2007-04

Secuenciamiento de actividades educativas orientado a la reutilización y la auto-organización en tutoría inteligente

Gutiérrez Santos, Sergio

UNIVERSIDAD CARLOS III DE MADRID

TESIS DOCTORAL

Secuenciamiento de actividades educativas orientado a la reutilización y la auto-organización en tutoría inteligente

Autor:
Sergio Gutiérrez Santos

Director:
Abelardo Pardo Sánchez

DEPARTAMENTO DE INGENIERÍA TELEMÁTICA

Leganés, Abril de 2007

UNIVERSITY CARLOS III OF MADRID

PhD Thesis

Sequencing of Learning Activities Oriented Towards Reuse and Auto-Organization for Intelligent Tutoring Systems

Author:
Sergio Gutiérrez Santos

Supervisor:
Abelardo Pardo Sánchez

DEPARTMENT OF TELEMATIC ENGINEERING

Leganés, April 2007

TESIS DOCTORAL

Secuenciamiento de actividades educativas orientado a la reutilización y la auto-organización en tutoría inteligente

Autor: Sergio Gutiérrez Santos

Director: Abelardo Pardo Sánchez

Firma del Tribunal Calificador:

Firma

Presidente:
Vocal:
Vocal:
Vocal:
Secretario:

Calificación:

Leganés, de Abril de 2007

A mi familia

Abstract

Three have been the main contributions of this thesis. First, a platform for the deployment of Intelligent Tutoring Systems (ITS) with a modular architecture has been designed. This platform, called SIT, focuses on the adaptation of the sequencing of learning content, not adaptation of the content itself. This separation permits specialization of pedagogical experts and encourages reuse of learning resources.

Second, a tool for the adaptation of the sequencing of learning units has been presented: Sequencing Graphs. It is a specialization of the finite automata paradigm, adapted for the specific needs of learning. Sequencing graphs focus on reuse, both of learning units and of adaptive sequencings definitions. They are hierarchical to prevent scalability problems. Two ITS have developed using sequencing graphs for SIT. Experimental results support the hypothesis that sequencing adaptation has a good influence on learning and that Sequencing Graphs are a useful tool to achieve this objective.

Finally, the thesis analyzes the current initiatives in the emerging field of swarm intelligence techniques in education. Apart of the theoretical overview, three results are presented: an experimental study performed on the Paraschool system, a system of pedagogical alarms based on learning pheromones on the same system, and a swarm paths information module for SIT. This module synthesizes the best results from swarm-based adaptation sequencing and collaborative filtering for providing an additional level of adaptation to the content sequencing in SIT

Acknowledgements

This thesis is not my work alone. Many have played a role in it over the years, and I owe them. May this page be a testimony of a debt that I will never be able to pay back.

I would like to thank Abelardo Pardo for his guidance, his support, the high scientific ethics that he has inspired in me and because it was always a pleasure to work with him. I am in debt as well with the members of the Gradient group and other colleagues from University Carlos III of Madrid, for their comments and suggestions on my research: Florina Almenárez, Celeste Campo, Raquel Crespo, José Pablo Escobedo, Luis de la Fuente, Rosa María García, Carlos García, JJ García, Pedro and Mario Muñoz, and others, specially Mari Carmen Fernández Panadero —for her everlasting effort in making me think as a scientist and not a programmer—, and Carlos Delgado Kloos —for trusting me since the beginning.

I would also like to thank all those researchers that have invited me to work with them. From all of them I learnt a lot, and all of them contributed to deepen and broaden at the same time the knowledge I have managed to acquire about my scientific field. First, *go raibh maith agat* to Joe Butler and his group at the IT Innovation Centre at Intel Ireland Ltd. (Leixlip, Ireland), for that great time in 2004 integrating IEEE LOM and Moodle. Also $\epsilon\upsilon\chi\alpha\rho\iota\sigma\tau o\ \ \pi o\lambda\upsilon$ to Simos Retalis and the Cosy Lab people at University of Piraeus (Piraeus, Greece), for those marvellous months in 2005 working on IMS SS and IMS LD. *Merci beaucoup* to Pierre Collet, and the Paraschool team —specially Grégory Valigiani— (Paris, France), for their invitation for working with them and the Paraschool system in 2006, and their help. I would like to thank Vanessa Fitzsimmons, John M. Kennedy, Eleni Koulikourdi, Isabel Pérez and Paco Rodríguez, as their generous help with logistic aspects in all those foreign countries was priceless. Finally, thanks to all other colleagues (specially Peter Brusilovsky, Ricardo Conejo and Vincent Wade) that with their insightful comments and suggestions helped me developing my ideas.

My gratitude is also with my master thesis students, that have played an important role in the development of this thesis, coding the tools I did not have the time to, and giving comments and suggestions that have been enlightening. Thanks to Pilar Prieto Linillos, Gemma Herrera Recio, José Manuel Durán Vivancos and Jaime Mayor Berzal.

Last, but not least, I am morally obligued to thank my family. Not only they have provided a lot of help and support on the difficult days that preceded the writing of this thesis, but they fill with significance and they give a sense to every day of my life.

To all of you, even those that my feeble mind (but never my heart) may have forgotten as I write these lines, *gracias*.

Dura lex, sed lex.

– Latin proverb

Nota sobre el uso del español y el inglés en la presente tesis

La presente tesis doctoral se presenta en la Universidad Carlos III de Madrid en el marco del programa de Doctorado en Tecnologías de las Comunicaciones, para optar al título de Doctor con mención *Doctor Europeus*. Para cumplir con la legislación española y las directivas europeas —por un lado— y con los reglamentos de la Universidad —por el otro— la tesis está escrita en inglés y en español.

La reglamentación de la Universidad Carlos III de Madrid[1] establece que las que opten a la mención *Doctor Europeus* deberán escribirse en español al menos en un 50 %. La legislación española[2] establece que, al menos, el resumen y las conclusiones deberán estar escritas en una lengua comunitaria distinta de las lenguas oficiales de España.

Con el objetivo de facilitar la difusión de la tesis se ha intentado escribir la mayor parte de la misma —especialmente las contribuciones principales— en inglés. En algún caso, se han repetido algo de información en diferentes partes del documento para dar mayor coherencia al texto y facilitar su lectura por personas que no lean español.

A note about the use of English and Spanish in this thesis

This PhD thesis is presented at University Carlos III of Madrid, in the Communication Technologies PhD programme, to obtain the degree of Doctor of Philosophy with *Doctor Europeus* mention. The thesis is written both in English and in Spanish in order to comply with Spanish laws and European directives as well as university regulations

University Carlos III of Madrid regulations[1] state that at least half of a thesis with *Doctor Europeus* mention must be written in Spanish. Spanish laws[2] state that —at least— the abstract and the conclusions have to be written in one of the languages of the EU, different from the official languages of Spain.

In order to make this work available to as many members of the scientific community as possible it has been written in English as far as legally possible. Specifically, the main contribution parts have been written in English. Some content has been repeated in different parts of the document for the sake of coherence and for helping non Spanish readers to follow the structure of the thesis.

[1] Reglamento de la Universidad Carlos III (*www.uc3m.es/uc3m/gral/TC/informaciongeneral/TESIS/tesisME.html*).
[2] Real Decreto 56/2005 de 21 de Enero.

Contents / Índice General

List of Figures / Índice de Figuras

List of Tables / Índice de Tablas

List of Patterns / Lista de Patrones

XML Excerpts / Fragmentos XML

Acronym index / Índice de siglas

ACO	Ant Colony Optimization	Optimización por colonia de hormigas
ADL	Advanced Distributed Learning initiative	La iniciativa “Aprendizaje avanzado distribuido”
AHAM	Adaptive Hypermedia Adaptive Model	Modelo de adaptación del hipermedia adaptativo
AICC	Aviation Industry CBT Committee	Comité de CBT de la industria de la aviación
BN	Bayesian Networks	Redes bayesianas
BNSeq	Bayesian Networks based sequencer	Secuenciador basado en redes bayesianas
CBT	Computer-based Training	Entrenamiento basado en ordenador
CoFIND	Collaborative Filtering in N Dimensions	Filtrado colaborativo en N dimensiones
IEEE	Institute of Electrical and Electronic Engineers	“Instituto de ingenieros eléctricos y electrónicos”
IMS	Originally, Instructional Management Systems	En origen, “Sistemas de gestión instruccional”
IMS-CP	IMS Content Package	Empaquetado de contenido de IMS
IMS-LD	IMS Learning Design	Diseño de aprendizaje de IMS
IMS-LIP	IMS Learner Information Profile	Perfil de información del discente IMS
IMS-QTI	IMS Question & Test Interoperability	Interoperabilidad de pregutas y test IMS
IMS-SS	IMS Simple Sequencing	Secuenciamiento simple de IMS
ITS	Intelligent Tutoring System(s)	Sistema(s) de tutoría inteligente
LCMS	Learning Content Management System	Sistema de gestión de contenido educativo
LLL	Life Long Learning	Aprendizaje durante toda la vida
LMS	Learning Management System	Sistema de gestión del aprendizaje
LN	Learning Networks	Redes de aprendizaje
LO	Learning Object	Objeto de aprendizaje
LOM	IEEE Learning Object Metadata	Metadatos de objetos de aprendizaje del IEEE
LTSC	IEEE Learning Technologies Standards Committee	Comité de estándares de tecnologías educativas del IEEE
OMG	Objects Management Group	Grupo de gestión de objetos
PE	Parametric Interface	Ejercicio paramétrico
SPI	Swarm Paths Information	Información de camino de enjambre
RLO	Reusable Learning Object	Objeto reutilizable de aprendizaje
RTE	Run-time Environment	Entorno de ejecución
SCO	Shareable Content Object	Objeto de contenido compartible
SCORM	SCO Runtime Model	Modelo de ejecución de SCOs
SG	Sequencing Graph	Grafo de secuenciamiento
SGed	Sequencing Graph Editor	Editor de grafos de secuenciamiento

SGS (& SGSeq)	Sequencing Graph based Sequencer	Secuenciador basado en grafos de secuenciamiento
SI	Swarm Intelligence	Inteligencia de enjambre
SIT	The System for Intelligent Tutoring platform	La plataforma “Sistema para tutoría inteligente”
UML	Unified Modelling Language	Lenguage unificado de modelado
UoL	Unit of Learning (in IMS-LD)	Unidad de aprendizaje (en IMS-LD)

List of Symbols / Lista de símbolos

Sequencing Graphs / Grafos de secuenciamiento (SG):

A_i	Set of actions on the i^{th} edge	Conjunto de acciones en la arista i-ésima
a_i^j	j^{th} action on the i^{th} edge	Acción j-ésima en la arista i-ésima
c_i	Condition on the i^{th} edge	Condición en la arista i-ésima
E	Set of edges in a SG	Conjunto de aristas de un SG
e_i	i^{th} edge in a SG	Arista i-ésima en un SG
I	Set of initialization actions in a SG	Conjunto de acciones de inicialización en un SG
N	Set of nodes in a SG	Conjunto de nodos en un SG
n_i	i^{th} node in a SG	Nodo i-ésimo en un SG
V	Set of variables in a SG	Conjunto de variables en un SG
η	Upwards interface of a SG	Interfaz hacia arriba de un SG
Π^*	The set of all the environments for all the users	El conjunto de todos los entornos de los usuarios
π	The environment of a user that traverses a SG	El entorno de un usuario que recorre el SG
Υ	Set of downwards interfaces in a SG, one for each container node	Conjunto de interfaces hacia abajo, uno por cada nodo contenedor
υ_i	Downwards interface for the i^{th} container node	Interfaz hacia abajo del nodo i-ésimo
Φ	Set of entry nodes in a SG	Conjunto de nodos entrada en un SG

Paraschool system / Sistema de Paraschool:

A	Normalized unadaptation factor normalized	Factor de inadecuación normalizado
C	Pedagogy-pheromones factor	Factor pedagogía-feromonas
I	Unadaptation factor	Factor de inadecuación
P	Passage valuation	Valoración de recorrido
W	Pedagogical weight	Peso pedagógico
ϕ^+	Amount of positive pheromones on an edge	Cantidad de feromonas positivas en una arista
ϕ^-	Amount of negative pheromones on an edge	Cantidad de feromonas positivas en una arista
ϕ^p	Amount of personal multiplicative pheromones on an edge	Cantidad de feromona personal multiplicativa en una arista
ϕ_i	Amount of pheromones (of type $+$, $-$ or p) on the i^{th} edge	Cantidad de feromonas (de tipo $+$, $-$ o p) en la arista i-ésima

1 Introducción

La verdadera ciencia enseña, por encima de todo, a dudar y a ser ignorante

– Miguel de Unamuno

Este capítulo proporciona un contexto al trabajo descrito en la presente tesis, introduciendo los conceptos básicos desde una visión panorámica y explicando cómo se relacionan las diferentes contribuciones. Esta introducción esquematiza los objetivos de la tesis y se cierra con una guía de lectura que analiza la estructura de la memoria relacionando los diferentes capítulos entre sí.

This chapter provides a context for the work described in the thesis, introducing the basic concepts from a broad perspective and explaining how the different contributions are linked together. This introduction summarizes the objectives of the thesis and provides a guide to its lecture presenting the different chapters and their relatioships.

1.1. Descripción de la tesis

Las tecnologías de la información están produciendo cambios significativos en la práctica educativa. Los métodos y técnicas para enseñar o aprender están estrechamente ligados a los dispositivos y recursos de que se dispone, y las nuevas tecnologías permiten extender los recursos disponibles y llevar a cabo tareas que, de otro modo, no son viables. Durante los últimos años, la tecnología ha empezado a influenciar de forma importante todo lo relacionado con el aprendizaje. Las aplicaciones son múltiples: aumentar la capacidad en entornos de comunicación a distancia, orientar al alumno en sus decisiones mediante técnicas de inteligencia artificial, desarrollar mecanismos de comunicación asíncrona eficientes y fiables, etc.

Existe ya una cantidad significativa de cursos a través de Internet a todos los niveles, desde educación obligatoria hasta reciclaje profesional y formación en el interior de las empresas. También

se han multiplicado las iniciativas que buscan complementar las clases presenciales tradicionales con actividades basadas en las nuevas tecnologías que proporcionan material en línea, abren nuevos canales de comunicación síncrona (vg. voz IP) y asíncrona (vg. correo y foros), facilitan la colaboración entre los alumnos, etc.

Uno de los efectos más trascendentes de las tecnologías de la información en el campo educativo tiene que ver con la capacidad de reutilizar y compartir. La tecnologías digitales han permitido e impulsado en gran medida la reutilización de material de aprendizaje. En un mundo digital, donde las copias de información son exactamente iguales que sus originales sin pérdida de calidad o de información a un coste prácticamente nulo, era inevitable que el mundo de la educación siguiera caminos similares a los del software [36, 39] o las artes [175], especialmente cuando el mundo académico siempre se ha basado más en compartir que en separar. Los docentes empezaron de forma espontánea a intercambiar material: transparencias, ejercicios en línea, etc. El coste de hacer esto es muchas veces inexistente y los beneficios son muchos.

Este proceso evolucionó hacia el concepto de objeto de aprendizaje reutilizable (*Reusable Learning Object, RLO* [177, 78]), el cual adapta en cierta forma al mundo del elearning el paradigma de la programación orientada a objetos. Es un concepto que tiene su origen en los procesos de instrucción del ejército norteamericano y se enmarca por tanto en el ámbito del diseño instruccional, una rama de la psicología del aprendizaje que sufrió un desarrollo muy fuerte a mediados del siglo XX gracias a los trabajos, entre otros, de Benjamin Bloom [13]. El objetivo de los RLO es tener diversas piezas de material educativo que se pueden combinar entre sí y reutilizarse en diferentes contextos. Sin embargo, la reutilización de objetos de aprendizaje plantea más problemas que la reutilización de objetos software. Por ejemplo, no está claro cómo recombinar estos objetos; la reutilización de objetos de programación está claramente definida por un interfaz, que suele ser sencillo. En el caso de los RLO, esta interfaz no es tan clara y la descripción de estos objetos es compleja [78].

En los últimos años el modelo parece estar en crisis. Algunos autores consideran que el modelo se ha vuelto demasiado elaborado para ser útil [103]. Otros afirman que el modelo parte de una base errónea al pensar que los objetos se pueden crear de forma descontextualizada, puesto que el contexto es una parte fundamental del aprendizaje [178]. Al mismo tiempo, iniciativas como IMS Learning Design modifican la idea de RLO y la orientan a enfoques pedagógicos complejos con varios participantes en colaboración.

La presente tesis se alinea parcialmente con estas críticas. Por un lado, emplea un enfoque pragmático respecto a los RLO y se centra en la utilización de recursos accesibles a través de la web. Por el otro, analiza el problema de la combinación y reordenamiento de estos recursos; modularidad, reutilización y combinación son los objetivos del paradigma de RLO.

Las dificultades de combinación y reordenamiento de recursos educativos constituyen el problema del secuenciamiento, que estructura la investigación de esta tesis. El problema de la adaptación del secuenciamiento es fácil de ilustrar mediante la metáfora de la lectura de un libro técnico. Cuando se consulta uno de estos libros, por lo general, éste no se lee desde el principio hasta el final de forma lineal, sino que la atención va saltando por el libro hacia adelante y hacia atrás, en función de nuestros intereses y necesidades, así como de nuestros conocimientos previos; es posible que al mismo tiempo se intercale la lectura de las páginas de este libro con páginas de otros libros que estén cerca y tengan relación con el problema a tratar. De esta forma, se está cambiando la *secuencia original* del libro (de capítulos, páginas o conceptos) por otra *secuencia* que

está *adaptada* al lector: diferentes lectores producen y obtienen diferentes secuencias.

La adaptación del proceso educativo a cada individuo tiene un efecto muy significativo sobre su aprendizaje. Bloom se refirió a ello como el "problema de las dos sigmas" [14], cuantificando las diferencias en los resultados de los alumnos que tenían acceso a una tutoría individualizada. La capacidad de adaptación es, por tanto, una característica fundamental en los sistemas de tutoría inteligente, los cuales intentan emular el comportamiento de un tutor humano. La presente tesis se centra en el problema de la adaptación del secuenciamiento. Propone un modelo basado en grafos para definir secuenciamientos adaptativos de material educativo. El modelo hace hincapié en la sencillez y en la reutilización, entendida desde dos puntos de vista: en primer lugar, la definición de un secuenciamiento adaptativo no debería perder su validez en el caso de que los recursos que secuencia cambien por otros equivalentes; en segundo lugar, la dificultad de utilizar un secuenciamiento creado en otro lugar o plataforma y su integración con otros secuenciamientos debería ser mínima. La primera interpretación es un resultado de la separación lógica entre el contenido educativo y su secuenciamiento, dos aspectos complementarios que pueden ser examinados en paralelo buscando soluciones óptimas para cada caso. La segunda es equivalente a minimizar los aspectos que deben ser retocados cuando se reutiliza un secuenciamiento ya definido: a medida que el número de modificaciones necesarias aumenta, el coste de crear desde cero el mismo material se hace comparable con el coste de reutilizarlo, lo cual no tiene sentido.

Este modelo, al que se ha llamado grafos de secuenciamiento (*Sequencing Graphs*, SG), ha sido probado en una plataforma que permite la creación de sistemas de tutoría inteligente orientados específicamente a la adaptación del secuenciamiento de una serie de actividades educativas. Estas actividades están por lo general asociadas a un recurso (vg. página web, imagen, vídeo, etc) el cual es accesible a través de una URL. Esta plataforma, llamada SIT, es otro de los resultados de esta tesis. La combinación de los SG con SIT ha permitido crear dos sistemas de tutoría en los que se adapta el secuenciamiento de un conjunto de ejercicios paramétricos (ver apéndice D). Los resultados indican que adaptar el secuenciamiento de una serie de ejercicios, definiendo una secuencia diferente para cada alumno en función de sus características, tiene un efecto positivo en el aprendizaje de los estudiantes, y que los SG son una manera efectiva de modelar este proceso.

Cada SG, al igual que cualquier otra estrategia instruccional, es diseñado por una persona con experiencia en el campo en concreto, con conocimiento del material de que se dispone para presentar a los alumnos y con experiencia sobre el aprendizaje de los conceptos ilustrados por dicho material. Esto tiene un aspecto positivo evidente, pero también una contrapartida negativa: el operador humano puede introducir problemas en el proceso de aprendizaje, bien por cometer errores o por cierto grado de desconocimiento (sobre el material, las estrategias adecuadas de secuenciamiento o los propios alumnos); y estos errores no pueden ser corregidos más que por otro operador equivalente. La constatación de estos errores ha llevado a la creación de herramientas de autor sencillas para la creación de los grafos y a una nueva línea de investigación que pretende hacer el sistema más robusto frente a esta clase de problemas.

Dado el carácter distribuido de las soluciones adoptadas se ha investigado el uso de técnicas de inteligencia de enjambre, con el objetivo de obtener sistemas educativos robustos y con capacidad de auto-organización [75, 76]. La inteligencia de enjambre es un tipo de inteligencia artificial que hace hincapié en la distribución, la robustez y la flexibilidad de los sistemas a través de la interacción directa o indirecta de una multitud de agentes. Esta interacción provoca la aparición de comportamientos del grupo cuya relación con los comportamiento de cada individuo no son evi-

dentes; las técnicas de inteligencia de enjambre reciben también el nombre de sistemas emergentes. En el caso de los sistemas de elearning estas técnicas tienen un interés especial. Dada la enorme complejidad de los sistemas de elearning, que suponen la interacción con un elemento complejo y en parte desconocido como es el cerebro humano —y sus procesos de aprendizaje—, un enfoque autónomo y emergente puede obtener mejores resultados que un sistema controlado y programado centralizadamente.

La presente tesis analiza los sistemas que utilizan técnicas emergentes, en particular el sistema diseñado para la empresa Paraschool por Valigiani [160], del que se presentan las aportaciones que se hicieron a su desarrollo. Las lecciones aprendidas de los diferentes sistemas se han cristalizado en la creación de un módulo para SIT que ofrece un grado adicional de adaptación del secuenciamiento, apoyado en técnicas emergentes y en el propio conocimiento meta-cognitivo de los estudiantes.

1.2. Objetivos de la tesis

Los objetivos de la presente tesis se enumeran a continuación:

- Diseñar una plataforma que permita crear sistemas de tutoría inteligente de forma eficiente en coste. Esta plataforma debe estar orientada al problema de la adaptación del secuenciamiento, pero debe ser independiente de aspectos como el modelo del usuario o las estrategias pedagógicas que se empleen en el proceso de tutoría; estos aspectos son responsabilidad de los diferentes tutores.

- Crear una herramienta o método que permita definir secuenciamientos adaptativos de material educativo, para su aplicación en la creación de uno o más tutores sobre la plataforma anterior. Esta herramienta debe ser sencilla de usar de forma que pueda ser utilizada por personas con un trasfondo tecnológico bajo. Crear una herramienta de autor basada en dicho método.

- Crear uno o más tutores usando la herramienta anterior utilizando ejercicios paramétricos y otros recursos. Verificar experimentalmente que el tutor así diseñado supone una mejora del proceso de aprendizaje.

- Estudiar la integración de la herramienta desarrollada con las especificaciones y estándares correspondientes, en particular IMS Simple Sequencing e IMS Learning Design, permitiendo —en su caso— la utilización de los resultados de la tesis en cualquier sistema que sea conforme a estas especificaciones.

- Adaptar la técnicas de inteligencia de enjambre para su aplicación en el campo del elearning, con el objetivo de conseguir sistemas de adaptación del secuenciamiento más robustos: capaces de detectar errores cometidos por los diseñadores humanos; y con capacidades de auto-organización, que les permitan adaptarse a cambios en su propio entorno (vg. la aparición de nuevo material educativo) o en los usuarios a lo largo del tiempo.

1.3. Estructura de la memoria

La parte I analiza el estado del arte, dando más importancia a los aspectos más novedosos, mientras que la parte II presenta los detalles de las aportaciones de la tesis. La parte III presenta las conclusiones del trabajo realizado y las líneas de investigación que quedan abiertas para el futuro. Finalmente, se incluyen diversos apéndices con información adicional que puede resultar relevante para determinados lectores. Estos apéndices presentan trabajo en desarrollo (como el apéndice C) o información de implementación.

Dividiendo la tesis temáticamente, los capítulos 2 y 6 tratan sobre sistemas de tutoría inteligente de una forma genérica. Los capítulos 3 y 7, y el apéndice A, tratan sobre los grafos de secuenciamiento, una especialización de los autómatas finitos orientada a su aplicación en tutoría inteligente, que permite definir secuenciamientos adaptativos de material educativo. El capítulo 7 y el apéndice C describen dos secuenciadores para conectar a SIT, la plataforma que se describe en el capítulo 6. Los sistemas complejos auto-organizativos y otros aspectos de la inteligencia de enjambre se tratan en los capítulos 5 y 9. Los interesados en las diferentes propuestas de estandarización en elearning encontrarán una panorámica en el capítulo 4 y un análisis de la relación entre los grafos de secuenciamiento e IMS-LD en el capítulo 8, así como un ejemplo de la aplicación del algoritmo de traducción en el apéndice B. El capítulo 11 describe las líneas de trabajo futuro. El capítulo 10 presenta las conclusiones de la tesis.

Parte I

Background / Estado del arte

2 Sistemas de tutoría inteligente

> It is reiterated that labor-saving devices should be possible in education. Such devices might well handle certain types of routine work even better than the teacher. They should save the teacher's time and energy from such routine, so that she (sic) may do more real teaching of ideal-developing and thought-stimulating type.
>
> – Sidney Pressey (1927)

Los sistemas de tutoría inteligente (ITS por sus siglas en inglés) proporcionan instrucción personalizada. Su objetivo es proporcionar los beneficios de la tutoría individualizada de forma eficiente. Estos sistemas, que suelen estar implementados en software, permiten que los discentes practiquen sus capacidades llevando a cabo tareas de forma interactiva y proporcionando una realimentación (vg. correcciones, pistas, etc) que se adapta en función de sus acciones y resultados. Los ITS analizan las acciones de cada usuario y desarrollan una estrategia instruccional adaptada a cada caso. Sin embargo, el gran problema de los ITS es el alto coste de producirlos. Este capítulo ofrece una visión general de los conceptos básicos de este campo, que será de ayuda en los capítulos 6 y 7. Después se presentan las dos principales alternativas para reducir el coste de producción de los ITS (herramientas de autor y arquitecturas modulares) junto con un análisis de los sistemas más característicos en cada caso.

Intelligent Tutoring Systems (ITS) provide individualized tutoring or instruction. An ITS is usually software-based. Its goal is to provide the benefits of one-on-one instruction automatically and cost effectively. They enable learners to practice their skills by carrying out tasks interactively, providing feedback (e.g. corrections, hints, etc) that is adapted to their actions and their results. ITS assess each learner's actions and tailor adapted instructional strategies. However, the main problem of ITS is the high cost of producing them. This chapter provides an overview of the main concepts of ITS, that will be of help on Chapters 6 and 7. Then, the main strategies for lowering the cost of creating ITS (authoring tools and modular architectures) are presented, along with an analysis of the most illustrative systems for each case.

2.1. Introducción

Algunos autores [136, 129] consideran que los orígenes de los sistemas de tutoría inteligente se pueden remontar a los trabajos de Sidney Pressey, que en la década de los 20 construyó una máquina mecánica que permitía enseñar a los estudiantes mediante la repetición de determinados ejercicios [138]. La máquina de Pressey (figura 2.1) permitía graduar el avance de los estudiantes, escondiendo la siguiente pregunta hasta que la pregunta actual era respondida correctamente. Aunque es cierto que las limitaciones del aparato son evidentes, Pressey consiguió que una máquina ayudara a aprender a sus estudiantes [137]; que lo consiguiera veinte años antes de que Mauchly y Eckert crearan el ENIAC [67, 156] sólo le da más mérito a su trabajo. Posteriormente, a partir del auge de la investigación en inteligencia artificial en los 60, la idea de usar los ordenadores como apoyo a la enseñanza ha sido una constante del devenir científico.

Un sistema de tutoría inteligente (*Intelligent Tutoring System, ITS*) intenta emular el comportamiento de un tutor humano, apoyando el proceso de aprendizaje de un discente al ofrecerle realimentación sobre sus errores o lagunas, y guiándole para conseguir sus objetivos educativos [152]. Cualquier proceso de aprendizaje se beneficia de un seguimiento individualizado y ajustado al estudiante [130]. Esta realidad se conoce como el *problema de las dos sigmas*, enunciado por primera vez por Bloom [14]: los estudiantes que tienen acceso a un tutor individual obtienen resultados alrededor de dos desviaciones típicas por encima de los que obtendrían mediante una instrucción clásica en grupo.

Para realizar una labor efectiva de tutoría, un ITS debe tener conocimiento de varios aspectos del proceso educativo. La concepción clásica describe estos sistemas como compuestos por tres partes (ver figura 2.2), que se corresponden con tres tipos de conocimiento: conocimiento del dominio (que debe incluir lo que el usuario pretende aprender), del estudiante (capacidades, conocimientos, metas, limitaciones, etc) y del diseño instruccional (cómo y cuándo presentar el material de que dispone) [73]. Algunos autores más modernos añaden un cuarto conocimiento como parte fundamental: el conocimiento de la comunicación [11], el cual abarca el conocimiento que debe tener un ITS sobre la interacción con los aprendientes tutelados. El funcionamiento de un ITS puede formularse de manera simplificada de la siguiente forma: el sistema presenta un problema al estudiante, que debe resolverlo; después se compara la solución entregada por el estudiante con la que el sistema ha hallado o tiene almacenada; en función de las similitudes o diferencias entre

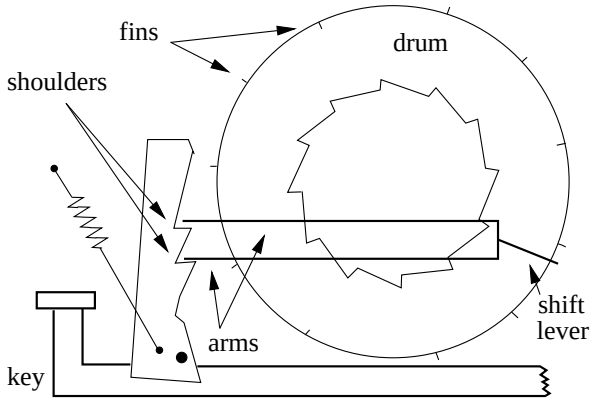


Figura 2.1: Dibujo esquemático de la máquina de S. Pressey (basado en [137])

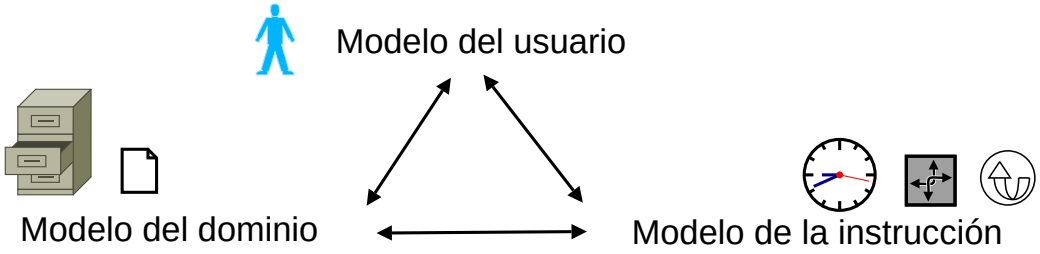


Figura 2.2: Los tres modelos presentes en un ITS

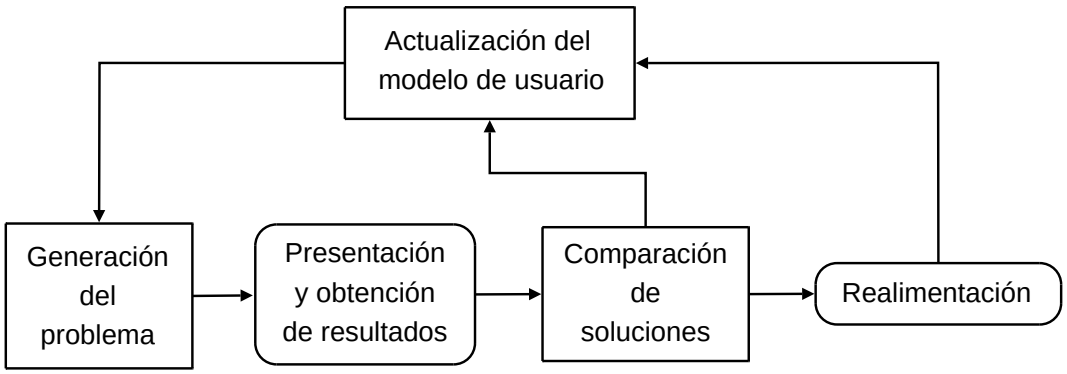


Figura 2.3: Ciclo típico de un ITS

ambas soluciones, se presenta algún tipo de realimentación (*feedback*) al usuario (por ejemplo, explicándole en qué y cómo ha fallado) y el proceso se repite. La información que el ITS posee sobre el estudiante se actualiza en cada iteración, así como los conocimientos del estudiante sobre el dominio en cuestión. Este ciclo se ilustra en la figura 2.3, donde los cuadros rectos representan operaciones que tienen lugar dentro del ITS y los cuadros redondeados denotan interacción con el usuario.

Estos fundamentos básicos llevan asentados desde la década de los 70 [73], aunque el término ITS no empieza a usarse de forma habitual hasta los años 80, siendo posiblemente Sleeman y Brown los primeros en utilizarlo [152]. En ámbitos concretos y muy específicos, como los simuladores de vuelo, los ITS se han introducido con notable éxito [110].

Durante los años 90, el desarrollo de los sistemas hipermedia se unió al trabajo realizado en ITS. Siguiendo un proceso lógico de integración —pues gran parte de los primeros sistemas hipermedia estaban orientados a la educación (algunos ejemplos son AHA! [37], InterBook [21] o CHEOPS [57]—, se evolucionó desde una primera generación de sistemas estáticos hacia sistemas dinámicos con capacidad de adaptación al usuario; esto fue a un tiempo causa y efecto de esta gradual fusión entre los campos del hipermedia educativo (*Educative Hypermedia, EH*) y de los ITS.

2.2. El problema del coste

Uno de los inconvenientes principales de los ITS es su complejidad. Para la construcción de un ITS es necesario un equipo con especialistas en diversos campos: programadores, expertos en los diferentes dominios de conocimiento, pedagogos, etc. Murray llegó a establecer la relación entre el tiempo de desarrollo de un ITS y el tiempo de instrucción real en aproximadamente 100 [124].

Para superar este problema se plantean dos enfoques complementarios. El primero es utilizar herramientas de autor que simplifiquen el proceso de desarrollo. El segundo es la definición de arquitecturas modulares que permitan la reutilización de componentes. La primera se presenta a continuación someramente y la segunda, más relevante para esta tesis, se estudia a continuación.

2.2.1. Herramientas de autor

Durante muchos años los sistemas de tutoría inteligente fueron una herramienta de laboratorio con una utilidad muy limitada. Al igual que ocurrió en otros muchos campos relacionados con la inteligencia artificial, esto se achacaba a la escasa capacidad de los ordenadores del momento; se esperaba que la evolución tecnológica y una potencia mayor de cálculo daría lugar de forma natural a máquinas que pensaran como los humanos. Sin embargo, a medida que parámetros como la velocidad de cálculo, la memoria o el coste de las máquinas dejaron de ser factores limitantes, se vio que el problema persistía [180]. Una de las causas es que la creación de un ITS supone mucho tiempo y esfuerzo[1]. Surgieron así diversas herramientas de autor que pretendían superar esa limitación. Sin embargo, las herramientas de autor comparten el mismo compromiso que cualquier ITS: cuanto más genéricas son, más difícil es que realicen una labor eficaz [122].

Los ejemplos de herramientas de autor son hoy innumerables, y cada una hace hincapié en diversos factores del proceso de autoría o del trasfondo del autor. VRCapture [107], que está orientada al dominio de la economía y a su uso en el ámbito de CIRCLE[2], hace hincapié en la utilización de técnicas similares a las de la programación orientada a objetos, combinadas con un esquema de reglas muy sencillo. REDEEM [111] incluye cinco herramientas de autor que permiten crear el material en diferentes niveles de granularidad (vg. secciones, páginas, etc), clasificar a los estudiantes y definir la actualización de reglas de adaptación en mitad de un curso. AthinaQTI [102] define un proceso de autoría sencillo basado en reglas combinadas orientado a la generación de elementos QTI [87]. CTAT [74] establece la diferencia entre tutores específicos de un problema (más fáciles de crear) y tutores cognitivos (más generales). La versión actual se centra en el primer tipo, e integra diversos módulos administrativos (vg. cuentas de usuario). CTAT incluye también editores orientados al desarrollo de modelos cognitivos, aunque esta funcionalidad está todavía en fase de investigación.

[1] Otra de las causas es que los tutores inteligentes no tienen en cuenta el aspecto social del aprendizaje [93]. Esto se trata en la tesis en los capítulos 5 y 9.

[2] Entornos de aprendizaje basado en contribuciones configurables, incrementales y reestructurables [106].

2.2.2. Arquitecturas modulares

En el campo de los ITS se produjo un salto cualitativo a finales de los años 90, con la explosión de la World Wide Web (WWW). El aumento exponencial de usuarios, y las necesidades de adaptación inherentes a una población tan vasta supuso un fuerte impulso para la investigación en sistemas hipermedia. También supuso un cambio sustancial para el campo de los ITS.

La mayor ventaja que supuso la llegada de la web para los ITS fue la capacidad de llegar a más usuarios. Un sistema web puede ser instalado y manejado en un solo sitio y ser utilizado por miles de usuarios de todo el mundo. Muchos sistemas dejaron de ser herramientas de laboratorio y pasaron a ser aplicaciones que se usaban fuera del ámbito reducido de su investigación, asistiendo a grandes poblaciones de estudiantes (bien en el ámbito académico, bien en el ámbito empresarial). Sin embargo, muchos de los primeros ITS basados en web evolucionaron directamente a partir de modelos anteriores; seguían siendo sistemas monolíticos que no tenían en cuenta la naturaleza distribuida de Internet, y esto suponía una restricción en sus capacidades [182]. El paradigma fue cambiando de forma gradual, apareciendo arquitecturas modulares y distribuidas. Este tipo de arquitecturas suponen una mayor estabilidad en el entorno de Internet, al tiempo que facilitan la evolución de un ITS o su reutilización para diferentes fines.

Durante los últimos años han aparecido numerosos modelos para describir la arquitectura de un ITS. Algunos autores intentan buscar patrones de diseño para buscar un cierto grado de consenso y reutilización [41, 8] pero los puntos en común son muy básicos y se limitan al uso de arquitecturas cliente servidor y/o la implementación de módulos intercambiables. No ha habido muchos esfuerzos para avanzar más allá en la reutilización de componentes entre sistemas de tutoría. Los ITS son escritos en multitud de lenguajes de programación diferentes incompatibles entre sí y, además, muchos de ellos sólo hacen la distinción entre los tres módulos a nivel teórico; es habitual que los tres componentes estén todos implementados en un mismo bloque monolítico, y que el sistema incluya también los propios contenidos.

Cliente-servidor

El paradigma cliente-servidor para aplicaciones distribuidas está presente en innumerables aplicaciones de todo tipo en Internet. En una arquitectura de este tipo, una parte de la aplicación (cliente) solicita la ejecución de una tarea, y ésta es ejecutada por la otra parte (servidor). El cliente se ejecuta en la máquina del usuario, mientras que el servidor se aloja en una máquina más potente donde puede realizar los mismos trabajos para muchos usuarios.

El cliente es responsable de la interacción con el usuario. Esto implica capturar las entradas del mismo (en ocasiones, también realizar un preprocesado de esos datos) y presentarle los resultados. El cliente envía los datos del usuario al servidor y recibe de éste los resultados. El servidor recibe los datos del alumno, los procesa y devuelve unos resultados al cliente. Por lo general, el servidor concentra la funcionalidad más importante (modelo del dominio, del usuario y de las estrategias instruccionales), y es al mismo tiempo el lugar donde se almacena el contenido que debe mostrarse al alumno.

En el caso de los ITS, la separación cliente-servidor abre la puerta a que el servidor utilice la información que almacena sobre todos los clientes para proporcionarles algún tipo de beneficio

adicional, como mayor capacidad de adaptación o conciencia relativa a otros usuarios [19]. El módulo de caminos del enjambre de SIT, que se explica en la sección 9.3.2, es un ejemplo del primer caso.

Modelos intercambiables

La división teórica de un ITS en tres dominios [73] puede trasladarse a una separación real en la implementación, donde cada dominio está implementado en una clase o en un módulo diferente. Esto facilita el intercambio de módulos equivalentes. En [182] se muestra una arquitectura para un ITS en que el dominio del conocimiento está implementado como un módulo basado en base de datos, que puede ser enchufado (*plugged*) al sistema en cualquier momento. Las otras partes del sistema (el modelo del usuario y el modelo de inferencia) están especificadas de forma genérica, y se instancian en función de determinados datos que proporcionan los diferentes módulos del dominio del conocimiento. Esto permite que el mismo sistema ofrezca tutoría sobre dominios diferentes.

Otro enfoque relevante para el ámbito de esta tesis es el descrito en [54], donde se hace uso de sistemas expertos para la creación de tutores inteligentes: sistemas expertos en diferentes temas producen diferentes tutores de dominio específico. El interfaz de SIT (presentado en el capítulo 6) se inspira parcialmente en los interfaces descritos en [54].

Por último, aunque no está orientado a tutoría inteligente, el sistema TANGOW[3] [26, 27] hace una distinción interesante entre control de tareas y generación de contenido. Su arquitectura es de tipo cliente-servidor. En la parte servidora hay dos módulos con funciones complementarias. El primero es el manejador (*manager*) de tareas se encarga de llevar el control del proceso de aprendizaje de un alumno durante la sesión (hay un manejador por cada alumno); si el alumno sigue varios cursos, tendrá un manejador de tareas diferente en cada uno de ellos. El segundo es el manejador de procesos, que es único y se encarga de recibir las peticiones de los usuarios y reenviarlas —junto los datos recogidos, de haberlos— al correspondiente manejador de tareas. TANGOW también tiene un módulo de presentación, cuya responsabilidad es producir las páginas web que ve el usuario a partir de la información que obtiene del manejador de tareas. Una arquitectura similar, SKILL, se presenta en [126] con una mayor orientación a procesos de tutoría añadiendo un espacio personal de usuario. Sin embargo, la integración entre los componentes de SKILL es alta, por lo que su capacidad de reutilización de módulos es menor que en los casos anteriores.

[3]Task-based Adaptive learNer Guidance On the Web.

3 Secuenciamiento

> . . . the reason you need all those heavy-handed instructional methods is because you're trying to teach people something they don't want to learn. When they want to learn it, if you create the right intellectual environment, they learn it quickly and easily.
>
> – Seymour Papert [132]

Este capítulo presenta varios enfoques al problema del secuenciamiento adaptativo, que son representativos de las iniciativas principales para definir secuencias en unidades educativas que se pueden adaptar a usuarios con capacidades distintas y/o metas diferentes. Todas tienen un trasfondo gráfico, pero hacen hincapié en asuntos diferentes de la cuestión: algunas se centran en usar herramientas muy conocidas como UML mientras que otras prefieren utilizar metáforas basadas en grafos sencillos.

This chapter presents several approaches to the problem of adaptive sequencing. They are representative of the main initiatives for defining sequencings of learning units that can be adapted to users with different capabilities and/or needs. All of them have some graphical background, but they focus on different issues: some focus on using well-know tools like UML while other focus on using simple graph metaphors.

3.1. La importancia de adaptar

Una de las ventajas de los sistemas educativos basados en web es la posibilidad de ofrecer una gran cantidad de recursos al usuario. A cambio, se corre el riesgo de quedar "perdido en el hiper-espacio" [53]. Para evitar que esto ocurra, es conveniente ajustar el material educativo mostrado a las necesidades de cada estudiante. De esta forma, no se le muestran los mismos contenidos a todo

el mundo, sino que el material es ajustado [89] y se crea un entorno personal apropiado para cada usuario y sus capacidades [35]. El compromiso se produce entonces, a la hora de crear un sistema de elearning adaptativo, entre la necesidad de adaptar y la necesidad de hacer un sistema lo más general posible. Cuanto más general es un sistema y su ámbito de actuación, más difícil es conseguir su adaptación a los usuarios. Por el contrario, si se reduce el ámbito a algo muy específico se obtienen buenos resultados con más facilidad [110].

A la hora de adaptar material educativo al estudiante, se piensa en adaptar diferentes aspectos [133, 154]: el contenido, el método de enseñanza, el estilo de enseñanza, la selección de medios, la secuencia de contenidos, las limitaciones temporales, la presentación de ayuda, las anotaciones, la ocultación de información, etc. Todas ellas se resumen en dos: adaptación del contenido y adaptación del secuenciamiento. Ambos procesos están relacionados y son igualmente importantes para fomentar el aprendizaje y hacerlo más efectivo.

El modelado de usuario

Los sistemas que buscan adaptar el contenido o la secuencia de actividades que se ofrecen a un alumno se apoyan en algún tipo de modelado de los estudiantes. Esta es una característica común de la mayor parte de los sistemas adaptativos; y para algunos autores constituye su punto débil [157]. Va mas allá de la intención de esta tesis tratar en profundidad el problema del modelado de los usuarios, pero un lector interesado encontrará una visión general del asunto en [96].

3.1.1. Adaptación del contenido

La mayor parte de los esfuerzos de los últimos años se han centrado en adaptar el contenido al estudiante, a sus capacidades y sus metas. (Se considera aquí que la adaptación de contenido educativo incluye procesos como la creación de anotaciones y la ocultación de información). En un sistema hipermedia educativo moderno, la adaptación del contenido puede tomar varias formas incluyendo, pero sin limitarse a:

Explicaciones comparativas: Los mismos conceptos se explican mediante comparaciones con conceptos que el estudiante ya conoce. Estas comparaciones serán diferentes para cada estudiante, en función de sus conocimientos. Por ejemplo, al tratar de explicar cómo funciona un sistema de control de segundo orden, algunos alumnos lo entenderán mejor si se realiza la comparación usando un circuito electrónico, mientras que otros preferirán una metáfora que emplee elementos físicos, como una masa, un amortiguador y un muelle [58].

Texto condicional: La explicación de los conceptos presenta más o menos detalle en función de determinadas *condiciones*, que están relacionadas con el conocimiento del usuario. Por ejemplo, al explicar el bucle “while” de Java a un estudiante, algunos recibirán un texto adicional explicando *qué es un bucle*, mientras que otros recibirán sólo la sintaxis [37]. Este tipo de técnicas reciben a veces el nombre de *andamio* o *tramoya* (*scaffolding*) [118].

Diferentes lenguas: En función del estudiante, la explicación se recibe en una lengua u otra [6].

El objetivo de esta tesis no es ahondar en este tema, puesto que el autor considera que ya hay suficiente esfuerzo dedicado a ello. Para un análisis más serio que esta somera introducción, se puede consultar [21].

3.1.2. Adaptación del secuenciamiento

La cuestión del secuenciamiento se remonta en el tiempo casi hasta los orígenes de los sistemas de tutoría inteligente. Algunos de los sistemas de tutoría inteligente más antiguos [9] ya eran capaces de manipular el orden de algunas preguntas o cuestiones de forma limitada. Otros sistemas posteriores presentaban capacidades similares, empezando a referirse la literatura a ello como secuenciamiento de tareas (*task sequencing*) [115, 142].

El siguiente paso fue lograr la capacidad de secuenciar *lecciones*, agrupaciones relativamente grandes de material educativo, que comprendían presentación del material y preguntas sobre el mismo [25]; algunos sistemas también complementaban las lecciones con ejemplos destinados a favorecer el aprendizaje [95].

Se llega así a lo que Peter Brusilovsky llama secuenciamiento de curso (*course sequencing*), que es en cierto modo el germen del secuenciamiento de actividades educativas. La idea detrás del secuenciamiento de curso es la de generar un curso individualizado para cada estudiante seleccionando la *operación de enseñanza* óptima en cada momento. En [22], Brusilovsky habla de cuatro tipos de operaciones en los sistemas basados en secuenciamiento de curso: presentación, ejemplo, pregunta y problema. Conociendo el grado de conocimiento de un estudiante, sus metas esperadas de aprendizaje y la estrategia de aprendizaje a emplear, un sistema de tutoría inteligente debe elegir cuál es la operación adecuada en cada momento.

3.2. Alternativas actuales

Es a partir de la integración de los sistemas hipermedia con los ITS, y especialmente con la llegada de la web, cuando el problema de la adaptación del secuenciamiento de información pasa a ser un tema fundamental en el estudio de los ITS. En un sistema web, con la multitud de enlaces y de recursos que se muestran al usuario, es fácil perderse [53]. Para evitar que esto ocurra, hay que buscar fórmulas para guiar al usuario y ayudarle en su navegación. En el marco de una herramienta de elearning, esta guía debería estar adaptada a sus capacidades y a sus necesidades, de forma que su proceso de aprendizaje sea óptimo.

Actualmente muchas de las aplicaciones de elearning comerciales no tienen apenas soporte a la creación de secuenciamientos adaptados a cada estudiante. Por el contrario, a menudo se basan en secuenciamientos estáticos iguales para todos. Dentro del mundo académico hay varias iniciativas que crean un marco para la creación de secuenciamientos dinámicos, por lo general basándose en metáforas gráficas. Las dos tendencias más importantes son el uso de diversas formas de grafos dirigidos y, en particular, del lenguaje UML.

3.2.1. UML

El Lenguaje de modelado unificado (*Unified Modelling Language*), referido habitualmente como UML, es un lenguaje gráfico para modelar sistemas de software. A pesar de que ha habido otros lenguajes con el mismo objetivo, el UML ha sido el que ha cosechado un mayor éxito, siendo el más usado hoy en día. El OMG (*Object Management Group*) lo apoya y lo ha convertido en unos de sus estándares. Las especificaciones están disponibles en su página web (www.omg.org). UML se usa para visualizar, especificar, construir y documentar un sistema de software.

Desde la primera versión del estándar, se han añadido múltiples elementos al UML para representar diferentes entidades. Esto ha producido críticas, la mayoría de las cuales alegan que el estándar se ha vuelto excesivamente vasto y difícilmente abarcable. Por otra parte, su amplio ámbito y la posibilidad de extenderlo a través del uso de perfiles y estereotipos lo ha hecho muy popular. Hoy en día se usa el UML para modelar nuevos tipos de sistemas basados en software, como por ejemplo procesos de negocios, ingeniería de sistemas o estructuras de organización.

Algunos investigadores del campo del elearning con un trasfondo de diseño de software han usado UML para describir diversas herramientas de elearning, o para modelar alguno de sus procesos. En esta sección nos centraremos en aquellos que han usado UML para definir estrategias de secuenciamiento aplicadas al contenido de aprendizaje. Se presentan dos enfoques distintos (aunque similares) en esta sección. El lector interesado puede encontrar aún otro enfoque que usa diagramas de estado UML para las definiciones de las diferentes secuencias en hipertexto y aplicaciones basadas en web en [38].

Una ventaja de usar UML para describir sistemas educativos basados en web es que es un lenguaje bien conocido para una gran parte de los ingenieros de software, por lo que no necesitan aprender ninguna herramienta adicional. La principal desventaja tiene que ver con los profesores o diseñadores, quienes no tienen un trasfondo tecnológico. UML es un lenguaje complejo y bastante específico; si no conocen UML de antemano, la perspectiva de intentar aprenderlo será posiblemente demasiado esfuerzo para una mayoría. Además, UML es un lenguaje diseñado para ingeniería del software. A pesar de su amplio ámbito y sus capacidades de expansión, puede no ser la mejor opción para diseñar sistemas cuyo objetivo sea el aprendizaje.

CADMOS-D

CADMOS [140] es una metodología para el desarrollo de sistemas de instrucción principalmente, que se centra en el desarrollo de aplicaciones web educativas. El método orientado a la fase de diseño de CADMOS es CADMOS-D [131]. Otras fases incluyen la captura de requisitos, la implementación y la evaluación.

CADMOS-D ofrece un modelo de proceso, enfocado a las diferentes etapas y sus relaciones temporales y secuenciales; y un modelo de producto, que se centra en los resultados de cada etapa y las dependencias entre éstos. El modelo propuesto está definido como un perfil UML, que está especificado por: una extensión de elementos del UML básico, la definición de semántica adicional para los nuevos elementos y la definición de restricciones sintácticas para la interconexión de los mismos. El modelo de diseño se divide en tres sub-modelos: conceptual, de navegación y de presentación. Esta separación simplifica el proceso de diseño. También permite reutilizar el tra-

bajo: por ejemplo, no se necesita cambiar el modelo del contenido para definir una estrategia de secuenciación diferente. Dentro del objetivo de este capítulo, el más importante es el modelo de navegación, que se analiza más detalladamente a continuación.

El diseño de navegación sigue al diseño conceptual, el cual define las actividades de aprendizaje que los estudiantes tienen que realizar así como las relaciones entre ellas. En el modelo conceptual se describen los asuntos pedagógicos habituales, como las dependencias entre los conceptos, los objetivos de aprendizaje y las jerarquías de conocimiento. Los recursos que están asociados con las actividades de aprendizaje, si hay, también se especifican en el diseño. El modelo conceptual define un esquema de secuenciamiento por defecto (de forma similar a IMS-CP [85]): los conceptos derivados deben ser cubiertos después de los conceptos principales, y los del mismo nivel deben cubrirse de izquierda a derecha según el mapa UML. Esto es —en definitiva— una secuencia lineal y estática, sin adaptación al usuario.

Una vez que el modelo conceptual está definido, el modelo de navegación realiza la correspondencia entre los conceptos (y recursos, en su caso) y las páginas (nodos). También lo hace para relaciones y enlaces. Adicionalmente, el modelo de navegación se divide en un modelo estructural y un modelo de comportamiento.

El modelo estructural de navegación define la estructura del sistema hipermedia y especifica las páginas web reales y los recursos asociados a ellas. La estructura está compuesta de: contenido (*content*, el contenedor superior), nodos contenedores (*container nodes*, que componen la jerarquía del contenido de aprendizaje —capítulos, etc—), nodos de contenido (*content nodes*, las páginas que tienen el contenido de aprendizaje), fragmentos (*fragments*, similares a los nodos de contenido pero sólo se asocian a los recursos en un nivel conceptual, mientras que los nodos de contenido se asocian a recursos y conceptos), elementos de acceso (*access elements*, enlaces para facilitar la navegación en el hipertexto como índices o recorridos guiados) y enlaces (*links*, enlaces asociativos — ver [144] — que implementan las relaciones del modelo conceptual; no son enlaces estructurales que permitan la transición de una pagina web a otra).

La estructura del modelo conceptual es seguida por el modelo estructural de navegación, por lo que éste define la misma secuencia por defecto: un recorrido preseleccionado por los diferentes contenedores, nodos de contenido y fragmentos (principales antes que secundarios, los del mismo nivel de izquierda a derecha). Esta secuencia por defecto se altera en el modelo de comportamiento de navegación.

El modelo de comportamiento de navegación define —en términos de la navegación— el comportamiento del sistema en tiempo de ejecución. Las reglas y las transiciones opcionales se crean en esta fase para permitir diferentes secuencias, adaptadas a las características del usuario.

Los contenedores y los nodos de contenedor del modelo estructural de navegación se convierten en estados en el modelo de comportamiento de navegación. De esa forma se conserva la jerarquía: la misma que se define para los elementos de navegación en el primero se conserva para los estados en éste último. Los enlaces en el modelo estructural de navegación corresponden a enlaces y eventos que los desencadenan en el modelo de comportamiento. Además, condiciones de guarda en la transición permiten definir transiciones alternativas de navegación. Éstas son activadas o desactivadas dependiendo de las características del alumno y, por lo tanto, adaptan la secuencia de estados al estudiante. Los nodos compuestos y de contenido tienen un atributo llamado “incluido” (*included*), para especificar si han de incluirse o no (junto a sus descendientes, en su caso) en el

hipertexto.

Una vez que el modelo de navegación completo está definido, la ultima etapa es la definición del modelo de presentación. La transición es sencilla. Los elementos en el modelo de presentación (básicamente elementos HTML y CSS [167, 168]) se asocian con los nodos del modelo de navegación. El modelo de presentación se divide también en dos: el modelo estructural de presentación que tiene que ver con los elementos web como las ventanas, los marcos, etc; y el modelo de interfaz del usuario que se ocupa de los colores, estilos, etc.

El modelo de usuario consiste en dos partes distintas: el revestimiento (*overlay*, relativo al dominio) y el estereotipo (*stereotype*, relativo al usuario). El modelo de revestimiento es la parte del modelo de usuario específica del dominio: define el nivel de conocimiento del aprendiente respecto a los conceptos específicos cubiertos en el material de aprendizaje. Como resultado de la interacción entre el discente y el contenido de aprendizaje, el estado de este modelo se actualiza constantemente: el revestimiento representa la consciencia del sistema respecto al nivel de conocimiento del dominio tal y como se describe en el modelo conceptual. Los elementos del revestimiento se llaman esquemas de usuario (*userSchemes*); sólo hay un elemento esquema para cada tipo de modelo conceptual. El estereotipo define elementos que se usan para representar el perfil de conocimiento predefinido para el usuario. Describe el conocimiento de un dominio en particular (vg. novato, intermedio, experto, etc) o correspondiente a las preferencias del usuario o a su estilo de aprendizaje (vg. de arriba abajo, de abajo arriba, etc). Los elementos de este sub-modelo reciben el nombre de Usuario (*user*).

UML-Guide

UML-Guide [43, 28] usa diagramas de estado UML para modelar la navegación del usuario a través de un hipertexto o, de manera más general, un sistema hipermedia. El propósito es construir sistemas hipermedia adaptativos (no sólo educacionales, sino en un sentido general). Los diagramas de estado se usan para visualizar mapas de navegación, en los que los nodos representan documentos y las flechas (transiciones) representan enlaces entre ellos.

Los eventos producen transiciones en una máquina de estados. Incluyen eventos generados por el usuario y por el sistema (vg. eventos asociados al tiempo). Pueden usarse restricciones para limitar las transiciones dependiendo de ciertas reglas. Cuando se crean transiciones, y antes de entrar en el nuevo estado, se pueden realizar diversas acciones. Además, se pueden realizar acciones *mientras* se produce la transición (acciones de efectos colaterales). Estas acciones procesan parámetros (por ejemplo, activando restricciones) y actualizan el modelo del usuario.

En este enfoque, el modelo del usuario se representa con un diagrama de clase. La estructura es similar a la de AHAM [18]. El modelo de usuario incorpora diversas características del usuario, así como datos sobre el uso de la aplicación hipermedia. En los diagramas de clase, las diferentes clases representan el conocimiento del usuario, sus preferencias, objetivos, trasfondo, etc. Solamente el primero es obligatorio. Los otros se pueden añadir si se considera necesario.

El modelo de usuario tiene operaciones para leer el estado actual de las características del usuario, así como para actualizar sus valores. Esto es una diferencia interesante respecto a otros enfoques, en los que el modelo del usuario sólo se ve como una fuente de datos y las operaciones forman parte de otro aspecto del modelo.

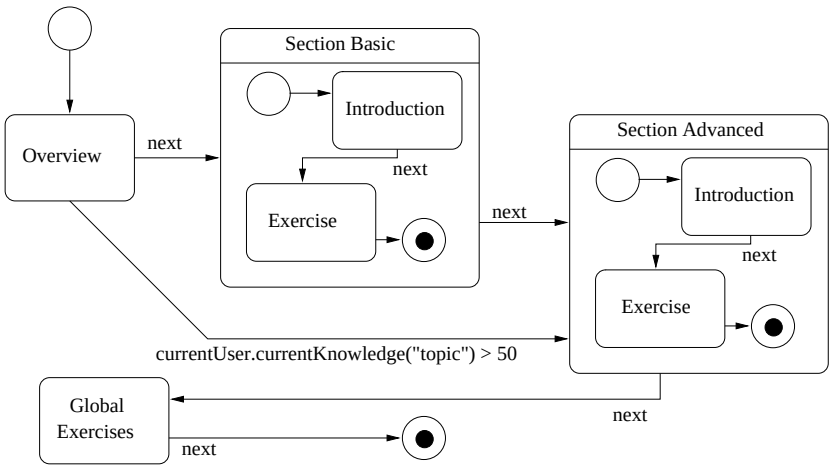


Figura 3.1: Ejemplo de secuenciamiento en UML-Guide

El diagrama de estado UML se puede exportar a XMI (*XML Metadata Interchange*). XMI es el estándar de OMG para intercambiar metadatos vía XML. Una vez que el diagrama está en XML, se usan diferentes técnicas para producir un mapa de secuenciamiento a partir de él.

Una descripción detallada del algoritmo esta más allá del objetivo de este capítulo. El lector interesado puede encontrar esta información en [43], así como también un enfoque similar a UML-Guide pero más orientado a lenguajes de modelado pedagógico en [113].

3.2.2. Grafos dirigidos

Un grafo se describe de manera informal como un conjunto de nodos (o vértices) unidos mediante una serie de aristas (o arcos). Los arcos conectan los nodos por parejas. Un grafo puede ser dirigido o no; en un grafo dirigido[1] una arista que conecta el nodo A con el nodo B no conecta el nodo B con el nodo A.

Múltiples problemas de interés práctico, tanto del mundo físico como del cibernético, pueden representarse como grafos de diferentes tipos. La estructura de enlaces de una página web puede verse como un grafo dirigido, en que los nodos representan las páginas web y una arista conecta el nodo A al nodo B si y sólo si A contiene un enlace web a B. Un enfoque similar puede aplicarse a problemas de viajes, biología molecular, evolución genética, diseño de circuitos integrados, redes de comunicaciones y múltiples otros campos. Se puede enriquecer la semántica de un grafo atribuyendo propiedades especiales a sus nodos y aristas, o añadiendo restricciones (vg. pesos, condiciones, etc). Algunos grafos han adquirido una importancia tal que son conocidos por un nombre específico, como es el caso de los autómatas finitos [101][2] o las redes de Petri [135].

Pueden usarse grafos para definir secuenciamientos de material educativo. A continuación se describen dos enfoques que se basan en un grafo con algo de semántica adicional especialmente diseñado para la adaptación del secuenciamiento.

[1] Un nodo dirigido es llamado a veces un *digrafo* (*digraph*).

[2] Los grafos de secuenciamiento, explicados en detalle en el capítulo 7.2, son una especialización de los autómatas finitos.

AHA!

AHA! [155, 17] es un marco de trabajo (*framework*) adaptativo, basado en web y de propósito general, orientado a la construcción de sistemas hipermedia. Hace hincapié en la simplicidad y permite múltiples tipos de reglas de adaptación, al mismo tiempo que intenta no promover ninguna en particular, permitiendo así una total libertad de presentación para el diseñador de sistemas. Aunque puede usarse para cualquier tipo de sistema, la mayoría de sus usos hasta ahora han sido diferentes clases de esfuerzos educativos [143, 30]. AHA! está fuertemente basado en el AHAM [18].

El énfasis está más en la adaptación del contenido que en la adaptación del secuenciamiento. De hecho, no hay un concepto de adaptación del secuenciamiento como tal en AHA! Sólo hay un cierto nivel de adaptación de la navegación mediante el uso de técnicas de ocultación de enlaces (*link hiding*) y anotación de enlaces (*link annotation*) [20], de acuerdo con unas reglas definidas por el diseñador; la aplicación de estas técnicas tiene una influencia indirecta en la secuencia de documentos a los que el usuario accede. Desde el punto de vista de esta tesis, el aspecto más relevante es la relación entre los conceptos y las reglas, y cómo se define mediante las herramientas de autor disponibles.

Existen dos herramientas de autor para AHA!: el editor de conceptos y el editor de grafos. La primera es una herramienta de bajo nivel con la que el autor define, usando una plantilla, un conjunto predefinido de atributos y reglas de adaptación para cada nuevo concepto creado. Existen dos tipos de reglas de adaptación: reglas de generación y reglas de requisito. El editor de grafos, por su parte, utiliza una metáfora gráfica (similar a la mostrada en el apéndice A). En este caso, los nodos en el grafo son conceptos y las aristas representan diferentes tipos de reglas de dependencia entre los conceptos: propagación de conocimiento, prerrequisitos de conocimiento, etc. Un proceso automático trasforma estas construcciones de alto nivel en reglas de adaptación de bajo nivel. De esta forma, los resultados del editor de grafos pueden ser refinados posteriormente por el editor de conceptos.

Las reglas y las relaciones se traducen después al modelo combinado de dominio y adaptación de AHA! [155]. Esto habilita o deshabilita algunas transiciones en las páginas en función de las reglas definidas en el paso anterior. Aunque el foco está aquí claramente en los conceptos y no en el secuenciamiento de material educativo, este proceso guarda cierta semejanza con la metodología CADMOS-D tratada en la sección 3.2.1.

Grafos estocásticos

En un grafo estocástico cada nodo está asociado a una unidad de aprendizaje y cada arista tiene asociada una probabilidad. Estas probabilidades determinan cuál será la siguiente unidad que será entregada al alumno. Las aristas con probabilidades más altas será elegidas más a menudo, y sus correspondientes unidades de aprendizaje serán habitualmente las entregadas. Estas probabilidades son diseñadas por un equipo pedagógico, que también diseña el resto del grafo. Este es el punto de partida para el trabajo descrito en [148].

El uso de probabilidades supone una gran diferencia frente a los enfoques presentados con anterioridad, en que las decisiones con respecto a la elección de la siguiente unidad de aprendizaje se tomaban de forma determinista (parcialmente, al menos) basándose en condiciones, restricciones

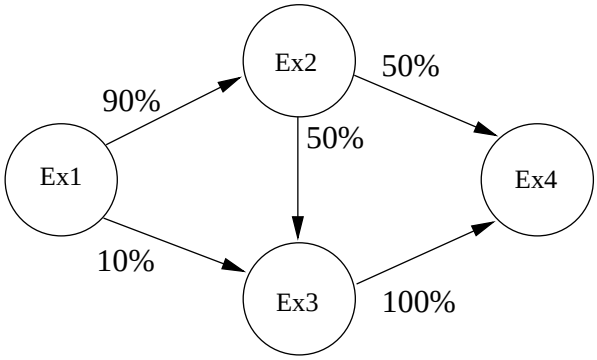


Figura 3.2: Ejemplo de secuenciamiento estocástico

y reglas.

Las aristas que conectan nodos equivalen a restricciones en el secuenciamiento de unidades de aprendizaje. Las probabilidades asociadas a dichas aristas determinan cuáles serán escogidas después de finalizar cada nodo. Tomando en consideración ambas cosas (restricciones y probabilidades), el secuenciamiento se diseña estocásticamente.

El diseño probabilístico paso a paso tiene ventajas e inconvenientes. La ventaja es clara: establecer probabilidades (o, visto de otra forma, importancias relativas) es un enfoque más intuitivo para personas (es decir, el equipo pedagógico) sin un trasfondo matemático o ingenieril importante. Sin embargo, a medio y largo plazo este método tiene el inconveniente de que requiere un análisis profundo para establecer secuencias adecuadas[3]. En caso contrario, pueden surgir múltiples errores en el proceso [65].

Para esta tesis, la importancia de los grafos estocásticos radica especialmente en que facilitan la utilización de técnicas de enjambre para la auto-corrección de errores. Esto se explica en las secciones 5.3.1 y 9.2.

[3]De hecho, un grafo como los presentados no es más que una cadena de Markov. Las cadenas de Markov se han utilizado para el análisis y diseño de redes de Petri estocásticas [10].

4 Estándares en elearning

> The nice thing about standards is that there are so many of them to choose from.
>
> – Andrew Tannenbaum

 La evolución y la ubicua presencia del elearning requieren que su material sea intercambiable entre diferentes plataformas. Especificaciones y estándares internacionales definen un marco común en el que el software educativo es adaptable, intercambiable y reutilizable. Pero el elearning es tan vasto y trata aspectos tan variados que dichos estándares tienen que crearse de forma gradual. Este capítulo proporciona una visión general del asunto y analiza dos especificaciones internacionales que están relacionadas con el problema del secuenciamiento: IMS-SS e IMS-LD.

 The evolution and widespread presence of elearning requires its material to be interoperable among different platforms. International specifications and standards define a common framework to make elearning software adaptable, interoperable and reusable. But elearning is so wide and touches so many aspects that the task of defining such standards is being done gradually. This chapter provides an overview of the topic and analyzes two international specifications that are related to the sequencing problem: IMS-SS and IMS-LD.

4.1. La importancia de reutilizar

El mundo del elearning ha evolucionado rápidamente en los últimos años. Las herramientas actuales son capaces de integrar tareas no sólo académicas sino también administrativas en las instituciones de enseñanza. Los sistemas de gestión de contenido educativo (*Learning Content Managament Systems, LCMS*) son capaces de hacerse cargo de una serie de tareas que abarcan desde el pago de matrículas hasta enfoques pedagógicos personalizados y actividades colaborativas.

Una presencia tan extendida de las actividades de aprendizaje hace recomendable, si no necesario, que el material educativo sea intercambiable entre diferentes plataformas. Un curso que sólo puede ser usado en una plataforma está condenado a que sólo un pequeño porcentaje de sus usuarios potenciales se beneficien de su uso. Adicionalmente, en este nuevo escenario el material de enseñanza incluye una variedad mucho más extensa de recursos y, por tanto, el proceso de producción ha incrementado su complejidad. En conclusión, las plataformas de elearning deben garantizar interoperabilidad entre ellas: ofrecer la posibilidad de reutilizar recursos (llamados en algunos contextos objetos de aprendizaje[1] —*Reusable Learning Objects*, *RLO*—) de otros cursos aumenta mucho la productividad del proceso de creación.

La respuesta a estas necesidades viene dada por los estándares internacionales que definen un marco común para hacer el software educativo adaptable, reutilizable e interoperable. Por ejemplo, instituciones como el IEEE o la ISO tienen numerosos grupos de trabajo elaborando estándares para garantizar que los LCMS ofrecen tanto un conjunto de funcionalidades como una forma de representar los datos comunes, de forma que el material educativo sea independiente de la plataforma. Definir dichos estándares es algo complejo, puesto que el elearning toca múltiples aspectos de varios campos. Las herramientas disponibles van adoptando estos estándares de forma gradual, proporcionando en algunos casos sólo un nivel parcial de conformidad (*compliance*).

Es importante notar que hay pocos estándares como tal en el mundo del elearning. El proceso es largo y laborioso hasta que un organismo oficial de estandarización ratifica una propuesta que ya es aceptada por toda la comunidad. Sin embargo, no es infrecuente referirse a especificaciones como las de IMS como “estándares”. Tras unos primeros años con numerosas propuestas de estándares incompatibles entre sí, el panorama se ha aclarado en los últimos años y se está produciendo una convergencia alrededor del proyecto SCORM, que está aglutinando a su vez el trabajo de AICC, IEEE LTSC y, sobre todo, IMS.

En este capítulo se proporciona una visión general del panorama de los estándares de elearning tal y como se encuentra en el momento presente. Posteriormente se analizarán más en detalle las especificaciones más relevantes para la presente tesis, es decir, aquellas que guardan relación con la reutilización de secuenciamientos.

4.2. La iniciativa Advanced Distributed Learning (ADL) y SCORM

La iniciativa Advanced Distributed Learning (ADL, *aprendizaje distribuido avanzado*) es una organización creada originalmente por el Departamento de Defensa de los U.S.A. (www.adlnet.gov) para trabajar con agencias federales, instituciones académicas y miembros de la industria, así como para desarrollar especificaciones para tecnología de apoyo al aprendizaje. ADL trabaja junto a diversas organizaciones de especificación y estandarización como ISO, IEEE

[1] No hay una definición universalmente aceptada de objeto de aprendizaje. Las dos más habitualmente referenciadas son la de David Wiley: “Cualquier recurso digital que puede ser reutilizado para apoyar el aprendizaje” [176]; y la del IEEE: “Cualquier entidad, digital o no, que pueda ser usada para el aprendizaje, la educación o la formación” [78]. La frontera entre lo que es un objeto de aprendizaje y lo que es sólo un recurso es difusa. Es habitual utilizar la denominación de objeto de aprendizaje sólo para aquéllos que están empaquetados en función de algún estándar de los analizados en este capítulo como SCORM o IMS Content Package.

e IMS para desarrollar guías orientadas a hacer el software educativo accesible, adaptable, interoperable y reutilizable. Más que jugar un papel de desarrollador de estándares, esta institución actúa como guía en el proceso.

La contribución más conocida de ADL es el modelo de referencia para objetos de contenido reutilizable (*Shareable Content Object Reference Model, SCORM [1]*). El modelo define la forma en que los sistemas de aprendizaje deben manejar el contenido basado en web y servirlo a los usuarios. SCORM utiliza especificaciones y guías propuestas por otras instituciones (como IMS, ver sección 4.4) pero lo agrupa todo bajo el paraguas de su propio modelo. La mayor contribución del modelo no está en ningún aspecto concreto del elearning sino en cómo deben integrarse todos ellos para conseguir un sistema de elearning efectivo. En el caso ideal, si los desarrolladores de contenido por un lado y los desarrolladores de sistemas por el otro siguieran este modelo, el material de aprendizaje se integraría sin problemas en cualquier plataforma.

La idea subyacente al modelo SCORM es la de considerar un curso basado en web como una colección de objetos de contenido que están interconectados. Para que esta idea se pueda trasladar a plataformas que funcionen es necesario considerar múltiples aspectos: organización del contenido, metadatos, secuenciamiento, etc. SCORM se divide en tres submódulos, cada uno de los cuales se encarga de una faceta del elearning: modelo de agregación de contenidos, entorno de ejecución, y navegación y secuenciamiento.

Modelo de agregación de contenidos El modelo de agregación de contenidos [42] define cómo hay que ensamblar, etiquetar y empaquetar el contenido. SCORM es un modelo basado en objetos, por lo que es necesaria una descripción detallada sobre cómo se conectan dichos objetos. Una experiencia de aprendizaje se compone, en este contexto, de activos (*assets*), objetos de contenido compartible (*Shareable Content Objects, SCOs*) y organizaciones de contenido.

Un SCO está compuesto de varios activos, y varios SCOs componen una organización de contenido. Ejemplos de activos son: documentos HTML, ficheros de audio o vídeo, etc. Los SCOs contienen un conjunto de activos y deben ser ejecutables por el LMS. Una organización de contenido describe cómo se organizan un conjunto de SCOs en una estructura de árbol de ramificación arbitraria. Las hojas de dicho árbol deben ser SCOs o activos ejecutables.

Entorno de ejecución El entorno de ejecución (*RunTime Environment, RTE*) describe el proceso de ejecución que debe realizar un LMS con un SCO, así como el proceso de comunicación entre ambos. Un estudiante sólo tiene un SCO activo en cada momento. El modelo no especifica qué controles de navegación deben presentarse al usuario para elegir diferente material. La comunicación entre el SCO y el sistema se produce a través de un interfaz de programación de aplicaciones que sí es parte del modelo.

Navegación y secuenciamiento SCORM hace uso de múltiples especificaciones así como del estándar IEEE LOM [78] (que se describe en la sección 4.3). El modelo propuesto describe como debe manejarse el material de elearning; las especificaciones se adoptan como parte del modelo para la mayoría de los asuntos concretos de este manejo. Por ejemplo, en el caso del secuenciamiento (uno de los múltiples aspectos considerados por SCORM), el modelo describe cómo dicho secuenciamiento interactúa con el resto del RTE; sin embargo, la descripción del proceso

de secuenciamiento se hace usando la especificación IMS Simple Sequencing [84] (descrita en la sección 4.4.4).

Es muy probable que, a medida que nuevas especificaciones aparecen en relación con nuevos aspectos relacionados con el elearning, las especificaciones de SCORM futuras los irán incluyendo como parte del modelo. Volviendo al ejemplo del secuenciamiento, SCORM 2004 usa Simple Sequencing para su descripción. Sin embargo, es probable que futuras versiones incluyan IMS Learning Design [81] para superar algunas de las limitaciones[2] de Simple Sequencing y proporcionar capacidades de secuenciamiento más flexibles.

SCORM es el modelo más ampliamente aceptado por los actuales LMS. Las herramientas actuales soportan la importación y exportación de cursos en SCORM, y esto se traduce en un cierto grado de reutilización. El mayor problema a la hora de crear un curso con una herramienta e importarlo en otras de forma efectiva está en el grado de conformidad. SCORM es un modelo muy vasto basado en un conjunto extenso de especificaciones, de forma que un fabricante suele soportar sólo un subconjunto de la funcionalidad.

Debido a las variaciones en estos subconjuntos, no es infrecuente disponer de material conforme a SCORM que se ejecuta en una plataforma pero no en otra. Esto ha llevado a que SCORM publique un conjunto de requerimientos de compatibilidad [2] especificando[3] cuáles son los aspectos críticos que las herramientas deben incluir para conseguir la certificación de “conforme a SCORM” (*SCORM compliant*).

4.3. IEEE Learning Object Metadata (LOM)

El Instituto de Ingenieros Eléctricos y Electrónicos (Institute of Electric and Electronic Engineers, *IEEE*) es una organización sin ánimo de lucro que ofrece información técnica y profesional, recursos y servicios a una gran parte de la comunidad ingenieril. Dentro del IEEE, el Comité de estándares de tecnologías educativas (*LTSC*) se enfoca al desarrollo y publicación de estándares técnicos así como a la recomendación de prácticas y guías en relación con las tecnologías educativas. Sus tareas se organizan en grupos de trabajo especializados en diferentes aspectos del abanico del elearning.

La contribución más relevante del IEEE LTSC es la publicación del estándar de metadatos de objetos de aprendizaje (*LOM*), un esquema conceptual de datos que define la estructura de una instancia de metadatos para objetos de aprendizaje. En este contexto, un objeto de aprendizaje es simplemente cualquier entidad que pueda ser utilizada para aprender.

El objetivo de este estándar es definir el conjunto mínimo de atributos que deben adjuntarse a estos objetos de aprendizaje para manejarlos. Estos atributos intentan capturar no sólo aspectos administrativos simples (vg. autor, título, fecha de creación, etc) sino también aspectos pedagógicos (vg. nivel de dificultad, enfoque pedagógico, prerrequisitos, etc). Una característica importante de LOM es que contempla la posibilidad de extender el conjunto de atributos para propósitos específicos.

[2] Estas limitaciones se analizan en las secciones 4.4.4 y 8.3.

[3] Es ilustrativo de la complejidad del proceso y de las especificaciones el hecho de que han aparecido empresas de consultoría especializada en certificación SCORM como *www.conform2scorm.com* y otras.

Los atributos se dividen en las siguientes categorías:

General. Incluye información general sobre el objeto tal como: el título, identificador único, número de catálogo, descripción (de alto nivel), palabras clave, etc.

Ciclo de vida. Atributos relacionados con la evolución del objeto tales como: número de versión, estado (vg. borrador o definitivo), personas que han contribuido, etc.

Meta-metadatos. Información sobre los propios metadatos adjuntos al objeto: idioma, personas que han contribuido al marcado, etc.

Técnicos. Estos atributos están orientados a explicar cómo debe manipularse el objeto. Describe aspectos como: versión, duración, cómo instalarlo, software necesario, etc.

Educacional. Esta categoría contiene información sobre aspectos pedagógicos como: nivel de interactividad, densidad semántica, nivel de dificultad, etc.

Derechos. Información legal en relación con la propiedad intelectual y las restricciones de copia.

Relacional. Estos atributos capturan cómo se relaciona esta objeto con otros. En el caso de relaciones muy complejas puede haber múltiples instancias de estos atributos.

Anotaciones. Información sobre cómo se ha usado el objeto en un entorno de aprendizaje. Esto permite el intercambio de información, sugerencias, etc.

Clasificación. El objetivo es localizar este objeto según una determinada taxonomía. Puede haber múltiples instancias de estos atributos si un objeto tiene que ser clasificado en función de varias taxonomías diferentes.

Actualmente, las herramientas (vg. LCMS, herramientas de autos, etc) tienen un nivel desigual de adopción de LOM. Algunas herramientas de autor (como RELOAD [90]) incluyen un soporte completo del estándar, pero no siempre es el caso.

4.4. IMS Global Consortium (IMS)

El consorcio IMS Global es una organización sin ánimo de lucro orientada a facilitar la adopción de tecnologías de elearning en todo el mundo. Entre sus miembros hay fabricantes, proveedores de contenidos, instituciones educativas, editores, organizaciones gubernamentales y otros consorcios, que pertenecen a más de 50 países. Originalmente el nombre correspondía a Sistemas de gestión instruccional (*Instructional Management Systems*), pero luego las siglas perdieron este significado para evitar confusiones. Según el propio consorcio:

> *El nombre original, cuando IMS comenzó en el año 1997, era el proyecto de sistemas de gestión instruccional (IMS). Con el paso del tiempo, quedó claro que el término "sistemas de gestión instruccional" implicaba más preguntas que respuestas porque términos diferentes eran empleados para describir las mismas cosas (...). IMS se preocupa de la interoperabilidad entre los sistemas de aprendizaje y el contenido educativo*

y la integración empresarial de estas capacidades. Por favor, llámennos simplemente IMS[4].

IMS no publica estándares internacionales. Su labor se centra en promover la adopción de especificaciones técnicas abiertas para alcanzar la interoperabilidad real entre tecnologías. El impacto del consorcio está fuera de toda duda: SCORM incluye varias de sus especificaciones y parte de la comunidad del elearning desarrolla una activa labor investigadora alrededor de las mismas. Varias se han convertido en estándares *de facto* para productos y servicios de elearning.

Las especificaciones de IMS se ocupan de múltiples aspectos del elearning, lo que hace difícil abarcar el conjunto en su totalidad. En el momento de presentar esta tesis, el número de especificaciones disponibles[5] es de veintiséis, pero la lista sigue actualizándose; en el resto de esta sección, se hace un somero análisis de las más importantes. En el caso de las dos especificaciones que son relevantes para el secuenciamiento de actividades educativas, IMS Simple Sequencing e IMS Learning Design, se hace un análisis más en profundidad. La primera esta específicamente diseñada para tratar el problema del secuenciamiento, pero tiene un alcance limitado. La otra tiene un ámbito mucho mayor pero no se centra en el secuenciamiento; sin embargo, atrae mucha más atención de la comunidad de investigadores en elearning, incluso de aquellos investigadores interesados en el secuenciamiento, ya que ofrece posibilidades que resuelven algunas de las carencias de la otra especificación.

4.4.1. Empaquetado de contenido (IMS-CP)

Cuando se diseña material educativo, además de una detallada descripción de cada objeto de aprendizaje, se necesita definir cómo se organiza este material para que pueda ser movido a otra plataforma y reutilizado. Este aspecto es muy importante, porque determina cómo se puede producir el contenido una vez y utilizarlo después en múltiples plataformas. El problema es entonces cómo producir y organizar material independientemente de la plataforma.

La especificación de Empaquetado de contenido (*Content Packaging, IMS-CP* [85]) define cómo debe organizarse el material para importarse, exportarse, agregarse y desagregarse. El elemento fundamental de esta descripción es un fichero XML llamado *imsmanifest.xml* (ver fragmento de XML 4.1), que contiene la colección de recursos usados en el paquete, así como una descripción de cómo se organizan.

4.4.2. Interoperabilidad de preguntas y test (IMS-QTI)

Un aspecto muy importante en el diseño de material de aprendizaje es el de la evaluación. En general, evaluar supone una tarea compleja con numerosas variantes. La especificación de interoperabilidad de preguntas y test (*Question and Test Interoperability, IMS-QTI* [87]) define un modelo de datos específicamente orientado hacia el diseño y la reutilización de preguntas. Este tipo de evaluación es susceptible de ser corregida y de ofrecer realimentación al alumno automáticamente.

[4] *www.imsglobal.org/background.html*
[5] *www.imsglobal.org/specifications.html*

XML 4.1 Ejemplo de imsmanifest.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Partially made with RELOAD Content Package Generator -
    http://www.reload.ac.uk-->
<manifest xmlns="http://www.imsglobal.org/xsd/imscp_v1p1"
          xmlns:imsmd="http://www.imsglobal.org/xsd/imsmd_v1p2"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          identifier="MANIFEST-F30E7446-2FBF-78F6-BC86-E93A532DEA63"
          xsi:schemaLocation="http://www.imsglobal.org/xsd/imscp_v1p1
          imscp_v1p1.xsd http://www.imsglobal.org/xsd/imsmd_v1p2
          imsmd_v1p2p2.xsd">
 <organizations default="ORG-43991866-E3C5-F542-89F8-55DF64F4A425">
  <organization identifier="ORG-43991866-E3C5-F542-89F8-55DF64F4A425"
                structure="hierarchical">
   <title>Organization</title>
   <item identifier="ITEM-F89D863A-C272-F83A-2DD6-E61B9B13DD84"
         isvisible="true"
         identifierref="RES-7A483CA0-08A9-CFB3-55C1-17352B4689C7">
     <title>entryLevel</title>
   </item>
   <item identifier="ITEM-E1851F09-54D5-3094-9C7D-73245E17BBAD"
         isvisible="true"
         identifierref="RES-79E0D0FC-C844-C8C1-F0AF-9CBF74805E17">
     <title>binToBase</title>
   </item>
   <!-- * * * Any number of items here * * * -->
  </organization>
  <!-- * * * More organizations here, if any * * * -->
 </organizations>
 <resources>
  <resource identifier="RES-79E0D0FC-C844-C8C1-F0AF-9CBF74805E17"
            type="webcontent" href="binToBase.jsp">
   <file href="binToBase.jsp" />
  </resource>
  <resource identifier="RES-7A483CA0-08A9-CFB3-55C1-17352B4689C7"
            type="webcontent" href="entryLevel.jsp">
   <file href="entryLevel.jsp" />
  </resource>
  <!-- * * * More resources here * * * -->
 </resources>
</manifest>
```

El modelo propone definiciones para los tipos de preguntas más comunes (vg. verdadero-falso, rellenar los huecos, etc), y define esquemas de puntuación, organización en secciones, aleatorización y producción automática de comentarios de realimentación.

IMS QTI es la especificación alrededor de la cual se ha desarrollado una actividad más importante, debido a que es relativamente fácil de implementar y su aplicación práctica es inmediata. La experiencia obtenida de los desarrolladores ha producido importantes cambios en la especificación, que es actualmente la única de la que ya se ha publicado una segunda versión.

4.4.3. Paquete de información del discente (IMS-LIP)

La especificación de paquete de información del discente (*Learner Information Package, IMS-LIP* [86]) se encarga de la recolección de información sobre un discente, grupo de discentes o productor de contenido educativo. El objetivo principal es facilitar la importación y exportación de este tipo de datos entre plataformas.

Hoy en día, el proceso de aprendizaje se está expandiendo hasta abarcar toda la vida del aprendiente. Este proceso se conoce como aprendizaje a lo largo de la vida (*Life Long Learning, LLL*). De esta forma, dado que un individuo aprende durante un periodo de tiempo mucho más largo y en un número elevado de organizaciones diferentes, la compartición de este tipo de información debe facilitar la adaptación del proceso de aprendizaje en su conjunto.

La especificación se organiza alrededor de varias estructuras: accesibilidad, competencias y habilidades, metas, intereses, notas, etc. Es también destacable el apartado dedicado a seguridad, tema de vital importancia puesto que se está tratando con los datos personales del individuo.

4.4.4. Secuenciamiento simple (IMS-SS)

La especificación de secuenciamiento simple (*Simple Sequencing, IMS-SS* [84]) define un método para definir la secuencia en la que un grupo de actividades de aprendizaje se presentan al estudiante. Incorpora reglas que describen el flujo o ramificación de las actividades según la interacción entre el usuario y las actividades. Su objetivo es ser el punto de encuentro entre los diferentes sistemas de gestión de aprendizaje en términos de la secuenciación de sus contenidos. Por lo tanto, el modelo IMS-SS es neutral intencionadamente en relación con modelos pedagógicos y el uso de estrategias instruccionales.

El nombre *simple* en la especificación no significa que sea relativamente corta o sencilla comparada con otras de IMS. Se la denota como simple porque su ámbito es limitado a algunas vías especificas de definir secuencias: secuencia directa, secuencia auto-guiada y secuencia adaptada. La última sólo es soportada por IMS-SS de forma limitada (ver figura 4.1).

IMS-SS está diseñado para ser integrado con IMS-CP (sección 4.4.1). En principio, podría integrarse con otras especificaciones equivalentes; sin embargo, IMS-CP es actualmente el único mecanismo que ha sido definido para intercambiar instancias de paquetes en IMS. La información del secuenciamiento puede estar integrada en el manifiesto de IMS-CP o en su propio archivo.

La especificación de secuenciamiento simple define un proceso de secuenciamiento completo

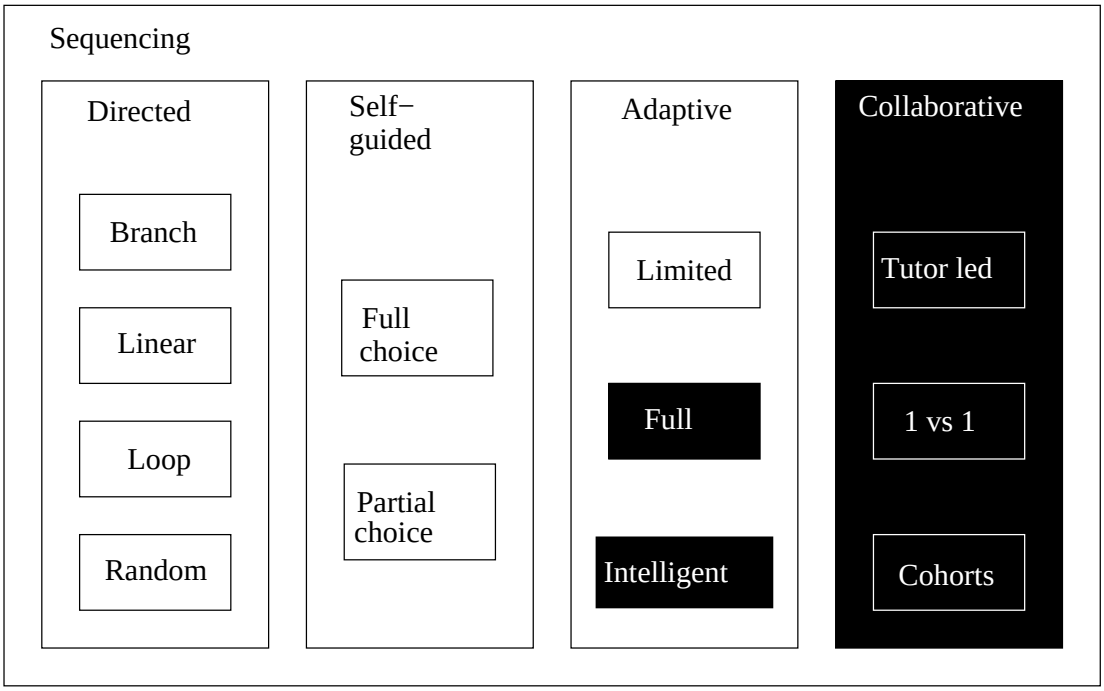


Figura 4.1: Ámbito de IMS-SS (basado en [79]).

con varias etapas (llamadas comportamientos: navegación, finalización, enrollado, secuenciamiento y entrega). Una descripción en detalle de los comportamientos en IMS-SS sería larga y poco relevante para la discusión presente. Esta sección se centrará en los mecanismos que la especificación proporciona al diseñador para definir diferentes secuencias que se adapten al estudiante. Los lectores interesados en el proceso completo pueden consultar [80].

En el contexto de IMS-SS, una actividad de aprendizaje se define como

> *. . . una unidad pedagógicamente neutral de instrucción, conocimiento, valoración, etc. Puede tener sub-actividades y anidarse hasta un nivel arbitrario de profundidad. Cada actividad puede tener un estado de seguimiento para cada estudiante que experimenta esta actividad. Las actividades pueden ser intentadas múltiples veces o únicamente un número determinado de veces. Se pueden saltar, abandonar, ser terminadas de forma normal, etc. Todas las actividades se realizan en el contexto de una actividad padre* [79].

IMS-SS considera que las actividades de aprendizaje se organizan de forma jerárquica. Sin embargo, las sub-actividades no se consideran parte de la actividad principal. Las actividades se entregan de una en una, normalmente la actividad padre primero y después las actividades hijo (ver figura 4.2). Las actividades pueden tener recursos auxiliares asociados a ellas; en caso contrario la especificación IMS-SS no define ningún comportamiento.

Las actividades de aprendizaje se ordenan en una estructura de árbol. Cada nodo y cada hoja en el árbol es una actividad. Cada nodo en el árbol puede tener cualquier número de nodos u hojas que dependan de él por lo que las ramas pueden no estar equilibradas. No hay ninguna relación entre conceptos como lecciones, cursos, etc, y la profundidad de los nodos en la jerarquía del árbol.

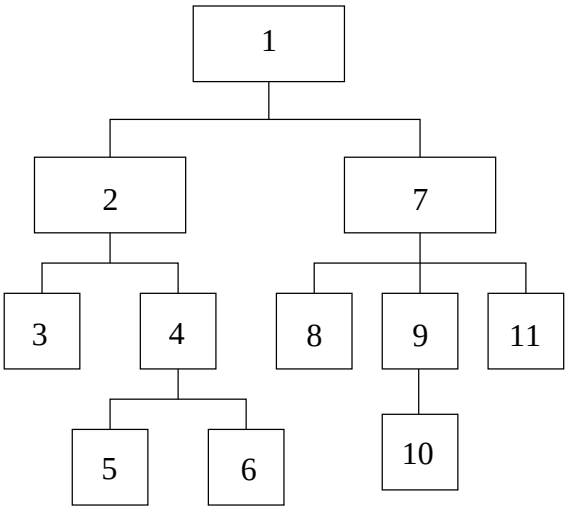


Figura 4.2: Recorrido en pre-orden en un árbol IMS-SS

IMS-SS define la forma canónica de recorrer un árbol de actividades como recorrido en pre-orden (*preorder traversal*): los nodos padre se recorren antes que los nodos hijo, los nodos del mismo nivel se recorren de izquierda a derecha y todos los descendientes de un nodo se recorren antes de pasar al siguiente nodo del mismo nivel. La figura 4.2 muestra un ejemplo.

IMS-SS permite al diseñador seleccionar si un grupo de actividades hijo deben ser secuenciadas de manera guiada (sin intervención por parte del estudiante) o si el estudiante debería elegir su próxima actividad. Esto se hace a través del uso de dos propiedades en cada nodo (no hoja): control de secuencia en flujo (*Sequencing Control Flow*) y control de secuencia por elección (*Sequencing Control Choice*). El estado de estas dos propiedades afecta al secuenciamiento de las actividades que sean hijos del nodo. Si dichas actividades hijo tienen a su vez algún hijo, deben definir su propia política. Adicionalmente, se puede dar un grado de aleatoriedad tanto a la política flujo como a la política elección.

El hecho de que una determinada política sólo afecta a sus hijos y no al resto de descendientes es importante. A una actividad y sus hijos se les llama clúster. El ámbito de las reglas en IMS-SS solo afecta a los clústeres: las reglas se definen en el padre, afectan al padre y/o sus hijos, y no tienen ningún efecto más arriba o abajo del árbol.

Se puede modificar el camino a recorrer por defecto gracias a la asociación de reglas de secuenciamiento creadas por el diseñador. El recorrido es activado por estudiante a partir de los eventos de navegación, o bien por el propio sistema de entrega. Las reglas de secuenciamiento son evaluadas en tiempo de ejecución. Las reglas que afectan al secuenciamiento pueden ser de tres tipos: de limite (*limit rules*), de enrollado (*roll-up rules*) o de secuenciamiento.

Las condiciones de límite definen las restricciones para acceder a una actividad (o nodo) basadas en condiciones como la hora del día, el tiempo empleado en la actividad y el número de intentos. Los procesos pueden hacer referencia a la descripción de las condiciones límite para cualquier actividad del árbol. Esto puede tener un efecto en el secuenciamiento (ver más adelante).

Las reglas de enrollado describen como el éxito o el fracaso en las actividades hijo de un clúster influyen en el secuenciamiento de la actividad padre y de sus hermanas. Las condiciones que afec-

tan a cada actividad subordinada pueden tener valores como: satisfecho, completado, intentado, etc. Puede haber más de una condición en cada subordinada y pueden combinarse con una conjunción lógica global (operador AND: todas las condiciones deben ser cumplidas para lanzar la regla) o una disyunción lógica global (operador OR: cuando una se cumple, es suficiente). Cada condición puede ser negada (es decir: NO satisfecha, NO intentada). Además, cualquier sub-actividad puede ser no tenida en cuenta (actividades opcionales sin efecto en el secuenciamiento global). Una regla de enrollado puede dispararse si todas las condiciones de las subordinadas se cumplen o si alguna, ninguna o un porcentaje en particular se cumple. Cuando se aplica la regla, se ejecutan acciones que tienen un efecto directo en la actividad padre del clúster. Las acciones pueden cambiar el estado de la actividad padre a: satisfecho, completado o cualquiera de sus opuestos. Esto tiene un efecto en el secuenciamiento (ver reglas de secuenciamiento, más adelante).

Hay tres tipos de reglas de secuenciamiento en IMS-SS: pre-condiciones, post-condiciones y condiciones de salida. Todas pueden asociarse a una misma familia de condiciones pero ejecutan acciones diferentes con diferentes efectos en el secuenciamiento. Las condiciones evalúan diversos aspectos de la actividad (de los que muchos son modificados con respecto a las condiciones límite o de enrollado). Algunos ejemplos de los aspectos considerados por las condiciones son: algún objetivo ha sido satisfecho o completado, algún objetivo ha sido satisfecho o completado hasta un umbral, el límite de intentos o el tiempo transcurrido ha excedido un cierto umbral, etc. Al igual que en el caso anterior, se puede elegir más de una condición para una actividad, las condiciones se pueden combinar con una conjunción lógica global o una disyunción lógica global, y cualquier condición puede ser negada (NO satisfecha, NO intentada, etc).

Acciones en IMS-SS En la descripción que se expone a continuación, el efecto de las acciones de una condición posterior o de una acción de salida ha sido simplificado puesto que no se cubre en detalle el modelo del Proceso de secuenciamiento de IMS-SS. El comportamiento detallado de estas acciones se explica en [80].

Las acciones de las pre-condiciones se usan cuando se recorre el árbol de actividades buscando una actividad para la entrega. Las pre-condiciones son evaluadas antes de entregar la actividad. Sus posibles acciones asociadas pueden ser: omitir (omitir la actividad actual en un flujo de procesos secuenciales; de forma indirecta, esto omite todos los subordinados de esta actividad), esconder la elección (esconder la actividad actual del grupo de actividades presentadas al usuario), inutilizar (combina el efecto de las dos anteriores), detener el recorrido (parar el movimiento en el árbol de actividades hacia delante) e ignorar (no tomar acción alguna).

Las acciones de las post-condiciones se aplican cuando el intento de realizar la actividad actual termina. Pueden ser: salir del padre (sale de la actividad padre), salir por completo (sale del sistema), repetir (intenta la actividad de nuevo), repetir todo (intenta otra vez todas las actividades del clúster), continuar (continúa; es importante notar que la siguiente actividad en el árbol no tiene por qué ser la próxima que se ofrece al estudiante, dependiendo de las pre-condiciones), previo (va a la actividad anterior; es importante notar que la actividad anterior en el árbol no tiene por qué ser la que se ofrece al estudiante, dependiendo de las pre-condiciones) o ignorar (no tomar acción alguna).

Las acciones de salida se aplican después de que un intento en una actividad subordinada termina. Pueden ser: salir (termina la actividad incondicionalmente) o ignorar (no tomar acción alguna).

Limitaciones de IMS-SS

La propuesta de IMS-SS es un paso en la buena dirección, pero adolece de ciertas carencias:

Falta de flexibilidad. Está basado en un paradigma sencillo, útil para realizar secuenciamientos lineales "a trozos" de la información. En otras palabras, su sistema de pre-condiciones y post-condiciones es muy sencillo y permite bastante generalidad, pero presenta complicaciones para definir un secuenciamiento más allá del camino lineal con pequeños saltos. Definir una secuencia con ciclos o con saltos hacia atrás produce unos enormes ficheros de descripción XML con infinitud pre-condiciones y post-condiciones [69]. Desde Ausubel y Piaget, varios autores han incidido en la importancia de los ciclos para el aprendizaje [112, 145] y estos son difíciles de definir sobre IMS-SS.

IMS-SS está pensado para intercambiar información entre aplicaciones a bajo nivel. Por tanto, un autor sólo debería preocuparse de definir un secuenciamiento para su material utilizando la herramienta que considere más oportuna, pudiendo luego traducir de alguna manera entre su lenguaje de secuenciamiento y la semántica de IMS-SS. Sin embargo, no es menos cierto que, hoy por hoy, hay numerosas herramientas que están utilizando esta semántica de forma directa, con sus limitaciones; más aún, es posible que no sea necesario introducir tantos pasos intermedios. Varias de las propuestas que intentan ir más allá de la especificación buscan solucionar esta limitación [69, 172].

Ausencia de modelo de usuario. En la propia especificación se comenta que no persigue ser capaz de definir un secuenciamiento totalmente adaptativo; se centra en los secuenciamientos estáticos, semi-estáticos (guiados sólo por el estudiante) y adaptativos de forma limitada (dejando fuera lo que IMS llama "totalmente adaptativo" —*full adaptive*— y adaptativo inteligente —*intelligent adaptive*—, así como secuenciamiento colaborativo —*collaborative sequencing*—). Previsiblemente, una especificación futura de secuenciamiento "no simple" busque paliar esta carencia.

Esta limitación lleva a que no sea posible complementar el secuenciamiento establecido por IMS-SS con un modelo de usuario externo. No se define ningún mecanismo para actualizar un modelo de usuario externo ni para capturar datos desde las actividades de aprendizaje. Sólo se permite tomar las decisiones de secuenciamiento en función del cumplimiento de objetivos (consecuencia del modelo de usuario usado en la especificación IMS Learner Information Package, IMS-LIP [86]). Se puede guardar dentro del alcance del IMS-SS alguna información del usuario (vg. el tiempo usado en una actividad, en cuántas actividades de un clúster ha tenido éxito) pero no es posible ir más allá de esto y definir/modelar conceptos abstractos (vg. si el estudiante prefiere o no seguir enfoques tipo "de lo general a lo concreto" o al revés).

Orientado sólo al estudiante: Está pensado para secuenciar actividades educativas para el estudiante. No está pensado para, por ejemplo, definir un sistema de adaptación de temario y sugerencias para profesores, en función de sus alumnos. Esto se debe, en parte, al modelo de usuario implícito (ver punto anterior).

El hecho de que IMS-SS no tenga un modelo generalizado para los estudiantes es posiblemente su mayor limitación. Ésta es, en gran medida, la razón por la que el interés de la mayor parte

de la comunidad se está enfocando hacia otra especificación IMS Learning Design. Aunque no está orientada específicamente al problema del secuenciamiento, ofrece algunos mecanismos para definir secuencias. Por otra parte, sus propiedades permiten una semántica que permite modelar usuarios. Por esto, mucha gente de diferentes campos trabajan ahora con IMS Learning Design, incluyendo a una gran parte de quienes están específicamente enfocados al problema del secuenciamiento.

4.4.5. Diseño de aprendizaje (IMS-LD)

El objetivo de IMS-LD (*Learning Design, IMS-LD* [81]) es describir estrategias de enseñanza y/o enfoques pedagógicos así como promover el intercambio entre éstos y los sistemas de gestión de aprendizaje (*Learning Management Systems, LMS*). Por lo tanto, intenta documentar estrategias de enseñanza, estableciendo y adhiriéndose a procedimientos prescritos para asegurar de la consistencia de dicha documentación [82].

Lo principal en este caso es codificar las estrategias educacionales de forma consistente, que pueda ser entendida por ordenadores y por humanos. De esta forma, el contexto de una unidad de aprendizaje, curso o programa, puede ser gestionado de forma separada del propio contenido. Esta información permite a los instructores describir el enfoque que ellos usan en su trabajo asociándolo a sus contenidos; ésto facilita la compartición y la reutilización del contenido, que se ha diseñado para su propia estrategia instruccional y disciplina, con otros colegas. Por otra parte, esta información puede ser usada para adaptar o interpretar contenido de aprendizaje bajo una estrategia instruccional que difiera de la estrategia para la cual ha sido diseñado. Además, tener información sobre el contexto pedagógico de algunos contenidos educativos puede facilitar su adaptación entre los LMS.

La especificación permite describir diversos tipos de estrategias de aprendizaje, pero mantiene distancia respecto a ellas y no se asocia con ninguna en particular. Por ello, no ofrece un vocabulario específico para describir los diferentes tipos de enfoques de aprendizaje. IMS-LD usa su propia metáfora para crear un meta-lenguaje que puede describir cualquier enfoque. Ésta metáfora es la del guión de una obra de teatro, película o juego [97]. Se asume que un guión es capaz de modelar todo tipo de comportamientos e interacciones entre los actores que ocurren en el contexto de un entorno concreto y, por ello, puede expresar todo tipo de situaciones. Puede ser muy estricto y detallado o puede dejar más espacio a la improvisación. Y aún más importante, el guión es una *descripción de alto nivel* de la obra que se centra en algunos aspectos pero se abstrae en otros: los guiones se escriben de la misma forma independientemente de la compañía de actores que vaya a representar la obra, o de si es una comedia o un drama. En otras palabras, los guiones deben ser *interpretados*. Se emplea por tanto un vocabulario basado en obras, actos, papeles, actividades y condiciones. Se supone que con este vocabulario se puede expresar cualquier estrategia pedagógica. Las *obras* se componen de *actos*, y éstos son el medio para sincronizar diferentes *actividades* que ocurren de forma simultanea. Las actividades son realizadas por la gente que tiene un *papel*. Hay dos familias o tipos de papeles: papeles de *estudiante* y papeles de *profesor*. Finalmente, las *condiciones* permiten una variabilidad en el transcurso de la obra (si algo ocurre, entonces ocurre ésto; si no, ocurre ésto otro). A todo el paquete que incluye la definición de una estrategia pedagógica o un diseño de aprendizaje, junto a los recursos o servicios necesarios para su desarrollo, se le denomina una Unidad de Aprendizaje (*Unit of Learning, UoL*). En IMS-LD, el proceso desde

que un UoL empieza hasta que acaba se conoce como una ejecución (*run*).

En relación con el objeto de esta tesis, las condiciones son el elemento más importante de la especificación. Las condiciones son las herramientas que permiten definir diferentes secuencias de actividades. Las condiciones se evalúan contra las *propiedades*; las propiedades en IMS-LD pueden ser:

Locales: su ámbito se limita a la ejecución actual.

Globales: se mantienen a lo largo de varias ejecuciones en el mismo UoL.

Además, el ámbito de las propiedades en un UoL puede ser:

General: están asociadas al UoL en su totalidad (vg. máximo número de estudiantes a los que se les permite tener un determinado papel).

Un papel (*role*): se asocian a un papel (vg. máximo tiempo permitido a cada estudiante para terminar una actividad).

Personal: describen o afectan a un usuario individual (vg. el tiempo que un estudiante está en un acto en particular).

Aunque se pueden usar para diversos fines, las propiedades son el medio para ofrecer información específica sobre los estudiantes. Esto hace posible que el UoL se adapte a las necesidades o capacidades particulares de cada estudiante o grupo. La asignación o el cambio del valor de las propiedades de un UoL conlleva la evaluación de las condiciones en IMS-LD. Las propiedades tienen un nombre y un identificador, un tipo, un valor y —en ocasiones— algunas restricciones sobre los valores que puede tener.

Las condiciones son evaluadas contra las propiedades; cuando se cumple una condición, se realiza una acción. Las condiciones se evalúan, como se ha dicho, cada vez que el valor de alguna de las propiedades implicadas cambia. Hay tres tipos de eventos que pueden cambiar el valor de una propiedad (por lo que se usan para desencadenar acciones de forma indirecta). Pueden ser: eventos proporcionados por un contador (vg. el reloj de ejecución del UoL); eventos proporcionados por el usuario (vg. cuando éste decide acabar una actividad); y, por último, pueden ser la finalización de una actividad, acto u obra por la razón que sea (vg. elección del usuario o límite de tiempo).

Las condiciones en IMS-LD son buleanas y se combinan con los operadores de conjunción (AND), disyunción (OR) y negación (NOT). Las cuestiones que se pueden comprobar incluyen:

- Si la persona actual tiene asignado un papel.
- Si alguna actividad, parte de papel (*role-part*), acto u obra ha sido ya completada.
- Si una determinada propiedad está definida o no.
- Si una determinada propiedad es igual o diferente a algún valor constante, y otras operaciones con operadores matemáticos y umbrales numéricos.

- Preguntas sobre el tiempo transcurrido.

Como ya se ha dicho más arriba, las condiciones se evalúan cada vez que el valor de una propiedad cambia. Las acciones que pueden ser disparadas por una condición son de cuatro tipos:

Cambiar el valor de una propiedad. Cambia el valor de una propiedad. Esto puede producir otra re-evaluación de todas las condiciones. Algunas condiciones que no se cumplieran anteriormente pueden cumplirse debido al cambio, lo cual lleva a nuevos cambios en otras propiedades, etc.

Ocultar. Oculta un objeto o servicio de aprendizaje, un entorno, una actividad o una obra. Si ya estaba oculta, no hay ningún efecto.

Mostrar. Muestra un objeto o servicio de aprendizaje, un entorno, una actividad o una obra. Si no estaba oculta, no hay ningún efecto.

Completar. Completa un acto o actividad. Como resultado de esto algunas condiciones pueden cambiar su valor, lo que provoca otra re-evaluación de todas las condiciones.

La combinación de las acciones de esconder/mostrar puede ser usada para realizar algún tipo de secuenciado adaptativo. Por ejemplo, dos actividades pueden estar escondidas para un estudiante en particular, resultando en un salto efectivo a una tercera. Esto se trata en más detalle en la sección 8.1.

Hay un último grupo de elementos de IMS-LD que son importantes para el secuenciamiento. Los elementos globales (*global elements*) son extensiones XML diseñadas para recursos escritos en XHTML [170]. Permiten modificar o acceder a las propiedades desde las actividades. Hay cuatro tipos de elementos globales, que se corresponden con cuatro operaciones diferentes: ver propiedad, ver grupo de propiedades, asignar (*set*) propiedad y asignar grupo de propiedades.

No hay otra forma de modificar las propiedades a partir de las actividades por lo que sólo los recursos en XHTML que hacen uso de los elementos globales pueden tener un efecto en el secuenciamiento durante una ejecución del UoL. El resto de las decisiones de adaptación del secuenciamiento deben ser realizadas antes de que se inicie el UoL, y tendrán efecto durante toda la ejecución. Es evidente que en el segundo caso se puede alcanzar un grado de adaptación muy limitado.

Ventajas de IMS-LD respecto a IMS-SS

Aparte de ser una especificación más amplia que IMS-SS y abarcar más aspectos de la práctica educativa, la principal ventaja de IMS-LD sobre IMS-SS es la presencia de propiedades. Las propiedades permiten construir un modelo de usuario de bajo nivel basado en pares <variable, valor>.

Este mecanismo sencillo permite construir estrategias más flexibles para el secuenciamiento y otros aspectos del elearning. En el ámbito de esta tesis, las propiedades juegan un papel fundamental a la hora de expresar un secuenciamiento complejo definido por un grafo de secuenciamiento (capítulo 8), lo cual es imposible en IMS-SS.

Limitaciones de IMS-LD

La especificación IMS-LD no está orientada específicamente al problema del secuenciamiento de actividades educativas. En parte por ello, presenta varias carencias:

Anidamiento de las condiciones. En IMS-LD las condiciones de tipo *if-then-else* sólo son anidables parcialmente[6].

Ausencia de una jerarquía genérica. Aunque la especificación permite definir cierto grado de jerarquía entre actividades (mediante elementos *activity-structure*, por ejemplo), esto no es una jerarquía real desde el punto de vista del secuenciamiento ya que la secuencia de actividades no se puede alterar de la misma forma en todos los "niveles" de la jerarquía. El agrupamiento de actividades de forma jerárquica es importante cuando se aborda el secuenciamiento de un número muy grande de recursos o actividades.

Comunicación con otras aplicaciones. Aunque IMS-LD dispone de propiedades y esto le permite almacenar estados, la forma en que estas propiedades pueden ser modificadas por aplicaciones externas es muy limitada. Un manifiesto puede tener múltiples condiciones que modifiquen el valor de sus variables internas, pero la única forma en que una aplicación externa puede alterar su valor es mediante el uso del elemento global *set-property*, el cual sólo puede ser modificado por el usuario y no por una aplicación directamente.

[6]Un elemento *then* no puede contener un elemento *if*, aunque un elemento *else* sí que puede ([83], páginas 53-54).

5 Inteligencia de enjambre

Un hormiguero puede echar abajo una montaña

– Proverbio japonés

La inteligencia de enjambre (SI, por sus siglas en inglés) es una técnica de inteligencia artificial basada en el estudio del comportamiento de sistemas descentralizados y auto-organizados (inspirada por el comportamiento de los insectos sociales). Las técnicas de inteligencia de enjambre constituyen un campo creciente, con una actividad investigadora muy activa. Sus aplicaciones fuera del elearning son numerosas. y abarcan desde el enrutado de paquetes hasta la robótica. Este capítulo hace una breve introducción al tema de la inteligencia de enjambre, presentando los conceptos básicos y algunas aplicaciones que no están relacionadas con el elearning. Después se muestran las dos familias principales de aplicaciones relacionadas con el elearning: el filtrado colaborativo y el secuenciamiento colaborativo. Dos aplicaciones representativas del primer tipo y otra del segundo se estudian después en detalle. El sistema de Paraschool es analizado en profundidad puesto que algunas de las contribuciones que se presentan en el capítulo 9 han sido desarrolladas con él.

Swarm intelligence (SI) is an artificial intelligence technique based on the study of collective behavior in decentralized self-organized systems (inspired by the behavior of social insects). Swarm intelligence techniques constitute a growing field, with a very active research community. The number of successful applications outside elearning is enormous, covering topics like packet routing and robotics. This chapter makes a brief introduction to the topic of swarm intelligence, presenting the basic concepts and some applications not related to elearning. The two main families applications related to elearning are then explained: collaborative sequencing and colaborative filtering. Two illustrative applications of the first type and another one of the second are studied in detail. The Paraschool system is thouroughly analyzed because some of the results that are later presented on Chapter 9 are developed with it.

5.1. El concepto de enjambre y sus aplicaciones

Existen diversas definiciones para el concepto de enjambre, pero consideramos que la más adecuada para los propósitos de esta tesis es la de Kennedy y Eberhart:

> *Un* enjambre *es una población de elementos que interactúan entre sí y que es capaz de optimizar un objetivo global a través de la búsqueda colaborativa en un espacio. Es habitual que se enfaticen las interacciones locales (en el sentido topológico)*[1].

En esta definición, los elementos aludidos pueden ser maquinas muy simples o seres vivos complejos. Hay dos restricciones que suelen darse adicionalmente: las interacciones son exclusivamente locales y normalmente la interacción no se realiza de forma directa sino indirecta, a través del entorno. La propiedad que hace que los enjambres sean interesantes es su comportamiento autoorganizado; en otras palabras, es el hecho de que la interacción y la combinación de los resultados de muchos procesos simples puedan conducir a resultados complejos. Ejemplos clásicos de enjambres son: las colonias de hormigas, los nidos de termitas, los cardúmenes de peces e incluso los atascos de tráfico.

El comportamiento de las hormigas es, tal vez, el ejemplo más conocido de inteligencia de enjambre. La mayor parte de las especies de hormigas deposita una substancia química llamada feromona mientras se mueven de una fuente de comida a la siguiente. Las hormigas no se comunican directamente unas con otras sino que siguen el rastro de feromonas (dejando sus propias feromonas en el camino con lo que el rastro se refuerza). Los rastros más cortos se refuerzan más puesto que la cantidad de hormigas que los atraviesan en el mismo periodo de tiempo es mayor, y esto hace que sean seguidos por más hormigas. Al final, el bucle positivo de realimentación termina con un camino optimizado conectando la fuente de comida y el nido, sin un conocimiento global del problema por ninguno de los elementos del enjambre (las hormigas). Este proceso de comunicación indirecta en un enjambre se llama estigmergia [16]. Otro ejemplo de proceso estigmérgico es la construcción del nido por las termitas, en el cual los insectos son capaces de construir edificios complejos con arcos, vestíbulos y sistemas de ventilación a partir de reglas locales simples, sin tener nunca un concepto o una visión global del termitero.

Las técnicas de inteligencia de enjambre constituyen un campo creciente, con una actividad investigadora muy activa y se han aplicado a distintos tipos de problemas. Los ejemplos van desde lo muy genérico, como la asignación de colores a un grafo [34] o la optimización restringida (en [183] se puede ver un resumen de los avances más relevantes), a aquellos aplicados a problemas muy específicos como distribuir tareas entre los robots de una fabrica [121], asignar una ruta a una flota de camiones [61] o incluso diseñar el horario para un grupo de asignaturas universitarias [153]. Se puede encontrar una panorámica de las diversas aplicaciones en robótica[2] con explicaciones en paralelo de los diversos comportamientos de las hormigas que les sirven de inspiración (desde recogida de comida a transporte colectivo o construcción de nidos) en [16]. Hoy es habitual usar el termino optimización por colonias de hormigas[3] (*Ant Colony Optimization, ACO* [45, 44]) para

[1] "A *swarm* is a population of interacting elements that is able to optimize some global objective through collaborative search of a space. Interactions that are relatively local (topologically) are often emphasized" [93].

[2] La primera aparición del término "inteligencia de enjambre" se produce dentro del ámbito de la robótica [12], aunque en aquel caso la relación con los enjambres del mundo real era escasa.

[3] El término se emplea por primera vez a finales de los noventa. En [46] aparece de manera informal, y se formaliza en [45] y [44].

el grupo de heurísticas usadas en estos problemas y que se inspiran en el comportamiento de estos insectos. La aplicación mas exitosa de este paradigma ha sido en el problema del encaminamiento en redes de conmutación de paquetes [47].

Internet permite que grandes cantidades de gente se comuniquen con mínimos retrasos. La comunicación puede ser: directa (como en el caso de la telefonía IP o de los chats) o indirecta (como en el caso de las encuestas de Internet o del intercambio de archivos), y además hay casos intermedios (como los foros de Internet). Un gran número de participantes, comunicación indirecta, conocimientos locales y acciones locales definen un marco en el que puede producirse la aparición de comportamientos de enjambre gracias a la combinación de las actividades de sus miembros. Los más relevantes por el momento son los sistemas de filtrado colaborativo y los sistemas de secuenciamiento colaborativo.

5.2. Aplicaciones en elearning

Los sistemas de ayuda a la educación basados en web han sufrido un marcado crecimiento en los últimos años; han aumentado y se han diversificado en cuanto a escenarios de aplicación. La web se ha convertido tal vez en la fuente más importante de acceso a material educativo. Sin embargo, la web no sólo ofrece acceso a información; también puede conectar un número elevado de personas de cualquier parte del mundo en un tiempo despreciable, y están apareciendo aplicaciones que se basan en este hecho. Algunas de ellas son aplicaciones de elearning.

Un campo incipiente en la investigación en técnicas de inteligencia de enjambre se centra en el estudio de la utilización de dos de las características fundamentales de Internet: retardos cortos en las comunicaciones (independientes de la ubicación geográfica) y un número muy elevado de usuarios. El resultado son los *sistemas sociales en línea* intentan emular el comportamiento de los grupos sociales en el mundo real. La aplicación al problema del filtrado de contenidos lleva a los sistemas de *filtrado colaborativo*; su aplicación al problema del secuenciamiento de material lleva a los sistemas de *secuenciamiento colaborativo*. Ambas familias de sistemas intentan extraer información del comportamiento del grupo como tal y usarla para beneficiar a sus usuarios (vg. proporcionando una mejor selección o hallando un mejor camino pedagógico).

El filtrado colaborativo se basa en la premisa de que la gente que busca información puede hacer uso de lo que otros ya han buscado, encontrado y evaluado. Los sistemas de filtrado colaborativo tradicionales almacenan las preferencias y las evaluaciones de los usuarios con respecto a varios elementos (desde novelas a canciones, pasando por recursos educativos en una clase a distancia). Estas preferencias permiten a otro usuario ver lo que sus iguales han preferido y usar esta información como guía a la hora de tomar decisiones. En los últimos años, el crecimiento del comercio electrónico ha estimulado el crecimiento de los sistemas de filtrado colaborativo para su uso como sistemas de recomendación[4]. De esta forma, la meta de un sistema de filtrado colaborativo es ser capaz de predecir la utilidad de un elemento para un determinado usuario basándose en los gustos pasados del usuario y las opiniones de los restantes miembros de la comunidad. El sistema CoFIND [49] es un ejemplo representativo de filtrado colaborativo aplicado a la enseñanza. Además, ha sido diseñado haciendo énfasis en los aspectos de auto-organización y estigmergia. Es importante para entender el desarrollo de esta tesis y por ello se analiza en este capítulo.

[4]Ejemplos conocidos están en *www.amazon.com*, *www.barnesandnoble.com* o *www.casadellibro.com*.

Centrándonos en el problema del secuenciamiento, hay varias iniciativas que permiten adaptar éste a diferentes usuarios (las más importantes se han tratado en el capítulo 3). Lamentablemente, todas suelen depender de un operador humano (vg. un profesor o un diseñador educativo) para diseñar o definir las diferentes posibilidades que existen para los estudiantes. Es el estudiante quién elige su propio secuenciamiento, sea por elección propia o de forma indirecta a través de algún mecanismo del que no es consciente; pero es el diseñador quien debe definir los diferentes secuenciamientos de entre los que estudiante puede elegir. Por ello, todos estos enfoques comparten una misma debilidad: al ser tan importante el papel del operador humano, un error por su parte afecta a todo el sistema. Las técnicas de secuenciamiento social pueden ayudar a paliar este problema. La aplicación de técnicas de inteligencia de enjambre al problema del secuenciamiento tiene diversas aplicaciones: detección automática de diseños instruccionales ineficaces, corrección automática de caminos pedagógicos mal diseñados, organización automática del material educativo, etc. En la sección 5.3 describen dos ejemplos representativos de este tipo de sistemas: el sistema de Paraschool y el sistema de Learning Networks.

5.3. Secuenciamiento colaborativo

El problema del secuenciamiento adaptativo es, dado un grupo de actividades de aprendizaje, encontrar la mejor secuencia para un estudiante en particular. Hay estudiantes que prefieren un enfoque de arriba hacia abajo y que aprenderán más si reciben antes una explicación general; otros, sin embargo, prefieren empezar con ejemplos y deducir los principios abstractos generales a partir de ahí. Algunos estudiantes entenderán unos conceptos antes mientras que a otros les costara más trabajo; los primeros preferirán secuencias cortas que no les aburran una vez que han aprendido lo necesario, mientras que los segundos necesitarán secuencias mas largas (quizá incluso con bucles que repitan algunas actividades). Dado un modelo de usuario apropiado, un sistema puede adaptar la secuencia de actividades de aprendizaje a cada estudiante. Lamentablemente, la mayor parte de los enfoques comparten una debilidad común.

En ocasiones, los errores humanos son evitables. Por ejemplo, un error al escribir el nombre de una actividad puede ser detectado por el propio autor fácilmente, e incluso se pueden utilizar diversas estrategias para detectarlos y corregirlos automáticamente: declaraciones previas, correcciones ortográficas (se puede consultar [164] para ver un análisis de diversas técnicas), etc. Sin embargo, el proceso del diseño secuencial esta ligado a algunos problemas inherentemente irresolubles.

En primer lugar, un diseñador siempre tiene un conocimiento limitado. Aparte de las actividades y los recursos propuestos por él como interesantes para los alumnos, éstos pueden conocer otros adicionales. Si la estrategia de secuenciamiento depende totalmente del diseñador o profesor, esas actividades de las que el profesor no es consciente nunca serán realizadas por sus alumnos. Esta es una limitación a la cual estamos muy acostumbrados y se acepta como normal, aunque las nuevas posibilidades que nos traen las tecnologías de la información (especialmente Internet) pueden darnos la oportunidad de superarlo. Por otro lado, es una realidad que los estudiantes van cambiando con el tiempo por diversas razones (cambios en los planes de estudio, posibilidades de acceso distintas a los medios de comunicación y a las fuentes de conocimiento, etc). Un diseño de secuenciamiento que proporcione buenos resultados hoy puede no ser tan efectivo el día de mañana. Esto es algo que el profesor no puede resolver solo [70].

Los sistemas sociales intentan emular el comportamiento de los grupos sociales en la vida real, aprovechándose de los pequeños retrasos en las comunicaciones y el gran número de usuarios que permite conectar Internet. Las aplicaciones de estas técnicas al problema del secuenciamiento se traducen en un sistema de secuenciamiento social o colaborativo.

5.3.1. Paraschool

Paraschool es la empresa más importante de Francia en el ámbito del elearning. Ofrece sus servicios a varios miles de estudiantes en el país. Éstos consisten en cursos adaptados a varias asignaturas, ejercicios en línea, correcciones y realimentación sobre los mismos, etc. Su sistema usa técnicas basadas en Optimización por colonia de hormigas (*ACO*) para detectar malas planificaciones pedagógicas y corregirlas de forma automática [163].

El grafo de ejercicios

El sistema divide las actividades que van a ser entregadas al estudiante en cursos y capítulos. Los cursos pueden variar en su alcance desde cursos cortos de formación o reciclaje (por ejemplo cursos de seguridad cuando se usa maquinaria pesada en una empresa) hasta cursos que abarcan un año académico entero en el colegio (vg. *troisième*[5]). Los cursos se dividen en capítulos. Dentro de cada capítulo, se define un grafo de actividades. Cada nodo del grafo representa una actividad, la cual consta típicamente de:

- una página web de teoría sobre un concepto (vg. resolución de ecuaciones de segundo),
- un ejercicio que ilustra el concepto presentado, y
- una tercera página que corrige las respuestas del estudiante y le ofrece realimentación sobre su respuesta[6].

Las aristas o arcos del grafo representan las posibles transiciones después de terminar un nodo. Los nodos no están necesariamente conectados unos a otros. Además, los mecanismos de navegación libre (ver más adelante, página 49) pueden crear nuevas aristas.

A cada arco del grafo se le asocia una probabilidad que determina cuál va a ser la siguiente unidad de aprendizaje que se va a entregar al estudiante. Las unidades con una probabilidad mayor serán elegidas más frecuentemente que aquéllas con una probabilidad menor. Un equipo pedagógico de profesores y expertos en la materia deciden estas probabilidades y las asignan a las posibles transiciones entre actividades en forma de pesos pedagógicos. Estos pesos se normalizan y se asignan a los arcos. Los arcos que conectan nodos son equivalentes a restricciones en la secuencia de las unidades de aprendizaje. Las probabilidades asociadas a estos arcos determina qué transiciones serán usadas normalmente después de cada nodo particular. Así, si de un nodo salen tres arcos con pesos[7] 2, 2 y 20, el diseñador considera que la mayoría de los alumnos deberían salir por el tercer

[5] *Troisième* es el curso equivalente (por edad) a cuarto de la ESO en España.

[6] Aunque la separación clásica [73] entre los dominios del conocimiento, del estudiante y de la instrucción no está formalizada en el sistema de Paraschool, el sistema es *de facto* un ITS.

[7] Por motivos de implementación, los pesos pedagógicos son números naturales mayores o iguales que 2. Los pesos iguales a 1 corresponden a arcos creados mediante navegación libre.

arco, con una proporción de 10 a 1 respecto a las otras dos opciones (en otras palabras, que la tercera opción es diez veces más importante). Teniendo en cuenta restricciones y probabilidades la secuencia del estudiante es especificada de forma estocástica [149].

El grafo define los posibles secuenciamientos de actividades de forma estocástica, pero su definición es estática (los pesos pedagógicos no cambian); por tanto, puede sufrir los problemas que se han comentado anteriormente. En primer lugar, un error en la planificación hecha por los profesores tendrá un impacto negativo en los estudiantes. En segundo lugar, al evolucionar la población estudiantil con el transcurso de los años (los nuevos estudiantes pueden ser distintos a los anteriores), la planificación puede ser cada vez menos adecuada.

Adaptación de ACO a Paraschool

Para resolver estos problemas, Paraschool toma un enfoque inspirado en ACO. Cada estudiante que se mueve en el grafo es considerado como una "hormiga". Al interactuar con los ejercicios, el estudiante va dejando feromonas en los arcos que tienen una influencia en la probabilidad de que dicho arco sea elegido.

Estas feromonas son de dos tipos: positivas y negativas. Las primeras (ϕ^+) se depositan cuando el estudiante resuelve correctamente el ejercicio. Las feromonas negativas (ϕ^-) se dejan cuando el estudiante falla. Combinando ambas y los pesos pedagógicos W asignados por el equipo pedagógico, se calcula una función de idoneidad (*fitness*) para cada arco (ecuación 5.1).

$$\omega_1 \cdot W + \omega_2 \cdot \phi^+ - \omega_3 \cdot \phi^- \tag{5.1}$$

donde W es el peso pedagógico, ϕ^+ son las feromonas positivas, ϕ^- son las feromonas negativas, y ω_1, ω_2 y ω_3 son pesos que permiten ajustar el sistema.

Al depositarse cada vez más feromonas, la influencia del peso pedagógico va decreciendo. Después de que un estudiante complete un ejercicio, todos los arcos de salida se ordenan en función de su idoneidad y uno es seleccionado en torneo estocástico (*stochastic tournament*) [149]. El estudiante sigue el arco elegido y se le muestra la siguiente actividad.

Para reforzar la noción de camino de aprendizaje, las feromonas se depositan en los últimos cuatro arcos recorridos por el estudiante, como se ilustra en la figura 5.1. Asumiendo que las actividades recientes han tenido mayor importancia en el éxito/fracaso del estudiante en el ultimo ejercicio, los arcos más antiguos reciben menos feromonas que los recorridos después.

Al depositar feromonas en los arcos, los arcos que formen parte de una secuencia exitosa de ejercicios se ven reforzados, mientras que aquéllos que llevan al fracaso se van reforzando negativamente. Con el tiempo, el sistema es capaz de refinar el diseño original e incluso de corregir diseños pedagógicos fallidos.

Evolución del sistema

Los primeros resultados mostraron un comportamiento no esperado. Se vio que los estudiantes acababan los capítulos de forma prematura, realizando pocos ejercicios. Un análisis en profun-

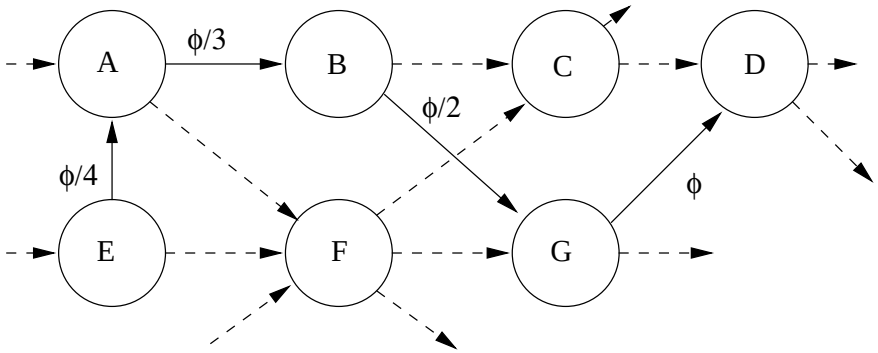


Figura 5.1: Depósito de feromonas y noción de camino: cuando el alumno sigue la secuencia E-A-B-G-D, en los arcos recorridos más recientemente la cantidad depositada es mayor

didad de este comportamiento reveló que el sistema estaba optimizando el camino recorrido por los estudiantes no en términos de aprendizaje, sino de *éxito*. De esta forma, el secuenciamiento óptimo resultaba ser aquél que permitía acabar los capítulos realizando el mínimo número de ejercicios posible, maximizando al mismo tiempo la facilidad de los mismos. Es evidente que el valor pedagógico de este enfoque no es óptimo.

Existen varias opciones para corregir este problema. Una posibilidad es modificar la política de validación de temas. Otra posibilidad es modificar la función de idoneidad para que el proceso de optimización tenga diferentes resultados. Ésta segunda opción fue la escogida por la compañía.

Se decidió que el sistema debía buscar una tasa de éxito en cada ejercicio del 60 %. De esta forma, los estudiantes deben enfrentarse a ejercicios de cierta dificultad, pero ésta no resulta excesiva para no desanimarles. Para ello, se modificó la ecuación de idoneidad como sigue:

$$C + \omega_4 \cdot A + (1 - \omega_4) \cdot P \tag{5.2}$$

donde C, A y P son función del nivel de feromonas en un arco, y C es función también del peso pedagógico. A es la normalización de la tasa de inadecuación I, que mide hasta qué punto la tasa de éxito de un arco se acerca al deseado 60 %, y se define como

$$I = \begin{cases} \omega_2 \cdot \left(\frac{60}{100} - \frac{\phi^+}{\phi^+ + \phi^-}\right) & \text{si } \frac{60}{100} \geq \frac{\phi^+}{\phi^+ + \phi^-} \\ \omega_3 \cdot \left(\frac{\phi^+}{\phi^+ + \phi^-} - \frac{60}{100}\right) & \text{si } \frac{60}{100} \leq \frac{\phi^+}{\phi^+ + \phi^-} \end{cases} \tag{5.3}$$

$$I_{max} = max(\omega_2 \cdot 0,6; \omega_3 \cdot 0,4) \tag{5.4}$$

$$A = \frac{I_{max} - I}{I_{max}} \in [0,1] \tag{5.5}$$

P es el factor de valoración del recorrido (*valorisation du passage*), que da más valor a los arcos que son recorridos más a menudo. C es el factor de confrontación entre equipo pedagógico y

feromonas, que reduce la importancia del peso pedagógico a medida que el sistema se va poblando con feromonas y gana más capacidad para auto-organizarse.

$$P = max\left(\omega_1 \cdot (\phi^+ + \phi^-), 1\right) \in [0, 1] \tag{5.6}$$

$$C = W \cdot (1 - P) \tag{5.7}$$

En la ecuación (5.2) la parte principal[8] es la dualidad entre la tasa de inadecuación normalizada A; y la valoración del recorrido P, que se regula mediante el peso ω_4. El factor C es equivalente a una bonificación para aquellos arcos que han sido diseñados por un profesor. La ecuación (5.2) es la que finalmente se usa en el sistema actual[9].

La evaporación de las feromonas

Los valores de las feromonas decrecen con el tiempo (del mismo modo que las feromonas naturales se evaporan en el aire) para permitir al sistema capacidad de reacción ante los cambios. Las primeras versiones del sistema usaban tiempo natural, lo cual presentaba varios problemas. En primer lugar, era difícil elegir los parámetros precisos de cantidad de feromonas depositadas y velocidad de evaporación, que permitieran al sistema ser sensible a los cambios sin caer en régimen oscilatorio, dependiendo del numero de estudiantes y de su nivel de actividad. En segundo lugar, y aún más importante, la actividad de los usuarios mostraba una gran estacionalidad, con cortos periodos de gran actividad y largos periodos de actividad casi nula (vg. las vacaciones de verano). Esto tenía el efecto de que las feromonas se evaporaban completamente durante los periodos de poca actividad, perdiendo así la mayor parte de las ventajas de la auto-organización conseguida hasta el momento.

Para resolver este problema, se ha empleado una especie de tiempo basado en usuario. En vez de decrecer todos los valores de las feromonas con el tiempo, los valores en los arcos solo son actualizados cuando un usuario pasa por ese arco o por uno vecino. En este contexto, un arco vecino es aquél que muere en el mismo ejercicio. Cuando un estudiante termina un ejercicio, las feromonas positivas o negativas se quedan en el último arco que ha seguido. Al mismo tiempo, los valores de las feromonas del resto de arcos vecinos son reducidas. Esto supone un mecanismo de competición entre los arcos, ya que aquéllos más usados tienen mayor cantidad de feromonas que los que se usan poco; en éstos las feromonas se evaporarán rápidamente[10]. Es evidente que el nivel no cambia si no hay usuarios que atraviesen un arco o sus vecinos (sea porque están de vacaciones, sea porque se mueven en otras secciones del grafo de transiciones, etc).

[8] Por claridad, no se han incluido algunos factores que no son de especial interés para la presente tesis como el factor de agenda, las restricciones inter-temática e inter-curso, y la restricción de nivel. Todos estos parámetros son multiplicativos y tienen un rango de valores discreto en vez de continuo. Responden a particularidades del sistema de Paraschool y su descripción está en [160], páginas 73-76.

[9] Tras el proceso de calibrado, los pesos en Paraschool son $\omega_1 = \omega_2 = \omega_4 = 0,5$, $\omega_3 = -3$.

[10] Valigiani llama *erosión* a este tipo de evaporación basada en el usuario [161].

Altruismo y egoísmo

El trabajo de Valigiani ha mostrado varias diferencias entre la aplicación de heurísticas inspiradas en hormigas a los problemas técnicos (como los ejemplos de la sección 5.1), en los que se optimiza empleando pequeños agentes software, y su aplicación a los problemas sociales, en los que es gente real la que debe realizar dicha tarea.

En primer lugar, los estudiantes no son intrínsecamente altruistas como es el caso de las hormigas y no se sacrifican voluntariamente por el bien de la comunidad. Un estudiante no acepta perderse en el grafo de ejercicios simplemente para que su exploración beneficie a los demás estudiantes. Todos los estudiantes quieren aprender; no les interesa optimizar los resultados del grupo sino únicamente los suyos propios. Esto implica que hay aspectos de las heurísticas de optimización por colonia de hormigas que no pueden ser usados o que no funcionan como ocurre en otros escenarios.

Además, hay que tener en cuenta que el proceso de aprendizaje debe ser optimizado para cada estudiante. El hecho de que un camino pedagógico sea óptimo para un grupo no significa que lo sea para cada uno de sus miembros. Para dar un nivel adicional de adaptación a cada estudiante, el sistema de Paraschool usa *feromonas personales*. Estas feromonas son depositadas por cada usuario mientras viaja por el grafo, y evitan que el alumno repita los mismos ejercicios de forma continua. . Este tipo de secuencia repetitiva no es recomendable por dos motivos. En primer lugar, puede resultar aburrida para el estudiante. En segundo lugar, si el mismo ejercicio se repite demasiadas veces, un estudiante puede tener éxito en el mismo aunque no haya realmente aprendido lo que debiera.

Las feromonas personales son depositadas por el uso y se evaporan en tiempo natural. Son un factor multiplicativo de la función de idoneidad: cuando se depositan valen cero, y tienden a uno al evaporarse (nótese la diferencia con las feromonas sumativas clásicas, que se evaporan hacia cero). Se pueden ajustar para impedir que el mismo ejercicio sea repetido en el periodo de tiempo que el administrador del sistema considere más adecuado para el aprendizaje. En Paraschool este periodo es de aproximadamente una semana[11].

Navegación libre

Una característica muy interesante del sistema es su habilidad para crear nuevos arcos que conecten los ejercicios, además de aquellos que fueron creados por el equipo pedagógico. Para entender este proceso hay que conocer la diferencia entre navegación guiada y libre. En el sistema de Paraschool, los usuarios pueden navegar de forma guiada (el modo que se ha explicado) o libremente. Esto último significa que los estudiantes pueden acceder a cualquier ejercicio de los presentes en los cursos en que están inscritos, usando los enlaces de que disponen en su historial y en la tabla de contenidos de cada curso. Cada vez que esto ocurre, el sistema almacena la transición y crea un nuevo arco que nace en el último ejercicio en el que estuvo el estudiante y muere en el siguiente. Desde ese momento, el arco se considera como un arco normal del grafo; como tal, puede seleccionarse en la navegación guiada, recibir feromonas, etc. Éste es un rasgo muy interesante que permite encontrar secuencias que el equipo pedagógico no había pensado en primera instancia. Analizando el sistema se apreció que el número de arcos creados por los propios

[11] En los resultados de la sección 9.1 se concluye que este tiempo es excesivo

estudiantes excedía claramente al número de arcos creados por los profesores: la media de arcos que salían de cada ejercicio era superior a 25, mientras que la media de arcos de profesor era de aproximadamente 2 ([160], página 41).

Otra ventaja de la navegación libre es que abre las puertas a añadir nuevos ejercicios al sistema aunque el equipo pedagógico no cree un grafo para ellos. Como las conexiones entre los arcos aparecen en la navegación libre, el algoritmo expuesto asignará una idoneidad a esos arcos con el tiempo, al irse depositando feromonas por las transiciones de los estudiantes. Al final el nuevo grupo de ejercicios se auto-organizará sin intervención directa externa. De hecho, este comportamiento ha ocurrido ya ([160], páginas 55 a 57) para algunos ejercicios que, por error en la planificación, estaban aislados en sus correspondientes grafos.

5.3.2. Learning Networks

Las redes de aprendizaje (Learning Networks, LN [97]) son sistemas educativos flexibles orientados a cubrir las necesidades de los estudiantes en varios niveles de competencia a lo largo de su vida. Permiten acceder de forma ubicua a sistemas educativos en el trabajo o desde casa.

Las LN consisten en eventos de aprendizaje (llamados nodos de actividad) en un dominio dado. Estos nodos pueden estar asociados a cualquier elemento que respalde el aprendizaje, desde un recurso web hasta un curso o un taller (*workshop*). Tanto los proveedores como los usuarios pueden crear nuevos nodos de actividad o adaptar los ya existentes (incluso borrar nodos). Por tanto, una LN representa un grupo de nodos de actividad, que está siempre cambiando y que proporciona actividades educativas a los estudiantes de distintos proveedores y con diferentes niveles de dificultad.

En una LN, los usuarios tienen un *objetivo*. Éste es la descripción del nivel de competencia que quieren alcanzar en un dominio en particular. Para alcanzar sus objetivos, los estudiantes pueden seguir una ruta o —a veces— varias. En otras palabras, hay una serie de selecciones y de secuencias de nodos de actividad que deben seguir para alcanzar sus objetivos. Además, se describe a los usuarios en una LN por su *posición* y su senda educativa (*learning track*). La senda educativa se define como el conjunto de nodos de actividad que ya han sido completados. La posición educativa incluye la senda así como todos los nodos de actividad que se pueden considerar como completados, bien porque están relacionados con algún nodo en la senda, bien por un estudio o experiencia laboral previos. Determinar la posición educative es un problema complejo que es estudiado en [165] junto a diversos enfoques para tratarlo.

Como es esperable en cualquier sistema orientado al aprendizaje a lo largo de la vida (*Life Long Learning*), las LN pueden dar acceso a un gran número de actividades y recursos. Esto conlleva una gran complejidad que estorba el progreso de los alumnos: produce que encuentren difícil conseguir una visión general del numero de módulos y la mejor manera de estudiarlos. Para ayudar a los usuarios en el proceso cognitivo de toma de decisiones necesario para elegir y secuenciar sus eventos de aprendizaje, Tattersall et al [158] se inspiran en el proceso que lleva a las hormigas a optimizar el camino de su nido a una fuente de comida interactuando indirectamente usando feromonas.

El sistema permite a los estudiantes elegir a partir de una lista de objetivos de aprendizaje en la LN. También permite identificar la ruta hacia ese objetivo. Las interacciones de los usuarios

	A	B	C	D	E	F
A	·	5	2	1	2	2
B	3	·	4	3	3	1
C	2	2	·	5	2	1
D	0	1	4	·	1	0
E	0	1	3	3	·	1
F	0	0	2	5	0	·

Tabla 5.1: Ejemplo de matriz de transiciones en LN

se guardan en un listado histórico, incluyendo información del usuario, el nodo de actividad, una marca temporal y una indicación del rendimiento. Las interacciones con los recursos y las actividades se guardan automáticamente a la vez que el usuario progresa en un conocimiento particular. El marcado temporal de esas interacciones permite identificar las secuencias. Algunas de las técnicas usadas para encontrar esas secuencias se tratan en [120]. Esas secuencias pueden ser procesadas y agregadas para derivar una ”fuerza de la intensidad de feromona”, con lo que se favorecen los caminos donde más estudiantes hayan tenido éxito.

Usando la información de las sendas de todos los usuarios se puede calcular una matriz de transiciones [40] que indica, para cada nodo de actividad, cuántos estudiantes han progresado con éxito de este nodo al siguiente. Esta información se le ofrece a otros estudiantes dándoles una guía para su navegación de forma indirecta.

Cuando un usuario acaba de completar un nodo de actividad, el algoritmo para seleccionar el nuevo nodo de actividad tiene tres pasos principales. Primero, el grupo de nodos de actividades a completar debe ser calculado. Después, la información del éxito para cada transición (empezando en el nodo actual y terminando en cada nodo) se recupera de la matriz de transiciones. A continuación, usando esas informaciones como pesos, se elige aleatoriamente la siguiente actividad.

La tabla 5.1 muestra un ejemplo. Imagínese que un estudiante acaba de terminar el nodo de actividad A. Para conseguir su objetivo, tiene que completar los nodos B, C, D y F. A partir de la matriz de transiciones, se puede saber cuántos estudiantes han ido de A a cada uno de esos nodos de actividad y los han completado con éxito. La definición de éxito puede variar con los nodos: puede tener calidad buleana (vg. responder correctamente una pregunta) o una medida de varios parámetros (vg. responder correctamente al 60 % de las preguntas y hacerlo en un tiempo menor que cierto umbral, etc). Los datos de la matriz dicen que 5, 2, 1 y 2 estudiantes completaron con éxito los nodos B, C, D y F respectivamente. Por tanto, el próximo nodo de actividad que se recomendará para el estudiante será el B con un 50 % de probabilidades, el C con un 20 %, etc. El resultado es que el camino que se recomienda más comúnmente es el que tiene más posibilidades de ser elegido. Para prevenir convergencias no óptimas a este camino (estancamiento) hay una posibilidad de que otros caminos sean elegidos.

Esta ayuda a la navegación está diseñada para facilitar las decisiones y reducir el riesgo de saturar de información al alumno, ofrecer información accesible. Esta información está centrada en el usuario, y tiene relación con su posición actual (su nivel de conocimiento en la red de aprendizaje). Como la realimentación utiliza estimaciones de éxito, ayuda a los estudiantes a realizar mejores elecciones basadas en secuencias ya intentadas y probadas por sus iguales.

Es importante notar que en este sistema no hay un proceso explícito de evaporación. Una vez que se detecta una transición exitosa entre dos nodos de actividad, la información se queda permanentemente el sistema. Esto puede llevar a una situación de estancamiento: si un determinado camino consigue demasiadas feromonas, el sistema recomendará dicho camino en la mayoría de las ocasiones, lo cual hace que el sistema pierda capacidad de reacción. En la sección 9.3.1 se tratan en más detalle estas limitaciones.

5.4. Filtrado colaborativo

A lo largo de los últimos años, el auge del comercio electrónico ha estimulado el desarrollo de sistemas de filtrado colaborativo como sistemas recomendadores. La meta de un sistema de filtrado colaborativo puede definirse como la predicción de la utilidad que va a tener un determinado elemento para un usuario conocidos sus gustos pasados y la opinión de otros usuarios que piensan de forma similar. El filtrado colaborativo se basa en la premisa de que los usuarios que han tenido gustos similares en el pasado valorarán de forma similar los mismos elementos en el futuro. Así, estos sistemas almacenan preferencias sobre diversos elementos (música, libros, etc) y usan dicha información para ayudar a futuros usuarios en sus decisiones. En cierto sentido, los sistemas de filtrado colaborativo son organizadores de conocimiento: las preferencias de los usuarios y su procesamiento posterior crean los esquemas de clasificación.

5.4.1. Aplicaciones fuera del elearning

Utilizando una clasificación sencilla, los sistemas de filtrado colaborativo se pueden clasificar en sistemas basados en memoria y basados en modelos. Los primeros emplean una base de datos de usuarios y elementos para generar una predicción. Estos sistemas usan técnicas estadísticas para hallar un conjunto de usuarios (vecinos) que tienen un perfil similar y que concuerda con el del usuario objetivo [134]. Los algoritmos de filtrado colaborativo basados en modelos crean en primer lugar un modelo de valoraciones de usuario. Este tipo de algoritmos toman un enfoque probabilístico y conciben el filtrado colaborativo como el cómputo del valor esperado de una predicción para el usuario dadas sus preferencias anteriores en relación con otros elementos. El proceso de creación del modelo se puede llevar a cabo por diversas técnicas entre las que destacan las redes bayesianas [119], el análisis semántico latente [77] y las estrategias basadas en reglas [15].

Ha habido varios sistemas de filtrado colaborativo diseñados específicamente como asistentes del proceso de aprendizaje, como PHOAKS (www.phoaks.com) o LON-CAPA (loncapa.msu.edu). Sin embargo, el más interesante para el ámbito de esta tesis es el sistema CoFIND de John Dron [49]. No sólo es un sistema de filtrado colaborativo orientado a la enseñanza[12], sino que fue diseñado para producir procesos de auto-organización y estigmergia a medida que el enjambre de estudiantes interactuara con él.

[12]En concreto, Dron pretende emular con su sistema varias de las “funcionalidades” de un profesor [49], páginas 88-98.

5.4.2. CoFIND

CoFIND es un sistema de filtrado colaborativo que intenta ayudar a los estudiantes a generar y usar una lista de recursos educativos [51]. Los recursos deben ser encontrados por los alumnos, y se organizan después de forma automática. Esta organización no es el resultado de la aplicación de unas reglas pre-definidas en la herramienta por un diseñador o un equipo pedagógico, sino que es el resultado de las interacciones de los estudiantes con los recursos que ellos y sus compañeros han encontrado, y entre ellos. Como la mayoría de los sistemas de elearning modernos, CoFIND se ha desarrollado como una aplicación web.

CoFIND busca facilitar la comunicación entre los aprendientes (sea esta comunicación directa o indirecta) y al mismo tiempo proporcionar información como un experto en una determinada materia. Esta información se estructura de forma que es mejor asimilada por los estudiantes. Para alcanzar dicha meta, el sistema emplea los votos que los usuarios dan a los recursos (y sus propiedades) así como el uso que dichos usuarios hacen de los propios recursos.

Hay cuatro conceptos importantes para entender CoFIND: recursos, propiedades (*qualities*), votos (*votes*) y temas (*topics*). Los recursos son identificados exclusivamente por una URL[13]. Tanto las propiedades como los temas sirven para describir los recursos. Las propiedades son etiquetas de metadatos que indican la opinión de un estudiante sobre un recurso y pueden escogerse de una lista —o crearse nuevas—. Los temas proporcionan un agrupamiento temático de los recursos.

Con el objetivo de auto-organizarse, el sistema adopta un enfoque evolutivo en el que los recursos deben competir entre sí. Los más aptos (esto es, los más útiles para los alumnos) sobreviven y los demás desaparecen.

Al igual que ocurre con los recursos, los estudiantes crean las propiedades del sistema. Aunque las propiedades suelen ser adjetivos, no hay restricciones al respecto: “útil”, “bien escrito” o “bajo nivel” son posibles propiedades, pero también lo es “bonitas letras amarillas”. Un esquema evolutivo similar al de los recursos evita que el sistema se inunde con propiedades inútiles: una propiedad desaparece del sistema si no es usada durante un periodo de tiempo. Las propiedades que son verdaderamente útiles para el conjunto de los estudiantes son utilizadas y votadas (*rated*) a menudo, y aparecen en los primeros puestos de la lista de selección. Las propiedades de utilidad marginal son desplazadas hacia abajo en la lista hasta que finalmente desaparecen.

En las primeras versiones de CoFIND, las propiedades desaparecían del sistema si no se usaban durante una semana. Sin embargo, al igual que sucedía en el caso del sistema de Paraschool (sección 5.3.1), se comprobó que el tiempo natural presentaba problemas. Si el sistema no se utilizaba durante una semana, todas las propiedades desaparecían. Esto hacía difícil presentar el sistema en congresos, e incluso reutilizar el conocimiento adquirido de un grupo de estudiantes con otro grupo diferente en cuanto había un periodo de vacaciones entre ellos.

Para solucionar este problema las versiones posteriores de CoFIND utilizan un sistema de pesos que equivale a un tiempo basado en usuario. Las propiedades nuevas entran en el sistema con un peso equivalente a la máxima votación dada a las propiedades presentes en el mismo, y cada vez que un usuario escoge una propiedad para describir un recurso el peso de dicha propiedad aumenta en uno. Por otro lado, cada vez que un usuario se conecta al sistema, el peso de todas las

[13] Versiones posteriores de CoFIND han permitido incluir recursos como libros o películas, que no son identificables por una URL. En todo caso, esto no afecta a los aspectos de auto-organización y filtrado del sistema.

propiedades se reduce en uno. Durante el funcionamiento normal del sistema, las propiedades que no se utilicen sufrirán una disminución progresiva de sus pesos hasta desaparecer. Sin embargo, los periodos en que la actividad del sistema disminuya (vg. vacaciones de verano) el estado no se modifica.

El interfaz de elección de propiedades fue modificado en versiones posteriores para diferenciar entre dos tipos de información: peso y utilidad. La utilidad es el número de recursos que han sido descritos con una propiedad o, en otras palabras, la cantidad de recursos que se pueden encontrar filtrando por esa propiedad; las propiedades más útiles se muestran con tipos de letra más grandes. El peso refleja si la aplicación ha sido utilizada recientemente; las propiedades con pesos mayores aparecen en los primeros puestos de la lista, y también las propiedades de nueva creación. De esta forma el usuario puede elegir las propiedades en función de su frescura (posiciones altas de la lista) o de su ámbito (tamaño mayor de letra).

Es importante distinguir que la utilidad de una propiedad es un conocimiento explícito, pero el peso supone un conocimiento implícito generado a partir de las estadísticas de uso del sistema. También conviene aclarar que una propiedad por sí sola no describe un recurso, pues puede darse el caso de que la propiedad tenga muchos votos negativos. Cuando los usuarios buscan recursos en el sistema en función de una propiedad, tienen la oportunidad de mostrar su conformidad con la selección obtenida. Esto se lleva a cabo mediante un proceso de votación en el que, para cada recurso obtenido, se evalúa la adecuación del resultado. Así, si un alumno busca por “adecuado para principiantes” y obtiene un recurso de mucha complejidad, le asignará una votación negativa.

Esta votación afecta al modo en que los recursos se muestran por pantalla: aquéllos con una votación media más alta aparecen primero. En caso de empate, se da preferencia a los que han sido votados más veces.

Por último, el cuarto concepto importante en CoFIND son los temas, que fueron añadidos en las últimas versiones del sistema debido a la necesidad de crear cierta compartimentación, tanto entre recursos como entre propiedades. Tanto unos como otras se pueden clasificar por temas y la carencia de esta distinción produce una merma en las prestaciones del sistema. Por ejemplo, la búsqueda de recursos relacionados con “puentes” debería producir resultados diferentes para un ingeniero telemático y otro de caminos.

Los temas son introducidos por los propios usuarios, como ocurre con recursos y propiedades. El interfaz para hacerlo consiste una división en cuadrantes, en los que los usuarios pueden añadir los temas que consideren oportuno. Esto permite que las relaciones inter-temáticas se reflejen en forma de agrupamiento espacial[14]. Por otra parte, los nombres de los temas que están en el mismo cuadrante compiten por el espacio. Cada vez que se escoge un tema, su tamaño de letra aumenta, a la vez que disminuye la de sus vecinos. El aumento es proporcional al número de vecinos (más vecinos implican un cambio más acusado), mientras que la disminución es proporcional a su inverso (más vecinos implican menos disminución); esto se hace para evitar situaciones de estancamiento que se producían en las pruebas preliminares. El resultado de este proceso supone un cierto grado de estigmergia, donde los estudiantes eligen los caminos más populares incrementando el éxito de los mismos; se amplifica así el patrón de comportamiento del grupo para producir una estructura en el material almacenado por el sistema.

[14]Un esquema similar se usa en el ITS descrito en [48], aunque en este caso, la agrupación no se hace de forma distribuida sino centralizada.

De los estudios realizados por Dron [49] deben extraerse varias lecciones. La primera de ellas es el compromiso a hora de establecer el tamaño de la comunidad que puede beneficiarse de un sistema como CoFIND. Un tamaño mayor será útil —en teoría— a más gente. Sin embargo, un tamaño pequeño seguramente compartirá un conjunto más homogéneo de valores e intereses y esto permitirá que el sistema evolucione más rápido, lo cual está en concordancia con los resultados de las ciencias naturales que muestran que la evolución se acelera en poblaciones aisladas [68]. Esto permite desarrollar un mecanismo útil para reducir el exceso de resultados que se obtiene de un directorio o motor de búsqueda típico. Si un grupo pequeño y con un interés educativo común compila un conjunto de recursos (posiblemente hallados usando motores de búsqueda), existe una alta probabilidad de que obtengan unos recursos que sean más relevantes para ellos a partir de los que ha hallado cada uno. El modelo evolutivo de CoFIND crea una taxonomía dependiente del contexto que captura el uso de las evaluaciones sobre los recursos, las cuales han sido negociadas por el grupo tácitamente. Los ambigüedades y los conflictos no se discuten, sino que se solucionan a medida que los usuarios votan y califican los recursos. Las últimas versiones de CoFIND incluyen un pequeño chat en el que los usuarios pueden comunicarse directamente entre ellos mientras interaccionan con el sistema[15].

No hay evidencia, dice Dron [49], de que la división en cuatro cuadrantes de los temas siguiera realmente una estrategia de agrupación por temáticas similares. Esto puede ser debido a un mal diseño del interfaz o a problemas de diferencias en el uso de categorías [99] por parte de los diferentes usuarios.

[15] Ésta funcionalidad está presente en muchos sistemas que hacen énfasis en la conciencia social (*social awareness*) del alumno, como iHelp [19].

Part II

Contributions / Contribuciones

6 SIT: A System for Intelligent Tutoring

> There is maybe nothing new in the LO thinking. We already have the Web where content is as reusable and modular as it can be.
>
> – Teemu Leinonen [103]

 SIT son las siglas en inglés del sistema para tutoría inteligente; es una plataforma modular que permite construir sistemas de tutoría inteligente (ITS) sobre ella. Estos sistemas deben estar enfocados a la adaptación del secuenciamiento, y obtienen el material educativo que secuencian de cualquier lugar de Internet. Este capítulo presenta la arquitectura de SIT y sus protocolos de comunicación; también se muestran las partes que se requieren para crear un ITS. La sección 7.4 describe dos ejemplos de ITS construidos sobre SIT.

 SIT stands for System for Intelligent Tutoring. It is a modular platform to build intelligent tutoring systems (ITS) with it. These tutoring systems must be focused on the adaptation of sequencing, and the learning content they sequence can be withdrawn from anywhere in the Internet. This chapter presents the architecture of SIT and its communication protocols. The parts needed to create an ITS are showed. Examples of ITS created for SIT are described in Section 7.4.

6.1. Background

Originally, SIT was created as a tutoring system for the domain of computer architecture with a focus on sequencing adaptation. When another ITS was created for an operating systems course [108], it was decided to reuse all the administrative aspects of the original tool. They were separated from the tutoring aspects (i.e. domain, instructional and user model [73]) and the actual content. This modular architecture was one of the main features of SIT version 2, which was used in the experiments described in Section 7.4.

SIT version 2 followed a deterministic paradigm, in which the sequencing of learning units was adapted to the student without any direct intervention on his/her part. Although this allowed for more precise measurements in the experiments performed, more freedom of choice was desirable. The platform was modified so it could give several options to the student. The student had some degree of freedom and felt some degree of control over the tutoring process; this has been reported as having a positive impact on learning [151]. The result was called SIT version 3.

The open choice paradigm of SIT version 3 opened the door to other improvements. A swarm paths module was developed to track student success over different sequences of activities. This information could be fed back to other students, giving them some additional information about their tutoring process collected from their peer stories of success or failure, thus providing an additional level of adaptation.

Finally, SIT incorporated another functionality in its last version to date (SIT version 4). Former versions of SIT worked only with simple activities, that is, activities associated to atomic resources; it allowed for the reusing of content as well as the reusing of sequencings. Next step was to reuse other applications, incorporating them as activities in the tutoring process for taking advantage of their achievements. A protocol, based on the work by Guzmán and Conejo [71], was defined for information interchange with those applications, so that the student could change from SIT to other platforms as transparently as possible.

6.2. Objectives

The main objective of SIT is to be a platform for the development of Intelligent Tutoring Systems. The platform is responsible of all administrative tasks (from managing user accounts to retrieving and forwarding learning content to users), thus alleviating a big load of work from the tutoring system developer, who is able to concentrate on the pedagogical [56], cognitive [109] and instructional [31] aspects of the system. It has been showed [123, 181] that the other tasks amount to more than half of the resources (e.g. time, developers) dedicated to the creation of an ITS. The goal is to allow teachers and designers to focus on learning and not on technology.

SIT is oriented towards activity *sequencing*. The main interest is to adapt a sequence of learning units to the student. The assumption is that the student learns more efficiently if the sequence of activities or resources is adapted to his/her personal characteristics (e.g. learning style, current expertise on the domain) as explained in Chapter 3. The adaptation considered in the tool is done at the level outside the resources. No adaptation is done within the resources, they are taken 'as is'. SIT is a research prototype and this feature is intentional: in order to assess the fitness of a sequencing strategy through experiments, all the other parameters must remain fixed. Once the suitability of the approach has been tested[1], the reasonable next step is to combine both approaches for making better tutoring tools.

As a feature related to this focus on sequencing, SIT is designed to be able to use any learning resource available through the Internet. The Internet is full of learning resources, some of them created by teachers, some of them with a collaborative origin (e.g. the Wikipedia —wikipedia.org—). They are created with different purposes, from academic to commercial; can be complex, like a

[1] See section 7.4

web page, or simple, like the images on the web page; they are usually static, but sometimes have the ability to change (e.g. active pages) or evolve over time (e.g. wikis). Sometimes the resources are under restrictive copyright licences, but others they are part of the public domain or may be licenced under open licences like Creative Commons [104, 105]. In the end, all those resources can be used by a teacher apart from those he/she might produce on his/her own. SIT tries to support this behaviour identifying all resources by an Uniform Resource Locator (URL). Therefore, the designer of a tutoring system may reference any resource on the Internet (with some constraints, as explained afterwards) and the platform retrieves it and forwards it to the corresponding user.

There are two design goals that are pursued by orienting to URL-named resources. First, this facilitates reuse of learning content on the part of the ITS designers. A significant amount of web content is structured enough to be used as is. Designers should not be obliged to an unnecessary repetition of efforts every time they create a tutoring system if there is a lot of content already available for many situations. Besides, referencing content by URL emphasizes the conceptual separation between content, on one side, and pedagogy and instruction, on the other. Even if the designer creates all the content by him/herself and it is placed in the same physical machine as the platform, it is *logically* separated. This separation is necessary in the development of efficient tutoring systems. In the words of Murray:

> *Understanding ITS authoring requires a conceptual separation of content from instructional method —a reconceptualization of content as flexible and modular* [123].

Retrieving the content from the Internet is as flexible as modular as can be, without losing perspective or getting lost in cyberspace, thanks to the sequencer modules.

Finally, SIT is a web-application. Everything the user needs to interact with the platform and its ITS is a web browser. Following the distribution philosophy presented in Chapter 2, its has been designed to be modular: the platform is designed as a series of different modules that communicate through clear interfaces. This provides several advantages to the platform. First, it is more robust, as the different modules are easier to code, maintain and debug. Second, it is more scalable, because new modules can be added with a minimal negative impact on the system, providing new functionality; one example is the SPI module, presented in Section 6.3.7. Specially important is the fact that it is easier to deploy different sequencing strategies when designing an ITS on top of SIT, as is explained in the next section. The creation of a new tutoring system only needs to implement a simple interface for communicating with the rest of the system, thus the creator can concentrate on the instructional design and the content while SIT is responsible of all logistic tasks, from managing of users' accounts to communication and distribution of the learning resources.

6.3. Architecture

This section describes the architecture of SIT. The general block diagram is first described, and then the issues of ITS creation and resource managing are explained. Finally, the three modules that support the tutoring process but are not part of it (administrative unit, manager module and Swarms Paths Information module) are presented.

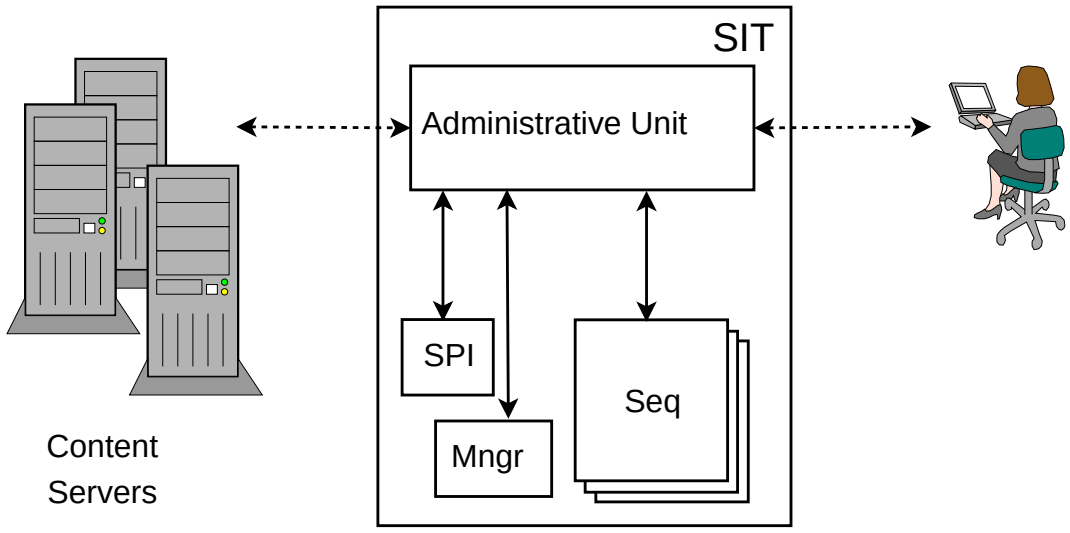


Figure 6.1: General architecture

6.3.1. General architecture

The general architecture of the platform is depicted in Figure 6.1. The architecture is oriented to the reuse of learning resources already available on the Internet. These resources are showed to the student using a web browser, and they can be either static (e.g. images, static HTML pages, etc) or dynamic (e.g. JSP pages, etc) content. Dotted lines indicate communication over HTTP.

The resources are organized within the tool as *learning modules*. These are defined as a collection of units (i.e. resources) with one or several learning goals. It should be noted that the term "module" has a different meaning in "learning module" and in "modules of SIT". The latter is a piece of software that integrates in the SIT structure. Examples of this kind of modules are the sequencers, or the Swarm Paths Information module described in Section 9.3. The former is an organized collection of learning resources; it is typically composed of an XML file where the corresponding resources are referenced, the instructional strategy is represented using different kinds of rules and/or models, and the generic part of the user model is sometimes represented.

6.3.2. Building tutoring systems with SIT

A tutoring system is produced by the combination of a sequencer and a learning module. The syntax of a module description depends on the sequencer used to process it (see section 7.3 or appendix C for two different examples of sequencers for SIT). Different sequencers may share an entire learning module description or parts of it.

The module references the resources that are needed, describes the pedagogical rules that control the sequencing of the resources and may also characterize the general part of the student model. Resources are described by a URL. Learning modules may reference resources directly or other learning modules, as long as they are manipulated by the same sequencer. Referencing of learning modules that are to be interpreted by a different sequencer is not supported in SIT (unless decoupled activities are used, see Section 6.4).

The sequencer module takes all the information from the learning module and builds the student model for each user. It is the responsibility of the sequencer to receive all data produced by the student from the Administrative Unit (e.g. answers to exercises, time spent performing an activity, selections made by the student, etc) and decide what activity is the best suited for him/her, giving this information to the Administrative Unit, so it can forward the corresponding resource to the student. Besides, the student model has to be updated after each iteration of the SIT protocol by the sequencer.

The platform allows for multiple sequencers to coexist. Each one is expected to be oriented towards different sequencing strategies, based on specific pedagogic rules and/or student modeling schemes. This brings one of the advantages of SIT: Using only one platform, two different tutoring systems can be easily compared, maybe even using the same resources. This is equivalent to compare two pedagogic strategies, two user modelling designs, two domain models of the same domain, etc, depending on the specific ITS tested. The only requirement to perform such an experiment is to create two sequencers, each of them implementing one of the two aspects of tutoring to be compared, and then run them in parallel: two groups of users are created by SIT, and the platform takes care of all logistic processes, from user accounts to forwarding of resources. After the experiment has finished, conclusions can be withdrawn about the fitness of both aspects. This feature of SIT was used to test the validity of sequencing graphs as an effective way to express adaptive sequencings of learning content, as it is described in Section 7.4.1.

6.3.3. Resources

Resources are designated by an Uniform Resource Locator (URL). Information about which resource is to be given to the student at every moment is given to the Administrative Unit by the sequencers, and the Administrative Unit retrieves them from the Content Servers and forwards them to the user.

Wrapping

In order to be useful in a SIT environment, the resources have to be wrapped up appropriately. As resources may or may not have been designed for their use in a SIT environment, some additional information needs to be given to the web browser of the student.

Resources are wrapped in an HTML form. Thus, they are showed to the student as a regular web page including two additional buttons with labels “Advance” and “Logout”. The former instructs the tool to decide which resource is to be showed next, while the latter stores the current state and terminates the session. Students may come back to continue with the activities at any time. Figure 6.2 shows a typical screen shot.

The “Advance” button is linked to the address of the SIT Administrative Unit servlet (this behaviour changes for the case of decoupled activities, see Section 6.4). When the “Advance” button is pressed, the first step of the SIT protocol (Section 6.4) is fired.

Figure 6.2: A learning unit with the Advance and Logout buttons

Content servers

Resources are retrieved from content servers. As resources are only specified by a URL, content servers may be a set of remote machines anywhere in the Internet, or a private one in the same LAN as the main SIT server. Any resource that can be accessed (i.e. downloaded) through a URL is suitable to be used.

Depending on the specific type of the resource, it may have to be slightly modified before being retrieved from the content server and sent to the user. First, it is important to note that in the case of dynamic web pages (PHP, JSP, etc) only the resulting HTML code for the HTTP request can be downloaded. Besides, for complex resources (e.g. HTML pages) some adaptation may have to be performed so that the resource is correctly presented on the student's web browser (e.g. relative links to images may need to be converted to absolute ones, other buttons pointing to different servlets may have to be removed, etc).

During the development of this thesis, all the tutoring systems used in the experimental phase of the research (Section 7.4) referenced resources that were mirrored in the LAN, in order to avoid broken links and similar problems. However, tests were made using resources from external servers to assess the robustness of the implementation.

Interface

Most tutoring systems are tightly integrated with the learning resources they use. These resources cannot be used by other ITS, nor can they use different resources. On the contrary, SIT

focuses on reusing learning resources that may be available anywhere on the Internet. Some mechanism had to be defined to give data to some resources (i.e. active pages). Additionally, there was the need of retrieving data from the resources, so it could be used by the sequencers to update student models and take decisions (i.e. find the next activity to be sequenced).

From the several possibilities analyzed, it was decided to take a pragmatic low-level approach, that was as easy and general to use as possible: data is exchanged through the HTTP request. As all transactions in SIT are performed over the HTTP protocol, this solution was always available, and it required almost no intervention on the resources.

Data is transferred to active pages by including it in the URL that is used to call them[2], like in

$$host : port/page\_name?param1 = value1\&param2 = value2...$$

This method can directly be used by the sequencers and the corresponding learning modules. Resources export data in the same HTTP request when the user presses the "Advance" button. There are cases in which no data is necessary (e.g. images alone), and most interactive resources are implemented as an HTML form. In the latter case, the values collected by the form are present in the request, and the Administrative Unit can forward them to the corresponding sequencer to be processed. This solution is robust and easy to use. It has been showed that pragmatic, robust and easy to use solutions have greater chances of success, in spite of eventual failures, or even lack of complex functionality [150].

6.3.4. Administrative Unit

The Administrative Unit is the central part of the system. It is the communication hub between the user, the sequencer module(s) and the content servers. This unit takes care of all the details of the HTTP communication such as sessions, requests, etc. It is also responsible of forwarding the learning units (i.e. resources) specified by the sequencers to the user, and collecting all the data to feed the sequencers.

No other module is responsible of communication procedures outside the SIT limits. They only have to communicate with the Administrative Unit; it performs the required communication tasks. It can be seen that the Administrative Unit is a single point of failure in the architecture[3], as it serves all the other parts of SIT. Therefore, its implementation has been the most careful.

The Administrative Unit holds three tables in the database to perform its tasks. These tables store information about the users of the system, the learning modules they are following, the SIT modules involved in the process, etc.

Table Users Stores data about the users (i.e. login, password, profile).

[2] In a sense, this means some degree of *content adaptation* that is being performed indirectly by the sequencer, i.e. the learning module designer.

[3] This central role is equivalent to that of the process manager in TANGOW [26].

Table Sessions Stores data about the sessions already open in the system. It must be noted that a session in this context runs from the point at which a student starts a new learning module until the moment that module is finished. This table stores the following information:

- which user is running which learning modules.
- which sequencer is used for each pair student-module.
- which was the last learning unit delivered.
- any other additional data needed by the sequencer (from parameters needed for the learning units to a dump of the current user model in case the session is in standby —offline— state).

Table Historical Stores historical information about the platform (i.e. users, modules, events, timestamps, data sent or received). This table can be analyzed[4] to extract information about users' behavior when interacting with the system, the learning units they went through (and how they performed), etc

6.3.5. Manager module

The Manager module (Figure 6.3) substitutes the Sequencer when an administrator logs in. Its mission is to perform all the administrative tasks from the point of view of the human user, such as:

- Opening and closing of accounts
- Password assignment
- Authentication
- Assignment or change of roles in SIT (i.e. student, teacher, admin, etc).

6.3.6. The Sequencer interface

The sequencer is the most important part of the SIT architecture. As it has been explained, several sequencers can be used at once for supporting different strategies, comparing different sequencers' results, etc. Sequencers communicate with the Administrative Unit through the Sequencer interface[5]. Any sequencer to be added to SIT must comply with this simple and generic set of methods. Apart from that, there are no restrictions on the complexity or any other characteristic of the sequencers. Two sample sequencers are described in Sections 7.3 and C.2.

[4] A study using data mining techniques is currently being held with the collaboration of professor Cristobal Romero (University of Córdoba), although it is still the preliminary stages of research at the moment of writing this thesis.

[5] Although SIT is implemented in Java and the Sequencer is implemented as a Java *interface* (purely abstract class), the term in used in a general sense and it could be implemented in any programming language (see Section 11.3).

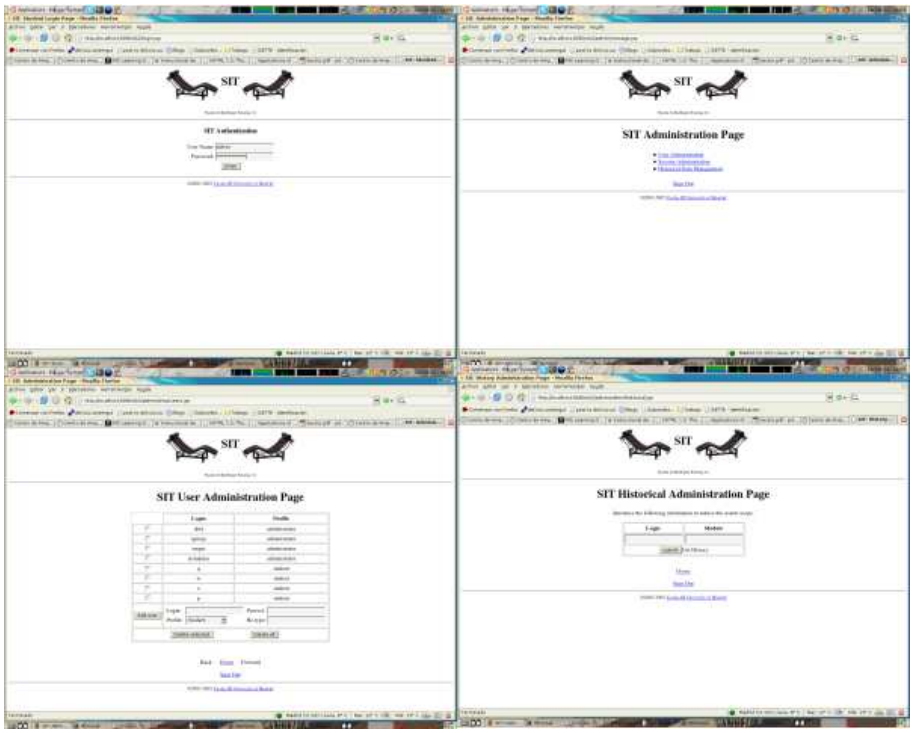


Figure 6.3: The Manager module interface in SIT

SIT version 2

SIT version 2 was the first version of SIT to separate the structure in modules. The Sequencer interface had three functions: init, close and process.

Init This function is called by the Administrative Unit the first time that the sequencer is used. When a user logs into the system, the Administrative Unit checks which learning module has been selected, and which sequencer has been set for that pair student-module (this information is set by the administrator). The sequencer is activated (i.e. instantiated) and the *init* function is executed.

The goal of this function is initializing the sequencer: building networks, maps or models; retrieving historic data from databases, etc.

Close This function is called by the Administrative Unit before disconnecting. When SIT is turned off in a sorted way, the *close* function is executed for all working sequencers.

The goal of this function is allowing the sequencer modules to store data for future use without having to dump their memory periodically. Sequencer modules are thus allowed to retain all data in memory until the *close* method is called with no risk of data loss.

In a production environment, the *close* method plays another role. If sequencers use an excessive amount of resources, it is the responsibility of the Administrative Unit to close those of them that are seldom used. The *close* method is necessary in such scenario.

Process This is the main function of this interface. This function is executed by the Administrative Unit every time the user finishes an activity. Through the execution of this method, the Administrative Unit passes to the sequencer all the information it has collected from the activity (from the HTTP request, see above). The sequencer, in return, is expected to return the URL of the resource associated to the next activity to be performed by the student.

SIT version 3

The goal of the development of SIT version 3 was giving some freedom of choice to the student, changing the deterministic paradigm of the former versions. It was decided that sequencers had the opportunity of selecting different activities as appropriate. This set of activities is afterwards showed to the student in a menu. The selection of the student is notified to the sequencer, in order to perform any updating procedure. Therefore, the third function of the interface (process) was split in two: process and update.

Process This function is executed by the Administrative Unit every time the user finished an activity. Through the execution of this method, the Administrative Unit passes to the sequencer all the information it has collected from the activity (from the HTTP request, see above). The sequencer, in return, is expected to return *a set* of activities that are suitable for the student, given the performance on the last activity and the past history, if relevant. This set of activities is to be presented to the student by the Administrative Unit in a menu, so one of them can be selected.

Update This function is executed after the user has selected one of the activities in the menu. The Administrative Unit communicates this decision to the sequencer which in turn performs any update necessary and returns the URL of the resource that corresponds to the selected activity. It must be noted that the tutoring process cannot continue until the sequencer has performed the corresponding update, as the Secretary Unit does not know the URL that corresponds to each node.

If there is only one activity as the result of *process*, the Administrative Unit can execute *update* immediately, without showing any menu or waiting for the user.

SIT version 4

SIT version 4 introduced decoupled activities, complex activities in which resources and processing are not performed by SIT itself, but by an external application. This extension to the SIT model allowed the platform to interact with other systems. A protocol, based on the work by Guzmán and Conejo [71], was defined for information interchange with those applications. This protocol, already implemented in systems like SIETTE [33], TAPLI [117] or MEDEA [159], is based on the exchange on a URL and combined easily with the SIT philosophy. The exchanged URL contains all data needed by the external application as well as a return address. When the external application has finished its tasks, it sends the user back to that address.

In conclusion, the *update* function had to be slightly modified to address this new concept. In SIT version 4, method *update* returns two pieces of data: a destination URL and a boolean

value that expresses whether the corresponding activity is “normal” or decoupled. This has been implemented to be backwards compatible with SIT version 3 sequencers.

6.3.7. Swarm Paths Information module (SIT version 3)

A module was added to SIT version 3 that applied some lessons learned from the swarm intelligence applications presented on Chapter 5 and tried to overcome some of their limitations. Due to the special characteristics of SIT version 2, in which the tutor followed a deterministic paradigm, it would have been impossible to implement it for SIT version 2.

The Swarm Paths Information module takes information about the success of the different learning paths followed by the students. It shows this information back to the students, helping them in their learning process. In Figure 6.1 it is represented by the letters SPI. It is covered in detail in Section 9.3.

6.4. Communication protocol

The communication protocol of SIT is now described. The protocol of SIT version 3 is presented as it is the most general case. Notes indicate differences between the general protocol and the protocol of version 2 of SIT when appropriate. Once the general protocol has been depicted, two special cases are explained: the protocol using the Swarm Paths Information module (see Chapter 9) and the protocol in the case of decoupled activities.

The SIT protocol starts with the user requesting an activity to perform (e.g. an exercise). This can occur after completing the previous activity, when logging into the system after a logout or when logging in for the first time. The protocol is repeated after each activity performed by the student until the student finishes a learning module or logs out.

General protocol

1. The user requests the next activity (Figure 6.4).
2. The Administrative Unit sends data collected from last activity (if any) to the sequencer.
3. The sequencer processes the data (if any). Taking into account all data collected from the activity, as well as any other relevant information from the inner student’s model, the sequencer decides which activities are suitable for the user. The resulting set of activities is sent to the Administrative Unit. Note: In SIT version 2, the sequencer had only to give the URL corresponding to one activity, and that resource was sent to the student (appropriately wrapped); next four steps were skipped.
4. The Administrative Unit creates a web page with a menu containing all possible activities and presents it to the user.
5. The user selects one activity.

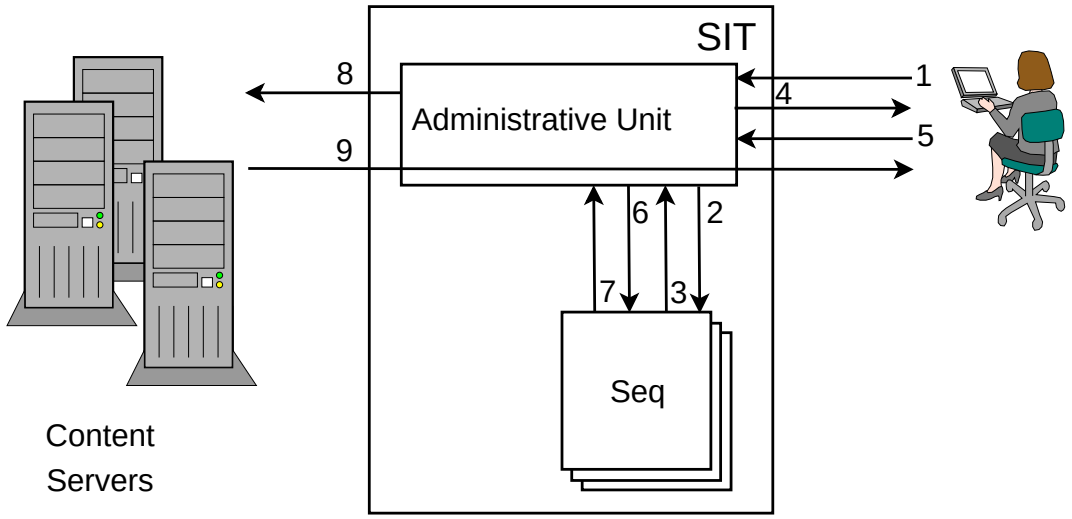


Figure 6.4: General communication protocol of SIT version 3

6. The selection is sent to the sequencer to perform any update that is needed.
7. The sequencer sends to the Administrative Unit the URL of the corresponding resource to be sent to the student.
8. The Administrative Unit opens a connection to the corresponding content server and retrieves the resource.
9. The resource is properly wrapped (see Section 6.3.3) and sent to the student.

Protocol using the Swarm Paths Information module

1. The user requests the next activity (Figure 6.5).
2. The Administrative Unit collects the data (if any) from last activity. If there is any *success information* in the return data, this information is stored in the Swarm Paths Information module; if not, nothing is done. All data collected is sent to the sequencer.
3. The sequencer processes the data (if any). Taking into account data collected and any other relevant information about the student's model, the sequencer decides which activities are suitable for him/her. The resulting set of activities is sent to the Administrative Unit.
4. The Administrative Unit withdraws from the SPI module the success information relating to those activities.
5. The Administrative Unit creates a web page with a menu, so the user can select which activity he/she prefers to perform next.
6. The user selects one activity.
7. The user's selection is sent to the sequencer to perform any update that is needed.

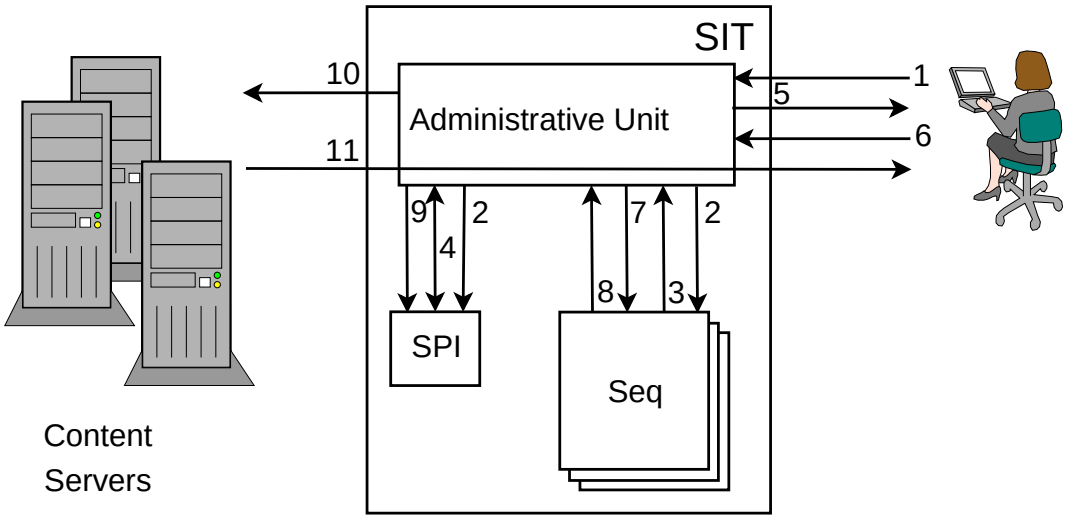


Figure 6.5: SIT version 3 protocol using SPI module

8. When the update is complete, it returns the URL of the corresponding resource to be sent to the student.

9. The path travelled by the student is updated on the Swarm Paths Information module, whether some *success information* was collected from last activity or not.

10. The Administrative Unit opens a connection to the corresponding content server and retrieves the resource.

11. The resource is properly wrapped (see Section 6.3.3) and sent to the student.

Protocol for decoupled activities (SIT version 4)

1. The user requests the next activity (Figure 6.6).

2. The Administrative Unit sends data collected from last activity (if any) to the sequencer.

3. The sequencer processes the data (if any). Taking into account all data collected and the learner's model it decides which activities are suitable. The resulting set of activities is sent to the Administrative Unit.

4. The Administrative Unit creates a web page with a menu, so the user can select which activity he/she prefers to perform next.

5. The user selects one decoupled activity.

6. The selection is sent to the sequencer to perform any necessary update.

7. The sequencer sends to the Administrative Unit the URL of next activity. It also sends a special message saying that the activity is decoupled, i.e. not managed by SIT.

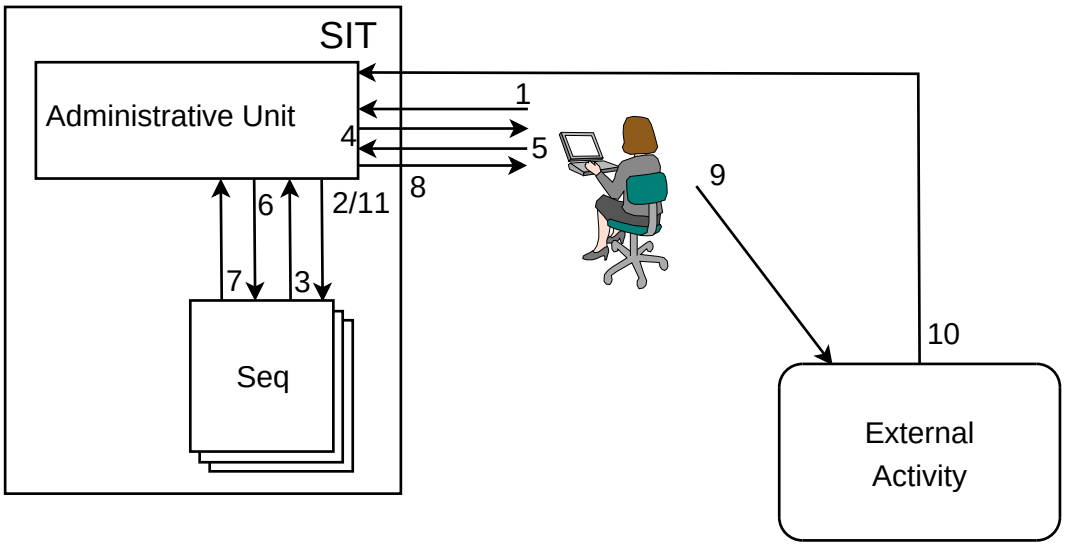


Figure 6.6: SIT version 4 protocol with decoupled activities

8. The Administrative Unit creates a page for explaining the user that an activity has to be performed in another platform. This pages explains that there can be changes in the user interface, etc, to prevent cognitive dissipation. The page contains a link to the external application; in the URL, a *return URL* (i.e. a link to SIT) is embedded.

9. The user follows the link and performs the corresponding activity.

10. When the activity is finished, it sends the user to the return link. This link contains any information that may be relevant (e.g. time needed to finish the activity, marks, etc) embedded in it. The Administrative Unit collects all those data.

11. The data is sent to the sequencer, and the cycle continues normally.

The Swarm Paths Information module can be used for learning modules that contain decoupled activities as well, but this has not been included for the sake of clarity.

6.5. Conclusions: modular and oriented to sequencing adaptation

SIT is a platform for the development of intelligent tutoring systems with a focus on the adaptation of the sequencing of learning activities. It alleviates the ITS designer from several tasks that are time-consuming but are not involved in the process of instruction. It has a modular architecture that makes it scalable, facilitates the addition of new features and makes the process of comparing different sequencing strategies as simple as using several sequencers on a combined experiment. The sequencers only have to comply with a very simple interface.

SIT has no resources of its own, and neither have ITS built with it. It is oriented to the reuse of educational resources that are available on the web. This encourages reuse of learning content.

It can be seen how content is interchanged and reused in blogs and other similar applications on a daily basis. SIT stimulates similar behaviours by identifying resources with URLs. Once the ITS designer has knowledge of some educational resource on the web —either made by herself or not— it can be referenced by its URL and SIT is responsible of retrieving it, showing it to the student and capturing the results of the interaction between them. The ITS designer only has to provide the URLs and to design the sequencing strategy.

The SIT architecture emphasizes an important separation between learning content and its sequencing. Content can be provided by different experts than those who design the order in which it has to be presented to the students. The creation of content and the definition of its sequencing are two conceptually different processes —yet very related— and this separation allows different learning providers to specialize on different tasks, becoming more efficient.

7 Sequencing Graphs

> This was inspired by the structure of the Oparian mafia: it was composed of isolated cells, integrated in the net exclusively through their bosses, who formed cells of second and third level.
>
> – Kiril Yeskov, “The Last Ring”

El sistema para tutoría inteligente descrito en el capítulo 6 hace hincapié en adaptar la secuencia de unidades de aprendizaje al estudiante. Los ITS desarrollados con esta plataforma deben basarse en alguna herramienta que produzca una secuencia adaptada de elementos. Esta herramienta debe implementar las partes principales de un sistema de tutoría como el modelo del dominio o de la instrucción. Este capítulo presenta una especialización de los autómatas finitos que se ha desarrollado para definir secuenciamientos adaptativos de actividades según estos requisitos. Un autómata finito produce secuencias de estados según ciertas reglas; puesto que cada estado se puede asociar a un recurso educativo, un autómata finito puede usarse para secuenciar material. Esta opción supone una curva de aprendizaje menor que otras opciones revisadas en el capítulo 3, y por eso fue escogida para realizar dos tutores inteligentes. Estos tutores fueron probados experimentalmente y los resultados se presentan aquí.

The System for Intelligent Tutoring described in Chapter 6 puts the focus on adapting the sequence of learning units to the student. Therefore, ITS built with it had to rely on some tool that produces an adapted sequence of items. This tool must able to implement the logical parts of a tutoring system like the domain model and the instructional design. This chapter presents an specialization of the finite automata metaphor that has been developed to define adaptive activity sequencings, complying with these requisites. A finite automata produces sequences of states following some rules. As every state can be linked to a learning resource, finite automata can be used to sequence material. This option presents a lower learning curve than other options that were reviewed in Chapter 3, and was thus selected to create two tutoring systems. These ITS were experimentally tested; results are presented herein.

7.1. Objectives

Four goals were pursued when designing a solution for sequencing learning content adaptively. The intention was not to create yet another sequencing approach or yet another graphical metaphor. None of the available options performed specially well in these four objectives: simplicity, scalability, reusability and expresiveness regarding cycles.

First, the solution had to be as simple as possible from the point of view of potential future users. Simple solutions tend to work better than complex ones for a number of situations, and only simple tools that are easy to use do not disappear over time. That is why the finite automata paradigm was selected: it is a well know metaphor for users with a fair technical background, while it is not difficult to understand for other users.

Then, the solution proposed should be scalable. This means that it should be able to handle a large number of activities to be sequenced. The issue is not on technical scalability or technical performance, although that is also important. The point here is that instructional designers must be able to define adaptive sequencings of learning content without their work becoming unmanageable.

The third goal is reusability. With a similar spirit to the original concept of reusable learning object [177] or the IMS Simple Sequencing specification [84], it was decided to promote reusability of sequencing. This has two dimensions. First, the change of an activity for another one that is equivalent should not produce a change in the overall sequence. For example, if I used books History 1, 2 and 3 (in that order) to learn about the History of Spain, then the new edition of History 2 —that covers the same concepts and aims at the same learning goals— should not alter the sequence of books that my little brother will follow when he attends school. Second, the effort on reusability has a second lecture of *integration* and *combination*. If a sequencing has been already defined for a set of activities, it should be easy to get everything (activities plus sequencing) as a black box and integrate it as part of a broader picture, with little or no effort. This is related to the former point about scalability.

Last, but not least, it had to be easy to define reflexive cycles. This is extremely important issue in learning, and the failure to address this fact is a big lack on the part of many elearning systems. In particular, as it will be showed in Section 8.1, this is something that is not well addressed in the IMS specifications related to the sequencing of learning material.

7.2. Sequencing Graphs

One problem that appears when designing a tutor is how to handle a large number of learning resources. A course includes a significant number of resources such as exercises, documents, manuals, etc. In the goal of designing a practical tutor, one of the issues that needed to be addressed is how to allow the learning content author to organize and manipulate a significantly large number of possible resources. The proposed strategy tackles this problem and at the same time aims at facilitating the definition of sequences of learning material by mimicking its inherently hierarchical structure.

7.2.1. Description

A sequencing graph can be briefly described as an oriented multigraph, without isolated nodes. Some of these nodes are bound to actual basic learning resources (exercises, documents, videos, audio, etc.) and some are logical containers that hierarchically contain another sequencing subgraph —this means *hierarchy* and not *clustering*—. In order to enable sequences that inter-operate on different levels of the hierarchy, each graph has one or more of its nodes defined as *input nodes*. Input nodes are the entry points from a higher level of hierarchy. Exit points are in the form of directed edges that connect nodes to their parent nodes. A node can be an input node and have an exit point at the same time, if that fits the pedagogical objectives of the sequences defined by a graph. Sequencing graphs do have a simple hierarchy: the parent node of each node is unique. This type of hierarchical transition structures are not a new concept since they have been already used in a number of areas such as system design [72, 64] or video compression [3].

Each edge in the hierarchical graph may contain a *condition* and a set of *actions*. The condition is an arbitrary boolean expression over a previously defined set of variables. The actions represent a set of operations to be performed over the values of these variables. The semantics of this pair of elements is that the transition represented by the edge is suitable to be performed if the condition is true, and if finally executed, the actions are reflected in the set of variables.

The idea behind this graph is to provide a sequential structure that captures a large set of possible node sequences over a large number of learning objects. These sequences are decided based on the answers obtained from the students in previous nodes and their current position in the graph. This information is stored in the "context".

Formal definition

A Sequencing Graph SG is recursively defined as follows:

A *sequencing graph* SG = (N, E, Φ, Π^*, I, V, η, Υ) is an eight tuple where elements of the set N are *nodes*, elements of the set E are *edges*, $\Phi \subset$ N is the set of *entry nodes* of SG, Π^* is the set of *environments* in SG (one for each user), I is the set of *initialization actions*, V is the set of variables, $\eta \subset$ V is the *upwards interface*, and elements of the set Υ are *downwards interfaces*. SG is called the *parent node* of each n in N, while they are called its *children nodes*.

A *node* n $\in$ N is either a state associated to a learning activity or a sequencing graph. In the former case, it is called an *activity node*; in the latter, it is called a *container node*. At the parent node, there must be one and only one *downwards interface* for each container node.

A *variable* v $\in$ V is a name that identifies a value in the *environment* of the student.

An *environment* $\pi \in \Pi^*$ is a set of pairs variable-value, where information about the student and its relation to the graph can be stored. For the sake of simplicity, attribute values are of two types: strings and integers. Changes in the environment are performed by the learning activities and by the *actions* at the edges. Each student has its own environment. Actions fired by a student when following an edge change only the environment of that student.

An *action* a determines a change in the environment. It assigns a value to a variable, that can be a constant or the result of a complex operation regarding others variables and constant data. Operations include addition, substraction, multiplication, division, max and min.

A *condition* c specifies a boolean expression. Operators allowed for integer comparison are $=$, $<$, $\leq$, $>$, $\geq$. Strings (including booleans) can only be checked for equality. The allowed boolean connectives are !, &, | for negation, conjunction and disjunction respectively.

An *edge* $e \in$ E is a tuple (n_s, n_d, c, A), where n_s is the *source node* and n_d is the *destination node*, c is the *condition* and A is the set of *actions*. An edge defines a possible transition from its source to its destination when its condition evaluates to "true". If the edge is followed, the corresponding set of actions A is executed, modifying the environment. For an edge in a sequencing graph, the parent node of its source must be its destination or the parent of its destination; in other words, there are not any edges connecting nodes in different SG or different levels of the hierarchy: all transitions are defined between siblings or from the child to the parent[1]. An edge whose destination is the parent node of its source is called an *exit edge*.

An *initialization action* $i \in$ I is an action that is executed when the user enters a container node. It adds a new pair to the environment. There must be an initialization action i for each variable $v \in$ V.

An *entry node* $\phi \in \Phi$ is a possible node in which to start the sequencing when going down in the hierarchy of nodes. See Traversal Algorithm bellow.

The *upwards interface* η is the set of variables that are exported to the upper level: when the student follows an exit edge, this variables are copied with their values into the environment of the user at the parent node (with maybe a change of name, see *downwards interface*).

The *downwards interface* $\upsilon \in \Upsilon$ defines how the variables exported from a child node are to be copied into the user's environment π. It may leave the variables' names unaltered or state a change on the name in order to avoid collisions.

In a SG, transitions are defined as a function of the current node and the conditions on the edges whose source node is the current node, evaluated against the environment of each student (see Traversal Algorithm).

$$\delta : N \times C \times \pi \rightarrow N$$

The output of this function is a sequence of learning activities adapted to each student.

7.2.2. Traversal algorithm

An example of a simple sequencing graph is given in Figure 7.1. The graph has two hierarchical levels (the root node is omitted for the sake of clarity). At the top level, node $B3$ contains the subgraph with nodes labeled $A1, \ldots A4$. With the exception of $B3$, all nodes are attached to a learning

[1] This restriction eases reusability.

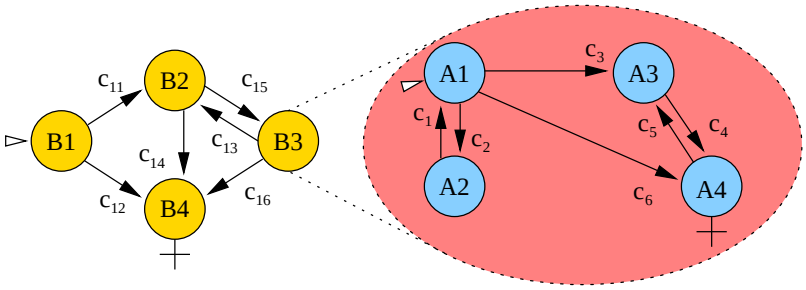


Figure 7.1: Sequencing transition graph example

activity. Input nodes at each level are marked with a white incoming arrow ($B1$ and $A1$), and exit edges are marked with a cross at the bottom of the node ($B4$, $A3$). If the current state is $B2$ and only condition c_{15} is true, then the transition function δ returns $A1$ as the following activity to be visited.

The algorithm to traverse a sequencing graph is showed in Algorithms 1 and 2, and is executed every time the student finishes with an activity. All outgoing edges' conditions are evaluated against the student's environment; those transitions where the conditions evaluates to true are enabled[2]. If any of the enabled edges is an exit edge, this process must be repeated on the parent node. It must be noted that the evaluations at the parent node have to be performed against the result of modifying the environment π of the student with the set of actions on the corresponding exit edge.

If the transition leads to a container node, a new environment π' is created, its set of initialization actions I is executed modifying π', and the next activity to be delivered is one of its entry nodes. If the node selected is itself an entry node, this entry node selection procedure is repeated. It must be noted that, although several changes of hierarchy level may be involved in a transition, only one horizontal edge (not exit) is followed in any case.

It can be the case that no transition is available. That means that the graph is not correctly designed. The algorithm is not specified for such a case: the behaviour of the system is left to the application.

This algorithm provides a powerful and intuitive framework to organize a large number of learning units, and at the same time, characterize multiple sequences of these units. This power is obtained because the sequencing decisions are hierarchically specified. When designing a sequencing graph, an author may focus on the low level decisions on how to sequence a set of learning units depending on the results obtained so far by a student. At the same time, once these sequences have been specified for several topics, the sequencing of such topics can be done by considering them as atomic units.

[2]Conditions can be viewed as prerequisites.

Inputs: current node n, environment π
Result: an activity node
initialize "list of possible transitions" Λ /* empty */
initialize "position" ϖ = n
Λ = find_transitions(n, π)
select one "transition" (sequence of edges) σ in Λ
forall *edges e in* σ **do**
  execute actions A in e, modifying π
  if *destination of e is the parent of* ϖ **then**
    forall *variables v in* η *(and in* π*)* **do**
      copy υ to parent's environment π_p (possibly changing the name according to Υ)
    end
    delete π
  end
  change ϖ to e's destination /* go to parent (0+ times) or sibling (once) */
end
while ϖ *is a container node* **do**
  create a new environment for the lower level graph π_l
  execute initialization actions of ϖ on π_l
  select one entry node ϕ of ϖ
  change ϖ to ϕ /* go down from ϖ to ϕ */
end
return ϖ

Algorithm 1: Traversal algorithm in SG

Inputs: current node n, environment π
Result: a set of transitions (i.e. sequence of edges), i.e. a list of lists of edges
initialize "list of possible transitions" Λ /* empty */
forall *edges e whose source node is n* **do**
  if *condition c in e evaluates to "true" (against* π*)* **then**
    initialize transition λ /* empty */
    if *e is exit edge* **then**
      initialize list of possible transitions from parent Λ_P /* empty */
      duplicate π in $\pi_{\mathbf{V}}$
      execute actions A in e modifying $\pi_{\mathbf{V}}$
      Λ_P = find_transitions(parent(n), $\pi_{\mathbf{V}}$)
      if Λ_P *is not empty* **then**
        forall *transition* λ_P *in* Λ_P **do**
          add e to λ_P, then add λ_P to Λ
        end
      end
    else
      add e to λ, then add λ to Λ
    end
  end
end
return Λ

Algorithm 2: Function find_transitions

7.2.3. Entry nodes, exit nodes and the inter-level interface

Although exit edges were not present in early versions of the definition of sequencing graphs, further research and development of the tutoring system showed that there was a need for them. Actually, the existence of output edges in necessary because it allows data from lower level graphs to be exported to higher levels. That is the way in which the inter-level interface is designed.

On the contrary he existence of *entry* edges was not necessary. First, allowing edges coming "from above" is negative to the philosophy of having sequencing graphs that are autonomous; the presence of entry edges would mean a higher degree of coupling between levels and would make reusability more difficult. Second, it can be seen that a graph should be initialized in the same way, not depending on how or where the student enters in it. That is why there are not entry edges, but just entry nodes and initialization phases, explained bellow.

7.2.4. Initialization phase and the intra-level interface

An initialization section for each graph is necessary. It defines the intra-level interface, that is, the variables that can be used in this level of hierarchy. Earlier versions of the sequencing graphs did not enforce explicit declaration and initialization of variables with a compulsory phase, but further tests and development of ITS showed its necessity. This, like the declaration of variables in compiled languages, allowed for automatic error detection at design time.

Besides, the initialization phase provides a coherent initial state for each graph. This does not only relevant from the instructional point of view, but it has opened the door for automatic testing of sequencing graphs [108].

7.2.5. The issue of reusability

Reusability is encouraged in SG in a double sense: from the point of view of the activities and from the point of view of the graphs themselves. From the point of view of the activities, an SG does not change if one of the activities in the nodes does, as long as the interface of the activity (the set of variables that it introduces in the environment) is the same. This implements a logical separation between learning content and its sequencing, allowing for different experts to specialize in the two aspects of tutoring. For example, activities could be modified to have a more interesting appearance or more complex interaction with the user. As long as the variables exported (e.g. time of interaction) are the same, the graph remains unchanged, as well as the instructional design that it defines.

From the point of view of the graphs, the clear separation between levels of hierarchy permits to reuse a whole sequencing graph as it was an atomic activity. Different SG can be combined with a graph of a higher level of hierarchy that establishes the sequencing of them. An example is showed in Section 7.4.2: several graphs, each of them implementing an instructional strategy for a domain, are combined together with a SG to create a tutoring system of a domain that includes them all. This is performed at a very low cost: only the variable exported (the upwards interface) needs to the known on the higher level. The rest of the information is hidden.

7.3. The SG Sequencer

SG define a set of possible sequencings of learning material, and that is the task that a Sequencer must perform in SIT. Thus, a sequencer has been implemented for SIT based on SG. It has been used for the creation of two ITS for the domains of Computer Architecture and Operating Systems. The resources used for these two ITS were web pages and parametric exercises (see appendix D). This section describes the implementation of the sequencer focusing on two issues: the implementation of the Sequencer interface and the implementation of x-nodes to encourage reuse of SG.

7.3.1. The Sequencer interface

The Sequencer interface of SIT (Section 6.3.6) describes the communication protocol between the platform and the sequencers. This section describes the behaviour of the SG Sequencer, analysing the implementation of the interface functions.

Init

Nothing is done when this function is executed. Graphs are not built until needed, as well as the environment for every student. As all data about a student are stored in his/her environment and position, no further initialization need to be done.

Close

When this method is called, sessions are closed for current users and their data is stored. Allocated resources (like memory occupied by currently used sequencing graphs) are freed. As the current implementation is done with Java, this is not accomplished directly, but the garbage collector takes care of it.

Process

This function includes the biggest part of the activity that takes place in the sequencer. It includes some initialization procedures and performs most procedures involved in the sequencing.

First, when a student demands an activity, and the sequencing graph for the corresponding learning module has not been built, it is instantiated in memory from the XML source description. Once built, the SG remains in memory and is shared by all the students that need it.

If the student is starting with the learning module, an initial environment is created for him/her. The environment stores all information that is needed from the student at every moment, apart of the current position in the graph. As the student moves between levels, more environments are created when necessary.

When the corresponding sequencing graph has been built and the student has an environment in the system, its position on the graph is restored. If any data has been retrieved from last activity, all is stored into the environment of the student. Then, for every edge going out of this (non-hierarchical) activity node, the corresponding condition is evaluated against the environment. For those arcs whose condition is true, the node at the other end is retrieved and added to the list of possible next activities. If that node is not an activity node but a container node, its children entry nodes are retrieved and the process is repeated. The only exception to this comes when the container node was reached "coming from a child" through an exit edge. If that is the case, it does not make sense to go down immediately (this would create a spurious loop). On the contrary, the edges going out of this *father* node are evaluated, and those whose condition evaluates to true are followed. It can be the case that many exit edges are followed in this manner.

If the user is starting this learning module, there is no position to restore: the position is then calculated going down from the root, following a sequence of entry nodes.

Once all the possible next activities have been collected, they are sent to the Administrative Unit[3]. If, because of the design of the graph and the evaluation of the conditions, the system returns to the root of the graph, a special message[4] is sent to the Administrative Unit so it can show an adequate screen to the student.

Update

When user selects an activity, this choice is informed to the sequencer. The position of the student in the graph and the environment (i.e. values of variables) is updated. It is important to note that, although the environment cannot be actually updated until the user has selected an option, all possible variations of the environment have been calculated in the former step.

Every edge is suitable of having several actions that modify the environment. At a high level this can mean leaving a mark that the student masters some learning objectives; at a low level this can mean adding information about how much time was needed for solving an exercise correctly. When an edge is "followed" during the *process* phase, all this actions have to be taken into account, as they influence the sequencing from that point on. Thus, as well as storing a set of possible next activities, the sequencer has to store a set of possible resulting environments.

Once the position and environment of the student have been updated, the URL corresponding to the next activity is sent to the Administrative Unit, that can forward the resource to the student. Since SIT version 4, an additional result is also returned, indicating if the activity is a resource-based one or it is a decoupled activity that requires complex interactions with an external application (e.g. SIETTE [33]).

7.3.2. The x-nodes and the inter-level interface

How can a SG be included in another SG at run time? X-nodes are the implementation of the answer for the SG Sequencer. They are act like container nodes that do not contain activity nodes

[3]This behaviour was different in SIT version 2: once a possible activity was found, it was sent directly to the Administrative Unit

[4]In the Java implementation, an exception is used.

but only a pointer to the included SG instead, a downwards interface and a set of edges that relates them to their neighbour nodes. Following the general philosophy of SIT, the SG description is determined by a URL and can be placed anywhere in the Internet.

This is why the inter-level interface is important, as it specifies which variables are exported by the *included* SG to the *including* one. It is assumed that the SG designer that includes another SG knows the interface of the included graph and initializes the corresponding variables at the initialization phase; otherwise, an error at design time is produced. The designer must also define the new names for the imported variables in the downwards interface in order to avoid naming conflicts.

7.4. Experimental results

In order to accurately assess the effectiveness of the Sequencing Graphs, two experiments were designed. For each of them, an Intelligent Tutoring System based on SG was designed. Parametric Exercises (see appendix D) were used as the basic learning content units used by the ITS.

Two different scenarios were envisioned for the tutor. In the first one, the objective was to assess the effectiveness of the approach regardless of the interest of the student on the topic. This leads to an experiment embedded in a short session in which a set of students have to use it. The data for the experiment were obtained explicitly through a pre-test and a post-test.

The second scenario was conceived to test the effectiveness of a large set of activities hierarchically organized. A higher amount of documents were produced [108]. The interaction with the tutor was not reduced to a single session but allowed during one entire semester. Results were obtained from the course final exam.

In both scenarios, there existed a set of students that did not take full advantage of all the features offered by the tutor. This *control group* was an objective reference point to compare the results obtained for the rest of students.

7.4.1. Experiment 1: The effect of the tutor on learning

The tutoring module used for this experiment consisted of a total of nine parametric exercises (Figure 7.2) containing each of them a brief explanation of a concept as well as a set of related questions. The experience took place in a lab session within an introductory course on computer architecture. The topics selected for the session were integer encoding, symbolic encoding, base conversions, binary operations, etc.

The designed sequencing graph used all nine parametric exercises each of them instantiated multiple times for a total of 46 nodes (Figure 7.3). At several points in the graph, the same exercise was instantiated with different levels of difficulty. Edges in the graph lead to easier or more difficult exercises depending on the number of correct answers given by the student in previous exercises.

The experiment had to be carefully designed as to capture an accurate measure of the effects of the proposed paradigm. To that end, the student group (a sample of 89 students) was randomly divided into two equally sized subgroups named *test* and *control*. Each group was to be offered a

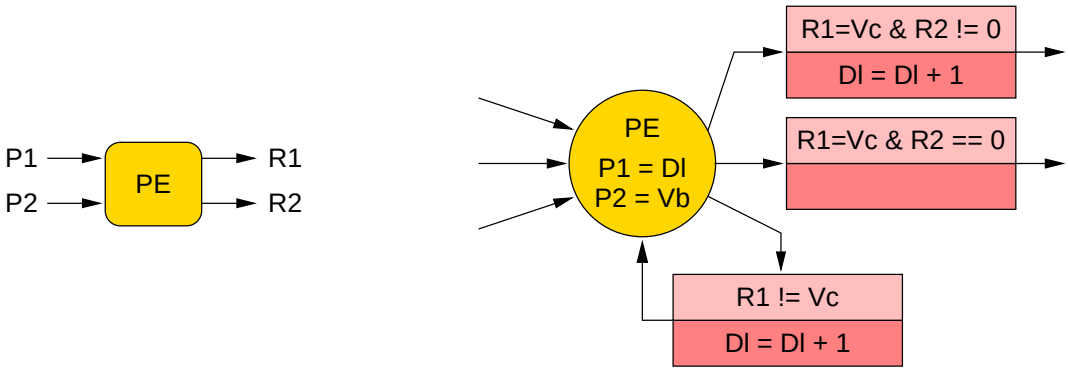


Figure 7.2: Schematic view of a PE

different graph, corresponding to a different instructional strategy: the test group would receive an sequence of exercises adapted to their knowledge and performance, while the control group would receive a linear sequence of exercises with no adaptation.

Offering each group different graphs was a requirement in the experiment, but it did not comply with academic fairness (all students may have access to the same resources in a course). The adopted solution to comply with both requirements was to design a tutor that had two parts, one in which the sequencing was done based on the intelligence captured by a sequencing graph, and another part in which the same exercises where traversed linearly. The test group used a graph in which the intelligent sequenced part was traversed first followed by the linear sequence. The control group, on the other hand, traversed first the linear sequence and then proceeded to repeat the intelligently sequenced exercises. A pre-test was done prior to any interaction with the tutor for all students. A post-test was inserted as an additional exercise in the tutor in the middle of the two parts previously described.

With this structure, measurements are taken before for both groups, whereas the post-test is taken after students in the test group interacted with the tutor and the control group simply solved a linear sequence of exercises with no adaptation. The control group then proceeded to the second part of the tutor in which they had access to the same set of exercises than their peers. The questions

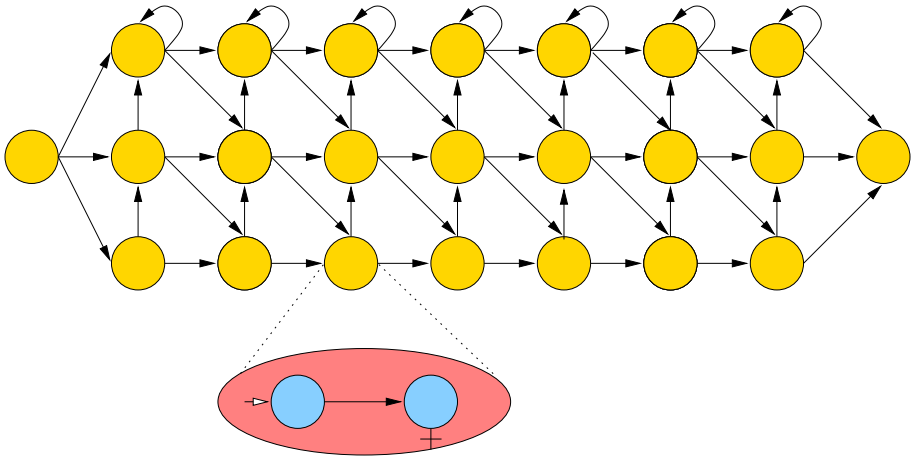


Figure 7.3: Schematic view of the graph used for experiment 1

Learning Goal	Control Group			Test Group			Diff
	Pre-test	Post-test	Improv.	Pre-test	Post-test	Improv.	
1	78 %	68 %	-10 %	76 %	89 %	+16 %	26%
2	56 %	58 %	+2 %	46 %	100%	+54 %	52%
3	43 %	68 %	+25 %	59 %	89 %	+30 %	5%
4	70 %	79 %	+9 %	72 %	82 %	+10 %	1%
5	64 %	49 %	-15 %	54 %	65 %	+11 %	26%
6	60 %	30 %	-30 %	58 %	32 %	-26 %	4%
7	6 %	26 %	+20 %	3 %	54 %	+51 %	31%
8	51 %	52 %	+1 %	63 %	73 %	+10 %	9%
9	23 %	39 %	+13 %	16 %	36 %	+20 %	7%
Average	50.11 %	52.11 %	1.67 %	49.67 %	68.89 %	19.56 %	17.89 %
(w/o #6)	48.76 %	55.63 %	6.87 %	48.63 %	73.50 %	24.87 %	18.00 %

Table 7.1: Pre-test and post-test results for test and control groups

in the tests were two groups of questions relating to nine learning goals [130], one for the pre-test and the other for the post-test, and all the students received the same questions.

Additional measures were taken in order to reduce the level of noise in the measurement. The session was only two hours long, thus reducing the effect of forgetting-remembering cycles. Furthermore, it was presented to the students as a normal lab session, in which they had to perform some “practice exercises” on the topics covered during the lectures. The students taking part in the experiment were first year students that could be distracted by the environment (i.e. lab session with computers), so the aim was to make all the aspects of the experience as “normal” as possible. This was the rationale behind the decision of integrating the pre-test and post-test into the tutor, in order to keep medium changes to a minimum. The obtained results are showed in Table 7.1.

Table rows show the results for each of the nine learning goals in both tests. For each of the two groups, three columns are showed: pre-test results, post-test results and the difference between them. These columns show the percentage of students that correctly solved the corresponding exercises. The rightmost column shows the difference between the improvement detected in both groups.

These results show several interesting aspects about the experiment. Comparing the second and fifth columns (pre-test results) both groups have similar level of understanding of the concepts, an average of 50.11% (control) and 49.67% (test). This suggests that the random selection of students for each group achieved the goal of creating two similar samples. As expected, solving the exercises produced a slight increase in the scores for the control group. Post-test scores averaged 52.11% for an increase of 1.67%. For the test group, on the other hand, after solving the exercises in a sequence decided by the tutor, the improvement in the post-test scores is significant, going from 49.67 % to 68.89% for an increase of 19.56%. There is even an exercise (number 2) that was solved correctly by all the students.

Some special comments have to be done on the sixth row. Learning Goal 6 shows an atypical effect. While students from both groups showed better results in the post-test for all other learning goals, the results for the sixth one are much worse for the two groups. Combining this information with the feedback from the students (as post-experience voluntary forms) and from the assistant

teachers, it was noted that the question related to this learning goal in the post-test amounted to a much more difficult question than intended. This bad result pulls down the overall results for both groups, but does not have an impact on the difference. The results without taking into account the sixth learning objective are presented.

Last column summarizes the difference between the increases in both groups. Scores in the test group improved 17.89% more than in the control group. This evidence allows us to conclude that the proposed framework to adapt content and exercise sequences to the students has a significant positive effect on the learning experience.

Additional data

Apart from the pre- and post-test, the students were given a survey to fill in anonymously immediately after the experiment (i.e. the lab session). Results of the survey show that students found the tutor useful, but the variance in the answers was high. This happened because of the anonymous nature of the survey, that mixed students from both the control and the test group. Although the control group gained some knowledge from the unadapted sequence of exercises, their results were modest compared to the test group.

After the experiment had finished, students were told that they could log into the system any time they wanted, from their homes or using the free hours of the computer labs. This had no effect on their marks, neither was it encouraged in any other way. The objective was to test how interesting was the tutor in the long run or, in other words, if the students had found it so useful that is worth their time. They were allowed to log in until the day of the exam. SIT logs showed that 18% of them continued logging into the tutor after the experiment, most of them from the test group. This is coherent with the results of the experiment and the survey and confirms the positive effect of the tutor on learning, as perceived by the students.

Conclusions of the experiment

The test group that had access to the tutor showed a noticeable improvement with regard to the test group. The tutor was created using nine parametric exercises as learning units and a sequencing graph that adapted the sequence of exercises to the student's performance. Results suggest that sequencing graphs are an effective technique to capture a tutoring strategy that improves learning.

But while the effectiveness of this experiment was showed with only nine exercises, the capability of an author to design and organize a large set of learning objects was not tested. The graph used for this experiment had a single hierarchy level, and it was designed for a two hour session.

7.4.2. Experiment 2: Testing the scalability and long term tutor usage

A second experiment was needed to check that this approach could also be used to sequence a larger number of learning resources over a longer period of time. The goals for the second case were to prove that:

- A large set of learning activities can be easily structured using this approach.

- Different activity sequences, adapted to different types of students, can be captured in that structure.
- The adapted sequences have a long-lasting effect on the students.

The experiment was presented to the students as auxiliary material for a course on Operating Systems [108]. The course objective is to become familiar with the design decisions when building an operating system and the most commonly used techniques. The exercises covered a subset of the course topics a summary of which is showed in Table 7.2.

The material provided partial coverage of the course topics and was complex enough to require multiple hierarchy levels in the tutor graph. A total of 40 parametric exercises and 34 auxiliary documents were produced. The documents included descriptions of the basic concepts covered by the exercises and were used when the obtained results led to think that the student should review them.

Part of the structure showed in Table 7.2 was translated to the sequencing graph. Process management is the basic set of concepts to understand how an operating system offers its functionality to users and applications. Mechanisms such as memory management require a solid understanding of how processes are manipulated. Disk and file management, as secondary storage require concepts of the previous topics. The first sequencing level then is trivial. Topics are covered sequentially as explained in the course lectures. All sequence adaptation is then contained in each of the sub-topics. Figure 7.4 depicts an example of a graph included at one level of the hierarchy.

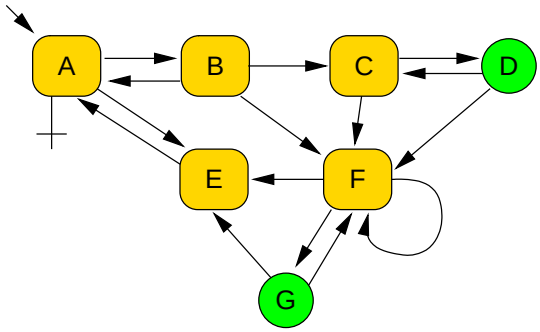


Figure 7.4: Schematic view of one of the hierarchy levels in the SG

In the example, squares represent nodes containing sub-graphs, so only nodes D and G contain parametric exercises. Each edge has its condition and set of actions, not showed for the sake of simplicity. The variables appearing in the condition of these edges store the results obtained in the exercises contained in the sub-graphs allowing a transition from one topic to the next based on these results.

The entire tutor material contained 40 parametric exercises, 34 auxiliary documents, 38 Java classes (for exercise parametrization) and 173 XML files for the graph description which where assembled with the help of the specialized editor described in appendix A.

In this case, students were not separated into two groups, but instead, the availability of the tutor was made public at the beginning of the course. They were informed of the interactive nature of the tool. They were encouraged to keep advancing through the exercises even if they found

them too difficult. Students could interact with the platform until the very same day where the final exam took place.

The exam had a total of 25 true/false questions about all the topics covered in the course. Out of this pool, only the questions related to topics covered by the tutor were taken into account for this study. A total of 115 students took the exam and 78 of them had some interaction with the tutor. Table 7.3 summarizes the obtained results.

The table shows the result for each topic covered in the tutor. Column labeled "Average Number of Events" contains the number of internal tutor events on average registered for each topic. There was no exact correspondence between the number of events and the number of parametric exercises done by the students. Different exercises produced different number of events, however, these events can be considered proportional to the exercises. The interest of this data is because it shows clearly that students interacted mostly with the first topic in the tutor. The further along the tutor, the less students reached that point.

The next column shows the number of questions included in the final exam about that topic. The exam designers had no knowledge about the tutor or its exercises, thus the overlapping between these two contexts was done a the level of "topic" (processes, memory management, etc.)

Fourth and fifth columns contain the average scores (out of 100) obtained for each topic for those students that had any interaction with the tutor and those that had none. In all five topics, scores for the first group were better than the second. The difference is showed in the sixth column. The main increase is for the first topic in which students had a score 11.43 points higher in average. Interestingly enough, this was also where the students spent most of their time interacting with the tutor. The rest of topics have a not so significant impact in average but nonetheless, all scores improved when studied with the support of the tutor.

These positive results could be due to the inherent interest of those students that took the effort of interacting with the tutor. However, the fact that they are coherent with the results of the first case study (in which no choice was given to the students and all of them interacted with the tutor) makes such possibility much less plausible.

Conclusions of the second experiment

Deploying experiments to prove the effectiveness of a tutoring tool like the one described in this document is extremely delicate. Apart from the difficulties inherent in any measurement process, the usual conditions imposed by the courses in which the experiment is embedded add an additional degree of complexity. The spirit followed in the described experiences was to present the tutor as another regular resource to study the course.

From the obtained results it can be concluded that a tutor like the one presented is a simple and effective resource to capture the expertise of the teaching staff and to improve the overall learning process of a regular course in a higher educational institution.

Topic	Learning Units	Exercises	Documents
Process Management	Scheduling Policies	5	6
	Scheduling Policies	5	6
	Scheduling Policy Analysis	2	1
	Performance Analysis	2	1
Cache Memory	Memory Access Time	1	1
	Direct Mapping	5	1
	Associative Mapping	2	1
	Set Associative Mapping	4	1
Memory Management	Contiguous Assignment	2	4
	Paging. Access Time	1	1
	Paging. Address Space	3	1
	Paging. Page Tables	2	2
	Paging. Process Allocation	1	1
	Paging. Page Replacement	2	3
Disk Management	Disk Access Scheduling	2	6
File Systems	Access Permissions	2	1
	Inodes	1	1
	Sector Allocation	3	2
Total		40	34

Table 7.2: Learning Units, Exercises and Documents for each Topic

Topic	Average Number of Events	Quest.	Average Score		Diff.
			With Tutor	Without Tutor	
Processes	276.90	2	58.22	46.79	11.43
Cache	72.97	3	74.81	69.44	5.36
Memory Management	71.54	5	71.54	65.26	6.28
Disk	31.44	1	82.89	74.03	8.87
File Systems	39.83	3	65.35	61.97	3.35
Average	101.52		70.56	63.50	7.06

Table 7.3: Final scores and tutor usage

7.4.3. Patterns

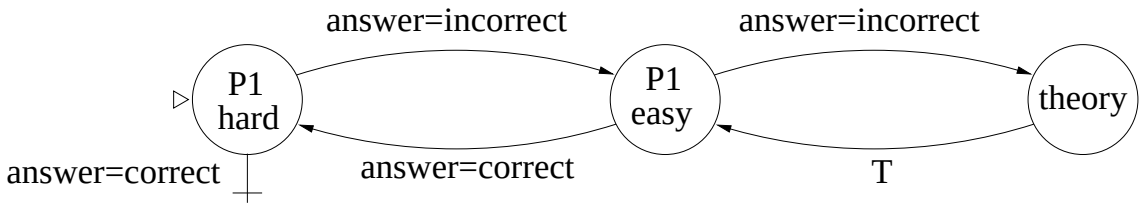
During the design phase of the tutors that were used in these two experiments, many recurring patterns arose. The designers found themselves using the same structures (graphs) over and over again. This situation has occurred in many knowledge fields before, from architecture [4, 5] to software engineering [62], leading to the concept of design patterns. In the words of Christopher Alexander:

> *. . . a pattern describes a problem which occurs very frequently, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice* [4].

Design patters are nowadays an active field of research in adaptive hypermedia [63, 60, 7, 88]. Designing and documenting design patterns for the sequencing of learning activities is a task that needs more experimentation and is beyond the scope of this thesis. However, the most important repeating patterns that appeared during the development of the tutors are presented here as a starting point.

Increasing difficulty

An exercise is presented to the student. If the student fails, the same exercise is presented with a lower level of difficulty. If the student fails again, a theory page is presented to her.



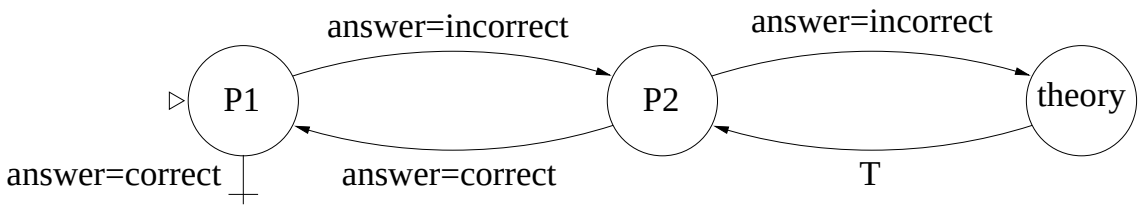
Pattern 7.1: Increasing difficulty

Different exercise

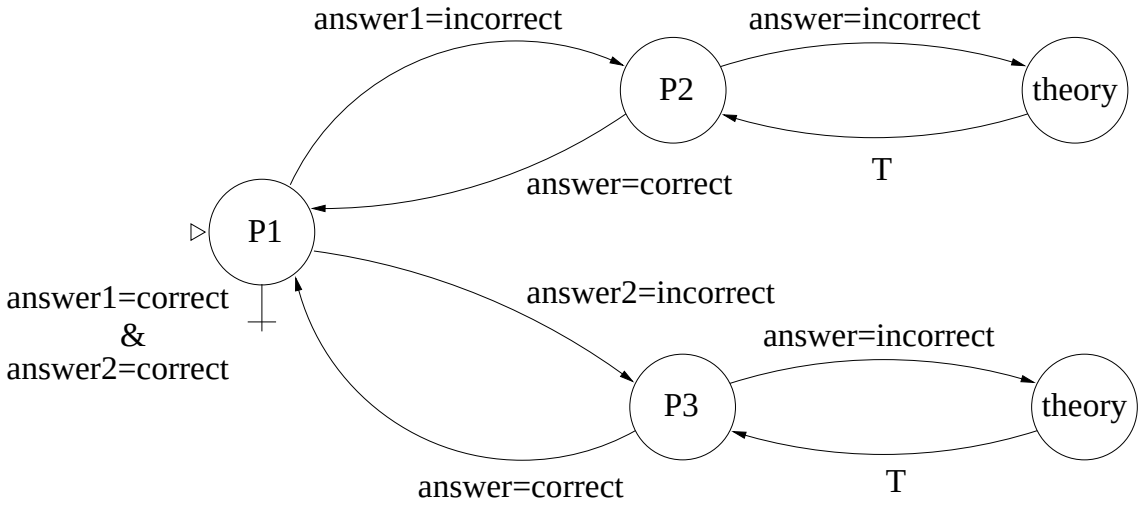
An exercise is presented to the student. If the student fail, a different exercise related to the same topics is presented. Different approaches or points of view can make things clearer to certain students. If the student fails again, a theory page is presented to her.

Different parts

An exercise might be composed of different parts. If the student fails when the exercise is presented to her, an exercise relating to the specific parts that she failed is presented to her. A new failure lead to a specific theory page.



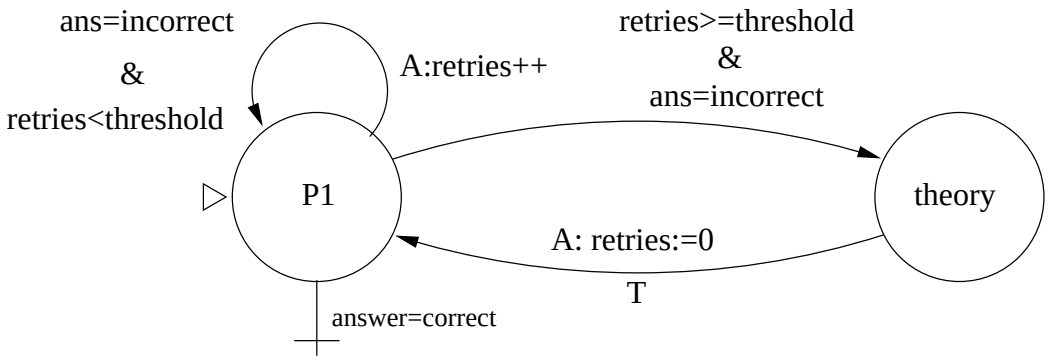
Pattern 7.2: Different exercise



Pattern 7.3: Different parts

Repetitions of one exercise

Sometimes the resources available are scarce. In the case that there is only one exercise to illustrate one concept, it can be presented several times to the student. If the student fails several times in a row (the threshold is customizable), a theory page is presented.

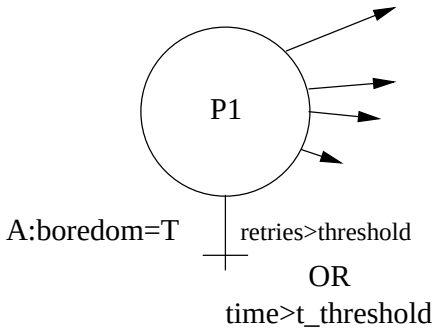


Pattern 7.4: Repetitions of one exercise

The “A:” denotes an action on that edge.

Avoid boredom

It is very important to avoid blocking situations, that may bore the student and have a negative effect on her learning process. If too many repetitions of an exercise are detected, or if the student spends too much time without any advance, the graph is left and a mark is put on the student’s environment. On the upper level, the graph should take care of this special case.



Pattern 7.5: Avoid boredom

The “A:” denotes an action on the exit edge.

7.5. Conclusions

This chapter has presented sequencing graphs as a tool for the definition of sequencings that can be adapted to different students with regard to their capabilities and needs. It has been described as a simple yet powerful extension to the finite automata paradigm that is able to express an instructional design, specifically making it easy to define learning cycles (e.g. reflexive reviews of past material, remembering forgotten concepts, etc). Their inherent hierarchy makes them scalable: sets of learning units can be grouped as a graph of a lower level of hierarchy allowing the designer to concentrate on the big picture. Sequencing Graphs are thus compact and expressive —small graphs can express complex sequencings— as well as compositional and modular. They enable viewing the description at different levels of detail, and make even very large designs manageable and comprehensible. Besides, the hierarchy falicitates reuse of SG, as the separation between levels is clear, avoiding unnecessary coupling with simple interfaces.

Experimental results suggest that useful instructional strategies can be defined with SG, improving learning of students using an ITS based on them. However, there are still some open questions regarding their usefulness that have not been completely clarified.

Next step is studying the relationship between SG and the main standardization initiatives in the field. During the last years, the pervasive presence of the Internet is curtailing the cost for deploying a telematic infrastructure for learning support. This has incremented the creation of commercial products (both proprietary and open source), which in turn has promoted an urge in the community to be able to set some standards to further encourage reuse and competition. In order to be able to have the benefits of SG for the definition of sequencings in major elearning applications, it has been studied if the semantics of SG can be expressed in terms of the specifications of the IMS Global Consortium. This is the topic of Chapter 8.

8 Exporting Sequencing Graphs

Roy: Time to go up a level!
Haley: Time to go down a level!

– Rich Burlew, “The Order of the Stick”

Este capítulo presenta el algoritmo de transformación que permite expresar los grafos de secuenciamiento introducidos en el capítulo 7 en los términos de la especificación IMS Learning Design. La imposibilidad de exportar los SG a IMS-SS se explica después. Ambas especificaciones fueron analizadas en el capítulo 4. El proceso de transformación entre SG y IMS-LD no es trivial, y presenta numerosas dificultades que son resueltas en este capítulo. De ello se extraen algunas conclusiones sobre la validez de la especificación IMS-LD para expresar determinados tipos de estrategias pedagógicas.

This chapter presents the exporting algorithm that exports the sequencing graphs introduced in Chapter 7 using the semantics of the IMS Learning Design specification. Exporting SG to IMS-SS is not possible; the reasons are explained herein. Both specifications were analyzed in Chapter 4. The process of transforming SG to IMS-LD is not trivial. It poses several difficulties that are solved in this chapter. Several conclusions are drawn from the procedure about the appropriateness of IMS-LD to express some pedagogic strategies.

8.1. Introduction

In Chapter 7, sequencing graphs have been presented as a flexible yet powerful technique to describe adaptive sequencings. Positive results from their application to the design and implemen-

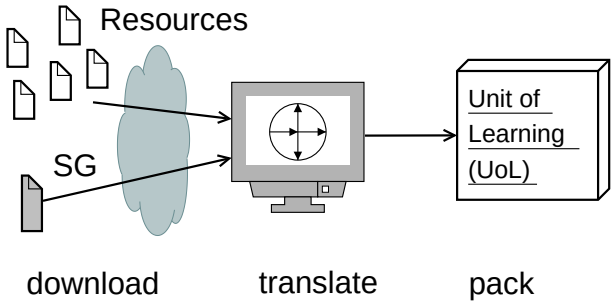


Figure 8.1: SG exporting process to IMS-LD

tation of intelligent tutoring systems have been presented and discussed.

There is a big effort in the elearning community for standardizing elearning processes. In Chapter 4 the main initiatives were presented, focusing on those that are more relevant to the topic of this thesis: IMS Simple Sequencing [84] and IMS Learning Design [81]. Both have the support of many companies and research centers and lots of efforts are concentrating around them, specially the latter.

The next section describes the process that translates from SG semantics into IMS Learning Design semantics. A brief note is made later about the relationship between SG and IMS-SS: early versions of SG could be exported to IMS-SS; unfortunately, this is not possible anymore due to the lack of a versatile user model in IMS-SS.

8.2. IMS-LD

IMS Learning Design is not designed specifically for addressing the sequencing problem. However, it provides some tools to be applied: properties, conditions and actions. These tools make it possible to express the sequencings that can be defined using SG using the IMS-LD elements.

The process described in the following pages has been implemented with a Java application. Implementation details (i.e. class diagrams, source code, etc) are thoroughly explained in [166]. This application is able to take the XML definition of a SG as input and, given some restrictions, produce a perfectly functional Unit of Learning (UoL) that runs on a Learning Design Player. Therefore, it is possible to create a tutoring system using the SG approach and then run the result in some IMS-LD compliant machine; the resources that are placed in one or many content servers are automatically included into the UoL and the graph structure is transformed and included into the *imsmanifest.xml* (Figure 8.1). This output UoL contains all the information from the tutoring system and is used by the IMS-LD players for showing it to the student. As any other IMS Content Package, it is usually a ZIP file.

In a sense, this process can be seen as a change of paradigm. Both the Sequencing Graphs and SIT (Chapter 6) are based on a distributed paradigm, in which resources and the definition of their sequencing are disjoint, resources are disjoint between them, and they can be withdrawn from anywhere on the Internet. On the other hand, IMS Learning Design is more oriented to the

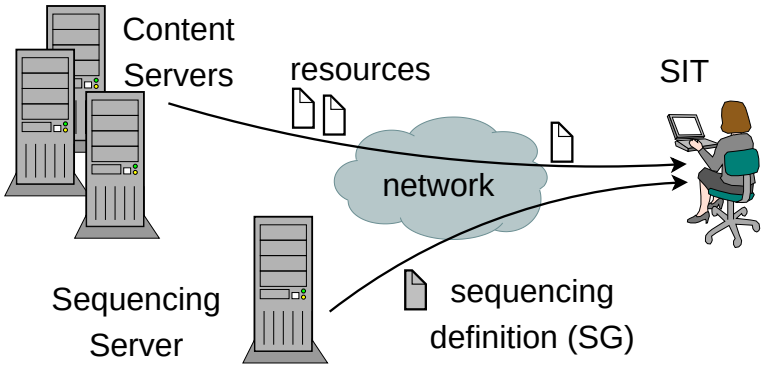


Figure 8.2: SG paradigm

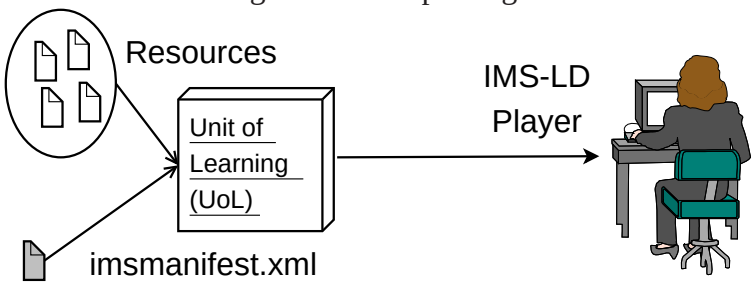


Figure 8.3: IMS-LD paradigm[1]

use of monolithic Units of Learning, in which the resources are included in the same package as the definition of their sequence[1]. This difference is illustrated in Figures 8.2 and 8.3.

8.2.1. The activities conundrum

When the user navigates though a SG, activities are showed to him/her one at a time. They are represented by the <sg:exercise> nodes, that are linked to a resource. This activity sequence does not have to be linear: there can be cycles in which a particular exercise is traversed multiple times, and there can be hops.

The ability of Sequencing Graphs to represent loops is an important difference with respect to IMS-LD, because the latter does not consider that performing the same activity more than once is very important. When a student has performed and completed an activity, it is assumed by the learning design that its goals have been achieved, so there is no point in repeating it. This philosophy is specifically stated in the specification with these words:

> *...a control must be available in the user-interface to set the activity status to 'completed'. A user can do this once (no undo). Once he/she indicated the activity to be*

[1] Strictly speaking, resources can be placed anywhere in the Internet and be referenced by a URL (e.g. in the context of a *iframe*). However, the recommended behaviour is as illustrated in Figure 8.3. In the words of the specification: "In order to avoid confusions it is good practice to include the imsldcontent with property-operations in the unit-of-learning-package" ([83], page 62). Additionally, this has been the observed behaviour of all the UoLs studied during the development of this thesis.

completed, then this activity stays completed in the run ([83], page 28).

This fact prevents the most intuitive way of translating from SG semantics into IMS-LD semantics, that is, creating a set of conditions that are capable of showing to the user the <imsld:learning-activity> that is needed at every moment, each of them associated to its corresponding <sg:exercise>. A direct translation from exercises to activities would not allow cycles because of this "complete once, no undo" characteristic. Without this limitation, translation from SG to IMS-LD would have been performed using the <imsld:on-completion> element of <imsld:learning-activity> to emulate the behaviour defined by the edges.

8.2.2. Solution to the conundrum

The solution to this problem requires a coupling between the *imsmanifest.xml* and the learning resources. In particular, the <imsld:class> property of the <imsld:show> element must be used. These <imsld:class> elements are directly related with the attribute of the same name in a DIV element in a XHTML document. A DIV element does not have any function by itself: it is used as a container for other elements. It is possible to apply effects in jointly to a group of elements contained in a DIV. These effects may vary: CSS styles [167, 168], alignment, etc. In a DIV, the goal of the *class* attribute is to associate the content elements with a style type defined in the CSS page.

IMS-LD uses the *class* attributes at the DIV elements for something different: using show and hide actions, DIVs can be used to reveal or conceal different parts of a resource. This is illustrated in XML excerpts 8.1 and 8.2, the former showing a fragment of the structure of a sample resource document and the latter depicting the corresponding condition. If the condition is true, the IMS-LD player hides all those elements inside the DIV with a *class* value of 'class2', and show all the elements inside the DIV with a *class* value of 'class1'. Thus, the user sees on his/her screen only the message: "Show this paragraph (class 1)".

XML 8.1 Attribute class in a XHTML document

```
<div class="class1">
  <p>Show this paragraph (class 1)</p>
</div>
<div class="class2">
  <p>Show this paragraph (class 2)</p>
</div>
```

The elements <imsld:show> and <imsld:hide> are used for implementing the behaviour defined by SG in IMS-LD, showing some activities and hiding others depending on some conditions. Resources have to be showed to and hidden from the user, with regard to their relation to an <sg:exercise>. The only option is to generate one single resource master file;the content of all the resource files related to the SG are stored in it. Therefore, following an edge in a SG is translated to one <imsld:show> and N-1 <imsld:hide> actions, where N is the total number of nodes here. Each of these resources is wrapped in a DIV; the value of the *class* attribute of that DIV is equal to the name of the original <sg:exercise>.

XML 8.2 SG condition exported into an IMS-LD condition

```
<imsld:if>
  <imsld:is>
     [Here the desired condition]
  </imsld:is>
</imsld:if>
<imsld:then>
  <imsld:hide>
    <imsld:class class="class2" />
  </imsld:hide>
  <imsld:show>
     <imsld:class class="class1" />
  </imsld:show>
</imsld:then>
```

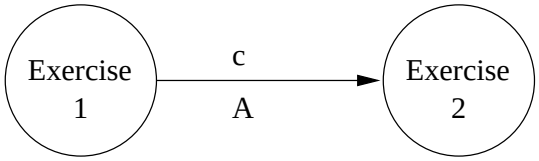


Figure 8.4: Sample graph of only two exercises

The trivial case of a graph composed of only two nodes (two <sg:exercises>) is illustrated in Figure 8.4 and XML excerpt 8.3 for the sake of clarity.

The file presents an invariable part. This is marked in XML excerpt 8.3 using XML comments. These fragments are always present in the *imsmanifest.xml* that is created in the process of exporting a SG. The rest of the file depends on the particular SG, the number of nodes <sg:exercise> contained in it, as well as the content of the XHTML files linked to them.

As a side effect of this process, the hierarchy of the SG disappears. All the nodes are exported to activities on the same level. All edges' conditions are written together at the condition section of the *imsmanifest.xml*. This has several implications: it makes it necessary to implement a mechanism to avoid collisions between properties' names (i.e. variables' names on different levels of hierarchy) and it makes the size of the manifest to increase non-linearly with respect to the number of nodes.

The manifest

The tool that exports a SG into a UoL uses a template for creating the *imsmanifest.xml*. The template contains all the information about the learning design that is invariable with respect to the SG. It is very long and is not included here for the sake of tidiness, but the main concepts are explained. An instance, filled with data from a particular SG, can be observed in appendix B. The invariant parts of the manifest template are:

- There is only one role: Student. SG are designed for personal, unassisted tutoring.
- There is only one <imsld:learning-activity> that is always visible. Its name is "Sequencing

XML 8.3 Sample of master resource file

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This is common -->
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:imsld="http://www.imsglobal.org/xsd/imsld_v1p0">
 <head>
   <title>Activities</title>
 </head>
 <body>

   <!-- This is specific to each SG and each UoL -->
   <div class="Exercise 1">
            [Code XHTML of the resource associated to node "Exercise1"]
   </div>
   <div class="Exercise 2">
            [Code XHTML of the resource associated to node "Exercise2"]
   </div>
   ...
   ...
   <!-- This is common again, and to the end of the file -->
   <div class="classExitDiv">
     <p>You have succesfully completed this module. Congratulations! </p>
   </div>
 </body>
</html>
```

graph". This activity is associated to the file *resourceFile.xml*.

- The method in the learning design is always structured in the same way: there is only one <imsld:play>, and there is only one <imsld:act> in it. In the <imsld:act> there is only one <imsld:role-part>, and it is composed of the "Student" role and the corresponding <imsld:learning-activity>.

Of course, most of the manifest depends on the specific graph. Their size is usually proportional to that of the original SG file. These parts are:

Title of the learning design . This element is generated with the same name as the root of the original SG.

Properties . They are the equivalent in IMS-LD to the variables in SG. The number of properties is proportional to the number of variables in the graph. Although there is no formal relation between the number of variables and the size of the SG description file, the graphs designed have showed a linear dependency.

Conditions . They are the equivalent in IMS-LD to the conditions of SG. There must be at least as many conditions in the manifest as conditions there were in the original graph. The number of conditions is linearly related to the number of properties, but is proportional to the square of the number of nodes. Besides, the length of the set of actions associated to IMS-LD conditions is proportional to the number of nodes. This means that the total number of actions follows the cube of the nodes. This is the total number of nodes in the graph; the

hierarchy does not help because it disappears in the exporting process. Most of the resulting manifest is occupied by conditions and their associated actions.

The common part

The beginning of the *imsmanifest.xml* is invariant. It comprises the XHTML header, including the title. The other invariant part is the footer.

On the footer there is a DIV with a value of *class* of "classExitDiv". This is what is showed to the user if he/she ever "finishes" the graph (i.e. returns to the root).

The SG do not define a behaviour when the user returns to the root of the graph. This aspect is left open to the application. In the case of SIT, a message is showed to the user, but there is a tacit assumption that most sequencing graphs never return to the root, letting the user stay in the upper levels until the need to go back to some part of the graph appears, because something has been forgotten.

When translating the sequences defined by a SG to IMS-LD, the behaviour cannot be left undefined. It has been decided to take the same approach as in SIT and present a message saying that the user has finished with a particular module. This decision has different consequences in the case of IMS-LD, because in many scenarios the user is able to start the same module (graph) in another *run* of the UoL.

In conclusion, in order to avoid unpredictable results when the user interacts with the UoL that is the product of exporting a SG into IMS-LD, a common resource (a DIV in the XHTML master document) is always created with *class* value of "classExitDiv". Should the user return to the root of the graph, the learning design shows this final resource.

The "continue" box

The only mechanism defined in IMS-LD to indicate that interaction with an activity has finished is to complete it. As completion of activities cannot be undone, the influence of this fact on the exporting process is important. The mechanisms explained in the specification do not allow to express that a user has introduced all data required by an exercise, then grade the results, move to another activity and then come back to the same exercise and solve it again. Using SIT as an example, there is no "Continue" button in IMS-LD. A complicated technique has to be used to implement this behaviour.

Inside the resource master file, each of the DIV elements is concatenated with a <imsld:set-property>[2]. This element is invariable and is showed in XML Excerpt 8.4 When the LD player finds this tag in the XHTML resource, it generates a text box in which the user can introduce directly a value for the property called "export_SG2LD-last-unit".

The element <imsld:set-property> is one of the four global elements in IMS-LD. Global elements are the method designed in IMS-LD for allowing users to interact with the learning design

[2]This is not showed in the XML excerpt for the sake of simplicity. It would be written in the section marked as "Code XHTML of the resource".

XML 8.4 Invariant set-property for export_SG2LD-last-unit

```
<imsld:set-property ref="export_SG2LD-last-unit"/>
```

through the activities. In particular, <imsld:set-property> permits to introduce data, e.g. the answers to an exercise. When the learning design is created from a SG, this technique is used for receiving the answers from the exercises as well as any other input data needed from the student.

The interesting point here is that the "continue" box achieves the same behaviour as a "Continue" button. When the user has finished with one activity and wants to advance to the next one he/she must enter the word continue in the text box, and then press "OK". This is showed in Figure 8.5.

When the user presses the "OK" button, the value "continue" is introduced in property *export_SG2LD-last-unit*. This change fires the evaluation of all conditions that are related to it, i.e. conditions related to edges. The process evaluation of those conditions finishes the update of several variables (notably *export_SG2LD-last-unit* and *currentActivity*) and a change on the visible state of the DIVs. This is covered in more detailed later on page 104.

Resource master file

The resource master file is always created with the same name: *resourceFile.xml*. Therefore, UoL's created with this application always have two files: *imsmanifest.xml* and *resourceFile.xml*.

The size of the master file is proportional to the size of the resources used in the graph. This presents problems for the web browser when the number of resources is high and/or they are heavy documents.

8.2.3. Resources

The problem posed by activities in IMS-LD and the solution proposed brings another problem: the specification recommends not to reference resources through a URL, because all resources should be packed into the master file. Thus, first step is downloading all those resources that are needed for the learning activities in the SG.

References to resource files are located into the SG description file. The inner structure of this XML file is showed in appendix B. Once all resource files haven been downloaded, they can be included in the UoL to be used by the learning design.

In other words, resources are located in the SG description file looking for elements of type

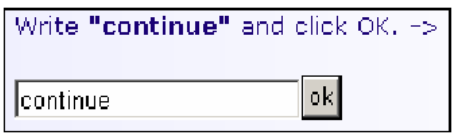


Figure 8.5: "Continue" text box in RELOAD

XML 8.5 Excerpt of a SG description file

```
<exercise name="Ex1"
          url="http://gradient.it.uc3m.es/AO/arch/h1_3.xhtml"
          is-entrance="true" is-exit="false">
  <edge destination="Ex2">
    <condition>
      <simple-condition>
        <parameter>alwaystrue</parameter>
        <operator>eq</operator>
        <condition-value>
          <type>STRING</type>
          <value>true</value>
        </condition-value>
      </simple-condition>
    </condition>
  </edge>
  <!-- More edges here -->
</exercise>
<exercise name="Ex2"
          url="http://gradient.it.uc3m.es/AO/arch/h1_4.xhtml"
          is-entrance="true" is-exit="false">
  <!-- More edges here -->
</exercise>
```

<sg:exercise>. They are downloaded and the corresponding <imsld:resource> entries are created in the <imsld:resources> section of the imsmanifest.xml. Therefore, all resources used by the UoL are declared in *imsmanifest.xml* as it is stated in the IMS Content Packaging specification. An example can be seen in the following code excerpts (XML 8.5 and 8.6): the resources at the machine *gradient.it.uc3m.es* are downloaded and referenced. To avoid naming conflict, they are named using a hash code of their original URL.

The <imsld:resource> tags are nothing more than the declaration of existence of those resources in the UoL. In order to use them, these resources must be associated to <imsld:learning-activity> elements.

XML 8.6 Excerpt of a *imsmanifest.xml*

```
<resources>
  <resource identifier="resource-1" type="imsldcontent"
            href="resource_hashCode1.xhtml">
    <file href="resource_hashCode1.xml"/>
  </resource>
  <resource identifier="resource-2" type="imsldcontent"
            href="resource_hashCode2.xhtml">
    <file href="resource_hashCode2.xml"/>
  </resource>
  <!-- More resources here -->
</resources>
```

8.2.4. Conditions

The most delicate phase of the process is translating all the semantics of a sequencing graph into IMS-LD conditions and actions. The main difficulty comes from the fact that we are changing the paradigm. SG follow an *imperative* paradigm: *if* some condition applies, *then* follow this edge; *after that*, *if* you have followed an exit edge, *then* look for more conditions, etc; *after that*, *if* you arrive to a node, *then* go down one level, etc. On the other hand, the conditions in IMS-LD are more related to a *functional* paradigm [114]. All conditions are evaluated at the same time; if a <imsld:condition> fires some action that produces some change in a <imsld:property> of the system, all conditions are evaluated again. There is no logical or chronological order or precedence between all instances of <imsld:condition>.

Conditions in IMS-LD must capture all the semantics of those in the SG, plus that of the spatial distribution of activities, i.e. current position, existence of edges, etc. To achieve this, some special properties have been defined.

Special properties

These special properties are generated in the process of exporting. Their mission is to control several tasks needed to provide a behaviour equivalent to that of traversing the SG. All these properties can be visualized in XML Excerpt 8.7.They are described bellow:

entering It may have two boolean values: *true* and *false*. Its mission is to indicate if the user is entering or exiting a node. If he/she is entering the node, the *actions* described in the <sg:init> section of the node have to be executed.

currentActivity Its mission is record where is the user, that is, which is his/her current node. This information is evident in SG, but not IMS-LD; thus, a set of conditions and actions has to be deployed to emulate the movement of the student. Its initial value is always the root node of the SG.

export_SG2LD-last-unit Its mission is to record which node the user is going to leave, so that all conditions associated to the <sg:edge>'s that start on that <sg:node> are evaluated (and not others). Additionally, this property takes a role in the process of moving from an activity: it is the property changed when the user presses the OK button in the "continue" box (Figure 8.5); the word "continue" is copied into it. This value change fires a new evaluation of the conditions.

validMove It may have two boolean values: *true* and *false*. A "true" value indicates that the user has correctly expressed his/her intention of leaving the current node. In other words, the user has introduced the word "continue" in the text box showed by the player (see Figure 8.5 on page 102). When this happens, the value of *currentActivity* is copied to *export_SG2LD-last-unit*. This fires the evaluation of those conditions related to the edges that start in the current node and it produces that the next activity is showed. If a value different from "continue" is introduced, nothing happens. Property *validMove* has an initial value of "true", so the first activity can be showed to the user without his/her intervention.

XML 8.7 Special properties and initial state

```
<imsld:locpers-property identifier="entering">
 <imsld:datatype datatype="string" />
 <imsld:initial-value>true</imsld:initial-value>
</imsld:locpers-property>

<imsld:locpers-property identifier="currentActivity">
 <imsld:datatype datatype="string" />
 <imsld:initial-value>
   &amp;Root Node Name&amp;
 </imsld:initial-value>
</imsld:locpers-property>

<imsld:locpers-property identifier="export_SG2LD-last-unit">
 <imsld:datatype datatype="string" />
 <imsld:initial-value/>
</imsld:locpers-property>

<imsld:locpers-property identifier="validMove">
 <imsld:datatype datatype="string" />
 <imsld:initial-value>true</imsld:initial-value>
</imsld:locpers-property>

<imsld:locpers-property identifier="variableZERO">
 <imsld:datatype datatype="integer" />
 <imsld:initial-value>0</imsld:initial-value>
</imsld:locpers-property>
```

variableZERO This special property is not related directly to traversal of the graph, but is explained here for the sake of completion. Its value is always zero (0). It is needed for some kind of mathematical operations because IMS-LD —unlike SG— does not allow to perform operations in which both operands are constants. Thus, to sum up two constant values it is necessary in IMS-LD to sum both with the variable *variableZERO* and then sum both results up.

In conclusion, edges, conditions and variables of SG are converted into IMS-LD conditions and properties. These combine: the SG conditions, topological information about edges (*currentActivity, export_SG2LD-last-unit*), hierarchical information (*entering*) and information regarding the navigation decisions of the user (*validMove* and the “Continue” box).

8.2.5. Other elements

Apart from nodes and conditions, there are many other elements in SG that need to be exported to IMS-LD semantics: variables, edges and actions. This section presents the issues regarding each of their processes of exporting.

Variables and properties

Apart from the special properties explained before, the exporting process must convert all variables in the SG to the UoL. In an SG description file variables are declared at the <sg:init> section of the <sg:node> in which those variables exist. If a variable is to be used in different levels of the hierarchy, it must be declared at every one of them.

XML 8.8 A variable declared in SG

```
<init>
  <set parameter="Qualification" value="0"/>
</init>
```

XML 8.9 The IMS-LD property equivalent to 8.8

```
<imsld:locpers-property identifier="qualification">
 <imsld:datatype datatype="integer" />
 <imsld:initial-value>0</imsld:initial-value>
</imsld:locpers-property>
```

XML excerpts 8.8 and 8.9 show an example of a SG variable and the corresponding property after exporting to IMS-LD. The data type of the property is *integer*. The property type is found during the exporting process analysing the actions performed (on the edges) on that variable. For instance, if at any point in the graph there is a mathematical operation with a variable (like <sg:sum>, <sg:mult>, <sg:max>, etc; see page 77), the corresponding property is marked as integer.

The name of the property is the same as the variable, save for the prefix. The prefix implements a sort of namespace [169] to avoid collisions between properties (i.e. variables). As it has been explained in page 99, the process of exporting SG to IMS-LD involves flattening a hierarchical structure. Thus, two SG variables with the same name (e.g. *time*, *qualification*, etc) at different levels of the hierarchy, or even on two siblings of the same level, collide once they are put at the same unique level on the *imsmanifest.xml*. In order to avoid this problem, a prefix is added to the name of each IMS-LD property. The prefix is the name of the node in which the original value was declared (i.e. the node of its <sg:init> section) as well as the name of every ascendant node up to the root. The root is not included because it is common for all variables in a graph; thus, the variables declared at the root node wear no prefix when exported.

Looking at the former example again, if the variable *qualification* is declared in a node *Exercise3*, included in *Cache* and included in a node *Computer Architecture Course*, the resulting portion of the *imsmanifest.xml* is presented in XML excerpt 8.10.

XML 8.10 The IMS-LD property equivalent to 8.8, with its prefix

```
<imsld:locpers-property identifier="Cache_Exercise3_qualification">
 <imsld:datatype datatype="integer" />
 <imsld:initial-value>0</imsld:initial-value>
</imsld:locpers-property>
```

A final step must be performed relating to variables, and is that of initialization. In a SG, the <sg:init> section has the mission of declaring variables as well as initializing their values, in order to make the sequencing predictable and reusable. Every time the student enters a node, the initialization actions must be performed. Therefore, some additional XML code must be added to the *imsmanifest.xml* to perform the same operations every time the user "goes down" into a node (i.e. goes to one of its entrance nodes): this results in the *if* condition depicted in XML excerpt 8.11. As the value of the special property *entering* value is checked to be *true*. This assures that the variables are only initialized when the student enters the node (i.e. not when returning to the entry node from a sibling node).

XML 8.11 Condition to initialize the properties when the user enters in a node

```
<imsld:if>
 <imsld:and>
  <imsld:is>
   <imsld:property-ref ref="currentActivity" />
   <imsld:property-value>Exercise3</imsld:property-value>
  </imsld:is>
  <imsld:is>
   <imsld:property-ref ref="entering" />
   <imsld:property-value>true</imsld:property-value>
  </imsld:is>
 </imsld:and>
</imsld:if>
<imsld:then>
 <imsld:change-property-value>
  <imsld:property-ref ref="Cache_Exercise3_qualification"/>
  <imsld:property-value>0</imsld:property-value>
 </imsld:change-property-value>
</imsld:then>
```

Edges

In a SG, transitions between nodes are determined by the edges, their conditions and their actions on the environment. For reproducing this behaviour in IMS-LD it is necessary to create equivalent structures of conditions and actions. The structure of an edge in SG is illustrated in XML excerpt 8.12. The equivalent in IMS-LS is a pair *if-then* in the section of conditions of the *imsmanifest.xml*, as portrayed in XML excerpt 8.13.

The structure changes in IMS-LD, but the functionality is preserved. In an informal way, this transition can be described as: "*If* going out of 'This exercise', *and* conditions for this edge are met, *then* perform actions associated to this edge and *then* set 'A neighbour exercise' as current activity". For any edge in a SG, a similar structure is created in the *imsmanifest.xml*. The first step must be checking if the last activity delivered was the origin of the edge. Therefore, conditions and actions are not taken into account unless their edge might be the one selected by the student; in other words, the student can only follow those edges that start at the current node.

XML 8.12 SG edge

```
<exercise name="This exercise" ...>
 <edge destination="A neighbour exercise">
  <condition>
   <!-- * * * Condition here * * * -->
  </condition>
  <actions>
   <!-- * * * Actions associated to this transition * * * -->
  </actions>
 </edge>
</exercise>
```

XML 8.13 IMS-LD equivalent to 8.12

```
<imsld:if>
 <imsld:and>
  <imsld:is>
   <imsld:property-ref ref="export_SG2LD-last-unit" />
   <imsld:property-value>This exercise</imsld:property-value>
  </imsld:is>
  <!-- * * * Condition here * * * -->
 </imsld:and>
</imsld:if>
<imsld:then>
 <!-- * * * Actions associated to this transition * * * -->
 <imsld:change-property-value>
  <imsld:property-ref ref="currentActivity" />
  <imsld:property-value>A neighbour exercise</imsld:property-value>
 </imsld:change-property-value>
</imsld:then>
```

Actions

Actions are used to modify the value of variables in the environment. SG actions are mostly equivalent to <imsld:change-property-value> actions in IMS-LD, and present no important difficulties for the exporting process. The example in appendixB illustrates the process.

8.3. Why not IMS-SS?

IMS Simple Sequencing (IMS SS) is a specification used to describe paths through a collection of learning activities, as described in Section 4.4.4. IMS SS relies on the concept of learning units, that are organized into a hierarchy tree. A parent activity and its children are referred to as a *cluster* of activities. Clusters may have sequencing rules and limit conditions associated with them.

Sequencing rules are used to influence the order in which activities are presented to the learner. Limit conditions, such as attempt limits, duration limits and date limits, are used by the sequencing rules to further influence which activity is sequenced next to a student. Sequencing rules and limit conditions are part of the definition model that describes the vocabulary, semantics and values required to execute IMS SS behaviours.

The IMS SS Tracking Model [80] only accepts two data types: booleans and floats. A further constraint on the float type is that the value is normalized between 1.0 and -1.0. The tracking model defines three data models to record the state of an activity and its objectives: a first model to track the timing and completion progress of each attempt on an activity, a second model to track the timing and completion progress over all attempts on an activity, and a third one to track the result status of the objectives of an activity. There is nothing else to record or process information about the student.

IMS Simple Sequencing lacks a student model in which to store part of the information that the system is able to track about him/her, apart from these three aspects. Thus, it is not possible to fully express all sequencings that can be defined by a SG in terms of this specification. For example, the sequencing cannot be influenced according to a "skill level" of the student, because there is no such concept in the specification and there are no means to include it.

It was not the case in the first versions of the SG. They were successfully exported into IMS-SS [69]. In this reference, some interesting conclusions were drawn on the limits of the specification, similar to those of the former section. Complex sequencings are difficult to express with the IMS-SS semantics and the resulting files are big, specially is there different levels in the graph.

8.4. Conclusions: beyond the limits of IMS-LD

The design of this exporting process and its implementation and subsequent tests, have drawn some interesting conclusions about the limits of the IMS-LD specification. The specification is being used for a purpose that is very different from its main goal. IMS-LD aims at describing pedagogic strategies in which collaborative learning and synchronization of different roles are very important. However, it has been used to describe a personal tutoring process in which the sequenc-

ing of learning activities is adapted. This strains the specification to its own limits, revealing clues about its current possibilities and conceivable extensions in the future.

The first realization is related to the condition model of IMS-LD. Its functional paradigm does not scale well if used "as-is". When the number of conditions grows, it becomes more and more difficult to keep track of all conditions and the possibility of side effects increases dramatically. Every condition must be checked for every possible case, leading to a debugging process that is both tedious and error-prone. Although this may be adequate for small control tasks, the creation of a big set of conditions and actions that control a big number of activities becomes infeasible unless some automatic tool is used to help. This clearly shows the value of higher level tools that produce an IMS-LD output[3], reducing this complexity. Appendix B shows a sample of the conditions that are needed for defining an almost trivial sequencing with only one cycle: the resulting sequence of conditions is so long that it is almost infeasible to maintain accessing directly with a XML editor. Complex sequencings like those involved in any serious tutoring process are totally impossible to create and/or maintain using IMS-LD conditions directly.

Another important issue is that IMS-LD does not provide a good support for reflexive cycles when accessing the material. This is a substantial limitation, because it has been showed that going over some known material for revision improves learning, and this is specially true when long periods of time are considered (like it is the case in Life Long Learning scenarios). However, the IMS-LD is oriented towards a sequence of activities that is strongly linear. This comes from the metaphor of the theatre play, and produces consequences as critical as the fact that an activity cannot be uncompleted once it has been set as "completed" by the student. This chapter has showed that it is possible to express a SG, even with its loops, but this comes at a cost: resulting *imsmanifest.xml* files are immense (appendix B provides an example), and a peculiar text box has to be introduced that may distract the student introducing noise in the learning process.

The restriction that, once an activity is completed, it "stays completed in the run" [83], has been found to be excessive and limiting. Such a behaviour is justified when dealing with synchronization elements (e.g. <imsld:act>), but not in the case of activities, when it could be interesting to come back to them and perform them again (e.g. exercises). This restrictions hampers the use of elements like <imsld:on-completion>, as they can be used only once. A possible solution to this problem is to include a distinction between the state "completed and unable to be uncompleted" and the state "completed, but able to be uncompleted and completed again". In such a scenario, both types of completion events would have the ability of performing actions <imsld:on-completion>) when they occurred. This would simplify the adaptation of graph-based sequencing definitions to IMS-LD, making it possible to use some tool like SG to define a complex sequencing strategy without producing enormous results that take too much resources to process.

The size of the resulting UoLs is not excessively big when the exported SG is small. This suggests that SGs could be used as an authoring tool for small UoLs designed for micro-tutoring (small courses composed of few items, more oriented towards remembering than teaching). The designs showed in Section 7.4.3 are very useful in such a scenario.

Finally, it has been found that the mechanism proposed by IMS-LD for the interaction with the user is more limited than other approaches proposed in this thesis. Interaction is only supported with the user through the *global-elements* <imsld:set-property> and <imsld:get-property>: "*The*

[3] An example of an authoring tool based on SG is presented in appendix A

user gets a control in the user-interface to set the value of the property"[4]. This behaviour is limiting, as it prevents to draw data from other resources apart from the XHTML specially prepared for IMS-LD. A lower level approach is more flexible, and the increase in complexity is small. In Chapter 6 it has been proposed to take student input data directly from the HTTP request. This approach presents several advantages like: any resource that places data in the HTTP request (including, but not limiting to, QTI resources and any web page) could be used in the context of a IMS learning design; it is a general solution, as it can be assumed that all communications in the context of IMS-LD are performed over the HTTP protocol; and it is a well-known technique that has been used for many years and has been implemented in many platforms and programming languages. Using the capacities of the HTTP protocol in the context of IMS-LD is a natural solution, as the specification is importantly coupled with the XHTML standard.

As a final conclusion, it has to be said that several problems have arose when trying to express instructional designs (expressed in SG) in terms of IMS Learning Design. It is true that it is possible, and that permits to define a flexible set of adapted sequencings of learning material with a SG and then use IMS-LD to use them in many platforms. But the cost of the process is high: it introduces noise in the approach (i.e. the "continue" box) and the result files (the manifest and the master resource file) are so big that they present memory problems in some systems, even when the original graph is not very complex. This suggests that the goal of IMS-LD, that is, wants to be able to express any instructional design is only partially true.

[4] [83], page 62.

9 Social Swarm Intelligence

> If you want to model human intelligence (. . .) then you should do it by modelling individuals in a social context, interacting with one another. (. . .) there is a relationship between adaptability and intelligence, and noting that social behaviour greatly increases the ability of organisms to adapt.
>
> – Jim Kennedy & Russ Eberhart [93]

Los sistemas basados en técnicas de inteligencia de enjambre fueron presentados en el capítulo 5. Se explicaron los conceptos básicos y se analizaron en detalle varios sistemas, entre ellos el sistema de Paraschool. En este capítulo se presentan tres contribuciones en este campo. Las dos primeras han sido desarrolladas con el sistema Paraschool: un sistema de detección automática de errores en la planificación pedagógica del sistema y un análisis de la validez de la propuesta en un sistema de soporte a la enseñanza. La tercera contribución aplica todo lo aprendido de los otros sistemas para la creación de un módulo para SIT. Este módulo analiza los caminos que siguen los estudiantes y sirve como apoyo para encontrar el camino de aprendizaje óptimo. Por último, se presenta el concepto de tiempo basado en usuario. Este concepto está embebido en los sistemas analizados y es muy importante para cualquier sistema basado en enjambres sociales.

Swarm intelligence based systems were presented on Chapter 5. The key concepts were explained and several systems were studied in detail. The Paraschool system was one of them. This chapter presents three contributions made in this field. First two have been developed with Paraschool: a system for automatic error detection by the pedagogical team and an analysis of the validity of such a system for supporting learning. The third contribution applies the ideas learnt from the other systems for developing a module for SIT. The module analyzes the paths the students follow and helps to find the optimum learning path. Last, the concept of user-based time is presented. It is embedded in the systems studied and it is very important for any system based on social swarms.

9.1. Empirical results of the man-hill paradigm

The ant-inspired optimization process of the Paraschool system, as described in Section 5.3.1, has been developed mainly by Grégory Valigiani, with contributions from other people at the company and the INRIA (www.inria.fr). After the system had been implemented and deployed successfully, the company was interested in having some feedback on how the system was contributing to the learning process of their students (i.e. clients).

This study analyzed the data stored by the system during twenty-one months, from January 2005 to September 2006. It was performed during a research stay during the Fall of 2006. The method employed and the results are presented in this section.

9.1.1. Method selected

The analysis of the empirical data presented serious difficulties. There were not any pre-test or post-test information, only individual items regarding events that had happened at a specific point in time. For example, the Paraschool system stores that a student has solved exercise 245 at a specific date and time. There was not any common reference to compare the students at two specific points in time, in order to observe the evolution of the group.

It was impossible to prepare an experimental setting based on a pre-test/post-test dynamic. The results had already been collected; it was impossible to gather more data for the same period. The plausibility of performing an *ad-hoc* experiment based on pre-test and post-test based on new data had to be discarded due to time constraints.

The Paraschool system has very limited student modelling capabilities. This prevents the possibility of performing an historical analysis of the evolution of the student models. The system does not record the success and failure of students, nor does it try to infer the knowledge level of them. The only student modelling capability of the system is an estimation of the students' global level based on the system designed by Elo [55] to test the level of chess players. The result is that the system records the level of the student and the exercises, and this level is recalculated after each interaction like in a chess tournament. Valigiani used the Elo notation for calculating automatically an estimation of success before students tried an exercise [162]. Unfortunately, preliminary analysis did not show a significant correlation between the Elo number and the progression of the students. This could be due to the fact that the modified fitness function (see Section 5.3.1) does not try to maximize learning by optimizing the results of the students throughout the whole graph, but tries to enhance it by confronting the students with slightly difficult exercises (60% success probability).

Finally, it was decided to analyze the evolution of the students when trying to solve exercises. Results on the exercises was the main information that the system had stored, and the best possibility therein was to study the evolution of the results of the student, i.e. the evolution of the score, an integer number between 0 and 100. Some exercises could produce any score in that interval, others had a binary grading (0 or 100), and there were many intermediate possibilities.

There are thousands of exercises in the Paraschool system, covering decades of topics and with different difficulty levels. It did not make sense to compare the results of the students as a whole.

Thus, the increments in score for different repetitions of the *same* exercise were studied for each learner. The assumption is that a student that learns more will show a higher increase in their scores over the same exercises (until the maximum score is reached).

For each student, the increment in score was calculated for each repetition of the same exercises. This increments were averaged for each exercise, obtaining the average increase that the student had obtained over repetitions of that exercise. (A weighting function was used to that early increments and late falls in score had a bigger impact). This was averaged again for each student, obtaining a number that represented the average increase in results for that student. In other words, this number is the "evolution index" of the student: the higher it is, the better the evolution showed by the student when repeating the same exercises. Assuming that evolution of results and learning are related, that number represents the average learning of each student, as measured by the exercises.

This method was applied three times, with different *time filters*. The concept of time filter was introduced in the study to address the fact that a student (typically those in group F) could obtain a high increase in score simply by repeating an exercise immediately after having failed at it. This behaviour was expected because students feel better when they succeed; it was expected that many would do this in order to end with a high score in the exercises. This behaviour only shows short-term memory, not long-term retention and deep learning of the concepts involved.

It was necessary to eliminate this kind of short-term anomalies in the study. Time filters were implemented for it. A time filter eliminates all repetitions that have taken place in a period of time lower than a certain threshold. For example, if a student has repeated an exercise seven times, five of them on day D (16.14h, 16.16h, 16.34h, 16.41h, 17.30h) and two more on D+1 (9.01h, 9.26h), a time filter of one hour eliminates all executions save D:17.30h and D+1:9.01h. It must be noted that there are two exercises on day D separated by more than an hour, but there are several repetitions in between that have a clear effect on the results.

Two time filters were adopted: one day and one week. The first one tried to eliminate the effects of short-memory and capture long-term memory effects fixed during sleep [24, 171, 125]. The second aimed at apprehending the effect on learning of the personal pheromone (see Section 5.3.1), that prevented the students from repeating the same exercise for a week.

The time filters eliminate partially the noise that exists on the measurement. However, the results of the students are affected by many events, like the realization of other exercises that may be related. This was impossible to address in the current study, as there was not any semantic description of the exercises that could be automatically analyzed. Given the size of the system (10,560 exercises), it was infeasible to add such information for this study due to time constraints.

9.1.2. The groups studied

In order to evaluate the effect of the system on the evolution of the students, it was necessary to divide the students population in at least two groups. The separation was operated with respect to the types of navigation. There are three main modes of navigation in the system[1]:

[1]There are eight different navigation modes in the Paraschool system, but the other five only amount for approximately 1% of the transitions. Additionally, there is a type "0" that corresponds to those transitions performed before the ant-inspired technique was implemented. Type "0" transitions represent an important fraction of the total, but they

Type	Students	Ratio
A	8,515	11.27%
F	40,188	53.20%
G	8,145	10.78%
Others	18,695	24.75%

Table 9.1: Population of groups in the study

Free navigation (F). The students have a table of contents with links to all the exercises in the courses in which they are registered. They can select any exercise and try it. When they do, a new arc might be created if it did not exist between the last exercise they tried and the current one. This type of transition amounted to approximately 50% of the total.

Guided navigation with a teacher (G). This mode is used when the Paraschool system is used in a blended learning environment as a complement to a traditional class. The teacher has all the information about the exercises, and instructs the students to try them at specific moments that fit with the normal course planning. These transitions represented 25% of the total.

Following of suggestions by the ant system (A). Every time the student finishes an exercise, the system described in Section 5.3.1 analyses the outgoing arcs, calculates a fitness value for each one and selects several arcs using a stochastic contest[2]. These options are presented to the student as suggestions for the next exercise. When the student chooses one of these options, that is registered as an A transition. Approximately 25% of the total transitions were of type A.

It should be noted that this is not a taxonomy of students. Students are allowed to use any type of transitions they want. For a example, a student may follow the suggestions of the ant system for a sequence of three exercises, then go the table of contents and select another exercise (free navigation), then follow the suggestions of the ants again. There is no limitation to this.

The students were divided into groups for affinity with each of the three navigation modes. For example, if a student had performed most of the transitions following the ant suggestions, he/she was assigned to group A. The population of the three groups is presented in Table 9.1, where the ratio of students with regard to the total number of user of the system is also presented. Most of the other students (16,734 out of 18,695, 89.51%) were already in the system before the inclusion of the different navigation modes and provided no information.

9.1.3. Results and discussion

The evolution of the students in each group was averaged. The results are summarized in Table 9.2. Each row represents one group: group A mainly follows the suggestions of the ant

have not been taken into account because they do not bear any comparative information.

[2]The real selection method was slightly modified following a requirement from the company, as explained in [160], page 46. The stochastic contest was only performed once, and the rest of the options were selected by order of fitness. This prevented the students from following low-fitness arcs. Although this is justified in the short term, it also reduces the exploration possibilities of the system and makes it more vulnerable to stagnation.

system, group F navigates freely and group G executes the exercises according to the guidance of a teacher. Each column represents the time filter used: on the first column no filter is used, on the second column only those repetitions of an exercise that were separated at least 24h were considered, etc.

Type	No filter	1 day	1 week
A	18.21	7.65	6.29
F	26.16	9.14	7.44
G	35.85	8.46	5.65

Table 9.2: Average of the evolution factor for each group (numbers represent points over 100)

As the minimum time between repetitions increases, a logical decrease on the results due to forgetfulness for the three groups can be observed: the three obtained the highest increases when no time filter was applied, while the worst results are obtained with the filter of one week. However, the effect of time is different for the three groups. The results of group A diminish the less with the one day filter, while those of group G show the steepest decrease in both cases. This is summarized Table 9.3.

Type	No filter $\rightarrow$ 1 day	1 day $\rightarrow$ 1 week
A	57.99%	17.78%
F	65.06%	18.60%
G	76.40%	33.22%

Table 9.3: Decrease on results due to forgetfulness

It can be observed that the group that followed the suggestions of the ant system (A) had much worse results than the other two groups when no time filter was used, but this difference diminished as longer time filters were applied. When the time filter is one week long, the differences are small with the other two groups (group G performs worse if the one week filter is used).

The analysis shows that preventing students of repeating the same exercise twice before one week had passed had a bad impact on their results. This restriction was present in the ant system by means of the personal pheromone ϕ^p (see page 49). Students of groups F and G could —and did— repeat exercises in shorter periods of time, and they obtained better results.

Students of group A could repeat exercises before one week had passed, but they usually did not because they followed the suggestions of the system. The ant system rarely suggests a student to repeat the same exercise before one week had passed, as the multiplicative factor ϕ^p lowers significantly the fitness of that arc. The possibilities for a student from group A to repeat an exercise in a shorter time are either free navigation, or finding another arc (with a different ϕ^p) that led to the same exercise from a different place. The results suggest than a smaller repetition time would have allowed group A to perform better.

The difference was specially evident when no time filter was used: this showed that the hypothesis that students repeat exercises in a short period of time with high increases in scores but modest effects on their learning was true. Therefore, the existence of a restriction that prevented

or restricted very early repetitions was a sensible option. Nevertheless, the current configuration is excessive and a shorter time barrier would be more adequate.

On the other hand, it was expected that this "early repeat" behaviour would be specially evident on group F, but it was group G that showed a bigger dependency with the minimum interval between repetitions. Group G obtained also the best results when no time filter was applied. Both evidences suggest that students under guidance of a teacher repeated the same exercise more frequently than the other groups.

Taking into account that ϕ^p should be better calibrated and allow for earlier repetitions of exercises, the analysis shows that the results of group A are comparable to the other two groups when a time filter of one week is used. Group F performed slightly better, and group G slightly worse, but the difference is not significant. This means that the ant approach does not impose a negative burden on learning. This was specially important for the company. It is true that the results are not better than in the case of free navigation, but the overall result after this research is a better system. The system provides a set of exercises, with both self-directed navigation and sequencing suggestion modes. Every student has the freedom of selecting the sequencing strategy that he/she prefers and the effect on his/her learning will be similar.

It must be noted, however, that the optimization algorithm achieved its goal of optimizing the sequencing of learners with regard to the objective of giving the learners a 60% ratio of success on each exercise ([160], page 92). The question that remains unanswered is if a stochastic model like that of Paraschool permits the correct expression of a good instructional strategy.

Course-related and topic-related groups

This study was repeated for specific groups of students, looking for differences to the general picture depending on some factors. The groups were selected according to affinity of matter or course (e.g. *Terminale S*[3]). Only those groups that had fifty enrolled students or more were analyzed.

There were thirteen of such groups; the results for most of them were similar to the general case. There was an interesting issue, however, in four cases (Table 9.4): the results of group G were much worse than those of groups F and A. This fact suggests that there is a problem between the planning of the course, and the teachers that make use of the material to support their students in a blended learning scenario.

Type	Group 1	Group 2	Group 3	Group 4
A	8.00	12.41	8.13	10.93
F	11.65	13.24	9.88	9.90
G	1.37	-1.37	3.08	0.87

Table 9.4: Results for four specific courses (time filter of one week).

These bad results may be due to the teachers, to an error in the planning (i.e. graph and weights) designed by the pedagogical team or to some lack of understanding and/or communication between

[3]Pre-university course, scientific option.

them and the company. The results do not provide any information to clarify the question. This was the reason that lead to the *Alarm system*, a module implemented for the Paraschool system that detected possible errors on the planning by analyzing the arcs and the level of pheromones deposited on them.

9.2. Error detection by the pedagogical team

The system devised for Paraschool, depicted in Section 5.3.1, uses a graph in which all the nodes are associated with a learning activity. The system was designed for personal tutoring and the learning activities consisted of a theory section, an exercise related to the theory, and a remediation part that corrected the solution proposed by the student and offered some feedback on it.

As the system contains thousands of items, it is very difficult —if not impossible— to perform a systematic check of each one. During their creation process, several stages of verification and test are performed on each item, but it is unfeasible for the teachers to check every case and every possible sequence. Besides, the pedagogical team is able to assure some homogeneity of the items contained in a topic, but they cannot do that for the whole system. The man-hill system can be used as an auditing system, detecting incoherences between the elements and in the default sequencings defined by the pedagogical team. This was first suggested by Semet [148], and acknowledged by Valigiani ([160], page 57), but not implemented.

It has been explained how transitions are chosen randomly. After an activity has been finished, the system assigns a probability to all outgoing edges and selects one of them to be followed. The probability is calculated as a fitness function depending on the pedagogical weight assigned to each arc by the pedagogical team, the amount of pheromones (positive, negative and personal) deposited on it and some additional factors as described at page 47.

As the students traverse the graph of activities in the Paraschool system, they interact with the exercises at every node. If they succeed, positive pheromones are deposited on the last edges they have traversed. When they fail, negative pheromones are deposited. In both cases, more pheromones are deposited in those edges that have been more recently traversed by the student. The relative proportion between the positive and negative pheromones at every edge determine the fitness of an arc when choosing whether it should be followed by the student or not, as explained in Section 5.3.1.

The analysis of these pheromone levels can be used to detect errors made by the pedagogical team that designed the graphs in the first place. We illustrate this point with an example. The pedagogical team can decide that some edge E, joining exercise X_1 to X_2, is very important and should be followed by students most of the time. Thus, they can assign to it a very high pedagogical weight. In spite of that, the students consistently fail when confronting X_2 after X_1, and the edge E is loaded with a big amount of negative pheromone. Over time, the fitness of this edge will be lower than that of the competing ones, and it will be selected very rarely. But, and this is the important point now, this fact can be detected and reported. It may suggest that there was an error on the initial planning: maybe some assumptions about the student population were wrong, or the pedagogical team made some mistake of another sort (e.g. a typo). The system is capable of detecting this malfunctions automatically without any human intervention apart from the indirect interaction between the students (i.e. *ants*).

A system that used this feature of the system to detect errors in the pedagogical planning was devised on the proposition of Yannick Jamont, chief engineer of Paraschool. This system is being used now in the company as a support tool for the pedagogical teams, in order to improve their course planning. The system is described herein.

Objective

Paraschool wanted a system to detect bad "paths" or "regions" in the graphs set up by their pedagogical team. In this context, bad is roughly equivalent to "avoided by the students because it is too hard". Learning is hampered by too easy or too hard challenges [179]. The fitness function was designed to avoid the former situation —too easy—, as explained in Section 5.3.1. The latter —too difficult— is examined by this system.

From the point of view of the company, there were two goals to be achieved through this process:

- Detect errors in the planning, so they can be corrected: if one transition between two exercises revealed itself to be too difficult, more exercises can be inserted in between to make the transition smoother; if several consecutive transitions had the same problem, it can be the symptom of deeper problems, like exercises not correctly organized into topics or chapters.
- Have a measurement on the quality of the different pedagogical teams.

Paraschool does not employ teachers, but hires groups of teachers for preparing each new course developed at the company (from complete courses for children to specific training courses for employees of other companies). Some of these groups understand better the functioning of the system and thus create a more appropriate pedagogical planning.

Paraschool was interested in developing a technique for measuring the results of the pedagogical teams. If the planning made by a team showed up to be very inadequate (i.e. so that most of it had to be redone by the system through the pheromones), it would not be advisable to hire again the same team for preparing future similar courses. On the contrary, if any of the pedagogical team appeared to be specially skillful for the task, the company would be interested in contacting them again for the preparation of other courses.

Algorithm description

The tool that looked for errors in the graph was divided in two steps. In a first analysis it looked for "suspicious" edges based on the amount of negative pheromones. Big amounts of negative pheromones can be a sign of bad performance on the part of the student and, thus, too difficult transitions between exercises. But negative pheromones are only part of the fitness function (Equation 5.1). Therefore, there was a need for a second and deeper analysis in which the fitness function was partially recalculated in order to find those areas of the graph that were poorly designed[4].

[4]In the jargon of Paraschool, these were the black spots (*points noirs*).

First step: negative pheromones

On the first step the tool looks for those nodes that have an amount of negative pheromones that is too high. The definition of "too high" is inherently ambiguous. In the tests performed at Paraschool, we looked for a measurement that presented a compromise between an excessive number of positives (that would have made the results unmanageable) and a extremely low one (that would have not been very useful). After several iterations, it was decided to take as "suspicious" edges those in which the amount of negative pheromones was *twice* the average amount of negative pheromones in the graph.

$$\phi_i^- > 2 \cdot \frac{\sum_{n=1}^{N} \phi_n^-}{N} \tag{9.1}$$

where i is the ID of the edge and N is the total amount of arcs in the graph. This decision led to an amount of suspicious edges around 10% of the total.

Second step: simplified fitness function

Once all these suspecting edges have been identified, next step is to get the source nodes for them. Then, for every edge that starts at those nodes, Equation (9.2) is calculated and compared with the pedagogical weights set by the pedagogical team. This is a simplified version of the fitness function (Equation 5.2), as follows:

$$W_{rel} + 0.5 \cdot \frac{I_{max} - I}{I_{max}} - (W_{rel} + 0.5) \cdot max(1, \phi^+ + \phi^-) \tag{9.2}$$

where I and I_{max} are defined in Equations 5.3 and 5.4 (page 47).

If there are big discrepancies between the pedagogical fitness and the *a priori* planning (e.g. an edge with a big pedagogical weight has become the one with the worst fitness), the edge is added to the list of "bad" edges. Bad edges are those reported for revision. (It is important to note that, indirectly, revision of a bad arc implies revision of all those arcs near it).

At the same time that they are being found, bad arcs are correlated with bad arcs found up to that point to find "bad paths": ordered sequences of edges that have showed —all of them— to perform worse than expected by the pedagogical team. Sequences of bad edges automatically implies sequences (in fact, trees) of bad nodes: bad paths, in a sense. This shows when the paths designed by the pedagogical team show —in the end— to be incorrect.

Final step: organize results

Finally, results were grouped by topics and course. Then, they were presented to the user. This was a request of Paraschool, so the data was easier to understand.

If it was the case that a bad edge represented a transition between topics, it was presented in both topics. In the case that an edge represented a transition between two courses, it was not

presented in any of them. On the contrary, all bad inter-course[5] edges were presented together in a special category.

It has to be noted that, from the point of view of Paraschool, bad inter-course edges presented little interest. Inter-course and inter-topic are only the result of free navigation on the part of the students, so they give little information about the planning itself. This might suggest interesting relationships between topics and/or courses, but does not serve to the original purpose of finding an indication of the quality of the job performed by the pedagogical teams[6].

The user had the option of looking at the nodes individually, or presented as paths. A modification of the Chu-Liu/Edmonds directed spanning tree algorithm [52, 100], using the inverse of the fitness for each arc as the cost function, was used to sort the set of bad nodes looking for these paths. At the beginning of the project it was suspected that bad arcs could be sorted into sequences of paths with a low fitness. However, results showed that arcs with a low fitness tended to be very close to each other, forming many cycles. Thus, the concept of bad path made little sense in the end and the branches of the spanning tree were not very representative.

The rationale of this issue is that the Paraschool heuristic aims at reinforcing the good learning paths but not the bad ones. A bad learning path (i.e. a sequence of four exercises that leads to failure in the fifth one) is reinforced negatively. This means than any of the arcs in that path is avoided by future students, thus avoiding further reinforcing of the path as a whole.

9.3. Application to SIT version 3: the Swarm Paths Information module (SPI)

When an ITS oriented towards adaptation of the sequencing is created, not all possible sequencings are well suited for learning. That's why some of them are permitted and some of them are not. (For instance, it makes no sense to deliver a student the last assessment if he/she has not been presented the former theory units). In the specific case of the SG sequencer (Section 7.2), that is why some arcs exist between units and some do not. Moreover, the arcs have conditions to match before the student is allowed to travel from one unit to the next one, and this is how the sequencing is adapted to different students with different capabilities or needs (this brings some similarities with *link hiding* [20]).

As it was discussed in Chapter 5, this approach has two weaknesses. First, it relies on some human designer/teacher to design the graphs. While this gives the opportunity of reusing the expertise of a teacher, it makes it harder to maintain the system in the long term. The use of hierarchy mitigates the problem, as some lower-hierarchy graphs can be remade from scratch without affecting the general graph, but it still requires a lot of work to add new learning units to a existing graph.

Additionally, student groups change over time. Different generations have, as a group, different capabilities and needs. It is desirable that graphs offer the possibility of adapting themselves to

[5]It has to be noted that there were not any bad inter-course arcs. Although inter-course arcs existed, it was very uncommon that they were followed frequently. Thus, they did not gain any significant amount of either positive or negative pheromone. In the end, the normal process of evaporation/erosion depletes their values.

[6]The selection of courses is performed by Paraschool, according to the necessities of their clients.

different populations of students, and not only adapting the sequencing of learning units to every student according with some rules. As it is pointed out in [149], the set of paths designed at first might not be adequate.

It was intended to overcome these problems with the addition of some stigmergic capabilities to SIT, borrowing part of the ideas from the Paraschool system. The result is the Swarm Paths Information module (SPI). It has been presented in Chapter 6, and its place in the SIT architecture is depicted in Figure 6.1 (page 62).

9.3.1. Limitations of former solutions

The objective of this module is to reinforce the successful paths, in order to guide students through the optimum path for their learning. Other initiatives presented in Chapter 5 were not adequate:

Paraschool. The ACO-based system used in Paraschool was not adequate for a number or reasons. First, that solution is only adequate for big numbers of students. When the number of students does not exceed 100, it is not feasible to base a system of small deposition of pheromones on an individual basis, because the system takes very long to start being effective (this is a case of cold start problem).

Besides, when the number of students is low, it does not make sense to allow for free navigation, because the number of created links would be excessive. As it was said on page 49, the mean average of arcs per node in the Paraschool system is approximately 25, which is only one order of magnitude bellow the number of students that use a tutoring system in many practical scenarios. This makes the cold start problem worse. Additionally, the objective was building a module for SIT, so it could be used for all tutoring systems. The creation of new arcs is not something to be done by SIT, but by each of the sequencers.

A third limitation of the system is that it optimizes the learning path for all the students. The differences between students are not taken into account. There is a need for an additional level of adaptation performed by pheromones: adaptation for the group is the first step, but adaptation for the individual learner is necessary. The use of a personal pheromone to avoid frequent repetitions of the same exercise provides this additional level partially, but the limitations of the approach are clear. First, it is a strictly personal approach, so the results from the group do not help the user, i.e. there is no real swarm intelligence working. Second, as it has been showed in Section 9.1, it has a negative effect on the results of the students.

Learning Networks. This system did not have any evaporation mechanism implemented, what can make it vulnerable to stagnation. Evaporation is a base feature of ant-inspired algorithms to avoid that. Although the authors claim that there is always a probability of choosing an option different from the dominant one, this happens so rarely that the system will not react to changes in the environment (e.g. new activity nodes, changes on the student population) once there are clearly stablished paths, even if they are not optimal.

Another negative point of the recommender system of LN is that it does not acknowledge the relative size of the pheromone values. The case in which two different transitions have

been successfully followed by 50 and 20 students or by 5 and 2 lead to the same results. The first case is clearly more significant, but this fact is not addressed by the system.

Finally, there are several desirable features of the Paraschool system that are not present in LN. The first one is the possibility of several suggestions: in LN there is only one activity node suggested, either if it is the dominant (i.e. more successful) or all the nodes have similar rates of success. This shortens the awareness of the student, as both cases are clearly different but the information is not showed.

The second feature is the recording of information regarding failure. It can be the case that a transition becomes the dominant one if the first students that follow it were successful. After that, most students fail, but this is not recorded. Therefore, students are usually suggested the same arc, even if they all fail. In evolutionary terms, this is called an Stalinist regime [91] This problem is addressed in Paraschool using negative pheromones.

Finally, the system lacked a personal level of adaptation apart from the communal one; and the learning path was of length 1, which is very limited.

There was another limitation of both systems that needed to be addressed: the awareness of the student had to be increased about the swarming process. Not only awareness of the context has a positive impact on the learning process [92], but human beings present swarm behaviours in which they are aware of other group members (unlike ants, but similarly to bird flocks [141]). Raising the awareness of the student about how their peers have performed, addressing both success and failure, was an issue that none of these two systems had confronted.

9.3.2. Proposed solution

The devised mechanism brings similarities to the ant-inspired mechanism used in Learning Networks. When a unit is delivered to the student, his/her success or failure is recorded. If the student was successful, information is stored about the actual activity and the former ones (the *pedagogical path*). Therefore, the edges have both a condition to be met by students and information about how many students were successful when following it *depending on the route taken before*.

This information is presented to the student every time he/she finishes one unit[7]. All the available units (those at the end of an edge with a fulfilled condition) are presented by SIT. If the SPI module is used, each activity is presented with some information attached[8], showing how many students have been successful on each of them, when they were *in the same state* as the student is now. The total number of students that tried the activity is also showed. That way, the student knows the 'ratio of success' for each unit, according to the data collected from his/her peers. As it can be seen in Figure 9.1, the result brings some similarities to a collaborative filtering system [50], but applied to adaptive sequencing.

[7]This behaviour of SIT can be turned on and off by the administrator for specific learning modules or groups of students.

[8]In Learning Networks, this information is showed *indirectly* in the form of one suggestion. This approach has the advantage of simplicity for the student, but it lacks a level of adaptation as the same recommendation is given to any student that is in the same situation.

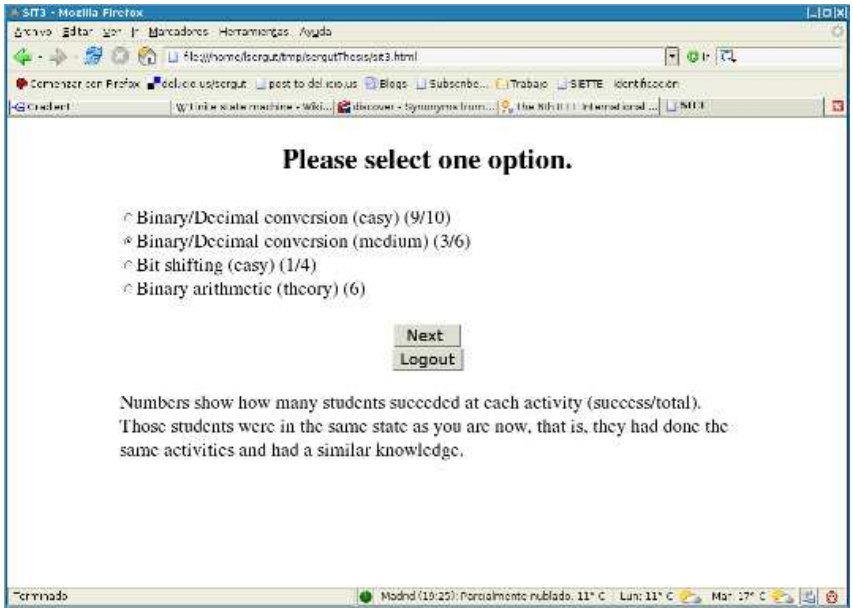


Figure 9.1: Selection screen on SIT3 with SPI module

The information is related to the student's state and the last steps followed, i.e. the learning path. Combining ideas from Paraschool and Learning Networks, the data collected is classified according to the origin of each student, i.e. to the path followed. The length of this path is configured in the SPI module by the SIT administrator. The followed path constitutes the context of the information, thus mitigating one of the problems of collaborative filtering systems (see [49], page 164). The information showed to the student does not classify available activities as according to their fitness only (understanding fit as "bringing the students to local success"), but the classification depends on the context of the students.

The goal of the "success ratio" information is to bring the student to a meta-cognitive state that provides another level of sequencing adaptation. A student that knows that he/she is above average compared with his/her peers can select a unit that has a lower ratio of success but represents a higher challenge. A not-above-average student is able to select those units in which many of his/her classmates were successful. In fact, the students are using the experience of others to help their own learning process.

This represents an additional degree of adaptation. Its big advantage is that it is achieved in a distributed and automatic manner, and gives a sensation of freedom and self-control to the students about its own learning, which is very positive [151].

The term classmate is used here in a broad sense. The SPI functionality is more easily deployed in a blended learning scenario, where students interact directly between them as well as with the tutor; but it can also be used in distance learning scenarios, where the concept of "class" is blurred or does not exist at all. Although students in distance learning scenarios do not have as rich information about their relative skill compared with their peers, they can get that information tacitly from direct (e.g. former exams) or indirect sources (e.g. conversations in fora).

There is an additional advantage on the use of the SPI. It helps the students to take a decision

when the number of activities presented is high, thus avoiding the Paradox of Choice: when the possibilities of choice are too many, no choice is made [147]; if too many possibilities are given to the user, the effort needed to make a selection is so high that the user may give up without making any decision. The information provided by the SPI module creates a differentiation of the available options, providing a level of filtering —achieved collaboratively— to help in the decision process. Additionally, the fact that the ratio is showed explicitly (i.e. 7/14 instead of 50%) enhances the awareness of the students and makes the cold start problem less critical at the same time. As students have the opportunity of knowing exactly how many students were successful at each moment, they intuitively assign a level of confidence to the information provided by the system. A cold system, with little data to provide, will not frustrate its users if it does not provide accurate suggestions, because the students are already aware of the situation. This result is applicable to any collaborative filtering system.

In conclusion, as more and more students use the system, learning paths appear in the same way as natural paths are created in the wilderness: openings become trails, best trails become tracks, and best tracks become paths (and best paths might become routes, i.e. formal training models). The interesting issue is that these learning paths are created with no interference from the human teacher. They are created distributively and automatically, using indirect communication between classmates. This is an stigmergic process.

9.4. Theoretical underpinning: User-based time

When applying swarm intelligence techniques to a social system, natural time cannot be used for evaporation-related processes. In nature, evaporation happens over time, but nature-inspired algorithms must be *inspired*, not *driven*.

Most of the swarm-inspired heuristics (and the first ones, chronologically) used small artificial agents, usually called ants (e.g. ACO). In those cases, all agents act at the same time[9]. In this case, it makes sense to use natural time to evaporate pheromones, or breed new generations and kill the old ones.

But it is not the case when the *agents* are people. People's activities are seasonal and very irregular. Periods of still quietness are followed by short periods of frenetic activity, without warning. In order to make systems that are responsive enough in both extreme situations, time must be made relative to the *activity* of the users. We have seen that both the Paraschool system and CoFIND had to change from evaporation policies based on natural time to user-based ones. The SPI module in SIT inherently uses user-based time. This issue will have to be addressed by any future system that works with social swarms.

9.5. Conclusions: social swarms and elearning

Three different kinds of results have been presented on this chapter: some are related to the Paraschool system, some are related to SIT, and there is a final theoretical result abstracted from

[9] In fact, this means "forever as far as the simulation runs".

the analysis of these systems and those described in Chapter 5.

In the scope of the Paraschool system, a study has been performed to assess the pedagogical effectiveness of its ant-inspired approach to the problem of collaborative adapting sequencing. The investigation was performed on an atypical set of data that demanded an original method to examine it. It has been showed than the use of the personal pheromones, while interesting because it allows for some personal adaptation, was not correctly calibrated: one week was too much time and students that followed the recommendations of the system showed worse results than those that preferred to rely on free navigation. Still, the swarm approach proved to be effective for the learning of the students, and slightly more appropriate than the designs by the pedagogical teams.

A second result is related to this point. An alarm system has been developed in order to detect bad pedagogical plannings. This was achieved analysing the pheromones deposited by the students on the graph.

Lessons from the Paraschool system, plus Learning Networks (and, to a extent, CoFIND) have been applied to the development of the Swarm Paths Information module for SIT. It combines the best ideas of the three systems. From the first one it borrows the success and failure pheromones that create two feedback loops and enhance the stability of the system, making it more robust against stagnation; the idea of learning path is also taken. From LN, it uses a set of pheromones based on integers, because this is easier to understand if showed on the screen: LN do not show stigmergy information to the student, but the SPI module does, getting some inspiration from CoFIND evolutionary interface.

Finally, the concept of user-based time has emerged from all the systems analysed and developed. Some of the social swarm systems used natural time in their early versions, and they suffered from seasonality (e.g. summer vacations depleted the system pheromones) and difficulty of calibration. User-based time avoids seasonality problems and makes a system easier to understand and calibrate.

Part III

Conclusions and Future Work / Conclusiones y trabajo futuro

10 Conclusions

εν οιδα οτι ουδεν οιδα
[Sólo sé que no sé nada]
– Socrates

Este capítulo presenta las principales conclusiones extraídas de los estudios realizados a lo largo de los capítulos anteriores. En concreto, se delinean las contribuciones de la presente tesis, enfatizando las relaciones entre ellas y cómo se han alcanzado los objetivos establecidos al principio de la tesis.

This chapter presents the main conclusions that have been drawn from the studies of the former chapters. In particular, the contributions of the thesis are depicted, emphasizing the relationships between them and how the objectives set at the beginning of the thesis have been achieved.

10.1. Towards social swarm enhanced elearning

This thesis situates itself between two worlds: the world of elearning and the emergent field of swarm intelligence applications. The first is a mature field that, despite the relatively recent appearance of information technologies, takes its roots in the sciences of education. The second one is an emerging field that is inspired by the studies on social insects, that show how complex behaviours can emerge from the actions of simple beings with limited communication capabilities, resulting in auto-organization of complex systems without any external guidance. The implications are changing our view about technology and even ourselves.

This thesis aims at strengthening the link between these two worlds, because elearning can

benefit from swarm intelligence techniques both for pursuing new goals and for supporting some of the characteristics of traditional learning that are not well captured by current systems. Self-organizing behaviour of groups of learners is common in traditional learning environments, and processes of collaboration and clustering take place continuously without external intervention or guidance. In a computer network, the ability to create virtual spaces which adapt and change opens a range of new opportunities. In particular, the possibilities afforded by systems which capitalize on stigmergy are immense, allowing effective use of the sheer scale of the Internet in appropriate ways.

Swarm intelligence applied to elearning transfers the responsibility of learning guidance from the teacher to the learners, distributing the cognitive load between them so the task is affordable. This responsibility balancing makes the teaching/learning process more robust, as the number and importance of critical spots in the system diminishes. The implications on distance learning scenarios —where interaction with the teacher is limited— and Life Long Learning ones —where the role of the teacher is blurred— is evident. Nevertheless, the process of softening the central guiding role of the traditional teacher is part of the philosophy encouraged from the new tendencies in education and the political construction of a European space for higher education.

From the many aspects of elearning, this thesis has focused on the problem of sequencing. The rationale is twofold. First, it is a problem whose current solutions are not totally satisfying. Second, the application of swarm intelligence techniques to it is natural, as the most successful applications of swarm intelligence outside elearning have dealt with path optimization problems. Although there are differences between the goals of these applications and those of sequencing in elearning —nonetheless the fact that the goal of sequencing adaptation is personal and not communal— there are many points in common.

It was decided that the ideas developed on the thesis were going to be applied to the field of intelligent tutoring because of many reasons. First, it is a well established field in the broader field of technology enhanced learning, whose foundational parameters are more than twenty-five years old and are based on psychological results from the middle years of the twentieth century. Second, ITS can be applied in many scenarios, from distance learning ones to blended learning where they support the work of the teacher. Related to this, the third reason for the creation of an ITS was that it had immediate application on the daily teaching routine of the university where this thesis has been developed.

The results of this research are many and varied. They are presented now individually, focusing on the contributions they represent, following the thematic distribution of the rest of the thesis.

10.2. SIT: A System for Intelligent Tutoring

A modular platform for the development of Intelligent Tutoring Systems with a focus on the adaptation of the sequencing of learning activities has been successfully designed and implemented. The name of the platform is SIT and has been described in Chapter 6. Two different ITS for two different domains have been designed and implemented with SIT, and they have been tested with good results, validating the “tutoring through sequence adaptation” approach.

SIT has no resources of its own, and neither have ITS built with it. It is oriented to the reuse

of educational resources that are available on the web. This encourages reuse of learning content. It can be seen how content is interchanged and reused in blogs and other similar applications on a daily basis. SIT stimulates similar behaviours by identifying resources with URLs; once the ITS designer has knowledge of some educational resource on the web —either made by herself or not— it can be referenced by its URL and SIT is responsible of retrieving it, showing it to the student and capturing the results of the interaction between them.

The platform has a modular architecture that makes it both robust and scalable. New functionality can be easily added to the platform by designing a new module at a low cost. This has several implications, specially for the creation of tutoring systems. In the context of SIT, an ITS is created by selecting the corresponding resources and implementing an instructional strategy in the form of a sequencer. A sequencer is a SIT module that analyzes the information it has got from a learner (e.g. last results from activities, past history, etc) and adapts the sequence of learning activities.

The SIT architecture emphasizes an important separation between learning content and its sequencing. Content can be provided by different experts than those who design the order in which it has to be presented to the students. The creation of content and the definition of its sequencing are two conceptually different processes —yet very related— and this separation allows different learning providers to specialize on different tasks, becoming more efficient.

Furthermore, the modular structure of SIT allows for the easy comparison of different sequencers or learning modules with the same objectives. This was a key factor in its design and will be used in the future to experimentally test the fitness of diverse sequencing strategies and learning models.

10.3. Sequencing Graphs

Sequencing graphs have been designed as a way to define a set of adaptive sequencings given some learning activities. Sequencing graphs are a specialization of finite automata that take into account many of the particular aspects of the learning process. They have been designed to be simpler to create than other alternatives, yet able to define any type of sequencing, in particular those involving cycles. They are hierarchical, which helps to manage big numbers of learning activities and makes the approach scalable.

Sequencing graphs have been designed with a focus on reuse of sequencings. This can be viewed from two different points of view. First, the upgrade or change of an activity does not affect the sequencing of the set. Because of this, sequencing graphs do not describe the activities themselves, but only reference to their interface description (i.e. variables exported). Second, a set of activities with their sequencing defined as a graph is suitable of being integrated in a broader set, that is, to be included as a container node in a graph of a higher level of hierarchy, in the same spirit that drives the reuse of learning objects. This is achieved using hierarchy and specifying clear inter-level interfaces for the interchange of information (variables).

A sequencer based on SG (SGSeq) has been implemented and used with SIT. Two different ITS have been created based on it. Results of the experiments performed with them have been positive. They have showed that adapting the sequence of a set of exercises using SG has a positive effect on learning, and SG have been validated as a useful tool for adapting those sequences. Besides,

they have also showed that a big number of exercises and theory pages for a middle size course can be sequenced using SG adaptively with good results on learning. Further studies are due for larger courses.

It must be noted that Sequencing Graphs have been tested in the context of SIT, so they have been used to sequence learning resources available on the web, i.e. every activity was linked to a resource. However, the results are relevant to Reusable Learning Objects in a general sense from two points of view. First, RLOs (e.g. IMS-CP files) could be sequenced by SIT following an instructional strategy modelled with SG. It is unclear, however that the model is as useful if the resources at the nodes of the graph are very complex; this is an issue that demands further investigation. Second, and related to this, a mechanism like SG can be used to define the internal sequencing of a complex RLO, surmounting some of the limitations of IMS-SS and achieving it more efficiently than a IMS-LD based solution.

10.4. The limits of IMS-LD for sequencing

A sequencing strategy modelled by a Sequencing Graph cannot be expressed in terms of IMS-SS due to its lack of user model, but this can be done in terms of IMS-LD. Although the second specification is not exactly designed for sequencing, the mechanisms provided by the second specification (i.e. properties, conditions and actions) permit to recreate the same strategy in a UoL. As the scope of SG is more specific than that of IMS-LD, this is an exporting process (i.e. one-way) and not a translating one.

The solution is not trivial and it comes at a cost. Resulting XML files are immense and grow non-linearly with respect to the size of the SG, causing shortage of resources even when the graphs to be exported are of moderate size. Furthermore, the lack of a technique for finishing activities (apart from completing, which cannot be undone in IMS-LD) obliges to use an unusual mechanism (the “continue” box) to implement the same behaviour that is implemented in the SGSeq. Finally, the process and its results have indicated that the condition model of IMS-LD is not adequate if the number of activities is high and the sequencing defined for them is convoluted.

Higher level tools are thus necessary to implement such complex reorganizations of content in IMS-LD, and SG are a sensible candidate for that. Additionally, the process is suitable for designing the sequencing of small numbers of activities, what has applications for processes of embedded micro-tutoring: small sets of exercises that work in a small domain, typically for recovering forgotten skills.

10.5. Swarm intelligence in education

Self-organising behaviour of groups of learners is common in traditional learning environments. In a computer network our ability to create virtual spaces which adapt and change opens up a range of new opportunities. The application of swarm intelligence techniques to elearning it a extremely new research field: the foundations of swarm techniques are themselves almost a newborn field, as its first results date from fifteen years ago; in the case of the application of swarm

intelligence to the problem of learning sequencing, the first preliminary results were published in 2003 [148].

Three different kinds of results in this field have been achieved by this thesis: some are related to the Paraschool system, some are related to SIT, and there is a final theoretical result abstracted from the analysis of the systems studied and/or implemented. All of them, specially the latter, are relevant to any social swarm application.

In the scope of the Paraschool system, a study was performed in order to assess the pedagogical effectiveness of the approach. The investigation was performed on a very atypical set of data, and it demanded an original method to examine it. Several conclusions were withdrawn. First, it was showed than the use of the personal pheromones, while interesting as it allowed for some personal adaptation of the sequence of exercises, was not correctly calibrated: one week was too much time, and students that followed the recommendations of the system showed worse results than those that preferred to rely on free navigation. Still, the swarm approach proved to be effective for the learning of the students, and slightly more appropriate than the designs by the pedagogical teams.

A second result is related to that last point. An alarm system was developed for Paraschool that detected bad pedagogical plannings. This was achieved analyzing the pheromones deposited by the students on the graph depending on their success or failure as they interacted with the exercises.

In the context of SIT, lessons from different systems (e.g. Paraschool, Learning Networks, CoFIND) were drawn and applied to the development of the Swarm Paths Information module. It combines the best ideas of the three systems. From the first one it takes the idea of positive/success and negative/failure pheromones, that creates two feedback loops and enhances the stability of the system, making it more robust against stagnation; the idea of learning path is also taken from there, because the recent history of a student has an impact on its results. Like LN, it uses a set of pheromones based on integers, because it is easier to understand if showed. LN do not show stigmergy information to the student, but the SPI module does. This is inspired in CoFIND's interfaces for the selection of qualities, that competed for the user's attention using different font sizes and positions on the screen. The SPI module helps the students to select the next activity they have to perform showing the relative levels of success that other students have achieved for each of them, ordered according to that information. The information provided by this module is adapted to each learner because it is based on the pedagogical path that the learner has followed.

Finally, from all the systems analyzed and developed the concept of user-based time has solidly emerged. Some of the social swarm systems used natural time in their early versions, and they suffered from seasonality (e.g. summer vacations depleted the system pheromones) and difficulty of calibration. On the other hand, user-based time can be stated as "time only passes if the user does". This avoids seasonality problems and makes the systems easier to calibrate: it is easier to think about explicit numbers of events that occur than to calculate time windows for events that have some probability of happening.

Evaporation of pheromones is important in swarm ant-inspired methods. Lack of evaporation might make prevent a system of reacting to changes. Evaporation driven by user-based time is a necessity for social swarm systems and it might be of interest to study its application even on systems using artificial ants.

The application of swarm intelligence techniques to elearning is a new field, but it is my im-

pression that it is going to be one of the main ideas on the field during the next decade. It has a lot of potential, the number of applications in other fields grows every year, and the social aspect of learning (i.e. collaborative learning, social navigation, social sequencing) is attracting the interest of an important fraction of the research community.

11 Líneas de trabajo futuro

> . . . as one never knows what will the lines of progress be, it is always dangerous to condemn what is not fashionable at the moment.
>
> – Bertrand Russell [146]

Durante el desarrollo de la presente tesis se han propuesto soluciones para determinados problemas, pero esto también ha traído a la luz otros problemas que originalmente no se habían previsto. Estos nuevos problemas se describen en este capítulo.

This thesis has proposed solutions to several problems, but this has produced other problems, different and unexpected. These unsolved problems are described here. Some of them involve a deeper study of the solutions proposed, while others are completely new questions the thesis has identified.

11.1. Grafos de secuenciamiento en aprendizaje colaborativo

Una posibilidad interesante es extender el uso de SG para actividades colaborativas. En la presente tesis se han utilizado los grafos en un escenario de tutoría personal, pero la importancia del trabajo colaborativo lleva años siendo remarcada desde el campo de la psicología cognitiva [127, 66], ha sido subrayado por los acuerdos políticos en materia de educación a nivel europeo y es uno de los puntos en que hace hincapié la especificación IMS Learning Design.

Sería interesante estudiar hasta qué punto el paradigma de los grafos de secuenciamiento es aplicable a entornos de trabajo colaborativo, especialmente a la hora de adaptar una secuencia de actividades a diferentes grupos y a la hora de sincronizar las acciones de grupos con diferentes papeles (*roles*).

11.2. Feromonas

En esta tesis se han mostrado tres sistemas que utilizan algún tipo de feromona para indicar un camino, inspirándose en el comportamiento de las hormigas. Su estudio ha abierto nuevas líneas de investigación que podrían llevar a mejores resultados.

Granularidad en la deposición

La primera línea es la evolución de políticas binarias de deposición de feromonas a políticas con un mayor grado de granularidad. Es esperable que una mayor granularidad en la certificación de éxito o fracaso permita obtener mejores resultados. Las políticas de decisión blanda han obtenido mejores resultados que sus contrapartidas duras en varios campos como el enrutado [59] o la detección y decodificación en sistemas 3G [94]. Posteriores investigaciones permitirán aclarar este asunto. Si los resultados fueran positivos, podrían aplicarse a otras aplicaciones de técnicas de enjambre, desde otros sistemas sociales hasta sistemas de optimización basados en agentes artificiales (v.g. ACO).

Adaptabilidad de ϕ^p

En el sección 9.1 se mostró que la feromona personal ϕ^p era demasiado restrictiva. Dificultar que los alumnos repitan el mismo ejercicio durante un periodo de una semana tenía un efecto negativo sobre el aprendizaje de los alumnos. Este intervalo debería ser menor. Sin embargo, no sabemos todavía cuál sería el tiempo óptimo. Adicionalmente, es muy probable que este tiempo sea diferente para cada persona. Por ello, sería conveniente el diseño de un segundo nivel de adaptación personal.

Familias de feromonas

En todos los sistemas estudiados, las feromonas son comunes a todos los individuos (la única excepción es la feromona personal presente en el sistema de Paraschool). Esto es una imitación del comportamiento de las colonias de hormigas, que suelen tener sólo un tipo de feromona para la optimización de caminos[1]. Sin embargo, esto resulta en la optimización del secuenciamiento en función del conjunto de la población, lo cual no es realmente óptimo para todo el mundo. El sistema de Paraschool ataca este problema mediante el uso de feromonas personales que sólo afectan al propio individuo que las deposita. El módulo SPI de SIT usa un enfoque diferente, proporcionando más información al alumno para aumentar su nivel de conciencia respecto a sus compañeros, y dejando en sus manos la posibilidad de adaptar su secuenciamiento en función de su relación con sus compañeros.

Ambos enfoques se podrían beneficiar de una agrupación de las feromonas por *clases* de individuos. Esto es una estrategia a medio camino entre las feromonas universales y las feromonas personales. Según este esquema, las feromonas se depositan en función de la clase del individuo,

[1] Tipos diferentes de feromonas diferentes se utilizan, sin embargo, para otras tareas como limpieza de hormiguero o transporte colaborativo [16].

y sólo afectan al secuenciamiento de aquellos de su misma clase. Las clases pueden escogerse mediante técnicas de modelado de usuario, o de forma voluntaria por los propios alumnos. Éstos pueden agruparse entre sí de forma autónoma, llevando a la creación de redes sociales similares a las existentes en sistemas de marcado colaborativo —como del.icio.us—, creación colaborativa —como algunos blogs— o las redes de mensajería instantánea.Otra posibilidad es estudiar el uso de feromonas *voluntarias*. En vez de depositarse automáticamente en función del éxito o el fracaso del alumno, sería el propio alumno quien depositaría feromonas en función de su sensación personal de utilidad o no utilidad de las actividades realizadas. Estas feromonas subjetivas podrían combinarse con las otras, o proporcionar un sistema paralelo de optimización.

11.3. Integración de SIT con .LRN o Moodle

El prototipo de SIT ha demostrado su fiabilidad en los experimentos que se han realizado para probar la utilidad de los grafos de secuenciamiento como herramienta para el desarrollo de estrategias pedagógicas. Sin embargo, el mundo del elearning ha evolucionado desde los comienzos de SIT hace cuatro años. Hoy en día, la ubicua presencia de Internet en empresas, universidades e instituciones de todo tipo ha permitido que florezcan varios productos destinados al apoyo telemático a la formación. Al cabo de los años, dos herramientas han conseguido consolidarse y reunir a su alrededor una mayor base de usuarios: .LRN [173] y Moodle [174].

.LRN fue desarrollado originalmente en el Massachussets Institute of Technology, y ahora su desarrollo es liderado por un consejo de dirección presidido por Carl R. Blesius. Moodle es el resultado de la tesis doctoral de Martin Dougliamas, y es desarrollado por una comunidad de desarrolladores liderada por el propio Dougliamas. Ambos proyectos son software libre, están disponibles en numerosas lenguas diferentes y varias empresas desarrollan negocios a su alrededor. .LRN está programado en Tcl/Tk y Moodle está programado en PHP.

La integración de SIT con estas plataformas aumentaría su difusión, y permitiría aumentar su base de usuarios. De esta forma, se profundizaría en su utilidad para la comunidad educativa, obteniendo un mejor producto. Por otro lado, SIT podría beneficiarse de varias de las funcionalidades ya implementadas en estas dos plataformas, como por ejemplo los mecanismos de comunicación síncrona (chat). El uso de estos mecanismos en colaboración con SIT abre interesantes líneas de investigación. La posibilidad de comunicarse directamente con otros alumnos que estén en el mismo contexto aumenta la conciencia del alumno, lo cual puede resultar positivo para el aprendizaje [19]. En el caso de SIT, el contexto es el camino recorrido recientemente. La posibilidad de hablar en tiempo real con otros estudiantes similares podría acelerar el proceso de creación de caminos si lleva a que los alumnos tomen decisiones en común y vayan a la vez por los mismos caminos.

La posibilidad de comunicación síncrona en función del contexto también abre la puerta a la interacción sincronizada con las actividades, incluso si son actividades individuales, lo cual lleva a un escenario a medio camino entre el aprendizaje colaborativo [128] y el aprendizaje individual. Las aplicaciones de este tipo de escenario mixto en aprendizaje a distancia son evidentes y suponen un grado de interactividad mayor que un simple foro.

Parte IV

Appendices / Apéndices

A El editor de grafos de secuenciamiento

Los grafos de secuenciamiento (SG) analizados en el capítulo 7 deben entendidos por la máquina para ser utilizados. En el caso que nos ocupa, la máquina es el secuenciador basado en SG de SIT (sección 7.3) y el formato de intercambio es un fichero XML (hay un ejemplo en el apéndice B). La realización de un fichero XML a mano es un proceso tedioso y propenso a errores. Para evitar estos problemas, se ha desarrollado un editor visual llamado SGed, que permite la creación ágil de los SG y facilita su mantenimiento. Adicionalmente, el proceso de exportación a IMS-LD de los SG (explicado en la sección 8.1) convierte al editor en una herramienta de autor de IMS-LD, especializada en los procesos de tutoría inteligente unipersonal.

Sequencing Graphs (SG), presented in Chapter 7, need to be expressed in some special format in order to be understood by the machine. In this case the machine is the SIT SG Sequencer (Section 7.3) and the interchange format is an XML file (there is a sample in appendix B). XML editing is a tedious and error-prone process. In order to avoid these problems, a visual editing tool called SGed has been implemented. This tool eases the creation of SG and their maintenance. Additionally, the exporting process that transforms SG to IMS-LD semantics (explained in Section 8.1) makes the editor an authoring tool for IMS-LD, specialized on personal intelligent tutoring processes.

A.1. El problema

En las primeras fases de la investigación los diseñadores creaban los grafos a mano. Pronto se vio que esto suponía una cantidad de trabajo abrumador y lastraba el proceso de diseño instruccional con dificultades importantes:

Errores tipográficos. Debido a la naturaleza textual del XML y al tamaño que alcanza un fichero XML de descripción de un SG en cuanto éste tiene una complejidad no trivial, el proceso de producción de un SG estaba sujeto a muchos errores. Aun en el caso de utilizar editores

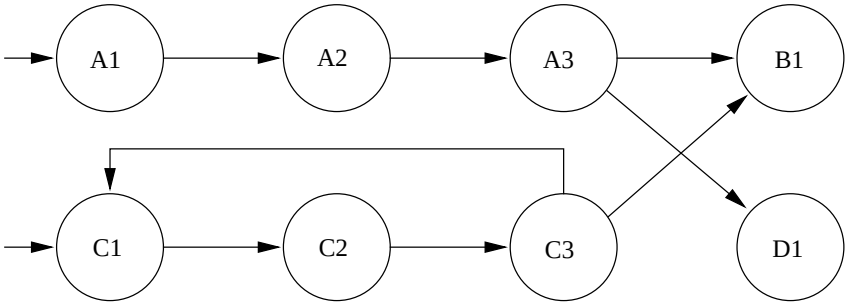


Figura A.1: Ejemplo de error fácil de detectar gráficamente pero difícil de detectar mediante herramientas textuales

especializados en la edición XML, varios errores eran inevitables: erratas en los nombres de los nodos, erratas en las variables, etc.

Se desarrollaron herramientas *ah hoc* para la detección y corrección de errores tipográficos, pero sólo suponían una solución parcial del problema ya que en numerosos casos no era posible distinguir si un nombre era erróneo. Por ejemplo, no es posible saber si un arco que apunta al nodo "Ejercicio 3" (que existe en el SG) es correcto o debería apuntar a "Ejercicio 13" (como era la voluntad original del autor. Otro ejemplo es el que se representa en la figura A.1. En ella se muestra una planificación en que varias actividades están agrupadas en bloques de tres[1]. Después de cada bloque de tres ejercicios, un arco nos lleva a otro bloque distinto en función de determinadas condiciones. En la figura se muestra como un simple (y habitual) error tipográfico C-D produce un bucle indeseado en el grafo.

Este tipo de errores son fáciles de detectar con una herramienta gráfica pero no son detectables mediante análisis textual salvo desarrollando complejas técnicas de inteligencia artificial que exceden el ámbito de esta tesis.

Es importante destacar que fue debido a estos errores que se añadieron a la especificación de los SG conceptos como interfaz inter-nivel y sección de inicialización. El objetivo era evitar errores en la especificación y uso de las variables. Estos resultados fueron útiles después cuando se diseño la herramienta gráfica SGed.

Falta de escalabilidad. A medida que un grafo crece en complejidad, la longitud de su correspondiente descripción XML crece —en el peor de los casos— de forma cuadrática con respecto al número de nodos. Adicionalmente, puesto que la información se muestra de forma lineal en un fichero es difícil obtener una visión de la jerarquía y de los ciclos. Se detectó una tendencia en los diseñadores a pensar en términos de *fichero XML* y no en términos de *grafo*. Esto provocaba que el modelo perdiera escalabilidad, lo cual era contrario a sus objetivos.

A.2. SGed

Para solucionar estos problemas se creó una herramienta gráfica de autor (SGed) para la realización de los grafos de secuenciamiento. En esta sección se presenta dicha herramienta. Los detalles de implementación (arquitectura de clases, lenguajes utilizados, etc) están descritos en [139].

[1] Un ejemplo es el sistema de Paraschool 5.3.1 y sus tríadas teoría-ejercicio-corrección.

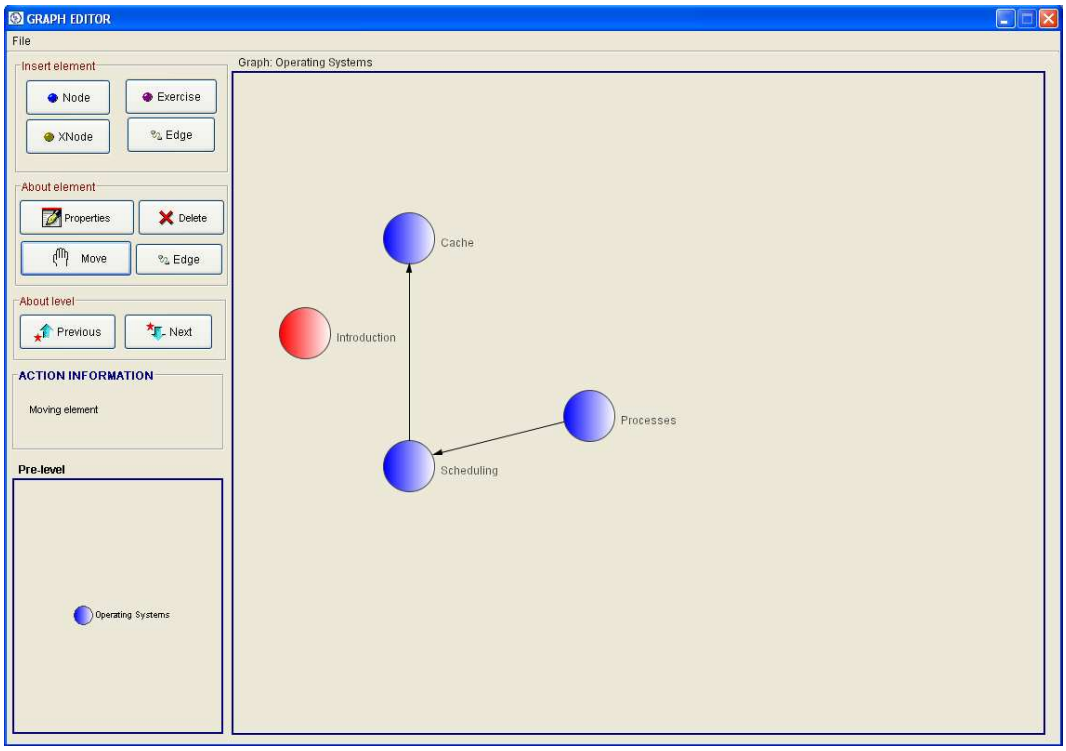


Figura A.2: Pantalla principal de SGed

La herramienta es una aplicación de escritorio que permite crear los SG mediante sencillas operaciones de “arrastrar y soltar”. Tiene una pantalla principal, en la cual se edita el nivel actual del SG y un marco secundario en el que se muestra el nodo padre en el contexto de su nivel como se aprecia en las figura A.2. En cada nivel, el usuario puede añadir al grafo nodos de los tres tipos explicados el capítulo 7: nodos contenedor, nodos de actividad (llamados *exercises* en el contexto de SGed) y x-nodos. Cada tipo de nodo tiene unas propiedades diferentes. Por ejemplo, los nodos “exercises” referencian a un recurso con una URL, pero los nodos contenedor no. Las propiedades se editan al crear los nodos, pero pueden ser modificadas después (figura A.3). Además de crear nodos, el usuario tiene la capacidad de crear aristas entre los diferentes nodos o hacia sí mismos. Cada arista debe tener una condición asociada antes de ser aprobada o el sistema produce un error. SGed facilita la edición de condiciones mediante un interfaz jerárquico (figura A.4). En la misma pantalla de creación y modificación de condiciones de una arista, otra pestaña permite crear y modificar el conjunto de acciones que se ejecutan cuando dicha arista es recorrida por un alumno.

La herramienta SGed permite editar con más facilidad un grafo de secuenciamiento sin perder la perspectiva del secuenciamiento completo. En cada momento el diseñador debe preocuparse sólo de un nivel de la jerarquía, pero no pierde la información de contexto que le proporciona el nivel superior.

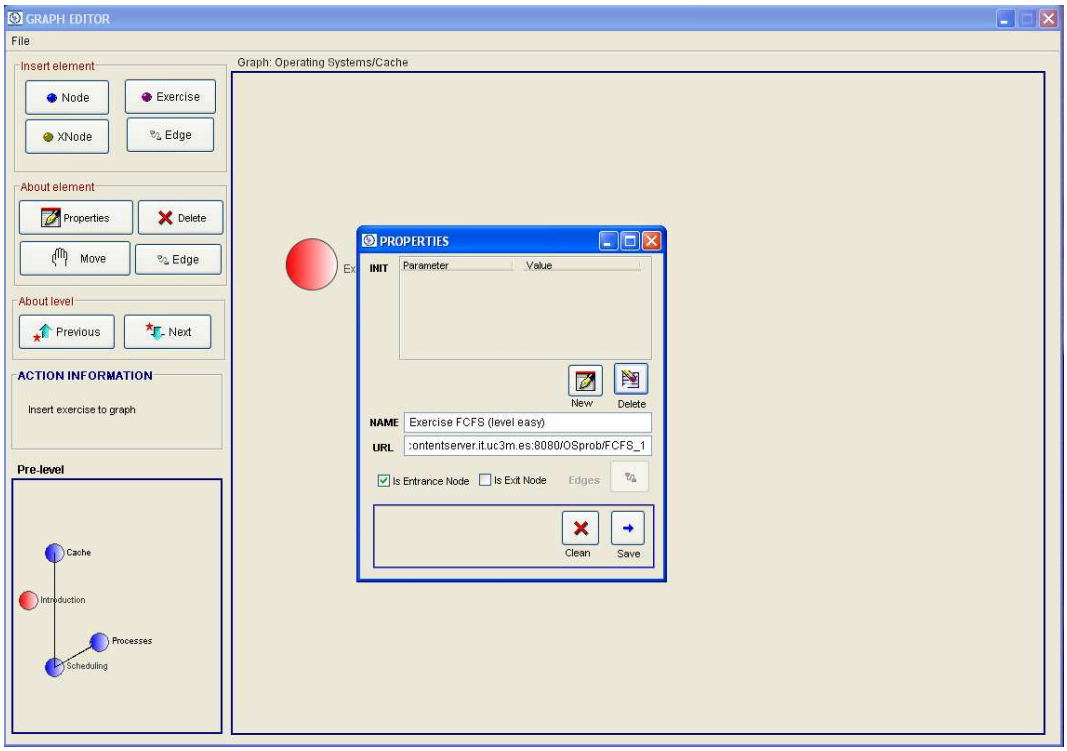


Figura A.3: Edición de las propiedades de los nodos

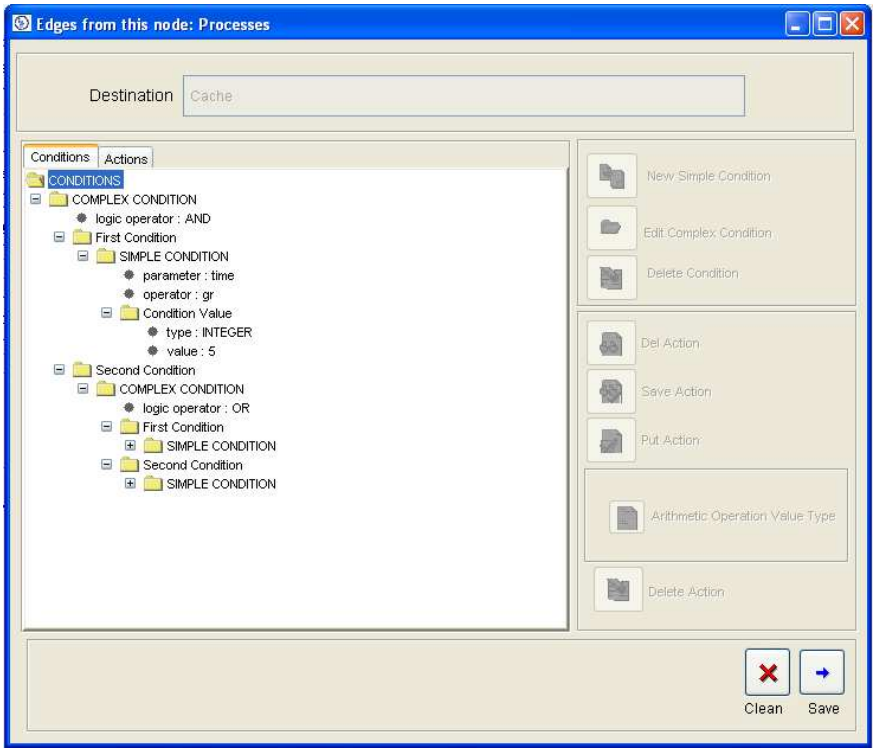


Figura A.4: Edición de condiciones y acciones

B Ejemplo de SG exportado a IMS-LD

En este apéndice se muestra el resultado del algoritmo de exportación de SG a IMS-LD (capítulo 8), para el caso de un grafo muy sencillo. El grafo consiste en sólo cuatro actividades. Tres de ellas están ordenadas en una secuencia lineal, al estilo de las actividades de Paraschool: una unidad de teoría, un ejercicio y una unidad de corrección y realimentación. Como se ve en la figura B.1, las aristas que unen estas actividades tienen una condición que siempre es cierta. En el caso de la arista que conecta la segunda y la tercera actividad, hay una acción que suma la calificación de dos preguntas para obtener una calificación global.

This appendix shows the result of applying the exporting algorithm from SG to IMS-LD (Chapter 8) for a very simple graph. The graph is composed of only four activities. Three of them are sorted in a linear sequence in the style of activities in the Paraschool system: a theory presentation, an exercise and some remediation information. Figure B.1 shows that conditions on the arcs that connect these three activities are always true. The example illustrates several aspects of the exporting process like conditions, actions and hierarchy.

En el nivel superior, una actividad se encarga de preguntar al usuario si quiere repetir el ejercicio. En caso afirmativo, se repite el ciclo; de lo contrario, no se produce ningún movimiento.

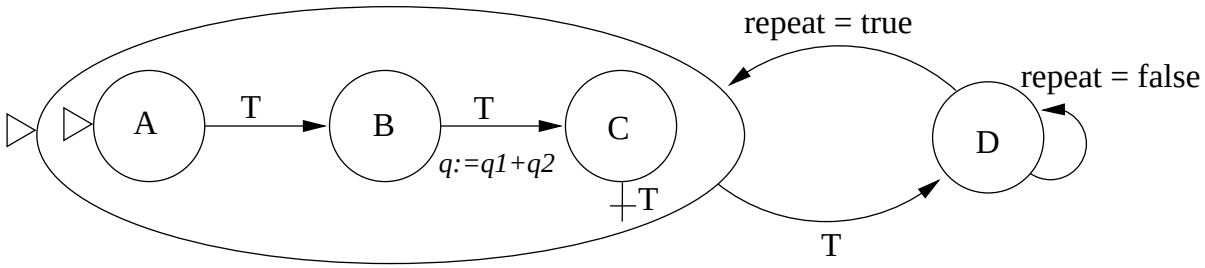


Figura B.1: Ejemplo sencillo de grafo de secuenciamiento

Éste es un ejemplo muy sencillo con escaso valor desde el punto de vista instruccional. Se presenta en este apéndice como complemento a las explicaciones de la sección 8.2. Se ha buscado

ilustrar diversos aspectos del proceso de exportación como: condiciones, acciones, ciclos, aristas reflexivas, etc. El código XML correspondiente al grafo de secuenciamiento de la figura B.1 se muestra a continuación:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<node xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:noNamespaceSchemaLocation="./sg.xsd"
     name="Example of SG export to IMS-LD" is-entrance="true">
 <init>
  <set parameter="repeat" value="false"/>
 </init>
 <interface>
  <variable prev-name=''q'' new-name=''q''/>
  <node name="Exercise a la Paraschool" is-entrance="true">
   <init>
    <set parameter="q1" value="0"/>
    <set parameter="q2" value="0"/>
    <set parameter="q" value="0"/>
   </init>
   <interface-upp>
    <variable>q</variable>
   <interface-upp>
   <exercise name="Presentation" url="http://host/presentation.html"
           is-entrance="true">
    <edge destination="Questions">
     <condition>
      <simple-condition>
       <parameter>alwaystrue</parameter>
       <operator>eq</operator>
       <condition-value>
        <type>STRING</type>
        <value>true</value>
       </condition-value>
      </simple-condition>
     </condition>
    </edge>
   </exercise>
   <exercise name="Questions" url="http://host/questions.html"
           is-entrance="false">
    <edge destination="Grade">
     <condition>
      <simple-condition>
       <parameter>alwaystrue</parameter>
       <operator>eq</operator>
       <condition-value>
        <type>STRING</type>
        <value>true</value>
       </condition-value>
      </simple-condition>
     </condition>
     <actions>
      <put parameter="q">
       <value>
        <arithOpValueType>
         <arithOp>
          <type>add</type>
          <operator>
           <variable>q1</variable>
          </operator>
          <operator>
           <variable>q2</variable>
          </operator>
         </arithOp>
        </arithOpValueType>
       </value>
      </put>
     </actions>
    </edge>
   </exercise>
```

```
   <exercise name="Grade" url="http://host/grade.html"
           is-entrance="false">
    <edge destination="parent">
     <condition>
      <simple-condition>
       <parameter>alwaystrue</parameter>
       <operator>eq</operator>
       <condition-value>
        <type>STRING</type>
        <value>true</value>
       </condition-value>
      </simple-condition>
     </condition>
    </edge>
   </exercise>
   <edge destination="Rest">
    <condition>
     <simple-condition>
      <parameter>alwaystrue</parameter>
      <operator>eq</operator>
      <condition-value>
       <type>STRING</type>
       <value>true</value>
      </condition-value>
     </simple-condition>
    </condition>
   </edge>
  </node>
 </interface>
 <exercise name="Rest" url="http://host/rest.html" is-entrance="false">
  <edge destination="Exercise a la Paraschool">
   <condition>
    <simple-condition>
     <parameter>repeat</parameter>
     <operator>eq</operator>
     <condition-value>
      <type>STRING</type>
      <value>true</value>
     </condition-value>
    </simple-condition>
   </condition>
  </edge>
  <edge destination="Rest">
   <condition>
    <simple-condition>
     <parameter>repeat</parameter>
     <operator>eq</operator>
     <condition-value>
      <type>STRING</type>
      <value>false</value>
     </condition-value>
    </simple-condition>
   </condition>
  </edge>
 </exercise>
</node>
```

A continuación se muestra el código IMS-LD equivalente. Se muestra sólo el *imsmanifest.xml*. El otro elemento fundamental de la conversión, el fichero *resources.xml* ha sido descrito en el capítulo 8. Su estructura no presenta complicaciones, por lo que se omite. Sin embargo, el resultado es en cualquier caso más largo de lo que es deseable en un documento como éste, y se incluye sólo para ayudar a entender los detalles del proceso de exportación. La referencia al fichero *resources.xml* aparece al final del manifiesto.

La complejidad que supone expresar todos los posibles secuenciamientos que describe un SG en los términos de IMS-LD hace que el resultado sea muy voluminoso. El incremento en líneas es superior al 400 % incluso en un caso trivial como el presentado. En grafos más complejos se ha

llegado a obtener un factor de multiplicación de líneas XML superior a 15, requiriendo de máquinas muy potentes para procesar los ficheros resultantes. Esto ilustra claramente las dificultades que supone expresar determinadas estrategias instruccionales en IMS-LD.

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest xmlns="http://www.imsglobal.org/xsd/imscp_v1p1"
          xmlns:imsld="http://www.imsglobal.org/xsd/imsld_v1p0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.imsglobal.org/xsd/imscp_v1p1
          http://www.imsglobal.org/xsd/imscp_v1p1.xsd
          http://www.imsglobal.org/xsd/imsld_v1p0
          http://www.imsglobal.org/xsd/IMS_LD_Level_B.xsd"
          identifier="manifest-116548939">
 <organizations>
  <imsld:learning-design identifier="ld-116548939" level="B" uri="http://www.it.uc3m.es">
   <imsld:title>Example of SG export to IMS-LD</imsld:title>
   <imsld:components>
    <imsld:roles>
      <imsld:learner identifier="role-Student">
       <imsld:title>Student</imsld:title>
      </imsld:learner>
    </imsld:roles>
      <imsld:properties>
        <imsld:locpers-property identifier="export_SG2LD-last-unit">
          <imsld:datatype datatype="string" />
          <imsld:initial-value />
        </imsld:locpers-property>
        <imsld:locpers-property identifier="export_SG2LD-currentActivity">
          <imsld:datatype datatype="string" />
          <imsld:initial-value>Example of SG export to IMS-LD</imsld:initial-value>
        </imsld:locpers-property>
        <imsld:locpers-property identifier="export_SG2LD-validMove">
          <imsld:datatype datatype="string" />
          <imsld:initial-value>true</imsld:initial-value>
        </imsld:locpers-property>
        <imsld:locpers-property identifier="export_SG2LD-entering">
          <imsld:datatype datatype="string" />
          <imsld:initial-value>true</imsld:initial-value>
        </imsld:locpers-property>
        <imsld:locpers-property identifier="export_SG2LD-variableZERO">
          <imsld:datatype datatype="integer" />
          <imsld:initial-value>0</imsld:initial-value>
        </imsld:locpers-property>
        <imsld:locpers-property identifier="Exercise a la Paraschool_q1">
          <imsld:datatype datatype="integer" />
          <imsld:initial-value>0</imsld:initial-value>
        </imsld:locpers-property>
        <imsld:locpers-property identifier="Exercise a la Paraschool_q2">
          <imsld:datatype datatype="integer" />
          <imsld:initial-value>0</imsld:initial-value>
        </imsld:locpers-property>
        <imsld:locpers-property identifier="Exercise a la Paraschool_q">
          <imsld:datatype datatype="integer" />
          <imsld:initial-value>0</imsld:initial-value>
        </imsld:locpers-property>
      </imsld:properties>
      <imsld:activities>
        <imsld:learning-activity identifier="la-1" isvisible="true">
          <imsld:title>Sequencing Graph</imsld:title>
          <imsld:activity-description>
            <imsld:item identifier="item-1" identifierref="resource-1" isvisible="true" />
          </imsld:activity-description>
        </imsld:learning-activity>
      </imsld:activities>
    </imsld:components>
    <imsld:method>
      <imsld:play identifier="play-116548939" isvisible="true">
        <imsld:title>Play</imsld:title>
        <imsld:act identifier="act-116548939">
```

```
    <imsld:title>Act</imsld:title>
    <imsld:role-part identifier="rolepart-1">
      <imsld:title>role_part</imsld:title>
      <imsld:role-ref ref="role-Student" />
      <imsld:learning-activity-ref ref="la-1" />
    </imsld:role-part>
  </imsld:act>
</imsld:play>
<imsld:conditions>
  <imsld:if>
    <imsld:is>
      <imsld:property-ref ref="export_SG2LD-currentActivity" />
      <imsld:property-value>Example of SG export to IMS-LD</imsld:property-value>
    </imsld:is>
  </imsld:if>
  <imsld:then>
    <imsld:change-property-value>
      <imsld:property-ref ref="export_SG2LD-currentActivity" />
      <imsld:property-value>Exercise a la Paraschool</imsld:property-value>
    </imsld:change-property-value>
    <imsld:change-property-value>
      <imsld:property-ref ref="export_SG2LD-entering" />
      <imsld:property-value>true</imsld:property-value>
    </imsld:change-property-value>
  </imsld:then>
  <imsld:if>
    <imsld:and>
      <imsld:is>
        <imsld:property-ref ref="export_SG2LD-currentActivity" />
        <imsld:property-value>Exercise a la Paraschool</imsld:property-value>
      </imsld:is>
      <imsld:is>
        <imsld:property-ref ref="export_SG2LD-entering" />
        <imsld:property-value>true</imsld:property-value>
      </imsld:is>
    </imsld:and>
  </imsld:if>
  <imsld:then>
    <imsld:show>
      <imsld:class class="Presentation" />
    </imsld:show>
    <imsld:hide>
      <imsld:class class="Questions" />
    </imsld:hide>
    <imsld:hide>
      <imsld:class class="Grade" />
    </imsld:hide>
    <imsld:hide>
      <imsld:class class="Rest" />
    </imsld:hide>
    <imsld:hide>
      <imsld:class class="classExitDiv" />
    </imsld:hide>
    <imsld:change-property-value>
      <imsld:property-ref ref="export_SG2LD-entering" />
      <imsld:property-value>true</imsld:property-value>
    </imsld:change-property-value>
    <imsld:change-property-value>
      <imsld:property-ref ref="export_SG2LD-currentActivity" />
      <imsld:property-value>Presentation</imsld:property-value>
    </imsld:change-property-value>
    <imsld:change-property-value>
      <imsld:property-ref ref="Exercise a la Paraschool_q1" />
      <imsld:property-value>0</imsld:property-value>
    </imsld:change-property-value>
    <imsld:change-property-value>
      <imsld:property-ref ref="Exercise a la Paraschool_q2" />
      <imsld:property-value>0</imsld:property-value>
    </imsld:change-property-value>
    <imsld:change-property-value>
```

```
    <imsld:property-ref ref="Exercise a la Paraschool_q" />
    <imsld:property-value>0</imsld:property-value>
  </imsld:change-property-value>
</imsld:then>
<imsld:if>
  <imsld:is>
    <imsld:property-ref ref="export_SG2LD-currentActivity" />
    <imsld:property-value>Presentation</imsld:property-value>
  </imsld:is>
</imsld:if>
<imsld:then>
  <imsld:change-property-value>
    <imsld:property-ref ref="export_SG2LD-entering" />
    <imsld:property-value>false</imsld:property-value>
  </imsld:change-property-value>
</imsld:then>
<imsld:if>
  <imsld:is>
    <imsld:property-ref ref="export_SG2LD-currentActivity" />
    <imsld:property-value>Questions</imsld:property-value>
  </imsld:is>
</imsld:if>
<imsld:then>
  <imsld:change-property-value>
    <imsld:property-ref ref="export_SG2LD-entering" />
    <imsld:property-value>false</imsld:property-value>
  </imsld:change-property-value>
</imsld:then>
<imsld:if>
  <imsld:is>
    <imsld:property-ref ref="export_SG2LD-currentActivity" />
    <imsld:property-value>Grade</imsld:property-value>
  </imsld:is>
</imsld:if>
<imsld:then>
  <imsld:change-property-value>
    <imsld:property-ref ref="export_SG2LD-entering" />
    <imsld:property-value>false</imsld:property-value>
  </imsld:change-property-value>
</imsld:then>
<imsld:if>
  <imsld:is>
    <imsld:property-ref ref="export_SG2LD-currentActivity" />
    <imsld:property-value>Rest</imsld:property-value>
  </imsld:is>
</imsld:if>
<imsld:then>
  <imsld:change-property-value>
    <imsld:property-ref ref="export_SG2LD-entering" />
    <imsld:property-value>false</imsld:property-value>
  </imsld:change-property-value>
</imsld:then>
<imsld:if>
  <imsld:is>
    <imsld:property-ref ref="export_SG2LD-last-unit" />
    <imsld:property-value>continue</imsld:property-value>
  </imsld:is>
</imsld:if>
<imsld:then>
  <imsld:change-property-value>
    <imsld:property-ref ref="export_SG2LD-validMove" />
    <imsld:property-value>true</imsld:property-value>
  </imsld:change-property-value>
  <imsld:change-property-value>
    <imsld:property-ref ref="export_SG2LD-last-unit" />
    <imsld:property-value>
      <imsld:property-ref ref="export_SG2LD-currentActivity" />
    </imsld:property-value>
  </imsld:change-property-value>
</imsld:then>
```

```
<imsld:if>
  <imsld:and>
    <imsld:is>
      <imsld:property-ref ref="export_SG2LD-last-unit" />
      <imsld:property-value>Exercise a la Paraschool</imsld:property-value>
    </imsld:is>
    <imsld:is>
      <imsld:property-ref ref="export_SG2LD-validMove" />
      <imsld:property-value>true</imsld:property-value>
    </imsld:is>
  </imsld:and>
</imsld:if>
<imsld:then>
  <imsld:hide>
    <imsld:class class="Presentation" />
  </imsld:hide>
  <imsld:hide>
    <imsld:class class="Questions" />
  </imsld:hide>
  <imsld:hide>
    <imsld:class class="Grade" />
  </imsld:hide>
  <imsld:show>
    <imsld:class class="Rest" />
  </imsld:show>
  <imsld:hide>
    <imsld:class class="classExitDiv" />
  </imsld:hide>
  <imsld:change-property-value>
    <imsld:property-ref ref="export_SG2LD-last-unit" />
    <imsld:property-value />
  </imsld:change-property-value>
  <imsld:change-property-value>
    <imsld:property-ref ref="export_SG2LD-validMove" />
    <imsld:property-value>false</imsld:property-value>
  </imsld:change-property-value>
  <imsld:change-property-value>
    <imsld:property-ref ref="export_SG2LD-entering" />
    <imsld:property-value>true</imsld:property-value>
  </imsld:change-property-value>
  <imsld:change-property-value>
    <imsld:property-ref ref="export_SG2LD-currentActivity" />
    <imsld:property-value>Rest</imsld:property-value>
  </imsld:change-property-value>
</imsld:then>
<imsld:if>
  <imsld:and>
    <imsld:is>
      <imsld:property-ref ref="export_SG2LD-last-unit" />
      <imsld:property-value>Presentation</imsld:property-value>
    </imsld:is>
    <imsld:is>
      <imsld:property-ref ref="export_SG2LD-validMove" />
      <imsld:property-value>true</imsld:property-value>
    </imsld:is>
  </imsld:and>
</imsld:if>
<imsld:then>
  <imsld:hide>
    <imsld:class class="Presentation" />
  </imsld:hide>
  <imsld:show>
    <imsld:class class="Questions" />
  </imsld:show>
  <imsld:hide>
    <imsld:class class="Grade" />
  </imsld:hide>
  <imsld:hide>
    <imsld:class class="Rest" />
  </imsld:hide>
```

```
    <imsld:hide>
      <imsld:class class="classExitDiv" />
    </imsld:hide>
    <imsld:change-property-value>
      <imsld:property-ref ref="export_SG2LD-last-unit" />
      <imsld:property-value />
    </imsld:change-property-value>
    <imsld:change-property-value>
      <imsld:property-ref ref="export_SG2LD-validMove" />
      <imsld:property-value>false</imsld:property-value>
    </imsld:change-property-value>
    <imsld:change-property-value>
      <imsld:property-ref ref="export_SG2LD-entering" />
      <imsld:property-value>true</imsld:property-value>
    </imsld:change-property-value>
    <imsld:change-property-value>
      <imsld:property-ref ref="export_SG2LD-currentActivity" />
      <imsld:property-value>Questions</imsld:property-value>
    </imsld:change-property-value>
  </imsld:then>
  <imsld:if>
    <imsld:and>
      <imsld:is>
        <imsld:property-ref ref="export_SG2LD-last-unit" />
        <imsld:property-value>Questions</imsld:property-value>
      </imsld:is>
      <imsld:is>
        <imsld:property-ref ref="export_SG2LD-validMove" />
        <imsld:property-value>true</imsld:property-value>
      </imsld:is>
    </imsld:and>
  </imsld:if>
  <imsld:then>
    <imsld:hide>
      <imsld:class class="Presentation" />
    </imsld:hide>
    <imsld:hide>
      <imsld:class class="Questions" />
    </imsld:hide>
    <imsld:show>
      <imsld:class class="Grade" />
    </imsld:show>
    <imsld:hide>
      <imsld:class class="Rest" />
    </imsld:hide>
    <imsld:hide>
      <imsld:class class="classExitDiv" />
    </imsld:hide>
    <imsld:change-property-value>
      <imsld:property-ref ref="Exercise a la Paraschool_q" />
      <imsld:property-value>
        <imsld:calculate>
          <imsld:sum>
            <imsld:property-value>
              <imsld:property-ref ref="Exercise a la Paraschool_q1" />
            </imsld:property-value>
            <imsld:property-value>
              <imsld:property-ref ref="Exercise a la Paraschool_q2" />
            </imsld:property-value>
          </imsld:sum>
        </imsld:calculate>
      </imsld:property-value>
    </imsld:change-property-value>
    <imsld:change-property-value>
      <imsld:property-ref ref="export_SG2LD-last-unit" />
      <imsld:property-value />
    </imsld:change-property-value>
    <imsld:change-property-value>
      <imsld:property-ref ref="export_SG2LD-validMove" />
      <imsld:property-value>false</imsld:property-value>
```

```
  </imsld:change-property-value>
  <imsld:change-property-value>
    <imsld:property-ref ref="export_SG2LD-entering" />
    <imsld:property-value>true</imsld:property-value>
  </imsld:change-property-value>
  <imsld:change-property-value>
    <imsld:property-ref ref="export_SG2LD-currentActivity" />
    <imsld:property-value>Grade</imsld:property-value>
  </imsld:change-property-value>
</imsld:then>
<imsld:if>
  <imsld:and>
    <imsld:is>
      <imsld:property-ref ref="export_SG2LD-last-unit" />
      <imsld:property-value>Grade</imsld:property-value>
    </imsld:is>
    <imsld:is>
      <imsld:property-ref ref="export_SG2LD-validMove" />
      <imsld:property-value>true</imsld:property-value>
    </imsld:is>
  </imsld:and>
</imsld:if>
<imsld:then>
  <imsld:change-property-value>
    <imsld:property-ref ref="export_SG2LD-last-unit" />
    <imsld:property-value>Exercise a la Paraschool</imsld:property-value>
  </imsld:change-property-value>
  <imsld:change-property-value>
    <imsld:property-ref ref="export_SG2LD-entering" />
    <imsld:property-value>false</imsld:property-value>
  </imsld:change-property-value>
</imsld:then>
<imsld:if>
  <imsld:and>
    <imsld:is>
      <imsld:property-ref ref="export_SG2LD-last-unit" />
      <imsld:property-value>Rest</imsld:property-value>
    </imsld:is>
    <imsld:is>
      <imsld:property-ref ref="export_SG2LD-validMove" />
      <imsld:property-value>true</imsld:property-value>
    </imsld:is>
    <imsld:is>
      <imsld:property-ref ref="repeat" />
      <imsld:property-value>true</imsld:property-value>
    </imsld:is>
  </imsld:and>
</imsld:if>
<imsld:then>
  <imsld:hide>
    <imsld:class class="Presentation" />
  </imsld:hide>
  <imsld:hide>
    <imsld:class class="Questions" />
  </imsld:hide>
  <imsld:hide>
    <imsld:class class="Grade" />
  </imsld:hide>
  <imsld:hide>
    <imsld:class class="Rest" />
  </imsld:hide>
  <imsld:hide>
    <imsld:class class="classExitDiv" />
  </imsld:hide>
  <imsld:change-property-value>
    <imsld:property-ref ref="export_SG2LD-last-unit" />
    <imsld:property-value />
  </imsld:change-property-value>
  <imsld:change-property-value>
    <imsld:property-ref ref="export_SG2LD-validMove" />
```

```
            <imsld:property-value>false</imsld:property-value>
          </imsld:change-property-value>
          <imsld:change-property-value>
            <imsld:property-ref ref="export_SG2LD-entering" />
            <imsld:property-value>true</imsld:property-value>
          </imsld:change-property-value>
          <imsld:change-property-value>
            <imsld:property-ref ref="export_SG2LD-currentActivity" />
            <imsld:property-value>Exercise a la Paraschool</imsld:property-value>
          </imsld:change-property-value>
        </imsld:then>
        <imsld:if>
          <imsld:and>
            <imsld:is>
              <imsld:property-ref ref="export_SG2LD-last-unit" />
              <imsld:property-value>Rest</imsld:property-value>
            </imsld:is>
            <imsld:is>
              <imsld:property-ref ref="export_SG2LD-validMove" />
              <imsld:property-value>true</imsld:property-value>
            </imsld:is>
            <imsld:is>
              <imsld:property-ref ref="repeat" />
              <imsld:property-value>false</imsld:property-value>
            </imsld:is>
          </imsld:and>
        </imsld:if>
        <imsld:then>
          <imsld:hide>
            <imsld:class class="Presentation" />
          </imsld:hide>
          <imsld:hide>
            <imsld:class class="Questions" />
          </imsld:hide>
          <imsld:hide>
            <imsld:class class="Grade" />
          </imsld:hide>
          <imsld:show>
            <imsld:class class="Rest" />
          </imsld:show>
          <imsld:hide>
            <imsld:class class="classExitDiv" />
          </imsld:hide>
          <imsld:change-property-value>
            <imsld:property-ref ref="export_SG2LD-last-unit" />
            <imsld:property-value />
          </imsld:change-property-value>
          <imsld:change-property-value>
            <imsld:property-ref ref="export_SG2LD-validMove" />
            <imsld:property-value>false</imsld:property-value>
          </imsld:change-property-value>
          <imsld:change-property-value>
            <imsld:property-ref ref="export_SG2LD-entering" />
            <imsld:property-value>true</imsld:property-value>
          </imsld:change-property-value>
          <imsld:change-property-value>
            <imsld:property-ref ref="export_SG2LD-currentActivity" />
            <imsld:property-value>Rest</imsld:property-value>
          </imsld:change-property-value>
        </imsld:then>
      </imsld:conditions>
    </imsld:method>
  </imsld:learning-design>
 </organizations>
 <resources>
  <resource identifier="resource-1" type="imsldcontent" href="resourcesFile.xml">
    <file href="resourcesFile.xml" />
  </resource>
 </resources>
</manifest>
```

C BNSeq: otro secuenciador para SIT

El modelo de SIT no tiene porque restringirse al uso de grafos jerárquicos. Su arquitectura modular permite utilizar diversos secuenciadores basados en diferentes teorías instruccionales. En este apéndice se describe otro posible secuenciador, basado en redes bayesianas (BNSeq). Las posibilidades de este nuevo secuenciador son muy interesantes, puesto que este tipo de inteligencia no da resultados rotundos sino que infiere un resultado y le da una determinada probabilidad de ser cierto. En la primera sección se da una breve introducción a la teoría de las redes bayesianas y se comenta su utilización en términos generales. En la segunda sección se analiza su aplicación al diseño de un secuenciador para SIT. El apéndice se cierra con detalles de implementación en relación con el interfaz Sequencer (sección 6.3.6).

The SIT model is not restrictred to the use of hierarchical graphs. Its modular architecture allows the utilization of different sequencers based on diverse instructional theories. This appendix describes a new sequencer based on bayesian networks (BNSeq). The possibilities of this sequencer are promising, as the artificial intelligence technique applied does not give hard results but probabilities of accuracy for several possible outcomes. The first section introduces the general concepts of bayesian networks and provides an overview of their use. The second section shows how bayesian networks are applied to the design of a sequencer for SIT. The last section covers implementation issues with regard to the Sequencer interface (Section 6.3.6).

C.1. Redes bayesianas

Una red bayesiana [29, 98] es un grafo dirigido acíclico en el que los nodos representan variables aleatorias de un dominio y las aristas representan dependencias directas entre las variables. Cada arista tiene asociada una distribución de probabilidad condicionada o, en otras palabras, una medida de la dependencia existente entre las variables que relaciona.

Las redes bayesianas permiten obtener conclusiones en situaciones en las que existe incerti-

dumbre. El uso de la teoría de la probabilidad saca el máximo partido del conocimiento disponible, pues capacita para la representación de conceptos difusos, como creencias u opiniones, y las relaciones de dependencia o independencia entre variables que representen hipótesis y/o evidencias. Los resultados obtenidos bajo incertidumbre son aquellos que se han obtenido con una evidencia incompleta, llevando a conclusiones que pueden ser erróneas. Hay al menos tres formas distintas de incertidumbre:

1. **Ignorancia (*ignorance*).** Los límites de nuestro propio conocimiento nos imponen incertidumbre: no conocemos todos los detalles que necesitaríamos saber. Un ejemplo es el conocimiento que tenemos sobre cuánto sabe un alumno respecto a un tema.

2. **Aleatoriedad (*physical randomness or indeterminism*).** Aunque se posea un conocimiento muy completo del problema, pueden existir factores que escapen a nuestro control. Un ejemplo claro es adivinar de que lado caerá una moneda lanzada al aire.

3. **Vaguedad (*vagueness*).** Las premisas que se utilizan son en ocasiones imprecisas. Los ejemplos clásicos son el amor y la belleza, pero el mismo conocimiento aludido más arriba es un concepto sin una definición clara.

El teorema de Bayes

El Teorema de Bayes es la base de la teoría de redes bayesianas. Afirma que la probabilidad de que se cumpla la hipótesis *h* conocida la evidencia *e* es igual a la probabilidad de la evidencia conocida la hipótesis, por la probabilidad de dicha hipótesis y dividido por la probabilidad de la evidencia (ecuación C.1).

$$P(h|e) = \frac{P(e|h) \cdot P(h)}{P(e)} \tag{C.1}$$

C.1.1. Elementos de una red bayesiana

El formalismo de las redes bayesianas es un modelo teórico procedente de la teoría de probabilidades que pretende modelar el conocimiento bajo incertidumbre que caracteriza el razonamiento humano. Se representa el dominio como un conjunto de variables aleatorias con dominios predefinidos. Como se ha comentado en la introducción, las redes bayesianas son modelos gráficos para razonamiento con incertidumbre, donde los nodos representan variables (discretas o continuas) y los arcos conexiones directas entre ellas (en ocasiones estas conexiones son de causalidad). Dichos arcos representan dependencias entre las variables y la fuerza de esta relación se expresa mediante las probabilidades condicionales asociadas a cada arco. La única restricción es que dichos arcos no pueden forzar un ciclo.

El primer paso para diseñar una red es identificar las variables de interés. Los valores deben ser mutuamente excluyentes y exhaustivos, es decir, cada variable debe tener exactamente un valor en cada momento.

La estructura o topología de la red también es importante. Mediante ella se ven las relaciones entre las distintas variables. Dos nodos deben estar conectados directamente si uno afecta o es causa del otro, indicando el arco la dirección causa→efecto. Hay que recalcar que la ausencia de un arco aporta también información, pues indica que hay independencia condicional entre dos variables. A los nodos que son causa de un nodo dado se les llaman *padres* (si hay arco directo al nodo dado) o *ascendientes* (si no lo hay). A los nodos que son efecto de otro dado se les llama *hijos* (si hay arco directo) o *descendientes* (si no lo hay). Cualquier nodo sin padres recibe el nombre de *nodo raíz*, y cualquier nodo sin hijos, *nodo hoja*. Los nodos que no son de ninguno de los dos tipos anteriores se denominan *nodos intermedios*.

C.1.2. Razonamiento con redes bayesianas

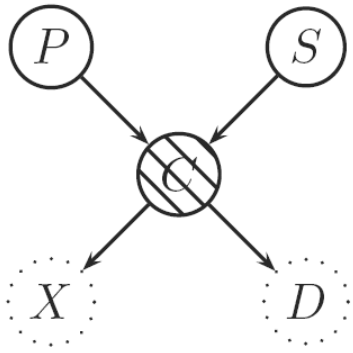
Cuando se observa el valor de una variable, se razona a partir de esa información (evidencia) diagnosticando sus causas o prediciendo sus efectos. En el primer caso necesitaremos calcular la probabilidad de los nodos que están por encima de ella —sus ascendientes—, que representan sus causas; en el segundo, se necesita calcular la probabilidad de los que están por debajo de ella —sus descendientes—, que representan sus efectos. También se pueden combinar ambos tipos de razonamiento. El procedimiento por el cual, a partir de la información de un nodo de la red, se va actualizando la información del resto de nodos se denomina propagación probabilística, inferencia o actualización de creencia (*belief updating*).

Tipos de razonamiento

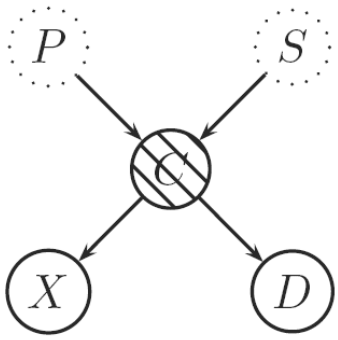
El flujo de información a través de la red no esta limitado a la dirección de los arcos. Como se ha dicho, se puede razonar en la dirección de los arcos (predicción) o en dirección contraria (diagnóstico). Para hacerlo es necesario calcular la distribución de probabilidad para un conjunto de *nodos incógnita* dando valores a algunos *nodos evidencia*. La evidencia es la información que está disponible.

Hay varios tipos de razonamiento, que se diferencian entre sí por el modo en que se propaga la evidencia. Se ilustran en las siguientes figuras, en las que los nodos en blanco representan nodos incógnita y los nodos rayados representan nodos evidencia. Los nodos punteados se muestran por simetría, y representan el conjunto de nodos que no son relevantes en cada caso.

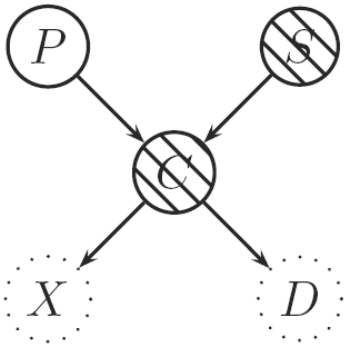
Diagnóstico El conocimiento se propaga desde el efecto (nodos evidencia), hasta las causas (nodos incógnita). Un ejemplo son los diagnósticos médicos. Por ejemplo, la *fiebre* (nodo C) puede ser síntoma de *infección vírica* (nodo P) o *bacteriana* (nodo S).



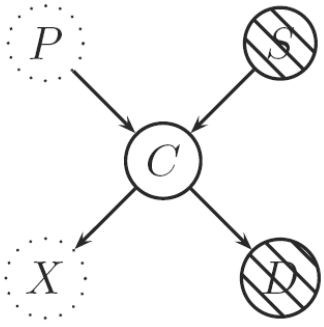
Predictivo El conocimiento se propaga desde las causas (nodos evidencia) hasta los efectos (nodos incógnita) siguiendo los arcos de la red. Un ejemplo sería la predicción meteorológica. Por ejemplo, un valor elevado de *humedad relativa del aire* (nodo C) puede ocasionar *lluvia* (nodo X) o *niebla* (nodo D).



Intercausal Este razonamiento involucra la evidencia del efecto y de alguna causa y se propaga hacia otra causa de la que no se tiene información. Las causas son independientes entre sí, pero la aparición de una evidencia en el efecto común crea una cierta dependencia entre las causas. Por ejemplo, supongamos el efecto *niebla* (nodo C) que tiene dos causas: *frío* (nodo S) y *humedad* (nodo P). Hay niebla, por lo que existe evidencia (del nodo C). En este momento la probabilidad de ambas causas (P y S) aumenta. También hace frío (evidencia de la causa, del nodo S). En este mismo momento la probabilidad de *humedad* ha disminuido por el hecho de la evidencia de *frío*. Como se ve, aparece una dependencia condicional entre las causas.



Combinado Es una combinación de evidencias. Se posee la evidencia de la causa y el efecto de un nodo y se propaga a dicho nodo. Un ejemplo es la red que Conati propone en [32]. En dicha red se trata de conocer las emociones del estudiante (nodo C) a través de causas como Personalidad (nodo S) y a través de los efectos que se manifiestan en las Expresiones Corporales (nodo D) del estudiante.



C.1.3. Propagación probabilística en redes bayesianas

Como se ha comentado, las redes bayesianas permiten realizar razonamientos en situaciones en las que existe incertidumbre. Para ello representan gráficamente las variables que intervienen en el razonamiento (nodos) y las relaciones entre ellas (arcos).

Una vez construida la red, a partir de la información de determinados nodos (evidencia), necesitamos extraer conclusiones sobre otros nodos (incógnita). Para hacerlo es necesario utilizar un mecanismo de propagación probabilística que permita calcular la probabilidad de los nodos incógnita dados los nodos evidencia. Esta propagación se realiza cada vez que aparece una nueva evidencia y se extiende a todos los nodos de la red. La propagación probabilística se hace utilizando la regla de la cadena.

La regla de la cadena de la teoría de probabilidad nos permite factorizar las probabilidades conjuntas. La probabilidad conjunta de $x_1, x_2, \ldots, x_n$ es igual a la probabilidad de x_1 por la probabilidad de x_2 condicionada a x_1, por la probabilidad de x_3 condicionada a x_2 y x_1, y así sucesivamente.

$$P(x_1, x_2, ..., x_n) = P(x_1) \cdot P(x_2|x_1) \cdot ... \cdot P(x_n|x_1, ..., x_{n-1}) = \prod_i P(x_i|x_1, ..., x_{i-1}) \qquad \text{(C.2)}$$

Desarrollo matemático de la propagación probabilística

En la figura C.1 se ve un ejemplo de red de propagación probabilística. Tiene un nodo central (X), sus padres (P_1 y P_2) y ascendientes (A), y sus hijos (H_1 y H_2) y descendientes (D).

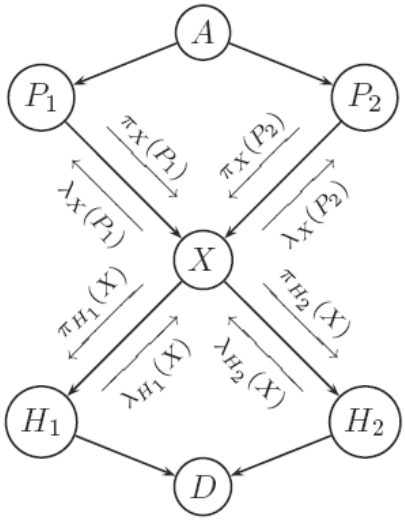


Figura C.1: Red ejemplo de propagación probabilística

Notación y definiciones

Definición 1 *La evidencia* por encima *de X ($e_X^{\uparrow}$) es la evidencia que proporcionan los antepasados de X* $\{e_{P_1}, e_{P_2}, e_A\}$.

Definición 2 *La evidencia* por debajo *de X ($e_X^{\downarrow}$) es la evidencia que proporcionan los descendientes de X* $\{e_{H_1}, e_{H_2}, e_D\}$.

Definición 3 *Hay* separación direccional *en un grafo acíclico conexo, si dado un nodo X, el conjunto de padres, $Pa(X)$, separa condicionalmente este nodo de otro subconjunto Y en el que no haya descendientes de X,*

$$P(x|Pa(X), Y) = P(x|Pa(X)) \tag{C.3}$$

Creencia, π-valores y λ-valores

La información que se desea obtener son las probabilidades a posteriori, también llamadas creencias (*beliefs*), de todos los nodos de la red. Para calcularlas se utilizará la siguiente ecuación

$$P(x|e) = \frac{P(x, e_X^{\uparrow}, e_X^{\downarrow})}{P(e)} = \frac{P(x, e_X^{\uparrow}) \cdot P(e_X^{\downarrow}|x, e_X^{\uparrow})}{P(e)} \tag{C.4}$$

Gracias a la ecuación (C.3) se obtiene $P(e_X^{\downarrow}|x, e_X^{\uparrow}) = P(e_X^{\downarrow}|x)$. La ecuación (C.4) se puede escribir:

$$P(x|e) = \alpha \cdot \pi(x) \cdot \lambda(x)$$

donde α es una constante de normalización tal que $\sum_x P(x|e) = 1$, $\pi(x)$ indica qué valor del nodo X es más probable según la evidencia relacionada con las causas de X (es decir, según la evidencia *por encima* de X) y $\lambda(x)$ indica qué valor del nodo X explica mejor los efectos de X (es decir, la evidencia *por debajo* de X).

$$\pi(x) \equiv P(x, e_X^{\uparrow}) \qquad \lambda(x) \equiv P(e_X^{\downarrow}|x)$$

Al desarrollar las ecuaciones anteriores son de utilidad dos términos adicionales: $\pi_X(u)$, que indica qué valor del nodo P (padre de X) es más probable según la evidencia *por encima* del arco XP; y $\lambda_Y(x)$, que indica qué valor X explica mejor la evidencia *por debajo* del enlace HX (H hijo de X).

$$\pi_X(p_i) \equiv P(p_i, e_{P_iX}^{\uparrow}) \qquad \lambda_H(x) \equiv P(e_{XH}^{\downarrow}|x)$$

Los mensajes π/λ siguen un formato del tipo $\pi_{Hijo}(Padre)$ y $\lambda_{Hijo}(Padre)$. Los mensajes π son intercambiados en el sentido del arco (de padre a hijo) por lo que se usan para predecir

efectos dadas las causas. Los mensajes λ son intercambiados en el sentido inverso del arco (de hijo a padre) por tanto se pueden usar para diagnóstico.

Estas ecuaciones se calculan directamente cuando la red es muy sencilla. En el caso de una red más complicada, es mejor utilizar un algoritmo de propagación de la probabilidad.

C.2. BNSeq: un secuenciador bayesiano para SIT

Todo lo visto anteriormente se refiere a la teoría propia de las redes bayesianas. A continuación, se explica cómo se adaptan para el diseño de un secuenciador en el ámbito de SIT. El objetivo de este secuenciador —que es trabajo en curso en el momento de presentar esta tesis— es la creación de tutores inteligentes a partir del mismo material educativo que se ha empleado en los tutores que se han creado basados en el secuenciador de grafos de secuenciamiento 7.3.

C.2.1. Cómo diseñar la red

En primer lugar hay que decidir lo que se quiere representar con la red bayesiana. Un tutor inteligente debe tener una descripción del dominio del conocimiento y ser capaz de secuenciar un grupo de actividades educativas. El conjunto de éstas, junto con los objetivos de aprendizaje que pretenden cubrir, forman módulos de aprendizaje. De dichas actividades se obtiene información necesaria para evaluar al alumno y al módulo de aprendizaje; en el caso de la red bayesiana, dicha información constituye las *evidencias*. Una vez realizada una actividad y evaluados sus resultados, hay que decidir cuál será la actividad más adecuada para que el estudiante consiga los objetivos del módulo y adquiera así el conocimiento buscado. Para ello se utilizan las evidencias obtenidas de los resultados del estudiante, así como las obtenidas del histórico de las interacciones de otros alumnos con el mismo recurso. Las *actividades* serán evaluadas de una forma predeterminada, es decir, cambiando las probabilidades de los nodos de la red de una forma fija. Por ejemplo, realizar correctamente un ejercicio hace que la probabilidad de poseer el conocimiento al que se refiere dicho ejercicio aumente y la de no poseerlo disminuya. Un ejemplo, en codificación XBN [116], se muestra en la figura C.3.

Los grandes bloques de conocimiento los denominaremos objetivos principales y serán representados por los nodos salida, mientras que los nodos entrada serán aquellos que obtienen sus valores de las actividades realizadas. Los nodos entrada poseerán tres estados o valores: bajo, medio y alto. Cuanta más probabilidad tenga un estado, más probable es que el estudiante se encuentre en ese nivel. Por ejemplo, un estudiante que tenga por probabilidades $P(bajo) = 0{,}7$, $P(medio) = 0{,}2$ y $P(alto) = 0{,}1$ tendrá una mayor probabilidad de no poseer el conocimiento, es decir, de no alcanzar el objetivo de aprendizaje. Si realiza bien una actividad, $P(bajo)$ disminuirá y aumentarán las otras dos. Siempre se ha de cumplir que la suma de las probabilidades de los estados sea uno. Un ejemplo de red bayesiana se muestra en la figura C.2 y corresponde al objetivo didáctico de la parte ”Procesos” de un tutor en el dominio de Sistemas Operativos[1].

[1]Los recursos de que dispone este tutor se han reutilizado del tutor que se presentó en la sección 7.4.2.

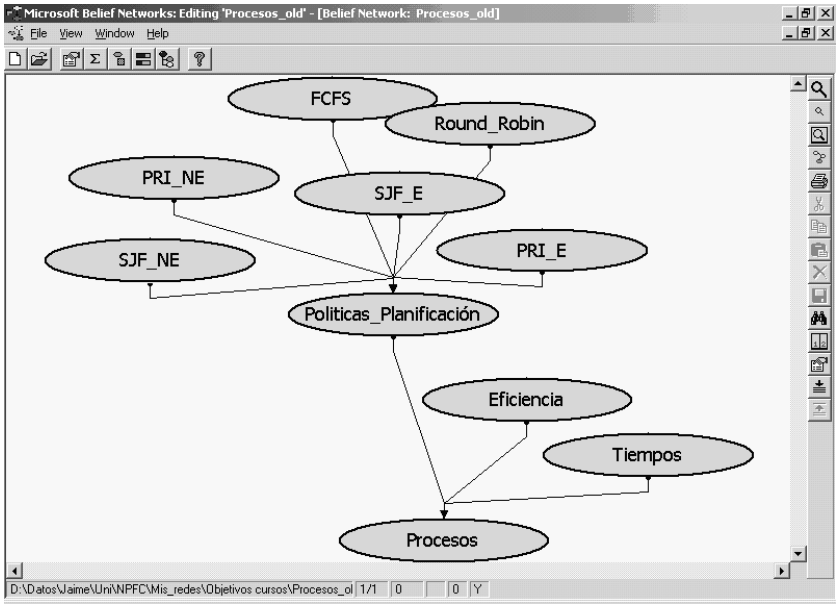


Figura C.2: Objetivo Procesos en BNSeq

```
<activities>
 <goal name="Procesos">
  <exercise>
   <condition nodename="PRIE" state="medio"
              maxlevel="0.6" minlevel="0.2">
   <condition nodename="PRIE" state="alto"
              maxlevel="0.3" minlevel="0.15">
   <url name="a1" url="localhost:8080/actividad1.jsp" />
  </exercise>
  <exercise>
   <condition nodename="SJF_E" state="medio"
              maxlevel="0.6" minlevel="0.2">
   <condition nodename="SJF_NE" state="bajo"
              maxlevel="0.3" minlevel="0.15">
   <url name="a2" url="localhost:8080/actividad2.jsp" />
  </exercise>
 </goal>
</activities>
```

Figura C.3: Ejemplo de actividades en BNSeq

C.2.2. Razonamiento Bayesiano en BNSeq

El secuenciador utiliza un razonamiento predictivo (de las causas se llega a los efectos). Las probabilidades calculadas mediante la inferencia se almacenan en la base de datos. De esta manera, cuando se realiza otra *actividad* y ésta no afecta a algunos nodos entrada, se toman los valores de la base de datos. En caso de que el tutor vaya a ser empleado durante periodos largos de tiempo, puede penalizarse que no se consulten ciertas unidades didácticas en un elevado periodo de tiempo (aumentando $P(bajo)$ y disminuyendo las otras dos).

Con las probabilidades obtenidas, se da la posibilidad al alumno de perfeccionar el objetivo de aprendizaje que esté ejercitando, cambiar a actividades más complicadas o cambiar de objetivo. Estas opciones dependen del nivel del estudiante en el objetivo que está ejercitando y del número de actividades que haya realizado. Esta última cláusula se debe al hecho de que un alumno puede no avanzar con las actividades de un objetivo y, si no se le da la alternativa de cambiar de actividad, dejará de utilizar el tutor por aburrimiento.

En nuestro caso, el nivel del alumno y el número de actividades que debe realizar para que se produzca un cambio en el tipo de actividad u objetivo está prefijado por defecto. El caso ideal se produce cuando hay valores distintos según el tipo de alumno y su motivación. El problema estriba en la dificultad de estimar el tipo de estudiante y su motivación. Discernir la personalidad de una persona es difícil y la motivación presenta más complejidad aún. La posibilidad de preguntarle al propio estudiante sólo introduce ruido en la aplicación, ya que no hay garantía sobre la fiabilidad de sus respuestas al respecto de sus propias emociones.

A pesar de todo, se pueden obtener características del alumno (como su personalidad y su motivación) a partir de su comportamiento con la aplicación. Así, pueden estimarse rasgos de personalidad como la responsabilidad (vg. un estudiante que se conecta a la aplicación regularmente e intenta resolver las actividades correctamente) o la neurosis (vg. un estudiante que cambia constantemente de actividad en un corto periodo de tiempo). La motivación también podría estimarse detectando que un estudiante se conecta regularmente, y cuanto mejores resultados obtiene, más tiempo dedica a la aplicación (más motivado está). Un signo de desmotivación puede ser un estudiante que se conecta muy poco y, tras realizar una actividad mal, se desconecta.

Mediante el uso del razonamiento bayesiano, un ITS no sólo puede hacer el secuenciamiento de actividades, sino que puede obtener información sobre el nivel del estudiante en cada parte del curso y adaptar éste a las necesidades de aquél. Al proporcionar diversas opciones al alumno, se le da libertad para que decida cuales han de ser sus siguientes pasos con la aplicación. Si el alumno realiza las mismas actividades una y otra vez, llegará un momento en que no aumentará su conocimiento y por tanto esas actividades no se le mostrarán.

C.3. Implementación del secuenciador

Para poder evaluar los conocimientos del alumno se utiliza una red bayesiana, y los resultados de dicha red se guardan en una tabla de la base de datos. Esta tabla se llama ”bn_objectives” y posee los siguientes campos:

login identificador del usuario.

type tipo de objetivo (0 primario, 1 secundario).

primary objetivo primario de la red; coincide con el nombre de la red.

name nombre del nodo objetivo.

state estado del nodo objetivo (bajo, medio, alto).

level probabilidad del estado del nodo objetivo.

De esta manera, al evaluar un objetivo primario, se recupera la información de la tabla, se actualiza la información con los resultados de las actividades, se evalúa la red de nuevo y se guarda la información en la tabla de probabilidades de cada estado de cada nodo. Según los valores de los estados de los nodos se podrán realizar unas actividades u otras.

Las actividades se clasifican por objetivos principales. Si se cumplen un cierto número de condiciones, la actividad estará disponible para el alumno.

Interfaz Sequencer

A continuación se describen con más detalle las ideas generales presentadas. Para ello, se analiza el comportamiento de cada uno de los métodos del interfaz Sequencer (sección 6.3.6) y como se implementan en el secuenciador BNSeq. Este secuenciador está siendo desarrollado para SIT version 3.

Init / close

No hay ninguna labor especial que llevar a cabo en estos métodos. Las redes bayesianas son diferentes para cada módulo de aprendizaje, y hay una dependencia adicional con los objetivos de cada actividad. Por ello, las redes se cargan de forma dinámica en el método “process”.

El método close debería encargarse de liberar la memoria ocupada por las redes bayesianas. Sin embargo, la implementación actual está hecha en Java y esa labor la lleva a cabo el propio recolector de basura.

Process

Al igual que en el caso del secuenciador basado en grafos de secuenciamiento, este método concentra la mayor parte de la funcionalidad. Devuelve las actividades que el usuario puede realizar en la siguiente iteración. El usuario es quien debe escoger la siguiente actividad de entre este conjunto.

Las actividades mostradas al alumno están etiquetadas y se diferencian según el objetivo didáctico al que pertenecen y el nivel de conocimiento de éste. El nivel del alumno se calcula a partir de la red bayesiana del objetivo y los valores obtenidos se guardan en la base de datos.

La información necesaria para poder evaluar la red se obtiene de las propias actividades. La evaluación de la red permite obtener los valores de los estados para cada nodo. Esta información se guarda después en la base de datos.

En el caso de que el alumno esté accediendo por primera vez a un módulo de aprendizaje, se cargarán aquellas redes que sean pertinentes en función de los objetivos de aprendizaje que tenga asignados. Cada objetivo tiene su propia red. El análisis de las redes bayesianas permite obtener unos valores de idoneidad para las actividades; aquellas con los valores más altos son devueltas a la unidad administrativa para que se las muestre al usuario.

Las actividades podrían llegar a repetirse si no se ha superado el nivel adecuado del correspondiente objetivo. La repetición tediosa de actividades se evita empleando otra red bayesiana que estima el nivel de aburrimiento del estudiante: una actividad no se entrega al usuario si la probabilidad de que produzca hastío es alta (por ejemplo, porque ya la ha realizado varias veces seguidas y sus resultados no mejoran). Adicionalmente, este comportamiento se evita en SIT versión 3 puesto que el usuario siempre tiene la posibilidad de escoger otra actividad entre las ofrecidas.

Update

Cuando el usuario escoge una actividad, sus niveles de conocimiento para cada objetivo involucrado en la misma son recalculados. Esta información se guarda en la base de datos. Es importante notar que el número de redes que está involucrado en el proceso puede variar en función de la actividad concreta: algunas actividades sólo tienen relación con un objetivo de aprendizaje (una red) mientras que otras son relevantes para más de uno al mismo tiempo.

Cuando los niveles de conocimiento pertinentes han sido actualizados, la URL de la correspondiente actividad es entregada a la unidad administrativa; la unidad administrativa debe reenviarla al usuario. Si el secuenciador se modificara para poder operar con actividades desacopladas en SIT version 4, este método tendría que enviar un bit adicional (ver página 68) indicando si la actividad escogida es una actividad orientada a un recurso atómico o si, por el contrario, requiere la interacción con una aplicación externa (vg. SIETTE [33]).

D Parametric Exercises

Un ejercicio paramétrico es un ejercicio que puede ser replicado un número ilimitado de veces con datos diferentes [23] a través de una secuencia de tres pasos: instanciación, procesado de la respuesta del alumno y realimentación.

A parametric exercise (henceforth PE) is an exercise that can be replicated an unlimited number of times with different data [23] through a sequence of three steps: instantiation, response processing and feedback.

In the first step, the exercise is produced based on a set of given parameters and posed to the student. The second step processes the given response and finally some feedback is returned to the student.

The parameters of an exercise have a role similar to function parameters. They may be assigned one value out of a previously defined set or may have default values. For each possible parameter combination, a different version of the exercise is created and submitted to the student. If the exercise requires a response, once received, specific feedback is sent to the student.

The main advantage of these type of exercises is that a large number of different instances varying in complexity can be obtained from a common concept. We believe that this concept efficiently captures the process for creating material followed by an author. The objective of an exercise is to help understand various underlying concepts. It is very common to think of multiple ways to translate this help into different instances of the same problem varying in difficulty, length, focus, etc.

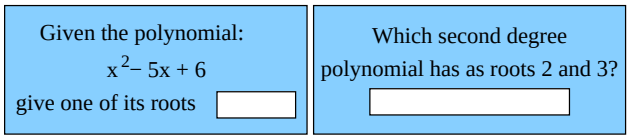
Parametric exercises formalize and capture this diversity and offer the possibility of intuitively producing a variety of exercises about a concrete topic suitable to be used in different learning scenarios. In our experience, this inherent variety in the exercise creation process is independent of the discipline being considered.

The goal is to capitalize this effect and take advantage of it within a tutoring system to increase its effectiveness. Solving several instances of an exercise can be seen as a technique to improve

student performance and comprehension. Also, the different levels of complexity that may derive from the parametric behavior are suitable to achieve content adaptation based on answers obtained from previous exercises.

Parametric exercises are also suitable to be used within collaborative learning environments. Students in a group each receive an instantiation of the same exercise but with different parameters. By working together and exchanging their views on the problem, they are in fact solving the *generic problem* behind each instance and therefore, might gain a more solid understanding of the given concept. Also, having each of them a different instance of the exercise, can be seen as a technique to promote individual contribution to the group discussion.

An example of a parametric exercise could be one that asks for the roots of a second degree polynomial. The exercise may have as parameters a_1 and a_2, the values of the roots used to obtain the polynomial. If no value is given, a random integer is chosen from the range $[-5, 5]$. A third parameter is included to choose between two different versions. The first one shows the polynomial to the student and asks for one of the roots, whereas the second gives the roots and asks the student to write for the polynomial. Figure D.1 shows the two possible versions of the same exercise.



a) First version b) Second version

Figure D.1: Parametric exercises on polynomials

In this example, a different level of difficulty could be assigned to these instances and therefore be used depending on the observation of previous answers given by the student. Also, the given parameters together with the answers of the students allows for personalized feedback generation.

Bibliography / Referencias

[1] Advanced Distributed Learning. *SCORM 2004 Sharable Content Object Reference Model*. ADL, 2004.

[2] Advanced Distributed Learning. SCORM conformance test suite (http://www.adlnet.gov/scorm/20043ed/cts.cfm, accessed feb 2007), 2004.

[3] J. Albert, S. Frank, U. Hafner, and M. Unger. Video compression with weighted finite automata. In *Data Compression Conference*, 1997.

[4] C. Alexander. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, 1977.

[5] C. Alexander. *The Timeless Way of Building*. Oxford University Press, 1979.

[6] I. Androutsopoulos, J. Calder, E. Not, F. Pianiesi, and M. Roussou. Multilingual personalised information objects. In *Proceedings of the International Workshop on Information Presentation and Natural Multimodal Dialogue*, 2001.

[7] P. Avgeriou, A. Papasalouros, S. Retalis, and M. Skordalakis. Towards a pattern language for learning management systems. *Educational Technology & Society*, 6, 2003.

[8] P. Avgeriou, D. Vogiatzis, A. Tzanavari, and S. Retalis. Design patterns in adaptive web-based educational systems: An overview. *Advanced Technology for Learning*, 2004.

[9] A. Barr, M. Beard, and R. Atkinson. The computer as tutorial laboratory: the Stanford BIP project. *International Journal of Man-Machine Studies*, 8:567–596, 1973.

[10] F. Bause and P. S. Kritzinger. *Stochastic Petri Nets*. Vieweg Verlag, 2002.

[11] J. Beck, M. Stern, and E. Haugsjaa. Applications of AI in education. *ACM Crossroads*, 3, 2004.

[12] G. Beni and J. Wang. Swarm intelligence. In *Proceedings of the Seventh Anual Meeting of the Robotics Society of Japan*, pages 425–428. RSJ Press, 1989.

[13] B. S. Bloom. *Taxonomy of Educational Objectives*. Longman, 1956.

[14] B. S. Bloom. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Research*, 13:3–15, 1984.

[15] H. Boley. Racofi: A rule-applying collaborative filtering system. In *Proceedings of 2003 IEEE/WIC International Conference on Web Intelligence / Intelligent Agent Technology*, 2003.

[16] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Inc., New York, NY, USA, 1999.

[17] P. D. Bra, A. Aerts, B. Berden, B. D. Lange, B. Rousseau, T. Santic, D. Smits, , and N. Stash. AHA! The adaptive hypermedia architecture. In *Proceedings of the ACM Hypertext Conference*, 2003.

[18] P. D. Bra, G. J. Houben, and H. Wu. AHAM: A dexter-based reference model for adaptive hypermedia. In *Proceedings of the ACM conference on Hypertext and Hypermedia*, 1999.

[19] C. Brooks, C. Hansen, and J. Greer. Social awareness in the iHelp courses learning content management system. In *Workshop on the Social Navigation and Community-Based Adaptation Technologies (in AH'06)*, 2006.

[20] P. Brusilovsky. Methods and techniques for adaptive hypermedia. *User Modelling and User Adapted Interaction*, 6:87–129, 1996.

[21] P. Brusilovsky. Adaptive educational systems on the www: A review of available technologies. In *Proceedings of the Workshop "WWW-Based Tutoring", at 4th Intelligent Tutoring Systems conference*, Lecture Notes on Computer Science. Springer-Verlag, 1998.

[22] P. Brusilovsky. Course sequencing for static courses? applying its techniques in large-scale web-based education. In *Intelligent Tutoring Systems*, volume 1839 of *Lecture Notes on Computer Science*. Springer-Verlag, 2000.

[23] P. Brusilovsky and P. Miller. Course delivery systems for the virtual university. In F. T. Tschang and T. D. Santa, editors, *Access to Knowledge: New Information Technologies and the Emergence of the Virtual University*, pages 167–206. Elsevier Science, 2001.

[24] G. Buzsák. Memory consolidation during sleep: A neurophysiological perspective. *Journal of Sleep Research*, 7, 1998.

[25] P. Capell and R. Dannenberg. Instructional design and intelligent tutoring: Theory and the precision of design. *Journal of Artifical Intelligence in Education*, 4:95–121, 1993.

[26] R. M. Carro, E. Pulido, and P. Rodriguez. TANGOW: Task-based adaptive learner guidance on the WWW. In *2nd Workshop on Adaptive Systems and User Modeling on the WWW*, 1999.

[27] R. M. Carro, E. Pulido, and P. Rodriguez. Improving web-site maintenance with TANGOW by making page structure and contents independent. In *Web Engineering*, 2001.

[28] S. Ceri, P. Dolog, M. Matera, and W. Nejdl. Model-driven design of web applications with client-side adaptation. In *Proceedings of ICWE 2004*. LNCS 3140, Springer Verlag, 2004.

[29] E. Charniak. Bayesian networks without tears: making bayesian networks more accessible to the probabilistically unsophisticated. *AI Magazine*, 12, 1991.

[30] A. Cini and J. V. de Lima. Adaptivity conditions evaluation for the user of hypermedia presentation built with AHA! In *Adaptive Hypermedia and Adaptive Web-based systems*, 2002.

[31] R. Clark and R. E. Mayer. *e-Learning and the Science of Instruction: Proven Guidelines for Consumers and Designers of Multimedia Learning*. Pfeiffer, 2002.

[32] C. Conati, A. Gertner, and K. Vanlehn. Using bayesian networks to manage uncertainty in student modeling. *User Modeling and User-Adapted Interaction*, 12, 2002.

[33] R. Conejo, E. Guzmán, E. Millán, M. Trella, J. L. P. de-la Cruz, and A. Ríos. SIETTE: Web-based tool for adaptive testing. *International Journal of Artificial Intelligence in Education*, 2004.

[34] D. Costa, A. Hertz, and O. Dubious. Embedding of a sequential algorithm within an evolutionary algorithm for coloring problems in graphs. *Journal of Heuristics*, 1, 1995.

[35] A. Cristea. Authoring of adaptive and adaptable educational hypermedia: Where are we now and where are we going? In *IASTED International Conference in Web-Based Education*, February 2004.

[36] C. Daffara and J. González-Barahona. Free software/open source: Information society opportunities for europe? In *Information Technologies Society conference*, November 1999.

[37] P. de Bra and L. Calvi. AHA! an open adaptive hypermedia architecture. *The New Review of Hypermedia and Multimedia*, 4:115–139, 1998.

[38] M. C. F. de Oliveira, M. A. S. Turine, and P. C. Masiero. A statechart-based model for hypermedia applications. *ACM Transactions on Information Systems*, 19, January 2001.

[39] M. J. C. del Corral, M. Sánchez-Ramos, E. Moro-Rodríguez, and R. Cárdenes-Medina. Software libre y código abierto en aplicaciones para patología. *Revista espańola de patologia*, 2003.

[40] M. Deshpande and G. Karypis. Selective markov models for predicting web page accesses. *ACM Transactions on Internet Technology (TOIT)*, 4, 2004.

[41] V. Devedzic. Applying patterns to ITS architectures. In *Intelligent Tutoring Systems*. Springer-Verlag, 2000.

[42] P. Dodds. *The Scorm Content Aggregation Model*. ADL, 2001.

[43] P. Dolog and W. Nejdl. Using UML and XMI for generating adaptive navigation sequences in web-based systems. In *Proceedings of UML 2003*, 2003.

[44] M. Dorigo and G. di Caro. The ant colony optimization meta-heuristic. In *New ideas in optimization*. McGraw-Hill, 1999.

[45] M. Dorigo and G. di Caro. Colony optimization: A new meta-heuristic. In *Proceedings of the 1999 Congress on Evolutionary Computation*. IEEE Press, 1999.

[46] M. Dorigo and L. M. Gambardella. Ant colonies for the traveling salesman problem. *Biosystems*, 43, 1997.

[47] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, 2004.

[48] C. T. dos Santos and F. S. Osorio. Integrating intelligent agents, user models, and automatic content categorization in a virtual environment. In *Intelligent Tutoring Systems*. Springer-Verlag, 2004.

[49] J. Dron. *Achieving self-organisation in network-based learning environments*. PhD thesis, Brighton University, January 2002.

[50] J. Dron, C. Boyne, and R. Mitchell. Footpaths in the stuff swamp. In *Proceedings of WebNet 2001*, 2001.

[51] J. Dron, R. Mitchell, P. Siviter, and C. Boyne. CoFIND-an experiment in n-dimensional collaborative filtering. In *Proceedings of WebNet 1999*, 1999.

[52] J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B, 1967.

[53] D. Edwards and L. Hardman. Lost in cyberspace: Cognitive mapping and navigation in a hypertext environment. *Hypertext: Theory into practice (R. McAleese ed.)*, 1989.

[54] E. El-Sheikh and J. Sticklen. Generating intelligent tutoring systems from reusable components and knowledge-based systems. In *Intelligent Tutoring Systems*, volume 2363 of *Lecture Notes in Computer Science*. Springer, 2002.

[55] A. E. Elo. *The rating of chessplayers, past and present*. Arco Pub, 1978.

[56] M. Engvig. *Best Practices in eLearning Case Study*. Themo Publishing, 2002.

[57] S. Ferrandino, A. Negro, and V. Scarano. Cheops: Adaptive hypermedia on the world wide web. In *European Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services*, 1998.

[58] G. Fischer, T. Mastaglio, B. Reeves, and J. Rieman. Minimalist explanations in knowledge-based systems. In *Proceeding of 23rd Hawai International Conference on Knowledge-Based Systems*, 1990.

[59] S. Floyd and V. Jacobson. On traffic phase effects in packet-switched gateways. *Internetworking: Research and Experience*, 3, 1992.

[60] S. S. Frizell and R. Hubscher. Supporting the application of design patterns in web-course design. In *Proceedings of ED-MEDIA 2002*, 2002.

[61] L. M. Gambardella, E. Taillard, and G. Agazzi. MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. *New Ideas in Optimization*, 1999.

[62] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[63] F. Garzotto, S. Retalis, A. Papasalouros, and K. Siassiakos. Towards patterns for designing adaptive/adaptable educational hypermedia. *Advanced Technology for Learning*, 2006.

[64] A. Girault, B. Lee, and E. A. Lee. Hierarchical finite state machines with multiple concurrency models. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1999.

[65] U. Gneezy and M. Das. Experimental investigation of perceived risk in finite random walk processes. Discussion paper, Tilburg University, Center for Economic Research, 1996. available at http://ideas.repec.org/p/dgr/kubcen/199685.html.

[66] A. A. Gokhale. Collaborative learning enhances critical thinking. *Journal of Technology Education*, 7, 1995.

[67] H. Goldstine and A. Goldstine. The electronic numerical integrator and computer. *Mathematical Tables and other Aids to Computation*, 2(2-3):97–110, 1946.

[68] S. J. Gould. *Ever Since Darwin - Reflections in Natural History*. Burnett, 1987.

[69] S. Gutiérrez, A. Pardo, and C. D. Kloos. Beyond simple sequencing: Sequencing of learning activities using hierarchical graphs. In *Web-Based Education 2004*, February 2004.

[70] S. Gutiérrez, A. Pardo, and C. D. Kloos. Finding a learning path: Toward a swarm intelligence approach. In *Web Based Education*. ACTA Press, 2006.

[71] E. Guzmán and R. Conejo. SIETTE: System of intelligent evaluation using tests. Technical report, 2005.

[72] D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3), 1987.

[73] J. Hartley and D. Sleeman. Towards more intelligent teaching systems. *International Journal of Man-Machine Studies*, 2:215–336, 1973.

[74] N. Heffernan, , K. Koedinger, and V. A. Aleven. Tools towards reducing the costs of designing, building, and testing cognitive models. In *Conference on Behavior Representation in Modeling and Simulation (BRIMS 2003)*, 2003.

[75] F. Heylighen. The science of self-organization and adaptivity. *The Encyclopedia of Life Support Systems*, 2003.

[76] F. Heylighen and C. Gershenson. The meaning of self-organization in computing. *IEEE Intelligent System*, 2003.

[77] T. Hoffman. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *Proceedings of the 26th ACM SIGIR Conference on Research in Information Retrieval*, 2003.

[78] IEEE Computer Society/Learning Technology Standards Committee. IEEE Standard for Learning Object Metadata. Final Specification v.1.0, 2002.

[79] IMS Global Consortium. IMS Simple Sequencing best practice and implementation guide. Final specification, March 2003.

[80] IMS Global Consortium. IMS Simple Sequencing information and behavior model. Final specification, March 2003.

[81] IMS Global Learning Consortium. IMS Learning Design. Final Specification v.1.0, January 2003.

[82] IMS Global Learning Consortium. IMS Learning Design Best Practice and Implementation Guide. Final Specification v.1.0, January 2003.

[83] IMS Global Learning Consortium. IMS Learning Design Information Model. Final Specification v.1.0, January 2003.

[84] IMS Global Learning Consortium. IMS Simple Sequencing. Final Specification v.1.0, March 2003.

[85] IMS Global Learning Consortium. IMS Content Package. Final Specification v.1.2, November 2005.

[86] IMS Global Learning Consortium. IMS Learner Information profile. Technical Report v.1.0.1, January 2005.

[87] IMS Global Learning Consortium. IMS Question and Test Interoperability. Public Draft v.2.1, January 2006.

[88] A. Inaba, R. Ohkubo, M. Ikeda, R. Mizoguchi, and J. Toyoda. An instructional design support environment for CSCL: Fundamental concepts and design patterns. In *Proceedings of AIED01*, 2001.

[89] A. Jameson. Systems that adapt to their users: An integrative overview. In *International Conference on User Modelling*, 2003.

[90] JISC. The RELOAD project (http://www.reload.ac.uk, accessed feb 2007).

[91] S. Kauffman. *At Home in the Universe: The Search for the Laws of Self-Organization and Complexity*. Oxford University Press, 1996.

[92] J. Kay. Scrutable adaptation: Because we can and must. In *Adaptive Hypermedia*, 2006.

[93] J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Morgan Kauffman, 2001.

[94] K. Kettunen. *Soft Detection and Decoding in Wideband CDMA Systems (PhD thesis)*. Helsinki University of Technology, 2003.

[95] R. Khuwaja, M. Desmarais, and R. Cheng. Intelligent guide: Combining user knowledge assesment with pedagogical guidance. In *Intelligent Tutoring Systems*, volume 1086 of *Lecture Notes on Computer Science*. Springer-Verlag, 1996.

[96] A. Kobsa. Generic user modeling systems. *User Modeling and User-Adapted Interaction*, 11:49–63, 2001.

[97] R. Koper. Designing learning networks for lifelong learners. In *Koper, R. and Tattersall, C. (eds.), Learning Design: A Handbook on Modelling and Delivering Networked Education and Training*, 2005.

[98] K. B. Korb and A. E. Nicholson. *Bayesian Artificial Intelligence*. Chapman & Hall/CRC, 2004.

[99] G. Lakoff. *Women, Fire, and Dangerous Things*. University of Chicago Press, 1987.

[100] E. Lawler. *Combinatorial optimization: networks and matroids*. Saunders College Publishing, 1976.

[101] M. V. Lawson. *Finite Automata*. Chapman and Hall, 2003.

[102] F. Lazarinis and S. Retalis. Engineering an interoperable adaptive hypermedia testing tool supporting user adaptable strategies. In *3rd IFIP Conference on Artificial Intelligence Applications and Innovations*, 2006.

[103] T. Leinonen. Intentional learning: reflecting the discussion in the blogsphere. In *FLOSSE Posse (http://flosse.dicole.org/?item=intentional-learning-reflecting-the-discussion-in-the-blogsphere, accessed Feb 2007)*, 2005.

[104] L. Lessig. The creative commons. *Florida Law Review*, 55, 1994.

[105] L. Lessig. *Free Culture: The Nature and Future of Creativity*. Penguin, 2004.

[106] T. Lin, Kinshuk, and A. Patel. Configurable, incremental and re-structurable contributive learning environments (CIRCLE). In *Proceedings of the IASTED International Conference on Computers and Advanced Technology in Education*, 2002.

[107] T. Lin, Kinshuk, and A. Patel. An authoring tool for variable relationship capturing: VR-Capture. In *Proceedings of EdMedia 2002*, 2005.

[108] P. P. Linillos. *Diseño de un tutor inteligente basado en grafos jerárquicos y aplicado a sistemas operativos (PFC)*. Universidad Carlos III de Madrid, 2005.

[109] P. G. Love and V. L. Guthrie. *Understanding and Applying Cognitive Development Theory*. Jossey-Bass, 2003.

[110] J. Ludwig, S. Ramachandran, and W. Howse. Developing an adaptive intelligent flight trainer. In *Proceedings of the Industry/Interservice, Training, Simulation and Education Conference (I/ITSEC 2002)*, 2002.

[111] N. Major, S. Ainsworth, and D. Wood. Redeem: Exploiting symbiosis between psychology authoring environments. *International Journal of Artificial Intelligence in Education*, 1997.

[112] E. Marek and A. M. Cavallo. *The Learning Cycle*. Heinemann, 1997.

[113] I. Martínez-Ortiz, P. Moreno-Ger, J. L. Sierra, and B. Fernández-Manjón. A general architecture for the authoring and the operationalization of e-learning applications with educational modelling languages. In *Proc. 8º Simposio Internacional de Informática Educativa*, 2006.

[114] M. Mauny and G. Cousineau. *The Functional Approach to Programming*. Cambridge University Press, 2003.

[115] D. McArthur, C. Stasz, J. Hotta, O. Peter, and C. Burdof. Skill-oriented task sequencing in an intelligent tutor for basic algebra. *Instructional Science*, 17:281–307, 1988.

[116] Microsoft. Decision Theory & Adaptive Systems Group, XML belief network (retrieved from http://research.microsoft.com/dtas/bnformat/default.htm, accessed feb 2007).

[117] E. Millán, E. García-Hervás, E. Guzmán, A. Rueda, and J. L. P. de-la Cruz. TAPLI: An adaptive web-based learning environment for linear programming. In *Current Topics in Artificial Intelligence*, 2004.

[118] T. Mitrovic and P. Suraweera. Evaluating an animated pedagogical agent. In *Intelligent Tutoring Systems*. Springer-Verlag, 2000.

[119] K. Miyahara and M. Pazzani. Collaborative filtering with the simple bayesian classifier. In *Pacific Rim International Conference on Artificial Intelligence*, 2000.

[120] B. Mobasher. Web usage mining and personalization. In *M. P. Singh (Ed.), Practical Handbook of Internet Computing*, 2004.

[121] R. Morley. Painting trucks at general motors: The effectiveness of a complexity-based approach. *Embracing Complexity: Exploring the Application of Complex Adaptive Systems to Business*, 1996.

[122] T. Murray. Having it all, maybe: Design tradeoffs in its authoring tools. In *Proceedings of the Third International Conference on Intelligent Tutoring Systems*, 1996.

[123] T. Murray. Authoring intelligent tutoring systems: An analysis of the state of the art. *Int. J. of Artificial Intelligence in Education*, 10:98–129, 1999.

[124] T. Murray and B. Woolf. Results of encoding knowledge with tutor construction tools. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, 1992.

[125] M. Nesca and D. Koulack. Recognition memory, sleep and circadian rhythms. *Canadian Journal of Experimental Psychology*, September 1996.

[126] G. Neumann and J. Zirvas. SKILL: A scalable internet-based teaching and learning system. In *Proceedings of WebNet 98 World Conference of the WWW*, 1998.

[127] A. M. O'Donnell, C. E. Hmelo-Silver, and G. E. (Eds.). *Collaborative Learning, Reasoning, and Technology*. LAWRENCE ERLBAUM Associates, Inc., 2006.

[128] A. Ovejero Bernal, M. de la Villa Moral Jiménez, and J. Pastor Martín. Aprendizaje cooperativo: un eficaz instrumento de trabajo en las escuelas multiculturales y multiétnicas del siglo XXI. *Revista Electrónica Iberoamericana de Psicología Social*, 1, 2002.

[129] L. Pagliaro. The history and development of CAI: 1926-1981, an overview. *The Alberta Journal of Educational Research*, 29:75–84, 1983.

[130] M. C. F. Panadero. *EPM: Un Modelo para la Caracterización y Diagnóstico de Procesos Educativos*. Universidad Carlos III of Madrid, 2004.

[131] A. Papasalouros, S. Retalis, and N. Papaspyrou. Semantic description of educational adaptive hypermedia based on a conceptual model. *Educational Technology and Society*, 1, 2004.

[132] S. Papert. A critique of technocentrism in thinking about the school of the future. In *Children in an Information Age: Opportunities for Creativity, Innovation and New Activities*, 1987.

[133] P. Paredes and P. Rodríguez. A mixed approach to modelling learning styles in adaptive educational hypermedia. In *IASTED International Conference in Web-Based Education*, February 2004.

[134] D. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles. Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI 2000)*, 2000.

[135] J. L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, 1981.

[136] S. Petrina. Sidney Pressey and the automation of education, 1924-1934. *Technology and Culture*, 45:305–330, 2004.

[137] S. L. Pressey. A simple apparatus which gives tests and scores - and teaches. *School and Society*, 23:373–376, 1926.

[138] S. L. Pressey. A machine for automatic teaching of drill material. *School and Society*, 25:549–552, 1927.

[139] G. H. Recio. *Desarrollo de una herramienta gráfica para el desarrollo de grafos jerárquicos: SGed (PFC)*. Universidad Carlos III de Madrid, 2006.

[140] S. Retalis. *CADMOS: web-based Courseware Development Methodology for Open Systems based on software engineering practices (PhD thesis)*. National Technical University of Athens, 1998.

[141] C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *Computer Graphics*, 21, 1987.

[142] A. Rios, J. L. P. de la Cruz, and R. Conejo. SIETTE: Intelligent evaluation system using tests for basic algebra. In *Proceedings of workshop WWW-Based Tutoring at 4th Intelligent Tutoring Systems Conference*, 1993.

[143] C. Romero, S. Ventura, P. de Bra, and C. de Castro. Discovering prediction rules in AHA! courses. In *Proceedings of the User Modelling conference*, 2003.

[144] G. Rossi, D. Schwabe, C. Lucena, and D. Cowan. An object-oriented model for designing the human-computer interface of hypermedia application. In *Proceedings of the International Workshop on Hypermedia Design (IWHD'95)*, 1995.

[145] J. J. G. Rueda. *Modelado y Diseño de Experiencias Educativas en la World Wide Web (PhD thesis)*. Universidad Politécnica de Madrid, 2002.

[146] B. Rusell. Review of McColl. *Mind*, 1906.

[147] B. Schwartz. *The Paradox of Choice: Why More Is Less*. Ecco, 2004.

[148] Y. Semet, E. Lutton, and P. Collet. Ant colony optimisation for e-learning: Observing the emergence of pedagogical suggestions. In *IEEE Swarm Intelligence Symposium*, 2003.

[149] Y. Semet, Y. Yamont, R. Biojout, E. Luton, and P. Collet. Artificial ant colonies and e-learning: An optimisation of pedagogical paths. In *10th International Conference on Human-Computer Interaction*, 2003.

[150] C. Shirky. In praise of evolvable systems. *ACM Net_Worker*, 1996.

[151] A. P. Silva. Hypermedia, influence of interactive freedom degree in learning processes. In *Hypermedia courseware, structures of communication and intelligent help*. Springer-Verlag, 1987.

[152] D. Sleeman and J. Brown. *Intelligent Tutoring Systems*. Academic Press, 1982.

[153] K. Socha, J. Knowles, and M. Samples. A MAX-MIN ant system for the university timetabling problem. In *Third International Workshop on Ant Algorithms (ANTS 2002)*, 2002.

[154] M. Specht and D. Burgos. Implementing adaptive educational with IMS learning design. In *ADALE Workshop on Adaptive Learning and Learning Design*, June 2006.

[155] N. Stash and P. D. Bra. AHA! A general-purpose tool for adaptive websites. In *World Wide Web Conference*, 2002.

[156] N. Stern. *From ENIAC to UNIVAC, An Appraisal of the Eckert-Mauchly Computers*. Digital Press, 1981.

[157] C. Tattersall, J. Manderveld, B. Berg, R. Es, J. Janssen, and R. Koper. Self organising wayfinding support for lifelong learners. *Education and Information Technologies*, 2005.

[158] C. Tattersall, J. Manderveld, B. van den Berg, R. V. Es, J. Janssen, and R. Koper. Swarm-based wayfinding support in open and distance learning. In *In Alkhalifa, E.M. (Ed). Cognitively Informed Systems: Utilizing Practical Approaches to Enrich Information Presentation and Transfer*, 2005.

[159] M. Trella, R. Conejo, E. Guzmán, and D. Bueno. An autonomous component architecture to develop WWW-ITS. In *Proceedings of the 3rd International Conference on Web Engineering*, 2003.

[160] G. Valigiani. *Développement d'un paradigme d'Optimisation par Hommilière et application a l'Enseignement Assisté par Ordinateur sur Internet (PhD thesis)*. Université du Litoral Côte d'Opale, 2006.

[161] G. Valigiani, R. Biojout, Y. Jamont, E. Lutton, C. B. Republique, and P. Collet. Experimenting with a real-size man-hill to optimize pedagogical paths. In *SAC '05: Proceedings of the 2005 ACM Symposium on Applied computing*, pages 4–8, New York, NY, USA, 2005. ACM Press.

[162] G. Valigiani, E. Lutton, and P. Collet. Adapting the ELO rating system to competing subpopulations in a "man-hill". In *ISPE CE*, pages 766–773, 2006.

[163] G. Valigiani, E. Lutton, Y. Yamont, R. Biojout, and P. Collet. Automatic rating process to audit a man-hill. *WSEAS Transactions on Advances in Engineering Education*, 3:1–7, January 2006.

[164] B. van Berkel and K. D. Smedt. Triphone analysis: a combined method for the correction of orthographical and typographical errors. In *Proceedings of the second conference on Applied natural language processing*, pages 77–83, Morristown, NJ, USA, 1988. Association for Computational Linguistics.

[165] J. van Bruggen, P. Sloep, P. van Rosmalen, F. Brouns, H. Vogten, R. Koper, and C. Tattersall. Latent semantic analysis as a tool for learner positioning in learning networks for lifelong learning. *British Journal of Educational Technology*, 2004.

[166] J. M. D. Vivancos. *Traducción de grafos de secuenciamiento SG a la especificación IMS-LD (PFC)*. Universidad Carlos III de Madrid, 2006.

[167] W3C. Cascading style sheets, level 1 (http://www.w3.org/tr/rec-css1, accessed sep 2006).

[168] W3C. Cascading style sheets, level 2 (http://www.w3.org/tr/rec-css2, accessed sep 2006).

[169] W3C. Namespaces in XML (http://www.w3.org/tr/rec-xml-names, accessed feb 2007).

[170] W3C. XHTML 1.0 the extensible hypertext markup language (http://www.w3.org/tr/xhtml1/, accessed dec 2006).

[171] U. Wagner, S. Gais, and J. Born. Emotional memory formation is enhanced across sleep intervals with high amounts of rapid eye movement sleep. *Learning & Memory*, 8, 2001.

[172] L. Wan, C. Zhao, and Q. Luo. Navigation and sequencing strategy of learning process in distance learning context. In *Frontiers in Education*, 2006.

[173] Wikipedia. http://en.wikipedia.org/wiki/.lrn (accessed feb 2007).

[174] Wikipedia. http://en.wikipedia.org/wiki/moodle (accessed feb 2007).

[175] Wikipedia. http://es.wikipedia.org/wiki/m%c3%basica_libre and http://en.wikipedia.org/wiki/jamendo (accessed feb 2007).

[176] D. A. Wiley. Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. In *The Instructional Use of Learning Objects*. Agency for Instructional Technology, 2002.

[177] D. A. Wiley and E. K. Edwards. Online self-organizing social systems: The decentralized future of online learning. *Quarterly Review of Distance Education*, 3:33–46, 2002.

[178] D. E. Wiley. RIP-ping on learning objects. In *Iterating toward openness (http://opencontent.org/blog/archives/230, accessed Feb 2007)*, 2006.

[179] M. Wolfe, M. Schreiner, B. Rehder, D. Laham, P. Foltz, W. Kintsch, and T. Landauer. Learning from text: Matching readers and texts by latent semantic analysis. *Discourse Processes*, 25, 1998.

[180] P. J. Woods and J. R. Warren. Rapid prototyping of an intelligent tutorial system. In *Proceedings of ASCILITE95, Australasian Society for Computers in Learning in Tertiary Education conference*, 1985.

[181] B. P. Woolf and P. Cunningham. Building a community memory of intelligent tutoring systems. In *National Conference on Artificial Intelligence (AAAI)*, pages 82–89, 1987.

[182] A. Yang, Kinshuk, and A. Patel. A plug-able web-based intelligent tutoring system. In *Proceedings of the Xth European Conference on Information Systems*, 2002.

[183] M. Zlochin and M. Dorigo. Model-based search for combinatorial optimization: A comparative study. In *7th International Conference on Parallel Problem Solving from Nature*, 2002.

Index / Índice alfabético

I once wanted to give a few words in the foreword which now actually are not in it, which, however, I'll write to you now because they might be a key for you: I wanted to write that my work consists of two parts: of the one which is here, and of everything which I have not written. And precisely this second part is the important one.

– Ludwig Wittgenstein, Letter to Ludwig von Ficker, 1919