Estudio comparativo del funcionamiento de Sistemas tutores inteligentes orientados a la enseñanza de los fundamentos de Control Automático.

Oscar Eduardo Cala Wilches

Universidad Nacional de Colombia

Facultad de Ingeniería, Departamento de Ingeniería de Sistemas e Industrial

Bogotá, Colombia

2014

Estudio comparativo del funcionamiento de sistemas tutores inteligentes orientados a la enseñanza de los fundamentos de control automático

Oscar Eduardo Cala Wilches

Tesis de investigación presentada como requisito parcial para optar al título de:

Magister en Ingeniería de Sistemas y Computación

Director (a):

Ph.D. Victor Hugo Grisales Palacio

Línea de Investigación:

Sistemas tutores inteligentes.

Grupo de Investigación:

Grupo de Investigación en Automática de la Universidad Nacional de Colombia - GAUNAL

Universidad Nacional de Colombia

Facultad de Ingeniería, Departamento de Ingeniería de Sistemas e Industrial

Bogotá, Colombia

2014

A mi querida madre Maria Ligia, quien con su valioso esfuerzo, humildad, comprensión e infinito amor, logró darme una vida que ha valido la pena y merece todos los esfuerzos del mundo. Sin su apoyo constante, preocupación, sacrificio, cariño y constantes enseñanzas nada de esto hubiese sido posible.

Agradecimientos

Al IngenieroVictor Hugo Grisales Palacio, por su valiosa colaboración en todos los momentos en que necesité compartir mis ideas y recibir realimentación acerca de la validez de las mismas. Por la increíble cordialidad con que nos trata a los estudiantes, docentes y funcionarios, motivándonos a siempre expresarnos de la mejor manera posible y a mantener un lenguaje adecuado en nuestras discusiones. Por la infinita disposición y compromiso con las actividades de la universidad, que requieren de más personas con los mismos niveles de compromiso. Por la confianza que depositó en mí, en el proyecto, la realimentación siempre adecuada para que uno pueda aprender de sus errores y de sus logros. Agradezco haber contado con un director cuyas enseñanzas no solo me hacen mejor profesional, sino una mejor persona.

A Jonathan Sewall, Project Director en el Human-Computer Interaction Institute de la universidad Carnegie Mellon, quien hizo que mi instancia en Pittsburg fuera realmente productiva.Con sus innumerables respuestas e invaluable conocimiento, me permitió comprender cosas que nunca hubiera podido sin su ayuda.

A Jay Holland y Tanja Mitrovic, por facilitarnos la documentación, de la herramienta ASPIRE, otorgarnos el acceso a esta y permitir que nuestros estudiantes fueran parte activa de este trabajo. Sin su valiosa ayuda y sus constantes respuestas no hubiésemos podido llevar este proyecto a cabo.

A Bernardo Islas y Fernando Cuenca, Project Managers en Globant LLC, por su comprensión, ayuda y confianza quehizo posible que mi tiempo pudiera dedicarse a este gran trabajo. No solo una, sino tres veces.

Resumen

En este trabajo se presenta un estudio comparativo de diferentes aspectos relacionados con el funcionamiento de varios tipos de sistemas tutores inteligentes orientados a la enseñanza de los fundamentos de control automático. Se realiza un análisis comparativo de la capacidad de modelamiento, que varias técnicas de modelamiento del conocimiento para sistemas tutores inteligentes presentan al tratar de modelar el conocimiento típico que se imparte en las clases de fundamentos de control automático. Se seleccionan dos de las técnicas analizadas y se realiza el diseño y la implementación de diferentes sistemas tutores inteligentes utilizando las técnicas seleccionadas. Se presentan varias consideraciones de diseño y se hace la comparación de las herramientas de autor utilizadas para construir los sistemas tutores inteligentes. Finalmente se hace un estudio de evaluación basado en una prueba piloto con estudiantes que permite evidenciar que la mayoría de las decisiones de diseño tomadas, permitieron crear sistemas tutores inteligentes que son bien valorados por los estudiantes en términos de usabilidad, calidad y motivación.

Palabras clave: Sistemas tutores inteligentes, model-tracing, example-tracing, constraint-based modelling, ASPIRE, CTAT, tutores cognitivos.

Abstract

This document presents a comparative study on different aspects of the performancefor various types of intelligent tutoring systems orientedto teaching the basics of automatic control. A comparative analysis of the modelingability of several knowledge modeling techniques for intelligent tutoring systems is presented. Two of the techniques are selected and the design and implementation of intelligent tutoring different systems is

accomplished using the techniques selected. Several design considerations are presented and a comparison of the authoring tools used to build the intelligent tutoring systems is also made. Finally an evaluation study based on a pilot test with students, demonstrates that most design decisions taken, helped the creation of intelligent tutoring systems that are highly rated by students in terms of usability, quality and motivation.

Keywords: Intelligent tutoring systems, model-tracing, example-tracing constraint based modeling, ASPIRE, CTAT, cognitive tutors.

Contenido

Lista de figuras

Lista de tablas

Introducción

Los Sistemas Tutores Inteligentes (ITS por sus siglas en ingles: Intelligent Tutoring Systems) son sistemas de aprendizaje que han demostrado proporcionar considerables avances educativos en comparación con otros modelos de instrucción (Koedinger, Anderson,Hadley, & Mark, 1997; Murray, Woolf, & Marshall, 2004; Ritter, Anderson, Koedinger, & Corbett, 2007; Vanlehn et al., 2005). De acuerdo con (Nwana, 1990) estos sistemas están diseñados para saber qué enseñar y cómo hacerlo, todo en función del estudiante que en particular se encuentre utilizando el sistema. Los Sistemas tutores inteligentes han sido investigados desde 1980 porque hubo un gran interés en el desarrollo de sistemas capaces de proporcionar instrucción individualizada a los estudiantes. El objetivo principal ha sido la creación de sistemas computacionalescapaces de desempeñarse de la misma forma que lo haría un profesor muy bueno y experto en un área determinada. Tener tal sistema permitiría abrir la oportunidad de proveer masivamente a los estudiantes instrucción individualizada y por lo tanto, tomar ventaja de las experiencias de tutoría uno a uno; experiencias que según investigaciones bien conocidas permiten tener mejoras de rendimiento en hasta dos desviaciones estándar (Bloom, 1984; Cohen, Kulik, & Kulik, 1982; Bourdeau & Grandbastien, 2010). El planteamiento de este interés tan amplio dio inicio al campo de estudio de la Inteligencia Artificial en la Educación (AIED por sus siglas en ingles: Artificial Intelligence in EDucation).

En este documento se presenta una descripción de las metodologías empleadas, distintos análisis realizados y diferentes resultados obtenidos en cada una de las etapas que fueron llevadas a cabo durante la elaboración de un estudio comparativo que permitió determinar cuál es el tipo de Sistema Tutor Inteligente más apropiado para la enseñanza de los fundamentos de control automático.

La teoría de control automático es una rama interdisciplinaria de la ingeniería que se ocupa del comportamiento de los sistemas dinámicos y del estudio de cómo su comportamiento se modifica por medio de la retroalimentación. El campo de la teoría de control se puede dividir en dos ramas: la teoría de control lineal, que se aplica a los sistemas gobernados por ecuaciones diferenciales lineales y la teoría de control no lineal, que estudia los sistemas que se rigen por ecuaciones diferenciales no lineales. Cursos

de introducción a estos campos, en especial para la rama de estudios de control lineal, se imparten en varios programas de la Universidad Nacional de Colombia, sede Bogotá, siendoIngeniería Mecatrónica, Ingeniería Electrónica y Eléctrica, e Ingeniería Química los de mayor relevancia respecto a la teoría de control automático.

Específicamente en el Grupo de Investigación en Automática de la Universidad Nacional de Colombia, existe actualmente un alto interés en el desarrollo de herramientas educativas y en el estudio teórico de los fundamentos que soportan dichas herramientas. El objetivo principal es y ha sido, dar un adecuado soporte a los currículos académicos en los cuales se imparten cátedras relacionadas con el campo de la teoría de control (Barrios et al., 2013; Cala & Grisales, 2014; Casteblanco & Avila, 2009).Esta es una de las razones por las cuales se llevó a cabo el estudio comparativo presentado en este documento, pues al finalizar el estudio, se reconocen y comprenden a profundidad los fundamentos teóricos de cada uno de los tipos de sistemas tutores inteligentes abordados; se identifican sus posibilidades reales de implementación tecnológica incluyendo ventajas, desventajas, posibilidades y limitaciones; y se conforma de esta forma un entendimiento riguroso que permite el planteamiento de nuevos retos en términos teóricos y de implementación hacia el objetivo de brindar un mejor soporte a los estudiantes en sus experiencias de aprendizaje.

Existen técnicas de carácter general para la construcción de los sistemas tutores inteligentes: Modelamiento basado en restricciones (Antonija Mitrovic, Martin, & Suraweera, 2007), Modelamiento basado en reglas de producción (Aleven, 2010), Modelamiento basado en ontologías (Bourdeau, Mizoguchi, Psyché, & Nkambou, 2004; Fournier-Viger, Nkambou, & Nguifo, 2010; Martin, Mitrovic, Suraweera, & others, 2007) y una gran comunidad de investigadores que producen anualmente resultados de investigacion muy importantes relacionados con dichas técnicas. En el dominio de los fundamentos de control automático las implementaciones de sistemas tutores inteligentes han estado limitadas a temáticas muy puntuales y adicionalmente, se han realizado usando técnicas que no sonaquellas que cuentan con el mayor soporte teórico y práctico. Esto ha causado que dichas implementaciones y sus resultados derivados no puedan ser reutilizados para otras temáticas u otros dominios de conocimiento.

El estudio comparativo presentado en este documento incluye el desarrollo y la evaluaciónde dos tipos de sistemas tutores para un conjunto de problemas seleccionados metodológicamente. Se consideran tres tipos de tutores inteligentes: **Tutores Model Tracing**(Aleven, 2010; Koedinger et al., 1997), **Tutores Example Tracing**(Aleven, Mclaren, Sewall, & Koedinger, 2009; Koedinger, Aleven, Heffernan, McLaren, & Hockenberry, 2004) y **Tutores Constraint Based**(Antonija Mitrovic, 2003; Antonija Mitrovic, Mayo, Suraweera, & Martin, 2001; A. Mitrovic & Ohlsson, 1999). El estudio comparativo también tiene en cuenta las herramientas de autor que se encuentran disponibles para ayudar en los procesos de construcción de estos sistemas.

Estos tres tipos de tutores inteligentes cuentan con un gran soporte teórico y práctico que ha sido el resultado de la investigación realizada por la comunidad de AIED durante los últimos 20 años (Nkambou, Mizoguchi, & Bourdeau, 2010; Trausan-Matu, Boyer, Crosby, & Panourgia, 2014) y por lo tanto son grandes exponentes del campo de estudio.

Debido al gran número de posibles problemas que un estudiante puede resolver con la ayuda de un sistema tutor inteligente, se tuvo que realizar la selección de un conjunto de problemas reducido, con el objetivo de implementar para dicho conjunto, la asistencia típica que los sistemas tutores inteligentes proporcionan. Los sistemas tutores inteligentes fueron creados por medio de lasdiferentes técnicas consideradas en este trabajo. Para lograr este objetivo (seleccionar un grupo reducido de problemas), se aplicó una metodología concebida durante este trabajo, que fue reportada satisfactoriamente a la comunidad científica y que permite la selección de los problemas que promueven la fácil implementación de buenas prácticas en el diseño de sistemas tutores inteligentes.

Un análisis teórico y comparativo inicial, permitió la selección de dos de los tres tipos de sistemas tutores inteligentes considerados. El marco de referencia para la comparación tuvo en cuenta características relacionadas con la factibilidad, la facilidad y la efectividad del modelamiento de las temáticas de control automático para cada uno de los tipos de tutores.

Se presenta una descripción de la fase de desarrollo de los prototipos de software, la cual permitió alcanzar como resultado la creación sistemas tutores inteligentes funcionales según los requerimientos planteados en términos de objetivos de

aprendizaje. El resultado final son dos productos de software que permiten la ejecución de los flujos de trabajo necesarios para la resolución de problemas propios de los fundamentos de control y que proporcionan los servicios propios de un sistema tutor inteligente.

Aunque teóricamente las técnicas consideradas en este estudio comparativo son de propósito general, su aplicación a un dominio de conocimiento específico suele evidenciar algunas limitaciones. Uno de los resultados de este trabajo fue reconocer cuales son las limitaciones especificas de las técnicas sujetas a comparación en el contexto especifico de un conjunto de temáticas típicas de los cursos introductorios en teoría de control. Estas limitaciones se reconocen a nivel teórico, a nivel de implementación tecnológica teórica y también a nivel de las herramientas de autor disponibles.

Los sistemas tutores inteligentes implementados fueron probados experimentalmente con un grupo de estudiantes voluntarios. Durante el despliegue experimental, los estudiantes pudieron interactuar con ambos tipos de tutores y de esta forma pudieron comparar su comportamiento y expresar sus opinionesacerca de la utilidad y usabilidad de cada sistema por medio de una encuesta.

La disponibilidad de los comentarios de los estudiantes permitió realizar un análisis adicional relacionado con las funcionalidades típicas de un sistema tutor inteligente permitiendo comprender la importancia de contar con sistemas que brinden un gran conjunto de opciones y la gran carencia tecnológica que existe debido a la inexistencia de herramientas de código libre que soporten la construcción de este tipo sistemas que cuenten con un conjunto amplio de funcionalidades.

1. Sistemas Tutores inteligentes

En este capítulo se hace una descripción general de los sistemas tutores inteligentes. Se describe su estructura básica y se realiza un breve recorrido de su desarrollo histórico que permite reconocer el origen de este tipo de sistemas, la evolución de su nombre y las diferentes definiciones dadas por algunos investigadores importantes en el área. También se presentan varios escenarios de aplicación en el campo de control automático y se discute acerca de la carencia de implementaciones en este campo de estudio. Adicionalmente en este capítulo se presenta una descripción de cada uno de los tres tipos de tutores considerados en el estudio comparativo y se indican los elementos más importantes de las teorías cognitivas que los soportan.

1.1 Introducción a los sistemas tutores inteligentes

Muchos sistemas tutores inteligentes han sido desarrollados en los últimos 30 años(Woolf, 2008).Cada uno ha contribuido de alguna manera, en la creación de la arquitectura general de sistemas tutores inteligentes que se usa hoy en día. Algunos de los trabajos más influyentes y que han tenido relación con la educación en control serán discutidos brevemente para dar una perspectiva histórica de las mayores contribuciones al campo.

1.1.1 Desarrollo histórico e importancia.

Desde la Invención del computador electrónico han existido varias opiniones diferentes acerca de cómo hacer uso de dicho dispositivo. Un grupo ve el computador como un dispositivo

numérico de cálculo con el cual se pueden realizar operaciones de manera ultra rápida y con un alto grado de exactitud. El otro grupo conceptualiza el computador como un manipulador de símbolos el cual puede ser entrenado de manera que pueda realizar decisiones de tipo lógico, semejantes a las decisiones que toman los humanos (Butz, Duarte, Miller, 2006). Este último grupo, fue el responsable de fundar la disciplina hoy conocida como Inteligencia Artificial (IA), la cual, con el paso del tiempo se ha dividido en muchas sub-disciplinas formando así nuevas áreas de trabajo como las Redes Neuronales, los Sistemas Expertos, la inteligencia Computacional, el Procesamiento Natural de Lenguaje, etc.

Una de las disciplinas de la Inteligencia Artificial (IA) que ha sido aplicada con suficiente éxito en la educación ha sido la que investiga los sistemas expertos: sistemas de computación que se desempeñan de manera muy similar a la forma como se desempeñaría un humano en un tema específico (Butz, 1987). El objetivo de los investigadores que han aplicado esta disciplina a la educación ha sido, entre muchos otros, desarrollar un tutor sistematizado que pueda desempeñarse de manera similar y al mismo nivel que un excelente y experto tutor humano. El planteamiento de este objetivo tan amplio dio inicio al campo de estudio de la Inteligencia Artificial en la Educación (AIED por sus siglas en ingles: Artificial Intelligence in EDucation).

AIED inició como un campo de estudio (Wenger 1987) cuando Benjamin Bloom definió el problema conocido como el “problema de dos Sigmas” ("two-sigma problem") (Bloom, 1984) y cuando Cohen (Cohen et al. 1982) presentó a la comunidad científica un meta -análisis acerca de los procesos de tutoría que se llevaban a cabo en diferentes escuelas de estados unidos (Bourdeau J,2010). En su trabajo, Bloom declara que aquellos estudiantes que reciben instrucción uno-a-uno se desempeñan mejor que los estudiantes que reciben instrucción grupal. Los estudiantes del primer grupo mejoran en sus calificaciones a los del segundo por una cantidad igual a dos veces la desviación estándar sobre las calificaciones del grupo completo, lo cual significa que un estudiante promedio que ha recibido instrucción uno -a-uno se desempeña tan bien como el 2% de los mejores estudiantes que reciben instrucción grupal . Resultados similares fueron presentados por Cohen quien confirmó que los estudiantes que recibían tutoría uno-a-uno tenían un mejor desempeño que los estudiantes del grupo de control quienes recibían cátedra de forma grupal (Cohen et al. 1982).

Como consecuencia de estos estudios, Bloom hizo un llamado a la comunidad para promover y mejorar la efectividad en la instrucción de grupos, debido a que popularizar la instrucción

individualizada era una opción costosa y poco viable (Bourdeau J,2010). Sin embargo los investigadores en ciencias de la computación, psicología y educación, vieron este hecho como una posibilidad prometedoray muy justificada para investigar y realizar la construcción de sistemas informáticos que pudieran impartir una instrucción efectiva. De esta forma, la popularización de dichos sistemas informáticos podría realizarse a menor costo y tener un impacto importante en el sistema educativo. Si estos sistemas lograran generar al menos la mitad del impacto que los tutores humanos, los beneficios para la sociedad, en términos educativos, podrían ser fundamentales (Corbett, Koedinger & Anderson, 1997).

Fue por esto que en la década de 1980 algunos investigadores definieron nuevos objetivos para la Instrucción basada en computador. Se aprobó el tutor humano como el modelo educativo ideal y se pretendió aplicar técnicas de inteligencia artificial para crear un modelo "inteligente" de instrucción por ordenador (Corbett, Koedinger & Anderson, 1997).

Los Sistemas Tutores Inteligentes (ITS por sus sigas en Ingles: Intelligent Tutoring Systems) son programas informáticos que están diseñados para incorporar las técnicas propias del campo de estudio de IA a fin de proporcionar tutores computarizados que saben lo que enseñan, a quien enseñan y como enseñarlo (Nwana, 1990).

Otras definiciones de un ITS pueden ser encontradas en la literatura, por ejemplo (Self, 1999) define los ITS como "*Sistemas de Aprendizaje basados en computador que intentan adaptarse a la necesidades de los aprendices y por lo tanto son los únicos tipos de sistemas que intentan preocuparse por el aprendiz en dicho sentido*"[1]. Corbett, Koedinger & Anderson, los definen como sistemas que pretenden hacer participar a los estudiantes en una actividad de razonamiento continua e interactuar con el estudiante basándose en una comprensión profunda de su conducta (Corbett, Koedinger & Anderson, 1997). Otras definiciones similares y relevantes pueden encontrarse en (Nkambou, Bourdeau, Mizoguchi, 2010), (Bourdeau J, 2010) y (VanLehn 2006).

En Inteligencia Artificial se considera que producir un comportamiento , en un ordenador , tal como si se realizará por un ser human o, se describiría como un comportamiento "inteligente"; Así mismo , los ITS pueden ser considerados como programas que intentan ejecutar

[1]Texto original de la Cita: "ITSs are computer-based learning systems which attempt to adapt to the needs of learners and are therefore the only such systems which attempt to 'care' about learners in that sense." (Self, 1999).

comportamientos tales que, si se realizaran por un ser humano, se describirían como comportamientos propios de la "buena enseñanza" (Elsom-Cook, 1987).

El diseño y desarrollo de este tipo de tutores se encuentran en la intersección de los dominios de conocimiento de las Ciencias de la Computación, la Psicología Cognitiva y la Investigación Educativa; esta zona de intersección normalmente se le conoce como Ciencia Cognitiva (Nwana, 1990). Por razones históricas, muchas de las investigaciones en el dominio del software de educación, que incluye características propias de Inteligencia Artificial se ha realizado bajo el nombre de "ICAI", un acrónimo de "Intelligent Computer-Aided Instrucción". Esta forma de llamar dichos programas, evolucionó a partir de lo que se denomina "Instrucción Asistida por Computadora" (CAI por sus siglas en ingles: Computer Aided Instruction) que a menudo se refiere a la utilización de ordenadores para la educación. No obstante, para la mayoría de los casos, los ITS y los sistemas ICAI son sinónimos (Nwana, 1990).

Aunque algunos investigadores prefieren el termino "ICAI", este termino se ha venido remplazando por el acrónimo "ITS" (Sleeman & Brown, 1982). Este último termino, el cual se utilizará a lo largo del documento, ha ganado completo apoyo, según lo confirmado por la Conferencia Internacional sobre Sistemas Tutores Inteligentes, celebrada en Montreal, Canadá, en junio de 1988.

Esta preferencia está motivada por la afirmación que, en muchos aspectos, la importancia del cambio en la metodología de la investigación va más allá que la simple adicióndela letra "I" a la abreviatura CAI (Wenger, 1987). Sin embargo, algunos investigadores se muestran comprensiblemente dudosos respecto a utilizar el término "inteligente". De esta manera dichos investigadores suelen usar otro tipo de etiquetas para hacer referencia al mismo tipo de programas como por ejemplo: "KBTS" ('Knowledge-Based Tutoring System) o (ATS, Adaptive Tutoring System) (Streitz, 1988), Knowledge Communication Systems (Wenger, 1987). Aún así, la mayoría de los investigadores parecen sentirse satisfechos con la sigla ITS. **El uso de esta sigla, estará justificado siempre y cuando todos los involucrados tengan el conocimiento que el uso de la palabra "inteligente", estrictamente hablando es inapropiada (Nwana, 1990).**

Los resultados obtenidos en el campo de Sistemas Tutores Inteligentes están bien documentados en dos libros clásicos, (Sleeman y Brown, 1982) y (Wenger 1987) y dos libros

modernos (R. Nkambou , 2010) y (Woolf, B. 2010). Los esfuerzos de investigación durante los siguientes veinticinco años posteriores a la década de 1980, han resultado en algunos éxitos notables que pueden ser consultados en (Lesgold, Eggan, Katz y Rao, 1992; Koedinger, Anderson, Hadley y Marcos, 1995) y (Aleven, V., McLaren, B.M., Sewall, J., & Koedinger, K.R. 2009).

1.1.2 Escenarios de aplicación en educación en control.

El dominio de conocimiento del control automático es el campo específico de estudio que se pretende abordar en este trabajo. Algunas de las temáticas principales de este dominio de conocimiento son (Ogata, 1998):

a) Modelamiento de Sistemas Dinámicos
b) Análisis de la respuesta transitoria de sistemas de dinámicos
c) Acciones básicas de control y respuesta de sistemas de control
d) Controles PID
e) Diseño de sistemas de control mediante el método del lugar geométrico de las raíces.
f) Análisis y Diseño mediante la respuesta en frecuencia de sistemas de control
g) Análisis y Diseño de sistemas de control en el espacio de estados
h) Introducción al control robusto

Alrededor de estas temáticas se han presentado algunos trabajos relacionados con sistemas tutores inteligenteslos cuales se presentan a continuación:

En (Nogami , Yokoi , Yanagisawa, Mitui, 1994) se presenta un sistema tutor inteligente basado en simulación. El sistema tiene como objetivo proveer ayudas individualizadas a los estudiantes de un curso de control específicamente en la temática de Control PID. El sistema tutor inteligente guía a los estudiantes mediante la presentación de objetivos locales en el contexto de resolución de un problema de sintonización de controladores PID. El sistema utiliza dos estrategias una orientada a la ejercitación del estudiante y otra orientada a la ilustración de conceptos. En la primera estrategia se provee al estudiante con pasos intermedios que deben ser realizados en el simulador y que permiten que el estudiante obtenga el conocimiento de si la acción intermedia realizada es conducente a la resolución del problema o por el contrario

genera un error. La segunda estrategia se utiliza para ilustrar al estudiante con los conceptos correctos cuando un error recurrente se detecta. El sistema fue evaluado en un curso de estudiantes y fue comparado con grupos de control demostrando que las técnicas utilizadas son eficientes para permitir el aprendizaje individualizado.

En la conferencia europea de control que se llevo cabo en el año 2003, (Nenad Bolf, Juraj Bozicevic, Slavomir Stankov, 2003) presentan un reporte acerca del desarrollo de un sistema de entrenamiento para estudiantes de ingeniería química quienes están iniciando en el campo de control automático. El sistema de entrenamiento incluye un sistema tutor inteligente que provee las siguientes dos funcionalidades principales:

a) Medida y diagnostico del conocimiento del estudiantes,
b) determinación del conocimiento faltante en un estudiante respecto al conocimiento de referencia.

La arquitectura del sistema tutor inteligente utilizado en este sistema de entrenamiento no se asemeja a la arquitectura general de un ITS. Sin embargo utiliza técnicas conocidas para el modelamiento de las unidades de conocimientos como lo son redes semánticas, frames y reglas de producción. El sistema fue probado con estudiantes de último año resultando en incrementos en la motivación de los estudiantes gracias a las flexibilidades que el sistema les provee.

(Prekas, Rangoussi, Vassiliadis, and Prekas, 2004) presentan en la Conferencia Internacional de Procesamiento de Señales ICSP'2004, un reporte del progreso en la construcción de un sistema tutor inteligente mediante el cual los estudiantes pueden realizar experimentos completos en un laboratorio de control automático vía remota desde la intranet de la institución o desde internet. El sistema tutor inteligente gira entorno a la arquitectura general y cuenta con algunas simplificaciones como por ejemplo reducir el modelo del estudiante a un mecanismo básico de calificación y reducir el modelo del tutor a un administrador de cursos que pueda secuenciar el material de estudio. Estas simplificaciones permiten a los autores crear un prototipo rápido que pueden poner a prueba fácilmente en la plataforma de experimentación remota.

En la revista Computer Applications in Engineering Education, (Ozek, Akpolat, Orha, 2010),

presentan un reporte acerca de la construcción de un sistema tutor inteligente que funciona en un entorno web y que esta construido usando técnicas de lógica difusa tipo-2. El sistema tutor provee a los estudiantes de un curso de control de la posibilidad de acceder a servicios que les permite el aprendizaje individualizado. El aprendizaje individualizado ocurre gracias a la clasificación que se realiza de cada estudiante mediante los estilos de aprendizaje propuestos por (Felder and Silverman, 2008). Los datos que se utilizan para clasificar los estudiantes provienen del uso que hacen de la interfaz gráfica y deldesempeño que van teniendo durante el proceso de tutoría. Una vez obtenida una clasificación del estilo de aprendizaje del estudiante, el sistema tutor selecciona los tipos contenidos (Imagen, Audio, Video, etc) apropiados y los presenta al estudiante. El sistema de lógica difusa tipo-2 es utilizado para realizar la clasificación en un entorno que los autores reconocen como de alta incertidumbre.

De los antecedentes presentados en esta sección los más relevantes respecto al campo de la Inteligencia Artificial en Educación, son (Ozek, Akpolat, Orha, 2010) y (Nogami , Yokoi , Yanagisawa, Mitui, 1994). Se identifica en estos trabajos un marco de trabajo compartido con los enfoques que presentan los investigadores más relevantes en el campo de los sistemas tutores inteligentes. Sin embargo es evidente la falta de aplicación de técnicas de inteligencia artificial en la educación en control especialmente de las técnicas que han tenido un mayor impacto en el campo de los tutores inteligentes como los tutores cognitivos basados en reglas de producción y los tutores cognitivos basados en restricciones.

La federación internacional de control automático (IFAC, por sus siglas en ingles) es una reconocida organización mundial que promueve los eventos académicos relacionados con muchos aspectos del control automático de sistemas. Dentro de su estructura organizacional cuenta con un comité técnico el cual esta dedicado exclusivamente a la gestión de temas relacionados de la educación en control. Cada 3 años se organiza el Simposio Internacional de avances en educación en control, (Symposium on Advances in Control Education) que en 1988 tuvo su primera realización. No más de 10 trabajos relacionados con sistemas tutores inteligentes se han presentado en el total de 11 eventos realizados.

Sin embargo, la riqueza de los trabajos presentados y de la discusión acerca de educación en control es bastante amplia. Se encuentran reportes acerca de modelos pedagógicos, currículos, nuevas plataformas de experimentación remota y virtual, estrategias de enseñanza y retos futuros para la educación en control. Todo este conocimiento debe ser adaptado y transformado

de manera tal que pueda ser utilizado en sistemas típicos y abiertos como los tutores cognitivos y los tutores tipo example tracing.

1.2 Estructura general de los Sistemas Tutores Inteligentes

Básicamente la arquitectura general de un Sistema Tutor Inteligente (Figura 1), divide el sistema en cuatro componentes aislados: El modelo del dominio, el modelo del estudiante, el modelo del tutor y la interfaz gráfica de usuario.

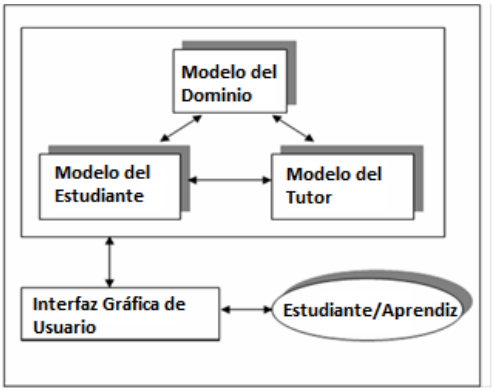


Figura 1. Arquitectura general de un sistema tutor inteligente.

El modelo del dominio almacena los conceptos, reglas y estrategias de resolución de problemas que el estudiante debe aprender. En general tiene las siguientes funcionalidades:

a) Servir como una fuente de conocimiento experto,
b) Permitir evaluar el rendimiento del estudiante
c) Permitir reconocer los errores que un estudiante comete al resolver un problema.

Para realizar la implementación de este componente, la mayoría de las tutores utilizan estructuras jerárquicas como redes semánticas, ontologías, técnicas de marcos (Frames) o reglas de producción. Sin importar cual técnica se utilice, el verdadero requerimiento para implementar este modulo es contar con la capacidad de razonamiento de manera que se puedan comprender las acciones que el estudiante esta realizando sobre la interfaz gráfica y se puedan proveer las explicaciones apropiadas de forma individualizada.

El modelo del estudiante contiene todo el conocimiento que sea necesario acerca del estado cognitivo y afectivo del estudiante. El estado cognitivo del estudiante permite identificar con precisión cuales conocimientos el estudiante ha aprendido de aquellos que debe aprender. El estado afectivo del estudiante permite conocer los estados emocionales del estudiante de manera que el Sistema Tutor Inteligente pueda adaptar su estrategia pedagógica para cambiar condiciones como la motivación, el cansancio, la frustración, etc.

El modelo del tutor es el modulo encargado de tomar las decisiones acerca de cuales estrategias de enseñanza se deben utilizar y cuales acciones se deben efectuar a través de la interfaz grafica para comunicarse con el estudiante. Este modulo se encarga de decidir si se realiza o no una intervención y en caso afirmativo decidir cual es la forma de realizar la intervención. Diferentes tipos de intervención e interacción se pueden llevar a cabo en un sistema tutor inteligente, algunos ejemplos son: diálogos Socráticos, presentación de indicios, pistas o ayudas, realimentación por parte del sistema utilizando contenidos, etc.

Para una comprensión más a fondo de cada uno de los componentes de las arquitecturas general, así como para la consulta de otras arquitecturas que han tenido éxito en el campo se sugiere revisar las siguientes referencias (Nkambou et al., 2010; Woolf, 2008; Nwana, 1990)

1.3 Tipos de Sistemas Tutores Inteligentes.

El estudio propuesto está enfocado en tres tipos de Sistemas Tutores Inteligentes que utilizan técnicas específicas para la creación de cada uno de los componentes que conforman un sistema tutor inteligente. Estos tipos de sistemas tutores han sido probados con suficiencia y han sido exitosos en diferentes dominios de conocimiento(Vanlehn, 2011).

Los dos primeros tipos (Model-Tracing y Constrainst-Based), corresponden a la categoría de sistemas tutores cognitivos pues ambos están basados en teorías que intentan modelar el entendimiento humano. Los tutores tipo Example-Tracing, corresponden al resultado obtenido tras la simplificación del proceso de construcción de los tutores Model-Tracing. En las siguientes secciones se presentan las características más importantes de cada uno de estos tutores así como los aspectos más importantes de las teorías cognitivas que dieron origen a su concepción.

1.3.1 Sistemas tutores basados en reglas de producción

- Teoría ACT-R

La teoría ACT-R del aprendizaje y resolución de problemas, es una revisión de la Teoría ACT* presentada en 1982 por John Robert Anderson en su libro Arquitectura de la cognición (Anderson, 1983). Gran parte de esta teoría trata la manera en que se realiza la adquisición de capacidades y habilidades cognitivas. La teoría sostiene que una habilidad cognitiva consiste, en gran parte, de unidades de conocimiento relacionadas entre sí por los objetivos. La adquisición de habilidades cognitivas implica la formulación de miles de reglas que relacionan los objetivos de una tarea y los estados de la tarea con las acciones y las consecuencias. La teoría utiliza un formalismo basado en reglas para representar el conocimiento orientado a objetivos. Por ejemplo, una las reglas para definir una demostración geométrica sería similar a:

Si, el objetivo es probar que dos triángulos son congruentes
Entonces, establezca como un sub-objetivo, el demostrar que los lados correspondientes son congruentes.

La teoría ACT-R tiene 3 principios básicos que están completamente relacionados con el desarrollo y diseño de sistemas tutores y son:

1. **Distinción entre lo procedimental y lo declarativo**: La teoría distingue entre el conocimiento declarativo (por ejemplo, conocer el teorema de lado-ángulo-lado) y el conocimiento procedimental (por ejemplo, la capacidad para utilizar el teorema del lado-ángulo-lado en una demostración). El supuesto de la teoría es que el conocimiento declarativo independientemente de los objetivo en un problema, puede ser adquirido de una forma codificada directamente de la observación y la instrucción. Las habilidades cognitivas dependen de la capacidad de convertir este conocimiento en reglas de producción como las mostradas anteriormente las cuales representan el conocimiento procedimental.

2. **Compilación de conocimiento**: Se supone que los estudiantes pueden utilizar diversos procedimientos de interpretación, tales como el seguimiento de instrucciones o la interpretación de analogías. Esto les permite generar comportamientos tales que puedan encontrar solución a los problemas planteados a través de relacionar el conocimiento declarativo que poseen con los

objetivos del problema. Este proceso de aprendizaje denominado compilación de conocimiento convierte las interpretaciones para la resolución de problemas en reglas de producción. Así, la teoría supone que las reglas de producción sólo se pueden aprender mediante el empleo del conocimiento declarativo en el contexto de una actividad resolución de problemas.

3. **Fortalecimiento**: Se supone que tanto el conocimiento declarativo como el procedimental adquiere fuerza con la práctica. La aplicación de conocimientos débiles puede resultar en situaciones de duda y errores. Por lo tanto, incluso después de que el conocimiento se ha adquirido con éxito, la práctica permite generar una ejecución más suave, más rápida, y con menos errores.

Estas tres hipótesis apuntan en la dirección de un método de enseñanza en la que a los estudiantes, se les presenta una breve instrucción inicial declarativa y a continuación reciben una gran cantidad de práctica guiada basada en la resolución de problemas. (Anderson; Corbett; Koedinger; Pelletier 1995).

- Tutores Cognitivos tipo Model-Tracing

Los tutores cognitivos tipo model-tracing son un tipo Sistemas Tutores Inteligentes que se basan en teorías de psicología cognitiva (Anderson; Corbett; Koedinger; Pelletier 1995, (Anderson; Boyle; Corbell; Lewis 1990) en particular la teoríaACT-R (Anderson, 1993). El desarrollo de un Tutor model-tracing implica la creación de un modelo cognitivo que modela la forma como un alumno aborda la resolución de un problema. Este modelo cognitivo se realiza escribiendo las reglas de producción que caracterizan la variedad de estrategias y conceptos erróneos que los estudiantes suelen adquirir. Las reglas de producción se escriben en forma modular de manera que se puedan aplicar a un objetivo en un determinado contexto independiente de lo que lleve al cumplimiento de ese objetivo.

Considérese el siguiente ejemplo (tomado de (Woolf, 2008)) que muestra tres reglas de producción en el dominio de la resolución de ecuaciones algebraicas:

Estrategia 1: ***Si***, el objetivo es resolver la ecuación A*(B*X + C) = D

Entonces, reescribir la ecuación en la forma (B*X + C) = D/A

Estrategia 2: ***Si***, el objetivo es resolver la ecuación A*(B*X + C) = D

Entonces, reescribir la ecuación en la forma A*B*X + A*C = D

Concepto Errado: ***Si***, el objetivo es resolver la ecuación A*(B*X + C) = D

Entonces, reescribir la ecuación en la forma A*B*X + C = D

Las dos primeras reglas de producción ilustran estrategias correctas para el mismo objetivo. Al representar diferentes estrategias, el tutor cognitivo puede seguir a los estudiantes a través de diferentes caminos para la solución de un problema. La tercera regla de producción, es conocida como una regla “Buggy” (Del Ingles Bug: Defecto, falla) y representa un error común que los estudiantes cometen cuando hacen frente a un problema.

Dos técnicas son utilizadas por este tipo de tutores para proveer individualización de las ayudas a lo largo de un problema y a lo largo de un conjunto de problemas: Model Tracing y Knowledge Tracing.

La técnica de Model Tracing es usada para monitorear el progreso del estudiante a través de la solución del problema (Anderson, Boyle, Corbell, Lewis 1990). Este seguimiento es realizando por el tutor, de manera silenciosa, tratando de emparejar las acciones del estudiante con aquellas que pueden ser generadas por el modelo. Cuando el estudiante necesita ayuda, el tutor conoce la ubicación del estudiante dentro del problema y puede proveer mensajes de ayuda, que son individuales para cada estudiante según la manera en que el estudiante se encuentre abordando el problema.

La técnica de Knowledge Tracing es usada para monitorear el aprendizaje del estudiante de problema a problema (Corbet, Anderson, 1992). Un procedimiento de estimación Bayesiana identifica las fortaleces y debilidades del estudiante relativas a las reglas de producción del modelo cognitivo. Esta información de valoración es usada para individualizar la selección de los problemas y optimizar el ritmo de los estudiantes a través del currículo.

Sintetizando, se puede decir que los tutores model tracing tienen la característica de tener un modelo psicológico del proceso cognitivo que ocurre en un estudiante con buen rendimiento (o

con un rendimiento cercano al rendimiento deseado) incluyendo las diferentes estrategias que se pueden abordar para la solución de un problema así como los errores comunes que se suelen cometer. Estas tres características le brindan a este tipo de tutores una gran flexibilidad y un comportamiento cercano al de un tutor experto lo cual han sido uno de las grandes razones por las cuales han sido tan exitosos y ampliamente estudiados.

1.3.2 Sistemas Tutores basados en restricciones

Los sistemas tutores inteligentes basados en restricciones, son un tipo de tutores cognitivos cuyas raíces teóricas están basadas en una teoría psicológica del aprendizaje que fue propuesta por Stellan Ohlsson en 1992 (Ohlsson 1992). Esta teoría, de la misma forma que la teoría ACT-R, también reconoce la existencia de un conocimiento procedimental y uno declarativo. Se reconoce también que el conocimiento procedimental se refuerza con la practica y que resulta de una transformación del conocimiento declarativo que ocurre cuando el estudiante se enfrenta a un problema. Esta teoría adicionalmente reconoce que el conocimiento procedimental esta relacionado con la ejecución de acciones mientras que el conocimiento declarativo cumple una función evaluativa, permitiendo al estudiante evaluar las consecuencias de sus acciones.

Ohlsson propone que el aprendizaje del conocimiento procedimental y/o declarativo ocurre primordialmente cuando los estudiantes se dan cuenta por ellos mismos (o por un tercero) que están cometiendo errores. Los estudiantes en ocasiones cometen errores incluso bajo el correcto conocimiento de lo que deben hacer. Esto significa que aunque el estudiante puede tener todo el conocimiento declarativo necesario para realizar una tarea, dicho conocimiento no es suficiente y el estudiante debe aprender a aplicar el conocimiento declarativo para poder lograr el manejo de un dominio de conocimiento específico.

El proceso de aprendizaje se describe en dos etapas: Una de reconocimiento del error y otra de corrección del error. En la primera etapa el estudiante detecta por el mismo que una acción fue incorrecta. Este es el caso cuando el conocimiento declarativo que posee el estudiante es correcto y le permite evaluar correctamente las consecuencias de su acción incorrecta. Cuando el conocimiento declarativo del estudiante es incorrecto, el estudiante no puede evaluar sus acciones o su evaluación es incorrecta y por lo tanto se requiere de la intervención de un

tercero, el tutor. El tutor, que puede ser humano o artificial, permite que el estudiante note su error y adicionalmente provee ayuda para corregir el error. Cuando esta ayuda se provee, inicia la segunda etapa. La ayuda suele conformarse de dos partes: Una identificación de la parte incorrecta en la acción o solución del estudiante y una descripción del principio que se ha violado, dentro del dominio de conocimiento, con la parte incorrecta de la solución.

La teoría de Ohlsson establece que el conocimiento declarativo se puede representar en forma de restricciones que modelan principios fundamentales del dominio de conocimiento que no pueden ser violadas. Esto corresponde al principio básico de los tutores basados en restricciones: Todas las soluciones correctas, independientemente del problema, comparten la característica de no violar ningún principio fundamental del dominio.

De esta forma la teoría de Ohlsson propone un método de tutoría que consiste en, dado un conjunto de problemas, definir un conjunto de restricciones que todas las soluciones correctas deben cumplir satisfactoriamente. Dada una solución se ejecuta una validación para confirmar que la solución cumple con todas las restricciones. En caso de cumplimiento total, la solución se clasifica como correcta. En caso de encontrar al menos una restricción que no se cumpla, la solución se clasifica como incorrecta. Las restricciones violadas identifican el principio del dominio que no ha sido satisfecho y por lo tanto permiten definir de manera directa un mensaje de ayuda para el estudiante, el cual normalmente, debería informarle al estudiante el principio que no se ha satisfecho como mecanismo de refuerzo del conocimiento declarativo.

Este modelo de tutoría asume que para poder diagnosticar errores, no se requiere conocer la secuencia exacta de acciones que el estudiante realizó para llegar a un determinado estado de la solución. Solo se requiere evaluar el estado de la solución respecto al conjunto de principios básicos que deben ser cumplidos.

Para darle flexibilidad al modelo, el concepto de restricción fue definido de manera muy conveniente como una pareja de condiciones (C_r,C_s). $C_r$se conoce como la condición de relevancia. C_s se conoce como la condición de satisfacción. La condición de relevancia corresponde a una condición que tiene que ser verdadera para considerar que la evaluación de la restricción tiene sentido, es decir, es relevante para el estado de la solución. La condición de satisfacción es la condición que permite evaluar si o no el estado actual de la solución cumple con los principios del dominio que debería cumplir.

Por lo tanto una restricción puede verse de la siguiente forma:

SI <Condición de relevancia> es verdadera
ENTONCES <Condición de satisfacción> DEBE ser verdadera
DE LO CONTRARIO la solución no es correcta

Aunque la restricción mostrada tiene una forma muy similar a una regla SI-ENTONES, caso típico de las reglas de producción, la restricción no representa una implicación lógica. Las dos partes de la restricción están ligadas a través del conector “DEBE”, significando esto que en el caso de ser cierta la condición de relevancia, DEBE ser cierta la condición de satisfacción para considerar que la solución es correcta. Una regla de producción, por el contrario, propone una acción a realizar (la parte ENTONCES) si se cumplen el objetivo y la situación indicada en la parte SI. Por el contrario, una restricción especifica las condiciones para que un estado de solución que sea correcto. Las reglas de producción son de carácter generativo, mientras que las restricciones son evaluativo, y se pueden utilizar para hacer juicio (Ohlsson y Mitrovic 2007).

Un conjunto de restricciones constituye de forma explicita,las características de las soluciones correctas. Si una solución dada por un estudiante, o alguno de los estados intermedios, viola una o más restricciones, se considera que la solución es incorrecta o que el estado intermedio es incorrecto; por lo tanto, el conjunto de restricciones modela los errores de forma indirecta, sin tener que enumerarlos de forma exhaustiva. De esta forma, un tutor basado en restricciones le permite a un estudiante explorar libremente el espacio de posibles soluciones; cualquier estrategia que este disponible para resolver un problema, es correcta siempre y cuando no se viole ninguna de las restricciones definidas.
Si para un problema, existenestrategias correctas de solución que no han sido consideradas por los diseñadores del sistema, estas estrategias serán igualmente aceptadas por el sistema, ya que al ser estrategias correctas, se supone no violan las restricciones de dominio.

La idea original del modelamiento basado en restricciones, se presento en (Ohlsson 1992). Este modelopresenta las restricciones como un mecanismo para representar conocimiento únicamente a través de sintaxis.Es decir a través de un conjunto de reglas que definen las secuencias correctas de los elementos que representan la solución correcta. Un ejemplo de este tipo de restricciones y quefue presentado en el documento de 1992 es el siguiente:

SIse requiere calcular a/b + c/d, y la solución del estudiante es (a+c)/n,
ENTONCES debe ser cierto que n=b=d

La restricción es relevante para la situación en la que el estudiante suma dos fracciones mediante la adición de los numeradores; esto sólo se permite cuando los denominadores de las dos fracciones y el denominador de la fracción resultante son iguales. A este tipo de restricciones se les conoce como *restricciones de sintaxis*. Estas restricciones permiten que un ITS identifique errores en las soluciones de los estudiantes que violan los principios generales de dominio.

A menudo las soluciones de los estudiantes son sintácticamente correctas, sin embargo las soluciones no son correctas en el contexto del problema que se intenta resolver. Es por eso que se hace necesario en algunos casos revisar que la solución sea correcta semánticamente. Es decir, no solo que la secuencia de símbolos que representa la respuesta sea correcta, sino que adicionalmente la respuesta sea correcta respecto a las posibles soluciones del problema. (Mitrovic 1998a, 1998b, 1998c; Mitrovic and Ohlsson 1999).

La semántica de un problema particular se capturado utilizando una solución ideal; por lo tanto una restricción semántica compara la solución del estudianterespecto a la solución ideal del problema. De esta forma, para la construcción de restricciones semánticas el autor del sistema especifica una solución ideal para cada problema. Esta solución ideal es cuidadosamente seleccionada para ilustrar algunas de las características importantes del dominio de interés.

El siguiente ejemplo permite comprender el concepto de este tipo de restricciones. El dominio de interés en este ejemplo es el lenguaje SQL. En general si una consulta SQl utiliza mas de una tabla en la clausula FROM, la consulta requiere de una condición que permita unir las tablas (join). Las condiciones para unir tablas se pueden especificar de dos formas: en la clausula FROM usando la palabra clave JOIN ó en la clausula WHERE usando los nombres de las tablas, de los campos, el operador AND y el signo =. La restricción mostrada en la figura, es relevante cuando la solución ideal especificada por el autor, tiene la palabra clave JOIN en la clausula FROM y al mismo tiempo, la solución del estudiante no tiene clausula WHERE pero tiene mas de dos tablas en la clausula FROM. Si ocurre este caso, la condición de satisfacción pone como requisito que la palabra JOIN se utilice en la respuesta del estudiante. La condición

de satisfacción no hace un chequeo de la sintaxis completa para la condición JOIN. Este chequeo es responsabilidad de una restricción de sintaxis.

Constraint 207:
Cr: the WHERE clause is empty in both the student's and ideal solutions
AND there is more tan one table in the student´s FROM clause
AND the FROM clause of the ideal solution contains the JOIN keyword
Cs: the JOIN keyword must appear in the student's FROM clause

1.3.3 Sistemas tutores tipo example-tracing

Un tutor tipo Example Tracing, o también llamado Pseudo-Tutor, es un sistema educativo que emula comportamiento de un Tutor Inteligente, pero lo hace sin usar código de programación de técnicas de Inteligencia Artificial. (Sería más exacto, aunque más confuso, llamar a estos tutores "Psudo Tutores Inteligentes" para enfatizar que es la falta de un motor interno de Inteligencia Artificial que los hace "pseudo" y no la falta de un comportamiento inteligente) (Koedinger, K., Aleven, V., Heffernan, N., McLaren, B. M., and Hockenberry, M, 2004).

Dos características clave existentes en los tutores cognitivos así como en otros tipos de Sistemas Tutores Inteligentes, son:

1) Ayudar a los estudiantes en la construcción de conocimiento a través de la realimentación y la instrucción en el contexto de la actividad que se está realizando.
2) Proveer a los estudiantes la flexibilidad necesaria para explorar diferentes estrategias que representen alternativas de solución a un problema, lo cual les permitirá aprender haciendo (Learning by Doing).

Los Pseudo-Tutores pueden proporcionar estas características, pero con algunas limitaciones, sin embargo, hay una ganancia en términos del tiempo requerido para realizar el desarrollo de estos sistemas. Los Pseudo-Tutores fueron presentados por (Koedinger, K., Aleven, V., Heffernan, N., McLaren, B. M., and Hockenberry, M, 2004) en la séptima conferencia Internacional sobre Sistemas Tutores Inteligentes. Con el paso del tiempo, los autores han preferido usar el término “Example-Tracing Tutors” para designar a este tipo de tutores. Este

nuevo nombre es más apropiado dado que presenta mayor significado respecto al funcionamiento del tutor. En los tutores cognitivos, la técnica “model tracing” permite hacer un seguimiento del modelo cognitivo del estudiante a través del problema. Para la creación de los Pseudo-Tutores no se crea ningún tipo de modelo cognitivo, en cambio se demuestran al tutor ejemplos de comportamiento correcto, de comportamiento subóptimo y de comportamiento incorrecto. Estos ejemplos son utilizados posteriormente por el tutor para hacer seguimiento al estudiante dentro del problema. Esta es la principal razón para el establecimiento del nuevo nombre para dichos tutores (Example-Tracing Tutors), pues a partir de los ejemplos demostrados (Example) los tutores hacen seguimiento del desarrollo del problema (Tracing).

La idea principal detrás de los Tutores Tipo Example-Tracing es sencilla. Así como los tutores que implementan la técnica “model-tracing” trabajan haciendo un seguimiento e interpretando el comportamiento del estudiante respecto a un modelo cognitivo, los tutores tipo Example-Tracing realizan su trabajo mediante la interpretación del comportamiento de los estudiantes en referencia a ejemplos específicos del comportamiento que se debe tener cuando se realizan actividades de resolución de problemas. Los tutores tipo Example-Tracing fueron originalmente concebidos como una herramienta para hacer tareas de análisis cognitivo y para el prototipado rápido de Tutores Cognitivos. Sin embargo, un número de extensiones a la idea básica la han convertido en una metodología viable para la construcción tutores. En particular, se han añadido una serie de técnicas que otorgan flexibilidad para comparar el comportamiento del estudiante contra los ejemplos de referencia, de modo que un tutor puede reconocer una amplia gama de comportamientos correctos y no sólo los ejemplos demostrados como tal (Aleven, V., McLaren, B.M., Sewall, J., & Koedinger, K.R. 2009).

Los Tutores tipo Example Tracing interpretan los pasos de solución que realiza un estudiante respecto a un gráfico de solución predefinido para un problema determinado, el cual, siguiendo las definiciones de Newell y Simon (1972), es llamado el “grafo de comportamiento" o “Behaviour Graph”. Un gráfico de comportamiento es un grafo dirigido y a -cíclico que representa las distintas maneras aceptables de resolver un problema. Los enlaces del grafo representan las acciones de resolución del problema y los nodos representan los estados del problema a través de la solución . Un gráfico de comportamiento puede contener varias rutas , correspondientes a las diferentes formas de resolver un problema. También puede contener enlaces que representan un comportamiento incorrecto. Un ejemplo de un gráfico de comportamiento, para la interfaz anteriormente mostrada, puede observarse en la siguiente figura.

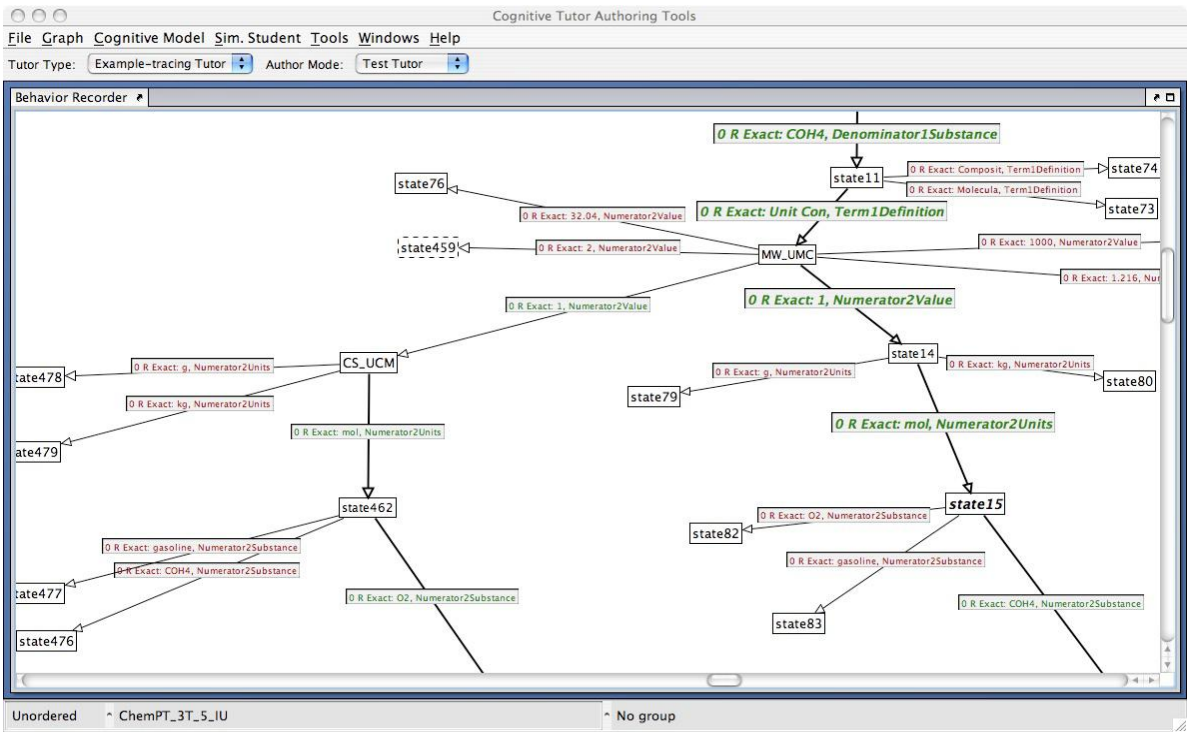


Figura 1. Ejemplo de un gráfico de Comportamiento para el Tutor de Estequiometria. Tomada de (Aleven, V., McLaren, B.M., Sewall, J., & Koedinger, K.R. 2009)

En esta figura, los enlaces del grafo representan las acciones que el estudiante puede realizar para resolver el problema y los nodos representan los estados del proceso de resolución del problema. Los enlaces que tienen etiquetas verdes corresponden a acciones correctas en el proceso de solución del problema. Los enlaces que tienen etiquetas rojas corresponden a acciones incorrectas que un estudiante puede cometer en la interfaz y que representan la apropiación de un concepto incorrecto.

Para la creación de un Tutor tipo Example-Tracing debe disponerse entonces de una interfaz de usuario que presente los posibles datos que un estudiante puede manejar en la solución de un problema. La interfaz también debe permitir la ejecución de acciones que representen pasos lógicos consecuentes a la obtención de la solución. Una vez se ha creado y diseñado la interfaz, el autor debe crear el grafico de comportamiento, demostrando, o mejor dicho, ejecutando los pasos para la correcta solución del problema. Una vez creado el grafico de comportamiento, el tutor es presentado al estudiante, y se inicia el proceso de comparación de los pasos que el estudiante realiza con los que el autor del tutor ha demostrado. Este proceso es conocido como “Seguimiento de Ejemplos” ó “Example Tracing” y es ejecutado por un componente de software el cual se le llamará “Example Tracer”

- Tutores tipo Example Trancing como ITS

De la misma forma en que muchos otros Sistemas Tutores Inteligentes, los tutores tipo Example-Tracing ayudan a los estudiantes a adquirir habilidades cognitivas complejas a través de la práctica guiada[2]. En esta sección, se establece que los tutores tipo Example Tracing son capaces de presentar, un gran subconjunto de los comportamientos catalogados en el artículo "The Behavior of Tutoring Systems", de Kurt VanLehn, (VanLehn 2006). Este sub-conjunto de comportamientos es el mismo ya cubierto por los Tutores Cognitivos. Sin embargo, según (Aleven, V., McLaren, B.M., Sewall, J., & Koedinger, K.R. 2009), esto no quiere decir que no hay diferencias de comportamiento entre los tutores tipo Example-Tracing y los Tutores Cognitivos.

En última instancia, los tutores cognitivos son más flexibles y más robustos. Aun así, esta flexibilidad no siempre es necesaria para todos los entornos y para todos los contextos, por lo cual se pone en consideración los gastos en que se incurren cuando se crea un modelo cognitivo basado en reglas, el cual es alto.

Como es típico de los Sistemas Tutores Inteligentes, los tutores tipo Example Tracing son construidos de tal forma que puedan proporcionar una interfaz que hace el pensamiento visible (Anderson et al., 1995), en el sentido en que el diseño visual de la interfaz gráfica del tutor hace visible los pasos intermedios en la solución de un problema.

Además de una interfaz que hace el pensamiento visible, los tutores Tipo Example Tracing proporcionan varios tipos de orientación que son típicos de los ITS, estos tipos de comportamientos han sido clasificados por (VanLehn 2006) como comportamientos de lazo interno (inner loop) y de lazo externo (outer loop) ver la Tabla 1 y Tabla 2.

Los comportamientos de lazo externo se refieren al mecanismo de selección de problemas apropiados a resolver para un estudiante. El lazo interno clasifica los comportamientos por medio de los cuales el tutor proporciona apoyo a un estudiante dentro de un solo problema, incluyendo comportamientos como la orientación paso a paso, mientras se resuelve el problema y la reflexión y revisión de la solución al final del problema.

[2] Una noción más amplia de ITS puede obtenerse de (duBoulay, 2006)

Tabla 1. Grado de Cumplimiento de Comportamientos de Lazo Interno por los Tutores Tipo Example-Tracing

Comportamientos de Lazo Interno (dentro del problema de orientación)	Grado de Cumplimiento de los Tutores Tipo Example Tracing
Mínima información sobre los pasos - clasificado como correcto, incorrecta, o debajo del nivel óptimo	Soportado
Retroalimentación inmediata	Soportado
Retroalimentación atrasada	Soportado de manera Limitada
Retroalimentación por demanda	No Soportado
Retroalimentación de errores específicos	Soportado
Consejos para el siguiente paso	Soportado
Evaluación de los conocimientos	Soportado
Revisión de la solución al final del problema	No soportado

Tabla 2. Grado de Cumplimiento de Comportamientos de Lazo Externo por los Tutores Tipo Example-Tracing

Outer loop (problem selection options)	Grado de Cumplimiento de los Tutores Tipo Example Tracing
Estudiante puede seleccionar Problemas	No soportado
Secuencia Fija de Problemas	Soportado

Aprendizaje	No Soportado
Macroadaptación	Será Soportado en el Futuro

Según (VanLehn 2006), la existencia de un lazo interno es lo que diferencia a los Sistemas Tutores Inteligentes respecto a toras formas de Instrucción por ordenador. Sin embargo, en (Aleven, V., McLaren, B.M., Sewall, J., & Koedinger, K.R. 2009) se considera que esta definición es excesivamente amplia, ya que abarca formas muy simples de enseñanza. Aun así se propone que los tutores Tipo Example-Tracing sean considerados ITS no sólo porque cuentan con comportamientos propios de un lazo interno, sino debido a la flexibilidad con la que pueden evaluar el comportamiento del estudiante en el bucle interno, es decir, son capaces de proporcionar orientaciones en relación a múltiples estrategias para resolver un problema dado, independientemente de que la estrategia que el estudiante decida tomar, además, son capaces de recrear múltiples interpretaciones del comportamiento de los estudiantes, cuando una actuación puede ser interpretada de múltiples maneras.

2. Fundamentos de Control: Temáticas, Objetivos de aprendizaje y problemas.

En este capítulo se realiza un recorrido de las temáticas que típicamente son abordadas por los textos y las cátedras de introducción a los sistemas de control. La presentación que se hace en este capítulo no es detalla, sin embargo en el Anexo D se encuentrauna descripción general de los conceptos involucrados así como de las expresiones matemáticas más importantes para cada temática y los procedimientos que tienen mayor relevancia dentro de cada tema.

Para cada temática, en la sección 2.2, se identifican los objetivos de aprendizaje que los estudiantes deben alcanzar y se realiza una identificación de posibles problemas a través de los cuales se podría transformar el conocimiento declarativo en conocimiento procedimental. Cada uno de estos problemas esta identificado de la forma mas granular posible para permitir que se puedan generar entornos de resolución de problemas de complejidad muy simple y que también, por medio de composición se puedan generar entornos de resolución de problemas complejos.

La descripción general de las temáticas realizadas en la sección 2.1 tiene como propósito ayudar al lector a reconocer cuales son los conceptos usualmente involucrados y a identificar la naturaleza propia del dominio de conocimiento de interés, esto con el objetivo de permitir relacionar fácilmente las temáticas con los objetivos de aprendizaje y los problemas propuestos. Para una compresión profunda y estricta de los detalles incluidos en cada temática, se sugiere acceder a las referencias y a los textos guía que son usados en las cátedras presenciales.

El análisis realizado sobre las temáticas corresponde a las primeras etapas que deben ser ejecutadas durante el desarrollo de los sistemas tutores inteligentes y corresponde a una etapa que se intersecta con la teoría de diseño instruccional y análisis cognitivo de tareas (Clark, Feldon, Merrienboer, Yates, & Early, 2008).

La actividad de análisis cognitivo de tareas es una actividad que permite entender cuales son los diferentes niveles de dificultad que existe en la temática de interés. Permite también identificar cuales son las porciones de conocimiento declarativo que hacen parte del dominio de conocimiento y también identificar las actividades por medio de las cuales este conocimiento

declarativo puede ser transformado en conocimiento procedimental. Otra de las ventajas de esta actividad es que permite identificar de forma exhaustiva las diferentes estrategias que están disponibles para resolver un determinado problema. Este resultado es de vital importancia para la construcción de los sistemas tutores inteligentes pues le entrega al autor del sistema el conocimiento necesario para implementar el sistema de manera tal que los estudiantes, durante su proceso de interacción con la interfaz gráfica, puedan ejecutar diferentes estrategias de resolución de problema.

Debido a la gran cantidad de problemas y actividades que resultan del análisis granular de los conceptos involucrados en el dominio de conocimiento, al final del capítulo se presenta con detalle el procedimiento por medio del cual se seleccionó un conjunto reducido de problemas para la implementación en los diferentes sistemas tutores inteligentes. Esta selección es totalmente necesaria pues la implementación de todo el conjunto de problemas identificados no resulta viable de acuerdo a los alcances de este trabajo. Sin embargo el proceso de identificación exhaustivo permite que, para futuros trabajos, se pueda ampliar con facilidad la base de datos de problemas de cada uno de los tutores implementados.

2.1 Temáticas abordadas

En esta sección se presenta el listado de las temáticas que fueron abordadas durante la realización de este trabajo. De la totalidad de 10 temáticas que son presentadas en un curso introductorio de control se abordaron 7. Las siete temáticas que fueron tenidas en cuenta tuvieron una prioridad ante las que no fueron tenidas en cuenta debido a que resultan de carácter más fundamental pues forman los fundamentos básicos del dominio de conocimiento de interés.

- Tratamiento matemático de sistemas LTI usando transformada de Laplace
- Paradigmas en Control
- Desempeño del Sistema
- Análisis de desempeño de sistemas de primer orden
- Análisis de desempeño de sistemas de segundo orden
- Análisis del Error en estado estacionario

En el anexo E de este documento se presenta con detalle la mayor parte de los conceptos, expresiones y procedimientos que se esperan que los estudiantes aprendan durante la experiencia de aprendizaje en el curso de fundamentos de control automático.

Se sugiere a los lectores que quieran tener un acercamiento con la teoría de control, realizar una revisión del anexo E, con el objetivo de comprender la naturaleza matemática, gráfica y conceptual del campo de estudio de interés. A partir del entendimiento de esta teoría y de la familiarizacióncon la terminología utilizada, se podrá abordar los siguientes capítulos los cuales aplican tanto el conocimiento de la teoría de control como los conocimientos de las técnicas de modelamiento en sistemas tutores inteligentes, para realizar el análisis comparativo propuesto en este trabajo.

2.2 Identificación de los objetivos de aprendizaje y de los problemas para el fortalecimiento del conocimiento procedimental.

El proceso de creación de un sistema tutor inteligente va mas allá que el simple uso de las herramientas de autor. Por ejemplo, una metodología típica para la creación de sistemas tutores tipo example-tracing, inicia con la etapa de la definición de los objetivos de aprendizaje y la ejecución de diferentes metodologías que guían otras etapas del proceso de construcción(Baker, Corbett, & Koedinger, n.d.; Clark et al., 2008; Koedinger & Corbett, 2006)

En este trabajo se ha seleccionado la metodología de Análisis Cognitivo Racional (CTA por sus siglas en ingles Cognitive Task Analysis) para ayudar a descubrir como los problemas deben ser resueltos por los estudiantes y como en realidad están siendo resueltos por los estudiantes (Lovett, 1998).Por lo tanto, el análisis también ha incluido varias de las experiencias que se tienen con los estudiantes en el aula de clase como mecanismo para ayudar a establecer esta diferencia. Aunque el análisis realizado es completamente teórico y no se ejecutaron otras metodologías existentes que son de carácter empírico, los resultados son útiles para la ejecución de las etapas de diseño de los diferentes tipos de tutores considerados.

En esta sección se presenta una descripción de los objetivos de aprendizaje que los estudiantes deben alcanzar de acuerdo con las temáticas presentadas en la sección 2.1. También se presenta un conjunto de posibles problemas a través de los cuales se considera que los estudiantes pueden transformar el conocimiento declarativo (aquel que se imparte en las cátedras y en los textos) en conocimiento procedimental.

Los problemas presentados en esta sección cuentan con un nivel de granularidad bajo y también alto. Los problemas con granularidad baja son aquellos problemas en los que solo se pone en práctica un solo concepto o procedimiento. Los problemas con granularidad alta son aquellos problemas en los que se requiere por parte del estudiante un mayor entendimiento de los sub-objetivos que deben ser alcanzados en cada tarea.

Los problemas con granularidad alta resultan de establecer problemas con mayor complejidad y en los cuales los datos iniciales del problema requieren varios pasos de transformación, análisis y cálculo intermedio como requisito para llegar a la solución final.

Los problemas se definen como un conjunto de datos que son inicialmente dados al estudiante, un conjunto de datos que se requieren sean calculados por el estudiante y que son también proporcionados al tutor como mecanismo de verificación de las acciones del estudiante.

Una característica de los problemas presentados en la siguiente sección es la variación de sus diferentes parámetros de entrada como mecanismo de variación de la dificultad de cada problema y para la generación de un conjunto más grande de problemas. Dado que un problema tiene varios datos de entrada y varios datos de salida, para ciertos problemas se pueden asignar valores diferentes a los parámetros de entrada con el objetivo de generar un nuevo problema en el cual la estrategia de resolución es diferenteo la complejidad es diferente.

2.2.1 Tratamiento matemático de sistemas LTI usando transformada de Laplace

- Objetivos de Aprendizaje

OB.1. Aplicar apropiadamente la transformada de Laplace y su inversa como herramienta matemática para el análisis de sistemas de control LTI

OB.2 Representar sistemas dinámicos bajo modelos de Ecuación Diferencial, Función de transferencia y Espacio de Estado.

2.2.2 Paradigmas en Control

- Objetivos de Aprendizaje

OB.3. Conceptualizar los paradigmas en control: respuesta, identificación y control orientados al análisis de desempeño de sistemas de control.

2.2.3 Desempeño del Sistema

- Objetivos de Aprendizaje

OB.4. Comprender los elementos que forman parte del el estudio del desempeño del sistema: respuesta transitoria respuesta en estado estacionario del sistema, análisis de estabilidad y ciertas medidas de la respuesta en frecuencia del sistema.

OB.5. Diferenciar correctamente la respuesta transitoria y la respuesta en estado estable y reconocer las características básicas de cada una de las respuestas.

2.2.4 Análisis de desempeño de sistemas de primer orden

- Objetivos de Aprendizaje

OB.6. Comprender el proceso general mediante el cual se realiza el cálculo de la respuesta temporal para un sistema de primer orden.

OB.7 Aplicar adecuadamente la transformada de Laplace y su inversa como herramienta para el cálculo de la respuesta temporal de un sistema de primer orden ante diferentes entradas de prueba.

2.2.5 Análisis de desempeño de sistemas de segundo orden

- Objetivos de Aprendizaje

OB.8. Reconocer la parametrización típica de un sistema de segundo orden y las razones por las cuales la parametrización típica resulta apropiada y conveniente.

OB.9 Reconocer la ecuación característica para una función de transferencia de segundo orden, estar en capacidad de calcular sus raíces y comprender su significado en términos de comportamiento del sistema.

OB.10 Identificar los diferentes comportamientos que puede tener un sistema dependiendo de la variación de sus parámetros e identificar con exactitud los valores para los cuales los comportamientos presentan cambios.

OB.11 Comprender las medidas estándar de comportamiento, su relación con los parámetros del sistema y la forma en que dichas medidas proporcionan información acerca de los parámetros del sistema.

2.2.6 Análisis del error en estado estacionario.

- Objetivos de Aprendizaje

OB.12. Reconocer el error en estado estacionario como una de las características principales de rendimiento de un sistema de control

OB.13. Estar en la capacidad de calcular expresiones matemáticas usando como mecanismo el algebra de bloques, de forma que se facilite el análisis del error en estado estable para un sistema

OB.14. Aplicar el teorema del valor final de la transformada de Laplace correctamente para la realización de cálculos del error de estado estable a partir de expresiones que dependan de los componentes del sistema y de las entradas aplicadas.

OB.15. Comprender las diferentes constantes de error definidas para las distintas señales de entrada y comprender la relación que estas constantes tienen con el cálculo del error de estado estable y los diferentes tipos de sistemas.

2.3 Selección de los problemas para implementación.

Como resultado del análisis granular de los conceptos involucrados en el dominio de conocimiento, una gran cantidad de problemas y actividades fueron identificadas.La implementación de todo el conjunto de problemas identificados no resulta viable puesto que los esfuerzos necesarios resultarían más allá del propósito y alcance de este trabajo. Se requiere entonces seleccionar un conjunto reducido de problemas para la implementación en los diferentes sistemas tutores inteligentes.

En esta sección se presenta el procedimiento mediante el cual se realizóla selección de los problemas. Este proceso de selección esta basado en un escalafón creado a partir de la evaluación de tres criterios los cuales comprenden recomendaciones bien documentadas que varios investigadores han realizado al estudiar los sistemas tutores inteligentes cognitivos.

El proceso de identificación exhaustivo permite que, para futuros trabajos, se pueda ampliar con facilidad la base de datos de problemas de cada uno de los tutores implementados.

2.3.1 Sistema de selección de problemas.

Durante su trabajo con sistemas tutores cognitivos, (Koedinger & Corbett, 2006)definieron seis principios de diseños para sistemas tutores inteligentes. Estos principios de diseño son los siguientes:

1. Representar las competencias del estudiante como un conjunto de reglas de producción.

2. Proveer instrucción en el contexto de un entorno de resolución de problemas

3. Comunicar la estructura de objetivos mediante la cual se resuelve el problema.

4. Promover un entendimiento general y correcto del conocimiento requerido para resolver problemas.

5. Reducir la memoria de trabajo requerida por los conceptos y actividades de aprendizaje que son externas o que no están directamente relacionadas con el dominio de interés.

6. Proveer realimentación inmediata en los errores que son relativos al modelo del rendimiento deseado del estudiante.

Aunque estos principios fueron establecidos para sistemas tutores cognitivos basados en reglas de producción, algunos de ellos tienen un alcance más general y pueden ser aplicados a otros tipos de tutores. Aun así, los tipos de sistemas tutores inteligentes considerados en este trabajo tienen algún grado de cercanía con los tutores investigados por Anderson y Koedinger. Los tutores basados en restricciones se encuentran también clasificados entre la categoría de tutores cognitivos, pues realizan un modelamiento del proceso cognitivo que debe ocurrir en el estudiante cuando éste tenga un buen rendimiento. Los tutores tipo example-tracing tienen un comportamiento altamente similar a los tutores basados cognitivos basados en reglas (este concepto se desarrollará con mayor precisión en el capítulo 3)y esta cercanía permite que varios de los principios de desarrollo para tutores basados en reglas también apliquen para los tutores tipo example-tracing.

Al realizar una revisión de los problemas identificados para cada temática, se puede notar que algunos problemas promueven en mayor grado ciertos de los principios que otros. Un grupo de problemas debido al mayor número de conceptos que involucran, causan que al implementarlos una mayor cantidad de instrucción en forma de realimentación y ayuda pueda proveerse al estudiante. Algunos problemas están mas cercanos a los problemas de la vida real y por lo tanto promueven en mayor medida la transferencia de conocimiento. Otros problemas por su

simplicidad y por su directa relación con el dominio del control promueven mejor que otros problemas la reducción de la memoria de trabajo y las actividades y procedimientos externos que tengan que realizarse para su solución. Los problemas más complejos suelen tener una estructura de sub-objetivos más grande y normalmente promueven mejor el entendimiento general de los conceptos involucrados.

Según lo anterior, si se realiza una valoración que evalúe qué tanto promueve un problema cierto principio y luego se realiza la valoración para todos los problemas y para todos los principios, entonces se podrá tener una forma objetiva de seleccionar los problemas mas relevantes para implementación a partir de seleccionar aquellos problemas que resulten con las valoraciones mas altas.

Algunos otros principios no tienen la generalidad suficiente para ser tenidos en cuenta en el sistema de valoración, por ejemplo los principios,uno y seis, no han sido tenidos en cuenta. La razón para omitir el principio número uno, radica en que la resolución de un determinado problema puede ser modelado mediante reglas de producción, como es sugerido por el principio, pero también puede ser modelado mediante restricciones o incluso demostraciones de comportamientos correctos. Es por esto que existen diferentes tipos de sistemas tutores cognitivos y la razón por la cual comparar este tipo de técnicas para ciertos dominios de conocimiento resulta útil, pues se puede encontrar diferencias criticas en los procesos de modelamiento. De está forma los problemas en si no promueven una técnica de modelamiento sino que podrían resultar mas fáciles de modelar usando una técnica u otra. Por esta razón no se ha incluido este principio en el sistema de valoración.

De forma similar ocurre con el principio número seis. La realimentación inmediata es solo una característica que un sistema tutor inteligente puede presentar. Sin embargo y en especifico para los sistemas tutores basados en restricciones este no es el caso. Aunque los sistemas basados en restricciones pueden soportar realimentación inmediata, el principio fundamental esta orientado a determinar cuales son las restricciones que se violan en la resolución del problema y proveer realimentación basada en las restricciones violadas, esto significa, una realimentación orientada al problema y no al paso o acción especifica e inmediata que ejecutó el estudiante.

Tres medidas fueron creadas para formalizar el sistema de valoración. Estas medidas son relevancia, Estructura de objetivos y generalidad.

La relevancia es una medida que tiene un valor alto para los problemas que a partir de la experiencia se reconocen como de alta importancia para el dominio. Esta medida tiene que ver con la transferencia de conocimiento, puesto que los problemas mas relevantes son aquellos que suelen aparecer en otros contextos relacionados donde el estudiante tiene que hacer uso del conocimiento que adquirió durante la experiencia de aprendizaje.

La estructura de objetivos es una medida que es calificada como alta para aquellos problemas que muestran una estructura de objetivos apreciable. La idea detrás de esta medida es promover los problemas en los cuales hayan suficientes pasos intermedios significativos como para modificar,de forma dinámica,la interfaz grafica en la que se resuelven los problemas.De esta forma, se promueven los problemas cuyo proceso de resolución permite adicionar ó modificar componentes gráficos que ayuden al estudiante a visualizar los pasos intermedios.

La medida de generalidad esta basada en qué tan general o especifico es el problema dentro del dominio de conocimiento. Los problemas que logran una calificación alta en este aspecto promueven soluciones que refuerzan el conocimiento general de los estudiantes. Los problemas que tienen datos de entrada muy específicos o que involucran conceptos que no son aplicables de manera transversal en el dominio de control reciben una calificación baja en esta medida.

Para cada una de las medidas, tres posibles valores pueden ser otorgados: Alto, Medio y Bajo. Estas tres etiquetas están relacionados con valores los valores numéricos 3, 2 y 1 respectivamente. La asignación de una de las tres etiquetas a cada una de las medidas por cada problema es una tarea más sencilla que la calificación numérica directa. Una vez todos los problemas han sido valorados,se requiere un mecanismo para comparar un problema con otro y es entonces cuando las etiquetas se convierten en números. Para cada problema, se calcula un puntaje que es el resultado de la suma de cada uno de los puntajes obtenidos en cada medida. De esta forma el puntaje mínimo por problema, es 3 y el puntaje máximo es 9. Una vez cada problema tienen un único puntaje numérico asignado, el proceso de ordenamiento es directo y pueden entonces seleccionarse los problemas con mejor puntaje dentro del conjunto.

2.3.2 Escalafón de los problemas seleccionados.

En el anexo A de este documento se presenta la descripción de cada problema, los objetivos de aprendizaje que están relacionados con cada problema, las valoraciones otorgadas a cada una

de las medidas de cada problema y el puntaje final obtenido por el problema. La lista se presenta ordenada de mayor a menor puntaje dentro de cada grupo. En letra negrita se presentan los problemas que han sido seleccionados para su implementación en cada uno de los sistemas tutores inteligentes.

3. Análisis comparativo de la capacidad de modelamiento de diferentes tipos de tutores inteligentes para temáticas de control

En este capítulo se hace una descripción detallada del análisis comparativo que se llevo a cabo para determinar dos tipos de sistemas tutores inteligentes que deben ser implementados, de los tres abordados. Debido al conocido costo en tiempo y recursos que la construcción de sistemas tutores representa para los desarrolladores y autores, la implementación de los tres tipos de tutores resultaría impráctica para los alcances de este proyecto.

Por lo tanto se presenta un análisis profundo de las características generales de los tutores y de cómo estas características afectarían la etapa de construcción de cada uno y el resultado mismo. El objetivo del análisis es determinar los dos tipos de tutores que provean la mejor relación costo-beneficio durante la etapa de implementación y también en términos de los resultados finales del estudio comparativo experimental.

Un análisis general de las características de cada una de las técnicas para creación de tutores permite reconocer las ventajas y desventajas de cada uno de las técnicas. Este análisis esta basado en características que han sido bien documentadas por diferentes investigadores y que se han reconocido como generales pues aplican para la mayoría de los dominios de conocimiento. Sin embargo, debido a la especificidad de los conceptos y de las estrategias que son enseñadas en cada dominio, resulta necesario estudiar las particularidades del dominio y relacionarlas con las características principales que cada uno de los tutores presenta. Este análisis especifico permite evaluar la efectividad que cada tipo de tutor presenta cuando se trata

de modelar el dominio del conocimiento de interés, en este caso sistemas de control. La factibilidad de modelamiento e implementación también son aspectos analizados durante esta etapa. Como último aspecto se evalúa la facilidad de implementación a partir del conocimiento previo que se tiene del funcionamiento de cada una de las herramientas de autor y del conocimiento general del funcionamiento de cada técnica.

3.1 Ejemplos de modelamiento para temáticas especificas de sistemas de control.

Con el objetivo de dar un mayor alcance en el entendimiento de las técnicas de modelamiento que se han abordado en este documento, esta sección presenta un conjunto de ejemplos teóricos que demuestran la forma en que cada una de estas técnicas pueden ser aplicadas al dominio de sistemas de control.

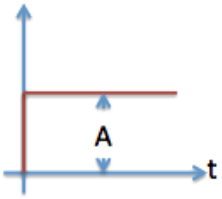
Del conjunto de problemas que fueron seleccionados en el capítulo 2, se seleccionó un subconjunto de 3 problemas para realizar un análisis teórico orientado a entender la forma en que cada una de las técnicas de modelamiento del conocimiento se aplica de forma especifica para modelar problemas en sistemas de control. Por cuestiones de espacio, debido a que el modelamiento involucra gran detalle, se presentará la descripción de solo un problema. Sin embargo debe tenerse claro que la esencia de la forma como se aplican las técnicas queda presentada y que se extiende para los demás problemas de la misma forma: Definición de variables, definición de funciones utilitarias, establecimiento de reglas o restricciones, ajuste de reglas o restricciones para tener mayor detalle en los errores y en los mensajes de ayuda.

Estos ejemplos brindaron la posibilidad de visualizar ventajas y desventajas en el momento de realizar el modelamiento. Brindaron también una medida del esfuerzo que se requiere para modelar cada uno de los problemas para cada una de las técnicas. Al final de la sección se denotan estas ventajas y desventajas y se presenta una discusión alrededor de los conceptos involucrados durante este análisis.

3.1.1 Análisis de desempeño: Cálculo de la salida de un sistema de primer orden.

Como ejemplo de modelamiento se presenta un problema en el cual el estudiante debe calcular la salida de un sistema de primer orden cuando el sistema es excitado con una señal de entrada conocida y cuando la función de transferencia que modela el sistema también es conocida.Para este tipo de problema, el enunciado del problema se presenta a continuación incluyendo las gráficas que se presentan al estudiante como ayuda para la comprensión del enunciado:

"El sistema LTI que se muestra en la figura, está representado por la función de transferencia G(S). Si se excita el sistema con una entrada tipo paso de amplitud A, aplique la transformada inversa de Laplace para calcular la expresión que representa la salida del sistema en el dominio del tiempo. Adicionalmente calcule la expresión para la Ganancia estática K y para la constante de tiempoτ"



$$G(s) = \frac{a}{b \cdot s + c}$$

Los pasos comunes de resolución del problema de acuerdo a la teoría de sistemas de control presentada en el capítulo 2 son:

1) Calcular la expresión para la señal de entrada en el dominio de Laplace, X(S). Esta expresión puede calcularse mediante dos estrategias, evaluación de la integral de Laplace ó uso de una tabla de transformadas.
2) Calcular la expresión para la señal de salida, Y(S), en el dominio de la variable compleja S.Este cálculo corresponde al producto de las expresiones de las expresiones X(S) y G(S).
3) Calcular una expresión para Y(S) en términos de funciones simples usando expansión en fracción parciales.
4) Calcular la expresión para y(t) a través del cálculo de la transformada inversa de Laplace de la expresión Y(S) calculada en el paso anterior. Esta expresión puede calcularse mediante dos estrategias, evaluación de la integral de Laplace ó uso de una tabla de transformadas.

5) Calcular el valor de la ganancia estática K y la constante de tiempo τ. Este cálculo se puede realizar mediante la parametrización de la función de transferencia, mediante la parametrización de la expresión para y(t) o mediante el cálculo directo de los valores de K y τ a través del análisis de la expresión de y(t) aplicando las definiciones para K y τ.

Para el problema anterior, las respuestas correctas que debe proporcionar el estudiante, en cada uno de los pasos son las siguientes:

1) Dado que la señal de entrada es una señal tipo paso con amplitud A. La expresión para X(S) que el estudiante debe calcular es $X(S) = \frac{A}{S}$. Este resultado se puede obtener por medio del uso de tablas de transformada de Laplace o por medio de la evaluación directa de la integral de Laplace.
2) El cálculo de Y(S) se realiza escribiendo el producto de las expresiones X(S) y G(S), lo cual resulta ser: $Y(S) = \frac{a}{bS+c} \cdot \frac{A}{S}$
3) Al descomponer Y(S) en funciones simples mediante la descomposición de fracciones parciales, la respuesta del estudiante debe ser: $Y(S) = A\ (\frac{-\frac{a}{c}}{S+\frac{c}{b}} + \frac{\frac{a}{c}}{S})$
4) Al calcular la transformada inversa de Y(S), mediante el uso de tablas de transformada inversa ó la evaluación directa de la integral inversa de Laplace, la expresión para la señal de salida en el dominio del tiempo que debe calcular el estudiante es: $y(t) = \frac{a}{c}(1 - e^{\frac{-c}{b} \cdot t})$.
5) Siguiendo cualquiera de las estrategias presentadas en el apartado anterior, el estudiante debe llegar a la conclusión que los valores para las constantes K y τ son: $K = \frac{a}{c}\ y\ \tau = \frac{b}{c}$

Para proporcionar una respuesta determinada, un estudiante puede interactuar de formas diferentes con el sistema tutor inteligente dependiendo de las posibilidades que el tutor brinde en términos de interfaz gráfica. De esta forma un estudiante puede proporcionar una respuesta correcta por medio de la selección de la respuesta correcta entre un grupo de opciones; puede introducir datos numéricos en campos de texto; puede escribir expresiones matemáticas utilizando una sintaxis determinada; puede completar tablas de datos, realizar acciones de arrastrar y soltar, interactuar con gráficos por medio del puntero, etc. Por lo

tanto para cada problema se define una forma de interacción que esta orientada a definir la estructura de la respuesta del estudiante y por lo tanto define variables que pueden ser usadas en las restricciones (o reglas) para evaluar si los valores de dichas variables son correctas o no. Para este ejemplo especifico la forma que se definió para la interacción entre el usuario y el sistema esta determinado por la siguiente interfaz gráfica de usuario:

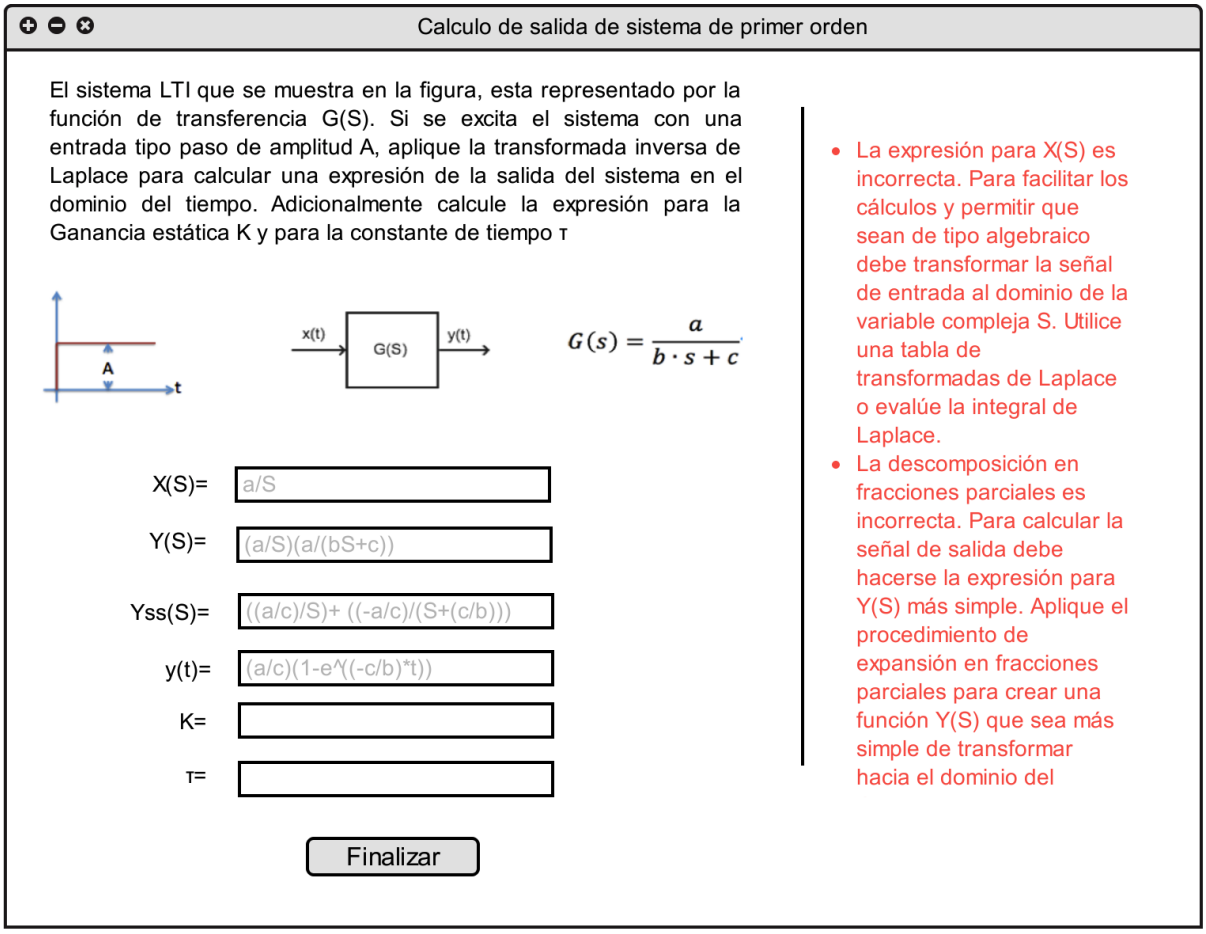


Figura 2.Interfaz gráfica hipotética para el ejemplo de modelamiento. En la parte izquierda se encuentra el panel de ayuda.

Para este problema se asume que el estudiante proporciona expresiones que se introducen en forma de texto, usando una sintaxis compatible con la de un lenguaje de programación de propósito general, por ejemplo Matlab..

El modelo de interacción definido por esta interfaz gráfica solicita que el estudiante introduzca todos las respuestas y luego pulse el botón “Finalizar”, momento en el cual el tutor creará las variables presentadas, cargará las funciones que se definan según cada técnica de modelamiento y evaluara el algoritmo típico de la técnica. Como resultado, el estudiante

obtendrá mensajes de éxito o de error dependiendo de si sus respuestas son correctas o incorrectas, respectivamente.

- Modelamiento basado en restricciones:

De acuerdo a la teoría básica presentada en el capítulo 1, los sistemas tutores basados en restricciones intentan modelar el dominio del conocimiento por medio de la definición de un conjunto de restricciones que no deben ser violadas por las respuestas que proporciona el estudiante. Estas restricciones son de carácter evaluativo y permiten evaluar que el conocimiento de tipo declarativo ha sido aplicado de forma correcta.

Para lograr definir las restricciones, se hace necesario identificar cuales son las diferentes componentes de la respuesta del estudiante. La respuesta del estudiante puede ser proporcionada de diferentes maneras y este conjunto de distintas maneras varía dependiendo de las posibilidades existentes en términos de interfaz gráfica de usuario.

Para el problema anterior, supóngase que las respuestas son proporcionadas como expresiones en sintaxis matemática, y que se definen las siguientes variables y funciones:

1. X_s:Variable que contiene la expresión introducida por el estudiante como respuesta al paso 1.
2. X_t: Variable que contiene la expresión conocida al inicio del problema que representa x(t).
3. Y_s: Variable que contiene la expresión proporcionada por el estudiante como respuesta al paso 2.
4. G_s: Variable que contiene la expresión conocida al inicio del problema representando G(S).
5. Y_{ss}: Variable que contiene la expresión introducida por el estudiante que representa la versión de Y(S) en términos de funciones simples.
6. Y_t: Variable que contiene laexpresión proporcionada por el estudiante que representa la salida del sistema y(t).
7. K: Variable que contiene la expresión proporcionada por el estudiante que representa la ganancia estática

8. Tau: Variable que contiene la expresión proporcionada por el estudiante que representa la constante de tiempo.
9. Laplace(arg): Función que retorna una expresión que corresponde a la transformada de Laplace de la expresión que se pasa como argumento.
10. LaplaceInv(arg): Función que retorna una expresión que corresponde a la transformada inversa de Laplace de la expresión que se pasa como argumento.
11. FracPar(arg): Función que retorna una expresión que corresponde a la expansión en fracciones parciales de la expresión que se pasa como argumento.

Para esta técnica de modelamiento, una vez el estudiante hace clic en el botón finalizar, se evalúan una por una todas las restricciones para encontrar cuales son las restricciones que no se cumplen. Para todas aquellas restricciones que no se cumplan, el tutor seleccionará los mensajes de ayuda relacionados con cada restricción infringida y procederá a mostrar los mensajes en la interfaz de usuario.

De esta forma, usando las funciones y variables definidas previamente, los principios básicos del dominio pueden ser modelados mediante el conjunto de restricciones mostrados en la siguiente tabla. Para cada restricción se muestra un ejemplo de los mensajes de ayuda que se pueden mostrar a los estudiantes cuando la respuesta proporcionadainfringe una determinada restricción.

Tabla 3. Tabla de restricciones iniciales para modelar el problema análisis de respuesta de sistemas de primer orden.

Restricción (Cr,Cs)	**Mensaje de ayuda en caso que la respuesta proporcionada infrinja la restricción.**
Cr: SI X_s no es nula **y** X_t no es nula ENTONCES DEBE SER CIERTO QUE Cs: X_s es igual a Laplace(X_t)	La expresión para X(S) es incorrecta. Para facilitar los cálculos y permitir que sean de tipo algebraico debe transformar la señal de entrada al dominio de la variable compleja S. Utilice una tabla de transformadas de Laplace o evalúe la integral de

	Laplace.
Cr: SI $Y_s$no es nula **y**$G_s$no es nula **y**X_s no es nula ENTONCES DEBE SER CIERTO QUE Cs: $Y_s$es igual a G_sX_s	La expresión para Y(S) es incorrecta. Para calcular la salida del sistema, el producto de la función de transferencia y la entrada debe ser calculado. Calcule el producto G(S)X(S).
Cr: SI $Y_s$no es nula **y**Y_{ss} no es nula ENTONCES DEBE SER CIERTO QUE Cs: Y_{ss}es igual a FracPar(Y_s)	La descomposición en fracciones parciales es incorrecta. Para calcular la señal de salida debe hacerse la expresión para Y(S) más simple. Aplique el procedimiento de expansión en fracciones parciales para crear una función Y(S) que sea más simple de transformar hacia el dominio del tiempo.
Cr: SI Y_{ss} no es nula **y**$Y_t$no es nula ENTONCES DEBE SER CIERTO QUE Cs: $Y_t$es igual a LaplaceInv(Y_{ss})	La expresión para la salida del sistema es incorrecta. La transformación inversa de Laplace contiene errores. Evalúe la Integral inversa de Laplace o utilice una tabla de transformadas inversas.
Cr: SI K no es nula **y**$G_s$es de la forma $\frac{p}{qS+r}$ ENTONCES DEBE SER CIERTO QUE Cs: $K\ es\ igual\ a\ \frac{a}{c}$	El valor calculado para K es incorrecto. La ganancia estática del sistema se calcula como la relación de los valores en estado estable de la señal de salida respecto a la entrada. También puede observarse que este valor está representado en la función

	de transferencia cuando tiene la forma G(s)=K/tau*S+1
Cr: SI τ no es nula **y** G_s es de la forma $\frac{p}{qS+r}$ ENTONCES DEBE SER CIERTO QUE Cs: $\tau \; es \; igual \; a \; \frac{b}{c}$	El valor calculado para Tau es incorrecto. La constante de tiempo del sistema se calcula como el tiempo que tarda el sistema en alcanzar el 62.8% del valor final de la señal de salida. También puede observarse que este valor está representado en la función de transferencia cuando tiene la forma G(s)=K/tau*S+1

Como puede observarse en la tabla anterior, las restricciones definidas representan los conceptos principales y de más alto nivel en el dominio de interés. Para la evaluación de estos principios de tan alto nivel se requiere contar con funciones que también son de alto nivel como las funciones FracPar(),Laplace() y LaplaceInv(). Adicionalmente la implementación de la evaluación "*x* es de la forma *y*" también implica una función de alto nivel de abstracción. La implementación computacional de estas funciones requieren un alto esfuerzo de implementación y solo para ciertas plataformas tecnológicas como Matlab, Octave y otros paquetes matemáticos, el esfuerzo se puede reducir por medio del uso de librerías disponibles.

Sin embargo la forma como se han definido las restricciones no permite tener un detalle mayor de cual ha sido el error que un estudiante ha cometido al resolver una determinada expresión. Por ejemplo al calcular la expresión para X(S), el estudiante puede cometer errores típicos como aplicar la tabla de transformadas de forma incorrecta, olvidar la amplitud de la señal paso y por lo tanto omitir el término A en la respuesta suministrada. Estos errores no se pueden identificar por medio de las restricciones presentadas en la tabla anterior. Se requiere, introducir nuevas restricciones que pueden ser de carácter sintáctico o semántico, pero que de todas formas, corresponden a restricciones que modelan principios de más bajo nivel y que son más

dependientes de los datos del enunciado del problema especifico y por lo tanto tienen un menor alcance de aplicación en términos de generalidad.

Por lo tanto, si se deseara establecer una restricción que pudiera detectar un error de bajo nivel como por ejemplo, detectar que la amplitud A ha sido omitida en la expresión de X(S), o si se deseara detectar en la expansión de fracciones parciales, cual de los numeradores de las fracciones simples esta errado, se requiere definir restricciones mucho más complejas como las que se muestran a continuación.

Tabla 4. Ajuste que se realiza a las restricciones para detectar errores más granulares.

Restricción (Cr,Cs)	**Mensaje de Ayuda**
Cr: SI X_s no es nula **y** X_t no es nula **y** X_t tiene la forma $\rho U(t)$ ENTONCES DEBE SER CIERTO QUE Cs: X_s tiene la forma $\frac{\delta}{S}$ **y** que $\delta = \rho$	La expresión para X(S) es incorrecta. La expresión para X(S) debe tener la forma $\frac{\delta}{S}$. Adicionalmente δ debe corresponder a la amplitud de al señal paso aplicada a la entrada.
Cr: SI Y_s no es nula **y** Y_{ss} no es nula **y** G_s no es nula **y** X_s no es nula **y** G_s tiene la forma $\frac{p}{qS+r}$ **y** X_s tiene la forma $\frac{\delta}{S}$ ENTONCES DEBE SER CIERTO QUE Cs: Y_{ss} tiene la forma $A\left(\frac{-\frac{a}{c}}{S+\frac{c}{b}}+\frac{\frac{a}{c}}{S}\right)$ **y** $A = \delta$ **y** $a = p$ **y** $b = q$ **y** $c = r$	La descomposición en fracciones parciales es incorrecta. Debido a que G_s es de primer orden y la entrada es tipo paso, la descomposición en fracciones parciales debe tener dos términos $\left(\frac{\alpha}{S+\beta}+\frac{\gamma}{S}\right)$. Calcule las constantes α, β, γ adecuadamente.

Nótese sin embargo que la primera restricción, al ser infringida, solo es capaz de detectar que alguna de las dos situaciones que se requierenes incorrecta. Puede ser que X(S) no tenga la forma correcta, o puede ser que $\delta \neq \rho$. Lo mismo ocurre con la segunda restricción mostrada, hay 5 posibles razones por las cuales la restricción se puede infringir: Y_{ss}**NO** tiene la forma $A\left(\frac{-\frac{a}{c}}{S+\frac{c}{b}}+\frac{\frac{a}{c}}{S}\right)$ **ó** $A \neq \delta$ **ó** $a \neq p$ **ó** $b \neq q$ **ó** $c \neq r$

Si se requiere que el tutor tenga un mayor grado de conocimiento sobre el estado del problemay que pueda detectar las dos (o las 5 para el segundo caso) situaciones de forma independiente, se requieren crear dos (o 5) restricciones a partir de la restricción mostrada en la tabla anterior.

Tabla 5. Ajuste a las restricciones para que el mensaje de ayuda pueda ser más especifico.

Restricción (Cr,Cs)	**Mensaje de Ayuda**
Cr: SI X_s no es nula Y X_t no es nula Y X_t tiene la forma $\rho U(t)$ ENTONCES DEBE SER CIERTO QUE Cs: X_s tiene la forma $\frac{\delta}{S}$	La expresión para X(S) es incorrecta. La expresión para X(S) debe tener la forma $\frac{\delta}{S}$.
Cr: SI X_s no es nula Y X_t no es nula Y X_t tiene la forma $\rho U(t)$ Y X_s tiene la forma $\frac{\delta}{S}$ ENTONCES DEBE SER CIERTO QUE Cs: que $\delta = \rho$	La expresión para X(S) es incorrecta. δdebe corresponder a la amplitud de al señal paso aplicada a la entrada.

Este hecho permite concluir que a mayor especificidad que se requiera en los mensajes de ayuda, mayor va a ser el número de restricciones que se requieren, mayor va a ser el número de pruebas booleanas que hacen parte de las condiciones de relevancia y menor va a ser el número de pruebas booleanas en las condiciones de satisfacción.

Si las restricciones no se diseñan correctamente, con la complejidad requerida e incluyendo las comparaciones y operadores necesarios para detectar correctamente y evaluar ciertos estados del problema, el efecto será que para ciertos estados del problema, algunas restricciones no resultaran relevantes y por lo tanto no se podrá comprobar la satisfacción de principios básicos, lo cual es un error en el modelamiento, o peor aun, que para un estado válido alguna restricción resulte infringida sin que el estado realmente este violando un principio básico del dominio.

- Modelamiento basado en reglas de producción

El objetivo principal del modelamiento basado en reglas es replicar en el sistema computacional el proceso mental que el estudiante debería ejecutar si tuviera un buen rendimiento. Para realizar esto, el modelo define un conjunto de variables conocido como la memoria de trabajo (Working Memory). Esta memoria de trabajo corresponde a un conjunto de variables y estructuras de datos cuyos valores van cambiando a medida que el tutor va avanzando en el seguimiento del estudiante. Para poder seguir a un estudiante a través del desarrollo del problema, lo que significa reconocer si cada una de las acciones que el estudiante realiza en la interfaz gráfica son correctas o no, el sistema tutor inteligente ejecuta todas las reglas de producción posibles que hagan que la memoria de trabajo del tutor sufra un cambio exactamente igual al cambio que sufre la memoria de trabajo del estudiante después de realizar la acción. La memoria de trabajo del estudiante esta representada por los valores de cada uno de los elementos de la interfaz gráfica.

Supóngase que un estudiante introduce el valor “a/b” en el campo K de la interfaz gráfica presentada anteriormente y que previamente el campo se encontraba vacío. Los demás campos se encuentran con un valor que puede ser NULO si el estudiante no ha les ha asignado un valor.En este caso, el cambio total la memoria de trabajo del estudiante, corresponde al cambio ocurrido solo en el campo K, que tras la acción, pasó de tener el valor NULO a tener el valor “a/b”. Los demás campos permanecieron con el mismo valor.

Si tras ejecutar un numero de reglas de producción el tutor logra reproducir la misma acción que el estudiante, es decir, que solo el campo K cambie y el nuevo valor sea exactamente igual a “a/b”, entonces el tutor asume que el estudiante ha ejecutado el mismo conjunto de reglas que el tutor y que por lo tanto su comportamiento se puede explicar mediante dichas reglas. En caso que alguna de las reglas ejecutadashaya sido marcada como incorrecta (Buggy Rule), entonces el comportamiento se reconoce como incorrecto y el tutor como consecuencia envía los mensajes de ayuda apropiados al estudiante. Cuando el tutor no encuentra un conjunto de reglas que pueda explicar la acción del estudiante, se considera que la acción del estudiante es incorrecta.

De acuerdo a lo anterior, la primera definición que se debe realizar para iniciar con este tipo de modelamiento es la definición de la memoria de trabajo. Una de las formas mas sencillas de hacerlo, según lo recomiendan los autores más influyentes de esta técnica, corresponde a diseñar la interfaz gráfica de usuario con la que el estudiante va a interactuar y a partir de dicha

interfaz crear las variables de la memoria de trabajo. Asumiendo nuevamente que la interfaz de interacción corresponde a la figura anterior, la memoria de trabajo podría ser definida de la siguiente forma:

Tabla 6. Nombre de variables y funciones definidas para el modelamiento basado den reglas de producción.

Nombre Variable	**Descripción**
objetivo	Esta variable permite almacenar el objetivo actual del problema. El objetivo es una cadena de texto que se usa como variable de control y que permite identificar el estado actual del problema.
X_s	Variable que almacena el resultado que el tutor ha calculado para la expresión que define X(S).
Y_s	Variable que almacena el resultado que el tutor ha calculado para la expresión que define Y(S).
Y_{ss}	Variable que almacena el resultado que el tutor ha calculado para la expresión que define X(S).
G_s	Variable que almacena la definición de G(S) que puede ser obtenida por medio de un cálculo o a partir de los datos iniciales de problema
X_t	Variable que almacena la definición de x(t) que puede ser obtenida por medio de un cálculo o a partir de los datos iniciales de problema
Y_t	Variable que almacena el resultado que el tutor ha calculado para la expresión que define y(t).

Para todas las variables el valor inicial es NULL. En el siguiente listado se presentan las reglas de producción que pueden ser ejecutadas por el tutor para calcular los valores correctos que un estudiante debe proporcionar en la interfaz y de esta manera poder detectar si su comportamiento es correcto o no.

Tabla 7. Definición de las reglas de producción requeridas para implementar el tutor deseado

ID	Reglas IF-THEN	Sugerencia o pista asociada
R1	**SI** y(t) no es NULL **ENTONCES** DONE	Ha finalizado correctamente el ejercicio.
R2	**SI** y(t) es NULL **ENTONCES** asignar objetivo igual a calcularYt	No ha calculado y(t) todavía, calcule y(t).
R3	**SI** el objetivo es calcularYt **y** Yss es NULL **ENTONCES** colocar como objetivo calcularYss	Para calcular y(t) debe calcular Yss. Calcule Yss.
R4	**SI** el objetivo es calcularYt **y**Yss no es NULL **ENTONCES** asignar y(t)=INVLAPACE(Yss)	El valor que ha calculado para Yss es correcto. Utilice este valor para calcular y(t). Aplique la transformación inversa de Laplace para obtener y(t) a partir de Yss.
R5	**SI** el objetivo es calcularYss **y**Ys no es NULL **ENTONCES** asignar Yss=FRACPAR(Ys) colocar como objetivo calcularYt	Y(S) debe ser transformado al dominio del tiempo. Sin embargo la transformación inversa de la expresión para Y(S) no es fácil de calcular. Utilice la descomposición en fracciones parciales para convertir Y(S) en una suma de funciones mas simples. Luego proceda a realizar la transformación.
R6	**SI** el objetivo es calcularYss **y**Ys es NULL **ENTONCES** colocar como objetivo calcularYs	Y(S) corresponde a la expresión de salida del sistema en el dominio de Laplace. Calcule este valor para luego transformar al dominio del tiempo y obtener la respuesta al problema.
R7	**SI** el objetivo es calcularYs **y**Gs no es NULL **y**Xs no es NULL **ENTONCES** asignar Ys=GsXs colocar como objetivo calcularYss	Ya ha calculado los valores de G(S) y X(S). Con el producto algebraico de estos valores puede calcular el valor de Y(S).
R8	**SI** el objetivo es calcularYs **y**Gs es NULL **ENTONCES** colocar como objetivo CalcularGs	Para calcular la salida del sistema en el dominio de Laplace es necesario conocer la función de transferencia, G(S).
R9	**SI** el objetivo es calcularYs **y**Xs es NULL **ENTONCES** colocar como objetivo CalcularXs	Para calcular la salida del sistema en el dominio de Laplace es necesario conocer la entrada al sistema en el dominio de Laplace X(S).
R10	**SI** el objetivo es calcularGs **y**Gs es NULL **ENTONCES** asignar Gs=ProblemData(Gs) **y**colocar	La función de transferencia del sistema, G(S), es un dato que fue establecido en el enunciado del

	como objetivo calcularYs	problema. Utilice el valor establecido para calcular Y(S).
R11	**SI** el objetivo es calcularXs **y**Xt es NULL **ENTONCES** colocar objetivo calcularXt	X(S) no ha sido calculada aún. Utilice el valor de x(t) para realizar el cálculo de X(S).
R12	**SI** el objetivo es calcularXs **y**Xt no es NULL **ENTONCES** asignar Xs=LAPLACE(Xt) **y**colocar como objetivo calcularYs	x(t) tiene un valor conocido. Puede utilizar la transformada directa de Laplace para calcular el valor de X(S).
R13	**SI** el objetivo es calcularXt **y**Xt es NULL **ENTONCES** Xt=ProblemData(Xt) **y**colocar como objetivo calcularXs	La señal de entrada al sistema, x(t), es un dato que fue establecido en el enunciado del problema. Utilice el valor establecido para calcular X(S).

Obsérvese que cada regla ha sido acompañada de un mensaje de ayuda. Es importante observar que los mensajes de ayuda no son de carácter evaluativo como en el caso del modelamiento basado en restricciones. Por su naturaleza las restricciones detectan errores en la respuesta, por lo tanto los mensajes están orientados a identificar el error y a impartir una instrucción mediante la cual se supone que el estudiante puede corregir el error. Para las reglas de la tabla anterior los mensajes están orientados a mostrar al estudiante cual es el siguiente paso a ejecutar. Si un estudiante proporciona un valor incorrecto para algún campo y como resultado el tutor no es capaz de encontrar una regla para simular tal comportamiento, entonces el comportamiento se considera incorrecto y se envía un mensaje de error simple y general al estudiante, por ejemplo: “El valor proporcionado es incorrecto.”. Si el estudiante solicita ayuda, dependiendo del estado en el que se encuentre y dado que el tutor siempre se encontrará en el mismo estado que el estudiante, el tutor,busca cuales reglas podrían ejecutarse a partir del estado actual de su memoria de trabajo. Para cada una de esas reglas, extrae los mensajes de ayuda (que ya no son generales sino que son específicos respecto al procedimiento) y los muestra al estudiante.

En la siguiente tabla se muestra una prueba de escritorio de la ejecución del conjunto de reglas usando un algoritmo ingenuo (“naive”) de encadenamiento hacia delante (forward chaining)(Hayes-Roth, Waterman, & Lenat, 1983). Para efectos demostrativos y gracias a que la tabla de reglas definida tiene un numero reducido de reglas (pues una sola estrategia de solución fue implementada), en las cuales se han reducido las posibilidades de conflictos utilizando la variable “objetivo”, la ejecución puede realizarse utilizando dicho algoritmo. Sin embargo, las implementaciones típicas en sistemas de reglas de producción como JESS y

CLIPS utilizan algoritmos más avanzados como el algoritmo de RETE (Forgy, 1982) para seleccionar las reglas que podrían activarse y en específico los tutores tipo model-tracing utilizan otro algoritmo que lleva su mismo nombre, el cual se encuentra documentado de forma detallada en (Aleven, 2010) y realiza unas modificaciones a los algoritmos tradicionales de reglas de producción, específicamente en las etapas de resolución de conflictos. La descripción detallada del algoritmo model-tracing y todos los conceptos que involucra respecto al comportamiento en especifico de los sistemas tutores inteligentes, se encuentra fuera del alcance de este trabajo. Sin embargo se recomienda la revisión de (Aleven, 2010) por ser tal vez el reporte que describe con mayor detalle dicho algoritmo en los últimos 20 años.

En esta prueba de escritorio se muestran los diferentes valores que son asignados a cada una de las variables de la memoria de trabajo. Utilizando letra negrita se muestran la variables que hacen parte de los datos iniciales del problema.

Tabla 8. Prueba de escritorio que muestra como se generan los valores adecuados para las variables del Working Memory

		Variables de la Memoria de Trabajo						
Paso	**Regla Activada**	**objetivo**	**Xt**	**Xs**	**Gs**	**Ys**	**Yss**	**Yt**
0	-	NULL	NULL	NULL	NULL	NULL	NULL	NULL
1	R2	calcularYt						
2	R3	calcularYss						
3	R6	calcularYs						
4	R8	CalcularGs						
5	R10	calcularYs			**a/bs+c**			
6	R9	CalcularXs						
7	R11	calcularXt						
8	R13	CalcularXs	**AU(t)**					
9	R12	calcularYs		*A/S*				

10	R7	calcularYss				*(A/s)(a/bs+c)*		
11	R5	calcularYt					*A[((-a/c)/(s+(c/b)))+((a/c)/s)]*	
12	R4							*AK(1-e^(-t/tau))*
13	R1	DONE						

Si se requiere que el tutor identifique errores específicos cometidos por el estudiante se necesita incluir "reglas de error" (Buggy Rules) que modelen específicamente acciones incorrectas.Por ejemplo, si se deseara detectar que la amplitud A ha sido omitida en la expresión de X(S), o si se deseara detectar en la expansión de fracciones parciales, cual de los numeradores de las fracciones simples está errado, se requiere definir las siguientes reglas de error:

Tabla 9. Dos reglas más que deben ser insertadas para contar con Buggy Rules.

ID	Reglas IF-THEN	Error detectado
R14	**SI**el objetivo es calcularXs **y**x(t) tiene la forma $\rho U(t)$ **ENTONCES**asignar a Xs el valor **1/S**	El valor para X(S) es incorrecto. Ha omitido la amplitud de la señal de entrada. La amplitud de la señal de entrada también debe ser considerada al hacer la transformada de Laplace.
R15	**SI**el objetivo es calcularYss y Ys es igual a (A/s)(a/bs+c) **ENTONCES** asignar a Yssel valor *A[((-**b/a**)/(s+(c/b)))+((a/c)/s)]*	La expresión para Yss es incorrecta. El numerador de la expresión simple $\frac{\alpha}{S+\beta}$no es **–b/a**. Revise el procedimiento de expansión en fracciones parciales para corregir el valor de Yss

Nótese en el ejemplo anterior que las reglas R14 y R15 solo tienen la capacidad de detectar los errores cuando el estudiante introduce **1/S** o la expresión *A[((-**b/a**)/(s+(c/b)))+((a/c)/s)],* respectivamente. Sin embargo el espacio de errores es infinito. Si por ejemplo el estudiante introduce el valor 1/S^2, en el campo X(S), la regla 14 será ejecutada pues el antecedente se cumple, la variable Xs en la memoria del trabajo del tutor será asignada con el valor 1/S, pero debido a que Xs en el tutor (1/s), es diferente de Xs en la interfaz (1/S^2), el tutor no podrá

reconocer esta regla como una regla que reproduzca el comportamiento del estudiante y por lo tanto no detectará el error. Un análisis sobre el efecto de este hecho, en comparación con el modelamiento basado en restricciones se realizará en la sección final de este capítulo.

Según lo anterior, para detectar un conjunto más amplio de errores se requiere incluir más reglas de producción como las que se muestran a continuación:

Tabla 10. Adición de más reglas para incluir más Buggy Rules.

ID	Reglas IF-THEN	Error detectado
R16	**SI**el objetivo es calcularXs **y** x(t) tiene la forma $\rho U(t)$ **ENTONCES**asignar a Xs el valor **1/S^2**	El valor para X(S) es incorrecto. Ha omitido la amplitud de la señal de entrada. La amplitud de la señal de entrada también debe ser considerada al hacer la transformada de Laplace.
R17	**SI**el objetivo es calcularXs **y** x(t) tiene la forma $\rho U(t)$ **ENTONCES**asignar a Xs el valor **1/S^3**	El valor para X(S) es incorrecto. Ha omitido la amplitud de la señal de entrada. La amplitud de la señal de entrada también debe ser considerada al hacer la transformada de Laplace.

Sin embargo, debido a que el espacio de posibles es errores es infinito, mediante la inclusión de nuevas reglas solo se puede tratar un conjunto limitado de errores, que en la práctica resultan ser los errores típicos de los estudiantes. Errores que no son típicos, son tratados mediante mensajes de ayuda de carácter general.

De igual forma que los tutores basados en restricciones, la ejecución de las reglas anteriores requieren la disponibilidad de funciones de alto nivel como INVLAPLACE, LAPLACE, FRACPAR. Estas funciones deben estar en la capacidad de tomar como entrada una cadena de texto que representa la expresión a transformar y producir como salida otra cadena de texto representado el resultado del cálculo. Este cálculo debe ser de tipo simbólico lo cual implica que la complejidad de las funciones y el esfuerzo que debe realizarse para desarrollar tales funciones es alto. Aunque desde el punto de vista del desarrollo de software se encuentran disponible muchas librerías de manejo simbólico de expresiones, es de notar que la mayoría de los entornos de ejecución de sistemas tutores inteligentes existentes utilizan lenguajes que presentan restricciones para el uso de dichas librerías, o lenguajes que son de carácter propietario donde la existencia de librerías realizadas por terceros es nula. Este aspecto será

aclarado con mayor detalle en la sección correspondiente a herramientas de autor, en este mismo capítulo.

- Modelamiento basado en demonstraciones

El modelamiento basado en demonstraciones tiene como objetivo facilitar el proceso de creación de los sistemas tutores inteligentes, a través de permitir que el autor pueda demostrar al sistema tutor cuales son los comportamientos correctos e incorrectos que un estudiante puede tener al desarrollar una tarea especifica. Estos comportamientos normalmente se materializan como acciones que se llevan a cabo con los componentes de la interfaz gráfica de usuario. De esta forma, introducir un valor en un campo de texto, seleccionar una o varias opciones que están disponibles en una lista, hacer clic en un botón ó cualquier otra acción típica que se pueda realizar sobre un componente de interfaz gráfica, resulta en una forma de interacción que puede ser demostrada por el autor al tutor.

Una vez el sistema tutor inteligente conoce cuales son las acciones que un estudiante debe realizar sobre la interfaz gráfica para manifestar un rendimiento correcto, el sistema tutor puede hacerle seguimiento de las acciones correctas, detectar acciones incorrectas y en dicho caso mostrar mensajes de error si se hace necesario.

Como se presentó en el capítulo 1, los tutores tipo example-tracing corresponden al tipo de tutores que se construyen usando demostraciones. Para este tipo de tutores, las acciones que se ejecutan en la interfaz gráfica, sean realizadas por el estudiante al resolver el problema o sean realizadas por el autor mientras hace las demostraciones, son descritas mediante una estructura de datos que permite identificar de forma única cada acción. Esta descripción en los tutores tipo example-tracing, se realiza mediante una tripla de información conformada por: el componente de interfaz gráfico sobre el cual se interactuó, el nombre de la acción que se ejecuto sobre el componente y la información que el usuario (estudiante o autor) proporcionó. Esta tripla se conoce como selección, acción y entrada, respectivamente (SAI del ingles Selection, Action, Input).

Para los tutores tipo example-tracing, cuando un autor demuestra el comportamiento correcto de un estudiante, procede a ejecutar acciones en la interfaz gráfica, estas acciones, que se pueden describir mediante una tripla SAI, son almacenadas en el gráfico de comportamiento

(ver sección 1.3). Posteriormente cuando un estudiante realiza acciones con el ánimo de resolver el problema, sus acción son comparadas con las acciones previamente almacenadas y de esta forma se puede hacer el seguimiento del estudiante a lo largo del problema.

La siguiente tabla muestra el conjunto de acciones que pueden ser demostradas a un tutor tipo example-tracing para poder hacer seguimiento del estudiante durante la solución del problema presentado en la sección 3.2.1. Se asume que la interfaz gráfica de usuario es la que se mostró en la figura anterior.

Tabla 11. Acciones requeridas para demostrar la solución correcta del problema

ID acción	Selección	Acción	Tipo de Acción	Entrada
1	campoXs	ingresarTexto	*Correcta*	*A/S*
2	campoYs	ingresarTexto	*Correcta*	*(A/s)(a/bs+c)*
3	campoYss	ingresarTexto	*Correcta*	*A[((-a/c)/(s+(c/b)))+((a/c)/s)]*
4	campoYt	ingresarTexto	*Correcta*	*AK(1-e^(-t/tau))*
5	campoTau	ingresarTexto	*Correcta*	*b/c*
6	campoK	ingresarTexto	*Correcta*	*a/c*
7	BotonFinalizar	Clic	*No aplica*	*No aplica*

El gráfico de comportamiento que conectaría la acciones de la tabla anterior, sería el que se muestra a continuación:

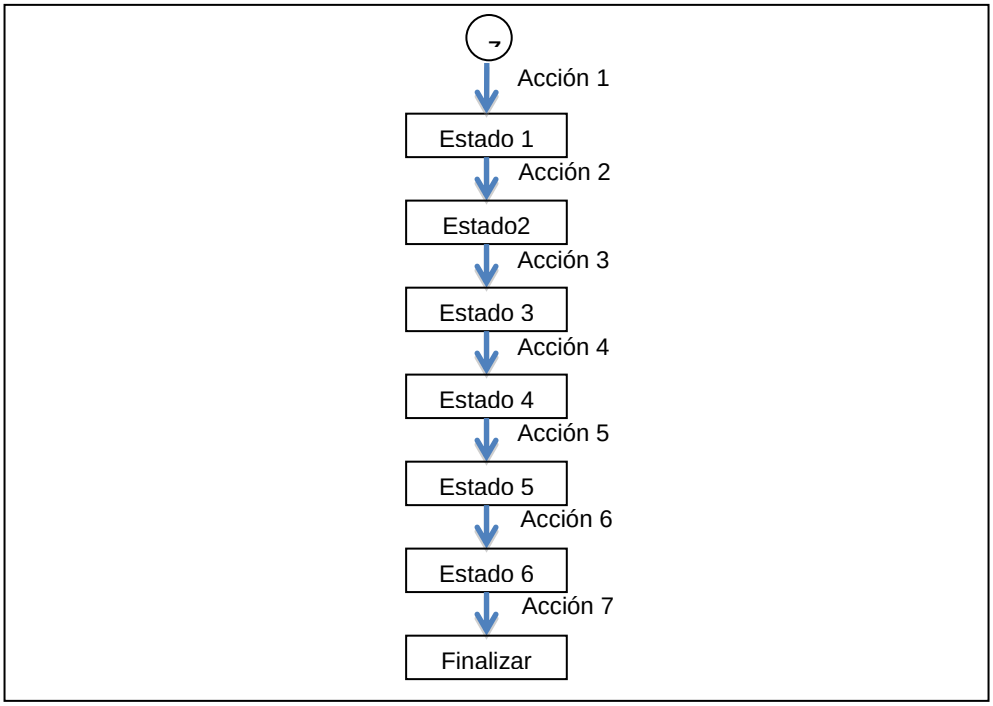


Figura 3. Grafo de comportamiento inicial, al demostrar las acciones definidas en la tabla anterior.

El grafico de comportamiento anterior debido a su simplicidad solo es capaz de reconocer comportamientos correctos que sean exactamente iguales a los descritos por sus acciones. Eso significa que si un estudiante ejecuta una acción incorrecta el tutor no puede detectar el error especifico y lanzará un mensaje de error general. Sin embargo, de la misma forma como ocurre en el modelamiento basado en reglas de producción, se pueden modelar acciones típicas incorrectas por medio de la definición de acciones de error (Buggy Actions). Si se desea por ejemplo, reconocer el error típico en el cual los estudiantes ingresan el valor 1/S en lugar de A/S, lo que significa que omiten el la amplitud de la señal de entrada, ó un error típico que ocurre durante el cálculo de Yss, se podrían entonces definir las siguientes acciones de error

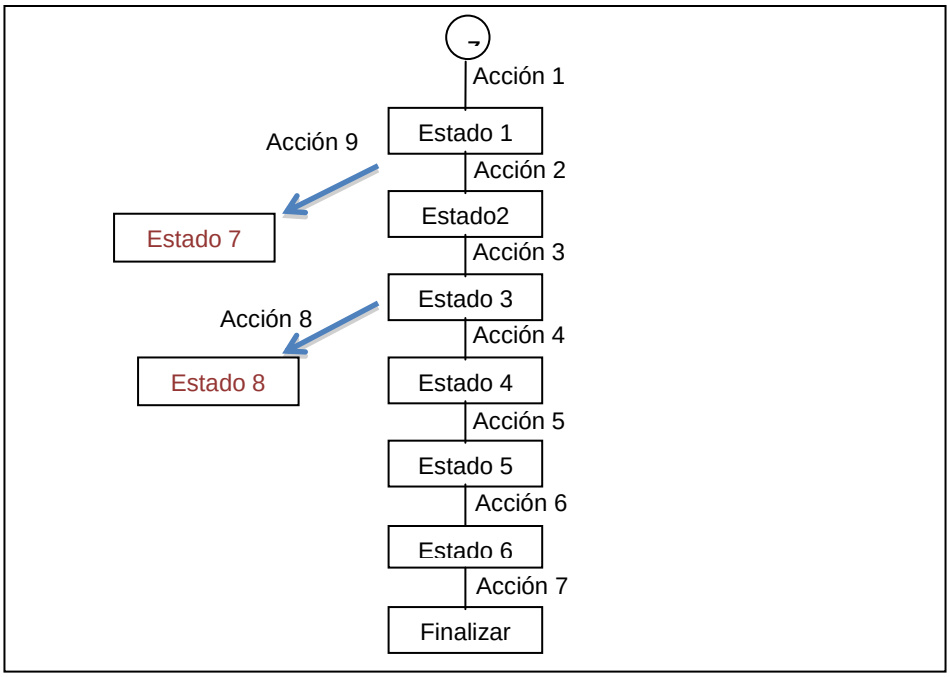
Tabla 12. Acciones adicionales que deben demostrarse para incluir Buggy Actions.

ID acción	Selección	Acción	Tipo Acción	Entrada
8	campoYss	ingresarTexto	*Incorrecta*	*A[((-**b/a**)/(s+(c/b)))+((a/c)/s)]*
9	campoXs	ingresarTexto	*Incorrecta*	*1/S*

Debido a que las acciones 8 y 9, pueden ser ejecutadas en lugar de las acciones 3 y 1 respectivamente, el gráfico de comportamiento debe actualizarse para que las acciones 8 y 9

puedan realizarse a partir de los estados correspondientes.De esta forma el gráfico de comportamiento modificado se puede observar en la siguiente figura:

Figura 4. Adición de acciones incorrectas al grafo de comportamiento



Una desventaja de demostrar las acciones que se presentaron en las tablas anteriores resulta ser que las cadenas en la columna “Entrada” son fijas y por lo tanto el estudiante debe introducir exactamente la misma cadena que fue demostrada para que el tutor reconozca la acción del estudiante equivalente a la acción demostrada. Para remediar esta desventaja se pueden hacer uso de funciones. Estas funciones pueden ser las mismas que se usaron para el modelamiento usando restricciones o reglas de producción. Las funciones se pueden usar para ayudar en la definición de cualquiera de las columnas de una acción. Este proceso se conoce como “generalización de las acciones” y se muestra a continuación:

Tabla 13. Generalizaciones que se deben hacer sobre la acción

ID acción	Selección	Acción	Tipo de Acción	Entrada
1	campoXs	ingresarTexto	*Correcta*	*LAPLACE(“A*U(S)”)*
2	campoYs	ingresarTexto	*Correcta*	*Gs * LAPLACE(campoXs)*
3	campoYss	ingresarTexto	*Correcta*	*FRACPAR(campoYs)*

4	campoYt	ingresarTexto	*Correcta*	*INVLAPLACE(Yss)*
5	campoTau	ingresarTexto	*Correcta*	*b/c*
6	campoK	ingresarTexto	*Correcta*	*a/c*
7	BotonFinalizar	Clic	*No aplica*	*No aplica*

Las funciones, asumiendo que pueden ser construidas y que pueden ser integradas en el entorno de ejecución del sistema tutor inteligente, permiten que la demostración tenga un nivel de generalidad alto y que no se deba aumentar el número de estados y de acciones para poder incluir otros valores correctos de entrada.

Otra ventaja de los sistemas tutores example-tracing radica en la facilidad con que se pueden generalizaciones las acciones de error para modelar errores típicos de los estudiantes.Si se modifican las acciones numero 8 y 9 como se muestra en la siguiente tabla, las acciones de error modelarían todos los comportamientos que no fueran compatibles con las acciones correctas.Esto significa que si el espacio de posibles errores se puede modelar mediante una función, dicha función puede ser utilizada para hacer el seguimiento de acciones erradas por medio de una sola acción demostrada.

Tabla 14. Generalización de acciones buggy.

ID acción	Selección	Acción	Tipo Acción	Entrada
8	campoYss	ingresarTexto	*Incorrecta*	*NOT_EQUAL_TO(FRACPAR(campoYs))*
9	campoXs	ingresarTexto	*Incorrecta*	*NOT_EQUAL_TO(LAPLACE(“A*U(S)”))*

- Resumen

En esta sección se ha presentado la forma en que cada una de las tres técnicas de modelamiento puede ser aplicada para el desarrollo de un sistema tutor inteligente que asista al estudiante en la resolución de un problema típico en el dominio de sistemas de control. Es

importante resaltar que cada técnica presenta ventajas y desventajas. Los sistemas tutores basados en restricciones, pueden modelar de forma más natural el espacio de posibles errores y los mensajes suelen proveer corrección sobre principios fundamentales del dominio que están mal entendidos por los estudiantes. Sin embargo estos tutores no presentan un buen rendimiento al momento de sugerir ayuda respecto a la siguiente acción que el estudiante debe tomar. El estudiante siempre debe caer en un error para poder recibir instrucción. Caso contrario son los sistemas tutores basados en reglas de producción, los cuales pueden dar sugerencias acerca del siguiente paso a ejecutar pero cuyo formalismo resulta un poco inconveniente cuando se requiere modelar un gran número de errores. Los tutores tipo example-tracing tienen un funcionamiento bastante parecido a los sistemas tutores basados en reglas, sin embargo su construcción es mucho más fácil debido a que se basa en demostraciones. La capacidad de generalización combinada con la de demostración de acciones de error resulta en una propiedad poderosa para modelar el espacio de posibles errores. Sin embargo para aquellos escenarios en los cuales la capacidad de generalización no está disponible, por ejemplo a causa de la inexistencia de las funciones o su dificultad para crearlas, los tutores tipo example-tracing resultan ser de carácter muy específicos y de difícil mantenimiento.

Se puede observar del análisis presentado, que en términos de capacidad de modelamiento las tres técnicas pueden ser usadas para modelar problemas de sistemas de control. Lo anterior se concluye a partir de la observación detallada de los conceptos teóricos y los problemas propuestos en el capítulo 2. Para todos los problemas propuestos existe una interfaz gráfica que puede ser diseñada para capturar información proveniente del estudiante. Independientemente de los componentes de interfaz gráfica que se requieran para lograr la interacción deseada, la información que el estudiante provee siempre se puede capturar y validar usando variables, funciones, restricciones, reglas de producción ó acciones demostradas. Debido a que el dominio de los sistemas de control es altamente matemático, la implementación de funciones que realicen cálculos es siempre factible. Adicionalmente, dado que los procedimientos para la resolución de los problemas están bien definidos y se pueden definir de forma exacta y algorítmica, la determinación de reglas, restricciones o acciones es siempre posible.

Aun cuando las tres técnicas de modelamiento analizadas en esta sección son factibles de implementación, dicha implementación sería bastante costosa si todos y cada uno de los componentes que conforman el sistema tutor inteligente tuvieran que ser creados por el autor.

Para evitar que la implementación se torne altamente costosa, típicamente se hace uso de herramientas de autor que facilitan la construcción del sistema tutor. En la siguiente sección se realiza un análisis de lascapacidades, ventajas y desventajas de cada una de las herramientas de autor, de forma que se pueda determinar si al introducir dichas herramientas de autor, la factibilidad de implementación de cada una de las tres técnicas cambia o se ve afectada de alguna manera.

3.2 Herramientas de Autor

Durante la última década se han planteado bastantes mecanismos acerca de cómo realizar y hacer más fácil la implementación de los Sistemas Tutores Inteligentes. Entre las técnicas más importantes se pueden mencionar:

- El uso de Patrones de Diseño (Harrer, Pinkwart, McLaren, y Scheuer, in press; Devedzic y Harrer, 2005)
- La creación y uso de componentes reutilizables y objetos de aprendizaje (Koedinger, Suthers, y Forbus, 1999; Ritter & Koedinger, 1997; Ritter, Blessing, & Wheeler, 2003)
- La creación en grupo usando herramientas colaborativas. (Aleahmad, Aleven, & Kraut, 2008)
- El uso de herramientas de Autor (por ejemplo, Murray, Blessing y Ainsworth, 2003).

Normalmente, cada herramienta de edición y creación de ITS se centra en un tipo específico de ITS, por ejemplo, algunas herramientas se centran en prestar facilidades para el diseño de tutores basados en restricciones (Mitrovic et al., 2006) mientras que otras permiten el diseño de Tutores tipo “Model Tracing” (Blessing, Gilbert, Ourada, y Ritter, 2007). En general el objetivo de las herramientas de autor es reducir significativamente la cantidad de tiempo necesario para crear un tutor, y disminuir la cantidad de conocimientos requeridos en temáticas no relacionadas conel dominio de interés como por ejemplo la programación deprocesos soportados con técnicas de inteligencia artificial.

En esta sección se realizará una descripción de dos herramientas de autor que se encuentran disponibles para la construcción de los tres tipos de sistemas tutores abordados en este análisis comparativo, una de estás, CTAT (Cognitive Tutor Authoring Tools), es una herramienta de autor que permite la creación de tutores tipo Model-tracing y example-tracing; mientras que

ASPIRE, es una herramienta de autor que permite la creación de sistemas tutores inteligentes basados en restricciones.

Aún cuando las herramientas permiten la construcción de sistemas tutores inteligentes de distintos tipos, comparten un modelo de desarrollo que es bastante similar: Ambas herramientas han sido desarrolladas mediante un modelo de código cerrado; el desarrollo de nuevas características ha estado basado en la necesidad de soportar diferentes estudios e investigaciones realizados por los grupos académicos que están encargados del desarrollo de software de cada herramienta; el uso de ambas herramientas esta autorizado solo para propósitos académicos, descartando así los propósitos de carácter comercial; durante la ejecución de este proyecto, parte de la documentación avanzada requerida para hacer uso de las herramientas, tuvo que ser solicitada expresamente a los grupos de investigación encargados de su desarrollo. La interacción que se mantuvo con los grupos de investigación propietarios de las herramientas, permitió la posibilidad de hacer uso avanzado de las herramientas, así como la posibilidad de entregar a dichos grupos, realimentación acerca del funcionamiento del software, orientado en su mayoría a describir los malos funcionamientos con el objetivo de que dichos problemas puedan ser removidos en un futuro por parte de los propietarios de cada herramienta de software.

La descripción de las dos herramientas de autor permitirá reconocer cuales son las capacidades de cada una de estas, lo cual es un aspecto bastante importante cuando se alcanza la etapa de construcción de los sistemas tutores inteligentes. En la medida en que las herramientas de autor permitan un fácil pero avanzado desarrollo de los tutores, el resultado final resultará de mayor calidad pero con un menor esfuerzo.

3.2.1 Descripción de capacidades

Aunque teóricamente las técnicas que se presentaron en la sección anterior pueden implementarse en cualquier entorno de ejecución de programas (Java, Flash, LISP, etc), es importante resaltar que en un escenario real dicha implementación es bastante costosa en términos de esfuerzo. Analizar las capacidades de una herramienta de autor y comprender la forma en que dichas capacidades pueden ser aplicadas para crear tutores inteligentes en el dominio del control automático resulta de mayor importancia puesto que el escenario de uso de una herramienta de autor es el primer escenario que cualquier autor interesado en el dominio, abordaría (y que debería abordar) para reducir los costos del desarrollo.

En esta sección se realiza una descripción muy breve de las capacidades de las herramientas CTAT y ASPIRE. Esta descripción esta orientada a realizar un análisis de aplicabilidad que será presentado en la siguiente sección. Para obtener un conocimiento más amplio y profundo de cada una de las capacidades de estas herramientas *se sugiere al lector ingresar a los sitios web y artículos referidos durante la descripción*. Debido al largo tiempo de desarrollo que han tenido estas herramientas, la cantidad de características que presentan son bastante amplias y en algunos casos complejas. Realizar una descripción detallada de cada una de estas características esta fuera del alcance de este trabajo.

- CTAT

CTAT es una herramienta de autor para la creación de sistemas tutores inteligentes tipo model-tracing y example-tracing. Ha sido desarrollada durante los últimos 15 años por el Human-Computer Interaction Group de la universidad Carnegie Mellon. (Aleven et al., 2009; Koedinger et al., 2004). La herramienta esta disponible para descarga en su sitio web http://ctat.pact.cs.cmu.edu, y está acompañada de una documentación que en algunos casos resulta desactualizada pero que en otros resulta bastante pertinente.

CTAT soporta la creación de dos tipos de tutores: Tutores tipo Example-Tracing, los cuales pueden crearse sin necesidad de programación y Tutores Model Tracing, que requieren la programación de reglas de producción usando un lenguaje de Inteligencia Artificial conocido como JESS.

La siguiente tabla presenta las capacidades de CTAT para la creación de tutores tipo example-tracing y una breve descripción de cada una de estas.

Tabla 15. Capacidades de la herramienta CTAT respecto a los sistemas tutores example-tracing

Capacidad	**Descripción**
Creación de Interfaz gráfica de Usuario	CTAT provee un entorno de ejecución basado en Java o Flash. Para cada entorno de ejecución pone a disposición del autor un conjunto de componentes de interfaz gráfica que incluye: campos de texto, botones, lista de selección, y algunos componentes avanzados. Todos estos componentes pueden ser agregados a un panel de trabajo para crear el entorno de resolución de problema. Los componentes pueden ser

	organizados y diagramados según las necesidades del autor. Se pueden asignar nombres a los componente de la interfaz y valores iniciales. Los componentes no pueden ser extendidos. Se pueden crear de componentes que pueden ser reutilizados posteriormente.
Creación del grafo de comportamiento por medio de demostraciones	CTAT permite crear los grafo de comportamiento de manera fácil y consistente. Implementa varios protocolos de comunicación entre la interfaz gráfica y el componente conocido como el grabador de comportamientos (Behavior Recorder). El grabador de comportamiento genera los estados y la definición de las acciones a partir de las demostraciones que el autor realiza en la interfaz gráfica. El grafo de comportamiento se implementa mediante definiciones en un archivo XML.
Anotación del grafo de comportamiento con mensajes de error, ayuda y éxito.	Para cada acción correcta demostrada en la interfaz gráfica CTAT permite la asociación de mensajes de éxito y mensajes de ayuda. Para cada acción incorrecta CTAT permite la asignación de mensajes de error. Cada acción demostrada puede ser marcada como correcta, incorrecta y sub-optima.
Generalización del gráfico de comportamiento.	Por cada acción definida en el grafo de comportamiento, CTAT permite la generalización de los elemento de la tripla SAI (Selection,Action,Input). La generalización se puede realizar mediante el uso de funciones que ya existen en el entorno de ejecución de CTAT o mediante funciones personalizadas que pueden ser escritas por los desarrolladores. Estas funciones pueden ser escritas en lenguaje JAVA.
Programación de acciones que son ejecutadas por el tutor	E ciertas ocasiones se necesita que el tutor realice acciones sobre la interfaz gráfica dependiendo del comportamiento del estudiante. Estas acciones pueden ser programadas en CTAT fácilmente. Como el resto de las acciones, estas acciones son también definidas mediante una tripla selección, acción y entrada. Cuando se alcanzan determinados estados del grafo de comportamiento, el tutor puede ejecutar acciones automáticas sobre la interfaz de usuario como por ejemplo: Cambiar el valor de un campo, ocultar o mostrar un componente, mostrar un mensaje en la interfaz gráfica, o en general realizar una acción de interfaz gráfica cualquiera para ayudar al estudiante con la ejecución de dicha acción.
Configuración del funcionamiento del	El grafo de comportamiento tiene varias configuraciones que cambian la forma como se ayuda al estudiante cuando se está

grafo de comportamiento	dando tutoría. CTAT provee bastantes configuraciones relacionadas con el grafo de comportamiento como: Restricciones de Orden en los pasos de ejecución, máximo y mínimo número de repeticiones por pasos, naturaleza de los mensajes de ayuda (Inmediatos o retrasados).

La siguiente tabla presenta las capacidades de CTAT para la creación de tutores tipo model tracing y una breve descripción de cada una de estas.

Tabla 16. Capacidades de CTAT respecto a la creación de tutores basados en reglas de producción.

Capacidad	**Descripción**
Creación de Interfaz gráfica de Usuario	CTAT provee un entorno de ejecución basado en Java o Flash. Para cada entorno de ejecución pone a disposición del autor un conjunto de componentes de interfaz gráfica que incluye: campos de texto, botones, lista de selección, y algunos componentes avanzados. Todos estos componentes pueden ser agregados a un panel de trabajo para crear el entorno de resolución de problema. Los componentes pueden ser organizados y diagramados según las necesidades del autor. Se pueden asignar nombres a los componente de la interfaz y valores iniciales. Los componentes no pueden ser extendidos. Se pueden crear de componentes que pueden ser reutilizados posteriormente.
Creación del grafo de comportamiento para depuración y análisis	De la misma forma que en los tutores tipo example-tracing. CTAT permite la creación de grafos de comportamiento para efectos de depuración y análisis del modelo cognitivo que se crea con las reglas de producción. En especifico el grafo de comportamiento ayuda a investigar el numero de pasos y de habilidades que deben ser modeladas por las reglas de producción. También permite mantener la memoria de trabajo del tutor en sincronía con la interfaz gráfica para poder realizar depuraciones del modelo cognitivo de forma sencilla.
Editor de la Memoria de Trabajo	El editor de la memoria de trabajo permite visualizar en todo momento la memoria de trabajo del modelo cognitivo. Para cada estado del árbol de comportamiento permite visualizar los valores de las estructuras de datos que conforman la memoria de trabajo. Adicionalmente permite modificar el estructura de datos, agregando y eliminando variables y cambiando su valor.

Árbol de conflictos	El árbol de conflictos es una herramienta de depuración que muestra las reglas que se dispararon total y parcialmente cuando la ejecución del algoritmo “forward chaining” exploró el espacio de reglas del intérprete sistema de producción. Permite conocer si las reglas que hacen parte del modelo cognitivo están siendo activadas en los momentos que corresponde y en caso que no, cuales partes de las reglas no están siendo activadas. La exploración del algoritmo de encadenamiento hacia delante, se puede representar mediante un árbol, por esta razón CTAT permite la visualización de este árbol de forma que el autor pueda tener detalles del proceso de exploración que el algoritmo llevó a cabo.
Herramienta “Why not?”	Para cada regla que no fue activada correctamente, la herramienta “Why not” permite visualizar con detalle cada una de las expresiones booleanas de las reglas y los valores asociados a estas.
Breakpoints	CTAT permite la definición de puntos de ruptura en la ejecución del algoritmo de encadenamiento hacia delante. De esta forma la ejecución del algoritmo de detiene cuando se alcanza una regla que tiene un punto de ruptura y el autor puede explorar la memoria de trabajo usando el editor de memoria de trabajo.
Consola de JESS	La consola de Jess es una ventana que permite la interacción por medio de línea de comandos para con el intérprete Jess.

Los tutores creados en CTAT, pueden ser desplegados de diferentes maneras según el tipo de tecnología sobre la cual se implemente la interfaz gráfica. Para interfaces gráficas creadas en FLASH, CTAT incluye un conjunto de instrucciones y de herramientas que permiten crear un paquete desplegable de todos los archivos que conforman la interfaz gráfica y sus recursos asociados. Para interfaces gráficas implementadas en lenguaje Java, varios mecanismos están puestos a disposición del autor para realizar despliegue de las interfaces mediante la tecnología Java Web Start. De los diferentes mecanismos de despliegue, el que tiene mayor importancia es el sistema de administración de aprendizaje (LMS) TutorShop. Tutorshop permite a los autores gestionar cursos en línea, en la forma tradicional que la mayoría de LMSs lo permiten. Sin embargo Tutorshop esta diseñado para desplegar los sistemas tutores creados en CTAT. De esta forma, un autor puede ingresar al sistema de administración de contenido y publicar en línea los paquetes que corresponden a cada uno de los tutores que fueron creados usando

CTAT. Tutorshop (https://tutorshop.web.cmu.edu/) brinda al autor la posibilidad de escoger un esquema de selección de los problemas de forma que cuando un estudiante ingrese al sistema, la secuencia en que los problemas son presentados, no sea la misma siempre y varíe de estudiante a estudiante. Tutorshop adiciona a los tutores creados con CTAT las capacidades de lazo exterior (outer-loop) típicas de un sistema tutor inteligente. Uno de los esquemas de selección más interesantes resulta ser el esquema Mastery Learning puesto que ejecuta una versión del algoritmo Knowledge Tracing.

Utilizando en conjunto de tutores creados con CTAT con Tutorshop, se pueden brindar a los estudiantes problemas que incluyan tutoría basada en mensajes de éxito, error y de ayuda, con las funcionalidades típicas de lazo interno y lazo externo de un sistema tutor inteligente.

- ASPIRE

ASPIRE es una herramienta de autor y servidor de despliegue que apoya el proceso de desarrollo de los sistemas tutores inteligentes basados en restricciones, mediante la automatización de algunas tareas y el acceso a herramientas que facilitan la ejecución de tareas que no son automatizables.ASPIRE consta de ASPIRE-Autor, el servidor de autoría, que sirve como ambiente de desarrollo de los sistemas tutores, y ASPIRE-Tutor, el servidor de tutoría, que permite desplegar a los estudiantes los sistemas tutores creados con ASPIRE-Author. ASPIRE-Autor facilita tareas como el definición de los conceptos involucrados en el dominio de interés, las tareas que los estudiantes va a realizar, así como para especificar los problemas y sus soluciones.

El proyecto ASPIRE es financiado por el“e-Learning Collaborative Development Fund” de Nueva Zelanda y es desarrollado y mantenido por el “Intelligent Computer Tutoring Group” de la Universidad de Canterbury en Nueva Zelanda. Aspire-Author y Aspire-Tutor son herramientas web, que pueden ser encontradas en la siguiente dirección http://aspire.cosc.canterbury.ac.nz/ y cuya documentación esta disponible a través de manuales de usuarios publicados en el sitio web y a través de diferentes artículos científicos publicados por el grupo de investigación que realiza su mantenimiento y desarrollo(Martin et al., 2007; Antonija Mitrovic et al., 2006, 2008, 2009, 2007).

La siguiente tabla presenta las capacidades de ASPIRE-Author para la creación de tutores tipo constraint-based y una breve descripción de cada una de estas.

Tabla 17. Capacidades de la herramienta de autor ASPIRE.

Capacidad	Descripción
Definición de la estructura del dominio	ASPIRE permite definir un concepto conocido como el dominio. El dominio se refiere a una estructura general que todos los problemas para un campo de conocimiento especifico van a tener. Esta estructura puede corresponder a dos tipos: Procedimental o No procedimental. Cuando un dominio se define como de tipo procedimental, los problemas que el estudiante va a resolver pueden estar compuestos por uno o más pasos. Cuando el dominio es No procedimental, los problemas solo están compuestos por un paso.
Definición de la ontología del dominio	ASPIRE permite a los autores la definición de una ontología que modela el conocimiento especifico del dominio de interés. Por medio de esta ontología, el autor puede definir relaciones de composición y de herencia entre conceptos. También puede definir tipos de datos asociados a los conceptos en caso que aplique. Las definiciones de los tipos de dato y las relaciones entre conceptos son usados posteriormente por ASPIRE para generar de forma automática, un conjunto de restricciones que se aseguran que las respuestas de los estudiantes cumplan con las restricciones de composición, herencia y tipos de datos definida para los conceptos.
Definición de las estructuras del enunciado del problema y de las soluciones.	Durante la definición de la estructura del dominio, solamente se define de cuantos pasos consta una solución. ASPIRE también permite al autor definir la estructura de cada paso. Para realizar esto, ASPIRE permite definir componentes de solución. Una componente de solución corresponde a un conjunto de conceptos tomados de la ontología mediante los cuales se resuelve una fracción del paso de un problema. De esta forma un paso puede tener uno o muchos componentes de solución. Un componente de solución puede tener uno o muchos conceptos mediante los cuales se completa el componente. ASPIRE permite seleccionar conceptos que hayan sido definidos en la ontología, y asignarlos a los componentes de solución y a los pasos del problema.
Definición de la	Los sistemas tutores inteligentes creados con ASPIRE

interfaz de usuario para la resolución de problemas.	pueden tener interfaces gráficas de usuario construidas en HTML o en Java. ASPIRE permite que el autor defina que tipo de interfaz gráfica desea utilizar. Para cada paso se puede definir cualquiera de los dos tipos de manera que se pueden tener unos pasos implementados usando HTML y otros usando Java. La interfaces de usuario HTML son automáticamente creadas por ASPIRE a partir de la asignación de conceptos a las componentes de solución. Las interfaces de usuario Java deben ser creadas específicamente por los autores siguiendo unas guías de desarrollo.
Definición de los problemas y soluciones.	Una vez se ha definido la estructura de los problemas, tanto de sus enunciados como de sus soluciones. ASPIRE le permite al autor crear instancias de problemas. Para crear una instancia de problema el autor proporciona valores a los enunciados de los problemas y proporciona valores a las componentes de solución. Debido a que todos los problemas tienen la misma estructura, el proceso de definición de las soluciones se torna fácil.
Generación y edición de restricciones de tipo Sintáctico y semántico.	A partir de la definición de conceptos en la ontología del dominio, de la asociación de conceptos a las componentes de solución y de las instancias de solución que se hayan definido, ASPIRE genera un conjunto de restricciones semánticas y sintácticas de forma automática. Este conjunto de restricciones puede ser editado por el autor para personalizar sus definiciones a fin de complementar o mejorar su funcionamiento.
Definición de funciones de dominio.	Las restricciones generadas por ASPIRE suelen hacer uso de funciones que están definidas por defecto en el entorno de ejecución. Sin embargo, un autor puede en algunos casos necesitar de funciones que no están definidas en el entorno de ejecución. ASPIRE permite realizar la definición de estas funciones, de forma tal que un autor que tenga conocimiento del lenguaje LISP puede escribir funciones en este lenguaje e incluir el uso de estas funciones en la definición de las restricciones.
Herramienta para la Depuración de restricciones	Para todas las restricciones generadas, ASPIRE permite realizar un proceso de depuración detallado. Este proceso permite la evaluación de una restricción especifica ante una solución de problema especifica. Durante el proceso de depuración ASPIRE entrega detalles de los valores

	calculados para las expresiones booleanas que definen la restricción. De esta forma un autor puede hacer revisión de restricciones que presenten fallos y posteriormente hacer ediciones sobre la especificación de las restricciones con el objetivo de resolver los fallos.

Aspire-Tutor corresponde al servidor de publicación de los tutores creados con Aspire-Author y brinda al sistema tutor inteligente las capacidades de lazo externo. En especifico las capacidades de Aspire-Tutor se resumen en la siguiente tabla:

Tabla 18. Capacidades de la herramienta Aspire-Tutor. Esta herramienta sirve para el despliegue de los STI creados.

Capacidad	Descripción
Gestión de Grupos de estudiantes	Aspire tutor permite la creación de cuentas de usuario para estudiantes, asignación de usuarios a grupos y la asignación de tutores a los grupos. De esta forma un estudiante con una cuenta de usuario puede acceder a todas los tutores asignados a su grupo.
Definición de las configuraciones para el modulo pedagógico.	Aspire tutor permite que durante el despliegue de los problemas, los estudiantes puedan seleccionar los problemas de forma manual y/o que el sistema seleccione los problemas por el estudiante. También permite definir el tipo de mensajes que recibirá el estudiante cuando el tutor proporcione ayuda, pudiendo un autor seleccionar entre 6 niveles diferentes de ayuda, que recorren desde ayudas rápidas hasta mostrar la solución del problema. También se puede configurar otras características del comportamiento de los tutores como el máximo número de intentos permitidos, si o no se requiere que se resalten los componentes de interfaz gráfica en los cuales existen errores, y otras configuraciones adicionales de menor relevancia.

3.2.2 Análisis de aplicabilidad

En esta sección se presenta un análisis de aplicabilidad de las herramientas de autor consideradas en la sección anterior. El análisis de aplicabilidad busca presentar las ventajas y desventajasde las capacidades que cada una de las herramientas brinda al autor en términos de construcción de un sistema real y del producto final que se obtiene.

El análisis de aplicabilidad se ha realizado a partir del uso detallado y profundo de las herramientas. Este uso se dio como pruebas de concepto antes de la implementación de los sistemas tutores finales y como uso avanzado durante su implementación. Por lo tanto esta sección recoge el conocimiento obtenido durante etapas iniciales del proyecto y durante etapas finales de implementación.

La razón por la cual se presenta esta análisis en esta sección es:Al adicionar las ventajas y desventajas técnicas que existen durante la etapas de implementación de los sistemas, las diferencias entre estos se hace más notoria. Esta diferencia permite una distinción más fácil del comportamiento real que tiene un sistema tutor implementado respecto al otro. El reconocimiento de estas diferencias con la antelación suficiente, le permitirá a futuros desarrolladores tomar decisiones a tiempo acerca de cual formalismo de modelamiento y/o herramienta de autor utilizary si de seguir o no las recomendaciones establecidas en este trabajo.

Para realizar el análisis de aplicabilidad se han clasificado las capacidades de cada una de las herramientas de autor en categorías. Cada categoría puede estar soportada por una o más capacidades de la herramienta. Para cada categoría se ha evaluado el grado de cumplimiento en una escala que solo tiene dos valores: SI o NO. Para cada categoría se han identificado las ventajas y desventajas que presenta el producto final o la herramienta de autor.

Las categorías definidas se presentan a continuación con una breve descripción de su significado:

- Funcionalidades para creación de interfaz gráfica.

  Esta categoría se refiere a la funciones que la herramienta de autor presta para poder diseñar, implementar y poner en funcionamiento una interfaz gráfica apropiada que represente los datos y procedimientos involucrados en el problema de forma correcta y usual para los estudiantes. La creación de la interfaz de usuario sobre la cual los estudiantes interactúan, es un aspecto de mayor importancia en el desarrollo de sistemas tutores inteligentes.Su grado de importancia es tan alto que en la arquitectura general presentada en el capítulo 1, uno de los cuatro módulos corresponde precisamente a la interfaz gráfica. La interfaz gráfica debe poder ser generada de acuerdo a varios principios de buenas prácticas, entre ellos los presentados en el

capítulo dos. Sin embargo dos aspectos adicionales tienen que existir en la interfaz gráfica:

1. La interacción que los estudiantes tienen con la interfaz gráfica tiene que ser reconocida por los estudiantes. Esto significa que la interacción debe darse en la forma usual en que los estudiantes interactúan con los datos que manejan.
2. La interfaz gráfica debe permitir la flexibilidad requerida para generar suficientes pasos observables a lo largo del desarrollo de un problema. Solo a través de la observación de ciertos pasos se pueden identificar diferentes estrategias de resolución del problema.

- Funcionalidades para crear la representación que define en el formalismo teórico.

  Esta categoría se refiere a las características que la herramienta de autor presenta para realizar la creación y puesta en funcionamiento de los algoritmos y técnicas específicas de cada uno de los formalismos que cada herramienta apoya. Esto incluye la implementación de procesos de tipo de general y la puesta a disposición de herramientas para que el autor pueda crear y personalizar los procesos, datos y estructuras que son de tipo especifico para el dominio de interés.

- Funcionalidades para generalizar y hacer mas avanzado la representación del formalismo

  En algunos casos se requiere que la representación que utiliza cada formalismo pueda ser modificado y complementado con funciones, procesos y estructuras de datos que han de la representación inicial una base de conocimiento más completo, generalizada y/o especializada. Esta categoría se refiere a las funciones que la herramienta de autor presta para permitir que las representaciones iniciales puedan modificarse de tal forma que tras la realización de dichas modificaciones pueda considerarse que la representación es mucho mas avanzada.

- Modificación de la Interfaz gráfica a partir del reconocimiento de ciertos estados del problema.

La comunicación entre la interfaz gráfica y los modelos de procesamiento del sistema tutor inteligente es una característica primordial para cumplir con el objetivo de implementar interfaces gráficas dinámicas, cuyo dinamismo esta basado en el comportamiento del estudiante. Las herramientas de autor que en conjunto con los marcos de trabajo, proporcionan estas funcionalidades presentan una mejor alternativa que aquellas que no lo presentan puesto que el producto final podrá incluir interfaces dinámicas como una funcionalidad que soporte la experiencia de aprendizaje de los estudiantes.

- Disponibilidad de Herramientas de Depuración

Esta categoría agrupa todas las funcionalidades que una herramienta de autor proporcione, para diagnosticar y corregir problemas que ocurran en los modelos creados. Aún cuando las herramientas de autor normalmente están orientadas a personas que no tienen un gran conocimiento en técnicas de programación, las herramientas de depuración pueden realizarse de tal manera que no resulten difíciles de controlar y cuyo uso resulte intuitivo dentro del contexto que el formalismo genera.

- Mecanismos de extensibilidad

Los mecanismos de extensibilidad se refieren a todas aquellas funcionalidades que permiten al autor crear nuevas representaciones a partir de las existentes o a partir de componentes disponibles en plataformas externas que sean compatibles. La extensibilidad es un aspecto de gran relevancia puesto que es el último mecanismo que un autor puede surtir para implementar funcionalidades especificas en el sistema tutor inteligente final. Si los mecanismos de extensibilidad no existen y adicionalmente las configuraciones y funcionalidades existentes en la herramienta de autor no son suficientes para lograr un determinado requerimiento especifico del dominio de conocimiento en el que se trabaja, el autor no tendrá otros caminos adicionales que resignarse de usar la funcionalidad ó cambiar de plataforma y/o marco de trabajo.

- Disponibilidad de Código fuente.

Asociados a los mecanismos de extensibilidad y aunque la disponibilidad del código

fuente no representa una categoría de funcionalidades especifica, la posibilidad de acceder al código fuente sobre el cual un determinado marco de trabajo funciona es de vital importancia. En algunos casos la disponibilidad del código fuente permite aliviar los problemas causados por la falta de documentación. Sin embargo para aplicaciones y marcos de trabajo cuya complejidad es demasiado alta, este alivio no resulta tan importante.

- Disponibilidad de Documentación

La disponibilidad de documentación acerca del uso de las herramientas de autor es un aspecto de total importancia. De todos los aspectos que no son de carácter técnico, la disponibilidad de documentación es el más importante. A partir de la documentación los autores pueden conocer el correcto uso de las herramientas, sus funcionalidades y posibilidades. La disponibilidad de la documentación reduce considerablemente los tiempos de desarrollo de los productos finales debido a que evitan que autor ingrese en ciclos, normalmente muy largos, de lectura de código fuente o de búsqueda de la información pertinente a través de otras fuentes como expertos o grupos de discusión.

- Modularidad de la Solución

Una vez finalizado un sistema tutor inteligente, es importante establecer cuales componentes del sistema finalizado pueden ser reutilizados para la construcción de nuevos sistemas. Básicamente esta posibilidad de reutilización depende en alto grado de la modularidad que presente la solución final. Esta modularidad puede darse a nivel de los componentes generales que usa el formalismo para su ejecución o a nivel de los componentes que el autor construye para implementar la tutoría en el dominio de conocimiento de interés. Dicho esto, desde el punto de vista del autor, es más valioso para las etapas de desarrollo que todas aquellas construcciones realizadas por el autor puedan ser utilizadas por el autor nuevamente en la implementación que ocurra en etapas posteriores.

En el apéndice D se presenta la tabla comparativa de las categorías descritas, junto con los comentarios más relevantes de cada categoría que muestran las ventajas y desventajas de

cada herramienta para cada categoría. En la última columna de la tabla se muestra la herramienta de autor que presenta una mejor valoración.

- Resumen

En esta sección se presentó el conjunto de funcionalidades más relevantes de cada herramienta de autor. Se describió con brevedad cada una de las funcionalidades y se presento un análisis comparativo basado en categorías. Aún cuando las funcionalidades son bien distintas en su implementación y forma de uso, los objetivos generales que cada funcionalidad intenta soportar son los mismos y por lo tanto se abre la posibilidad de comparar con mayor facilidad.

Del análisis comparativo se puede concluir que hay una mayor preferencia hacia la herramienta de autor CTAT. Sin embargo la preferencia no es lo suficientemente amplia para descartar la implementación de sistemas tutores inteligentes en la herramienta ASPIRE.

En la siguiente sección se describirá con ayuda de unas referencias, la similitud que existe en términos de funcionamiento, entre los tutores Model-Tracing y Example-Tracing. Esta similitud cobra bastante importancia debido a que en el supuesto caso que se implementarán estos dos tipos sistemas tutores inteligentes, siendo su comportamiento en tiempo de ejecución tan similar, los estudiantes no podrían percatar ninguna diferencia apreciable y por lo tanto la comparación experimental carecería de sentido en términos de las opiniones que los estudiantes puedan tener acerca del comportamiento de los tutores.

Realizar la implementación de los tutores inteligentes usando tanto la herramienta ASPIRE como la herramienta CTAT, aunque resulta más costoso en términos de esfuerzo, permite realizar una comparación de las verdaderas posibilidades de cada una de las herramientas, ante un conjunto de requerimientos específicos que vienen otorgados por los problemas seleccionados y el conjunto de las interfaces gráficas diseñadas.

3.3 Análisis comparativo final.

En (Antonija Mitrovic, Koedinger, & Martin, 2003) se realizó un análisis comparativo entre las técnicas de modelamiento tipo model-tracing y los tutores tipo contraint-based. Este reporte es uno de los más completos en su categoría debido a que fue realizado por los investigadores líderes en cada una de las técnicas de modelamiento que involucró el análisis comparativo.

El comparativo realizado por estos autores analiza 11 características representativas de cada una de las técnicas de modelamiento. En la siguiente figura se muestran las 11 características y el nivel de cumplimiento o el valor de cada característica para cada tipo de técnica. Para comprender en detalle cada una de las propiedades se sugiere la lectura de la referencia presentada en el párrafo anterior.

Tabla 19. Once propiedades que pueden ser comparadas entre los tutores tipo model-tracing y constraint-based

Property	Model Tracing	Constraint-Based Modeling
Knowledge representation	Production rules (procedural)	Constraints (declarative)
Cognitive fidelity	Tends to be higher	Tends to be lower
What is evaluated	Action	Problem state
Problem solving strategy	Implemented ones	Flexible to any strategy
Solutions	Tend to be computed, but can be stored	One correct solution stored, but can be computed
Feedback	Tends to be immediate, but can be delayed	Tends to be delayed, but can be immediate
Problem-solving hints	Yes	Only on missing elements, but not strategy
Problem solved	'Done' productions	No violated constraints
Diagnosis if no match	Solution is incorrect	Solution is correct
Bugs represented	Yes	No
Implementation effort	Tends to be harder, but can be made easier with loss of other advantages	Tends to be easier, but can be made harder to gain other advantages

De las once características, la que merece mayor atención para efectos de este proyecto, es la primera. Representación del conocimiento. La representación del conocimiento es muy importante porque define los alcances y limitaciones de una determinada técnica respecto a un dominio de conocimiento especifico.

Para el dominio de la teoría de control, se presentó en la sección 3.1 un ejemplo de un problema típico el cual fue modelado usando las tres técnicas. Se presentaron al final de la sección, varias razones establecidas por la vía de la generalización, mediante las cuales se establece que las tres técnicas de modelamiento son suficiente generales para abarcar los problemas que conciernen el dominio de conocimiento de interés. Sin embargo esta generalización no es del todo útil para realizar una definición de diferencias, similitudes, ventajas y desventajas que conlleven a la consideración de solo dos tipos de técnicas en la etapa de desarrollo.

Se realizó entonces un recorrido detallado por cada uno de los conceptos, modelos y procesos matemáticos y de ingeniería que hacen parte de la teoría impartida a los estudiantes. A partir del recorrido se identificaron unos requerimientos de modelamiento de conocimiento que el dominio impone a cada una de las técnicas. Para cada uno de los requerimientos de modelamiento se establecieron algunos ejemplos de tareas que el estudiante debe resolver y en las cuales el tutor debe estar en la capacidad de proporcionar ayuda. Luego para cada ejemplo se realizó un breve ejercicio de modelamiento con cada una de las tres técnicas para validar la dificultad o pertinencia de una técnica ante un escenario más específico y real que las generalizaciones presentadas en la sección 3.1. El resultado completo puede observase en los ANEXOSC y D. Es de notar que los breves ejercicios de modelamiento mostrados en los apéndices pueden ser difíciles de interpretar a primera vista, sin embargo realizar una definición de cada una de las suposiciones requeridas para hacer más explicativo los ejemplos (como se hizo en la sección 3.1 en donde se definen las variables, las funciones y el significado de cada una de estas), tomaría una cantidad de espacio considerable. Como guía general para interpretar estos ejemplos se sugiere considerar que si se hace un correcto modelamiento de los parámetros de entrada que utiliza una regla, restricción o acción demostrada; se puede diseñar funciones que calculen información a partir del modelo. Por ejemplo,para el modelamiento de diagramas de bloques una representación útil son los grafos y las definiciones en lenguaje XML. A partir de un modelamiento correcto que se haga de los diagramas, se pueden diseñar funciones que calculen información a partir del modelo, como dependencias, o expresiones.

Para efectos de referencia se presentan en esta sección el listado de requerimientos identificados

Tabla 20. Requerimientos mínimos en términos de modelamiento de conocimiento para un tutor que soporte el currículo completo y las operaciones básicas y avanzadas para el dominio de fundamentos de sistemas de control

Categoría	**Requerimiento: "Que el tutor Pueda"**
Manipulación de Diagramas de Bloque	a) Identificar las dependencias y causalidades entre señales b) Identificar los efectos de los bloques sobre las señales c) Calcular expresiones para una determinada señal en términos de otras señales y bloques. d) Asociar un sentido físico a las señales y bloques e) Establecer equivalencias ó transformaciones entre diagramas con la ayuda de expresiones matemáticas
Manipulación de	a) Realizar y/o comprender operaciones típicas algebraicas como expansión,

expresiones matemáticas	factorización, reemplazo, reducción, simplificación, despeje de variables, multiplicación y división por el mismo factor, b) Realizar el cálculo de Transformadas directas e inversas de Laplace. c) Realizar el cálculo de Fracciones Parciales d) Realizar el cálculo de Límites, Integrales, Derivadas, Productorias, Sumatorias. e) Realizar análisis del efecto de la variación de variables independientes sobre variables dependientes
Manipulación de gráficas	a) Realizar y/o verificar la Generación de gráficas en el dominio del tiempo y de la frecuencia a partir de la expresión matemática. b) Generar expresiones matemáticas a partir de gráficas parametrizadas en el dominio del tiempo y la frecuencia. c) Identificar medidas y de parámetros como características de las gráficas: rangos (medidas de tiempo y amplitud), pendientes, forma de la gráfica (exponencial, senoidal, sub-amortiguado)
Abstracción de Contenidos.	a) Mantener una relación de conceptos por medio del uso de memoria. b) Memorizar procedimientos

Esta tabla es de gran importancia porque representa las habilidades de tipo general que los estudiantes deben aprender y demostrar durante su experiencia de aprendizaje y por lo tanto constituyen las habilidades mínimas que un sistema tutor inteligente debe tener o debe poder llegar a tener para poder garantizar, que con un mismo tipo de tutor, se puede soportar todo el currículo.

Con el análisis realizado, en los apéndices C y D, sobre los requerimientos de modelamiento que impone el dominio se observa un preferencia sobre elegir las técnicas Example-Tracing y Contraint-Based como las técnicas sujetas a implementación.

En términos generales estas preferencias están establecidas a partir del menor esfuerzo en tiempo de desarrollo con que estas dos herramientas cuentan respecto a la técnica model-tracing. Adicionalmente para ciertos requerimientos de modelamiento, la facilidad que representa hacer las demostraciones sobre la interfaz gráfica comparada con la conocida dificultad impuesta por la técnica de escribir reglas de producción, hacen de la técnica example-tracing una opción que requiere el menor costo en términos de esfuerzo pero que brinda unas funcionalidades de igual nivel que los demás métodos. Especialmente para el modelamiento de los criterios de diseño que están basados en especificaciones, la técnica basada en restricciones presenta un buen comportamiento y facilidad de modelamiento.

Para concluir el análisis se hace una discusión de la pertinencia, efectos y de la necesidad que las 11 propiedades mostradas, tienen respecto al campo de estudio de los sistemas de control. La idea es evaluar si el hecho que existan diferencias en las propiedades de los tipos de tutores, causa algún efecto negativo o positivo respecto a la posible implementación de sistemas tutores inteligentes en el área de control. Este análisis se presenta en la siguiente tabla:

Tabla 21. Análisis del impacto de cada una de las once propiedades que caracterizan los tutores cognitivos respecto a los requerimientos, necesidades de desarrollo, funcionalidades deseadas y conveniencia del estudio comparativo.

Knowledge representation	Los tres paradigmas son capaces de modelar el conocimiento requerido según se mostró en la tabla de requerimientos. Todos los paradigmas tienen requerimientos en los que son mejor opción y también requerimientos en los que no son muy buena opción. Todos los paradigmas requieren el uso de funciones de alto nivel que no son fáciles de desarrollar.
Cognitive fidelity	La fidelidad cognitiva se vuelve más indispensable en el momento de hacer análisis de habilidades (Skills), por ejemplo para las funcionalidades de lazo externo de un tutor inteligente. Sin embargo, tanto los sistemas tutores Example Tracing, como los sistemas tutores basados en restricciones tienen otros sistemas que dan soporte al lazo externo. Si implementar una fidelidad cognitiva alta viene asociada con un gran esfuerzo en desarrollo se recomienda el uso de técnicas que requieran menos esfuerzo puesto que para estás técnicas los análisis de habilidades ya se encuentran disponibles e integrados. De esta forma se define una preferencia sobre los tutores tipo example-tracing y los basados en restricciones.
What is evaluated	Para la todos los requerimientos de conocimiento establecidos en la tabla, hay acciones de interfaz gráfica que se pueden diseñar para que el estudiante proporcione sus respuestas, de esta forma siempre hay acciones que evaluar. Para todos los problemas siempre existirá almenos un estado de solución, dicho estado contará con algunas características que pueden ser evaluadas por usando el modelamiento basado en restricciones. De esta forma no hay una preferencia específica.

Problem solving strategy	Específicamente para las temáticas de fundamentos en control el número disponibles de estrategias no es muy alto. Por lo tanto utilizar una técnica que solo detecta soluciones correctas cuando la estrategia se codifica específicamente en la base de conocimiento, no es un problema. De esta forma habría preferencia sobre los tutores tipo example-tracing debido a la facilidad que presentan a la hora de definir nuevas estrategias.
Feedback	La capacidad que cada paradigma tiene para proveer realimentación sobre la interacción y mensajes de ayuda no es un aspecto que específicamente afecte negativamente el dominio. En este caso, este aspecto afecta la tasa con la que los estudiantes aprenden. En general se considera que la realimentación inmediata es más favorable para el aprendizaje que la realimentación retrasada. Sin embargo, para efectos del estudio comparativo conviene realizar la selección de técnicas que brinden un sistema de realimentación diferente, para poder evaluar el efecto en los estudiantes. Esto lleva a que la selección de paradigmas para implementación necesariamente debe incluir los sistemas tutores basados en restricciones pues es el único formalismo abordado que provee realimentación atrasada.
Problem Solved	Independiente de si el problema finaliza cuando se ejecuta la regla de producción marcada como DONE, o cuando no hay restricciones violadas ó cuando se alcanza el estado DONE en el grafo, la forma como detecta la finalización del problema no tiene ninguna implicación respecto al dominio de interés. Tampoco el dominio de los sistemas de control imponen algún requerimiento especial respecto a este aspecto.
Solutions	Los tres formalismos permiten el cálculo y el almacenamiento de respuestas por lo tanto no es un factor de discriminación. Sin embargo se recomienda que las respuestas sean calculadas para poder garantizar un buen nivel de generalización.

Problem-solving hints	Solamente el modelamiento basado en model-tracing puede calcular las ayudas y pistas de forma dinámica a partir de conocer el estado actual de estudiante. Sin embargo los otros dos tipos de modelamiento soportan niveles de incrementales de ayuda por lo cual es viable la implementación de esta característica también. Sin embargo, se toma mayor provecho de la característica que presenta el modelamiento model-tracing cuando existen definiciones de reglas que no corresponden a pasos observables. De esta forma todas las ayudas asociadas a estas reglas "ocultas" se convierten en niveles incrementales de ayuda. En los casos en que las reglas solo modelan los pasos observables, la diferencia no es grande respecto a los otros tipos de modelamiento.
Diagnosis if no match	Este aspecto es especialmente peculiar para los tutores basados en restricciones. Resulta una ventaja cuando se modelan todas las características que una solución correcta debe tener y la respuesta del estudiante, sin importar como haya sido calculada, es reconocida como correcta. Sin embargo, esta ventaja se convierte en una desventaja muy grande cuando no todas las características de una solución correcta se evalúan en las restricciones, por ejemplo por falta de exactitud en el modelo, lo cual es muy común cuando se inicia en la práctica de esta técnica. Para este escenario, respuestas incorrectas pueden ser reconocidas como correctas, lo cual es un gran error en términos pedagógicos y representa un gran riesgo al realizar este tipo modelamiento. Por supuesto que hay controversia al respecto, sin embargo desde el punto de vista del autor, es especialmente mas dañino que un estudiante que resuelve un ejercicio de forma incorrecta, crea que lo hizo correctamente a que no pueda resolver el problema, hecho que pasaría en un modelamiento inexacto realizado con model-tracing o example-tracing.

Bugs represented	Los tutores tipo example-tracing tienen una buena evaluación en este aspecto debido a que la creación de una librería de errores es bastante más fácil que la creación de una librería de reglas de errores. Adicionalmente según como se mostró en la sección 3.1, estos tutores pueden detectar errores típicos, soluciones correctas, soluciones sub-optimas y soluciones incorrectas de forma generalizada, siendo de mayor ventaja la detección generalizada de acciones incorrectas puesto que se posibilita el cambio de los mensajes estándar de error, por mensajes que sean más específicos respecto al paso de solución del problema.
Implementation effort	De acuerdo a la tabla de requerimientos presentada, se tienen una preferencia por los tutores tipo Example-Tracing y para los tutores basados en restricciones debido a su bajo esfuerzo requerido en implementación.

Finalmente se decide la selección de los tutores tipo example-tracing y constraint-based, para que alcancen las etapas de desarrollo y pruebas debido a todos los comentarios y análisis presentados en las secciones 3.1,3.2, 3.3, Anexos D y E.

4. Implementación de Sistemas tutores inteligentes para control.

En este capítulo se realizará una descripción detallada de muchos aspectos relacionados con la implementación de los dos tipos de tutores seleccionados. La descripción inicia por medio de la presentación de las metodologías típicas de implementación sugeridas por los investigadores lideres en el campo de los sistemas tutores inteligentes. Para cada tipo de tutor hay una metodología diferente y por lo tanto se requiere la ejecución de actividades diferentes. Para algunas de las actividades dentro de cada una de las metodologías, se hace referencia a las dificultades que dichas actividades puedan presentar o las decisiones tomadas en cada una de las actividades. Cuando es necesario también se mencionan limitaciones por parte de las herramientas utilizadas respecto a los requerimientos de diseño que se definen para el dominio de interés.

Luego se presentan las interfaces gráficas que fueron diseñadas durante la etapa de concepción de los tutores. Estas interfaces gráficas fueron diseñadas para algunos de los problemas que tuvieron el escalafón más alto según lo presentado en el capítulo 2. Para cada uno de los problemas se presentan las decisiones de diseño tomadas, se realiza su argumentación y también se describen algunas funcionalidades que se definieron para el comportamiento de los tutores.

Tras la etapa de concepción, varios componentes de software tuvieron que ser desarrollados para soportar tanto los requerimientos de diseño de interfaz de usuario, como requerimientos de funcionalidad y requerimientos de calidad de la solución de software. Para cada uno de los componentes o artefactos de software que fueron creados se realiza una descripción breve, con el fin de dar a entender al lector el esfuerzo en desarrollo que para algunos dominios de conocimiento debe realizarse si se quiere surtir una etapa completa de implementación de sistemas tutores inteligentes creados con las herramientas abordadas en este trabajo.

Aspectos específicos de implementación de cada uno de los tipos de sistemas tutores inteligentes son también descritos en este capítulo. Debido a que se ha realizado un uso avanzado de las herramientas de autor, es importante describir las funcionalidades especificas que requieren este uso avanzado y detallar la forma en que se realiza para permitir que futuros desarrollos puedan beneficiarse de la documentación realizada.

Por último se presenta una discusión acerca de los tiempos de desarrollo experimentados durante la implementación realizada en este trabajo, respecto a los tiempos documentados por otros desarrolladores en el área. Una discusión acerca de los hechos más importantes experimentados durante la etapa de desarrollo se presenta a forma de resumen para finalizar el capítulo.

4.1 Metodologías de Implementación

En las siguientes secciones se presentarán las metodologías de implementación típicas de los tutores tipo example-tracing y los tutores basados en restricciones. Se describen cuales resultados de ciertas actividades que hacen parte de la metodología típica para la creación de sistemas tutores inteligentes, fueron también reutilizadas para la creación de los sistemas tutores basados en restricciones. Se describen las actividades que resultan muy generales en ambas metodologías y se describe la forma en como se logro la especificidad suficiente para ejecutar dichas actividades.

4.1.1 Sistemas Tutores tipo Example Tracing

Para la implementación de los sistemas tutores tipo example-tracing, se ha puesto en práctica la metodología descrita en (Aleven et al., 2009). Está metodología ha sido bastante probada por los investigadores líderes de este tipo de sistemas tutores inteligentes y se muestra a continuación:

1. Definir los objetivos Instruccionales

2. Identificar problemas y categorías de problemas para los cuales se proveerá la tutoría

3. Realizar Análisis Cognitivo de Tareas (Cognitive Task Analysis) mediante ejercicios como Think-Aloud o el análisis de factores de dificultad.

4. Diseñar y desarrollar el tutor.

    a. Diseñar y crear la interfaz gráfica de usuario

    b. Para cada problema (o categoría)

i. Demostrar el comportamiento correcto e incorrecto

ii. Generalizar el grafo por medio de fórmulas y anotar el grafo de comportamiento utilizando mensajes de ayuda.

iii. (Opcional) Utilizar mecanismos de producción en masa basados en plantilla para crear múltiples problemas que presenten árboles isomorfos (árboles con las mismas estructuras pero con diferentes valores en las hojas)

c. Organizar el currículo y crear los archivos del currículo.

5. Desplegar una versión prototipo.

6. Ejecutar una prueba piloto.

7. Iterar todo el procedimiento.

8. Desplegar la versión final del tutor.

Aunque inicialmente esta metodología es bastante completa, también es bastante general. Revisando con detenimiento cada uno de los pasos, se puede evidenciar que ciertas actividades a ejecutar son bastante generales y requieren una metodología propia de ejecución. Por ejemplo para poder realizar la selección de los problemas se requiere una metodología que guíe la selección de dichos problemas. Para este trabajo, dicha metodología se presento en el capítulo 2.

Actividades como el diseño de la interfaz gráfica y el diseño de los mensajes de ayuda no resultan de ejecución directa, se requiere la integración de varias fuentes de conocimiento y de tener claras las implicaciones de utilizar diferentes estrategias en caso que existan. La actividad de generalización del grafo utilizando formulas puede también resultar en una actividad que requiera desarrollo específico para el dominio de conocimiento abordado puesto que en algunos casos las formulas y funciones disponibles en la herramienta no son suficientes para generalizar el grafo de la manera adecuada, por ejemplo, si se requiere alguna de las funciones que se presentaron en la tabla del apéndice D estas funciones deben ser desarrolladas por el autor.

4.1.2 Sistemas Tutores basados en restricciones.

Para la implementación de los sistemas tutores basados en restricciones, se ha puesto en práctica la metodología descrita en . Está metodología ha sido bastante probada por los investigadores líderes de este tipo de sistemas tutores inteligentes y se muestra a continuación:

1. Modelamiento de la estructura de dominio
2. Creación de la ontología de dominio
3. Modelamiento de las estructuras de problemas y soluciones
4. Diseño de la interfaz de los estudiantes
5. Adición de problemas y soluciones
6. Generaciónde restricciones de sintaxis
7. Generación de restricciones semánticas
8. Despliegue el dominio

Es de notar que todas las actividades propuestas por esta metodología están soportados por la herramienta de autor ASPIRE. Específicamente ASPIRE utiliza la mayoría de la información que se provee en la mayoría de los pasos para alcanzar dos objetivos:

1. Generar automáticamente las restricciones sintácticas y semánticas a partir de la ontología del dominio.
2. Generar una interfaz gráfica basada en HTML de forma automática con el fin de reducir el costo de desarrollo de las interfaces gráficas.

Algunas actividades de la metodología presentada para la creación de sistemas tutores tipo exmaple-tracing, también es aplicable a los sistemas tutores basados en restricciones. Estas actividades corresponden a la actividad 1, 2 y 3. De esta forma los resultados que se obtengan tras ejecutar dichas actividades también forman un insumo para iniciar las etapas de implementación de los sistemas tutores basados en restricciones.

4.2 Diseño de interfaces gráficas

Al aplicar la metodología presentada en el capítulo 2, se obtuvo un conjunto de 15 problemas que tienen el puntaje de calificación máximo. Debido a que el diseño de interfaces gráficas y

funcionalidades, es una actividad que tiene un alto costo en tiempo, realizar esta actividad para un número alto de problemas se encuentra fuera del alcance de este proyecto.

De esta forma, se seleccionaron 6 problemas para los cuales se realizó diseño de interfaz gráfica y definición de funcionalidad. La selección de estos problemas, se hizo nuevamente mediante la evaluación de la pertinencia que cada problema tiene dentro del dominio del conocimiento y evaluando la complejidad que cada problema presenta dentro de la temática especifica de control. Se dio prioridad a los problemas cuya complejidad resulta menor, puesto que el diseño y la implementación de tutores para temáticas complejas requiere del desarrollo especifico de componentes de interfaz gráfica; requieren de un mayor tiempo de diseño en la definición de la funcionalidad y por supuesto mayor tiempo en la implementación final. Dando prioridad a los problemas de complejidad menor, se pueden recorrer más fácilmente un mayor número de problemas en cada tipo de sistema y se puede realizar una comparación con mayor exactitud manteniendo los tiempos de desarrollo dentro de los limites deseados.

Los problemas sujetos a diseño de interfaz gráfica fueron los siguientes:

- Problema P4.
- Problema P7.
- Problema P8
- Problema P26.
- Problema P36
- Problema P60

Para diseñar el tutor y la interfaz gráfica se requiere consolidar y aplicar, en un mismo ejercicio de diseño, un conjunto de conocimientos provenientes de diferentes actividades:

1) Conocer el problema particular a resolver. Este conocimiento se obtiene mediante la metodología propuesta en el capítulo 2.
2) Conocer la forma como el problema se resuelve. Este conocimiento se obtiene mediante la experiencia específica en el dominio de conocimiento de interés y mediante la ejecución de análisis cognitivo de tareas.
3) Conocer la forma como los estudiantes resuelven el problema y cuales son los errores que típicamente comenten. Este conocimiento se obtiene a partir de ejercicios como think-aloud y la experiencia de impartir clases y calificar evaluaciones.

4) Conocimiento de las posibilidades técnicas de la herramienta de autor y del entorno de ejecución.

Teniendo en cuenta lo anterior y a partir del hecho que en el capítulo 3 se realizó un análisis profundo de los requerimientos de conocimiento que tanto el tutor como el estudiante enfrentan. Se han definido unas guías de diseño generales que aplican para cada una de las categorías de requerimientos de conocimiento definidas. Estas guías permiten abordar las actividades de diseño no solo de los problemas seleccionados sino también de cualquiera de los problemas que no fueron seleccionados.

El listado de guías se presenta a continuación por categoría de requerimiento:

Diagramas de bloque: Para esta categoría se requieren realizar operación de asignación de nombre de señales a bloques y conectores en un diagrama. También se requiere poder editar los diagramas de bloques de forma interactiva. Para realizar asignación de nombres a las señales y bloques de un diagrama se define que en general, se debe utilizar una imagen con selectores tipo combo que permitan escoger los nombres de las señales de entrada, salida y del bloque según se requiera en el ejercicio.

Figura 5. Guía de diseño para representar ejercicios relacionados con el nombramientos de señales y el cálculo de dependencias simples, de las señales en un diagrama de bloque.

Para realizar actividades más complejas como la transformación de diagramas de bloques por medio de expresiones auxiliares, o la aplicación de algebra de bloques se requiere que la interfaz provee funcionalidades similares a la de un editor profesional de diagramas de bloque como los que se pueden crear con la herramienta de software libre JGraphX[3]. Para poder implementar este tipo de herramienta en los entornos de resolución de problema típicos de los sistemas tutores inteligentes se requiere que las herramientas de autor tengan mecanismos de extensibilidad suficientemente profesionales para soportar esta característica.

[3]https://github.com/jgraph/jgraphx

Manipulación de expresiones: Para la categoría de manipulación de expresiones se requieren normalmente realizar simplificaciones, factorizaciones y en general manipulaciones algebraicas. Cada uno de los pasos intermedios de estas manipulaciones pueden representarse gráficamente con un formato muy similar al formato con el que las expresiones algebraicas se presentan en libros y artículos científicos. La siguiente gráfica muestra como los campos de selección tipo combo y los campos de texto pueden usarse para que un estudiante provea cualquiera de los elementos involucrados en una expresión: coeficientes, variables, exponentes, signos.

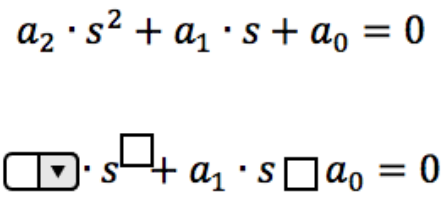


Figura 6. Ejemplo de cómo ubicar componentes de interfaz gráfica de manera que la representación sea similar a la que se encuentra en libros y artículos. Las variables, símbolos, signos, coeficientes y exponentes se pueden convertir en campos de entrada. Sin embargo se sugiere que no todos los elementos de la expresión se conviertan en campos y que solo aquellos que tienen que ver más con el concepto que se debe enseñar lo permitan.

Manipulación de gráficos: Para la manipulación de gráficas, el estudiante requiere realizar la generación gráficas a partir de las expresiones que las definen.Esto se puede lograr con una librería que permita realizar estas operaciones según el entorno de ejecución en el cual se este ejecutando o se vaya a ejecutar el sistema tutor inteligente. De igual manera que para el caso de los diagramas de bloque se requiere que las herramientas de autor permitan extender componentes existentes y/o crear nuevos componentes. Para que el estudiante pueda reconocer expresiones matemáticas a partir de gráficas presentadas se puede reutilizar el criterio definido para la categoría anterior. Para la identificación de parámetros, medidas y otras características a partir de las gráficas, los componentes normales como cajas de texto o listas de selección son suficientes para permitir que el estudiante proporcione estos datos al tutor. Los patrones que se definen para realizar la selección de imágenes que cumplen con una característica definida es el siguiente:

Figura 7. Diferentes forma de ubicar imágenes y asociar dichas imágenes con un sistema de selección único o múltiple que puede ser usado para el reconocimiento de imágenes a partir de conceptos.

En la parte izquierda se muestra una lista de selección de imágenes, donde solo una imagen puede ser seleccionada, y la imagen seleccionada ocupa el espacio total del componente tras ser seleccionada. En la derecha se muestran imágenes asociadas a radio botones o a cajas de chequeo que permiten seleccionar una o varias imágenes respectivamente.

Conceptualización y abstracción de contenidos: Para la conceptualización de conceptos, normalmente se requiere responder preguntas cuyas respuestas no puede ser calculadas, sino que deben ser almacenadas. Para este tipo de interacciones los componentes típicos como cajas de texto, listas de selección y radio botones son suficientes.

Los diseños realizados aplican para cada problema son los mismos para ambos tipos de tutores con el ánimo de comparar que tan efectivas son las herramientas de autor para cumplir con los requisitos de diseño.

En el apéndice F se presentan los diseños a cada uno de los problemas sujetos a diseño. El proceso de diseño se puede resumir mediante las siguientes actividades:

a. Definición de un enunciado de problema y ubicación del enunciado del problema en la parte superior de la interfaz.
b. Selección de pasos observables a partir del desarrollo matemático, numérico o gráfico que se debe realizar para resolver los problemas.
c. Utilización de la guía de diseño adecuada dependiendo si el paso observable corresponde a una expresión, selección de una imagen o definición de un número.

En las imágenes presentadas en el apéndice F se puede notar que la aplicación de estas guías de diseño se ha realizado de forma amplia, a lo largo de todas las interfaces. Utilizando líneas horizontales y paréntesis de gran tamaño se logró obtener una diagramación para los componentes de interfaz gráfica de usuario, que luce bastante similar al formato típico al que los estudiantes están acostumbrados.

El proceso de diseño de estas interfaces fue iterativo e incremental. El proceso de lluvia de ideas para generar una interfaz con el detalle en el que se presentan en el apéndice F, puede tomar entre 2 y 4 horas. A medida que van ocurriendo las interacciones, se van ajustando los textos y elementos que se presentan en el enunciado, se modifican los pasos observables de acuerdo a: las necesidades de aprendizaje de los estudiantes, la experiencia que se tiene de los errores típicos que comenten, la experiencia que se tiene acerca de los pasos que requieren mayor conocimiento, la experiencia que se tiene de las estrategias típicas para el cálculo. Toda esta experiencia fue aportada por el director de este trabajo, gracias a su amplia experiencia en el campo de la educación en sistemas de control.

Para todas las interfaces presentadas en el apéndice F, se realizó este proceso iterativo que tiene en cuenta diferentes fuentes de conocimiento: Experiencia en la educación en control, buenas prácticas para la creación de sistemas tutores inteligentes, conocimiento de las posibilidades de las herramientas de autor y conocimientos específicos en desarrollo de software para atender requerimientos no soportados por las herramientas de autor.

En la siguiente sección se presentan los componentes de software que tuvieron que ser desarrollados para atender los requerimientos específicos impuestos en la etapa de diseño.

4.3 Creación de componentes personalizados

Durante el desarrollo de los diferentes tipos de sistemas tutores inteligentes, se llegó a la necesidad de realizar desarrollo de software personalizado de componentes, para poder garantizar el cumplimiento de ciertos requerimientos de diseño, establecidos durante la etapa de concepción.

Otros artefactos de software tuvieron que ser creados para atender requerimientos de funcionalidad definida sobre los sistemas tutores inteligentes.

También se crearon artefactos de software que permiten la fácil reutilización de componentes. Permitir la fácil reutilización de componentes, ayudó en el objetivo de reducir los costos en tiempo de desarrollo de algunas interfaces gráficas y también en mantener baja la complejidad de la solución para que futuros mantenimientos pudieran hacerse con mayor facilidad.

Los artefactos de software desarrollados fueron:

1) Selector de imágenes y expresiones versión JAVA y HTML.

   El selector de imágenes que se definió en la sección 4.2 para selección de gráficas fue implementado tanto en JAVA como en HTML. En la siguiente figura se muestra el selector en funcionamiento.

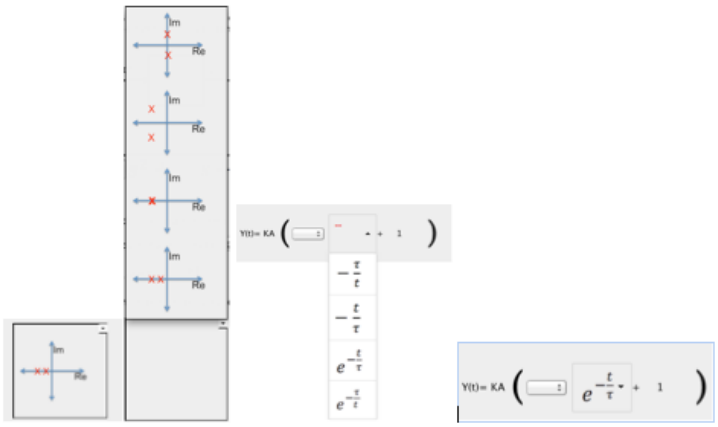


Figura 8. Representación gráfica de los selectores de imágenes y expresiones en pantalla. A la izquierda s e muestra la versión implementada en lenguaje Java , a la derecha la versión HTML. Se muestran dos estados del selector, cuando se está seleccionando la opción y cuando se ha terminado con la selección

   El esfuerzo realizado especialmente para construir la versión en lenguaje Java de este componente fue importante. Los componentes de selección en Java soportan imágenes sin embargo su presentación suele tomar más espacio del requerido por este proyecto. En la siguiente figura se muestra un ejemplo de un selector de imágenes creado con las técnicas usuales en lenguaje Java y la diferencia como luce respecto al selector desarrollado.

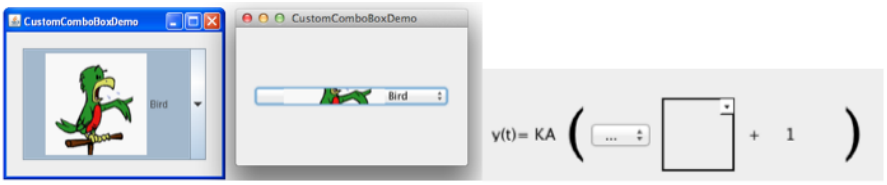


Figura 9. Se muestran las diferencias de la distribución y forma como luce y se siente (look and feel) de un componente selector de imágenes típico del lenguaje Java, y uno que ha sido creado específicamente para que se integre a los requerimientos de diseño impuestos.

Se puede observar que el selector de imágenes usual de Java, no solo luce diferente en sus implementaciones para sistemas operativos Mac OS y Windows, sino que su diseño no se integra de igual manera que el desarrollado para este proyecto, mostrado al lado derecho de la figura. Adicionalmente, de cambiar la forma en que luce el componente se realizó una personalización para otorgarle al componente todas las características necesarias de forma que pueda ser integrado con el grabador de comportamiento de CTAT. Las descripción detallada de cómo se otorgaron dichas capacidades, esta fuera del alcance de este documento, sin embargo una buena guía para comprender este proceso se puede encontrar en el documento Tool-Tutor Application Interface que hace parte del manual de usuario de la Herramienta CTAT.

La versión HTML del componente se creó utilizando un plugin de la famosa librería JQuery. Es de notar que el esfuerzo realizado para obtener el componente en su versión HTML fue unas 5 veces menor que el esfuerzo realizado para obtener la versión en lenguaje Java. La versión HTML se utilizó en los sistemas tutores inteligentes creados con la herramienta ASPIRE. Aunque dicha herramienta no soporta este tipo de componentes por defecto en la sección 4.5 se presenta la forma como varios componentes Javascript pudieron ser integrados a la plataforma.

2) Capacidad para ejecutar TPA sobre grupos de componentes en JAVA usando la clase PanelTPAProxy.

Una de las funcionalidades de los sistemas tutores example-tracing, es la posibilidad de ejecutar acciones realizadas por el tutor de forma automática, una vez se alcanza un determinado estado del problema. Las acciones normalmente son: definición de variables y modificaciones de los componentes de la interfaz gráfica (por ejemplo ocultar un componente que ya no es necesario ó mostrar un nuevo componente que representa un paso intermedio). Sin embargo existe una limitación importante para los tutores creados

utilizando interfaces en lenguaje Java: solo se pueden modificar componentes discretos de la interfaz gráfica, es decir no se pueden ocultar, mostrar o desplazar grupos de componentes. Esta limitación es bastante grave debido a que normalmente los pasos intermedios requieren mas de un componente. Para resolver este problema, se ha extendido un componente del tipo JCommTextField. Este es un componente que hace parte del grupo de componentes con los que CTAT cuenta por defecto. Este componente es una caja de texto que se comunica adecuadamente con el grabador de comportamientos de CTAT.

La extensión del componente tipo JCommTextField, se realizó de manera tal que el nuevo componente, llamado PanelTPAProxy, se mantiene siempre oculto. Cada vez que el componente PanelTPAProxy recibe una señal proveniente desde el grabador de comportamientos, se reenvía la señal al objeto que representa el contenedor de más alto nivel en la interfaz gráfica. A partir de allí, el contenedor de más alto nivel puede mostrar u ocultar cualquier panel que desee. Debido a que las señales enviadas por el grabador de comportamientos a la clase PanelTPAProxy llevan parámetros, los parámetros son reenviados al contenedor de más alto nivel y por lo tanto este contenedor puede distinguir claramente cual grupo de componentes modificar y que tipo de acción realizar.

3) El asistente en al creación de interfaces para ASPIRE

En la sección 4.5.1 se describe en detalle una capacidad de ASPIRE que permite la inyección de código HTML, CSS y Javascript para mejorar la apariencia de las interfaces generadas automáticamente por ASPIRE.

Para facilitar el posicionamiento de los componentes de forma que se pudiera cumplir con las guías de diseño establecidas, en la sección anterior, se desarrollo un pequeño asistente para construir las interfaces en ASPIRE.

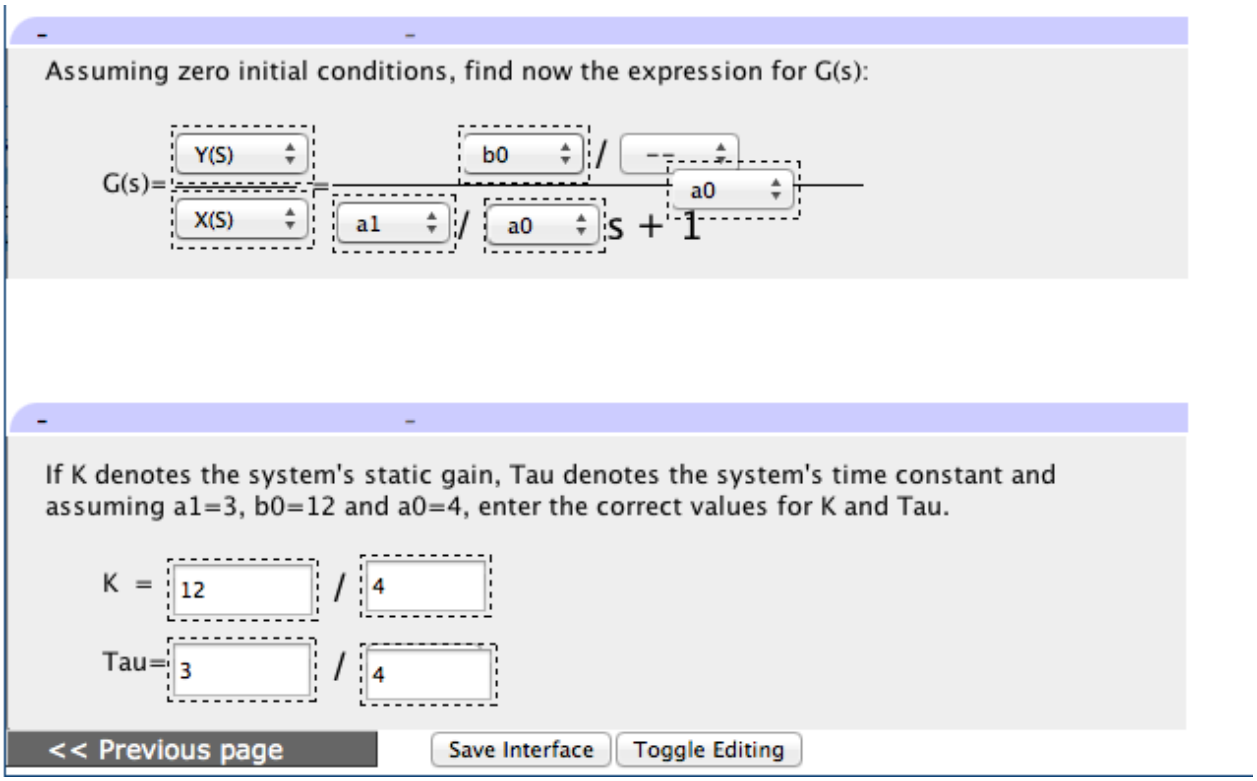


Figura 10. Se muestran los botones personalizados que ejecutan las funciones del asistente de construcción de interfaces en ASPIRE. Los bordes punteados alrededor de los componentes sugieren que está activo el modo de edición y que por lo tanto los componentes pueden ser arrastrados para cambiar su posición y dimensiones.

El asistente ayuda a incluir imágenes de fondo sobre los paneles que definen los pasos procedimentales de la solución. Luego con la ayuda de la funcionalidad “Toggle Editing” que hace parte del asistente desarrollado, se pueden transformar todos los campos de texto y listas de selección en elementos que se pueden arrastrar y soltar y cuyas dimensiones alto y ancho, se pueden modificar con el mouse. Finalmente la funcionalidad “Save Interface”, utiliza código Javascript que extrae los valores de los parámetros CSS de cada uno de los campos que fueron posicionados y transforma esos valores CSS en instrucciones Javascript que son almacenadas en una de las componentes del enunciado del problema.

Cuando el estudiante ejecuta el problema, las funciones de JavaScript almacenadas son recuperadas y se ejecutan para restablecer las posiciones y dimensiones que fueron definidas en el paso anterior para cada uno de los campos.

Este asistente tuvo un impacto importante en la reducción de los tiempos de desarrollo y fue uno de los primordiales mecanismos que permitieron cumplir con los requerimientos de diseño impuestos para las interfaces gráfica.

Se espera poder compartir este asistente con los desarrolladores de ASPIRE de manera que lo puedan incluir por defecto en el sistema.

4.4 Aspectos de Implementación sobre tutores example-tracing usando CTAT.

Varios aspectos de uso avanzado sobre la herramienta CTAT fueron necesarios para cumplir con los requisitos de diseño y de funcionalidad que se plantearon desde la etapa de concepción.

En el siguiente listado se muestran las características implementadas mediante la herramienta CTAT que son consideradas de uso avanzado y se realiza una descripción de la funcionalidad que cada una provee.

4.4.1 Scafolding mediante acciones sub-óptimas y TPA

Por medio del uso de la clase PanelTPAProxy, que fue descrita en el apartado anterior, fue posible el desarrollo de scafolding avanzado en las interfaces Java. A través de esta técnica fue posible mostrar grupos de componentes, agrupados por un Panel, que le muestran al estudiante un paso intermedio que tiene que realizar cuando ha cometido un error en un paso más avanzado.

4.4.2 Generación de datos aleatorios al inicio del problema mediante TPA

Mediante el uso de acciones ejecutadas automáticamente por el tutor, fue posible la generación de datos aleatorios al inicio de cada problema que fue implementado con la herramienta CTAT. La ejecución de estas acciones automáticas, realizan el llamado de la función interna de CTAT llamada *assign("myvar", "blue").* Utilizando esta función en conjunto con la función Math.random() del lenguaje JAVA, es posible crear variables, cuya asignación se realiza con un numero aleatorio, y que a partir del momento de la asignación almacenan dicho número hasta el final de la ejecución del problema. Gracias a este comportamiento, se pueden generar variables con números aleatorios o incuso con números que cumplan ciertos requisitos, luego se puede utilizar acciones ejecutadas por el tutor para enviar dichos datos a las interfaz gráfica

y finalmente se pueden generalizar los pasos posteriores en el problema por medio de formulas que hagan referencia y uso de las variables creadas al inicio del problema.

Esta técnica de inicialización de los datos del problema tiene dos ventajas importantes:

a. Para todos los estudiantes los datos de trabajo son diferentes, de manera que se reduce la probabilidad de que se engañe al sistema mediante la copia de resultados de otro estudiante que este cercano.
b. Permite que se puedan tener problemas con datos numéricos diferentes sin tener que realizar la producción en masa de grafos isomorfos utilizando plantillas.

4.4.3 Despliegue en la plataforma Tutorshop.

Como se presento en la sección tres, Tutorshop es un LMS creado para administrar los tutores generados con la herramienta CTAT. Mediante sus funcionalidad de despliegue los sistemas tutores inteligentes tipo example-tracing y model-tracing se pueden poner a disposición de los estudiantes. El sistema crea registros avanzados de la interacción entre los estudiantes y los interfaces gráficas. También guarda registros estadísticos que permiten a los autores conocer el progreso que los estudiantes tuvieron a lo largo de un conjunto de problemas, conocer estadísticas especificas sobre el uso de cada interfaz y conocer el progreso de los estudiantes respecto a los Skills definidos en los tutores creados con CTAT.

Durante las etapas de implementación y despliegue se hizo uso de las funcionalidades de Tutorshop, encontrando que es una herramienta bastante poderosa debido al número de reportes que pone a disposición del autor, pero que presenta algunos fallos de funcionalidad. Específicamente los fallos de funcionalidad encontrados, causan que los registros de interacción sobre las interfaces creadas en entorno de ejecución Java no queden almacenados. Adicionalmente las implementaciones de los algoritmos de secuenciamiento de problemas (simple, aleatorio, knoledge-tracing) no funcionan correctamente para el momento en que se escribe y se oficializa este documento.

Se realizaron pruebas exhaustivas sobre la herramienta tutorshop y el reporte de esas pruebas mediante mensajes de correo electrónico. El reporte de las pruebas incluye videos acerca de cómo reproducir los malos funcionamientos de la aplicación, los cuales fueron enviados a los propietarios de la herramienta. En general los desarrolladores se han mostrado dispuestos a resolver los malos funcionamientos, por lo cual varios mensajes de correo fueron

intercambiados durante las etapas de implementación y pruebas. Sin embargo, para la fecha de este reporte, algunos de los problemas no fueron resueltos mientras que a otros problemas se les dio solución conjunta.

4.5 Aspectos de Implementación sobre tutores basados en restricciones usando ASPIRE.

Varios aspectos de uso avanzado sobre la herramienta ASPIRE fueron necesarios para cumplir con los requisitos de diseño y de funcionalidad que se plantearon desde la etapa de concepción.

En el siguiente listado se muestran las características implementadas mediante la herramienta CTAT que son consideradas de uso avanzado y se realiza una descripción de la funcionalidad que cada una provee.

4.5.1 Mejoramiento de la interfaz gráfica mediante CSS HTML y Javascript.

En la sección 4.1.2 se describió la metodología de implementación que se utiliza para la construcción de sistemas tutores inteligentes basados en restricciones usando la herramienta ASPIRE. En especial se mostró que uno de los pasos de la metodología incluye la definición de componentes de problema y de componentes de solución.

A través de las pruebas exhaustivas que se hicieron del sistema, se pudo evidenciar que el sistema no tiene protección alguna ante ataques de tipo XSS (Cross Site Scripting). Aunque este hecho, pareciera ser una gran desventaja del sistema, resultó ser el mayor mecanismo de extensibilidad posible en esta herramienta.

Debido a que la herramienta ASPIRE esta implementada en su parte cliente, utilizando tecnologías basadas en la web (CSS, HTML y Javascript) , el código fuente de la aplicación cliente es completamente accesible y se puede estudiar para entender en detalle la forma como la aplicación funciona. A partir de este estudio se pudo determinar que si durante la definición de las componentes de problema, se definen componentes de tipo texto, dichas componentes pueden aceptar como valores validos posibles código fuente HTML, CSS y Javascript que

resulta ser almacenado tras la definición y recuperado e incluido dentro del entorno de ejecución de la página web cuando el estudiante inicia la ejecución del problema, una vez el problema se ha desplegado.

Esta capacidad del sistema, que en términos de seguridad informática resulta ser una gran vulnerabilidad, se ha usado para extender la herramienta y permitir que nuevo código Javascript sea inyectado para definir nuevas funciones, código CSS sea incluido para mejorar los aspectos gráficos y código HTML sea adjunto a la estructura de HTML inicial para completar lo elementos de interfaz que sean necesarios.

El principal uso que se dio a esta capacidad del sistema, fue la del reordenamiento de los componentes gráficos, mediante la asignación de valores a las propiedades CSS utilizando funciones definidas en el entorno de ejecución de Javascript. Usando esta técnica se puede sobre-escribir completamente los estilos CSS iniciales definidos por los desarrolladores de ASPIRE y se puede dar un nuevo diseño gráfico a la interfaz gráfica de usuario. A continuación se muestran varias modificaciones que se realizaron sobre las interfaces gráficas implementadas en ASPIRE.

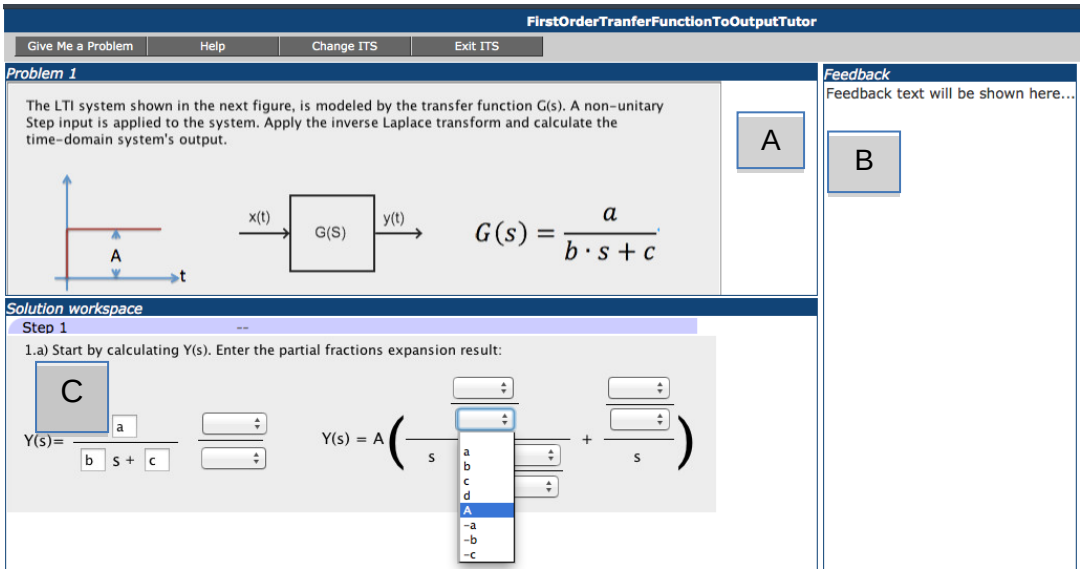


Figura 11. Se muestran los cambios logrados sobre la interfaz gráfica de aspire

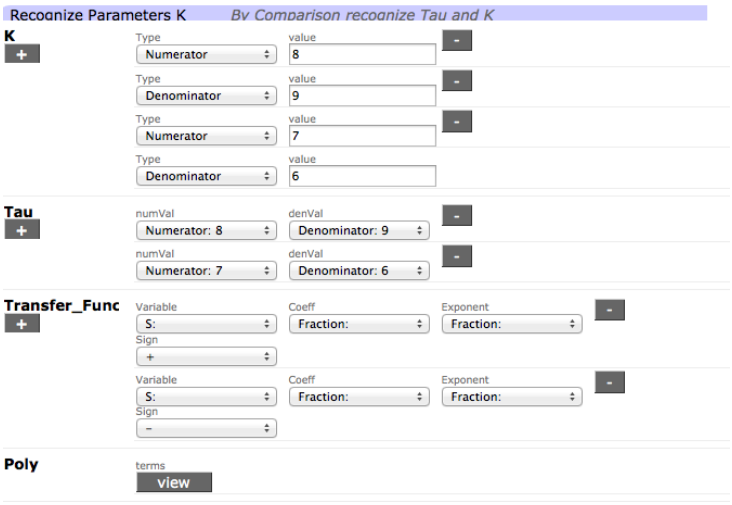


Figura 12. Se muestra la apariencia que tiene una interfaz gráfica típica de aspire cuando la apariencia no se puede cambiar utilizando Javascript, CSS y HTML. Las diferencias son basntate notorias y la ganancia en apariencia con la posibilidad de inyección de código es muy grande.

En A se muestra que se puede manipular la sección donde se presenta el enunciado del problema para que tenga una mejor apariencia. En B se muestra que la sección donde se presenta la ayuda se movió hacia izquierda y se estableció que tuviera posicionamiento estático, de manera que al bajar la barra de scroll esta sección siempre está visible. En C se muestra que todos los componentes se pudieron disponer de forma tal que la diagramación coincide con los requerimientos de diseño.

4.5.2 Reutilización de las interfaces creadas en CTATen el sistema ASPIRE.

Adicional a los componentes personalizados presentados en al sección 4.3. Un desarrollo experimental adicional fue realizado para permitir que las interfaces gráfica creadas en con la herramienta CTAT se pudiera utilizar en la herramienta ASPIRE. La técnica básica que se utilizó fue la creación de Mock Objects cuyos nombres de clase, paquetes, nombres de métodos y firmas de métodos, corresponden completamente con los objetos reales que conforman las dependencias de software de las interfaces gráficas creadas con CTAT. La implementación de todos los métodos y constructores en los objetos Mock creados fue completamente vacía. De esta forma, el código fuente de las interfaces gráficas se puede mantener sin modificaciones alguna y solo se requiere cambiar la librería que se incluye en el CLASSPATH del compilador de Java para permitir que la interfaz gráfica utiliza todas las funcionalidades de CTAT ó para permitir que dichas dependencias sean simuladas, con un

comportamiento vacío por las clases Mock. El tamaño de los Applets finales que se deben desplegar a la herramienta ASPIRE se reduce a 150KB comparado con los 20MB de librerías que se requieren para ejecutar CTAT.

Gracias al desarrollo de la componente de software MockCTAT se hizo posible la reutilización de las interfaces gráficas creadas para la herramienta de autor CTAT dentro de la herramienta ASPIRE. Una de las ventajas que tiene esta técnica es que se reduce el tiempo de desarrollo de las interfaces gráficas para los tutores basados en restricciones.

Sin embargo, el uso de Applets como mecanismo de generación de la interfaz gráfica para este tipo de tutores tiene un costo en desarrollo adicional. Al utilizar programación en Java se debe generar a partir del estado de la interfaz gráfica una representación en lenguaje XML de la solución del estudiante. Esta representación es enviada al servidor y es analizada por el conjunto de restricciones. Se debe entonces entender la estructura de representación de las soluciones en lenguaje XML y se requiere escribir el código Java para generar dicha representación puesto que no existen librerías para la generación especifica de la estructura del documento XML que fue definida por los desarrolladores de ASPIRE. Dado que la estructura de los documentos XML que se deben enviar al servidor no es conocida y no esta públicamente disponible, no es posible utilizar un Framework de desarrollo rápido como JAXB.

Se concluye entonces que aun cuando se hace un esfuerzo mínimo creando la interfaz de usuario gracias al artefacto MockCTAT, se debe en todo caso, realizar un esfuerzo grande creando la representación en lenguaje XML de la solución, esto hace que la elección de Java como mecanismo para generar la interfaz no sea tan atractivo y que se prefiera el mecanismo de inyección de CSS, HTML y Javascript, puesto que al utilizar esta técnica, el entorno de ejecución Javascript programado por los desarrolladores de ASPIRE, ya incluye las funcionalidades para crear la representación de la solución en lenguaje XML.

Adicionalmente, por medio de las pruebas exhaustivas a la herramienta fue posible determinar que las interfaces creadas en lenguaje Java, no podrían tener soporte para el resaltado de campos que contienen datos incorrectos, si no se hace una extensión, utilizando Javascript al entorno de ejecución de la página.

4.5.3 Inclusión de Ayuda mejorada mediante el uso de CSS, HTML y Javascrtipt.

Gracias a la posibilidad de inyección de CSS, HTML y Javascript dentro de las componentes de problemas se hace posible, la inclusión de librerías de componentes típicos en el desarrollo web dentro del entorno de ejecución del sistema tutor inteligente. Gracias a que ninguna parte del sistema tiene protección contra el ataque XSS, los mensajes de ayuda que se definen para las restricciones también puede incluir código HTML. Esto hace que el sistema de ayuda se convierta en un sistema muy versátil y poderoso. Teniendo la posibilidad de incrustar HTML en los mensajes de ayuda, se puede poner a disposición de los estudiantes herramientas didácticas como videos, imágenes, sonidos, hipervínculos, y en general, se podría incluir cualquier tipo de interacción avanzada que se pueda construir con HTML, CSS y Javascript.

En la siguiente figura se muestra la forma como uno de los tutores implementados presenta a los estudiantes una imagen que puede ser ampliada para que abarque la totalidad de la pantalla. En el caso especifico, se presenta una imagen que muestra el procedimiento de cálculo de fracciones parciales hasta el antepenúltimo paso. El último paso debe ser ejecutado por los estudiantes.

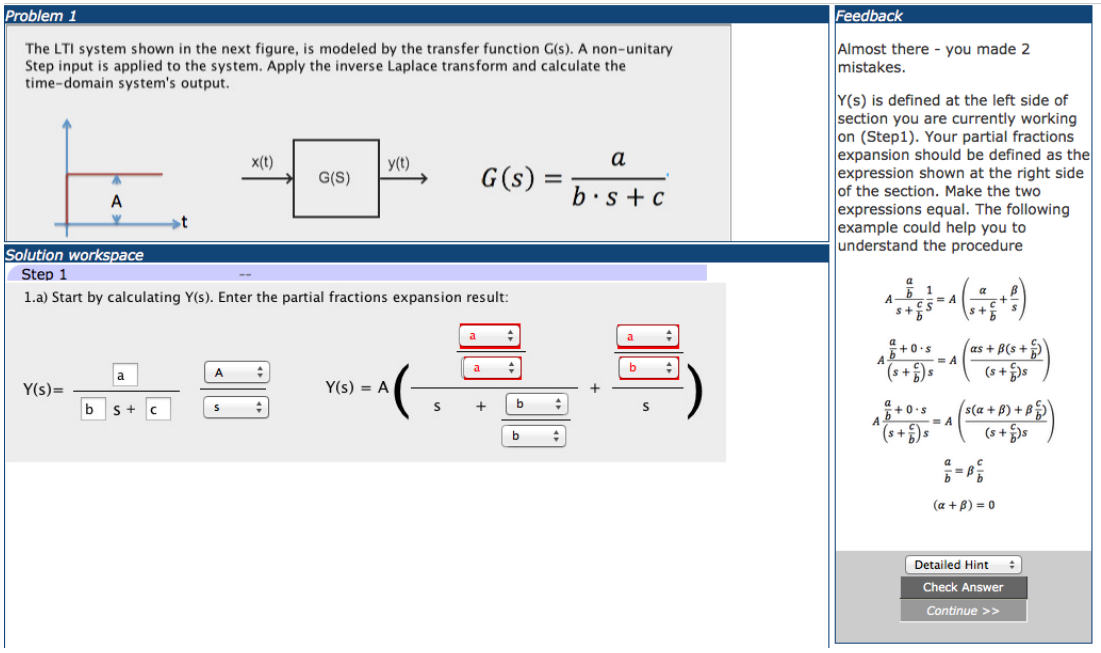


Figura 13. Cuando se detectan errores, la ayuda presentada puede incluir HTML. En la figura se muestra como una imagen que detalla un procedimiento matemático es incrustada entre el mensaje de ayuda.

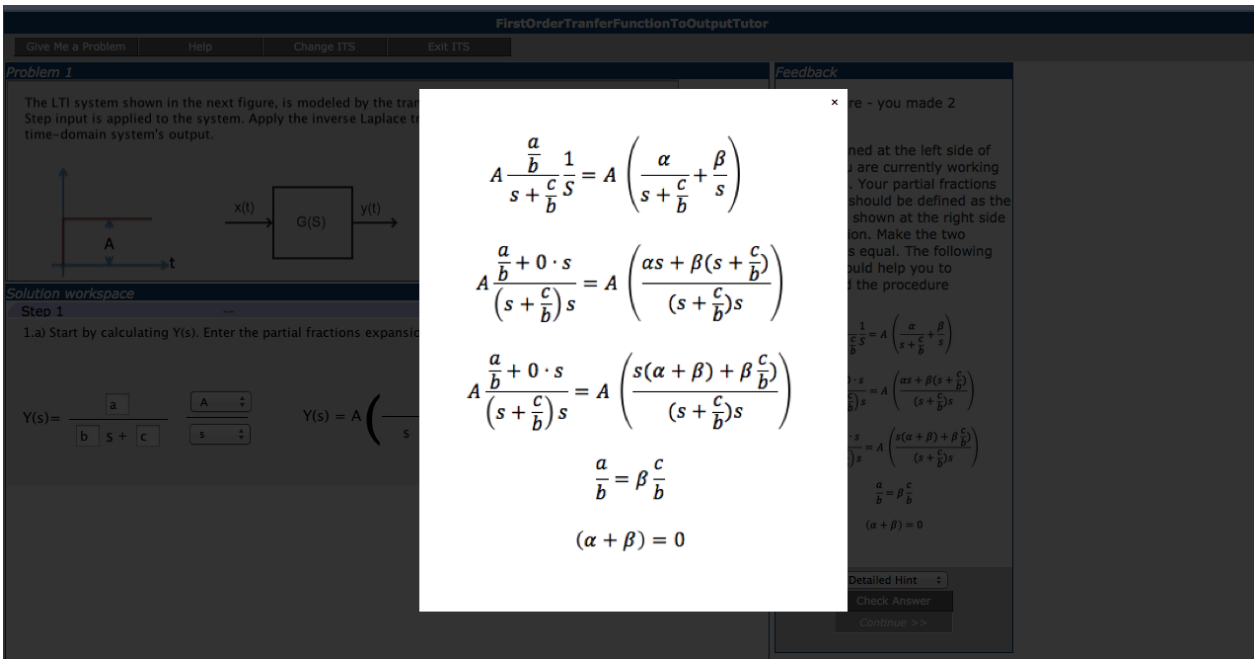


Figura 14. Una interacción avanzada permite al estudiante hacer clic sobre la imagen y ampliarla utilizando un patrón conocido como Lightbox.

4.6 Tiempos de desarrollo.

4.6.1 Sistemas tutores tipo example-tracing.

Según (Aleven, McLaren, Sewall, Koedinger, 2006) los resultados experimentales sugieren que la creación de Tutores Cognitivos usando CTAT es mucho más rápida que con otras herramientas de autor. Las mejoras se estiman en 1.4 a 2 veces más rapidez comparando con las herramientas estándar para aplicar la técnica "*Model Tracing*". Entre los mayores beneficios que se obtienen usando la herramienta de autor CTAT, se encuentra la disminución del número de ciclos de Edición-Prueba-Depuración necesitados para la creación de un modelo Cognitivo.

La relación del tiempo usado para el desarrollo al tiempo instruccional de los tutores tipo example-tracing creados con CTAT es alrededor de 23 a 1. Comparado con la misma relación para los tutores Cognitivos que es de 200:1, los Tutores Tipo example-tracing proveen un ahorro significativo en el tiempo de desarrollo lo cual permite que el proceso de desarrollo del material instruccional sea más eficiente.

Específicamente para el trabajó de implementación que se llevo a cabo usando le herramienta CTAT, se observaron que en promedio las actividades de ejecución de todo el ciclo presentado en la sección 4.1 toma alrededor de unas 9 horas discriminadas de la siguiente forma:

Tabla 22. Calculo del tiempo total de desarrollo para un problema asistido por sistemas tutores inteligentes tipo example-tracing

Actividad	**Tiempo Promedio**
Cálculo de la solución correcta en papel	30 Minutos
Selección de pasos observables	1 Hora
Diseño de la interfaz de usuario	1 Hora
Construcción de la interfaz de usuario	1 hora y 30 minutos
Tiempo gastado en la demostración	15 Minutos
Anotación del grafo con mensajes de error	2 Horas
Demostración de Acciones incorrectas, sub-óptimas y anotación con mensajes de ayuda	40 Minutos
Desarrollo de una acción de scafolding.	1 Horas
Agrupamiento de acciones y definición de restricciones de ordenamiento	15 Minutos
Anotación del grafo con Skills	15 a 30 Minutos.
Total	9 Horas

El tiempo promedio de interacción con la interfaz gráfica es de 15 minutos, de manera que para poder lograr una hora de instrucción se requieren en promedio 4 interfaces gráficas diferentes. Esto nos lleva a definir la relación del tiempo usado al tiempo instruccional como **36:1.**

La relación calculada tiene en cuenta todo el ciclo de desarrollo que inicia desde la solución del ejercicio en papel. Se considera que desde esta etapa debe medirse el tiempo gastado en las actividades puesto que una vez un problema se selecciona y se realiza el análisis completo

para poder determinar los pasos observables y las diferentes estrategias de solución, se inicia con la inversión de tiempo que termina cuando la interfaz y el grafo de comportamientos quedan completamente construidos. El mayor valor en la tasa mostrada puede estar relacionado con la relativa poca experiencia que se tiene con la herramienta comparada con la experiencia con que cuentan los investigadores principales. También debido a que en los trabajos presentados no se realiza una presentación detallada de los tiempo por cada una de las actividades que conforman la metodología, no es posible determinar si en los reportes presentados se incluyen más o menos actividades que las mostradas en la tabla anterior.

4.6.2 Sistemas tutores basados en restricciones

Respecto a la experiencia con la herramienta de autor ASPIRE, la comparación del tiempo utilizada y su eficiencia se puede realizar de dos formas. Una basada en el tiempo de trabajo por restricción y otra , de manera similar al apartado anterior, basada en el tiempo de trabajo por hora de instrucción.

En (Antonija Mitrovic, 1998) se reporta que el tiempo promedio de trabajo requerido para desarrollar 1 restricción en el sistema SQL-Tutor es de 1.3 horas. Este tiempo es comparado en dicha publicación con las 10 horas que deben gastarse para el desarrollo de una regla de producción en un sistema model-tracing, resaltando qué, a parte de otras razones, la diferencia en el tiempo gastado entre una restricción y una regla puede deberse a que el formalismo basado en restricciones es más apropiado.

Debido a que el desarrollo de las restricciones que se utilizaron para crear los sistemas tutores inteligentes implementados en este trabajo no se realizó de forma manual sino que se utilizó el generador de restricciones de ASPIRE, resulta bastante difícil calcular el tiempo empleado por restricción. Calcularlo no sería significativo porque en promedio se generan unas 30 restricciones por problema y el tiempo que se requiere para corregir las restricciones cuando estas no resultan suficientemente generales no supera 1 hora para todas las 30 restricciones.

Varias restricciones fueron creadas de forma diferente a las que el generador automático es capaz de calcular. Esta actividad se llevó a cabo porque para ciertas secciones de la interfaz gráfica de usuario y en especifico para el dominio de sistemas de control, los mensajes de ayuda no se pueden proveer para un solo campo sino para un grupo de campos. Aún así el

tiempo de desarrollo de estas restricciones no tomó una hora debido a que se utilizaba como base las restricciones ya generadas.

Resulta más conveniente realizar el cálculo del tiempo completo invertido en todo el ciclo de desarrollo del sistema tutor basado en restricciones y luego medir el tiempo de instrucción generado. De esta forma, la tabla con el listado de actividades y tiempos registrados es la que se muestra a continuación:

Tabla 23. Calculo del tiempo total de desarrollo para un problema asistido por sistemas tutores inteligentes tipo example-tracing

Actividad	**Tiempo Promedio**
Cálculo de la solución correcta en papel	30 Minutos
Selección de pasos observables	1 Hora
Diseño de la interfaz de usuario	1 Hora
Definición de la estructura del Problema (Pasos del problema: Procedimental o No Procedimental)	5 Minutos
Definición de los conceptos de ontología	20 Minutos
Definición de las componentes de Problema y Solución	40 Minutos
Construcción de la interfaz de usuario	1 hora y 30 minutos
Definición de los problemas específicos y las soluciones ideales.	15 Minutos
Edición y corrección de restricciones generadas.	1 Horas
Anotación de las restricciones usando mensajes de ayuda.	2Horas
Total	8 Horas

El tiempo promedio de interacción con la interfaz gráfica es de 15 minutos, de manera que para poder lograr una hora de instrucción se requieren en promedio 4 interfaces gráficas diferentes. Esto nos lleva a definir la relación del tiempo usado al tiempo instruccional como **32:1.** Esta

relación de tiempos no se encuentra muy alejada de la que se encontró para sistemas tutores tipo exampe-tracing y se considera bastante acertada debido a que la mayor cantidad de funcionalidades que están disponibles en CTAT, causa que el tiempo de autoría aumente.

4.7 Discusión.

Los sistemas tutores basados en restricciones permiten enfocarse mas en las reglas del modelo del dominio que en la interacción. La interacción en los tutores tipo example-tracing es bastante poderosa, sin embargo el paradigma se centra más en la construcción de la interfaz gráfica lo cual es un proceso costoso. Mayor interacción se puede tener para los tutores tipo example-tracing, por ejemplo se pueden tener problemas cuya interacción presenten al estudiante pasos intermedios por cada uno de los pasos intermedios en el proceso real en papel. Para estos casos la instrucción es mas elevada, sin embargo el costo de desarrollo es bastante alto pues requiere un gran conocimiento de la tecnología sobre la que funcionan las herramientas de autor, así como una definición completa y muy detallada de la interacción deseada. Esto causa que el desarrollador pase mucho tiempo ajustando aspectos técnicos relacionados con tecnología y con la forma como se implementa la interfaz grafica y pierda la posibilidad de hacer un mayor análisis del objetivo educativo detrás de la creación de cada tutor.

Cada problema se resuelve por medio de la ejecucion de un conjunto de pasos. La informacion que resulta de la ejecucion de cada paso debe ser provista/registrada ante la interfaz gráfica para notificar al tutor del resultado de la ejecucion del paso. Aun cuando los pasos no estan limitados por la interfaz grafica, la forma como el estudiante provee la informacion a la interfaz gráfica si lo esta. De acuerdo a las posibilidades de interaccion que otorgue la interfaz gráfica, se hace factible o no, proveer la información resultante de un paso. Si las limitaciones de la interfaz gráficas son bastantes, asi mismo se limitará la posibilidad de proveer informacion al tutor y de la misma forma se limitará la posibilidad de desarrollar ciertos pasos y como consecuencia ciertos problemas y objetivos de aprendizaje.

Es importante poder escribir mensajes de ayuda en lenguaje natural. Esto permite utilizar los términos propios de cada dominio y hacer que el discurso de dichos mensajes tengan una gran similitud con los que se encuentran en libros y con los que se imparten en la clase personal. Específicamente para control es importante poder incluir notación matemática en los mensajes de ayuda.

5. Evaluación de funcionamiento

En este capítulo se realiza una descripción detallada de la evaluación de funcionamiento que se realizó para un conjunto de interfaces seleccionadas. La evaluación de funcionamiento incluyó el despliegue de las interfaces seleccionadas en el sistema Tutorshop y en el sistema ASPIRE-Tutor y la realización de una prueba piloto con un conjunto de usuarios.

El objetivo de la prueba piloto, fue permitirle a un conjunto de usuarios que accedieran a las interfaces gráficas diseñadas, probaran las funcionalidades típicas de un sistema tutor inteligente y realizarán una evaluación de la experiencia.

En este capítulo se presentan las características que se definieron para realizar la evaluación y describe el instrumento que se utilizó para realizar dicha evaluación. Las características definidas fueron usabilidad, calidad del contenido presentado, utilidad del sistema de ayuda del sistema tutor inteligente, algunos aspectos de la motivación que el sistema genera en los usuarios y el grado de aceptación de este tipo de sistemas como método de estudio.

También se presenta en este capítulo una descripción de la pruebas pilotorealizada con ayuda de los usuarios. La convocatoria para participación en las pruebas de evaluación se realizó vía correo electrónico a un conjunto de estudiantes de control automático, que se consideran el tipo de usuarios ideales para este tipo de sistemas.

Al final del capítulo se presentan un reporte de los datos que fueron recolectados durante la actividad de pruebas, se presentan conclusiones acerca de los datos obtenidos, se hace referencia a algunos comentarios importantes realizados por los usuarios que participaron en las pruebas.

5.1 Características de funcionamiento y métrica de comparación

En esta sección se presentan algunas consideraciones que fueron tenidas en cuenta en el momento de definir las características de evaluación de los sistemas tutores inteligentes implementados. También se exponen las razones por las cuales no se realiza un estudio del impacto en el aprendizaje de los sistemas tutores implementados. Tras exponer las consideraciones necesarias se realiza la definición de las características de funcionamiento que serán evaluadas, se selecciona la metodología de evaluación y se presenta la encuesta que se utilizó como instrumento de captura de datos.

5.1.1 Consideraciones acerca de las características para evaluación funcionamiento.

En (Mark & Greer, 1993) se hace un estudio de las metodologías de evaluación para sistemas tutores inteligentes. Entre los aspectos principales que revisa dicho trabajo, se encuentran la clasificación de los tipos de evaluación como formativa y acumulativa; y la caracterización de varias técnicas de evaluación que posteriormente son relacionadas con los componentes de un sistema tutor inteligente de acuerdo a la pertinencia de una técnica para ser aplicada en la evaluación formativa o evaluativa de un componente. La evaluación formativa es considerada como la evaluación que se realiza en etapas de desarrollo del proyecto para identificar problemas y guiar las actividades de modificación. La evaluación acumulativa es considerada como la evaluación que se lleva a cabo para soportar afirmaciones formales acerca de la construcción, comportamiento o resultados obtenidos con un sistema ya terminado.

Para los componentes de modelo del dominio e interfaz gráfica, (Mark & Greer, 1993) definen ciertos métodos de evaluación que se adaptan bien para evaluar estos componentes de acuerdo a los objetivos y naturaleza de los mismos.

Para el modelo de dominio las posibles técnicas para la evaluación formativa son la inspección de expertos y la prueba de Turing. Se hace énfasis en que para este componente, la evaluación formativa es de mayor relevancia de la evaluación acumulativa. Esta evaluación estaría orientada a la garantizar la precisión de la información que de un tema o dominio se puede presentar al estudiante, por ejemplo: descripciones textuales, ejemplos, problemas a resolver,

preguntas a ser respondidas, pantallas gráficas activas o interactivas y la retroalimentación que responden a las acciones del estudiante.

Para el componente de interfaz gráfica de un sistema tutor inteligente, los modelos de evaluación sugeridos son revisión de los diseños utilizando guías de buenas prácticas y pruebas piloto. Sin embargo se menciona que las pruebas piloto, realizadas con miembros de la población para la cual la aplicación fue desarrollada, son probablemente la mejor manera de detectar problemas en la interfaz gráfica y realizar la evaluación de este componente.

De tres tipos de prueba piloto, existe una de ellas en la cual un sistema que se encuentra en etapas finales de desarrollo (donde los contenidos están casi estables), se prueba utilizando un grupo pequeño de participantes que sea representativo de la población objetivo ("*Small group pilot testing*" (Mark & Greer, 1993)).

Según lo anterior y teniendo en cuenta que:

- El proceso de desarrollo del estudio reportado en el presente documento, comparado con otras investigaciones que se han realizado en el campo de los sistemas tutores inteligentes, puede ser considerado muy corto y por lo tanto no apto para realizar afirmaciones formales sobre los resultados o comportamiento de un sistema, que tampoco, puede considerarse terminado. Para realizar este tipo de afirmaciones se requiere de la ejecución de estudios experimentales, en sistemas que ya hayan ejecutado varias pruebas piloto y que por lo tanto se puedan considerar terminados. Este proceso, puede tomar entre tres y cinco años de acuerdo a los reportes realizados por (Anderson, Corbett, Koedinger, & Pelletier, 1995; Baker et al., n.d.; Koedinger et al., 1997; Antonija Mitrovic, 2012).

- En este estudio comparativo, el mayor esfuerzo se ha dedicado a desarrollar los modelos de dominio y los componentes de interfaz gráfica necesarios para la ejecución de sistemas tutores inteligentes que impartan instrucción en temáticas relacionadas con los sistemas de control automático.

- Los productos tanto de software como de contenido se encuentran en una etapa de desarrollo estable pero aun podrían estar sujetos a modificación.

- Se tiene fácil acceso a un grupo de usuarios que potencialmente podrían probar el sistema y que representan muy bien la población objetivo.

Se decide seleccionar la prueba piloto tipo "*Small group pilot testing*" como mecanismo de evaluación para los sistemas tutores inteligentes desarrollados en este estudio comparativo.

5.1.2 Definición de las características para la evaluación de funcionamiento.

Una vez seleccionados los métodos de evaluación de los sistemas tutores inteligentes, es necesario realizar una selección y definición de las características que se van a evaluar cuando se realice la ejecución de laprueba piloto.

(Mark & Greer, 1993) no realizan un acercamiento a las diferentes características que se deben evaluar en cada uno de los métodos que abordan en su estudio, sin embargo estas características pueden ser extraídas de otros estudios relacionados que tienen bastante relevancia y relación con el estudio comparativo llevado a cabo en este proyecto. (Shute & Wesley, 1993) propusieron una lista de siete principios que deben ser considerados cuando se conduce un estudio de evaluación para sistemas tutores inteligentes. Uno de los siete principios, el número seis, se refiere a las pruebas piloto que se tienen que llevar a cabo tanto del sistema tutor inteligente como del diseño del estudio de evaluación. Para realizar las pruebas piloto del tutor se sugiere tener en cuenta los siguientes aspectos:

a) Se encuentra el tutor libre de malos funcionamientos de software?
b) Conocen los participantes de la prueba piloto, lo que deben estar realizando durante la prueba?
c) Aprenden algo los participantes de la prueba?
d) Indican los participantes de la prueba que les gusta el sistema?
e) Se ha estimado el tiempo de interacción apropiadamente?
f) Finalizan todos los participantes los ejercicios?

Adicionalmente (A. Mitrovic & Ohlsson, 1999) presentan un estudio de evaluación de un sistema tutor basado en restricciones en el cual se evalúan tres aspectos específicos: usabilidad, aprendizaje y efecto del uso del tutor en el desempeño de curso. De estos tres

aspectos el de mayor interés para este estudio comparativo, teniendo en cuenta las consideraciones presentadas en la sección anterior, es la usabilidad.

Para medir la usabilidad del sistema, (A. Mitrovic & Ohlsson, 1999) hacen uso de una encuesta como instrumento de captura de datos. La encuesta es presentada en la publicación y relacionando dicha encuesta con los aspectos establecidos en el principio seis de (Shute & Wesley, 1993) se logra evidenciar que es bastante apropiada no solo porque cubre los seis aspectos sino porque también es compatible con la definición de usabilidad emitida por la norma ISO/IEC 9126: “La usabilidad se refiere a la capacidad de un software de ser **comprendido**, **aprendido**, **usado** y ser **atractivo** para el usuario, en condiciones específicas de uso”.

Atendiendo los principios anteriores, se toma la decisión de seleccionar la encuesta presentada por (A. Mitrovic & Ohlsson, 1999) como una buena práctica metodológica para la evaluación de la usabilidad. Por lo tanto, se decide desarrollar un instrumento de captura de datos basado en encuesta. Dicho instrumentoestará basado en las preguntaspropuestas por los autores mencionados ydebe realizar las adaptaciones necesarias para el presente trabajo.

Las preguntas especificas relacionadas con usabilidad en el instrumento desarrollado para este trabajo son:

- Pudo entender fácilmente la representación de las expresiones matemáticas en la pantalla?
- Le resultó fácil de usar la interfaz gráfica?
- Considera que las gráficas presentadas, los elementos de interacción (campos y listas de selección) y los textos descriptivos son apropiados para la representación matemática del problema?

Otras preguntas incluidas en la encuesta del presente trabajo y que están relacionadas con los aspectos del principio seis de (Shute & Wesley, 1993) son:

Acerca de la actividad de interacción:

- Hora de Inicio
- Hora de Finalización
- Cuál es su experiencia en el análisis de respuesta de sistemas LTI?

Acerca del Uso y utilidad de mensajes de ayuda:

- En cuales pasos del procedimiento tuvo que pedir la respuesta final al tutor?
- Respecto a los mensajes de ayuda presentados por el sistema. Le parecieron útiles?
- Preferiría que los mensajes de ayuda fueran más detallados acerca de los procesos de cálculo?

Respecto a la motivación que causa el sistema y la aceptación del sistema como método de estudio:

- Qué tanto aprendió acerca del análisis de respuesta de respuesta de sistemas LTI?
- Una hora de interacción con este tipo de sistemas es más valioso que una de lectura del texto guía?
- Disfruto la experiencia de interacción con el sistema tutor inteligente?
- Le gustaría utilizar un Sistema tutor inteligente que abordara más temáticas relacionadas con la teoría de control?
- Le recomendaría a otros estudiantes el uso de Sistemas tutores inteligentes para el aprendizaje de temáticas de teoría de control o temáticas relacionadas?

Para la mayoría de las preguntas se usó una escala 5 niveles, mientras que para otras se utilizó una escala de 3 niveles.En el apéndice G se presenta la encuesta que se formuló a los usuarios y que corresponde alinstrumento para recolectar información acerca de las características de funcionamiento desde el punto de vista de los participantes.

Finalmente es de importancia mencionar que la medición de la ganancia en el aprendizaje de los estudiantes es una medida cuantitativa que normalmente se calcula para realizar reclamaciones formales acerca de la efectividad de un sistema tutor inteligente. Esta ganancia, es una medida que suele ser calculada durante etapas de evaluación acumulativa del sistema y que según (Mark & Greer, 1993), su medición se realiza a través de pruebas de campos o investigaciones experimentales. La alta complejidad con que cuenta la realización de estas investigaciones experimentales o pruebas de campo, dificulta que durante la ejecución de este proyecto se puedan ejecutar las actividades propias para la medición de la ganancia de conocimiento en los estudiantes. Es importante resaltar que dichas actividades deben estar orientadas a definir de forma correcta las características de un escenario de evaluación en el cual se seleccione de forma correcta aspectos como: cantidad de lecciones impartidas, duración de las lecciones, método de generación del grupo de control, aspectos que mejoren la

consistencia en la interacción con el sistema tutor inteligente. Dentro de la investigación experimental, se debe también como mínimo, poder controlar las siguientes variables, con el objetivo de permitir una generación de resultados libres de sesgo: Orden de presentación de los problemas, número de problemas disponibles en el tutor y tomados por los estudiantes, tiempo de interacción, datos demográficos como : edad, género, condiciones de conocimiento previo, número de estudiantes incluidos en la investigación y actividades alternas de refuerzo del conocimiento declarativo que usan los estudiantes que no interactúan con el sistema tutor inteligente. De esta forma, la medida de ganancia de aprendizaje en los estudiantes, no fue objeto de la prueba piloto que se describe en la siguiente sección.

5.2 Prueba Piloto

El primer paso para realizar la pruebas piloto con usuarios fue realizar la invitación de participación a las pruebas de evaluación. Esta invitación fue enviada a un total de 32 estudiantes que participaron durante el semestre lectivo 2014-3 en la cátedra de Sistemas de Control impartida en los Programas de Ingenierías Mecánica, Mecatrónica, Eléctrica y Electrónica de la Universidad Nacional de Colombia - sede Bogotá. La aceptación de la invitación para los estudiantes tuvo carácter voluntario. Como respuesta a la convocatoria 9 estudiantes se mostraron interesados en disponer del tiempo requerido.

La prueba piloto fue diseñada para tener una duración de máximo dos horas. Se dispusieron 2 estaciones de trabajo con sistema operativo Windows y una estación de trabajo con sistema operativo Mac OS X. La prueba pilotó se realizó durante un total de 6 horas durante las cuales los 9 estudiantes asistieron en grupos, cada uno de 3 integrantes. A cada grupo de estudiantes se les impartió las mismas instrucciones y se les puso a disposición el mismo instrumento evaluativo. La actividad de despliegue se realizó en las instalaciones del laboratorio de automatización, sala 411-101 en el campus universitario de la Universidad Nacional de Colombia - sede Bogotá.

Las instrucciones impartidas a los estudiantes incluyeron una presentación general de la actividad, en la cual se describía el objetivo de evaluar desde el punto de vista de un estudiante, que tiene conocimientos previos en el área de control, la usabilidad, funcionalidad y en general la experiencia de interacción con los sistemas tutores inteligentes.

Cada estudiante tuvo la oportunidad de interactuar con tres sistemas tutores inteligentes, cada uno dedicado a proveer asistencia e instrucción en problemas diferentes. Los problemas seleccionados para estas pruebas fueron los siguientes: Problema P4, Problema P7, Problema P8, Problema P9. Este conjunto de problemas está principalmente orientado al análisis de respuesta de sistemas LTI de primer y segundo orden.

De los tres sistemas tutores inteligentes que los estudiantes probaron, dos fueron creados utilizando una técnica (example-tracing o constraint-based) y el tercero fue creado utilizando la otra técnicaabordada por el estudio comparativo (constraint-based o example-tracing). De esta forma todos los estudiantes pudieron experimentar ambos tipos de sistemas y se pudo mantener el tiempo acotado a una longitud razonable que no causara cansancio o desmotivación durante la prueba piloto.

La prueba piloto estuvo siempre asistida por dos asistentes de la sala de laboratorio, el autor y el director de este trabajo. Durante la estancia en la sala, se asistió a los participantes para que pudieran tener acceso a los problemas puesto que el sistema Tutorshop presentaba un malfuncionamiento que no permitía secuenciar los tres problemas de forma correcta. Durante la prueba piloto no se impartieron ayudas verbales a los estudiantes, relacionadas con la temática de los problemas. En algunas ocasiones los estudiantes intentaron solicitar ayuda y la única instrucción que se impartió fue solicitar la ayuda al sistema tutor inteligente primero. De esta forma el rendimiento de los estudiantes durante las pruebas piloto nunca estuvo sujeta a conocimientos o ayudas impartidas por parte del grupo de asistencia durante la prueba.

Se logró evidenciar en dos estudiantes que participaron en el primer grupo, el excesivo uso de una calculadora científica como ayuda para realizar el cálculo de las fracciones parciales que algunos problemas requieren. A partir de ese momento se indicó a los estudiantes no realizar uso de esta ayuda, a menos que las ayudas brindadas por el sistema tutor no fueran suficientes. Se logró observar que el resto de los estudiantes, lograron avanzar en el procedimiento sin necesidad de la ayuda que provee dicho dispositivo.

El proceso general que se indicó a los estudiantes para realizar la actividad se resume en los siguientes pasos:

1. Reciba información general por parte del asistente de sala acerca de la actividad

2. Lea atentamente los anexos del cuestionario, en especial el consentimiento de participación.
3. Diligencie la primera pregunta del cuestionario.
4. Indique al asistente de sala su disponibilidad para iniciar la interacción con los tutores.
5. Registre la Hora de Inicio del problema 1
6. Inicie la resolución del problema 1.
7. Registre la hora de finalización del ejercicio.
8. Avise al asistente de sala cuando finalice la resolución del problema
9. Diligencie las preguntas relacionadas con el problema.
10. Ubíquese en una nueva estación de trabajo si es necesario.
11. Repita los pasos 5 al 11.
12. Finalice el diligenciamiento del cuestionario.
13. Entregue el cuestionario al asistente de sala.

Debido a que algunas estaciones de trabajo del laboratorio en el que se realizaron las pruebas tienen un mejor rendimiento que otras y dado que las interfaces creadas en lenguaje Java exigen un rendimiento mayor que aquellas creadas en tecnologías Web, en algunos casos, se solicitó a los estudiantes que se cambiaran de estación para que pudieran trabajar en otra que cumplía los requerimientos de la tecnología usada.

Cada vez que un participante finalizaba la actividad y entregaba el formulario, se realizó una discusión corta, fuera de la sala, en donde se solicitaba a los estudiantes informar sus comentarios acerca de la experiencia. Mediante esta práctica se pudo tener una discusión con los participantes y por lo tanto tener un mayor entendimiento de los comentarios, opiniones y valoraciones que registraron en el cuestionario.

5.3 Resultados de la evaluación.

5.3.1 Prueba Piloto

Los datos recolectados con la encuesta realizada fueron analizados y varias conclusiones se pudieron obtener a partir de su estudio. En las siguientes tablas se presentan las valoraciones otorgadas por los participantes a las preguntas realizadas en la encuesta. Cada tabla se

presenta agrupando varias preguntas que permiten medir una misma característica de funcionamiento de los sistemas tutores inteligentes.

Tabla 24. Resultados de la prueba: Tiempos de interacción, sistemas probados y experiencia previa.

Interacción con el sistema y experiencia previa.			
Usuario	Tiempo de interacción	Sistema	Experiencia Previa
unal1	0:10	ASPIRE	Solamente la clase de control.
	0:14	ASPIRE	
	0:13	CTAT	
unal2	0:07	ASPIRE	Solamente la clase de control.
	0:15	ASPIRE	
	0:21	CTAT	
unal3	0:09	ASPIRE	Solamente la clase de control.
	0:10	ASPIRE	
	0:14	CTAT	
unal4	0:30	CTAT	La clase de control y otras clases relacionadas.
	0:12	CTAT	
	0:07	ASPIRE	
unal5	0:20	CTAT	Solamente la clase de control.
	0:11	CTAT	
	0:13	ASPIRE	
unal6	0:20	CTAT	Solamente la clase de control.
	0:15	CTAT	
	0:18	ASPIRE	
unal7	0:17	CTAT	Solamente la clase de control.
	0:19	CTAT	
	0:15	ASPIRE	
unal8	0:16	CTAT	Solamente la clase de control.
	0:05	ASPIRE	
	0:14	CTAT	
unal9	0:13	ASPIRE	Solamente la clase de control.
	0:15	ASPIRE	
	0:21	CTAT	
Promedio	0:14		

El tiempo promedio de interacción requerido por los estudiantes fue de 14 minutos por cada problema considerado. El máximo tiempo que un estudiante experimentó fue de 30 minutos y el

mínimo fue de 5 minutos, para la gama de problemas considerados. Las recomendaciones realizadas por los líderes en investigación sugieren que las interacciones de tutores inteligentes deben estar en el rango de los 5 a 10 minutos de interacción(Aleven et al., 2009).Sin embargo para el dominio de sistemas de control, el número de procedimientos que un estudiante tiene que realizar para resolver un problema completo (que tiene un alto grado de relevancia, donde se utilizan conceptos generales y donde la estructura de objetivos es alta); puede tomar de10 a 60 minutos utilizando papel y/o tablero y depende del grado de conocimientos del estudiante y de las temáticas específicas.

Se realizaron 13 usos de los sistemas tutores creados con la herramienta ASPIRE y 14 usos de los que fueron creados con la herramienta CTAT, lo cual nos permite establecer que los sistemas tutores inteligentes fueron usados en igual medida.

Solo un estudiante reportó tener experiencia en más de una cátedra relacionada con la teoría de control. El resto de los estudiantes reportaron tener solo el conocimiento adquirido durante la mayor parte de un semestre académico. Para la fecha en que se realizó la prueba piloto,los estudiantes ya contaban con el conocimiento declarativo requerido en la temática de análisis de respuesta de sistemas LTI. Este resultado es muy importante porque los participantes de la prueba piloto son una muestra representativa de la población ideal a la cual está orientado este tipo de sistemas. Debido a que la interacción con los sistemas tutores inteligentes refuerza el conocimiento declarativo y permite desarrollar el conocimiento procedimental, es un requisito que los usuarios de estos sistemas tengan el conocimiento declarativo antes de interactuar con el sistema, de otra forma la interacción no podrá cumplir con su objetivo. Para usuarios muy expertos los problemas presentados por el sistema tutor inteligente pueden ser muy simples y pueden generar desmotivación. De esta forma, para los problemas que fueron implementados y presentados a los estudiante, se considera que la población de participantes está ajustada a lo deseado y que por lo tanto sus evaluaciones y opiniones tienen gran validez.

En la siguiente tabla se muestran las valoraciones otorgadas por los estudiantes, en relación a la usabilidad del sistema.

Tabla 25. Resultado de la evaluación de usabilidad.

Usabilidad

Usuario	Sistema	Facilidad	Adecuada Representación	Calidad de los textos y graficas
unal1	ASPIRE	5	Si	5
	ASPIRE	5		5
	CTAT	4		5
unal2	ASPIRE	4	Si	4
	ASPIRE	4		4
	CTAT	3		4
unal3	ASPIRE	5	Si	4
	ASPIRE	4		4
	CTAT	4		5
unal4	CTAT	5	Si	4
	CTAT	5		5
	ASPIRE	5		5
unal5	CTAT	4	Si	5
	CTAT	4		5
	ASPIRE	5		5
unal6	CTAT	5	Si	4
	CTAT	5		5
	ASPIRE	5		4
unal7	CTAT	5	Si	5
	CTAT	3		4
	ASPIRE	3		4
unal8	CTAT	5	Si	4
	ASPIRE	4		5
	CTAT	5		5
unal9	ASPIRE	4	Parcialmente	4
	ASPIRE	4		5
	CTAT	4		4
Promedios		4.37037037	0.88	4.51

Las valoraciones otorgadas por los estudiantes a las preguntas relacionadas con usabilidad del sistema fueron en general altas. En una escala de 1 a 5, el promedio de las valoraciones fue 4.37, la valoración máxima fue 5 y la valoración mínima fue 3. Solamente un estudiante (12%) se mostró parcialmente en desacuerdo con la representación gráfica que se dio a una fracción numérica en un problema específico. Dicho estudiante no considera que la representación con la barra diagonal "/" sea una representación apropiada para mostrar una fracción. Sin embargo, la representación fue comprendida y no impidió que el estudiante resolviera el problema. De

hecho esta opinión fue expresada en la encuesta y nunca evitó que el estudiante finalizara el problema.

Respecto a la calidad de los gráficos y de los textos presentados, la valoración promedio fue de 4.51, siendo 5 la máxima valoración otorgada y 4 la mínima. En general, la opiniones verbales y las opiniones escritas en la encuesta, evidencian que los estudiantes se sienten cómodos con los gráficos, textos y distribución de los mismos a través de los enunciados del problema. Este buen resultado se puede explicar gracias a que el proceso de desarrollo de los problemas se realizó con la participación de un experto en el área, el director de tesis. Adicionalmente el esfuerzo por mantener las representaciones en la interfaz gráfica lo más similares posibles a las representaciones en libros y artículos parece ser una buena decisión que ayuda en el entendimiento y uso del sistema.

En la siguiente tabla se presentan las valoraciones otorgadas por los participantes de la prueba piloto en relación a la utilidad de los mensajes de ayuda y al uso que hicieron de estos.

Tabla 26. Resultados de evaluación: Uso y utilidad de los mensajes de ayuda

Uso y utilidad de mensajes de ayuda				
Usuario	**Sistema**	**Usó ayuda final?**	**Utilidad mensajes de ayuda**	**Mas detalle ayuda?**
unal1	ASPIRE	No	5	NR[4]
	ASPIRE		5	No
	CTAT		4	Si
unal2	ASPIRE	No	4	Si
	ASPIRE		3	Si
	CTAT		5	Si
unal3	ASPIRE	No	5	No
	ASPIRE			NR
	CTAT			NR
unal4	CTAT	No	5	No
	CTAT			NR
	ASPIRE		5	Si
unal5	CTAT	Si	5	Si
	CTAT			NR
	ASPIRE		5	No

[4] No responde

unal6	CTAT	Si	5	No
	CTAT		4	No
	ASPIRE		5	No
unal7	CTAT	Si	4	No
	CTAT		3	SI
	ASPIRE		3	Si
unal8	CTAT	No	5	No
	ASPIRE		5	Si
	CTAT			No
unal9	ASPIRE	No	5	No
	ASPIRE		4	No
	CTAT		4	No
Promedio		0.333	4.45	0.40

Durante la prueba piloto se pudo observar que los estudiantes experimentabanatascamientos en diferentes partes de los problemas. En general, las secciones que causaron mayor dificultad a los estudiantes fueron las relacionadas con el desarrollo de fracciones parciales durante el cálculo de transformadas inversas de Laplace. Para estas actividades donde los estudiantes enfrentaban problemas, se observó que en la mayoría de los casos las sugerencias otorgadas por el sistema tutor inteligente eran suficientes para que el estudiante pudiera corregir los procedimientos incorrectos y lograra continuar con el desarrollo. Algunos estudiantes se les observó menos ansiosos de pedir ayuda del sistema, mientras que otros hicieron uso de la ayuda ampliamente. Según la tabla anterior, 3 estudiantes (33.3%) hicieron uso de la funcionalidad del sistema tutor inteligente que permite consultar la respuesta correcta al paso que el estudiante ha resuelto de forma incorrecta. El promedio de la valoración otorgada a la pregunta que indaga que tan útil fueron los mensajes de ayuda, fue de 4.45. Sin embargo también se observa que algunos estudiantes no hicieron uso de los mensajes de ayuda, debido a que su desarrollo fue libre de errores para algunos problemas. Un estudiante reportó que aún cuando sus respuestas eran correctas, realizó una exploración por el sistema de ayuda para evaluar su utilidad.

Para finalizar, en el 40% de utilizaciones, los estudiantes reportaron que se pueden incluir mas detalles en los mensajes de ayuda. Este es un aspecto de gran complejidad, debido a que incluir más ayuda en el sistema podría hacerlo de dificultad baja y por lo tanto reducir en los estudiantes su capacidad de esfuerzo por lograr una respuesta correcta. En general, lo que se puede concluir del hecho en la prueba piloto y del comentario anterior, es que las herramientas

necesitan adaptar los niveles de ayuda que el estudiante requiera. El nivel de adaptación que las herramientas tienen en su actual nivel de desarrollo no es muy alto debido a que durante su construcción no existe la posibilidad de definir reglas que cambien el funcionamiento del sistema de ayuda, a partir de datos estadísticos del desempeño del estudiante durante el problema. Este tipo de adaptación es más común en las funcionalidades de lazo externo de un sistema tutor inteligente. Sin embargo, la anterior estadística es una muestra, que si se implementara dicha adaptación en el lazo interno, se podría obtener una gran mejora en términos de individualización.

En la siguiente tabla se presentan las valoraciones otorgadas por los participantes de la prueba piloto en relación a la motivación que el sistema causa en el estudiante y a la aceptación del sistema como método de estudio..

Tabla 27. Resultados de evaluación: Motivación y Aceptación.

		Motivación que causa el sistema		Aceptación como método de estudio		
Usuario	**Sistema**	**Que tanto aprendió**	**Que tanto disfruto la experiencia**	**Más valioso que 1 hora de libro.**	**Mas temas de control**	**Recomendaría el uso**
unal1	ASPIRE	4	5	Si	Si	Si
	ASPIRE					
	CTAT					
unal2	ASPIRE	2	4	No	Si	Si
	ASPIRE					
	CTAT					
unal3	ASPIRE	4	4	Si	Si	Si
	ASPIRE					
	CTAT					
unal4	CTAT	4	5	Si	Si	Si
	CTAT					
	ASPIRE					
unal5	CTAT	5	5	Si	Si	Si
	CTAT					
	ASPIRE					
unal6	CTAT	3	4	Si	Si	Si
	CTAT					
	ASPIRE					
unal7	CTAT	4	5	Si	Si	Si
	CTAT					

	ASPIRE					
unal8	CTAT	4	4	Si	Si	Si
	ASPIRE					
	CTAT					
unal9	ASPIRE	4	4	Si	Si	Si
	ASPIRE					
	CTAT					
Promedio		3.77	4.44	88%	100%	100%

A la pregunta que indaga qué tanto aprendió un estudiante durante la actividad, la valoración promedio fue de 3.77. Mientras que para la pregunta que indaga si el estudiante disfrutó la experiencia, el promedio fue 4.44. El valor de 3.77 resulta bajo respecto a otros indicadores, debido a que varios estudiantes reportaron este indicador con valores de 3 y 2. Es un resultado entendible debido a que la cantidad de conceptos que se pueden aprender con una interacción de apenas 45 minutos no es grande. Sin embargo los estudiantes reportan que disfrutan en muy buen grado la interacción del sistema valorando esta característica con un promedio de 4.44, siendo 4 la calificación mínima otorgada. Respecto a la aceptación de este tipo de sistemas como método de estudio, los estudiantes se muestran especialmente motivados: 100% lo recomendarían y desean tener más problemas para resolver en la temática de control. Durante las discusiones con los estudiantes al finalizar la prueba piloto se les observó bastante satisfechos con el sistema y realizaron comentarios bastante útiles para futuros desarrollos. A continuación se muestra una tabla con los comentarios mas destacados de cada participante.

El 88% de los estudiantes mencionaron que 1 hora de interacción con los sistemas tutores inteligentes es mas valiosa que una hora de lectura en un texto guía. Esto de ninguna manera significa que los libros puedan ser reemplazados por este tipo de sistemas. Como ya se mencionó en esta sección, el conocimiento declarativo es necesario, un pre-requisito para iniciar la interacción con estos sistemas y además dicho conocimiento, se adquiere normalmente de las lecturas y la participación en clases grupales. Lo que más se debe resaltar de esta pregunta, es que se hizo referencia a los estudiantes de las diferencias que existen con un libro convencional. Durante los comentarios escritos y verbales hechos por los estudiantes, ellos mencionaron con bastante seguridad que la capacidad que tienen los sistemas tutores inteligentes para proveer ayuda solamente en los pasos en los que se requiere, genera una gran diferencia con los textos convencionales. Algunos mencionan que los libros son muy

simples y otros que son muy complejos. De esta forma el sistema tutor inteligente, al proveer ayuda progresiva se ajusta más fácilmente a las necesidades de cada estudiante.

Los textos mostrados corresponden en general a un resumen de los comentarios registrados por los estudiantes. Se ha elegido esta manera de presentar sus comentarios debido a que los textos originales son mucho más largos; sin embargo como parte de los anexos digitales a este documento se dejará como registro los archivos originales diligenciados durante la prueba piloto, para efectos de archivo.

Al realizar un análisis detallado de los textos que los estudiantes registraron en las encuestas, se observan los siguientes hechos:

- 2 de 9 (22%) estudiantes se mostraron dispuestos a introducir texto en forma de sintaxis matemáticas para incluir las respuestas. Esto es bastante importante porque confirma que algunos estudiantes pueden ver una opción interesante si se implementara dicha forma de interacción. La implementación de esta forma de interacción no se abordó en este trabajo por requerir el uso de funciones muy avanzadas que no se encuentran disponibles en los entornos de ejecución que CTAT y ASPIRE ofrecen. Sin embargo abren una muy buena opción para los trabajos futuros.
- 5 de 9 (55%) estudiantes se mostraron interesados en poder incluir funcionalidades de graficado de las expresiones y de incluir procesos que modificación de parámetros para entender los efectos de los estos en las gráficas. Esta es otra funcionalidad que resultaría en mayor motivación, aprendizaje si se integrara a los sistemas tutores inteligentes creados.
- 9 de 9 (100%) estudiantes reconocieron que las interfaces en Java son mas lentas que las implementadas en HTML. Este hecho fue bastante fácil de reconocer debido a que el tiempo de respuesta de la interfaz gráfica (responsiveness) era de alrededor de 1 segundo o 2 al seleccionar una opción. En general este hecho no es típico de los sistemas desarrollados en lenguaje Java que han sido bien diseñados, sin embargo para efectos de la prueba piloto se activaron unas características de CTAT conocidas como Logging. Se pudo establecer que al desactivar estas características el tiempo de respuesta se reduce a los valores normales, sin embargo en un despliegue real estas características deben estar activas puesto que es la forma en que se recolecta información de uso de la herramienta para hacer análisis de minería de datos. Según lo

anterior, se considera que este tiempo de respuesta en un despliegue real es bajo y que por lo tanto debe ser reportado de esta forma.

- Solo un estudiante reportó haber tardado 30 minutos en comprender las funciones del sistema y como utilizarlo. Sin embargo fue el estudiante que hizo comentarios más detallados, incluyendo aspectos como: Internacionalización, homogeneidad de las imágenes, uso de scrolls, Posibilidad de ampliación de las imágenes, Facilidad de adivinar las respuestas, contraste de las gráficas.
- 66% de los estudiantes reconocieron la capacidad de los tutores example-tracing para dar realimentación inmediata y se sintieron complacidos y más satisfechos con este tipo de ayuda. Es notable puesto que no se les mencionó en ningún momento que esta diferencia existe. Muchos de los comentarios hechos por los estudiantes, muy correctos por supuesto, es que tener la realimentación tan pronto como se pueda les ayuda a evitar errores en futuros procedimientos y les ayuda a aprender más rápido. Esto es una característica bien reconocida tanto de los tutores model-tracig como example-tracing. Sin embargo el otro 33%, menciona que la posibilidad de tener realimentación retrasada le da mayor posibilidad al estudiante de hacer un esfuerzo reflexivo por encontrar la respuesta correcta. Esta posición también es correcta y referenciada en artículos donde se exponen este tipo de sistemas. De igual manera que el nivel de detalle de la ayuda, este aspecto debería poder adaptarse basado no en configuraciones, como en las herramientas actuales, sino ser calculada en tiempo de ejecución, a partir de datos estadísticos del rendimiento del estudiante en el lazo interno y externo.
- el 33% de los estudiantes reportaron que se podría incluir más flexibilidad en el tratamiento de las expresiones matemáticas: 2 de ellos sugieren incluir campos de texto donde se pueda incluir sintaxis matemática, uno de ellos sugiere que la estructura es rígida y no se permiten hacer simplificaciones posibles. Sin embargo otro 33% de estudiantes reconocieron que la forma en que se pide proporcionar la respuesta presenta ventajas porque induce a realizar operaciones matemáticas adicionales que merecen algún esfuerzo.

5.3.2 Evaluación mediante el método de Inspección de expertos

La metodología de inspección de expertos también se aplicó como mecanismo de evaluación de los componentes de conocimiento e interfaz gráfica durante varias iteraciones de desarrollo.

Debido a que el número de iteraciones fue considerable, no se puede hacer un reporte completo de todas las inspecciones y sus resultados. Sin embargo en el apéndice H, se presenta el reporte de la última inspección realizada por un experto en el área de sistemas de control, antes de la prueba piloto. Se evidencia que muchos de los cambios sugeridos están orientados a garantizar la precisión de los textos, gráficas y soluciones a ciertos problemas, cumpliendo de esta manera los objetivos planteados por este mecanismo de evaluación formativo.

6. Conclusiones

En este trabajo se ha presentado un estudio comparativo del funcionamiento de diferentes tipos de sistemas tutores inteligentes orientados a la enseñanza de los fundamentos de control automático. Los aspectos del funcionamiento que abarca el estudio son: la teoría subyacente que soporta los diferentes tipos de sistemas tutores inteligentes, el modelamiento de dichos sistemas para cumplir los objetivos de aprendizaje propios de la teoría de control, las herramientas disponibles para su construcción, los aspectos de implementación específicos para cada tipo de sistemay una evaluación basada en la valoración que usuarios reales del sistema realizaron por medio de una prueba piloto.

El estudio ha presentado una metodología para la selección de los problemas que tienen una mejor capacidad, para lograr que el sistema tutor inteligente final presente un conjunto de buenas prácticas de desarrollo conocidas y bien probadas para la creación de este tipo de sistemas.

El estudio comparativo ha mostrado en paralelo, la forma detallada como se realiza el modelamiento teórico de aspectos típicos del conocimiento de la teoría de control, utilizando tres técnicas de modelamiento para la creación de sistemas tutores inteligentes: model-tracing, example-tracing, constrained-based.

El estudio comparativo ha exhibidolas capacidades delas herramientas de autor, para la construcción de los sistemas tutores inteligentes y cómo las diferentes capacidades afectan de mayor o menor manera el desarrollo de sistemas funcionalespara el dominio de conocimiento de los sistemas de control.

El estudio comparativo ha presentado un conjunto de requerimientos de conocimiento que los sistemas tutores inteligentes deben estar en capacidad de cumplir, con el objetivo de garantizar que pueden enseñar y comprobar que los estudiantes han aprendido las

habilidades requeridas y necesarias para tener un buen desempeño durante la participación de una cátedra en fundamentos de sistemas de control.

Durante la ejecución de las etapas de desarrollo de los sistemas tutores inteligentes sujetos al estudio comparativo, se han encontrado diversos retos tecnológicos que, aplicando conocimiento específicos en desarrollo de software, fueron resueltos de diferentes formas permitiendo que los productos finales cumplan con unos requisitos en términos de diseño y funcionalidad. Estos retos se reportarony se han contrastado con el propósito de hacer más evidente sus similitudes y diferencias.

Por medio de la revisión detallada de la literatura existente en el campo de los sistemas tutores inteligentes, se tomaron decisiones acerca de un mecanismo de evaluación que satisface los escenarios este estudio comparativo y que permitió realizar el diseño y ejecución de una prueba piloto en la cual algunos estudiantes que representan la población objetivo, participaron y valoraron de forma positiva las decisiones de diseño y de implementación tomadas durante la etapa de desarrollo.

A la luz de los resultados obtenidos y después de ejecutar todas las etapas del estudio comparativo se proponen los sistemas tutores inteligentes tipo example-tracing como el tipo de sistema que genera una mejor relación entre los costos de su desarrollo y los beneficios de sus funcionalidades y capacidades; cuando se requiere construir sistemas tutores inteligentes para soportar la enseñanza de los fundamentos de control.

Los sistemas tutores inteligentes tipo constraint-based, se consideran como elsegundo mejor tipo de sistema respecto a la relación entre los costos de su desarrollo y los beneficios de sus funcionalidades y capacidades. Aunque sus tiempos de desarrollo son algo menores respecto a example-tracing, las funcionalidades del producto final también lo son. Adicionalmente, la ganancia en tiempo se obtiene gracias al generador automático de restricciones, herramienta que para el modelamiento avanzado de problemas, no podría ser utilizada lo que aumentaría los tiempos de desarrollo.

Los sistemas tutores inteligentes tipo model-tracing, proveen funcionalidades similares a los que proveen los sistemas tipo example-tracing, pero su desarrollo es alrededor de diez veces más costoso. Aún así, es el tipo de sistema tutor inteligente que presenta una fidelidad cognitiva más alta; permite modelar con toda fidelidad el proceso mental del estudiante a través de reglas de producción. Este hecho, en combinación con el hecho

que presentan las mismas funcionalidades que los tutores tipo example-tracing, los hacen el tipo de sistema tutor más completo y robusto, pero no el más práctico.

Desde un punto de vista menos teórico y más relacionado con la implementación de sistemas funcionales, se observa que las herramientas de autor disponibles tienen algunas deficiencias de funcionalidad tanto para el autor como para los usuarios avanzados (programadores). En especial se destacan los mecanismos de extensibilidad, los cuales no muestran la robustez típica que se puede identificar en otros sistemas de software y los sistemas de licenciamiento que hacen difícil parala comunidad en general, iniciar procesos de colaboración en el desarrollo de nuevas funcionalidades para las herramientas.

Estas carencias limitan la posibilidad de realizar algunas implementaciones prácticas, pero no limitan las posibilidades teóricas de cada formalismo. Por lo cual, la implementación de comportamientos avanzados en sistemas que se construyan utilizando cualquiera de las técnicas presentadas, es siempre posible; las limitaciones solo estarán dadas por las tecnologías seleccionadas para la implementación y la cantidad de funcionalidades que se implementen.

La construcción de un marco de trabajo que sea extensible y escalable, que se construya basado en estándares conocidos, como los estándares web, que soporte un número amplio de tecnologías, que tenga un mecanismo de licenciamiento que lo haga sujeto de uso masivo y que implemente funcionalidades para tipos de usuariosbásicos y avanzados, podría ayudar en la masificación de los sistemas tutores inteligentes.

Un marco de trabajo como el descrito permitiría que usuarios básicos puedan sacar provecho de las automatizaciones y ayudas típicas de las herramientas de autor, mientras que permitiría a los usuarios avanzados desarrollar de forma abierta y colaborativa nuevas funcionalidades que luego puedan ser agregadas, configuradas y orquestadas por los usuarios básicos que son la mayor cantidad de usuarios disponibles.

Desde el punto de vista de la teoría de control se presenta un reporte completo de las metodologías empleadas y de cada uno de los aspectos que son tenidos en cuenta para la construcción de los sistemas tutores inteligentes en este campo de estudio. Próximos desarrollos en este campo, orientados a la creación de sistemas tutores

inteligentes,podrían utilizar el reporte realizado como una guía para reconocer las habilidades que un sistema tutor inteligente deba contemplar.

La identificación del conjunto de requerimientos de conocimiento que se realizó, puede ser modelado utilizando cualquiera de las tres técnicas presentadas. Se presentaron ejemplos de la forma como se podría llegar a realizar dicho modelamiento de forma exhaustiva. Este primer acercamiento asumefunciones y estructuras de datosque pueden ser diseñadas e implementadas en el corto plazo con el objetivo de iniciar la creación de un conjunto de sistemas tutores inteligentes que tengan un buen nivel de generalidad y adicionalmente respondan a las necesidades de aprendizaje del campo de estudio de la teoría del control.

Anexo A: Escalafón de prioridad para implementación de problemas.

Descripción	Objetivos de Aprendizaje relacionados	Relevancia	Estructura de Objetivos	Generalidad	Puntaje Final
Problema P1: Presentar una descripción del funcionamiento de un sistema en la cual se haga referencia a dos variables: una que resulta ser causa de otra que se considera efecto. A partir de la descripción solicitar al estudiante que identifique la variable de entrada y la variable de salida del sistema.	OB.1 OB.2	BAJA	BAJA	MEDIA	4
Problema P2: Presentar cuatro diagramas de sistemas. Para cada diagrama se presenta un sistema en donde el flujo de entrada se dirige de derecha a izquierda, arriba hacia abajo, abajo hacia arriba e izquierda derecha y solicitar al estudiante identificar la entrada y la salida del sistema para cada uno de los 4 sistemas.	OB.1 OB.2	BAJA	BAJA	MEDIA	4
Problema P3: Dada una ecuación diferencial lineal de primer orden, se asumen condiciones iniciales cero. Solicitar al estudiante aplicar la transformada de Laplace y realizar el manejo matemático requerido para calcular la función de transferencia.	OB.1 OB.2	ALTA	ALTA	ALTA	9
Problema P4: Para el problema **P3**, considerar condiciones iniciales diferentes de cero y solicitar los mismos resultados.	OB.1 OB.2	ALTA	ALTA	ALTA	9
Problema P5: Dada una función de transferencia de primer orden solicitar al estudiante obtener una ecuación diferencial que represente el sistema en el dominio del tiempo asumiendo condiciones iniciales cero.	OB.1 OB.2	ALTA	ALTA	MEDIA	8
Problema P6: Dada una ecuación diferencial lineal de segundo orden y condiciones iniciales cero. Solicitar al estudiante aplicar la transformada de Laplace y realizar el manejo	OB.1 OB.2	ALTA	ALTA	ALTA	9

matemático requerido para calcular la función de transferencia.					
Descripción	**Objetivos de Aprendizaje relacionados**	**Relevancia**	**Estructura de Objetivos**	**Generalidad**	**Puntaje Final**
Problema P8: Dada una *ecuación diferencial* de primer ó segundo orden y una entrada definida en el dominio del tiempo, solicitar al estudiante calcular la salida del sistema en el dominio del tiempo.	OB.1 OB.2	ALTA	ALTA	ALTA	9
Problema P9: Dada *una función de transferencia* de primer/segundo orden y una función de entrada definida en el dominio del tiempo solicitar al estudiante calcular la salida del sistema en el dominio del tiempo.	OB.1 OB.2	MEDIA	MEDIA	ALTA	8
Problema P10: Dada una *función de transferencia* de primer/segundo orden y una función de entrada definida en el *dominio del laplace* solicitar al estudiante calcular la salida del sistema en el dominio del tiempo.	OB.1 OB.2	MEDIA	ALTA	MEDIA	7
Problema P11: Dada una *función de transferencia* de primer/segundo orden y una función de entrada definida en el *dominio del tiempo* solicitar al estudiante calcular la salida del sistema en el dominio de laplace.	OB.1 OB.2	BAJA	ALTA	MEDIA	6
Problema P12: Dada una *función de transferencia* de primer/segundo orden y una función de entrada definida en el dominio del laplace solicitar al estudiante calcular la salida del sistema en el dominio de laplace.	OB.1 OB.2	BAJA	MEDIA	BAJA	4
Problema P13: Presentar al estudiante diferentes gráficas de comportamientos típicos de respuesta y solicitar al estudiante identificar la gráfica que presenta: P13.1) El comportamiento de primer orden P13.2) El comportamiento de segundo orden sobre amortiguado	OB.1 OB.2	ALTA	BAJA	MEDIA	6

P13.3) El comportamiento de segundo orden críticamente amortiguado P13.4) El comportamiento de segundo orden sub-amortiguado. P13.5) El comportamiento inestable sin oscilaciones P13.6) El comportamiento oscilatorio sostenido. P13.7) El comportamiento oscilatorio inestable.					
Problema P14: Presentar de forma grafica los 3 posibles paradigmas de control y presentar los tres nombres de cada paradigma no relacionados espacialmente con las representaciones gráficas. Solicitar al estudiante relacionar cada uno de los nombres de paradigma con cada representación gráfica correspondiente.	OB.3.	ALTA	BAJA	BAJA	5
Problema P15: Presentar al estudiante una descripción de un problema en la que tanto la función de entrada como la función de transferencia queden absolutamente determinadas por los datos del problema. Se solicita al estudiante que identifique el paradigma de control teniendo como tres opciones: Control, Análisis de desempeño e Identificación.	OB.3.	ALTA	BAJA	ALTA	7
Problema P16: Para el problema P15. Modificar la descripción del problema para que a partir de su descripción quede totalmente determinada la función de transferencia y la salida deseada del sistema. Solicitar al estudiante identificar el paradigma.	OB.3.	ALTA	BAJA	ALTA	7
Problema P17: Para el problema P16. Modificar la descripción del problema para que a partir de su descripción quede totalmente determinada la función de entrada y la función de salida. Solicitar al estudiante identificar el paradigma.	OB.3.	ALTA	BAJA	ALTA	7
Problema P18: Presentar una grafica de respuesta de un sistema en el dominio del tiempo. Solicitar al estudiante identificar el periodo	OB.4. OB.5.	ALTA	BAJA	BAJA	5

transitorio y el periodo estacionario.					
Problema P19: Presentar una pareja de gráficas mostrando la entrada y salida de un sistema. La entrada debe tener cambio abruptos y zonas estables por lo tanto la salida también debe presentar transitorios y zonas de estado estable. Solicitar al estudiante que identifique los periodos transitorios de la señal y los periodos estacionarios.	OB.4. OB.5.	MEDIA	MEDIA	ALTA	7
Problema P20: Dada una ecuación diferencial SISO LTI de primer orden, solicitar al estudiante calcular la función de transferencia de un sistema cuya dinámica se describe por la ecuación diferencial.	OB.6. OB.7.	BAJA	MEDIA	MEDIA	6
Problema P21: Dada la función de transferencia del problema P21. Para una entrada paso de amplitud A solicitar al estudiante calcular la expresión de la salida en el dominio de Laplace y en el dominio del tiempo.	OB.6. OB.7.	BAJA	ALTA	ALTA	7
Problema P22: Dada la función de transferencia del problema P21. Para una entrada rampa de pendiente M calcular las expresiones para la salida en el dominio de Laplace y tiempo.	OB.6. OB.7.	BAJA	ALTA	ALTA	7
Problema P23: Para una entrada parábola solicitar al estudiante calcular las expresiones para la salida en el dominio de Laplace y del tiempo.	OB.6. OB.7.	BAJA	ALTA	ALTA	7
Problema P24: Dada una función de transferencia de la forma G(s)=a/bs+c calcular la expresión de salida en el tiempo, sin utilizar la transformada de Laplace para entradas paso, rampa y parábola.	OB.6. OB.7.	MEDIA	MEDIA	BAJA	5
Problema P25: Dada una función de transferencia de la forma G(s)=a/bs+c calcular la expresión de salida en el tiempo, para una entrada definida por una función que no hace parte del conjunto de funciones de prueba estándar.	OB.6. OB.7.	MEDIA	ALTA	ALTA	7

Problema P26: Para una función de transferencia se desconoce las constantes K y Tau. El sistema se excita con una entrada paso de amplitud A. Se conoce la respuesta del sistema la cual se muestra en una gráfica. La gráfica muestra los ejes Y y T correctamente marcados. Solicitar al estudiante calcular las constantes Tau y K. Se solicita diligenciar la tabla de valores con las parejas [t,Y(t)] donde t=N*tau con N=1,2,3,4,5,6	OB.6. OB.7.	ALTA	ALTA	ALTA	9
Problema P27: Para una función de transferencia de la forma G(S)=K/Tau*S+1 se muestran cuatro posibles gráficas para la salida del sistema cuando el sistema ha sido excitado con una función paso de amplitud A. Se solicita al estudiante identificar la gráfica que representa correctamente el comportamiento del sistema.	OB.6. OB.7.	BAJA	MEDIA	BAJA	4
Problema P28: Para un sistema SISO LTI definido por una ecuación diferencial de la forma a1*(dy/dt),a0*y = bo*U(t) , Solicitar al estudiante calcular la función de transferencia.	OB.6. OB.7.	BAJA	MEDIA	ALTA	6
Problema P29: Para una función de transferencia de la forma G(S)=bo/(a1*s+a0) solicitar al estudiante calcular la ecuación diferencial que describe la dinámica del sistema.	OB.6. OB.7.	BAJA	ALTA	MEDIA	6
Problema P30: Dada una ecuación diferencial SISO LTI de segundo orden y una descripción de las señales de entrada y salida. Solicitar al estudiante reconocer los valores de las variables de parametrización K, Zeta y Wn junto con sus unidades.	OB.8. OB.9. OB.10. OB.11.	MEDIA	ALTA	MEDIA	7
Problema P31: Dada una función de transferencia G(S) de segundo orden de la forma a/bs^2+cs+d solicitar al estudiante reconocer los valores de las variables de parametrización K,Zeta y Wn y reescribir la función de transferencia usando las nuevas variables.	OB.8. OB.9. OB.10.	MEDIA	BAJA	BAJA	4

	OB.11				
Descripción	**Objetivos de Aprendizaje relacionados**	**Relevancia**	**Estructura de Objetivos**	**Generalidad**	**Puntaje Final**
Problema P33: Dada una función de transferencia de la forma G(S)=C*Kwn^2/C*S^2+2*C*zeta*Wn*S+C*Wn^2 y una entrada paso. Solicitar al estudiante ingresar la ecuación característica y las dos raíces que conforman la solución de la ecuación característica.	OB.8. OB.9. OB.10. OB.11	ALTA	MEDIA	MEDIA	7
Problema P34: Dado un valor de Zeta mayor o igual que cero. Solicitar al estudiante reconocer el tipo de comportamiento que se genera en la respuesta transitoria.	OB.8. OB.9. OB.10. OB.11	BAJA	BAJA	BAJA	3
Problema P35: Dada una función de transferencia de segundo orden. Solicitar al estudiante reconocer el tipo de comportamiento que se genera en la respuesta transitoria. Solicitar al estudiante ingresar dibujar una función que se aproxime al comportamiento esperado del sistema. Solicitar al estudiante la ubicación en el plano S de las raíces de la ecuación característica. La solución de este problema tiene dos estrategias. Una basada en reconocer el valor de zeta y utilizar dicho valor para relacionar el tipo de comportamiento, la otra estrategia está basada en evaluar el discriminante de la ecuación característica.	OB.8. OB.9. OB.10. OB.11	ALTA	ALTA	MEDIA	8
Problema P36: Dadas 4 funciones de transferencia con z y wn conocidos, donde cada función de transferencia tiene un a valor de zeta,wn diferente y cada valor de zeta genera en el sistema una respuesta trasiente que se clasifica dentro de uno de los 4 grupos de comportamientos posibles. Solicitar al estudiante calcular las raíces, ubicar en el plano S las raíces, proveer la expresion que define Y(t) en el tiempo y seleccionar una aproximación a la gráfica de	OB.8. OB.9. OB.10. OB.11	ALTA	ALTA	ALTA	9

respuesta.					
Problema P37: Dada una ubicación para las raíces de la ecuación característica solicitar al estudiante aseverar el rango en el que se encuentra la variable Zeta y la aproximación gráfica de la curva de respuesta.	OB.8. OB.9. OB.10. OB.11	MEDIA	MEDIA	MEDIA	6
Problema P38: Dada una respuesta temporal solicitar al estudiante aseverar la ubicación de las raíces de la ecuación característica y el rango en el que se encuentra la variable zeta.	OB.8. OB.9. OB.10. OB.11	MEDIA	MEDIA	MEDIA	6
Problema P39: Dada una grafica en la que se muestran diferentes curvas de respuestas de varios sistemas de segundo orden a una entrada paso. Solicitar al estudiante etiquetar de forma correcta las gráficas asignando un valor de variable Z a cada grafica y el nombre del comportamiento.	OB.8. OB.9. OB.10. OB.11	MEDIA	MEDIA	ALTA	7
Problema P40: Dado un sistema de segundo orden con Wn definido y K definido. Calcular las diferentes respuestas a la entrada paso si se hace variar la variable Zeta.	OB.8. OB.9. OB.10. OB.11	MEDIA	ALTA	ALTA	8
Problema P41: Dado un sistema de segundo orden y una entrada impulso, varias salidas se muestran en la gráfica. Cada salida corresponde a un valor de zeta específico. Solicitar al estudiante especificar el valor de zeta para cada una de las gráficas. Los valores de zeta posibles se encuentran especificados como parte del problema.	OB.8. OB.9. OB.10. OB.11	MEDIA	MEDIA	ALTA	7
Problema P42: Dado un sistema de segundo orden y una entrada rampa de pendiente m, varias salidas se muestran en una gráfica. Cada salida corresponde al un valor de zeta diferente. Solicitar al estudiante	OB.8. OB.9.	BAJA	MEDIA	ALTA	6

especificar el valor de zeta para cada una de las gráficas. Los valores de zeta posibles se encuentran especificados como parte del problema.	OB.10. OB.11				
Problema P43: Dada una gráfica que muestra la respuesta subamortiguada de un sistema de segundo orden a una entrada paso. Solicitar al estudiante identificar los puntos de la gráfica que corresponden a Tiempo de subida, Tiempo pico, tiempo de establecimiento, error en estado estable y Porcentaje de sobrepico.	OB.8. OB.9. OB.10. OB.11	ALTA	BAJA	MEDIA	6
Problema P44: Dado un sistema de segundo orden que responde a una entrada paso con una respuesta de tipo sobreamortiguada. Solicitar al estudiante Calcular el tiempo de subida.	OB.8. OB.9. OB.10. OB.11	BAJA	BAJA	BAJA	3
Problema P45: Dado un sistema de segundo orden que responde a una entrada paso con una respuesta de tipo sobreamortiguada y cuyo tiempo de subida es conocido y cuyo Wn es conocido. Solicitar al estudiante calcular el valor de la variable Zeta.	OB.8. OB.9. OB.10. OB.11	BAJA	BAJA	BAJA	3
Problema P46: Dado un sistema de segundo orden cuyo valor de zeta se conoce y cuyo valor de Wn es ajustable. Se excita el sistema con una entrada paso, se registra la salida, se modifica Wn y se repite nuevamente el procedimiento. Solicitar al estudiante relacionar correctamente los posibles valores de Wn con las curvas de respuesta mostradas en la gráfica.	OB.8. OB.9. OB.10. OB.11	MEDIA	ALTA	MEDIA	6
Problema P47: Dado un sistema de segundo orden cuyos valores de Zeta y Wn son conocidos. Solicitar al estudiante calcular los valores de Tp, PO, y Tr. Relevancia: Alta, , Goal Structure: Medium.	OB.8. OB.9. OB.10. OB.11	ALTA	MEDIA	MEDIA	7

Problema P48: Dado un sistema de segundo orden cuyos valores de Zeta y Wn son desconocidos. Se conoce la respuesta del sistema a una entrada paso y se conoce que la respuesta es de tipo subamortiguado con parámetros Tp y PO conocidos. Se solicita al estudiante calcular los valores de Wn y Zeta.	OB.8. OB.9. OB.10. OB.11	ALTA	BAJA	MEDIA	6
Descripción	**Objetivos de Aprendizaje relacionados**	**Relevancia**	**Estructura de Objetivos**	**Generalidad**	**Puntaje Final**
Problema P49: Dado un sistema de segundo orden cuyos valores de Zeta y Wn son conocidos. Si se excita el sistema con una entrada paso, se solicita al estudiante calcular el tiempo de asentamiento con un criterio de 5% y con un criterio de 2%.	OB.8. OB.9. OB.10. OB.11	MEDIA	BAJA	ALTA	6
Problema P50: Dado un sistema de segundo orden cuyos valores de Zeta y Wn son desconocidos. Se excita el sistema con una entrada paso, se registran los parámetros de la respuesta Ts y PO. Solicitar al estudiante calcular los valores de Zeta y Wn.	OB.8. OB.9. OB.10. OB.11	ALTA	BAJA	MEDIA	6
Problema P51: Dado un sistema de segundo orden cuyos valores de Zeta y Wn son desconocidos. Se excita el sistema con una entrada paso, se registra la respuesta. Solicitar al estudiante calcular los valores de Zeta y Wn , K , y seleccionar las ecuaciones que hace uso para realizar dicho cálculo.	OB.8. OB.9. OB.10. OB.11	ALTA	ALTA	ALTA	9
Problema P52: Dado una función de transferencia G(S)= K/ s(s+p) que se encuentra en lazo cerrado. Se solicita al estudiante seleccionar la variable K y el parámetro P, con el fin de cumplir con las siguientes especificaciones: La respuesta transitoria a entrada paso debe ser lo más rápida posible. El porcentaje de sobrepico debe ser menor a 5% El tiempo de asentamiento al 2%	OB.8. OB.9. OB.10. OB.11	ALTA	ALTA	ALTA	9

debe ser menor a 4%.					
Problema P53: Se presenta una función de transferencia de tercer orden con función de lazo cerrado cuyo tercer polo esta ubicado en la posición 1/R. Se solicita al estudiante identificar si el tercer polo es un polo que puede ser despreciado o no.	OB.8. OB.9. OB.10. OB.11	ALTA	BAJA	BAJA	5
Problema P54: Se presenta una gráfica en la que se muestran 3 curvas de respuesta. Cada curva corresponde a la respuesta de un sistema de tercer orden con función de lazo cerrado cuyo tercer polo tiene una constante de tiempo diferente. Se solicita al estudiante relacionar las posibles constantes de tiempo con las curvas de forma correcta.	OB.8. OB.9. OB.10. OB.11	ALTA	BAJA	MEDIA	6
Problema P55: Se presenta una gráfica en la que se muestran 3 curvas de respuesta. Cada curva corresponde a la respuesta de un sistema de tercer orden con función de lazo cerrado cuyo cero ubicado diferente en diferente posición. Se solicita al estudiante relacionar las posibles constantes de tiempo con las curvas de forma correcta.	OB.8. OB.9. OB.10. OB.11	ALTA	BAJA	MEDIA	6
Problema P56: Dado un conjunto de 12 diferentes comportamientos que un sistema de segundo orden puede presentar ante una entrada paso dependiendo de la ubicación de sus polos en el plano S. Se solicita al estudiante ubicar los 12 comportamientos en el plano de forma correcta.	OB.8. OB.9. OB.10. OB.11	MEDIA	BAJA	ALTA	6
Problema P57: Dado un diagrama de bloques de lazo abierto solicitar al estudiante identificar el punto donde se puede medir el error en estado estable y proveer una expresión que permita calcularlo	OB.12 OB.13 OB.14 OB.15	MEDIA	BAJA	ALTA	6
Problema P58: Dado un diagrama de bloques de lazo cerrado con realimentación unitaria solicitar al estudiante identificar el punto donde se puede medir el error en estado	OB.12 OB.13	MEDIA	MEDIA	MEDIA	6

estable y proveer una expresión que permita calcular el error.	OB.14 OB.15				
Problema P59: Dado un diagrama de bloques con realimentación no unitaria solicitar al estudiante identificar el punto donde se puede medir el error en estado estable y proveer una expresión que permita calcular el error.	OB.12 OB.13 OB.14 OB.15	MEDIA	ALTA	MEDIA	7
Problema P60: Dado un diagrama de bloques con realimentación y pre-alimentación no unitarias solicitar al estudiante identificar el punto donde se puede medir el error en estado estable y proveer una expresión que permita calcular el error.	OB.12 OB.13 OB.14 OB.15	ALTA	ALTA	ALTA	9
Problema P61: Dada una expresión para la variable de error definida en el dominio de Laplace, solicitar al estudiante aplicar el teorema del valor final y calcular en valor del error en estado estable para una entrada definida.	OB.12 OB.13 OB.14 OB.15	MEDIA	ALTA	MEDIA	7
Problema P62: Dada una expresión para la variable del error en estado estable, definida en términos de las constantes o parámetros que describen los diferentes bloques del sistema de control, solicitar al estudiante que indique cual es el efecto que tiene la variación de ciertas constantes en el valor del error en estado estable.	OB.12 OB.13 OB.14 OB.15	MEDIA	BAJA	BAJA	4
Problema P63: Dada una expresión para la variable de error en estado estable, que esta definida en términos de algunos bloques desconocidos en el sistema y en términos de algunos bloques conocidos, solicitar al estudiante determinar el efecto de la variación de los parámetros conocidos en el valor del error de estado estable.	OB.12 OB.13 OB.14 OB.15	MEDIA	MEDIA	ALTA	7
Problema P64: Dadas las tres constantes de error utilitarias para el estudio del error en estado estable y dadas las tres funciones para las que	OB.12	MEDIA	BAJA	BAJA	4

estas constantes se han definido, solicitar al estudiante realizar la relación de cada una de las constantes con la entrada respectiva para la cual se realiza su definición.	OB.13 OB.14 OB.15				
Problema P65: Se provee al estudiante con definiciones para las funciones de transferencia que representan el bloque de control y el bloque del sistema a controlar en un sistema de control con realimentación unitaria. Se solicita al estudiante calcular el error en estado estable para las tres señales de entradas estándar.	OB.12 OB.13 OB.14 OB.15	MEDIA	ALTA	MEDIA	7
Problema P66: Se presenta al estudiante una función de transferencia de lazo cerrado que corresponde a un sistema de tipo N. N puede ser un numero entre 0, 1, 2, 3, 4 o el literal N. Se solicita al estudiante que reconozca cual es tipo de sistema presentado.	OB.12 OB.13 OB.14 OB.15	MEDIA	BAJA	BAJA	4
Problema P67: Se presenta al estudiante un sistema de bloques en donde el bloque de control está definido, así como el sistema a controlar. El lazo de realimentación es unitario. Las funciones de transferencia de los bloques pueden tener un tipo diferente N y M respectivamente. N y M pueden ser números entre 0, 1, 2, 3, 4 o los literales N y M. Se solicita al estudiante que reconozca cual es tipo de sistema presentado.	OB.12 OB.13 OB.14 OB.15	MEDIA	MEDIA	MEDIA	6
Problema P68: Dado un sistema de control con realimentación unitaria negativa y cuya función de transferencia del camino principal es conocida. Solicitar al estudiante identificar el tipo de sistema y proveer el valor teórico para el error en estado estable el cual puede ser (cero, constante o infinito) para una entrada determinada.	OB.12 OB.13 OB.14 OB.15	ALTA	ALTA	ALTA	9
Problema P69: Dado un sistema de control cuya función de transferencia del camino principal es conocida pero que resulta ser no estable. Solicitar al estudiante identificar el tipo de sistema y proveer el valor	OB.12 OB.13	MEDIA	BAJA	MEDI	5

teórico para el error en estado estable el cual puede ser (cero, constante o infinito) para una entrada determinada	OB.14 OB.15				

Anexo B: Tabla comparativa de aplicabilidad de Herramientas de Autor

		CTAT Example Tracing	CTAT Model Tracing	ASPIRE	Preferencia
Funcionalidades para creación de interfaz gráfica	Ventajas	Dos opciones de entorno de ejecución: Java y FLASH. Los componentes son medianamente configurables (borde, fuente, tamaño, opciones de despliegue, etc). Conjunto de componentes de propósito general completo.		Para problemas simples la interfaz generada automáticamente es suficiente. Soporta Applets con los cuales se puede personalizar la interfaz gráfica de acuerdo a las necesidades. Las interfaces generadas automáticamente están basadas en HTML, por lo cual son muy rápidas.	CTAT, en tiempo de desarrollo porque los componentes ya están listos.
	Desventajas	ntes no se pueden extender fácilmente. Los componentes no se pueden reutilizar, solamente copiar y pegar.		La interfaz gráfica generada automáticamente luce bastante antigua. Para especificaciones de la ontología que son complejas, la interfaz no se no se genera de acuerdo a las expectativas. Cuando se utilizan Applets, no hay soporte para resalta los campos con errores. Cuando se utilizan Applets, se debe crear un	

				contenido XML que represente la solución y no existe una librería para generar dicho contenido esto último supone un gran esfuerzo de desarrollo.	
Funcionalidades para crear la representación que define en el formalismo teórico.	Ventajas	Permite crear los grafos de comportamiento a partir de las demostraciones y tiene varias opciones para manipularlos y hacer fácil el trabajo con estos grafos	Con el editor de JESS y de la memoria de trabajo, se permite crear las representaciones y reglas IF-THEN propias del formalismo.	El generador de restricciones crea las restricciones a partir de las soluciones proporcionadas al sistema.	CTAT. Porque sus funcionalidades para crear los grafos son más avanzadas que la simple generación de restricciones.
	Desventajas	No	Las representaciones deben crearse en un lenguajes propietario que extiende CLIPS, que es un lenguaje conocido pero no tan popular como los de propósito general. Esto implica un gran esfuerzo en términos de aprendizaje y desarrollo para el autor.	En muchas ocasiones la inducción de las restricciones es incorrecta y el autor se ve obligado a editar manualmente las restricciones, no para mejorarlas, sino para corregirlas. El lenguaje propietario es algo complicado y la documentación no está disponible públicamente. Se obtuvo por petición directa al investigador líder.	
Funcionalidades para generalizar y hacer mas avanzado la representación del formalismo	Ventajas	Cuenta con funciones avanzadas para generalizar basadas en la ejecución de algoritmos escritos en Java lo que lo hacen muy poderoso y conveniente.	Soporta la creación de funciones en JESS y la invocación de dichas funciones.	Soporta la creación de funciones de dominio usando lenguaje LISP.	CTAT. Porque las funciones en Java se pueden escribir con menos esfuerzo en capacitación y porque dicho lenguaje tiene mayor soporte para hacer integración con aplicaciones externas.
	Desventajas	No	La creación de funciones debe realizarse en el lenguaje propietario JESS en el cual es reducido el número de recursos disponibles (librerías 3rd party)	La creación de funciones debe realizarse en el lenguaje LISP el cual tiene un conjunto de recursos disponibles importante, pero que no tiene una popularidad y una comunidad tan grande como otros lenguajes modernos.	
Modificación de la Interfaz gráfica a partir del	Ventajas	Hay soporte nativo y documento para esta funcionalidad.		Para interfaces basadas en applet se soportan parcialmente para los estados en que se reconocen errores específicamente.	CTAT. Porque aspire no lo soporta sin hacer desarrollo.
	Desventaja	Para interfaces implementadas en lenguaje Java, no existe		Para las interfaces HTML no hay soporte pero se	

reconocimiento de ciertos estados del problema.	s	la posibilidad de tomar acciones de interfaz (ocultar, mostrar, desplazar) sobre grupos de componentes. La capacidad existe solamente para componentes discretos.		puede implementar el soporte en JS. Sin embargo la implementación no estaría basada en código o documentación oficial.	
Herramientas de Depuración	Ventajas	El grabador de árbol y el “edit” de las acciones del estudiante también permiten depurar errores	Si. Hay bastante soporte y hay alguna documentación disponible.	Si. Herramienta de depuración de restricciones es suficientemente buena para ayudar en la resolución de creación de restricciones.	Sin preferencia. Puesto que las tres herramientas de depuración cumplen con sus objetivos.
	Desventajas	No	No	No	
Mecanismos de Extensiblidad	Ventajas	Solo a nivel de generalización del árbol.	Solo a nivel delas reglas de producción.	A nivel de componentes gráficos se pueden realizar ciertos cambios, pero no son documentados ni tampoco oficiales.	ASPIRE. Porque la interfaz gráfica del tutor esta basada en CSS, HTML y JS. La interfaz es más rápida en tiempo de carga y se ajusta al nuevo modelo de web.
	Desventajas	Los componentes gráficos no se pueden extender , tampoco el algoritmo principal del funcionamiento del tutor	Los componentes gráficos no se pueden extender , tampoco el algoritmo principal del funcionamiento del tutor	El algoritmo principal de funcionamiento tiene características de extensibilidad.	
Documentación Disponible	Ventajas	Considerable, está formada por guías de usuario, tutoriales de desarrollo, artículos científicos, grupos de usuarios en foros, y eventos anuales para capacitación.	Considerable, está formada por guías de usuario, tutoriales de desarrollo, artículos científicos, grupos de usuarios en foros, y eventos anuales para capacitación.		CTAT.
	Desventajas	No existe documentación avanzada para desarrolladores debido a que no hay mecanismos de extensibilidad disponibles	La documentación especifica para JESS es reducida. JESS tiene restricciones de licenciamiento.	Bastante reducida, se limita a artículos científicos publicados y a unos pocos documentos que son proporcionados por el desarrollador.	
Disponibilida	Ventajas	No	No	Por estar el front-end implementado en tecnologías	ASPIRE.

d de Código fuente.				web CSS, HTML y Javascript. El código fuente del cliente puede ser accedido, estudiado y extendido con algún esfuerzo avanzado en desarrollo.	Porque al menos el front-end del sistema puede ser estudiado y modificado gracias a las características de un lenguaje de tipado débil como Javascript y a la precedencia de aplicación de reglas de CSS.
	Desventajas	La licencia prohíbe expresamente el acceso al código por medio de técnicas de ingeniería reversa.	La licencia prohíbe expresamente el acceso al código por medio de técnicas de ingeniería reversa.	No existe para el back-end del sistema.	
Modularidad de la Solución	Ventajas	Si los componentes gráficos se agrupan en paneles que no son instancias de clases personalizadas, dichos paneles pueden ser reutilizados	Las definiciones de la memoria de trabajo pueden reutilizarse fácilmente, igualmente la definición de las reglas y funciones en lenguaje JESS.	Las restricciones generadas y editadas pueden ser descargadas y reutilizadas nuevamente. Dependiendo de su grado de abstracción pueden reutilizarse entre problemas.	ASPIRE. Si las restricciones se definen de manera adecuada pueden reutilizarse y es más fácil trabajar con las restricciones que con las reglas JESS y los archivos XML que definen los grafos pues son poco legibles. Adicionalmente todo el Javascript y CSS que se incluya para mejorar las interfaces en aspire puede reutilizarse fácilmente.
	Desventajas	Agrupar en paneles es un mecanismo de reutilización poco avanzado y a largo plazo y con bastante uso resulta de difícil mantenimiento. Un mecanismo de reutilización mas avanzado no existe. Los grafos de comportamiento no se puede incluir unos en otros por lo tanto no son reutilizables.	Lograr el conocimiento para realizar estas tareas tiene un alto esfuerzo que es mucho mayor que para un lenguaje de propósito general.	Reutilizar las restricciones supone un alto grado de conocimiento del lenguaje propietario, sin embargo se estima que el esfuerzo es menor que el que se debe realizar para lograr el mismo objetivo en lenguajes como JESS.	

Anexo C: Tabla de requerimientos de modelamiento de conocimiento y ejemplos de modelamiento.

Requerimientos de modelamiento de conocimiento	***Ejemplos de Preguntas***	***Ejemplos de Respuesta***
Diagramas de bloque:		
Que el tutor pueda Identificar las dependencias y causalidades entre señales	De cuales señales depende la señal de error E(S)?	Depende de la Entrada X(s) y la salida Y(s). Aunque la salida Depende de U(s) y de E(s). Si la salida del sensor es alta entonces la salida del transmisor sera alta tambien, por lo tanto la entrada al controlador será una señal de gran amplitud.
Que el tutor pueda identificar los efectos de los bloques sobre las señales	Que pasa con la señal U(s) al ingresar al bloque G(s)	Hay una amplificación de la señal debido a la ganancia, la señal de salida del bloque tiene una respuesta de segundo orden. Las unidades se transforman de voltios a temperatura.
Que el tutor pueda calcular expresiones para una determinada señal en terminos de otras señales y bloques.	Dado el diagrama calcule la expresión para el error E(S)?	El error se puede expresar como E(S)=Y(S)-X(S)

Que el tutor pueda asociar un sentido físico a las señales y bloques	Que subsitema representa el bloque Gc(S)? Que señal en especifico es Y(S) dentro del sistema? Cuales son las unidades de la señal U(S)? En el sistema real donde se puede hacer mediciones de la señal de error?	Representa el controlador, que en el sistema físico resulta ser el microcontrolador. Y(S) es la salida de la planta ampliada y resulta ser la salida del sensor y no de la planta. Las unidades de U(S) son voltios. El error se puede medir en el restador 1 ó El error es invisible y no puede ser medido físicamente sino que debe ser calculado a partir de la medicion de otras señales físicas.
Que el tutuor pueda establecer equivalencias ó transformaciones entre diagramas con la ayuda de expresiones matemáticas	Calcule la expresión para la funcion de transferencia de lazo cerrado y actualice el diagrama de acerudo a la nueva representación	La expresión para la función de lazo cerrado es Gcl(s)=xxxxxx el nuevo diagrama realizando el reemplazo de sería yyy
Expresiones matemáticas		
Que el tutor pueda realizar y/o comprender operaciones tipicas algebraicas como expansión, factorización, reemplazo, reducción, simplificación, despeje de variables, multiplicación y división por el mismo factor,	Factorize Y(s) y luego despeje Y(S)	si Y(S)(a+b+c)=X(s) entonces Y(S)=X(S)/(a+b+c)
Que el tutor pueda realizar el Cálculo de Transformadas directas e inversas de Laplace.	Calcule la transformada de Laplace de la señal de entrada?	Laplace(step(t)) = 1/S
Que el tutor pueda realizar el Cálculo de Fracciones Parciales	Calcule la exapnsión en fraciones parciales de Y(S)	Y(S)=(1/S)*(a/(bs+c)) entonces Y(S)=(a/c)/s - (a/c)/(s+(c/b))
Que el tutor pueda realizar el Cálculo de Límites, Integrales, Derivadas, Productorias, Sumatorias.	Evalue el límite cuando S tiende a cero de la expresión s*Glc(s)*X(s)	El resultado es K/2 lo que indica que el error en estado estable será igual a la mitad del parámetro K
Que el tutor pueda realizar análisis del efecto de la variación de variables independientes sobre variables dependientes	Si se aumenta la ganancia proporcional del controlador que ocurre con el sobrepico? Si el controlador tiene la forma mostrada, cual sería el error de aceleración para una plata de segundo orden?	Debido a que la relación entre el sobrepico y la ganancia estática es directa, el sobrepico aumentaria. Reemplazando Gc en la expresión de Ea, y evaluando el limite se tiene que el error en estado estable sería cero.
Manipulación de gráficas		

Que el tutor pueda relizar y/o verificar la Generación de gráficas en el dominio del tiempo y de la frecuencia a partir de la expresión.	Si la señal de entrada al sistema es la señal paso de amplitud 1 calcule y gráfique la salida.	Tras los cálculos y(t) resulta ser sin(wt), de manera que la gráfica es la siguente.
Que el tutor pueda reconocer expresiones matemáticas a partir de gráficas parametrizadas en el dominio del tiempo y la frecuencia.	La señal que se muestra en la figura corresponde a la Entrada del sistema.	La señal de entrada es una señal rampa con pendiente 4. Por lo tanto x(t)=4t, para t >0
Que el tutor pueda Identificar medidas y de parámetros como caracteristicas de las gráficas: rangos (medidas de tiempo y amplitud), pendientes, forma de la gráfica (exponencial, senoidal, subamortiguado)	La señal que se muestra en la gráfica es la salida del controlador del sistema, Identifique el tipo de controlador y sus parámetros.	la señal mostrada corresponde a la señal de salida de un controlador PI, de acuerdo a la pendiente se puede identificar que la constante Kc es igual a 5 y la contante Ti es igual 3. De esta forma la expresion para dicho controlador sería Gc(s)=5*(1 + 1/3*s)
Conceptualización y abstracción de contenidos.		
Que el tutor pueda mantener una relacion de conceptos por medio del uso de memoria.	Cual es la forma general para representar un sistema de primer orden Si el coeficiente de amortiguamiento es mayor a 1, como se categoriza el comportamiento del sistema? Cual es la expresión que representa un control derivativo? Cuales son los parametros de configuración de un controlador PID? Que ocurre si se aumenta considerablemente la ganancia proporcional del controlador?	G(S)_K/tau*S+1 El comportamiento sería sobre amortiguado. Gc(S)=Kd*s Se afecta negativamente la respuesta temporal, se aumenta el sobrepico, y el sistema puede llegar a oscilar o ser completamente inestable.

Que el tutor pueda memorizar procedimientos	Indique los pasos para calcular la respuesta del sistema a una entrada definida a trozos, teniendo en cuenta que el sistema se modela por la funcion de transferencia mostrada Indique los pasos para resolver un problema de diseño de control	1) Aplicar transformada de laplace a la entrada 2) calcular la salida en el dominio de laplace. 3) Calcular la transformada inversa de laplace. 1) Escoger un modelo de planta, 2) Definir un desempeño deseado 3) Imponer una estructura de control o un comportamiento deseado de lazo cerrad 4) realizar el cálculo de los parámetros de control necesario 5) Implementar el control 6) Validar el comportamiento respecto al deseado 7) iterar desde el paso 4 para ajustar

Anexo D: Tabla de requerimientos de modelamiento de conocimiento y ejemplos de modelamiento II.

Requerimientos de modelamiento de conocimiento	***Ejemplos de Preguntas***	***Ejemplos de Respuesta***	***Hay una interfaz gráfica posible que genere una acción tal que se pueda modelar de forma generalizada el conocimiento requerido ? Si lo hay cual?***	***Hay un conjunto de reglas de producción mediante los cuales se pueda modelar de forma generalizada el conocimiento requerido? Si lo hay cual?***	***Hay un conjunto de restricciones mediante los cuales se pueda modelar de forma generalizada el conocimiento requerido? Si lo hay cual?***	
Diagramas de bloque:						
Que el tutor pueda Identificar las dependencias y causalidades entre señales	De cuales señales depende la señal de error E(S)?	Depende de la Entrada X(s) y la salida Y(s). Aunque la salida Depende de U(s) y de E(s). Si la salida del sensor es alta entonces la salida del transmisor será alta también, por lo tanto la entrada al controlador será una señal de gran amplitud.	Si. Por ejemplo se puede tener un selector múltiple de señales y bloques. Los datos iniciales del selector pueden provenir de una definición del diagrama. Una vez seleccionada una opción, por medio de una función procedimental Java se puede determinar si la selección es correcta a partir del análisis de la definición del diagrama. Para el segundo caso se puede modelar teniendo pasos intermedios y selectores de alto bajo. Por ejemplo: Salida del sensor (alta,baja), Salida del transmisor(alta, baja), Entrada al Controlador (alta, baja). Y se realizan demonstraciones especificas de las opciones correctas.	Si. La estructura del diagrama se puede codificar en el conjunto de reglas. Por ejemplo: Ej1: IF isInput($signal_1,$system) and isOutpu($signal_2,$system) THEN $signal2_depends_on = $signal_1 Ej2: IF dependsOn($signal1,$signal2) THEN $signal2_depends_on = $signal_1 Ej3: IF $signal1.system == $signal2.system AND $signal1.isOutput AND $signal2.isInput THEN $signal1.dependsOn=$signal2 o IF $salidaSensor==ALTA THEN $salidaTransmisor=ALTA, IF $salidaTransmisor=ALTA THEN $entradaControlador=ALTA	Cr: IF $signal1 is defined in terms of $signal2 THEN IT SHOULD BE TRUE THAT Cs: dependsOn($signal1,$signal2)	El modelamiento por restricciones es preferible para este requerimiento
Que el tutor pueda identificar los efectos de los bloques sobre las señales	Que pasa con la señal U(s) al ingresar al bloque G(s)	Hay una amplificación de la señal debido a la ganancia, la señal de salida del bloque tiene una respuesta de segundo orden. Las unidades se transforman de voltios a temperatura.	S: valorSalida, A:updateText, V: $valorEntrada*$ganancia En términos cualitativos se definen las opciones de respuesta y se demuestran.	IF $amplitudEntrada is not null and $ganancia is not null THEN $amplitudSalida = $amplitudEntrada*ganancia.	Cr: IF $amplitudSalida is not NULL THEN IT SHOULD BE TRUE THAT Cs: $amplitudSalida=$amplitudEntrada*$ganancia	No hay una preferencia
Que el tutor pueda calcular expresiones para una determinada señal en términos de otras señales y bloques.	Dado el diagrama calcule la expresión para el error E(S)?	El error se puede expresar como E(S)=Y(S)-X(S)	S:expresion A:updatetext I: getExpressionForSignal(E,Diagram)	IF $signal is the output of a SUM node THEN $signal =SUM (node.input) IF $signal is the output of a block THEN output=block.input*block.transfer_function	Cr: IF $signal is defined as a SUM of $signals THEN IT SHOULD BE TRUE THAT Cs: $signal is the output of a SUM node	El modelamiento basado en reglas resulta mas apropiado para este caso
Que el tutor pueda asociar un sentido físico a las señales y bloques	Que subsistema representa el bloque Gc(S)? Que señal en especifico es Y(S) dentro del sistema? Cuales son las unidades de la señal U(S)? En el sistema real donde se puede hacer mediciones de la señal de error?	Representa el controlador, que en el sistema físico resulta ser el micro controlador. Y(S) es la salida de la planta ampliada y resulta ser la salida del sensor y no de la planta. Las unidades de U(S) son voltios. El error se puede medir en el restador 1 ó El error es invisible y no puede ser medido físicamente sino que debe ser calculado a partir de la medición de otras señales físicas.	Se requiere un mapa de relacionamiento de conceptos explicito. Esto es lo mismo que demostrar las respuestas sin hacer ninguna generalización. S:respesta A:selectOption I:Microcontrolador S.repsuesta A:selectOption I:No Medible, Señal Invisible	IF $tipoSistema==PosicionSatelite THEN $controladorTipo=Microcontrolador IF $tipoSistema==feedforwad AND numSenalesPuntoSum() > 2 THEN $errorTipo: No medible	Cr: IF $errorTipo==NoMedible THEN IT SHOULD BE TRUE THAT Cs: $tipoSistema==feedforward AND numSenalesPuntoSuma() > 2	Ningún mecanismo resulta apropiado para este tipo de modelamiento.
Que el tutor pueda establecer equivalencias ó transformaciones entre diagramas con la ayuda de expresiones matemáticas	Calcule la expresión para la función de transferencia de lazo cerrado y actualice el diagrama de acuerdo a la nueva representación	La expresión para la función de lazo cerrado es Gcl(s)=xxxxxx el nuevo diagrama realizando el reemplazo de sería yyy	Se puede calcular usando algebra de bloques toda transformación: S:expresion A:updatetext I:areDiagramsEquivalent(DiagramaEnunciado,DiagramaEstudiante) AND getExpressionForBlock(DiagramaEstudiante)== Thisfield.value	IF areDiagramsEquivalent(DiagramaEnunciado,DiagramaEstudiante) THEN $expresionEquivalente = getExpressionForBlock(DiagramaEstudiante)	Cr: IF $expresionEquivalente IS NOT NULL AND DiagramaEnunciado IS NOT NULL AND DiagramaEstudiante IS NOT NULL THEN IT SHOULD BE TRUE THAT Cs: $expresionEquivalente == getExpressionForBlock(DiagramaEstudiante)	No hay preferencia.
Expresiones matemáticas						

Que el tutor pueda realizar y/o comprender operaciones típicas algebraicas como expansión, factorización, reemplazo, reducción, simplificación, despeje de variables, multiplicación y división por el mismo factor,	Factorize Y(s) y luego despeje Y(S)	si Y(S)(a+b+c)=X(s) entonces Y(S)=X(S)/(a+b+c)	S:expresionActual A:updateText I: factor(expresionAnterior,'x') S:expresionActual A:updateText I: Laplace(expresionAnterior,'x') S:expresionActual A:updateText I: lim(expresionAnterior,'S',0)	IF $objetivo="calcularX(S)" AND $x_tiempo is not null THEN $X_laplace=Lapalce($X_tiempo) IF $objetivo="calcularEss" AND $Glc is not null THEN $Ess=lim(s*Glc/(1+Glc),'s',0)	Cr: IF $expresionEss IS NOT NULL AND Glc is NOT NULL THEN IT SHOULD BE TRUE THAT Cs: $expresionEss == lim(s*Glc/(1+Glc),'s',0)	Utilizando funciones de calculo simbólico se pueden demostrar acciones de sean suficientemente generales. Para los tres paradigmas. En este caso se preferiría example-tracing por requerir menos conocimientos: No lenguajes propietarios, depuración mas sencilla.
Que el tutor pueda realizar el Calculo de Transformadas directas e inversas de Laplace.	Calcule la transformada de Laplace de la señal de entrada?	Laplace(step(t)) = 1/S				
Que el tutor pueda realizar el Calculo de Fracciones Parciales	Calcule la expansión en fraciones parciales de Y(S)	Y(S)=(1/S)*(a/(bs+c)) entonces Y(S)=(a/c)/s - (a/c)/(s+(c/b))				
Que el tutor pueda realizar el Calculo de Límites, Integrales, Derivadas, Productorias, Sumatorias.	Evalúe el límite cuando S tiende a cero de la expresión s*Glc(s)*X(s)	El resultado es K/2 lo que indica que el error en estado estable será igual a la mitad del parámetro K				
Que el tutor pueda realizar análisis del efecto de la variación de variables independientes sobre variables dependientes	Si se aumenta la ganancia proporcional del controlador que ocurre con el sobre pico? Si el controlador tiene la forma mostrada, cual sería el error de aceleración para una plata de segundo orden?	Debido a que la relación entre el sobre pico y la ganancia estática es directa, el sobre pico aumentaría. Reemplazando Gc en la expresión de Ea, y evaluando el limite se tiene que el error en estado estable sería cero.	S:respuestaEstudianteRelacion A:SelectOption I: getRelationship($previousExpression, $varSubject)	IF HAS_DENOMINATOR($expression) AND IS_DEFINED($varSubject,$expression) AND BELONGS_TO_DEN($varSubject,$expression) THEN $RELATIONSHIP="Inverse" IF HAS_NUMERATOR($expression) AND IS_DEFINED($varSubject,$expression) AND BELONGS_TO_NUM($varSubject,$expression) THEN $RELATIONSHIP="Direct"	Cr: I$RELATIONSHIP="Direct" AND $expression IS NOT NULL AND $varSubject IS NOT NULL THEN IT SHOULD BE TRUE THAT Cs: HAS_NUMERATOR($expression) AND IS_DEFINED($varSubject,$expression) AND BELONGS_TO_NUM($varSubject,$expression)	No hay una preferencia
Manipulación de gráficas						
Que el tutor pueda realizar y/o verificar la Generación de gráficas en el dominio del tiempo y de la frecuencia a partir de la expresión.	Si la señal de entrada al sistema es la señal paso de amplitud 1 calcule y grafique la salida.	Tras los cálculos y(t) resulta ser sin(wt), de manera que la gráfica es la siguiente.		Debido a que la operación de graficado en el computador requiere de una expresión, la validación del tutor se realiza respecto a la expresión y no respecto ala gráfica en sí		No hay Preferencia
Que el tutor pueda reconocer expresiones matemáticas a partir de gráficas parametrizadas en el dominio del tiempo y la frecuencia.	La señal que se muestra en la figura corresponde a la Entrada del sistema.	La señal de entrada es una señal rampa con pendiente 4. Por lo tanto x(t)=4t, para t >0		Si la figura que se muestra ha sido generada por el tutor, el tutor conoce los parámetros y la expresión para generar la gráfica. Realizar reconocimiento de expresiones a partir de imágenes generadas por terceros o a partir de datos provenientes de fuentes terceras no hace parte del alcance de este análisis		No hay Preferencia
Que el tutor pueda Identificar medidas y de parámetros como características de las gráficas: rangos (medidas de tiempo y amplitud), pendientes, forma de la gráfica (exponencial, senoidal, subamortiguado)	La señal que se muestra en la gráfica es la salida del controlador del sistema, Identifique el tipo de controlador y sus parámetros.	la señal mostrada corresponde a la señal de salida de un controlador PI, de acuerdo a la pendiente se puede identificar que la constante Kc es igual a 5 y la contante Ti es igual 3. De esta forma la expresión para dicho controlador sería Gc(s)=5*(1 + 1/3*s)	S:ParametroKc A:updateText I:$parametroKC S:TipoComportamiento A:selectOption I:getBehaviorTagFromZeta($zeta)	IF $parametroKC is not null AND $parametro Ti is not null THEN $expression= Gc(s)=$parametroKC*(1 + 1/$parametroTi*s) IF $parametroZeta is not null AND $parametroZeta> 0 AND $parametroZeta < 1 THEN EtipoComportamiento=Subamortiguado	Cr: $expression is not NULL and $parameterKc is not null and $parameterTi is not NULL IS NOT NULL THEN IT SHOULD BE TRUE THAT Cs: $expression==$parametroKC*(1 + 1/$parametroTi*s) Cr: $tipoComporatamient is not NULL and $ipoComportamiento==Subamortiguado IS NOT NULL THEN IT SHOULD BE TRUE THAT Cs: $parametroZeta < 1 and $parametroZeta > 0	No hay Preferencia
Conceptualización y abstracción de contenidos.						
Que el tutor pueda mantener una relación de conceptos por medio del uso de memoria.	Cual es la forma general para representar un sistema de primer orden Si el coeficiente de amortiguamiento es mayor a 1, como se categoriza el comportamiento del sistema? Cual es la expresión que representa un control	G(S)_K/tau*S+1 El comportamiento sería sobre amortiguado. Gc(S)=Kd*s Se afecta negativamente la respuesta temporal, se aumenta el sobre pico, y el sistema puede llegar a oscilar o ser completamente inestable.	S:valorRespuesta A:selectOption I: G(S)_K/tau*S+1	IF $objetivo="recordarExpresionCanonicaPrimerOrden" THEN $expresion=G(S)=K/Tau*s+1	Cr: IF $expression IS NOT NULL AND $step==2 THEN IT SHOULD BE TRUE THAT Cs: $expresion == G(S)=K/Tau*s+1	Hay preferencia por la versión example tracing debido a su simplicidad

	derivativo? Cuales son los parámetros de configuración de un controlador PID? Que ocurre si se aumenta considerablemente la ganancia proporcional del controlador?					
Que el tutor pueda memorizar procedimientos	Indique los pasos para calcular la respuesta del sistema a una entrada definida a trozos, teniendo en cuenta que el sistema se modela por la función de transferencia mostrada Indique los pasos para resolver un problema de diseño de control	1) Aplicar transformada de Laplace a la entrada 2) calcular la salida en el dominio de Laplace. 3) Calcular la transformada inversa de Laplace. 1) Escoger un modelo de planta, 2) Definir un desempeño deseado 3) Imponer una estructura de control o un comportamiento deseado de lazo cerrad 4) realizar el calculo de los parámetros de control necesario 5) Implementar el control 6) Validar el comportamiento respecto al deseado 7) iterar desde el paso 4 para ajustar	Se modela como una secuencia de demostraciones S:expresionXs A:updateText I: Laplace($expressionXt,'t'); S:expresionYs A:updateText I: Simplify($expressionXt*$expressionGs)	Se modela como estructura de objetivos IF objetivo=="diseñar" validacion==false THEN objetivo="validar" IF objetivo=="validar" implementacion==false THEN objetivo="implementar" IF objetivo=="implementar" parametrosCalculados==false THEN objetivo="calcularParametros"	Cr: IF $step=1 AND $plantModel is not NULL AND $performanceSpecBehavior=="Sumamortiguado" THEN IT SHOULD BE TRUE THAT Cs: $plantModelOrder = "2"	Hay preferencia por el modelamiento basado en reglas para las estructuras de objetivos. Sin embargo para los procesos de diseños basados en especificaciones de rendimiento se prefiere el modelamiento basado en restricciones.

Anexo E: Fundamentos de control automático

Tratamiento matemático de sistemas LTI usando transformada de Laplace

Los sistemas LTI (Linear Time Invariant), son sistemas dinámicos lineales invariantes en el tiempo que suelen ser modelados mediante el uso de ecuaciones diferenciales lineales de coeficientes constantes. Para un sistema LTI, con una entrada y una salida, como el que se muestra en la figura, el modelo de su dinámica puede representarse mediante una ecuación diferencial como la que se presenta en la ecuación (1).

$$a_n \frac{d^n y(t)}{dt} + a_{n-1} \frac{d^{n-1} y(t)}{dt} + \cdots + a_0 y(t) = bx(t) \qquad (1)$$

Si *x(t)*tiene un valor conocido, es decir se conoce la entrada del sistema, la resolución de la ecuación diferencial lleva al conocimiento de la variable de salida *y(t)*. Resolver la ecuación diferencial resulta entonces en lograr el conocimiento del comportamiento que tendrá la salida del sistema para una determinada señal entrada. Usualmente la solución de este tipo de ecuaciones diferenciales, se realiza directamente en el dominio original, que para el caso de los sistemas de control resulta ser el dominio del

tiempo. Sin embargo, cuando se aplica la transformada de Laplace, se realiza una transformación a una variable compleja y la ecuación diferencial puede resolverse por medio de unos procedimientos que principalmente son de carácter algebraico. La transformada de Laplace resulta ser un mecanismo bastante elegante para la resolución rápida y esquemática de ecuaciones diferenciales lineales con coeficientes constantes.

La transformada de Laplace es una transformación integral que mapea una función definida en el dominio del tiempo, por ejemplo *f(t)*, a una función *F(S)* definida en el dominio de una variable compleja que normalmente se denota con la letra *S.* Este mapeo se realiza aplicando la integral de Laplace de la función original según la siguiente expresión:

$$F(S) = \int_0^{\infty} f(t) \cdot e^{s \cdot t}\, dt \quad (2)$$

La variable S es una variable compleja definida como $S = \sigma + j \cdot \omega$. Durante el tratamiento de sistemas control, las funciones originales se encuentran definidas en el dominio del tiempo. Una vez realizada la transformación y debido a que la variable S contiene en su definición la frecuencia ω, la función *F(S)* se conoce a menudo como una función en el dominio de la frecuencia.

Para resolver la ecuación diferencial se lleva a cabo un proceso que generalmente consta de los siguientes cuatro pasos:

1. Transformación de la ecuación diferencial al dominio de Laplace.
2. Resolver la variable de salida en el dominio de Laplace
3. Expresar la variable de salida usando fracciones simples mediante la expansión en fracciones parciales.
4. Realizar la transformación inversa de Laplace para volver al dominio original.

La transformación de la ecuación diferencial al dominio de Laplace se realiza aplicando varias propiedades de la transformada de Laplace entre ellas las mas importantes son: Propiedad de linealidad, Propiedad de Diferenciación y propiedad de corrimiento en el tiempo. El siguiente paso, involucra la resolver o despejar la variable que represente la salida del sistema en términos de la variable que representa la entrada del sistema. Esto se realiza utilizando factorización y los procedimientos típicos de algebra para resolver variables en ecuaciones. Una vez resuelta la variable

de salida se debe proceder a realizar la transformación inversa de Laplace, sin embargo, este proceso no presenta la misma facilidad que los dos anteriores. El cálculo de la transformada inversa se realiza también por medio de una transformación integral dada por la siguiente expresión:

$$\mathcal{L}^{-1}\{F(S)\} = f(t) = \frac{1}{2\pi j}\int_{c-j\infty}^{c+j\infty} F(S)\cdot e^{s\cdot t}\, ds \quad t > 0 \quad (3)$$

Debido a la dificultad que presenta la evaluación directa de esta integral, un paso intermedio se ejecuta con el objetivo de hacer más simple el cálculo de la transformada inversa. Debido a que la transformada de Laplace, dadas las restricciones con las que se aplica en el dominio del control, es uno a uno y debido a que para una gran mayoría de funciones que son elementales e importantes en el análisis de sistemas de control, la transformada directa es conocida y dicho conjunto de transformación están recopiladas en tablas como la que se presenta en el apéndice X, el cálculo de la transformación inversa se realiza a través de la descomposición de la función *F(S)* en una suma de funciones más simples como se muestra en la siguiente expresión y luego para cada una de las funciones simples (gracias a la propiedad de superposición de la transformada inversa), se calcula la transformación inversa por medio de un mapeo que puede ser realizado acudiendo a la tabla de transformadas.

$$F(S) = F_1(s) + F_2(s) + \cdots + F_n(s)$$

$$\mathcal{L}^{-1}\{F(S)\} = \mathcal{L}^{-1}\{F_1(s)\} + \mathcal{L}^{-1}\{F_2(s)\} + \cdots + \mathcal{L}^{-1}\{F_n(s)\}$$

Para la mayoría de los procesos de análisis en el dominio del control, la función F(S) es una función racional, una división de dos polinomios en la variable S, donde el grado del polinomio numerador es menor que el grado del polinomio denominador. Este tipo de funciones pueden descomponerse en funciones simples utilizando el método de descomposición en fracciones parciales.

Paradigmas en Control

En el dominio del control de sistemas, existen tres tipos de análisis fundamentales que pueden llevarse a cabo dependiendo de cuales sean los elementos conocidos y

desconocidos del sistema. Los elementos considerados son tres: La señal de entrada del sistema, la señal de salida del sistema, el modelo del sistema. Cada uno de los tres tipos de análisis se caracteriza por encontrar el valor de un elemento desconocido siendo los otros dos conocidos.

Para el caso en el que la señal de entrada es conocida y el modelo del sistema bajo estudio es conocido mientras que la señal de salida del sistema es desconocida, el análisis es conocido como Análisis de respuesta. La idea de este análisis es conocer con exactitud completa las expresiones que definen la señal de salida del sistema en términos de los componentes conocidos que son, la señal de entrada y el modelo del sistema. Durante este análisis se obtiene un conocimiento más profundo del comportamiento del sistema puesto que el análisis no solo incluye la respuesta de una respuesta sino el análisis de varias señales de entrada que hacen parte de un conjunto de señales conocidas como el conjunto de señales de entrada estándar. De esta forma el conocimiento que se obtiene del sistema es mucho más amplio que reconocer el comportamiento ante una sola señal de entrada.

Si el modelo del sistema es conocido y existe una definición de la señal de salida del sistema, el análisis realizado se conoce como: Control del sistema. Normalmente la señal de salida ya conocida conoce a una señal de salida que se requiere que el sistema logre. Es una señal deseada que se quiere obtener ante una señal de entrada que no es conocida. Este análisis es útil porque permite al diseñador del sistema de control, determinar cuales son los bloques adicionales que debe incluir en el diseño final del sistema para lograr que la señal de salida sea exactamente la señal requerida bajo las restricciones propias del problema.

El último análisis, conocido como Identificación del sistema, ocurre cuando se conocen una pareja de señales de entrada y salida, pero no se conocen las expresiones matemáticas que describen el comportamiento del sistema. En este caso, el objetivo es obtener dicha representación matemática por medio de diferentes análisis. Este análisis es bastante útil y utilizado, pues permite que para cualquier sistema físico real que requiera ser integrado en un sistema de control, se puedan calcular las expresiones matemáticas que gobiernan su comportamiento y que posteriormente se pueda realizar un análisis de control de sistema dada una señal de salida deseada.

Desempeño del Sistema

El estudio del desempeño de sistemas de control corresponde al estudio de cinco aspectos importantes de los sistemas que son:

- Estudio de la respuesta dinámica del sistema, respuesta transitoria.
- Estudio de la respuesta estática del sistema, respuesta de estado estacionario.
- Estudio de la estabilidad intrínseca del sistema.
- Estudio de los márgenes de ganancia y fase en el dominio de la frecuencia.

A través del mecanismo conocido como realimentación, se puede ajustar la respuesta dinámica y la respuesta estática de los sistemas de control. La respuesta dinámica corresponde a la respuesta transitoria del sistema. Este es el tipo de respuesta que desaparece con el paso del tiempo y que es causada por cambios importantes en la señal de entrada. La respuesta estática del sistema corresponde a la respuesta en estado estacionario del sistema. Este es el tipo de respuesta que existe por largos periodos de tiempo una vez la señal de entrada tiene cambios que no son abruptos ni suficientes en magnitud o frecuencia para generar una respuesta transitoria en el sistema.

El estudio del desempeño de los sistemas de control tiene como objetivo facilitar las actividades de análisis y diseño que deben ser realizadas para modificar los sistemas de manera tal que puedan generar respuestas deseadas. Por medio de este análisis se puede medir varias características del sistema las cuales en conjunto con las especificaciones del desempeño del sistema permitirán ajustar los parámetros del sistema con el fin de obtener la respuesta deseada.

El estudio de la estabilidad del sistema permite determinar con exactitud cuales son los caos bajo los cuales el sistema es estable o aquellos bajos los cuales el sistema podría llegar a ser inestable. De esta forma cuando se ejecute la etapa de diseño, la selección de los parámetros puede realizar de forma que no se lleve al sistema hacia uno de los casos en los que se genere inestabilidad.

El estudio de rendimiento del sistema se completa con el análisis de varias medidas en el dominio de la frecuencia. Estas medidas permiten entender cual va ser la

respuesta del sistema ante señales de entrada en las que las componentes de frecuencia tengan un papel importante.

Como conclusión del estudio de rendimiento del sistema, se logra conocimiento de cómo el sistema reacciona ante cambios abruptos en la señal de entrada y ante periodos de no cambio, como reacciona el sistema cuando se incluye un lazo de realimentación y cuando el sistema podría convertirse en un sistema inestable y también como reacciona el sistema ante señales de entrada que cambian en frecuencia o periodo.

Análisis de desempeño de sistemas de primer orden

Los sistemas que se estudian en esta sección son los sistemas SISO (Single Input Single Output) LTI que pueden ser modelados mediante una ecuación diferencial lineal de primer orden con coeficientes constantes. Una representación gráfica de dicho sistema y la expresión que define la ecuación diferencial que modela dicho sistema se presentan a continuación.

$$a_1 \frac{dy(t)}{dt} + a_0 y(t) = bx(t)$$

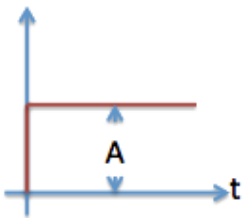
El análisis de desempeño para sistemas de primer orden, inicialmente incluye el estudio de la respuesta transitoria y en estado estable ante determinadas entradas de prueba. Para comprender la naturaleza de estas dos respuestas se hace necesario definir una señal de entrada de prueba y posteriormente solucionar la ecuación diferencial para la variable *y(t).* Como se mencionó en secciones anteriores, la solución de la ecuación diferencial se realiza por medio de la aplicación de la transformada de Laplace. Tras ejecutar la transformada de Laplace (asumiendo que las condiciones iniciales son todas igual a cero) y resolver la ecuación algebraica para la señal de salida se tiene que:

$$Y(S) = \frac{b}{a_1 \cdot s + a_0} X(s)$$

La expresión $\frac{b}{a_1 \cdot s + a_2}$ se conoce como la función de transferencia del sistema. Y resulta de la relación entre la entrada y la salida del sistema. La función de transferencia suele notarse con la letra *G.* Por lo tanto la función de transferencia para el sistema de primer orden mostrado anteriormente es:

$$G(s) = \frac{Y(S)}{X(S)} = \frac{b}{a_1 \cdot s + a_0}$$

Si se requiere conocer la respuesta del sistema ante una señal de entrada tipo paso con amplitud A, como la que se muestra a continuación, la función X(S) debe definirse como $\frac{A}{S}$ pues es esta la transformada de Laplace para dicha función.

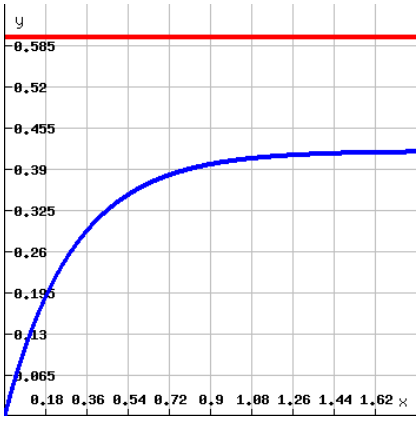


Tras expandir en fracciones parciales *Y(S)* y calcular la transformada inversa de Laplace, se obtiene como resultado la expresión matemática que permite calcular la señal de salida en el dominio del tiempo:

$$y(t) = \frac{b}{a_0} A \left(1 - e^{-\frac{a_0}{a_1} t}\right)$$

Las relaciones $\frac{b}{a_0}$ y $\frac{a_0}{a_1}$ se reescriben como dos constantes que tienen un significado visible en la curva que representa *y(t).* Las constantes son *K* y τ respectivamente. La constante K es conocida como la ganancia estática del sistema y corresponde al factor por el cual el valor en estado estable de la señal de entrada se escala y se convierte en un nuevo valor durante el estado estable de la señal de salida. La constante τ representa el tiempo que la señal y(t) tarda en alcanzar el 63.2% del valor que la señal de salida presenta cuando alcanza el estado estable. La definición de este par

de constantes se conoce como la parametrización del sistema, pues a partir del conocimiento de K y τse puede conocer el comportamiento exacto del sistema. La siguiente figura muestra con detalle la forma de la función y(t) (línea azul) para un par de valores K y τdeterminados (0.7 y 0.3 respectivamente) y una entrada tipo paso de amplitud 0.6 (línea roja).



Análisis de desempeño de sistemas de segundo orden

El análisis del desempeño para los sistemas de segundo orden corresponde a uno de los temas más amplios en los cursos de introducción a control. Los sistemas de segundo orden, a diferencia de los de primer orden, presentan una variedad de diferentes comportamientos que hace que para una señal de entrada determinada se deban estudiar diferentes casos que aumentan el esfuerzo del análisis.

Un sistema de segundo orden se puede representar mediante una ecuación diferencial de segundo orden como la que se muestra a continuación.

$$a_2 \frac{d^2y(t)}{dt} + a_1 \frac{dy(t)}{dt} + a_0 y(t) = \ bx(t)$$

Para conocer cual es la señal de respuesta del sistema ante una señal de entrada *x(t)* especifica, se debe resolver la ecuación diferencial para la señal de salida *y(t)*. Cuando este solución se realiza por medio de la transformada de Laplace, se obtiene como función de transferencia la siguiente expresión.

$$G(S) = \frac{Y(S)}{X(S)} = \frac{b}{a_2 s^2 + a_1 s + a_0}$$

De igual forma que en los sistemas de primer orden, el procedimiento para conocer la salida del sistema dada una entrada, consiste en determinar la transformada de Laplace *X(S)* para la señal de entrada, resolver la ecuación algebraica para obtener Y(S) en términos de un polinomio en la variable S y posteriormente calcular la transformada inversa de Laplace usando como mecanismo de ayuda la expansión en fracciones parciales. Todo este proceso resulta más sencillo de analizar cuando se realiza una parametrización conveniente en la función de transferencia del sistema. La parametrización incluye la creación de tres constantes (K, Ganancia estática; ζ, factor de amortiguamiento yω_n, frecuencia natural)y la sustitución de los coeficientes que provienen de la ecuación diferencial por las nuevas constantes. La función de transferencia del sistema se puede escribir en forma parametrizada usando la siguiente expresión.

$$G(S) = \frac{K{\omega_n}^2}{s^2 + 2\zeta\omega_n s + {\omega_n}^2}$$

Una vez se definen los nuevos parámetros, varios comportamientos pueden determinarse a partir de la forma como estos parámetros varían. La respuesta transitoria del sistema esta determinada por la raíces del denominador de la función de transferencia. La ecuación resultante de igualar el denominador de la función de transferencia a cero, se conoce como la ecuación característica del sistema $\Delta(s)$.

$$\Delta(s) = s^2 + 2\zeta\omega_n s + {\omega_n}^2 = 0$$

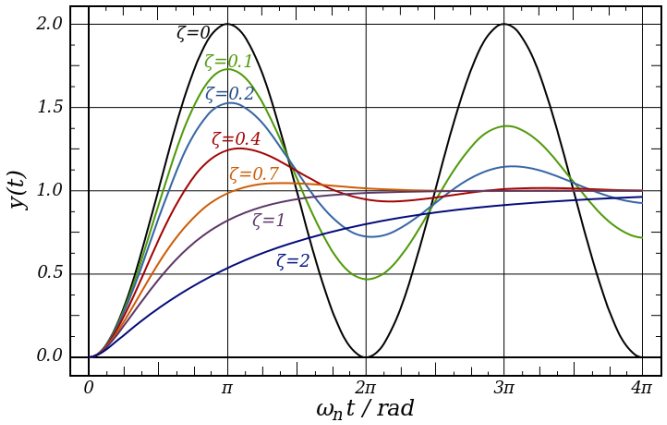
La solución general de esta ecuación viene dada por dos raíces que se pueden escribir como:

$$s_{1,2} = -\zeta\omega_n \pm \omega_n \sqrt{\zeta^2 - 1}$$

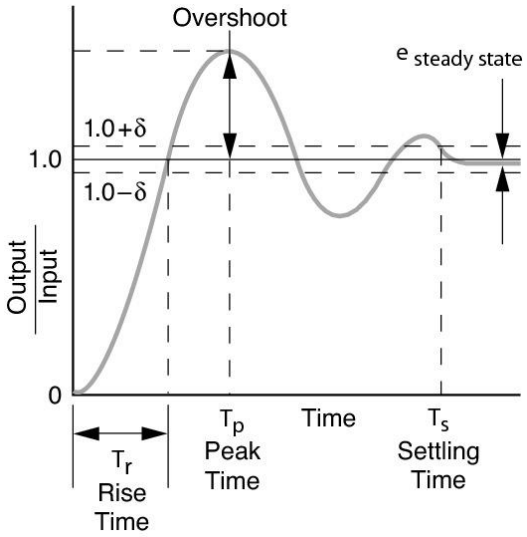
Dependiendo del valor que tome el parámetro ζse generan varios casos de solución para las raíces de la ecuación característica así como diferentes comportamientos del sistema. A continuación se describen los cuatro casos:

1. Caso sobre-amortiguado: Ocurre cuando el valor de $\zeta > 1$. Para este caso las soluciones de la ecuación característica pertenecen al conjunto de los números reales y las soluciones son diferentes.
2. Caso críticamente Amortiguado: Ocurre cuando del valor de $\zeta = 1$. Para este caso las soluciones de la ecuación característica pertenecen al conjunto de los números reales, pero las soluciones son iguales, es decir $s_1 = s_2$.
3. Caso sub-amortiguado: Ocurre cuando el valor de $0 < \zeta < 1$. Para este caso las soluciones de la ecuación característica pertenecen al conjunto de los números complejos, lo que significa que las soluciones tienen tanto parte real como parte imaginaria. Para este caso las soluciones son diferentes.
4. Caso oscilatorio: Ocurre si el valor de $\zeta = 0$. Para este caso las soluciones a la ecuación característica pertenecen al conjunto de los números imaginarios. La parte real de la respuesta se hace cero. Las soluciones a la ecuación característica son diferentes pero ambas son imaginarias.

Al definir como señal de entrada la función paso de amplitud A, se debe proceder a resolver *Y(S)* a partir de la ecuación que define la función de transferencia del sistema. Al ejecutar este cálculo y mientras se descompone la expresión de *Y(S)* en fracciones parciales, se puede notar que la descomposición de Y(S) es diferente para cada uno de los casos mencionados en el apartado anterior debido a que la descomposición en fracciones parciales depende de las raíces del denominador de la función racional a descomponer. Dado lo anterior, para una señal de entrada de tipo paso, existen 4 posibles señales de salida que dependen altamente en su forma y naturaleza del parámetro ζ. Las cuatro posibles señales de salida se muestran a continuación en la siguiente figura.



De los cuatro comportamientos mencionados uno de especial interés es el comportamiento sub-amortiguado. La respuesta transitoria de este tipo de comportamiento es en general una señal senoidal acotada por dos envolventes de tipo exponencial, una superior y una inferior. La amplitud de la señal senoidal tiene un decaimiento exponencial y simétrico con el paso del tiempo. Durante la duración de la respuesta transitoria, varias características de la señal resultan de importan puesto que dichas características se encuentran directamente relacionadas a través de expresiones matemáticas con los parámetros utilizados para describir la función de transferencia. Estas características de la señal se conocen como las medidas de rendimiento estándar para un sistema sub-amortiguado de segundo orden. En la siguiente figura se muestran las cinco características, para un caso en el que un sistema de segundo orden con respuesta de tipo sub-amortiguada, es excitado con una entrada tipo paso de amplitud uno.



Estás cinco características y las relaciones matemáticas con los parámetros de la función de transferencia son:

- Sobrepaso (Overshoot) : Corresponde a la magnitud máxima de la señal de salida que sobrepasa el valor de la señal de entrada.

$$Ov = e^{-\frac{\zeta\pi}{\sqrt{1-\zeta^2}}}$$

- Tiempo de levantamiento (Rise time): El tiempo de levantamiento es el tiempo que transcurre desde el momento en que inicia la respuesta transitoria del sistema hasta que dicha respuesta alcanza el valor de la señal de entrada.

$$t_r \cong \frac{3}{\zeta\omega_n}$$

- Tiempo pico (Peak time): El tiempo pico es el tiempo que transcurre desde el inicio de la respuesta transitoria hasta el momento en que la señal de salida alcanza su valor máximo.

$$t_r \cong \frac{2.16\zeta + 0.60}{\omega_n}$$

- Tiempo de asentamiento (settling time): El tiempo de asentamiento mide el tiempo que transcurre desde que inicia la respuesta transitoria hasta que esta termina. El tiempo de asentamiento también puede verse como el tiempo en el que inicial la respuesta estacionaria del sistema.

$$t_r \cong \frac{3}{\zeta\omega_n} \; 0 < \zeta < 0.69$$

$$t_r \cong \frac{4}{\zeta\omega_n} \; \zeta > 0.69$$

- Error de estado estable (Steady state error): El error en estado estable es la magnitud de la señal que hace falta para que la señal de salida sea exactamente igual a la señal de entrada. Se calcula como la diferencia de las magnitudes de la señales de entrada y de salida cuando el sistema ha alcanzado el estado estable. Las expresiones matemáticas que relacionan el error en estado estable con los parámetros de la función de transferencia se estudian en la siguiente sección.

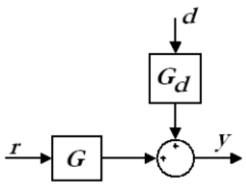
La relación entre los parámetros de la función de transferencia y las medidas de rendimiento son de especial importante debido a que abren la posibilidad de realizar un proceso de identificación del sistema de una forma directa. Una vez la señal de respuesta del sistema es conocida, solo se requiere medir cada una de sus características de rendimiento y reemplazar el valor de las mediciones en las expresiones mostradas anteriormente para obtener como resultado los valores de los parámetros de la función de transferencia.

Análisis del Error en estado estacionario

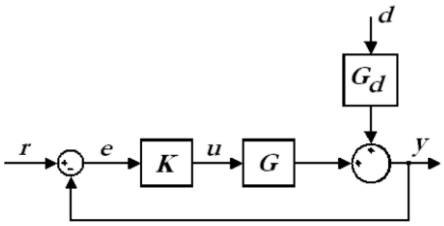
El error en estado estacionario es una de las medidas de rendimiento más importante que se estudian en los sistemas de control. Su importancia radica en el hecho que esta medida permite determinar que tan cerca o lejos se encuentra la salida del

sistema controlado respecto a la señal de salida que se desea. Esta medida de rendimiento, en el dominio del tiempo, se calcula como la diferencia en magnitud de la señal de salida del sistema controlado respecto a la señal de referencia que representa la señal de salida deseada, una vez el sistema de control ha alcanzado el estado estable. Sin embargo, en el dominio de los sistemas de control el cálculo de esta medida de rendimiento se puede realizar en el dominio de la variable compleja S, evitando de esta forma tener que realizar el cálculo de las transformadas inversas de Laplace. El teorema del valor final de la transformada de Laplace es la herramienta básica que hace posible que este análisis pueda realizarse en el dominio de la variable S.

Dado un sistema de control como el que se muestra en la siguiente figura, es importante reconocer que el error en estado estable no es directamente medible en algún punto del sistema como una variable física.



Debido a que la definición del error corresponde a la diferencia entre la señal de salida y la señal de entrada y esta diferencia no se encuentra implementada como parte del sistema, el valor del error no es directamente medible a partir de los componentes y conexiones del sistema. Sin embargo para un sistema de control con realimentación unitaria como el que se muestra en la siguiente figura, si existe un punto físico en el cual se puede realizar la medición del error. Para el caso de la imagen la salida del sumador, correspondiente a la señal marcada con la letra e, resulta ser el punto donde se puede medir físicamente el error del sistema para un momento determinado.



Debido a que ciertos sistemas de control no cuentan con esta capacidad, el análisis en el dominio de la variable compleja S toma mayor importancia. Para los sistemas de la figura anterior las expresiones que permiten calcular el error en el dominio de Laplace se presentan a continuación:

$$E(s) = R(s) - Y(s) = [1 - G(s)]R(s)$$

$$E(s) = R(s) - Y(s) = \frac{1}{1 + K(s)G(s)} R(s)$$

Para calcular el valor que tiene la variable de error cuando el sistema alcanza el estado estable, se requiere calcular el valor del error cuando el tiempo tiende a infinito. Este cálculo se realiza de forma sencilla utilizando el teorema del valor final de la transformada de Laplace el cual se presenta a continuación:

$$\lim_{t \to \infty} f(t) = \lim_{s \to 0} sF(s)$$

De esta forma el error, cuando el tiempo tiende a infinito, momento en el cual se garantiza que el sistema ha alcanzado el estado estable es:

$$e_{ss} = \lim_{t \to \infty} e(t) = \lim_{S \to o} S \cdot E(s)$$

Si se supone que el sistema presentado en la figura, es excitado con una entrada paso unitario, entonces la expresión del error en el dominio de Laplace en conjunto con el teorema del valor final permiten establecer que:

$$e_o(\infty) = \lim_{s \to 0} s \left[\frac{1}{1 + K(s)G(s)} \right] \frac{1}{s} = \frac{1}{1 + K(0)G(0)}$$

Para el sistema de lazo abierto el mismo cálculo resulta ser:

$$e_o(\infty) = \lim_{s \to 0} s[1 - G(s)] \frac{1}{s} = 1 - G(0)$$

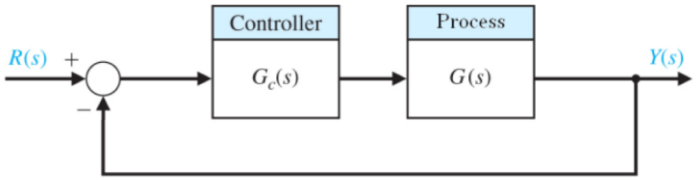
K(0) y G(0) suelen denominarse las ganancias estáticas de los bloques de K y G. Usualmente el bloque G representa el proceso físico a controlar, también conocido como la planta y normalmente su ganancia estática no puede ser modificada pues es una característica intrínseca del sistema. El bloque K, corresponde con frecuencia al bloque de control del proceso físico y sus parámetros pueden ser modificados a discreción del diseñador puesto que es un sistema parametrizable que normalmente

se introduce en el sistema para mejorar el rendimiento de la respuesta del sistema físico.

Comparando las dos expresiones se puede evidenciar que para el sistema que cuenta con realimentación unitaria, el error en estado estable puede reducirse si aumenta la ganancia estática del bloque de control. Esta posibilidad no existe en el sistema de lazo abierto y esta es una de las razones más importantes por las cuales el mecanismo de realimentación es bastante útil e importante.

El error en estado estable como se puede observar en la ecuación x, depende tanto de los componentes del sistema como de la entrada al mismo. Por esta razón la cantidad de posibilidades de cálculo del error en estado estable al variar la entrada, aumenta de forma lineal con el numero de entradas posibles. Para reducir el número de cálculos a realizar parte del análisis incluye la definición de unos casos estándar, los cuales pueden ser usados con facilidad pues la mayoría de sistemas de control y de topologías pueden ser transformadas al caso estándar y de esa forma reutilizar los resultados de análisis del caso estándar fácilmente.

Si se define un sistema de control con realimentación unitaria como el que se muestra en la siguiente figura, la expresión para el error en el dominio de la variable S estaría definida por la expresión mostrada.



$$E(s) = \frac{1}{1 + G_c(s)G(s)} R(s)$$

Y de forma general el error es estado estable puede calcularse como:

$$e_{ss} = \lim_{t \to \infty} e(t) = \lim_{s \to 0} \left[s \frac{1}{1 + G_c(s)G(s)} R(s) \right]$$

Si la entrada R(S) es la entrada paso unitario entonces el error en estado estable se determinar por medio de la siguiente expresión:

$$e_{ss} = e_p = \lim_{s\to 0}\left[\frac{s(A/s)}{1+G_c(s)G(s)}\right] = \frac{A}{1+\lim_{s\to 0}[G_c(s)G(s)]}$$

Esta expresión se puede simplificar si se crea una nueva variable utilitaria llamada constante de error de posición definida de la siguiente forma:

$$K_p = \lim_{s\to 0}[G_c(s)G(s)]$$

$$e_p = \frac{A}{1+K_p}$$

Utilizando esta nueva constante de posición solo se requiere calcular limite cuando la variable s tiende a cero del producto de dos bloques, una vez este valor ha sido calculado, el cálculo de error en estado estacionario es directo.

En forma similar, se pueden definir constantes de error para las entradas rampa y parábola. Estas constantes se conocen como constantes de velocidad y aceleración y se presentan a continuación:

$$K_v = \lim_{s\to 0}[sG_c(s)G(s)] \quad \text{y} \quad K_a = \lim_{s\to 0}[s^2G_c(s)G(s)]$$

Los valores del error en estado estacionario con la definición de estas nuevas constantes sería, respectivamente:

$$e_v = \frac{m}{K_v} \quad \text{y} \quad e_a = \frac{\alpha}{K_a}$$

En general la función de transferencia de lazo tiene la forma que se muestra a continuación. El exponente N se conoce como el tipo del sistema y puede variar de cero a infinito siendo siempre un numero entero.

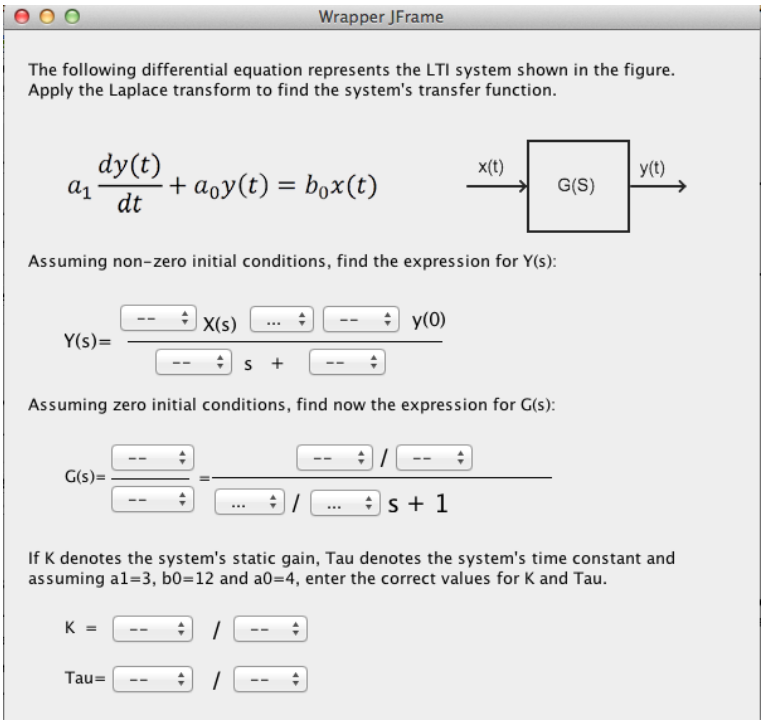
$$G_c(s)G(s) = \frac{K\prod_{i=1}^{M}(s+z_i)}{S^N\prod_{k=1}^{Q}(s+p_k)}$$

Esta expresión general, permite el cálculo directo de cada una de las constantes de error, pues al estar definido el producto Gc(S)G(S), los limites involucrados en las definiciones de cada una de las constates pueden ser evaluados. El resumen de dicha evaluación se presenta en la siguiente tabla.

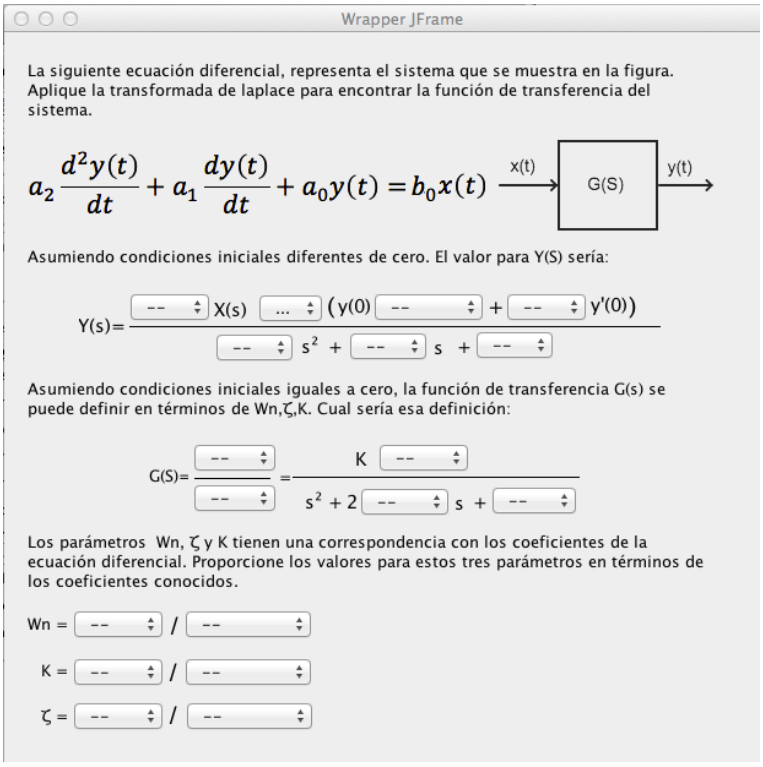
T(s)	e_p $e_{ssp} = \frac{A}{1+K_p}$	e_v $e_{ssv} = \frac{m}{K_v}$	e_a $e_{ssa} = \frac{\alpha}{K_a}$
Type 0	kte	∞	∞
Type 1	0	kte	∞
Type 2	0	0	kte
Type 3	0	0	0

Anexo F: Interfaces gráficas para problemas seleccionados.

Problema P4

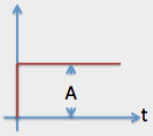


Problema P7



Problema P9

The LTI system shown in the next figure, is modeled by the transfer function G(s). A non-unitary Step input is applied to the system. Apply the inverse Laplace transform and calculate the time-domain system's output.



$$G(s) = \frac{a}{b \cdot s + c}$$

1.a) Start by calculating Y(s). Enter the partial fractions expansion result:

$$Y(s) = \frac{a}{b\,s + c} \qquad Y(s) = A\left(\frac{\square}{s + \square} + \frac{\square}{s}\right)$$

1.b) Enter the K and Tau values. Complete y(t) using the shown the expression.

K = ... / ...

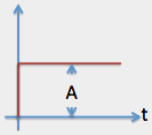
Tau= ... / ...

$$y(t) = KA\left(\square\; e^{-\frac{\tau}{t}} + 1\right)$$

Problema P8

La siguiente ecuación diferencial, representa el sistema que se muestra en la figura. Aplique la transformada de laplace para encontrar la función de transferencia del sistema. SI la entrada al sistema es la señal que se muestra en la figura, cual es la expresión en el dominio del tiempo que describe la salida del sistema.

$$a_1 \frac{dy(t)}{dt} + a_0 y(t) = b_0 x(t)$$



Asumiendo condiciones iniciales iguales a cero. Ingrese el valor para G(S), K y Tau

$$G(s) = \frac{\square}{\square} = \frac{\square / \square}{(\square / \square)\,s + 1}$$

K = -- / --

Tau= -- / --

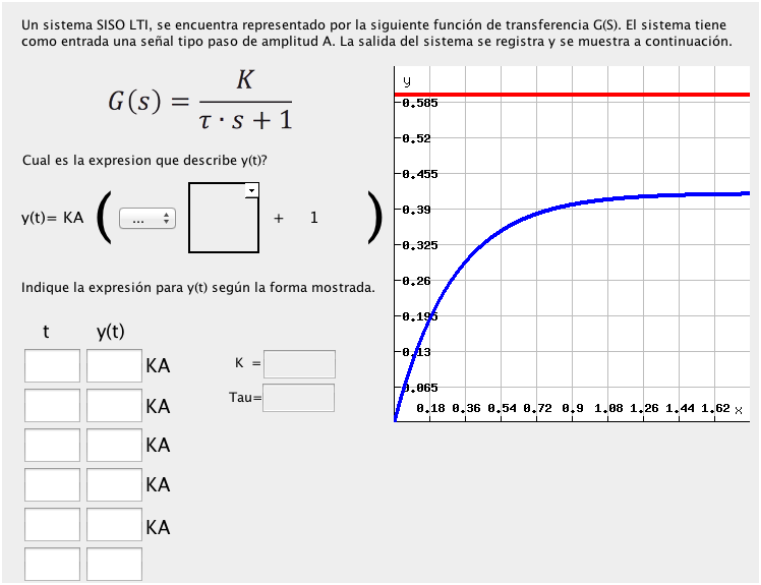
Calcule el valor de Y(S) en su forma expandida en fracciones parciales

$$Y(s) = A\left(\frac{\square}{s + \square} + \frac{\square}{s}\right)$$

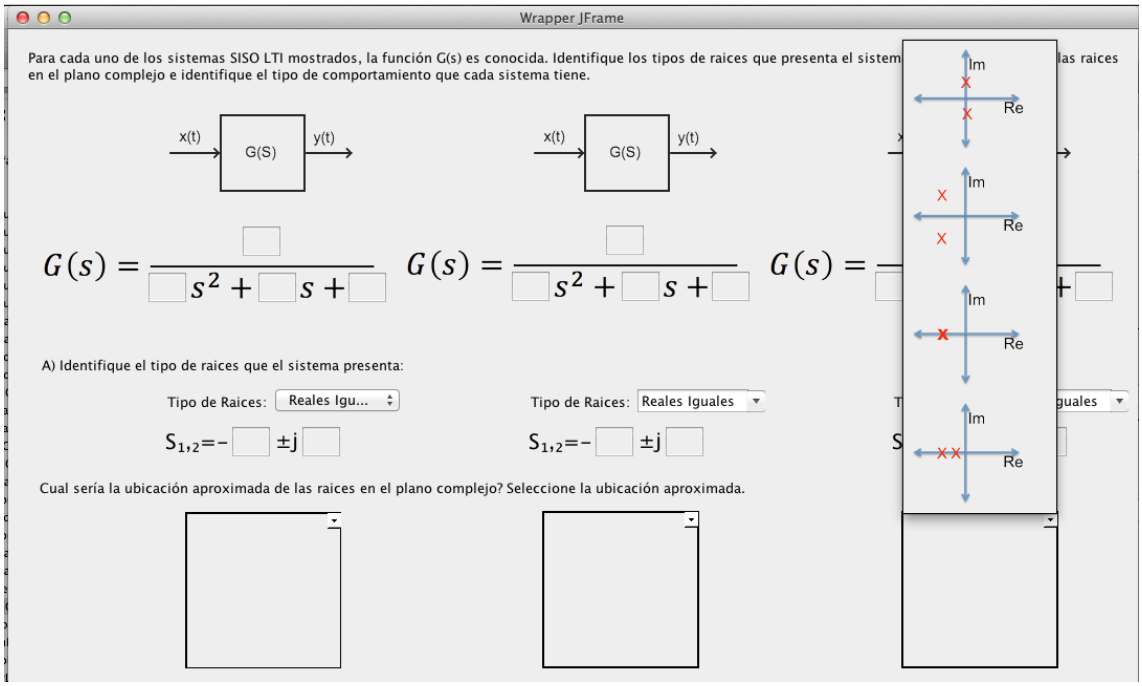
Indique la expresión para y(t) según la forma mostrada.

$$y(t) = KA\left(\square + 1\right)$$

Problema P36



Problema P60



Anexo G: Instrumento de recolección de datos aplicado durante la prueba piloto.

Problema 1

Hora de Inicio:

Tipo Sistema: a) CTAT b) ASPIRE

Hora de Finalización:

1. Le resultó fácil de usar la interfaz gráfica?

   Muy Difícil Muy Fácil

   1 2 3 4 5

2. Pudo entender fácilmente la representación de las expresiones matemáticas en la pantalla?
   a) Si.
   b) No, tuve que solicitar asistencia para entender la interfaz gráfica
   c) No, aun con asistencia no comprendí completamente como usar la interfaz gráfica.

   Por favor indique las razones:

3. Considera que las gráficas presentadas, los elementos de interacción (campos y listas de selección) y los textos descriptivos son apropiados para la representación matemática del problema.

   Nada Muy Apropiadas

   1 2 3 4 5

   Por favor indique las razones:

4. En cuales pasos del procedimiento tuvo que pedir la respuesta final al tutor?

5. Respecto a los mensajes de ayuda presentados por el sistema. Le parecieron útiles?

   Nada Muy Útiles

   1 2 3 4 5

   Por favor indique las razones:

6. Preferiría que los mensajes de ayuda fueran más detallados acerca de los procesos de cálculo?

   a) Si b) No c) No lo tengo claro.

7. Le gustó algo en particular del sistema

8. Encontró algo frustrante o que en particular le haya disgustado durante la interacción con el sistema?

9. Según su opinión, como se podría mejorar la interfaz gráfica?

10. Según su opinión, que funcionalidades adicionales le gustaría que el sistema tuviera?

Anexo H: Reporte de la última inspección realizada porun experto.

Tutor 1 – Aplicación Trans. de Laplace

Sugerencia: Editar
The following differential equation represents the LTI system shown in the figure.

Sugerencia: Editar
Assuming non-zero initial conditions, find the expression for Y(s)

Corregir desarrollo matemático de Expresión de Y(s): La condición inicial va con signo más y le falta el término a1.

Sugerencia: Editar
Assuming zero initial conditions, find nowthe expression forG(s)

Editar: Todas las variables “s” son minúsculas.

Sugerencia: En el feedback de Tau, editar:
Versión original
Delay time, is the time the system's output takes to reach 69% of the final value. It is a common constant used in the transfer function to describe the system's response speed.
Versión sugerida:
Time constant (Tau), in a First Order Lag model is the time the system's output takes to reach 63.2% of the final value when a step input is applied. It is a common constant used in the transfer function to describe the system's response speed.

En la parte de cálculo de K y Tau, tal vez convendría que los cocientes no fuesen simples reemplazos de a0, a1 y bo, sino expresiones equivalentes, por ejemplo: 12/4 expresarlo como 3/1 y 3/4 si dejarlo como 3/4.

Tutor 2 – Aplicación Trans. Inversa de Laplace

Sugerencia: Ajustar texto del siguiente Feedback:
Calculate the Laplace transform of the input signal. Lookup a table if necessary.
(Pareciera sugerir que se use una Tabla, en vez de buscar en una tabla de transformadas).

Revisar redacción de enunciado 1b). Aparece dos veces “the” en la misma frase.

Ajustar agrupamientos de respuestas para que el Tutor “no imponga” un orden estricto de entrada de las respuestas. Por ejemplo, en la apertura en fracciones parciales está obligando a hacer primero numerador y luego denominador. Podría resolverse también una parte del numerador, denominador y luego otra vez la parte derecha del numerador.

Revisar Feedback:
If Tau is system's time constant. Given the transfer function shown in the figure, What should be the value for Tau?

Editar: Todas las variables “s” son minúsculas.

Tutor 3 – Sistema de Segundo Orden

Observación: Descripción aparece en Español. Si se deja así, faltan algunas tíldes.

Editar
La ecuación en la imagen aparece con coeficiente “b”, pero al momento de seleccionar aparece b0. Revisar.

Revisar
Signo de las condiciones iniciales. A la izquierda sería negativo, al pasar a la derecha quedaría positivo.

Revisar
Cálculo de zita y mensaje de Feedback asociado. Diría que el mensaje está bien, pero la selección no corresponde. La respuesta correcta es zita=a1/(2*sqrt(a0*a2)).

Editar: Todas las variables “s” son minúsculas.

Tutor 4 – Sistema de primer orden pero ahora con cálculo de G(s), Trans. Inversa y expresión y(t)

Revisar
Mensaje de Feedback en el cálculo de Tau. Si uno para después de introducir el numerador de Tau, el mensaje del <Help> hace referencia a K, no a Tau. Revisar:
Por comparación calcule la ganancia estática K.

Revisar
Faltaría un signo menos en el primer término del desarrollo en fracciones parciales?

Editar: Todas las variables “s” son minúsculas.

Anexo I: Resumen de comentarios realizados por los participantes de la prueba piloto .

Usuario	Sistema	Lo que más gustó	Comentarios varios	Comentarios destacados
unal1	ASPIRE	ayuda directa, interfaz grafica	Tamaño adecuado de las expresiones, buena organización de expresiones, facilidad entender expresiones, indica errores, da consejos de cómo resolver y teoría de cómo funciona,	Indicaba que parte estaba mal , además de dar consejos o teoría de que hacer para responder bien esa parte. Podría tener la opción de pasar a otros idiomas
	ASPIRE			
	CTAT	Feedback Inmediato	Interfaz lenta, aumentar tamaños, mas cantidad de problemas	
unal2	ASPIRE	Que permite corregir y aclarar dudas	Ayudas muy generales, se puede simplificar y el sistema no lo reconoce, incluir flexibilidad para las expresiones matemáticas, que no sean rígidas	Podría tener la opción de pasar a otros idiomas. Podrían incluirse formas alternativas para las funciones matemáticas expuestas con el fin de no brindar al estudiante la sensación de que existe una única forma de expresar una respuesta. Los mensajes me fueron útiles
	ASPIRE			
	CTAT	Feedback Inmediato	Mejorar tamaño de los textos, ajustar dimensiones de las listas desplegables	

				cuándo cometí errores de cálculo y de despejes.
unal3	ASPIRE	Resalta los errores, Aclara conceptos	Hacer uso de paréntesis, Proporcionar la respuesta en la forma en que la piden, Mostrar la gráfica de respuesta.	La interfaz es bastante clara en cuanto a las expresiones matemáticas, simplemente uno tiene que darse cuenta la forma correcta de expresar la función de transferencia cuando la piden, La vez que me equivoqué, el tutor me hizo caer en cuenta del error que había cometido al realizar el despeje de Y(s), fue bastante útil
	ASPIRE	Es bueno porque se tiene que hacer factorización para llegar a la respuesta deseada,	no use ninguna ayuda, en el paso 2 no entendí que debía hacer, reflexioné y luego lo entendí, incluir la respuesta del sistema para los valores que el usuario define de a,b,c y A	
	CTAT		Mejorar la interfaz, problemas con las enunciados, Usar tutores para variables de estado sería útil	
unal4	CTAT	La forma en que indica si la respuesta es correcta o no	Muy lento el sistema, Los valores para seleccionar Zeta son confusos, adicionar paso a paso si el estudiante se equivoca varias veces	Nuevamente se menciona que el tema de variables de estado debería incluirse. Usuario que hace poco uso del sistema de ayuda.
	CTAT		No es clara la ultima pregunta, Es útil la grafica de la entrada del sistema, incluir imágenes en las ayudas	
	ASPIRE	Nada que resaltar	Tuve un error la pista ayudó a salir del error, incluir variables de estado, los libros son a veces simples y no contienen todos los pasos	
unal5	CTAT	Fácil de diligenciar	Sistema lento, Las ayudas incrementales gustaron, se debe mostrar la forma de calcular la respuesta final, la interfaz es intuitiva	Se destaca la diferencias en velocidades del sistema, se destaca que las interfaces son intuitivas
	CTAT	Es fácil de usar y resalta si uno se equivoca.	Lentitud del sistema	
	ASPIRE	Mas rápido y es intuitivo. Cuando se pide la	Se desea que tenga feedback inmediato y no retrasado	

		respuesta del sistema es importante que se indique como se calcula.		
unal6	CTAT	Que los mensajes de ayuda son detallados,	Las ecuaciones en la ayuda deben presentarse de igual forma que en la interfaz, la interfaz puede ser mas amigable pero es fácil de manejar, agrandando botones, mejorar rapidez	Es una idea nueva, creativa, y uno no tiene la frustración de que si no entiende algo, se queda sin argumentos para afrontar el problema, ya que el tutor da consejos o recomendaciones para realizarlo.
	CTAT	Enunciados claros.	Mejorar la visualización en Windows pues en Mac luce mucho mejor, que se puedan introducir expresiones para graficar, o para hacer simulación	
	ASPIRE	La interfaz es más limpia y rápida.		
unal7	CTAT	Es práctico, genera satisfacción personal.	no pasar por todas las ayudas sino seleccionar ayudas de un listado de acuerdo al problema, se obliga al estudiante a hacer un mayor esfuerzo debido a que hay muchas respuestas disponibles que son parecidas	Claro, no es lo mismo depender solo de materiales como libros, porque no todos usan una metodología que sea entendible para todos, por lo cual el tutor brindaría una oportunidad interesante de aprendizaje.
	CTAT	Es organizado y claro el desarrollo	Es lento cuando se proporcionar algunas respuestas, falta claridad mensajes de ayuda, incluir ejemplos de fracciones parciales,	
	ASPIRE	Se muestra la gráfica de comportamiento de primer orden	Falta más detalle en la ayuda,	

unal8	CTAT	La idea general es excelente, la ubicación de la ayuda al lado es útil	Lento al seleccionar opciones, el uso de listas es bueno pero se puede incluir opciones de texto que luego se vuelvan opciones si el usuario se equivoca muchas veces	Incluir posibilidad de graficar expresiones con parámetros modificables, Es mucho mejor aprender con una corrección inmediata ya que de esta manera se puede avanzar mucho más rápido a la hora de estudiar y entender.
	ASPIRE		No es del todo claro como debe ser usado el sistema	
	CTAT	La forma como da las alertas de los errores	Lentitud en la selección de opciones	
unal9	ASPIRE	Es sencillo y amigable.	Es frustrante no recordar como es una transformada de Laplace con condiciones no nulas.	El uso de ingles es adecuado y apropiado. La barra inclinada para representar fracciones no es apropiada. Reducir barras de scroll, Unificar en un solo formato las imágenes, es decir , las que existen son buenas y apropiadas, pero pareciera en casos un copy paste. Incluir referencias Bibliográficas. Existe la oportunidad de Equivocarse y eso es bueno. No reemplaza al profesor ni al texto.
	ASPIRE	Posibilidad de ampliar imágenes de ayuda	Los mensajes dan ayuda pero no quitan la labor de pensar, inducen a hacerlo, hubo una caída de red y la experiencia se cortó	
	CTAT	Realimentación inmediata, aunque puede prestarse para que se adivinen las respuestas	La presentación parece anticuada, es muy sobria. Se debe mejorar la distribución de los campos,	

Bibliografía

Aleven, V. (2010). Rule-Based Cognitive Modeling for Intelligent Tutoring Systems. In R. Nkambou, J. Bourdeau, & R. Mizoguchi (Eds.), *Advances in Intelligent Tutoring Systems* (pp. 33–62). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-14363-2_3

Aleven, V., Mclaren, B. M., Sewall, J., & Koedinger, K. R. (2009). A New Paradigm for Intelligent Tutoring Systems: Example-Tracing Tutors. *Int. J. Artif. Intell. Ed.*, *19*(2), 105–154.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive Tutors: Lessons Learned. *Journal of the Learning Sciences*, *4*(2), 167–207. doi:10.1207/s15327809jls0402_2

Baker, R., Corbett, A. T., & Koedinger, K. R. (n.d.). *The Difficulty Factors Approach to the Design of Lessons in Intelligent Tutor Curricula*.

Barrios, A., Panche, S., Duque, M., Grisales, V. H., Prieto, F., Villa, J. L., … Canu, M. (2013). A multi-user remote academic laboratory system. *Computers & Education*, *62*, 111–122. doi:10.1016/j.compedu.2012.10.011

Bloom, B. S. (1984). The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring. *Educational Researcher*, *13*(6), 4–16. doi:10.3102/0013189X013006004

Bourdeau, J., & Grandbastien, M. (2010). Modeling Tutoring Knowledge. In R. Nkambou, J. Bourdeau, & R. Mizoguchi (Eds.), *Advances in Intelligent Tutoring Systems* (Vol. 308, pp. 123–143). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from http://link.springer.com/10.1007/978-3-642-14363-2_7

Bourdeau, J., Mizoguchi, R., Psyché, V., & Nkambou, R. (2004). Selecting Theories in an Ontology-Based ITS Authoring Environment. In J. C. Lester, R. M. Vicari, & F. Paraguaçu (Eds.), *Intelligent Tutoring Systems* (pp. 150–161). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/978-3-540-30139-4_15

Cala, O., & Grisales, V. H. (2014). Development of Example-Tracing Tutors for teaching Control Systems Performance fundamentals.

Casteblanco, D., & Avila, I. (2009). *Diseño de una plataforma de laboratorios virtuales orientada al curso de control de sistemas en tiempo continúo de la Universidad Distrital*. Universidad Distrital Francisco José de Caldas.

Clark, R., Feldon, D., Merrienboer, J. J. van, Yates, K., & Early, S. (2008). Cognitive task analysis. *Handbook of Research on Educational Communications and Technology*, 577–593.

Cohen, P. A., Kulik, J. A., & Kulik, C.-L. C. (1982). Educational Outcomes of Tutoring: A Meta-analysis of Findings. *American Educational Research Journal*, *19*(2), 237–248. doi:10.3102/00028312019002237

Forgy, C. L. (1982). Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, *19*(1), 17–37. doi:10.1016/0004-3702(82)90020-0

Fournier-Viger, P., Nkambou, R., & Nguifo, E. M. (2010). Building Intelligent Tutoring Systems for Ill-Defined Domains. In R. Nkambou, J. Bourdeau, & R. Mizoguchi (Eds.), *Advances in Intelligent Tutoring Systems* (pp. 81–101). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-14363-2_5

Hayes-Roth, F., Waterman, D. A., & Lenat, D. B. (1983). *Building Expert Systems*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Koedinger, K. R., Aleven, V., Heffernan, N., McLaren, B., & Hockenberry, M. (2004). Opening the Door to Non-programmers: Authoring Intelligent Tutor Behavior by Demonstration. In J. C. Lester, R. M. Vicari, & F. Paraguaçu (Eds.), *Intelligent Tutoring Systems* (pp. 162–174). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/978-3-540-30139-4_16

Koedinger, K. R., Anderson, J. R., Hadley, W. H., & Mark, M. A. (1997). Intelligent Tutoring Goes to School in the Big City. *International Journal of Artificial Intelligence in Education*, 30–43.

Koedinger, K. R., & Corbett, A. T. (2006). Cognitive tutors : technology bringing learning science to the classroom. In R. K. Sawyer (Ed.), *The Cambridge Handbook of the Learning Sciences* (pp. 61–77). Cambridge University Press. Retrieved from https://hal.archives-ouvertes.fr/hal-00699796

Lovett, M. C. (1998). Cognitive Task Analysis in Service of Intelligent Tutoring System Design: A Case Study in Statistics. In B. P. Goettl, H. M. Halff, C. L. Redfield, & V. J. Shute (Eds.), *Intelligent Tutoring Systems* (pp. 234–243). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/3-540-68716-5_29

Mark, M. A., & Greer, J. E. (1993). Evaluation Methodologies for Intelligent Tutoring Systems. *Journal of Artificial Intelligence in Education*, *4*, 129–53.

Martin, B., Mitrovic, A., Suraweera, P., & others. (2007). Domain modelling with ontology: A case study. In *Proceedings of the Fifth International Workshop on Authoring of Adaptive and Adaptable Hypermedia* (pp. 4–11). National Center of Scientific Research'Demokritos'.

Mitrovic, A. (1998). Learning SQL with a Computerized Tutor. In *Proceedings of the Twenty-ninth SIGCSE Technical Symposium on Computer Science Education* (pp. 307–311). New York, NY, USA: ACM. doi:10.1145/273133.274318

Mitrovic, A. (2003). An Intelligent SQL Tutor on the Web. *International Journal of Artificial Intelligence in Education*, *13*(2), 173–197.

Mitrovic, A. (2012). Fifteen years of constraint-based tutors: what we have achieved and where we are going. *User Modeling and User-Adapted Interaction*, *22*(1-2), 39–72. doi:10.1007/s11257-011-9105-9

Mitrovic, A., Koedinger, K. R., & Martin, B. (2003). A Comparative Analysis of Cognitive Tutoring and Constraint-Based Modeling. In P. Brusilovsky, A. Corbett, & F. de Rosis (Eds.), *User Modeling 2003* (pp. 313–322). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/3-540-44963-9_42

Mitrovic, A., Martin, B., & Suraweera, P. (2007). Intelligent Tutors for All: The Constraint-Based Approach. *IEEE Intelligent Systems*, *22*(4), 38–45. doi:10.1109/MIS.2007.74

Mitrovic, A., Martin, B., Suraweera, P., Zakharov, K., Milik, N., Holland, J., & Mcguigan, N. (2009). ASPIRE: An Authoring System and Deployment Environment for Constraint-Based Tutors. *Int. J. Artif. Intell. Ed.*, *19*(2), 155–188.

Mitrovic, A., Mayo, M., Suraweera, P., & Martin, B. (2001). Constraint-Based Tutors: A Success Story. In L. Monostori, J. Váncza, & M. Ali (Eds.), *Engineering of Intelligent Systems* (pp. 931–940). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/3-540-45517-5_103

Mitrovic, A., McGuigan, N., Martin, B., Suraweera, P., Milik, N., & Holland, J. (2008). Authoring constraint-based tutors in ASPIRE: a case study of a capital investment tutor. Retrieved from http://researcharchive.lincoln.ac.nz/handle/10182/3336

Mitrovic, A., & Ohlsson, S. (1999). Evaluation of a constraint-based tutor for a database language. Retrieved from http://ir.canterbury.ac.nz/handle/10092/327

Mitrovic, A., Suraweera, P., Martin, B., Zakharov, K., Milik, N., & Holland, J. (2006). Authoring Constraint-Based Tutors in ASPIRE. In M. Ikeda, K. D. Ashley, & T.-W. Chan (Eds.), *Intelligent Tutoring Systems* (pp. 41–50). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/11774303_5

Murray, T., Woolf, B., & Marshall, D. (2004). Lessons Learned from Authoring for Inquiry Learning: A Tale of Authoring Tool Evolution. In J. C. Lester, R. M. Vicari, & F. Paraguaçu (Eds.), *Intelligent Tutoring Systems* (pp. 197–206). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/978-3-540-30139-4_19

Nkambou, R., Mizoguchi, R., & Bourdeau, J. (2010). *Advances in Intelligent Tutoring Systems*. Springer Science & Business Media.

Nwana, H. S. (1990). Intelligent tutoring systems: an overview. *Artificial Intelligence Review*, *4*(4), 251–277. doi:10.1007/BF00168958

Ritter, S., Anderson, J. R., Koedinger, K. R., & Corbett, A. (2007). Cognitive Tutor: Applied research in mathematics education. *Psychonomic Bulletin & Review*, *14*(2),

249–255. doi:10.3758/BF03194060

Shute, V. J., & Wesley, J. (1993). Principles for evaluating intelligent tutoring systems. *Journal of Artificial Intelligence in Education*, *4*(2-3), 245–271.

Trausan-Matu, S., Boyer, K., Crosby, M., & Panourgia, K. (2014). *Intelligent Tutoring Systems: 12th International Conference, ITS 2014, Honolulu, HI, USA, June 5-9, 2014. Proceedings*. Springer International Publishing.

Vanlehn, K. (2011). The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems. *Educational Psychologist*, *46*, 197–221. doi:10.1080/00461520.2011.611369

Vanlehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., … Wintersgill, M. (2005). The Andes physics tutoring system: five years of evaluations. In *In Proceedings of the 12th international conference on Artificial Intelligence in Education* (pp. 678–685). IOS Press.

Woolf, B. P. (2008). *Building Intelligent Interactive Tutors: Student-centered strategies for revolutionizing e-learning* (1 edition.). Amsterdam ; Boston: Morgan Kaufmann.