PROYECTOS INFORMÁTICOS Y CIENCIAS DEL DEPORTE: UN CASO DE ESTUDIO DETECPOS

CARLOS RODOLFO TORRES SANCHEZ
Magister en Ciencias de la Educación
Profesor Universidad de Pamplona
Grupo de Investigación: Actividad física, Recreación y Deportes
ctorres@unipamplona.edu.co

HUMBERTO PARADA CARVAJAL
Magister en ciencias de la actividad física y del deporte
Profesor jubilado Universidad de Pamplona
hparada@hotmail.com

RESUMEN

En este artículo se muestran las formas de abordar el proceso de desarrollo de software aplicado al entorno del aprendizaje y sin importar el entorno específico del ámbito educativo;en el presente caso, aplicado a las ciencias del deporte. En él se exponen las herramientas de comunicación con el cliente bajo la forma de mapa conceptual, el seguimiento básico del proyecto y la facilidad de un método práctico de desarrollo adaptable al contexto de cada producto. En la conducción del diseño instruccional se adopta el modelo de Dick y Carey. Entre los múltiples enfoques para el desarrollo de software se seleccionan las fases del modelo en V que se adaptan al desarrollo de prototipos. Estos aspectos hacen más fácil la participación del docente en proyectos informáticos específicos.

Palabras claves: Software educativo, diseño instruccional, prototipado, tecnología aplicada, mapas conceptuales, CPM, PERT, desarrollo de software.

T PROJECTS AND SCIENCES OF THE SPORT: A CASE OF STUDY DETECPOS

ABSTRACT

This paper deals with the approaching forms related with software development process applied to the learning without any specific teaching environment. In this case it is related with applied sciences to the sports. Tools are shown with the client communication in a Conceptual Map way, the basics of the project and the easiness from a practical method of adaptive development to the context of each project. For the conduction of the instructional design the model of Dick and Carey is adopted. Among the multiple focuses for the software development the phases of the pattern are selected in V model that had been adapted to the development of prototypes. All these aspects make easier that teachers become involved in any software development project.

Key words: Educational Software, Instructional Design, Prototyping, Applied technology, Conceptual map, CPM, PERT, Software development

Artículo recibido 24 de febrero del 2012 y aceptado para su publicación el 18 de mayo del 2012.

Se considera un artículo T 1 de Investigación científica y tecnológica.

INTRODUCCIÓN

Referirse al equilibrio del ser humano remite siempre a la concepción global de las relaciones ser-mundo. Es por esta razón por la que se profundiza en su análisis tanto desde lo psicobiológico, lo psiconeurológico, o la anatomía y fisiología evolutivas, como se aborda también su estudio desde las llamadas ciencias del movimiento. Es importante señalar que esa misma globalidad obligaría a que en cualquier aproximación teórico-práctica sobre este tema se considerasen, por lo menos, algunos de los siguientes elementos:

(a) los datos de la filogénesis y la ontogénesis del ser humano;

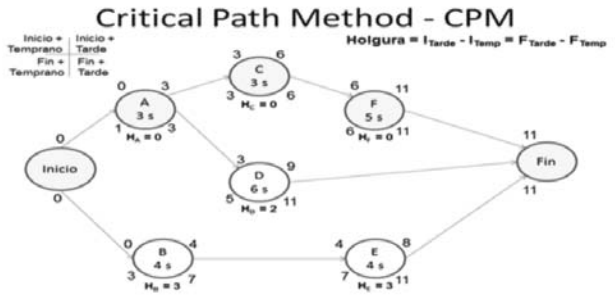


Fig. 1 *Modelo Hipotético de CPM con tiempos semanales*

(b) la explicitación de los mecanismos en virtud de los cuales los centros inferiores y superiores del sistema nervioso central regulan y modulan adaptativamente las reacciones equilibratorias;

(c) sus perturbaciones y la posible influencia de éstas sobre lo cognitivo, y

(d) no tendrían que olvidarse los aspectos relacionados con la vida socio-emocional.

No es la intención desarrollar exhaustivamente todos estos aspectos, muchos de los cuales están ya esbozados y planteados en otros trabajos (Alexander, 1996; Giese, 1998;

Fig. 2. *Detección del descentramiento del tronco. Desviación del diámetro biacromial*

Platonov, 1992; Ramón, 2004). Cada vez más se considera que el "equilibrio-postural-humano" es el resultado, antes que nada, de distintas integraciones sensorio-perceptivo-motrices que -al menos en una buena medida- conducen al aprendizaje en general, y al aprendizaje propio de la especie humana en particular, y que, a su vez, puede convertirse, si existen fallos, en obstáculo más o menos importante, más o menos significativo, para esos logros. Esta integración es susceptible de ser cuantificada mediante una herramienta que permita mediante el análisis de fotografías de manera no invasiva al sujeto.

La idea del presente trabajo, en primer lugar, es fijar la atención en el papel que desempeña la postura o descentramiento del tronco y su acción en relación con la capacidad de equilibrio de cualquier sujeto humano; en segundo término, encontrar una posible herramienta informática o de software, que ayude tanto al educador físico como al profesional de la salud a diagnosticar, tempranamente, mediante un método no invasivo, la condición básica de la columna y su incidencia en la postura corporal[1].

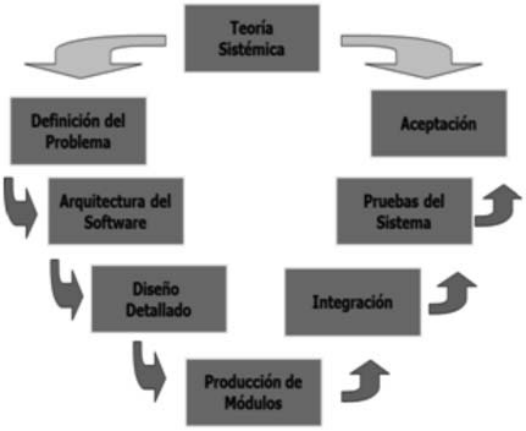
Ahora bien, una de las cosas más preocupantes al desarrollar aplicativos software, es la de llegar a una fecha límite y encontrarse con que los aspectos relevantes de un aplicativo aún no tienen completa funcionalidad. Esta preocupación, bastante frecuente entre quienes se dedican al desarrollo de software, se debe principalmente a que el software se crea, no se fabrica; es decir, se adopta el modelo de la realidad y sobre él se trabaja.

Al adoptar este punto de vista, trabajar sobre hechos y procesos reales, hace que se toque lo que los expertos han denominado los procesos críticos del negocio. Esto agrega, a la presión de la creación, un factor más de tensión, puesto que son procesos vitales para el correcto funcionamiento y éxito del proyecto en desarrollo. Desde un comienzo, al abordar el desarrollo de un proyecto en el equipo de trabajo, se busca crear un vínculo entre el equipo de trabajo y el "cliente-usuario" a través de un mapa conceptual utilizando el CMapTools como herramienta de trabajo[2], mediante esta herramienta se hacen explícitas las relaciones entre las funciones del aplicativo que se quiere desarrollar y las posibles restricciones o dificultades que se puedan encontrar en su implementación[3]. No está de más decir que con este enfoque, se busca aprovechar la experiencia pedagógica del o los integrantes del equipo, que hacen un buen uso de las tecnologías.

Una vez establecida la relación entre el cliente-usuario, a fin de minimizar el estrés del nuevo proyecto, se procede a establecer el camino más expedito para su desarrollo utilizando el CPM[4] (Critical Path Method) que permita visualizar las etapas del proceso de desarrollo y la holgura o disponibilidad de tiempo para llevar a cabo cada una de las etapas; así se establece la ruta más larga que involucra los aspectos críticos del proyecto y permitiendo saber de antemano la duración de cada etapa y por ende, la duración total del proyecto. Al utilizar esta técnica, se busca lograr tener un proyecto bien desarrollado al estar completo, a tiempo y ajustado al presupuesto

Otra técnica complementaria de evaluación de proyectos complejos con la que los ingenieros de software están familiarizados, se conoce como PERT[5] por su nombre en inglés (Program Evaluation and Review Technique) en la cual, cada etapa es analizada según su peso e importancia dentro del proyecto. A diferencia del método anterior, éste, permite involucrar la in-

certidumbre para calcular los tiempos de cada fase, teniendo en cuenta escenarios de duración normal, optimistas y pesimistas, para establecer sus duraciones. En los proyectos que involucran o tienen un carácter "educativo", es decir, aquellos que de alguna manera pretenden llevar o mostrar contenidos pedagógicos o de aprendizaje, se adoptan las fases generales de desarrollo sugeridas por los principios de la Ingeniería del Software[6] controlados mediante el método CPM. Al interior del equipo de trabajo se selecciona un modelo general de desarrollo de software; en este caso se utilizará el denominado modelo en "uve", por ilustrar no solo las partes generales del desarrollo, sino porque su configuración representa con mayor claridad la cualidad de cada fase, bien sea ésta de diseño u operativa.



Conjuntamente con la etapa de diseño computacional, se hace la conducción del diseño instruccional. Se debe definir qué tipo de aplicativo se va a desarrollar, según éste aborde una perspectiva instruccional, activa o interactiva, así como el tipo de recursos que empleará. Para ello se ha adoptado el modelo propuesto por Walter Dick y Lou Carey. Información general sobre el modelo se puede encontrar en diferentes sitios de la web.

1. EL DIRECTOR DEL EQUIPO

El proceso de desarrollo clásico de un software pasa por diferentes etapas. Cada una tiene sus propias características particulares.

Tal como se describe en Pressman (2002), este proceso es posible aplicarlo en cualquier situación o tipo de desarrollo, siempre y cuando se hayan definido un conjunto específico de pasos procedimentales. A partir de aquí, se determina la naturaleza de un aplicativo software[9].

Las etapas generales que se encontrarán en todo proyecto clásico son: análisis, diseño, implementación y prueba. Dependiendo principalmente de las necesidades de cada proyecto, cada etapa podrá ser subdividida en subtareas más específicas. Para el caso de un proyecto informático de carácter educativo, la fase de diseño deberá incluir el diseño instruccional. En esta etapa se tendrá en cuenta el objetivo instruccional y las habilidades que prentende desarrollar. Un modelo que permite un trabajo seguro en esta etapa del diseño es el de Dick y Carey (1990). En términos generales el modelo se puede presentar como se muestra en la siguiente gráfica.

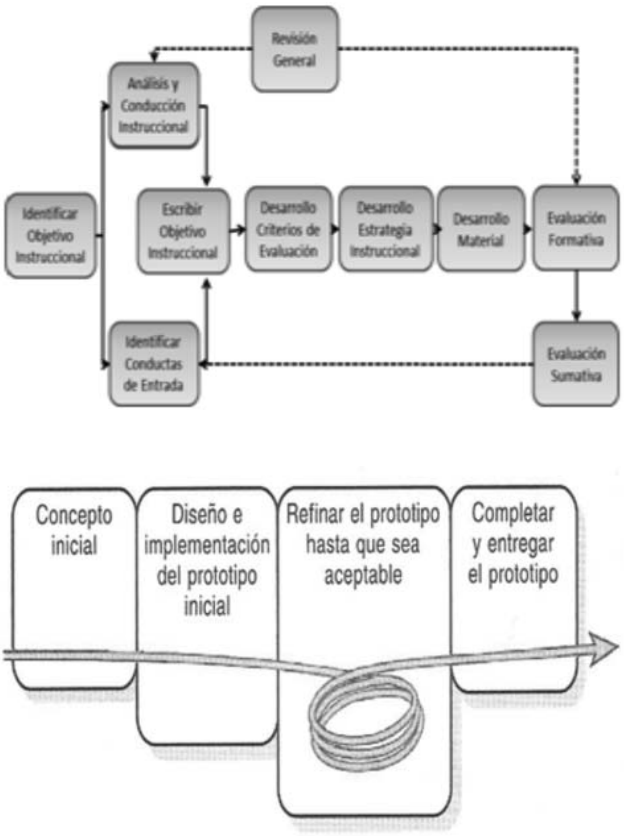


Fig. 3 *Esquema general de conducción del diseño instruccional*

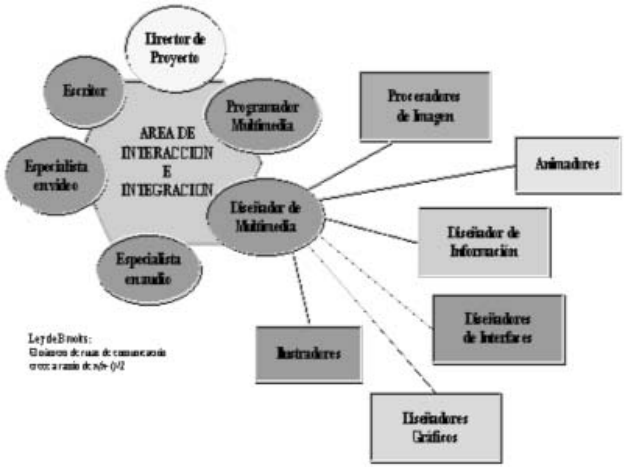


Fig. 4 *Modelo General del Prototipo Evolutivo. Tomado de McConnel, Steve (1997) p. 160*

Los principios generales de ingeniería del software[10] establecen un ciclo básico que es adaptable al contexto. En palabras de McConnel (1997) sobre la selección de un ciclo de vida rápido, "no existe un modelo de ciclo de vida de desarrollo rápido, debido a que el modelo más efectivo depende del contexto en

el que se utilice"[11]. Entre los múltiples enfoques[12] o métodos de desarrollo[13]; se ha adoptado, con modificaciones, el sugerido por Pressman (2002), denominado en general Modelo Adaptable de Desarrollo ó MAD. En el proceso, se genera un documento de seguimiento y especificaciones, que consigna las características principales del producto y el proyecto por desarrollar. El inicio del proyecto lo marca la creación del mapa conceptual con el cliente usuario. Para ello se puede utilizar la herramienta CMap Tools.

2. SELECCIÓN DEL MODELO DE DESARROLLO

2.1 Modelo Adaptable de Desarrollo - MAD

En términos generales para el desarrollo de los diferentes aplicativos en que se ha trabajado se adoptó el modelo de desarrollo evolutivo de prototipo[16], dadas las características de los proyectos y de los integrantes del equipo de trabajo.

El Modelo adaptado MAD incluye en sus etapas un proceso de prototipo evolutivo, que es un modelo de ciclo de vida en el que se desarrolla el concepto del sistema a medida que avanza el proyecto. Normalmente se comienza desarrollando los aspectos más visibles del sistema.

Una vez presentadas las partes importantes del sistema al cliente, se continúa y refina el desarrollo del prototipo basándose en la realimentación que recibe.

Una vez se considere que el producto es "aceptable", se completa cualquier trabajo pendiente en el sistema y se entrega el prototipo como "producto final". La Figura 3 describe este proceso iterativo gráficamente.

El prototipo evolutivo se utiliza especialmente cuando los requerimientos cambian con rapidez, cuando el cliente es reacio a especificar el conjunto de los requerimientos, o cuando ni usted ni el cliente identifican de forma apropiada el área de aplicación. También es útil cuando los desarrolladores no están seguros de la arquitectura o los algoritmos adecuados a utilizar.

El principal inconveniente de este tipo de prototipo es la imposibilidad de conocer al comienzo del proyecto lo que se tardará en crear un producto aceptable. Incluso no se sabe cuántas iteraciones se tendrán que realizar. Esta aproximación puede convertirse fácilmente en una excusa para realizar el desarrollo con el modelo de codificar y corregir.

Un prototipo evolutivo incluye análisis de requerimientos, diseño y codificación, pensado para el mantenimiento real en niveles ligeramente inferiores de los que se utilizan con las aproximaciones tradicionales. Esta es una de las razones por la cual se hace válida la adopción de este modelo; se trabaja sobre condiciones reales.

2.2 Proceso General de Desarrollo

*** Análisis general:** el líder del equipo de trabajo, en condiciones ideales, debe ser el especialista en la materia sobre la que se hace el desarrollo. Éste, es asesorado por los especialistas en ingeniería del software. En el proceso se establecen las preguntas y las razones o respuestas a esas preguntas por las cuales, sobre el concepto inicial, se debe buscar una solución software, u otras alternativas a las necesidades planteadas[17].

La cantidad de integrantes podrá variar, de acuerdo a la naturaleza del proyecto. De acuerdo con el modelo, se debe tener especial cuidado al proceso de comunicación entre las partes, porque los canales crecen según la Ley de Brooks[19]. Este aspecto comunicacional se convierte en crucial, sobre todo en las fases iniciales de análisis y diseño, porque en ella recae la mayor parte del peso del proyecto.

Se determinan los objetivos y alcances reales del aplicativo y el proceso se adapta a los requerimientos formulados. Se hacen explícitas las relaciones de las funciones del aplicativo. De ser necesario, las relaciones halladas o establecidas dentro de los mapas conceptuales se representan en forma de Diagramas de descomposición Funcional o DDF para el equipo de desarrollo. Se establecen las necesidades humanas que conformarán el equipo de trabajo y sus relaciones de dependencia. El Diseño e implementación inicial del prototipo se desagrega en subproductos así:

* Diseño de datos: en caso de manejar datos, se establece el diseño de éstos y las maneras de generar los diferentes reportes basados en esos datos. Un alto porcentaje de aplicaciones software utiliza las bases de datos como elemento principal de trabajo. Su diseño depende de los requisitos del solicitante, cliente o usuario y las necesidades por suplir. Es posible que esta etapa sea breve y de paso rápidamente a la siguiente fase, en la que se determinará la estructura, funciones a que serán sometidos esos datos.

* Diseño arquitectónico: se especifican las funciones generales que integrarán el software, según las necesidades especificadas por el especialista. Se buscan las prestaciones, tanto para docentes como para estudiantes que pueda tener el aplicativo. Desde esta etapa se deben prever las posibles restricciones que tenga el producto así como el grado de acceso a cada una de las funciones o prestaciones establecidas en su funcionalidad.

* Diseño de interfaz de usuario: el equipo de trabajo interviene para aportar ideas e iniciativas sobre la GUI[20]. En esta etapa se aclara como será la interacción del usuario con el aplicativo. Enun nivel más detallado se establece también la manera como el software se comunica consigo mismo y con el usuario. Se busca, entonces, el apoyo de diseñadores gráficos o de multimedia y de los documentos de apoyo apropiados, que permitan establecer con claridad la metáfora que se empleará en el diseño del ámbito del aplicativo o micromundo en el que se desarrollará el trabajo.

* Refinamiento paso a paso, se avanza de acuerdo a la retroalimentación del usuario y al criterio de los especialistas del grupo. El proceso se sigue hasta considerar el producto como "aceptable" y se hace la entrega del prototipo.

Esta metodología busca asegurar la calidad del producto y evitar la excesiva formalidad que presentan otros modelos de trabajo haciendo más fácil el acercamiento de los docentes e infor-

máticos dentro del proyecto. El centrarse sobre los puntos antes mencionados, permite mantener un proceso ágil de desarrollo rápido de aplicativos a fin de cumplir las metas, expectativas y plazos de tiempo en las pruebas de cada aplicación; para el caso específico del trabajo Universitario, se hace dentro del semestre lectivo. En Bezier (1990) [21]se ofrece una visión bastante completa de las pruebas de software que enfatiza los modelos formales para la prueba.El autor da una visión general del proceso de pruebas y las razones y metas para las mismas.Se muestra también cómo implementar las pruebas de software basado en las estrategias.

De acuerdo a Boehm (1996) y el principio de las Preguntas Básicas (W^5HH)[22] "[…] se necesita un principio de organización que haga una simplificación con el fin de proporcionar planes de proyectos sencillos para proyectos pequeños." El principio W^5HH de Boehm es aplicable sin tener en cuenta el tamaño o la complejidad del proyecto de software.

El objetivo detrás de las preguntas señaladas es proporcionan un perfil de planificación al gestor del proyecto y al equipo de software. La unión de varios métodos efectivos de desarrollo sugeridos por MacConnell (1997) (tanto en planificación como en desarrollo) permiten hacer un incremento real en la productividad de un proyecto software. De esta manera se evita adoptar enfoque de desarrollo un tanto más complejo, al no considerar métricas de complejidad o de tamaño dentro del desarrollo de los aplicativos. Solo se toman en consideración los diseños de las diferentes interfaces de usuario a fin de hacerlo lo más amigable posible sin sacrificar la funcionalidad del producto.

3. HOJA DE RUTA DEL PROYECTO

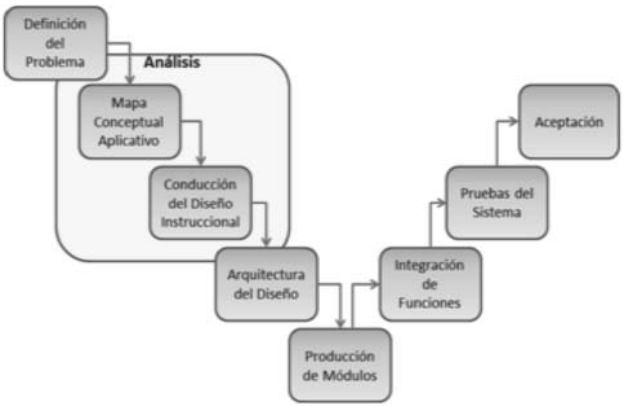


Fig. 5. *Ejemplo de Mapa conceptual para el proyecto DetecPos*

La ruta principal de trabajo está determinada por los principios básicos de la Ingeniería del software. Independientemente del modelo adoptado, las etapas generales son las ya mencionadas anteriormente en Pressman 2002).

De esta manera el avance del proyecto se puede realizar cuidando los pasos generales propuestos para el mismo. Dentro del análisis, el docente puede elicitar los requerimientos del nuevo proyecto y recurre a una forma más "blanda" de trabajo. Esta forma blanda se refiere al uso de los mapas conceptuales (MC) como herramienta de dentro de la fase de análisis. Ver Fig. 5

El MC responde a la pregunta básica de trabajo propuesta por el director del proyecto; para el presente caso se trata de establecer una respuesta a ¿es posible diseñar una herramienta de diagnóstico temprano de las desviaciones de columna, no invasiva y de fácil manejo? Dentro del proceso de creación del mapa conceptual se debe poner especial cuidado a las relaciones de los conceptos, toda vez que ellas son parte de la respuesta al interrogante principal. A continuación, se presenta un modelo de mapa conceptual aplicado al proceso de desarrollo de software; en este caso particular se utilizó para ayudar a un docente en la creación de un aplicativo para "DETECtor de POStura corporal". En él se puede apreciar el grado de claridad del docente a la hora de avocarse al desarrollo de una solución informática a su situación particular.

Una vez establecidas las principales funciones del nuevo programa se sigue con el proceso general de desarrollo del software. Se aclaran las dependencias de las funciones y su prioridad. Se establece la arquitectura del aplicativo y las prestaciones importantes tanto para docentes como estudiantes, así también se establecen las jerarquías de acceso a cada una de ellas.

A partir del mapa conceptual se puede ilustrar de manera más concreta en un diagrama de funciones, la dependencia de éstas, que harán más evidente la arquitectura del sistema a desarrollar. El diagrama de descomposición funcional para el proyecto DETECPOS se ilustra en la Fig. 6.

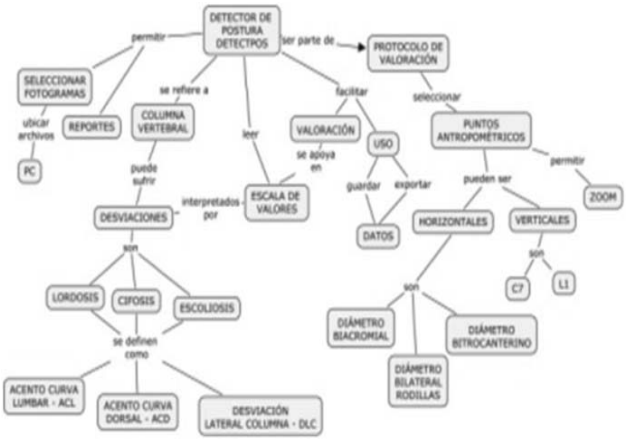


Fig. 6 *Diagrama de Descomposición Funcional Proyecto DETECPOS*

Este proceso garantiza que una vez se inicie la etapa de codificación o de creación de módulos como aparece en la Fig. 6, se pueda proceder con mayor seguridad y garantía de tener cubiertos los aspectos relevantes del software tal como el usuario final lo desea.

El proceso iterativo, tal como se ilustra al inicio del documento, ver Fig. 4. , incluye las pruebas del sistema. Cada presentación del desarrollo hace que el prototipo madure hasta encontrar el grado de "aceptación" convenido entre las partes interesadas participantes en el proyecto.

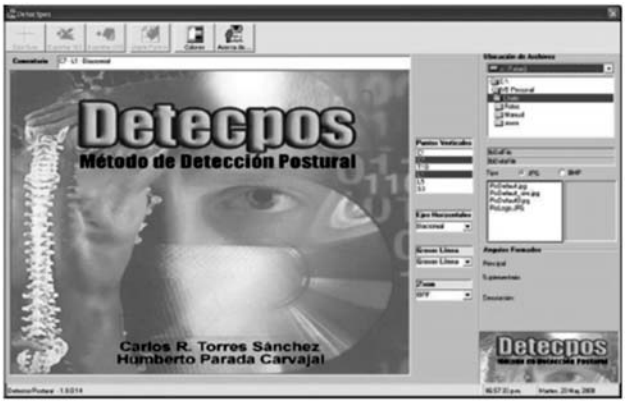
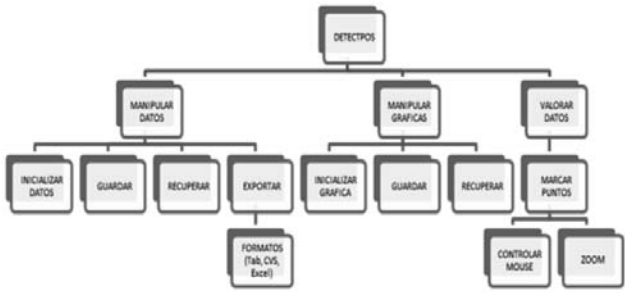


Fig. 7. *Pantalla Principal de DetecPos*



Una vez alcanzado el nivel de aceptación, se surten las etapas finales del proceso de desarrollo, tanto para usuario final como para el equipo de trabajo. Para usuario se establecerá el proceso de seguimiento y mantenimiento del sistema desarrollado, y para el equipo se concluirá con el informe "post mortem" del proyect,o de modo que sirva de guía para el desarrollo de nuevos proyectos, ya que ayuda a establecer parámetros más confiables sobre los estimados básicos de tiempo, completitud y presupuesto.

CONCLUSIONES

- Al establecer un mapa conceptual como herramienta de comunicación con el cliente-usuario, mejora la comprensión y el alcance de un proyecto.
- La interdisciplinariedad permite reconocer las zonas límite entre las disciplinas como fuente de potenciales investigaciones.
- La interdisciplinariedad es fundamental al momento de hacer desarrollo de software educativo.
- Contar con profesionales del área en que se desarrolla el proyecto, representa una ventaja enorme en el proceso de comprensión del problema y el establecimiento de los requerimientos del aplicativo de apoyo.
- Es posible desarrollar herramientas específicas que respondan a necesidades igualmente puntuales dentro del ámbito educativo.
- Las funciones de una herramienta o aplicativo software está determinado por la visión y necesidades de quienes integran el equipo de trabajo interdisciplinario.
- La conducción de Diseño Instruccional, definida por uno de los múltiples modelos existentes, es fundamental a la hora de presentar contenidos. Este marca un derrotero seguro y mensurable.
- El MAD es una vía apropiada para el desarrollo de prototipos que se acomodan a las restricciones de tiempo del ciclo educativo marcado por los períodos de actividad y recesos académicos.
- Con la asesoría del experto en contenidos, que, en condiciones ideales es el profesor de una asignatura determinada, se pueden crear funciones puntuales que se adapten a requerimientos específicos del aula de clase.
- El modelo MAD es lo suficientemente flexible y puede adoptarse en condiciones restringidas de tiempo, requerimientos y recursos.
- La mezcla de recursos y lenguajes de programación, tanto de lenguajes de autor como de alto nivel, ayuda a dar una respuesta más adecuada a los requerimientos y necesidades tanto de usuario como de método.

BIBLIOGRAFÍA GENERAL

1.ALESSI, S.M. and TROLLIP, S,R. (1985) Computer–Based Instruction methods and development. Englewood Cliffs, NJ. Prentice Hall.

ALEXANDER P. (1996) Aptitud física, características morfológicas y composición corporal Pruebas estandarizadas en Venezuela. Caracas: Depoaction.

AYRES, Jean A. (2003) The Effects of a Sensory Motor Activities Protocol Based on the Theory of Sensory Integration on Children Diagnosed with Preprimary Impairments. Volume: 17 Issue: 2, ISSN: 0738-0577 Pub Date: 3/1/2003

BALLESTEROS, J. M. (1993) Manual de entrenamiento básico de atletismo. Inglaterra: I.A.A.F..

BERXOSHANSKI, V. I. (1985) La programación de las cargas de entrenamiento. Moscú:.

BETANCUR, José Luis. (2004) "Modulo de Actualización en Entrenamiento Deportivo", Pamplona: Universidad de Pamplona.

BUILINA A. y N., Kuranshina. (1981) Fundamentos de la preparación de los jóvenes deportistas. Moscú:.

BELL, Doug, MORREY lan, PUGH John. (1992) Software Engineereing. A programming approach. New York: Prentice Hall

BEIZER, B. (1990) Software testing Techniques. Van Nostrand Rdnhold,

BEDNAR, Anne, [et all.] (1993) Instructional Message Design: Principles from the Behavioral and Cognitive Sciences. Educational Technology Publicacions: Englewood Cliffs NJ.

BOEHM, Barry. (1996) Anchoring the Software Process. IEEE Software, Vol 13. No. 4 Julio 1996, pp 73-82

BELL, Doug, MORREY lan, PUGH John. (1992) Software Engineereing. A programming approach. New York: Prentice Hall.

CALDERÓN, C. y Colectivo. (1993) Fundamentos generales de la teoría y metodología de la educación física. La Habana: Pueblo y Educación.

COAD, Peter and YOURDON, Edward. (1991) Object-Oriented Analisys. New Jersey: Prentice Hall.

COAD, Peter and YOURDON, Edward. (1991) Object-Oriented Design. New Jersey: Prentice Hall.

COX, Kevin and WALKER, David. (1993) User–Interface Design. New York: Prentice Hall

DONSKOI, D. D. (1982) Biomecánica con fundamentos de la técnica deportiva. Ciudad de la Habana-Cuba: Pueblo y Educación.

DICK, Walter and CAREY, Lou. (1990), The Systematic Design of Instruction, Third Edition, Harper Collins

EHLENZ, GROSSER y ZIMMERMAN. (1991) Los principios, la fuerza y la planificación del entrenamiento deportivo. México: Ediciones Roca S.A..

FLEMING, Malcolm and LEVIE, W. Howard. (1978) Instructional Message Design: principles from the behavioral and cognitive science. Educational Technology Publications: Englewood Cliffs, NJ.

FLEITAS I., y colectivo. (1990) Teoría y práctica general de la gimnasia. La Habana: Enpes.

FOMIN N.A y V. P. FILIN. (s.f.) En el camino hacia la maestría deportiva. Moscú, Moscú.

FORTEZA A. y A. RANZOLA. (1988) Bases metodológicas del entrenamiento deportivo. La Habana: Científico-Técnica,.

GALVIS, Panqueva Alvaro. (1989) Ingeniería del Software Educativo. Bogotá: Universidad de los Andes

GOWIN, Bob y NOVAK, Joseph. (1989) Aprendiendo a Aprender. Martinez Roca, Madrid

GROSSER, Manfred. (1980) "Principios del Entrenamiento Deportivo". Barcelona: Ediciones Martínez Roca S.A.,.

GIESE C. ARTHUR. (1968) Fisiología General, Estructura y dinámica Celular. Tercera Edición. México: Interamericana S.A.

HARRE, D. (1973) Teoría del entrenamiento deportivo. La Habana: Científico-Técnica.

IEEE. Standards collection: Software Engineering, IEEE Standard 610.12-1990, IEEE, 1993.

KOTSA, A. M. (1986) Fisiología deportiva. Moscú: Fisicultura y Deportes,.

KUZNETSOV, V.I (1973) La preparación de fuerza para deportistas de alto rendimiento. Moscú: Moscú.

LLINÁS, Rodolfo. (1999) El cerebro y el mito del yo. Bogotá. Norma. ISBN: 958-04-6798-6, 2003.

MATVEEV, L. Periodización del entrenamiento deportivo. Moscú: Raduga, 1977.

_____ y NOVIKOV. (1977) Fundamentos Generales de la Teoría y Metodología de la Educación Física. Moscú: Raduga.

_____. (1983) Fundamentos del entrenamiento deportivo. Moscú: Raduga.

_____. (1983) Fundamentos de la preparación de los jóvenes deportistas. Moscú: Moscú.

MENSHIKOU, V.U. y N. l., Volkov. (1991) Bioquímica de los ejercicios físicos. Moscú: Moscú.

MICROSOFT PRESS (2000) Diseño de Interfaz de Usuario para aplicaciones Windows. McGraw-Hill / Interamericana de España.

McCONNELL, Steve. (1997) Desarrollo y Gestión de Proyectos Informáticos. Madrid: McGraw–Hill

NAVATNIKOVA, M. I. (1982) Fundamentos de la preparación de los jóvenes deportistas. Moscú: Moscú,.

NIEVERGELT, Jay; VENTURA Andrea y HINTERBERGER Hans. (1986) Interactive computer programs for education. Philosophy, Techniques and Examples. Canada: Addison-Wesley

OZOLIN, N. G. (1973) El entrenamiento deportivo contemporáneo. Moscú: Moscú.

PARNIX, V. I. (1981) Metodología de la preparación de los juegos deportivos. Moscú: Fisicultura y Deportes,.

PETROBAKI, B. B. y Colectivo. (1985) Adaptación de los deportistas a las cargas de entrenamiento y competitivas. Moscú: Raduga.

PLATONOV, B. P. (1987) El entrenamiento deportivo. Barcelona-España: Paidotribo.

PLATONOV, B. P. (1992) La adaptación en el deporte. Barcelona-España: Paidotribo.

PRESSMAN, Roger (2002). Ingeniería del Software. Un enfoque Práctico. 5ª Ed. Mc Graw-Hill/ Interamericana de España. pp 16-50

RANZOLA, A. (1989) La planificación del entrenamiento deportivo. Caracas: Claced.

RANZOLA, A. (1989) La preparación competitiva. La Habana: Ediciones Inder.

RUIZ, A. y Colectivo. (1985) Metodología de la enseñanza de la educación física. Tomos I y II. La Habana: Pueblo y Educación.

RUIZ, A. y Colectivo. (1989) Gimnasia básica. La Habana: Pueblo y Educación.

RAMÓN SUAREZ, Gustavo. (2004) "Modulo Respuesta y Adaptación Fisiológica al Ejercicio" Pamplona: Universidad de Pamplona.

ROSSI, Peter H. y BIDDLE, Bruce J. (compiladores). (1970) Los nuevos medios de comunicación en la enseñanza moderna. Buenos Aires: Raidos. 455 p.

SHNEIDERMAN, Ben. (1987) Designing the User Interface: Strategies for Effective Human-Computer Inteaction. Addison-Wesley Publishing Company: Massachusetts,.

TAPSCOT, Don. (1998) Creciendo en un entorno Digital: La generación Net. Bogotá: McGraw-Hill

TORRES SÁNCHEZ, Carlos R. (1996) Introducción a la producción Multimedia. Apuntes. Universidad de Pamplona: Pamplona (Inédito)

TORRES SÁNCHEZ, Carlos R. (1998) Fundamentos de la Ingeniería del Software Educativo. Universidad de Pamplona: Pamplona. (Inédito).

TORTORA, Gerard y GRABOWSKI, Sandra. (2000) Principios de Anatomía y Fisiología. México: Servicios Editores Gráficos.

VOLKOV, V. M. y V. P., Filin. (1989) Selección deportiva. Moscú: Fisicultura y Deportes.

WEINECK, J. (1988) Entrenamiento óptimo. Barcelona: Hispano Europea.

ZATSIORSKI, V. (1988) "Biomecánica de los Ejercicios Físicos " Ciudad de la Habana, Cuba: Ráduga.

ZIMKIN, N. (1975) Fisiología del ejercicio. Moscú: Moscú.

ZINTL, F. (1991) Entrenamiento de la resistencia. México: Ediciones Roca S.A.