyar el MEC. Otro aspecto a tener en cuenta es el diseño de comunicación, en el que la zona de comunicación en la que se maneja la interacción entre usuario y programa se denomina interfaz. Para especificarla, es importante determinar cómo se comunicará el usuario con el programa, estableciendo mediante qué dispositivos y usando qué códigos o mensajes (interfaz de entrada); también se hace nece-

sario establecer cómo el programa se comunicará con el usuario, mediante qué dispositivos y valiéndose de qué códigos o mensajes (interfaz de salida). Y, finalmente, el diseño computacional, que con base en las necesidades establece qué funciones es deseable que cumpla el MEC en apoyo de sus usuarios, el profesor y los estudiantes. Entre otras cosas, un MEC puede brindarle al alumno la posibilidad de controlar la secuencia, el ritmo, la cantidad de ejercicios, de abandonar y de reiniciar. Por otra parte, un MEC puede ofrecerle al profesor la posibilidad de editar los ejercicios o las explicaciones, de llevar registro de los estudiantes que utilizan el material y del rendimiento que demuestran, de hacer análisis estadísticos sobre variables de interés, etcétera. La estructura lógica que comandará la interacción entre usuario y programa deberá permitir el cumplimiento de cada una de las funciones de apoyo definidas para el MEC, por tipo de usuario. Su especificación conviene hacerla modular, por tipo de usuario y mediante refinamiento a pasos, de manera que haya niveles sucesivos de especificidad hasta que se llegue finalmente al detalle que hace operacional cada uno de los módulos que incluye el MEC.

La estructura lógica deberá ser la base para formular el programa principal y cada uno de los procedimientos que requiere el MEC. Finalmente, es necesario determinar de cuáles estructuras de datos es necesario disponer en memoria principal y cuáles en memoria secundaria (archivos de disco), de modo que el programa principal y los procedimientos de que se compone el MEC puedan cumplir con las funciones definidas.

Desarrollo de MEC

Desde la fase de análisis, cuando se formuló el plan para efectuar el desarrollo, debieron haberse asignado los recursos humanos temporales y computacionales necesarios para todas las demás fases. Tomando en cuenta esto, una vez que se dispone de un diseño debidamente documentado es posible llevar a cabo su implementación (desarrollarlo) en el tipo de computador seleccionado, usando herramientas de trabajo que permitan, a los recursos humanos asignados, cumplir con las metas en términos de tiempo y de calidad de MEC.

Prueba piloto de MEC

Con la prueba piloto se pretende ayudar a la depuración del MEC a partir de su utilización por una muestra representativa de los tipos destinatarios para los que se hizo y la consiguiente evaluación formativa. Para llevarla a cabo apropiadamente, se requiere preparación, administración y análisis de resultados en función de buscar evidencia para saber si el MEC está o no cumpliendo con la misión para la cual fue seleccionado o desarrollado.

Prueba de campo de MEC

La prueba de campo de un MEC es mucho más que usarlo con toda la población objeto. Sí exige hacerlo, pero no se limita a esto. En efecto, dentro del ciclo de desarrollo de un MEC es necesario buscar la oportunidad de comprobar, en la vida real, que aquello que en el ámbito experimental parecía tener sentido, lo sigue teniendo.

Metodología de *desarrollo de software educativo* propuesta por S. Gustavo Peláez Camarena y Bertha López Azamar

Dicha metodología consta de 13 pasos fundamentales [5], en los cuales se toman en consideración aspectos de *ingeniería de software, educación, didáctica y diseño gráfico*, entre otros. Es importante que el desarrollador del software educativo planifique su producto de software y considere las características planteadas en cada fase del desarrollo, ya que el objetivo mismo de la metodología es la creación de productos de software creativos, pero que vayan de la mano con los planteamientos de una materia, método didáctico y tipo de usuario específico. No todos los aprendizajes pueden, ni deben ser planteados de la misma manera, porque las capacidades de los usuarios varían según la edad, medio ambiente y propuesta educativa.

No está de más mencionar que los conocimientos generales de la ingeniería de software (IS) son la base principal sobre la cual se ubican las fases de la metodología y sus pasos respectivos, y que el ingeniero de software debe cumplir y aplicar los planteamientos generales del área de ingeniería del software.

A continuación se describen las fases de la metodología:

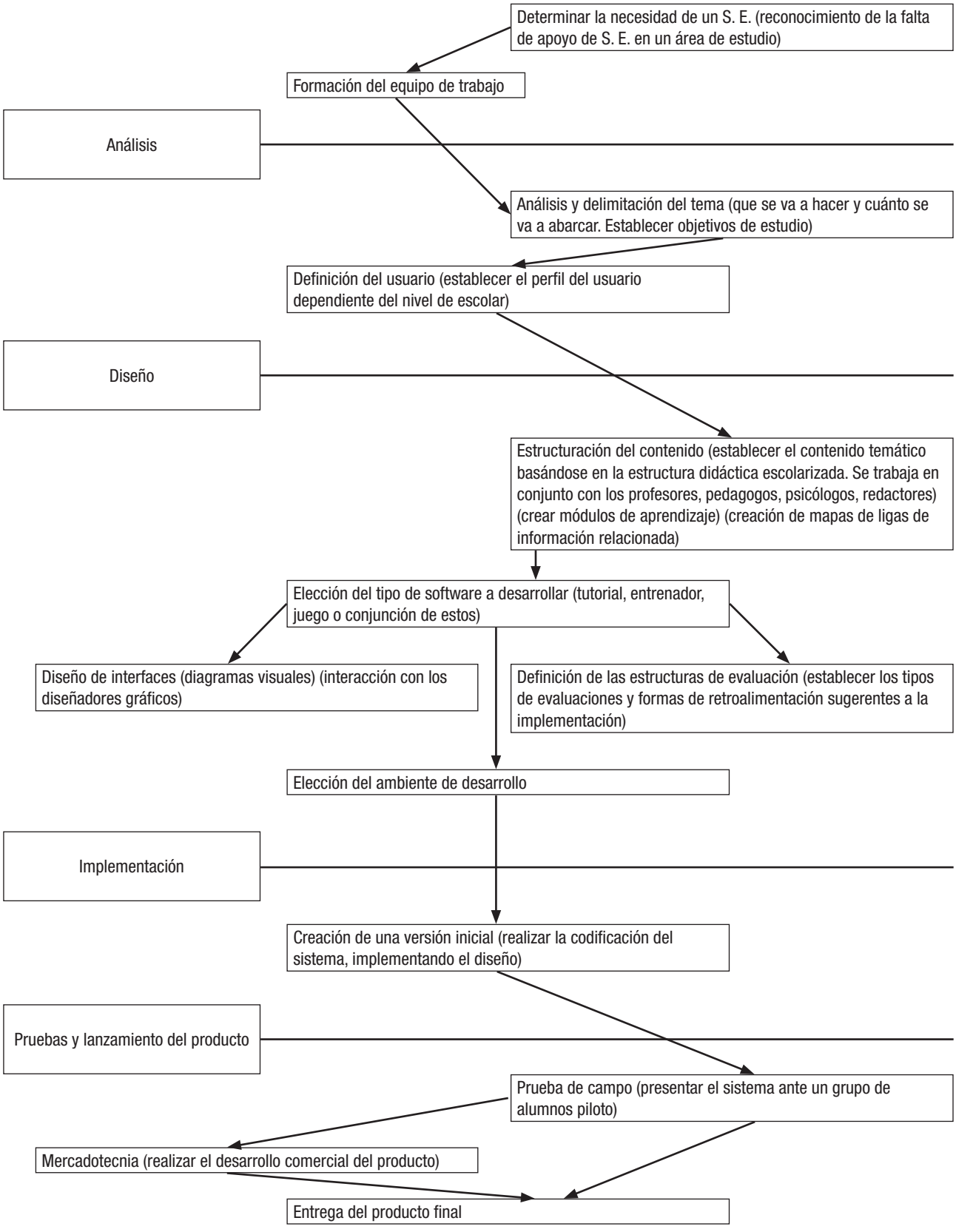

Figura 3. Esquema general de la metodología Desed

Fuente: S. Gustavo Peláez Camarena, Bertha López Azamar [5]

Determinar la necesidad de un software educativo (SE)

El SE deberá estar en capacidad de cubrir los aspectos primordiales del área o materia de estudios, según se trate. Desde esta perspectiva, se recomienda al ingeniero de software estar al tanto de la información de la asignatura y las técnicas didácticas que pudieran ser utilizadas al impartir normalmente la asignatura.

Formación del equipo de trabajo

De acuerdo con diversos autores, es necesario conformar un grupo de trabajo adecuado para poder desarrollar un SE completo; esto, debido a que lo más importante ya no es solo la información, sino que también debe tenerse en cuenta la forma de presentarla, que en últimas es el conocimiento que debe ser adquirido por los estudiantes.

Análisis y delimitación del tema

Aquí se define la amplitud del SE. Se analizan las necesidades presentadas por los usuarios que requieren el software, determinándose los objetivos particulares de trabajo, es decir, las necesidades deben permitir establecer el ámbito de la materia, y determinar los temas específicos de los planes de estudio que deben ser considerados para el desarrollo del producto; este último aspecto es de gran importancia ya que se debe delimitar la amplitud de los temas a cubrir.

Definición del usuario

Basados en la definición del nivel de enseñanza al cual va dirigido el software educativo, se deben determinar las características del usuario. Es importante definir con claridad al usuario final potencial del SE, ya que dentro de cada nivel de enseñanza la edad de los alumnos será determinante para la elección y aplicación de las técnicas o estrategias de enseñanza que se vayan a tener presentes en el desarrollo del software.

Estructuración del contenido

En esta parte de la metodología se deben definir los conceptos a considerar para establecer los contenidos temáticos que se abarcan en el SE. El trabajo conjunto entre el experto en el tema (profesores) y los pedagogos, psicólogos, redactores y editores de la información se lleva a cabo en este punto. El experto en el tema y los redactores definen el alcance de los contenidos temáticos específicos que deberán ser mostrados a los alumnos.

Elección del tipo de software a desarrollar

En el momento de elegir un tipo de software a desarrollar, es preciso tener presentes los niveles de complejidad de las áreas de aprendizaje. El software educativo puede ser visto como un recurso de enseñanza-aprendizaje, pero también, de acuerdo con una determinada estrategia de enseñanza, el uso de un determinado software puede llevar unas técnicas o métodos de aplicación implícitas o explícitas; ejercitación y práctica, simulación, tutorial; uso individual, competición, pequeño grupo, etcétera.

Diseño de interfaces

La interfaz es un punto focal, ya que a través de ella se lleva a cabo la comunicación entre el usuario y la computadora, y es lo que contribuirá a la motivación, eficiencia, comprensión y uso del SE que se desarrollará. Aquí es en donde se hacen realidad algunas de las especificaciones definidas hasta el momento, se toman en cuenta las consideraciones didácticas expuestas en la definición de necesidades. El desarrollador debe hacer en este punto prototipos de muestra de la interfaz elegida, para poderla mostrar al equipo de trabajo.

Definición de las estructuras de evaluación

La finalidad misma del SE es lograr que los alumnos aprendan los contenidos establecidos dentro de la planeación didáctica del curso. Al realizar el SE, deben proporcionarse a la par de los contenidos de aprendizaje las formas de evaluación de los contenidos mismos, para que con estas evaluaciones: el maestro pueda evaluar los aprendizajes, sugerir los repasos de los temas por parte de los alumnos, y los alumnos puedan retroalimentarse y reafirmar los conceptos aprendidos.

Elección del ambiente de desarrollo

Es importante que la delimitación del campo de aplicación del SE esté perfectamente definida, ya que cada desarrollador deberá buscar la herramienta que le permita involucrar todas las peticiones de los usuarios potenciales. Cada lenguaje de programación permite el desarrollo de uno u otro tipo de software. Asimismo, se puede explotar según sean las necesidades que el desarrollador tenga, razón por la cual se debe tener especial cuidado en la elección del ambiente de desarrollo.

Creación de una versión inicial

Una vez que se tiene la información requerida del índice temático, se ha elegido el ambiente de desarrollo y el tipo de software a realizar, se deben comenzar a planificar los aspectos de implementación y realizar la implementación en sí. Se deben respetar en todo momento los acuerdos a los que llegó el grupo de trabajo hasta el momento antes

de llegar a la implementación, y que debieron recopilarse a lo largo de cada etapa del proceso de desarrollo. La creatividad del ingeniero de software es la única limitante en su desarrollo.

Prueba de campo

La primera versión del sistema debe ser puesta a prueba frente al equipo de trabajo para su evaluación y rectificación de características; asimismo, para verificar que las especificaciones establecidas en el análisis y diseño fueron respetadas por el desarrollador. Una vez se detecten los posibles errores u omisiones, debe retomarse el desarrollo y volver a orientar la implementación del nuevo diseño de las modificaciones realizadas, creando una nueva versión del se.

Mercadotecnia

En el caso de que el se haya sido diseñado para comercializarlo, en este paso de la metodología debe realizarse un recuento de características de mercadotecnia que harán que el producto sea vendible. Debe elegirse un nombre, un empaque y el modo de distribución. La estrategia de mercado elegida es la que hará que nuestro software incursione y se presente ante los usuarios finales potenciales, para que pueda afianzarse un mercado.

Entrega del producto final

Debe presentarse un producto final a los usuarios potenciales, que debe tener el apoyo documentado en características de instalación y operación.

Conclusiones

De la revisión de las fuentes y la experimentación se puede concluir que el desarrollo de software educativo tiene en el Lenguaje Unificado de Modelado una herramienta fundamental de la que se puede rescatar, principalmente, su gran capacidad de vincular a los usuarios y desarrolladores en torno a lo que se espera de una determinada aplicación que se solicita, y lo que los entendidos en ingeniería del software deberán tomar en cuenta a través de los casos de uso para proponer alternativas a los requerimientos de sistemas con propósito educativo. Si bien esta notación nació con la metodología del Proceso Unificado (pu), es utilizable dentro de cualquier metodología o proceso de software, como es el caso de la metodología de *ingeniería de software educativo,* propuesta por Galvis Panqueva, y la metodología de *desarrollo de software educativo*, de S. Gustavo Peláez Camarena y Bertha López Azamar, ya que es independiente de la tecnología a desarrollar y, sobre todo, permite obtener modelos que representan los requerimientos del cliente, los cuales se traducen en diseños implementables bajo cualquier lenguaje de programación.

La notación gráfica del uml posibilita la comunicación y su aprendizaje por parte de los integrantes del equipo multidisciplinario de desarrollo ajenos al área computacional toma poco tiempo. Si cuando menos se logra involucrar a los expertos en el dominio y a los profesores de carrera en la identificación de casos de uso para desarrollar una aplicación educativa, se habrá logrado un avance significativo en el mejor desarrollo de software para la educación.

Por otra parte, la metodología aquí presentada, en la que el uso del uml es muy importante, servirá a los desarrolladores inexpertos y con conocimientos de algunos lenguajes de programación para realizar software educativos sencillos; y para los ingenieros de software que deseen coordinarse con un equipo de desarrollo de se, servirá para realizar una implementación más estructurada y sustanciosa, recordando que las limitaciones solo son impuestas por el propio desarrollador. Se plantean en dichos procesos aspectos característicos generales que sirven para organizar el ámbito del software, que se pretende realizar mediante un marco de trabajo, y se trata de llevar de la mano al desarrollador para definir su producto final, razón por la cual, al seguir los pasos planteados, se podrán tener presentes los aspectos que debe tener un se.

Referencias

[1] G. Booch. *Object-oriented analysis and design with applications.* Segunda Edición. Ciudad de México: Addison Wesley, 1994, pp. 320-326.

[2] A. Galvis Panqueva. *La ingeniería de software educativo*. Primera edición. Bogotá: Editorial Uniandes, 2001, p. 95.

[3] "El software educativo". P. Marqués. Universidad Autónoma de Barcelona. Consultado: 19 de febrero del 2001, disponible en: http://dewey.uab.es/pmarques/concepci.htm

[4] R. Pressman. *Ingeniería del software. Un enfoque práctico*. Sexta edición. McGraw-Hill. 2006, pp. 340-345.

[5] S. G. Peláez y B. López. *Metodología para el desarrollo de software educativo*. Segunda edición. México: Instituto Tecnológico de Orizaba, 2006, pp. 160-164.

[6] I. Jacobson, G. Booch y J. Rumbaugh. *El proceso unificado de desarrollo software*. Tercera edición. México: Addison Wesley. 1999, pp. 230-234.