Sistemas Tutores Inteligentes: Procedimientos, métodos, técnicas y herramientas para su creación

Zulma Cataldi, Fernando J. Lage,
zcataldi@posgrado.frba.utn.edu.ar, liema@fi.uba.ar, flage@fi.uba.ar

Facultad Regional Buenos Aires. Universidad Tecnológica Nacional
Facultad de Ingeniería. Universidad de Bs. As.

Resumen

En esta comunicación se analiza, sistematiza e integra el conocimiento existente sobre Sistemas Tutores Inteligentes (STI) orientados a la enseñanza de la programación. Se busca detectar las problemáticas que se presentan a la hora de pensar en un nuevo STI y destacar las perspectivas y tendencias a la luz de las posibilidades actuales de los desarrollo en áreas como la ingeniería de software y los sistemas inteligentes, entre otras disciplinas que le dan sustento a los desarrollos.

Palabras clave: *Sistemas Tutores Inteligentes*

Abstract

In this communication the knowledge on Intelligent Tutoring Systems oriented to programming is analyzed and systematized (ITS). First, one look for to detect the problems the time of thinking about a new ITS and emphasizing the perspective and tendencies to the light of the present possibilities. The tools for the developments are taken for areas like software engineering and intelligent systems, among other disciplines that give theoretical and practical sustenance.

Keywords: *Intelligent Tutoring System*

1. Introducción

Los STI (Sistemas Tutores Inteligentes) permiten la emulación de un tutor humano para determinar qué enseñar, cómo enseñar y a quién enseñar a través de un módulo del dominio: que define el dominio del conocimiento, un módulo del estudiante: que es capaz de definir el conocimiento del estudiante en cada momento, un módulo del tutor: que genera las

interacciones de aprendizaje y finalmente la interface con el usuario: que permite la interacción del estudiante.

A través de la interacción entre los módulos básicos, los STI son capaces de juzgar lo que sabe el estudiante y cómo va en su progreso, por lo que la enseñanza, se puede ajustar según las necesidades del estudiante, sin la presencia de un tutor humano. Por otra parte, *el problema se centra entonces en que cada estudiante debería poder elegir el método de enseñanza del tutor de acuerdo a sus preferencias* [1], sea: instruccional, orientador, socrático, otros [2,3,4], y si lo deseara debería poder cambiarlo de acuerdo a sus propios requerimientos. El estudio que se presenta se inició para el caso de la programación básica debido a las necesidades de los estudiantes de contar con tutores que los asistan al avanzar en sus estudios. Se busca proveer de una alternativa al tutor humano, cuando no puede invertir más tiempo con sus estudiantes y para los estudiantes que buscan aprender en forma más autónoma [1].

Los *objetivos* de esta comunicación son a) Sistematizar la evolución y el panorama actual de los STI orientados a la programación, b) Evidenciar algunas de las debilidades a superar en los desarrollos actuales y c) Plantear las necesidades que aún no han sido cubiertas a fin de orientar las investigaciones en el tema.

2. La evolución de los STI

Los Sistemas Tutores Inteligentes (STI) comenzaron a desarrollarse en los años 80 con la idea de impartir conocimiento con base en alguna forma de inteligencia para guiar al estudiante en el proceso de enseñanza con un comportamiento similar a un tutor humano. Lo ideal sería que se adapte al proceder del estudiante, identificando la forma en que éste resuelve un problema y de brindarle ayuda cuando cometa errores. Un tutor inteligente, por lo tanto: *"es un sistema de software que utiliza técnicas de inteligencia artificial (IA) para representar el conocimiento e interactúa con los estudiantes para enseñárselo"* [5].

Wolf [6] define los STI como: *"sistemas que modelan la enseñanza, el aprendizaje, la comunicación y el dominio del conocimiento del especialista y el entendimiento del estudiante sobre ese dominio".* Giraffa [7] los define como: *"un sistema que incorpora técnicas de IA (Inteligencia Artificial) a fin de crear un ambiente que considere los diversos estilos cognitivos de los alumnos que utilizan el programa".*Entre los STI desarrollados orientado a la programación se pueden destacar: *Scholar* [8], *Why* [9], *Sophie* [10], *Guidon* [11], *West* [12], *Buggy* [13], *Debuggy* [14] *Steamer* [9], *Meno* [15], *Proust* [16], *Sierra*[5].

Así, los STI permiten la emulación de un tutor humano en el sentido de *saber que enseñar, cómo enseñar y a quién enseñar.* La mayoría de los sistemas implementados poseen una arquitectura básica común de tres *módulos básicos: 1) Un módulo del dominio*: que define el dominio del conocimiento (conocimiento sobre *qué enseñar*), 2) *Un módulo del estudiante*: que es capaz de definir el conocimiento del estudiante en cada punto durante la sesión de trabajo (conocimiento sobre *a quién enseñar*) y 3) *Un módulo del tutor*: que genera las interacciones de aprendizaje basada en las discrepancias entre el especialista y el estudiante (conocimiento sobre *cómo enseñar*) y finalmente *la interface*[1] con el usuario: que permite la interacción del estudiante con un STI de una manera eficiente (conocimiento sobre *cómo presentar* el material).

En los 90, los avances de la psicología cognitiva, las neurociencias y los nuevos paradigmas de programación, han hecho evolucionar a STI desde una propuesta instructiva [17] hacia entornos de descubrimiento y experimentación del nuevo conocimiento [18,3,19,20]. Las dificultades de representación actuales se centran en la identificación de los preconceptos o concepciones erróneas y en los diferentes estadíos evolutivos del estudiante [21]. A través de la interacción entre los módulos básicos, los STI son capaces de juzgar qué sabe el estudiante y cómo va en su progreso, por lo que la instrucción, puede ser ajustada según las necesidades del estudiante, sin la necesidad de un tutor humano. Un STI actúa como un tutor particular del estudiante ya que como un entrenador humano, posee libertad para actuar sobre las necesidades más complejas del estudiante.

Los STI aún no proveen de un modo de aprendizaje adaptable [22] de acuerdo a los conocimientos previos y a la capacidad de evolución de cada estudiante *y las concepciones epistemológicas que subyacen en las prácticas de enseñanza* [23,24]. En este contexto surgen *Andes* [25, 26] en el Pittsburgh Science of Learning Center's LearnLab, que su consorcio con miembros de Carnegie Mellon University, University of Pittsburgh y Carnegie Learning. *Metutor* es un tutor de medios-fines del Department of Computer Science, U.S. Naval Postgraduate School, Monterey [27,28]. *ITSpoke* es un Proyecto que usa un sistema de diálogos basado en textos y medios fines [29] que se desarrolló en la University of Pittsburg, Department of Computer Science & Learning Research and Development Center Pittsburgh. El STI *CircSim*, fue desarrollado en conjunto por el Departamento de Ciencias de la Computación del Illinois Institute of Technology y el Departamento de Fisiología del Rush College of Medicine. Este tutor es el más avanzado actualmente en su tipo, y se lo utiliza en el Rush

[1] Se siguen los principios del diseño, implementación y evaluación de Sistemas Computacionales Interactivos para su utilización por seres humanos (HCI: Human Computer Interaction), es decir que estudian y tratan de poner en práctica procesos orientados a la construcción de interfaces lo más usables posible, es decir con alto grado de facilidad en el uso del sistema interactivo. (Estándar ISO 92401 de requisitos ergonómicos para el trabajo de oficina con terminales visuales).

College of Medicine para complementar las clases teóricas sobre problemas cardiovasculares [30,31,32,33,34,35,36]. *AutoTutor*: es un STI basado en la web por un grupo interdisciplinario de la Office of Naval Research and the National Science Foundation [37,38,39,40, 41].

El Computer Tutoring Group (ICTG), que trabaja en el University's Computer Science and Software Engineering Department, en la University of Canterbury ha desarrollado una serie de STI: *Aspire* [42], *Sql-Tutor* es un sistema de enseñanza basado en el conocimiento que enseña SQL a los estudiantes [43,44] y *Kermit; EER-Tutor*, *ERM-Tutor* y *Normit* están orientados a la Informática.

3. Algunas debilidades en los STI

Se ha observado que los STI desde la concepción trimodular de Carbonell [8] (módulos de estudiante, tutor y dominio) deben presentar cada una de las funcionalidades de los módulos bien definidas para obtener sistemas que pudieran ser reutilizados desde una visión multidisciplinaria y multingüística. Además debería estar centrado en las necesidades reales de los estudiantes. Otro punto débil detectado en el modelo más difundido es que muchas funciones están relacionadas tanto al módulo tutor como al módulo del estudiante. El análisis este problema requiere una definición más clara de las interfaces para diferenciar la implementación de los módulos. Por ello, se debe identificar al módulo encargado de realizar cada una de las funciones del STI a fin de que las mismas queden definidas sin solapamientos. De este modo se obtendrán módulos completamente independientes del dominio de la aplicación que podrán intercambiarse (por ejemplo para un dominio diferente) sin necesidad de modificar el resto de la estructura del sistema tutor inteligente.

La definición de cada una de las tareas particulares de los módulos, permite realizar los agrupamientos de tareas en cada uno de ellos. A estos agrupamientos se los define como submódulos pudiéndose realizar un análisis similar al de los grandes bloques del sistema tutor inteligente, definiendo sus funcionalidades básicas y sus interfaces, para permitir la independencia de los submódulos. Por otra parte, al diseñar los módulos componentes del sistema, se ha observado que algunas de las tareas básicas se deben redefinir. Se ha propuesto el rediseño de módulos con asignación de tareas específicas para cada uno de ellos.

Se han analizado los STI existentes, a fin de dar cuenta de los métodos de tutorizado utilizados y de obtener datos para identificar los métodos de enseñanza más efectivos en

relación a la población estudiantil de programación básica. La primera aproximación ha sido la implementación de modelos basados en redes bayesianas. Para ello, han estudiado los diferentes métodos para la determinación de los estilos de aprendizaje de los estudiantes de las carreras de ingeniería y en particular en *Programación Básica* para obtener el perfil de los alumnos y relacionar los estilos de aprendizaje con los métodos de enseñanza [45]. De este modo se busca construir un sistema basado en los modelados de los actores con las siguientes funcionalidades básicas: un módulo tutor que sea capaz de impartir conocimientos de distintas maneras para lograr una adaptación a las necesidades del alumno con respecto a un tema en particular y un módulo de estudiante que se pueda adecuar al espectro de necesidades en las carreras de ingeniería en cuanto al estilo de aprendizaje.

4. Perspectiva para el diseño e implementación de los STI

Para desarrollar STI desde una concepción multidisciplinaria y multilingüística (es decir que los módulos ya construidos se puedan usar cambiando el dominio o solo el idioma) se puede partir de la arquitectura de Carbonell [8], aunque algunos investigadores [46,47] presentan una arquitectura que difiere de la estructura real implementada debido a la existencia de solapamiento de funcionalidades. Se observa que muchos de los conocimientos particulares del dominio (pertenecientes al módulo de dominio) se encuentran dentro de los módulos del *tutor* y del *estudiante* con las consecuentes zonas de solapamiento entre módulos. Por ello, se debe identificar qué módulo será el encargado de realizar cada una de las funciones del STI a fin de quedar definido en su totalidad. De este modo se obtendrán *módulos completamente intercambiables e independientes del dominio de la aplicación.* Además de la modularidad e independencia, se busca STI centrados en las necesidades reales de los estudiantes. Esto significa contar con varios *protocolos*[2] pedagógicos que se ajusten de acuerdo a las necesidades y las preferencias de cada alumno en particular.

El propósito es que el sistema de tutorizado exhiba un comportamiento similar al de un tutor humano, es decir, que se adapte al comportamiento del estudiante en lugar de ser un modelo rígido. Un sistema con estas características *"es un sistema de software que utiliza sistemas inteligentes para asistir al estudiante que requiere de un tutorizado uno a uno y lo guía en su aprendizaje, adicionalmente posee una representación del conocimiento y una interface que permite la interacción con los estudiantes para que puedan acceder al mismo"* [5]

4.1. El módulo del tutor

[2] Se utiliza el término protocolo de enseñanza en referencia al método de enseñanza o método docente.

En este sistema*, el modelo del tutor* es el encargado de definir y de aplicar una estrategia pedagógica de enseñanza (socrática, orientadora, dirigida etc.), de contener los objetivos a ser alcanzados y los planes utilizados para alcanzarlos. Es el responsable de seleccionar los problemas y el material de aprendizaje, de monitorear, y proveer asistencia al estudiante. También de integrar el conocimiento acerca del método de enseñanza, las técnicas didácticas y del dominio a ser enseñado (con integración de planificación y curriculum). Es decir, un sistema de este tipo debe tratar además, los aspectos esenciales del curriculum y de la planificación, ya que los aspectos de curriculum involucran la representación, la selección y la secuenciación del material a ser utilizado y la planificación se refiere a cómo ese material va a ser presentado [48].

Un sistema para tutorizado, no solo debe emular al tutor humano sino que además debería estar diseñado desde una concepción epistemológica acerca de lo que significa enseñar programación en las carreras de ingeniería en relación al perfil y la identidad del futuro ingeniero. Un sistema de este tipo debería proveer algunas características en función de los propósitos por los que el estudiante recurre a él, tales como:

– La perspectiva desde la debe impartir los conocimientos a los alumnos.
– La forma de adaptación a los conocimientos previos de los alumnos.
– La selección de la estrategia de enseñanza más adecuada para el alumno que lo consulta.

Y, cuando el mismo *guíe* al alumno deberá tener "*reglas*" almacenadas para saber qué hacer cuando

– El alumno no puede contestar una pregunta que le hace el tutor.
– El alumno contesta en forma incompleta una pregunta que le hace el tutor.

Es decir un modelado del tutor flexible, es el eje central para el desarrollo y deberá responder a las preguntas: *a) ¿Qué debe hacer el tutor cuando el alumno no puede contestar una pregunta? y b) Qué debe hacer el tutor cuando el alumno contesta en forma incompleta una pregunta?.*

En la literatura analizada se han encontrado dos posturas para la implementación de los conocimientos: una se basa en la estructura sintáctica de lo producido por los tutores humanos [49] y la otra en las metas pedagógicas que se deben cumplir a fin de que el alumno pueda comprender el tema [34,50]. Pero, revisando el problema y utilizando ambas teorías en forma conjunta se obtendrían una serie de pasos que pueden resumir la forma de impartir los conocimientos [51]: a) El tutor debe mantener una jerarquía de *metas* que debe cumplir mientras imparte los conocimientos al alumno quien producirá un resultado que el tutor no

puede predecir de antemano y b) El tutor debe poder explicar un mismo concepto de *diferentes maneras*, de modo que si el alumno no entiende el concepto, el tutor puede continuar efectuando otro acercamiento al mismo tema, explicando el concepto para continuar utilizando un método iterativo a fin de profundizar en el concepto cada vez más, pero paso a paso, o descartar este acercamiento al tema e intentándolo de otra manera.
En este contexto, surgen las posibilidades de aplicabilidad de los sistemas inteligentes a la resolución de problemas de modelado de este tipo. En el campo de los sistemas inteligentes se encuentran las redes neuronales (RN), las cuales son interconexiones masivas en paralelo de elementos simples, que responden a una cierta jerarquía intentando interactuar con los objetos reales tal como lo haría un sistema neuronal psicológico [52]. Las redes neuronales (RN) poseen la característica interesante de asimilar conocimiento en base a las experiencias mediante la generalización de casos [53,54]. Otra posibilidad se encuentra dada por los algoritmos genéticos (AG), que se fundamentan en el concepto biológico de la evolución natural y son utilizados en procesos de optimización [55,56]. La base de estos algoritmos está en los mecanismos de la selección natural de supervivencia de los individuos más aptos, de una población de posibles soluciones [53].

4.2. El módulo del estudiante

El *modelo del estudiante o aprendiz,* es el responsable de establecer un perfil del estudiante, diagnosticando sus deficiencias, según el nivel de conocimiento, formando una imagen instantánea de su comprensión de los contenidos. El modelado del alumno una caraterística que distingue los CAI (Computer Aided Instruction) [57] tradicionales de los STI por su capacidad de adaptación a las necesidades del estudiante. Es decir, el sistema debe determinar el *"estado cognitivo"* del mismo, o sea, cuáles son los conocimientos previos entendidos como las secciones del dominio que el estudiante ya sabe. De este modo, el sistema podrá recomendar la estrategia de estudio más conveniente y el tipo de acción a seguir a través de la resolución de problemas, por ejemplo, y, dentro de ellos, el nivel de adecuación de los ejercicios a dicho dominio [58].

Las acciones del estudiante sobre el modelo de dominio de conocimientos, se pueden modelar a través de conjuntos de reglas que permiten evaluar el conocimiento del aprendiz. En la literatura se describen los modelos: diferencial, de *"overlay"* o superposición, de perturbación o *"buggs",* por simulación, de creencias, de agentes inteligentes, y otros [4]. Un sistema asesor que pudiera diagnosticar el tipo de estudiante, es decir su estilo de aprendizaje, y determinase su estado actual daría información muy útil para saber en qué estadío evolutivo se halla el

estudiante universitario [59]. Un sistema con modelado del estudiante podría aportar cursos de acción ante las diversas dificultades de los alumnos.

El problema del modelado del estudiante se puede dividir en dos partes: a) La selección de una estructura de datos (en el sentido de variables, enlaces y parámetros) [60], y b) La elección de un procedimiento para efectuar el diagnóstico del estado actual del estudiante. En el marco situacional planteado se considera que el problema del modelado podría encararse también a través de la aplicación de sistemas inteligentes tales como las redes neuronales y los algoritmos genéticos que se describen en el apartado anterior.

Tipo de conocimiento	Características
Conocimiento declarativo (*declarative knowledge*)	Es un conjunto de hechos que se organizan para razonar sobre ellos. Tal es el caso de Geografía, y se lo puede representar con una red semántica. Este es el caso de *Scholar* [8] posee nodos que representan los hechos y los enlaces representan relaciones jerárquicas. Con esta estructura sen pueden definir procedimientos de inferencia flexibles sobre la base de conocimientos.
Conocimiento de procedimientos (procedural knowledge)	Es el conocimiento sobre cómo llevar a cabo una tarea y es específico de cada dominio. Una forma de representación es una base de conocimientos y un conjunto de reglas, como en los sistemas expertos. Son ejemplos: el *Tutor de Geometría* de Anderson [62] y *Buggy* [63].
Conocimiento cualitativo	Se usa para modelar relaciones espaciales y procesos dinámicos. El razonamiento causal en los sistemas de diagnóstico de averías es una parte muy importante del conocimiento cualitativo. El trabajo sobre la estructura causal de un dispositivo se usa para determinar problemas potenciales y la redes bayesianas (RB) son una estructura adecuada para modelar este tipo de conocimiento con nodos que modelan relaciones de tipo causal. Un ejemplo es *Hydrive* [64].

Tabla 1: Tipos de conocimiento [61]

4.3. El módulo del dominio

El dominio proporciona los conocimientos presentados en forma adecuada para que el alumno pueda adquirir las habilidades y conceptos, es decir, la capacidad de generar preguntas, explicaciones, respuestas y tareas, y además debe ser capaz de dar respuesta a los problemas y corregir las soluciones presentadas y analizar las diferentes aproximaciones válidas a la solución.

Se debe considerar qué tipo de conocimiento se está modelando según sea: declarativo, de procedimientos y cualitativo como se resume en la Tabla 1 [61]. Anderson agrupa los modelos expertos en tres categorías [65]: los modelos de *caja negra*, los modelos de *caja de cristal* y los modelos *cognitivos* (ver Tabla 2).

Tabla 2: Modelos expertos [61]

Modelo	Características
de caja negra	Es capaz de resolver problemas sobre el dominio. Las soluciones a dichos problemas se usan como ejemplo para los alumnos y para determinar si las soluciones presentadas por éstos son o no correctas. Este modelo es opaco para el alumno a pesar que se realice cientos de cálculos.
de caja de cristal	En este modelo, cada paso en el razonamiento puede ser revisado e interpretado. Para construirlo, se debe utilizar la metodología de un sistema experto. El módulo experto parece adecuado para enseñar al alumno, ya que posee una representación de la forma en que un humano razona para resolver el problema. Un ejemplo es el usado en *Guidon*

	[66] que reutiliza el módulo experto del sistema *Mycin* [67] para enseñar conocimientos relativos a enfermedades infecciosas.
modelos cognitivos	Simulan al humano en el uso del el conocimiento que se quiere enseñar. El objetivo es descomponer el conocimiento en componentes con significado, y usar ese conocimiento de modo similar al humano. La construcción de modelos cognitivos es un proceso complicado y largo que requiere determinar las componentes psicológicas esenciales para modelar el aprendizaje y cuáles no lo son para una menor complejidad computacional. Como ejemplo se tienen los trabajos de Anderson en tutores cognitivos [68].

5. Limitaciones y posibilidades de los STI actuales

Las evidencias acerca de las diferencias entre los modelos teóricos y los implementados realmente surgen desde los planteos de Clancey y Joerger [69] quienes se lamentan que *"...la realidad hoy es una empresa que solo experimentan los programadores",* ya que la mayor parte del desarrollo teórico realizado en las últimas décadas no aportó mucho en cuanto a la estandarización de los STI. Los desarrollos se centraron en tutores para aplicaciones individuales o muy dependientes del dominio y los STI se construyeron en general por programadores y profesionales del área de la informática con poca experiencia en el campo de las ciencias de la educación y en las teorías del aprendizaje. Esto daba buenos sistemas en términos de diseño, uso de recursos y rendimiento, pero muy poco eficientes en la puesta en producción en un ambiente educativo real. En este sentido, se observa que las investigaciones se deben orientar a paliar las debilidades detectadas en el análisis debido a la falta de estandarización para generar STI con componentes reutilizables y de propósito general que puedan aplicarse a una gran diversidad de dominios, en particular el rediseño de los módulos del tutor y del estudiante.

Los nuevos desarrollos de los STI, se deben caracterizan por la inclusión de experiencia adicional basada en el entorno de aprendizaje del estudiante y en los métodos y técnicas de enseñanza. Esto permitirá obtener sistemas más flexibles, adaptados a los intereses del estudiante y con métodos pedagógicos que faciliten el proceso de aprendizaje. Los STI se los ve como un intento para proveer de nuevas oportunidades a los estudiantes permitiéndoles desarrollar procesos mentales de indole superior tales como la resolución de problemas [70]. Estas nuevas formas de interacción posibilitarán a los estudiantes adentrarse en una de las condiciones esenciales de la educación continua permitiendo la relación de sus aprendizaje con los problemas de la vida real. Por otra parte, se pueden concebir tutores que trabajen para eliminar paulatinamente los conceptos erróneos (misconceptions) a fin de poder reelaborar el cambio conceptual [3,19].

5.1. Análisis de los problemas metodológicos

Existen una serie de problemas a resolver a la hora de plantear una propuesta de STI: a) *¿Cuál es el nivel de control del aprendiz sobre sus aprendizajes?* b) *¿Cómo deben interactuar los aprendices? ¿con el sistema o de modo colaborativo?* c) *¿Cómo se debe conformar el equipote desarrollo del STI?* d) *¿Se integran psicólogos cognitivos y pedagogos, diseñadores de interfaces y especialistas en representación del conocimiento?.*

Carbonell [8] y Sleeman y Brown [71] introdujeron las primeras arquitecturas de STI, pero es a partir de los 90 cuando se comienza a dotar a los mismos de cierta forma de tutorizado adaptativo en algunas de sus versiones. Las líneas actuales de desarrollo muestran STI que no se restringen a un método de enseñanza único para todo tipo de alumno y que por otra parte, se puede desarrollar de modo distribuido para bajar los requerimientos de hardware. Así, se busca desarrollar sistemas que permitan: a) Presentar al estudiante el contenido de acuerdo a su estilo de aprendizaje b) Asesorar al estudiante acerca de cómo debería aprender un contenido determinado y cuáles son las habilidades esperadas, c) Tutorizar al estudiante a fin de que pueda cumplir los objetivos del tema en tiempo y forma, d) Asistir al mismo en los procesos de trabajo colaborativo con el tutor y con los pares, e) Efectuar los diagnósticos sobre el rendimiento de los estudiantes y proveerles de herramientas para mejorar su producción. En este sentido, el sistema, debería ser lo suficientemente flexible para permitir que cada estudiante, de acuerdo a su nivel inicial y a su estilo de aprendizaje pudiera elegir *"su propio"* método de enseñanza.

En la actualidad, existen problemas metodológicos en el desarrollo de los STI que no han sido resueltos totalmente, tales como: a) La superposición de funcionalidades que existe en los módulos básicos del sistema, b) El conocimiento del experto que está *hard coded*[3] en aplicaciones individuales, c) Los componentes del STI en general no son reutilizables, tales como los módulos del tutor y del estudiante y la interface de usuario, d) La necesidad de contar con un lenguaje estandarizado para representar el conocimiento y las herramientas para manipularlo [72].

De este modo, cada STI nuevo debe comenzar desde cero, lo que no permite un gran avance en este sentido con la pérdida de tiempo y costo que ello conlleva. De acuerdo a lo señalado entonces: a) Existe un alto costo inicial y de operación para desarrollar desde cero todos los componentes partiendo de la representación del conocimiento en el módulo del dominio, b) Si bien las tendencias en los desarrollo de software son: la modularidad e interoperabildad, en este campo no se ha llegado a desarrollos que reutilicen los módulos del tutor con sus estrategias de enseñanza y del estudiante con su modelado basado en su estilo,

[3] Partes del código fuente que se encuentran subsumidas "*hard-coded*" se definen de tal manera de que no se pueden modificar sin recompilar todo el código fuente.

c) Los STI en general son programas muy pesados que consumen recursos con altos requerimientos de hardware, por lo que se debería pensar en arquitecturas distribuidas y que deriven en multiplataformas pensando en los dispositivos móviles, d) Existe un alto costo de mantenimiento debido a la poca atención que se pone en la actualización, ya que no se apunta a la reusabilidad, intercambio y la distribución. En vista de ello, se puede decir que la tendencia es encauzar los esfuerzos en STI con: a) Componentes modulares reusables y bien documentados, b) Experiencias y evaluaciones sobre rol de las arquitecturas de software en su diseño tendientes a la estandarización para poder obtener el intercambio de módulos, c) Modelos de ontologías para definir y organizar los atributos relevantes del dominio pedagógicamente, permitiendo escribir y compartir las estrategias instruccionales en términos de esos atributos, d) Interfaces con diseño modular y estrategias de comunicación bien definidas, e) Arquitecturas distribuidas que permitan la colaboración y el conocimiento compartido orientadas a la modularidad y reusabilidad del tipo cliente-servidor a través de agentes y con arquitecturas web que integren las aplicaciones de base. f) Técnicas de personalización a fin de obtener patrones de comportamiento de los estudiantes en el sistema a través de técnicas de minería de datos y f) Incorporación de diferentes niveles de adaptación del modo de enseñanza a los requerimientos del estudiante.

El problema global que subsiste hoy día se puede reformular entonces del siguiente modo: *"Existe una necesidad de desarrollar una arquitectura para los STI, de modo que sus componentes puedan ser reutilizables por lo que se deberá contar con las herramientas apropiadas de diseño (estandarizadas) que permitan desarrollar interfaces y submódulos con funciones perfectamente definidas".*

Se requiere de un mayor análisis sobre cada uno de los módulos y esto significa contar con los protocolos pedagógicos adecuados de acuerdo a las necesidades y las preferencias de cada estudiante que tenga en cuenta la diferencia entre los distintos tipos de conocimientos a explicar: el conocimiento declarativo, que incluye los hechos, conceptos y vocabulario y el conocimiento procedural que incluye los pasos, las fórmulas y los algoritmos a utilizar en la resolución de problemas.

Es decir, la premisa principal para el modelado del estudiante será: *"Con un STI que se adapte a las preferencias del estudiante este obtendrá mejores resultados"*. Por otra parte, ya se ha señalado la necesidad de contar con herramientas que realicen el diagnóstico sobre el rendimiento de los estudiantes y que provean al STI de datos basados en predicción, para cambiar la estrategia de enseñanza cuando fuera necesario o simplemente recomendarle al estudiante nuevos ejercicios y problemas. Para determinar el perfil de los alumnos las redes

neuronales son una salida atractiva para agrupar estudiantes que poseen características similares. Un tipo de redes neuronales que se pueden utilizar para esta clasificación son los mapas autoorganizados de Kohonen [52], que permiten realizar una *"clusterizacion"* o agrupamiento a partir del conjunto de individuos que originalmente se utilizó para la etapa de entrenamiento de las mismas.

6. La necesidad de un entorno de desarrollo

Desde el punto de vista del desarrollo de software, se ha observado la necesidad de un *framework* como estructura de soporte definida, plataforma o entorno. Así cada nuevo proyecto podría organizar y desarrollar a partir del mismo. Estos entornos suelen incluyen algunas herramientas: soporte de programas, bibliotecas, lenguaje de *scripting* y software para desarrollar y unir diferentes componentes de un proyecto de desarrollo de programas y facilitan el desarrollo de software, evitando los detalles de bajo nivel, así se pueden centrar los esfuerzos y el tiempo en identificar los requerimientos de software. *"Framework"* es un término usado en programación orientada a objetos para definir un conjunto de clases que definen un diseño abstracto para solucionar un conjunto de problemas relacionados.

El Cambidge Advanced learner´s Dictionary[4] se lo define como: *una estructura de soporte en torno a la cual se puede construir algo*. También a) Es un conjunto de clases que cooperan y forman un diseño reutilizable para un tipo específico de software que brinda una guía arquitectónica que parte el diseño en clases abstractas y define sus responsabilidades y colaboraciones [73], b) Es una infraestructura software que crea un entorno común para integrar aplicaciones e información compartida dentro de un dominio dado [74]. En resumen, se puede decir entonces que: *un framework para el desarrollo es un conjunto de clases que cooperan y forman un diseño reutilizable formando una infraestructura que facilita y agiliza el desarrollo de las aplicaciones.*

En la Figura 1 se presentan en forma resumida los métodos, técnicas y herramientas que se requieren para desarrollar el STI desde la perspectiva planteada y se señalan el área de pertenencia de cada uno de ellos: psicología cognitivas, teorías de aprendizaje y evaluación, diseño de interfaces, computación gráfica, bases de datos, herramientas y técnicas de sistemas inteligentes y herramientas y métodos de ingeniería de software.

[4] Disponible en www.dictionary.cambridge.org y consultado el 29/06/09.

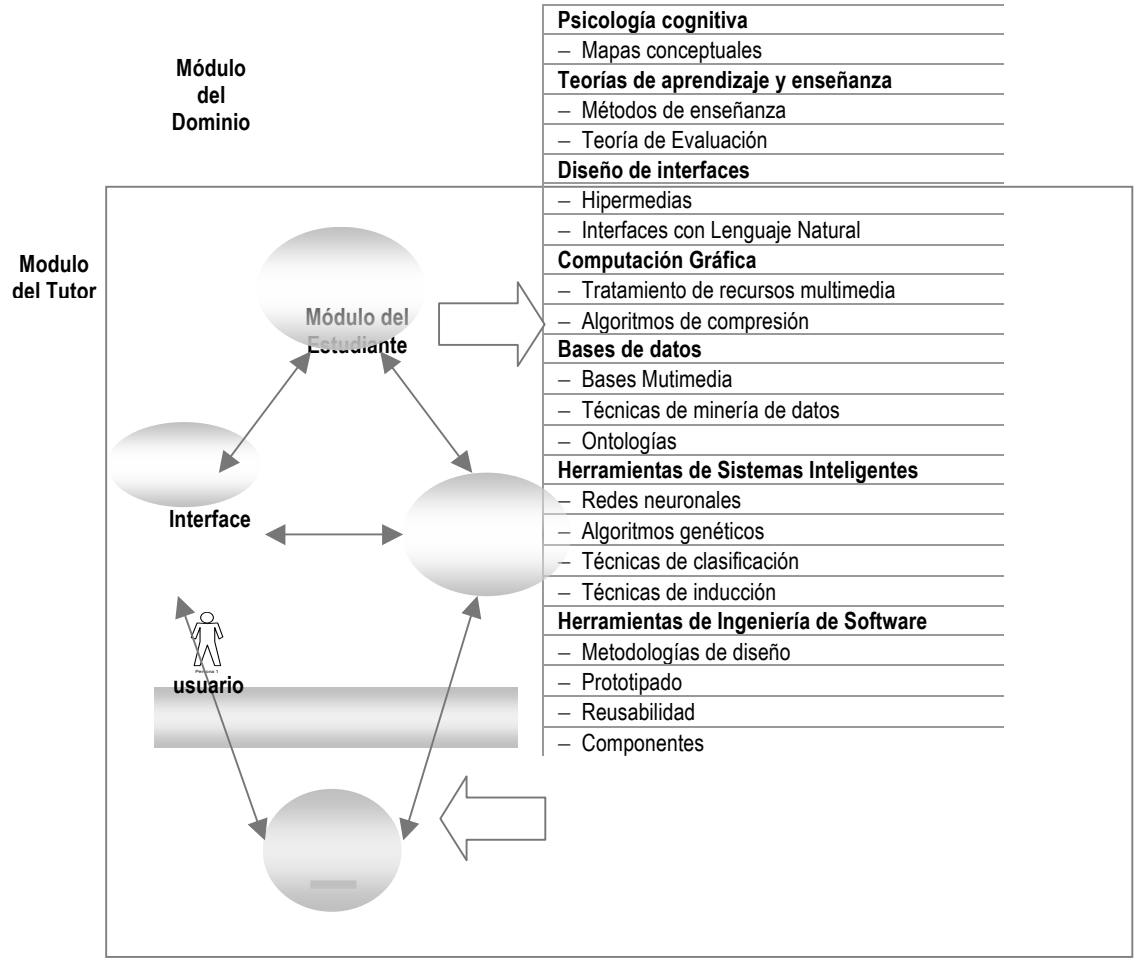

Figura 1: Métodos, técnicas y herramientas para la creación de STI [75]

7. Conclusiones

Se presentó la evolución y el estado actual de los desarrollos de STI sistematizando el conocimiento existente del tema. Se analizaron las debilidades principales en los STI y se plantearon las directrices que servirán de guía a los futuros desarrollos. Ya se han rediseñados los módulos principales y sus funcionalidades asociadas y se prevé continuar la investigación hacia una metodología para el diseño, desarrollo de los STI. En comunicaciones previas se presentaron los dos esquemas de STI posibles para una arquitectura distribuida donde muchas de las partes pueden modificar su posición, pero estos cambios se deben efectuar a la luz de las tecnologías implementadas a fin de optimizar los recursos disponibles [76]. Por otra parte, se piensa que una opción interesante es la creación de un entorno que facilite la tarea de construcción.

Como ya se señaló se buscan STI con: a) componentes modulares que sean reusables y que estén bien documentados, b) Aplicaciones y evaluaciones del papel de las arquitecturas de software en el diseño a fin de poder aplicar criterios de estandarización para intercambio entre módulos, c) Estrategias de comunicación bien definidas, d) Técnicas de personalización a fin de obtener patrones de comportamiento de los estudiantes en el sistema a través de

algoritmos para agrupamiento y técnicas de minería de datos y e) Incorporación de diferentes niveles de adaptación del modo de enseñanza a los requerimientos del estudiante.

8. Agradecimientos

Este artículo es parte del PID *Modelado del tutor basado en redes neuronales para un sistema tutor inteligente.* SeCyT 2007-2008. UTN-FRBA EZINBA639. Programa Incentivos código 25/C099.

9. Referencias

[1]. Khuwaja, R.A. (1994) *A Model of Tutoring: Facilitating Knowledge Integration Using Multiple Models of the Domain.* Ph.D., Illinois Institute of Technology
[2]. Shim, L. (1991) *Student Modeling for an Intelligent Tutoring System: Based on the Analysis of Human Tutoring Sessions.* Ph.D., Illinois Institute of Technology
[3]. Perkins, D. (1995) *La escuela inteligente*. Gedisa
[4]. Casas, M. (1999) *contribuições para a modelagem de um ambiente inteligente de educação baseado em realidade virtual.* Tesis Doctoral Universidade Federal de Santa Catarina. Programa de Pós-graduação em Engenharia de Produção.
[5]. VanLehn, K. (1988). *Student Modelling.* M. Polson. Foundations of Intelligent Tutoring systems. Hillsdale. N.J. Lawrence Erlbaum Associates, 55-78
[6]. Wolf, B. (1988). *Theoretical frontiers in building machine tutor*. En Kearsley, G. (Ed.) Artificial Intelligence and Instruction. Applications and methods. Addison-Wesley. p. 259-267.
[7]. Giraffa, L.M.M.; Nunes, M. A.; Viccari, R.M. (1997) *Multi-Ecological: an Learning Environment using Multi-Agent architecture.* MASTA'97: Multi-Agent System: Theory and Applications. Proc. Coimbra: DE-Universidade de Coimbra.
[8]. Carbonell, J. R. (1970). *AI in CAI: An artificial intelligence approach to computer assisted instruction.* IEEE transaction on Man Machine System. Vol.11, Nro. 4, p. 190-202.
[9]. Stevens, A.; Collins, A. (1977). *The goal structure of a Socratic tutor*. In Proceedings of the National ACM Conference. NY.
[10]. Brown, S. y Burton, R. R. and J. de Kleer. (1982) *Pedagogical*, *natural language and knowledge engineering techniques in Sophie* I, II and III. In D. Sleeman and J. S. Brown, editors, ITS p 227-282, NY, 1982. Academic Press.
[11]. Clancey, W. J. (1991). *Intelligent tutoring systems: A tutorial survey*, en Applied Artificial Int. A Sourcebook. McGraw-Hill.
[12]. Burton, R. R.; Brown, J. S. (1981). *An investigation of computer coaching for informal learning activities.* In: Sleeman, D., Brown, J. (eds.): ITS, Ch. 4, p. 79-98, London: Academic Press.
[13]. Brown, J. y Burton, R. (1978) *Diagnostic Models for Procedural Bugs in Mathematical Skills*, Cognitive Science, No. 2, pp. 155-192.
[14]. Brown, E.; Palincsar, B. (1989) *Guided, Cooperative learning and the individual knowledge acquisition*. En Resnick B. (comp.) Knowing, learning, and instruction. Hillsdale. N.J.
[15]. Wolf, B. (1984). C*ontext Dependent Planning in a Machine Tutor.* Ph.D. Dissertation, University of Massachusetts,
[16]. Johnson, W. L. and Soloway, E. (1984). Proust:Knowledge-based program understanding. In Proceedings of the 7th international conference on SE, Florida, pages 369380.
[17]. Cruz Feliú, J. (1997). *Teorías del aprendizaje y teorías de la enseñanza.* Trillas.
[18]. Bruner, J. (1991) Actos de significado. Más allá de la revolución cognitiva. Madrid: Alianza.
[19]. Pozo, J. I. (1998). *Teorías cognitivas del aprendizaje*. Morata.
[20]. Pozo Municio, I. (1999). *Aprendices y Maestros*. Alianza.
[21]. Wu, H. B. (1995). *Rough set approach to user modeling*. http://www.eecs.lehigh.edu/~bhw2/ Consultado 21/06/07
[22]. Waern, A. (2001) *What is an intelligent interface?.* Notas de Seminario de introductorio. Centro de Investigaciones en Computación de Suecia. Disponible en http://www.sics.se /~annika/publications.html. Consultado el 13/03/07.
[23]. Escolano, A. *et al.* (1987) *Epistemología y educación*. Sígueme. Salamanca.
[24]. Cubero, L.N. (2003) *Pensar la Educación.* Pirámide.

[25]. Gertner, A. S; Conati; C y VanLehn, K. (1998). *Learning Procedural help in Andes: Generating hints using a Bayesian network student model*. Research & Development. AAAI.
[26]. Gertner, A.S. y VanLehn, K. (2000). *Andes: A Coached Problem Solving Environment for Physics.* Lecture Notes In Computer Science; Vol. 1839 Proceedings of the 5th International Conference on ITS. Pages: 133-142.
[27]. Galvin, T. (1994) Tesis Doctoral: *Mebuilder*: An Object-Oriented Lesson Authoring System for Procedural Skills Master's Thesis, Naval Postgraduate School. Monterey.
[28]. Rowe N. C. and T. Galvin (1998) An authoring system for intelligent tutors for procedural skills. *IEEE Intelligent Systems, 13*, 3. May/June, 61-69.
[29]. Litman D. J. and Silliman. S. (2004). *Itspoke: An Intelligent Tutoring Spoken Dialogue System.* In Proc.of the Human Language Technology Conf. 4th Meeting of the North American Ch. of the Assoc. for Computational Linguistics Boston.
[30]. Kim, J. H. (1989). *CIRCSIM-Tutor: An Intelligent Tutoring System for Circulatory Physiology*. Ph.D. Tesis, Illinois Institute of Technology.
[31]. Kim, J. H. (2000) *Natural Language Analysis and Generation for Tutorial Dialogue.* Ph.D. Tesis, Illinois Inst. of Technology.
[32]. Cho, B. (2000). *Dynamic Planning Models to Support Curriculum Planning and Multiple Tutoring Protocols in Intelligent Tutoring Systems.* Ph.D. Tesis, Illinois Inst. of Tech.
[33]. Hume, G.; Evens, M. (1992) *Student modeling and the classification of errors cardiovascular intelligent tutoring system.* Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society Conference, Utica, IL.
[34]. Hume G., Michael, J; Rovick, A.; Evens, M. (1996), *Hinting as a tactic in one-on-one tutoring*. Journal of Learning Sciences Vol. 5, No. 1 (1996), p. 23-47.
[35]. Shah, F. (1997). *Recognizing and Responding to Student Plans in an Intelligent Tutoring System: Circsim-Tutor* Ph.D. Tesis, Illinois Institute of Technology.
[36]. Evens, M. W.; Stefan, B.; Ru-Charn, C.; Freedman; Glass, M; Hee Lee, Y.; Leem Seop, Shim; Woo Woo, C.; Yuemei, Z.; Yujian, Z,; Joel, A. M.; Rovick, A. A. (2001). *CIRCSIM-Tutor: An Intelligent Tutoring System Using Natural Language Dialogue.* MAICS 2001, Oxford, OH, p. 16-23.
[37]. DiPaolo, R.E., Graesser, A.C., Hacker, D.J., White, H.A., y TRG (2002). *Hints in human and computer tutoring.* In M. Rabinowitz (Ed.) The impact of media on technology of instruction. Mahwah, NJ: Erlbaum.
[38]. Graesser, A.C., Chipman, P., Haynes, B.C. y Olney, A. (2005a). AutoTutor: An intelligent tutoring system with mixed-initiative dialogue. *IEEE Trans. in Ed.*, 48, 612-618.
[39]. Graesser, A.C., Olney, A., Haynes, B.C. y Chipman, P. (2005b). *AutoTutor: A cognitive system that simulates a tutor that facilitates learning through mixed-initiative dialogue*. In C. Forsythe, M. Bernard & T. Goldsmith (Ed.), Cognitive systems: Human cognitive models in systems design. NJ: Erlbaum
[40]. Graesser, A.C., Jackson, G.T. y McDaniel, B. (2006). AutoTutor holds conversations with learners that are responsive to their cognitive and emotional states. *Ed. Tech.* (in press).
[41]. Chipman, P., Olney, A., & Graesser, A. C. (2005). *The AutoTutor 3 architecture: A software architecture for an expandable, high-availability ITS.* Proceedings of WEBIST 2005: p. 466-473. Portugal: INSTICC Press.
[42]. Mitrovic, A., Suraweera, P., Martin, B., Zakharov, K., Milik, N., Holland, J. (2006) *Authoring nstraint-based tutors in ASPIRE.* M. Ikeda, K. Ashley, and T.-W. Chan (Eds.): ITS 2006, LNCS 4053, pp. 41-50.
[43]. Mitrovic, A., Martin, B. & Mayo, M. (2002) *Using evaluation to shape ITS design: Results and Experiences with SQL-Tutor*. Int. J. User Modeling and User-Adapted Int., 12 (2-3), p. 243-279.
[44]. Mitrovic, A. (2003) An intelligent SQL tutor on the Web Int. *J. Artificial Intelligence in Educ.*, vol. 13, no. 2-4, 173-197.
[45]. Cataldi, Z. y Lage, F. (2007). *El problema del modelado del estudiante en Sistemas Tutores Inteligentes. .* II TE&ET 12-15 de junio. Facultad de Informática, Univ. Nacional de La Plata.
[46]. Salgueiro, F. A, Costa, G., Cataldi, Z., García Martinez, R. y Lage, F. J. (2005). *Sistemas inteligentes para el modelado del tutor.* Proc. in GCETE'2005**,** marzo 13-15
[47]. Costa, G.; Salgueiro, F. A., Cataldi, Z., García Martinez, R. y Lage, F. J. (2005). *Sistemas inteligentes para el modelado del estudiante* Proc. In GCETE'2005**,** marzo 13-15.
[48]. Coll, C. (1994). *Psicología y curriculum*. Editorial Paidos, Barcelona.
[49]. Seu, Jai, Ru-Charn Chang, Jun Li, Evens, M.; Michael, J. and Rovick, a. (1991). *Language Differences in Face-to-Face and Keyboard-to-Keyboard tutoring Session*. Proceedings of the Cognitive Science Society.
[50]. Evens, M. W.; Spitkovsky, J.; Boyle, P.; Michael, J.; Rovick, A. A. (1993). *Synthesizing tutorial Dialogues*. Preoceedings of the 15th Annual Conference of the Cognitive Science Society.

[51]. Freeva, R.; Evens, M. (1996). *Generating and revising multi-turn text plans in STI*. LN in Computer Science. P 632-640.
[52]. Kohonen, T. (1988). Self-*Organizing Maps Springer Series in Information Sciences*. Vol. 30, Springer, Berlin, NY. P. 236.
[53]. García Martínez, R.; Servente; M. y Pasquini (2003) *Sistemas Inteligentes*. Nueva Librería.
[54]. del Brío, M. y Sanz Molina, M. (1997) *Redes Neuronales y Sistemas Borrosos*, Ed. Ra-Ma.
[55]. Davis, L. (1991). *Handbook of Genetic Algorithms.* New York. Van Nostrand Reihold.
[56]. Falkenauer, E. (1999). *Evolutionary Algorithms: Applying Genetic Algorithms to Real-World Problems*. Springer, New York, Pag 65-88.
[57]. Ayala Rivera, V.; González López, L. (2003) *Herramienta para la generación de lecciones de Español bajo el esquema establecido por el CSLR*. Universidad de las Américas-Puebla.
[58]. Brachman, R.J. (1988) *The basis of knowledge representation and reasoning.* AT&T. Technical Jurnal. 67, 1:15.
[59]. Piaget, J. (1969). *The mechanisms of perception*. Rutledge & Kegan Paul, London.
[60]. Conejo, R., Millán, E., Pérez de la Cruz, J.L., y Trella, M. (2000). *An Empirical Approach to On-Line Learning in SIETTE.* In Proc. of 3rd Int. Conf. on ITS. LNCS 1839, Springer Verlag.
[61]. Millán, E. (2001) *Sistema bayesiano para modelado del alumno.* Tesis Doctoral Universidad de Málaga.
[62]. Anderson, J.R., & Reiser, B.J. (1985). The LISP tutor. *Byte, 10*, 159-175
[63]. Brown, J. y Van Lehn, K. (1980) Repair Theory a generative theory of bugs in procedural skills. *Cogn. science* 4, 379-426.
[64]. Mislevy, R., & Gitomer, D. H. (1996). The Role of Probability-Based Inference in an Intelligent Tutoring System. *User Modeling and User-Adapted Interaction, 5*, 253-282.
[65]. Anderson, J. R. (1988). The Expert Module. En M. C. Polson & J. J. Richardson (eds.), *Foundations of Intelligent Tutoring Systems*. Hillsdale, NJ: Lawrence Erlbaum Assoc. Publishers.
[66]. Clancey, W. J. (1987). *Knowledge-Based Tutoring: the GUIDON Program*. Cambridge, MA: MIT Press.
[67]. Shortlife, E. H. (1976). *Computer Based Medical Consultation: MYCIN*. New York: Elsevier Science Publishers.
[68]. Anderson, J. R., Corbett, A., Koedinger, K., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences, 4*(2), 167-207.
[69]. Clancey, W. J.; Joerger, K. (1988). *A practical authoring shell for apprenticeship learning*. Proc. Of ITS'88: First Intl. Conf. on Intelligent Tutoring Systems, Montreal, Canada, pp. 67-
[70]. Vigotsky, L. (1978) *Mind in society. The development on Higher psychological process*. Harvard University Press.
[71]. Sleeman, D; Brown, J.S. (1982) *Intelligent Tutoring Systems*. Academic Press,
[72]. Rodrígues, M. (2005) *Future Challenges in Intelligent Tutoring Systems. A framework.* m-ICTE2005. Cáceres Junio 7-10.
[73]. Gamma E., *et al.* (1995), Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley.
[74]. SEMATECH (1998) Computer Integrated Manufacturing (CIM) Framework Specification version 2.0, SEMATECH Tech. Transfer #93061697J-ENG. http://www.sematech.org
[75]. Cataldi, Z. y Lage, F. (2009). *Entorno de desarrollo para Sistemas Tutores Inteligentes.* ICECE 2009. VI Congreso International en Educación en Ingeniería y Computación. Buenos Aires. ITBA 8-11 de marzo. 978-85-89120-63-0
[76]. Cataldi, Z., y Lage, F. (2008). *Modelo de Sistemas Tutor Inteligente distribuido para educación a distancia.* IX Encuentro Internacional Virtual Educa Zaragoza 2008. 14-18 de julio.