UNIVERSIDAD DE NAVARRA

ESCUELA SUPERIOR DE INGENIEROS

SAN SEBASTIÁN

Sistemas Interactivos Inteligentes de Ayuda al Aprendizaje de Destrezas Físicas

M E M O R I A

que para optar al Grado de Doctor
presenta

AITOR AGUIRRE ORTUZAR

bajo la dirección de

Alberto Lozano Rodero

Luis Matey Muñoz

San Sebastián, Diciembre 2013

Nire gurasoei

AGRADECIMIENTOS

Tal como dice la célebre frase, "*everything that has a beginning has an end*". Han sido cuatro intensos años donde he conocido a mucha gente y donde muchos me habéis aportado algo para que pudiera llegar a este "*final*". A todos ellos, les quiero agradecer su apoyo.

Doy las gracias a TECNUN, a la Universidad de Navarra y a CEIT por haberme dado la oportunidad de trabajar y llevar a cabo este trabajo. Quiero dar las gracias a Luis Matey por haber depositado su confianza en mí, y además, apoyarme en este largo recorrido. Me quedo con tu claridad de ver fácil lo difícil y con algún palazo que hubo en esos partidos de pala que solíamos jugar.

Quiero agradecer especialmente a Alberto Lozano, por todas las horas que ha pasado conmigo y por todo el apoyo que me ha dado. Sin su ayuda, no hubiera podido llegar a este punto. En definitiva, puedo decir que ha sido la persona que me ha hecho madurar en el mundo de la investigación y mi ejemplo a seguir durante estos años. Su metódica forma de trabajar y su afán por buscar la genericidad en todo lo que toca son únicas. Gracias.

También quiero dar las gracias a Josune y a Diego, por darme la oportunidad de iniciarme en la docencia universitaria y de conocer mejor los entresijos de la universidad. Muchas gracias también a Bego y Mikel, por haber colaborado en mis trabajos y en esos experimentos que tanto trabajo nos han dado.

Como no podía ser de otra manera, doy las gracias a todos mis compañeros de departamento que han pasado por aquí durante estos años. ¡No sé que hubiera hecho sin vosotros! Borja, pasé los primeros años tanto en el CEIT como en Donosti contigo, nunca olvidaré esas largas horas de Intrasim. Gorka, Luis, Alex e Ilaria, mis primeros compañeros de mampara, siempre os recordaré aunque al final yo haya sido el único (a excepción de Alex) en defender este lado del despacho con dignidad. Y no me olvido de Fran, por todo en lo que me ha

ayudado con mi trabajo. En el otro lado de la mampara también he tenido grandes compañeros. Goretti, con quien tantas risas y alegrías he compartido, espero que las sigamos compartiendo. Imanol, espero que esas discusiones tan transcendentales en nuestras vidas nunca se esfumen. Hi Pedro, ze esatek? Ibai y Hugo, los principales verdugos de las rimas mañaneras y de mis chistes. Grandes maestros del paint. Alba, guardaré tus recuerdos en un álbum. Ibon, una gran persona y más tozuda que yo. No me olvido de Dimas, Iñigo, Ainitze, Ainara, Tomak Jaket Lens, Álvaro, Carlos, Aiert, Iker, Iñaki, Alfon, Gaizka, Manolo, Imanol, Maite, Ane, Melo, Jorge, Sergio, Ángel, Yaiza, Jairo, Nerea y del resto de compañeros que he conocido en CEIT.

Gracias a Barrena, Hodei, Lopez, Gurru, Oscar, a mis amigos de Donosti, a mi cuadrilla, a mi cuadrilla en la península y al resto de compañeros de piso que he tenido durante estos años por hacer que estos cuatro años se me hayan pasado en cuatro días. Gracias a Berta, por animarme, comprenderme, alegrarme y sobre todo, aguantarme durante estos años.

Por último, gracias a mi familia, por estar ahí, apoyarme y preocuparse por mi trabajo durante estos años.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

La combinación de Sistemas Interactivos basados en Realidad Virtual con la Inteligencia Artificial educativa ha resultado de gran utilidad en muchos ámbitos de formación. Estos sistemas, también denominados como Sistemas Interactivos Inteligentes de Ayuda al Aprendizaje (SIIAA) suelen ofrecer un amplio abanico de funcionalidades educativas. Sin embargo, es difícil encontrar SIIAA que sean capaces de ofrecer asistencia a la hora de entrenar destrezas físicas.

Este proyecto de tesis presenta un novedoso enfoque para evaluar destrezas físicas en los SIIAA basados en el framework ULISES, el cual permite la integración de Sistemas Interactivos con distintos Sistemas de Ayuda al Aprendizaje. Asimismo, se presenta la herramienta de autor PATH, la cual minimiza el coste de creación de los modelos de conocimiento necesarios para el framework ULISES.

Por un lado, el trabajo desarrollado ha consistido en diseñar y desarrollar los modelos y algoritmos necesarios para que el framework ULISES pueda gestionar la incertidumbre temporal y la intencionalidad asociadas a los movimientos de los alumnos cuando aprenden una nueva destreza física. Asimismo, se presenta la integración de dichos modelos y algoritmos en los niveles y subsistemas del framework ULISES encargados de la observación, interpretación y diagnóstico de movimientos para la evaluación de destrezas motoras.

Por otro lado, este trabajo aporta la metodología de adquisición de conocimiento para crear SIIAA con ULISES. Además, la herramienta de autor implementada facilita la aplicación de la metodología y su aprendizaje.

Se ha creado un SIIAA compuesto de un sistema de captura para evaluar destrezas físicas en el dominio del tenis y se ha realizado un experimento que demuestra la validez de la metodología presentada en este trabajo. En lo referente a PATH, se ha validado en la creación de tareas para tres SIIAA de diferentes dominios y se han realizado

experimentos adicionales que demuestran el valor de la herramienta a la hora de crear tareas para los SIIAA basados en el framework ULISES.

ABSTRACT

An Intelligent Interactive Learning System (IILS) is the result of combining a Virtual Reality Interactive System with educational technologies. These systems have been used in the training of procedural tasks, but there is a lack of research with regard to providing specific assistance for acquiring motor skills.

This work presents a novel approach to the diagnosis of motor skills for the IILSs that are based in the ULISES framework. This framework allows for the integration of Interactive Systems with different Intelligent Learning Systems. What is more, this work also presents the PATH authoring tool, an authoring tool that facilitates the creation of the knowledge models needed by the ULISES framework.

This research work involves developing new models and algorithms for the ULISES framework so that temporal uncertainty and the intentionality related to the students' movements when they are learning a new movement are managed. The methodology presents the integration of these models and algorithms for different layers and subsystems of ULISES, which aim at observing, interpreting and diagnosing students' movements in order to evaluate their motor skills.

Additionally, this work describes the creation of an authoring tool that facilitates the knowledge acquisition to create IILSs based on ULISES. Moreover, a knowledge-acquisition-methodology is included, which adds an extra value to the authoring tool.

The methodology has been validated with the creation of an IILS to the diagnosis of tennis motor skills, which is composed of a Motion Capture System. An experiment has also been conducted, which validates the methodology that has been proposed in this work. PATH has been validated with the creation of three IILSs in different domains and additional experiments have been carried out, which demonstrate the tool's value to create tasks for ULISES based IILS.

LABURPENA

Errealitate Birtualean oinarritutako sistemen eta Adimen Artifizialaren arteko elkarlana garrantzi handikoa da hezkuntzaren arlo askotan. Konbinazio honetan oinarritutako sistemak Ikasketarako Laguntza Sistema Interaktibo Adimentsuak (ILSIA) bezala ezagutzen dira eta hezkuntzarako funtzionalitate gama zabal bat eskaintzen dute. Halaber, zaila da abilezia fisikoen entrenamenduan asistentzia eskaintzen duten ILSIAk aurkitzea.

Tesi proiektu honetan ULISES frameworkean oinarritutako ILSIA-ekin abilezia fisikoak ebaluatzeko ikuspuntu berri bat proposatzen da. ULISES frameworkak, Sistema Interaktiboen eta Ikasketarako Laguntza Sistemen arteko integrazioa ahalbidetzen du, hain zuzen ere. Era berean, PATH izeneko autore-tresna ere aurkezten da lan honetan. Honen helburua ULISES frameworkarentzako beharrezkoak diren ezagutza ereduen sorkuntza prozesuaren kostua minimizatzea da.

Alde batetik, egindako lanaren ardatzetako bat ULISES frameworkak ziurgabetasun tenporalari eta intentzionalitateari aurre egiteko algoritmo eta ereduak garatzea izan da. Bi ezaugarri hauek ikasleak mugimenduak ikasten ari denean sortzen dira, eta beraz ezinbestekoa da ULISES frameworka hauek kudeatzeko gai izatea. Era berean, aipatutako eredu eta algoritmo horiek ULISES frameworkaren behatze, interpretazio eta diagnostiko maila eta azpisistemekin integratzeko lana ere aurkezten da.

Bestalde, ULISESen oinarritutako ILSIAk sortzeko beharrezkoa den ezagutza eskuraketarako metodologia bat ere gehitu da. Gainera, garatutako autore-tresnak metodologia honen aplikazio eta ikasketa erraztea du helburu.

Tenisaren domeinuan abilezia fisikoak ebaluatzeko atzemate sistema batez osatutako ILSIA bat garatu da, eta lan honen baliotasuna frogatzeko esperimentu bat aurrera eraman da. PATH autore-tresnari dagokionez, hiru domeinu desberdinetako hiru ILSIAren ariketa

sorkuntzan balioztatu da. Gainera, PATHekin esperimentu osagarriak egin dira, ULISES frameworkean oinarritutako ILSIAentzako ariketak sortzeko tresna baliogarria dela frogatuz.

CAPÍTULO 0

INTRODUCCIÓN

Una tarea procedimental se puede definir como una tarea cuyo aprendizaje requiere la integración de dos tipos de capacidades humanas: destrezas intelectuales y destrezas motoras (Gagné, 1965). Aparcar un coche, hacer un saque de tenis o el mantenimiento de maquinaria industrial son distintos ejemplos de tareas procedimentales, aunque las demandas cognitivas y psicomotoras que requiere cada una de esas tareas sean muy diferentes. Sin embargo, todas las tareas procedimentales tienen algo en común: se realizan siguiendo una secuencia de pasos.

El aprendizaje de tareas procedimentales es algo muy común en nuestras vidas y las empezamos a aprender desde el mismo momento en el que nacemos. No obstante, el aprendizaje de algunas tareas puede llegar a ser complicado, ya sea porque la tarea es arriesgada o porque no se dispone de los recursos de entrenamiento adecuados (espacio, tiempo, dinero etc.). Por esta razón, los sistemas basados en Realidad Virtual y Mixta (RV/RM) se han mostrado como herramientas útiles para el entrenamiento de tareas procedimentales. El principal objetivo de los sistemas de RV/RM suele ser la fiel reproducción de la realidad de los entornos de entrenamiento, pero si se quiere conseguir una verdadera efectividad educativa, estos sistemas deben combinarse con Sistemas de Ayuda al Aprendizaje (Bossard et al., 2010). Lozano-Rodero (2009) aborda en su trabajo de tesis la problemática que surge al integrar

ambas tecnologías, denominando la combinación de ambas como **SIIAA (Sistemas Interactivos Inteligentes de Ayuda al Aprendizaje)**.

0.1 MOTIVACIÓN

ULISES es un framework definido por Lozano-Rodero en su trabajo de tesis doctoral, gracias al cual es posible la creación de SIIAA para el aprendizaje de tareas procedimentales. ULISES establece los modelos de conocimiento necesarios para transformar la información que proviene de los sistemas RV/RM en una representación válida para un Sistema de Ayuda al Aprendizaje.

Como se ha citado al inicio de la introducción, las tareas procedimentales pueden ser de distinta naturaleza. Si nos centramos en el aprendizaje de destrezas motoras, existe un vacío en lo referente a SIIAA que sean capaces de evaluar destrezas motoras. En la mayoría de los casos, los SIIAA abarcan una amplia gama de funcionalidades educativas, como feedback en tiempo real, guiado instruccional o recomendaciones. Estas funcionalidades no suelen estar adaptadas al dominio de las destrezas físicas. Por ejemplo, si se quiere entrenar a una bailarina de ballet, habría que darle consejos para que mejore las poses y la coordinación, más allá de recordarle la secuencia de pasos que debe seguir.

En el caso de ULISES, tampoco se afronta la problemática necesaria para la evaluación de destrezas físicas, ya que para poder hacerlo hay que hacer frente a la **incertidumbre temporal**. Cuando una persona comienza a ejecutar un movimiento, resulta imposible saber qué movimiento está llevando a cabo hasta que pasa un mínimo instante tiempo. Es decir, la evaluación del movimiento se hace a posteriori. ULISES trabaja y evalúa la información que le llega en cada ciclo de simulación, pero no tiene en cuenta el manejo de esa incertidumbre temporal que es necesaria en el dominio de las destrezas motoras.

Por otro lado, la creación de un SIIAA requiere un proceso de adquisición de conocimiento que es costoso: ULISES recoge los datos que provienen de un sistema de RV/RM y los transforma en una representación de la actividad de los estudiantes que es analizable desde

un punto de vista educativo. Si el modelado del conocimiento para crear un SIIAA a través de ULISES ya es complejo por naturaleza, aun se convierte en mucho más complejo para algunos dominios como los que implican la evaluación de destrezas motoras. Por esta razón, es necesario disponer de herramientas que ayuden a adquirir y generar la información necesaria en la creación de un SIIAA.

Viendo esta problemática, este proyecto de tesis aborda el problema de la creación de SIIAA para la evaluación de destrezas motoras. El trabajo incide en la identificación de las características que hacen diferente la evaluación de las habilidades motoras y definen estas características dentro del marco de desarrollo para crear un SIIAA. Además, se aporta la metodología de adquisición de conocimiento y la herramienta de autor que la implementa para agilizar el proceso de adquisición de conocimiento de ULISES.

0.2 OBJETIVOS DE LA TESIS

Este trabajo de tesis tiene dos objetivos generales:

- Añadir capacidades a ULISES para que pueda trabajar con incertidumbre temporal y poder llevar a cabo el diagnóstico de destrezas físicas.
- La creación de una herramienta de autor que facilite la adquisición de conocimiento necesaria para ULISES a la hora de crear un SIIAA.

ULISES fue diseñado para facilitar la integración de funcionalidades educativas en todo tipo de Sistemas Interactivos creados para la práctica de tareas procedimentales. La metodología de desarrollo que define es independiente de la arquitectura interna, los dispositivos y las técnicas de interacción utilizados en los Sistemas Interactivos. Dado este punto de partida, se tiene como objetivo añadir capacidades de evaluación de destrezas motoras a ULISES, manteniendo intactas todas las funcionalidades genéricas del proceso de integración establecido en su metodología. Para ello, se llevará a cabo un estudio de las necesidades específicas para el diagnóstico de destrezas motoras y se añadirán nuevas capacidades a ULISES, incluyendo la de trabajar bajo incertidumbre temporal.

Más aun, es objetivo de este trabajo la creación de una herramienta de autor para facilitar la creación de los modelos de conocimiento de ULISES maximizando la usabilidad del framework. Además, la herramienta se centrará en aplicar el proceso de adquisición de forma metodológica para que los autores que usen la herramienta interioricen de una manera más fácil los conceptos del framework.

0.3 ESTRUCTURA DE LA MEMORIA

Esta memoria está organizada en 5 capítulos. A continuación se expone brevemente el contenido de cada uno de ellos.

En el primer capítulo se hace un análisis de los antecedentes de este trabajo. Para ello, en primer lugar se describen los SIIAA más significativos para este proyecto. A continuación, se presentan aquellos sistemas que son capaces de trabajar bajo incertidumbre y los trabajos más significativos sobre diagnóstico de habilidades motoras. Por último, se analizarán las herramientas de autor más relevantes para adquisición de conocimiento de SIIAA.

En el segundo capítulo se explica la creación de SIIAA basados en la plataforma OLYMPUS, una plataforma creada en torno al framework ULISES. Ya que la base de esta plataforma es ULISES, se hará una descripción detallada del framework y de las herramientas que componen OLYMPUS.

El tercer capítulo presenta el desarrollo que se ha llevado a cabo para que ULISES sea capaz de diagnosticar destrezas motoras. Se describe como se han implementado los distintos subsistemas de ULISES y cómo se aplica la metodología para diagnosticar destrezas físicas. A su vez, se presenta la validación sobre el diagnóstico de destrezas motoras con el framework ULISES.

El cuarto capítulo detalla el proceso de adquisición de conocimiento creado para ULISES, y la herramienta de autor, llamada PATH, que se ha desarrollado para facilitar su puesta en práctica. Asimismo se describe la evaluación llevada a cabo para validar la herramienta y se presentan los resultados y conclusiones obtenidas de la misma.

Finalmente, en el capítulo 5 se exponen las conclusiones y las líneas futuras de investigación, además de resaltar las aportaciones más relevantes de este trabajo de investigación.

CAPÍTULO 1

ANÁLISIS DE ANTECEDENTES

Hoy en día existen SIIAA que han servido de gran ayuda en el ámbito de la educación o la industria. Durante estos años no ha cesado el trabajo de desarrollar sistemas que entrenen a los estudiantes para conseguir que el alumno transfiera con éxito las habilidades adquiridas durante el entrenamiento asistido por computador al entorno real. Sin embargo, aún existe un vacío en lo que se refiere a SIIAA que asistan en el entrenamiento de destrezas físicas. El propósito de este capítulo es ilustrar la evolución que han seguido los SIIAA durante los últimos años y analizar hasta donde se ha llegado en el entrenamiento de destrezas físicas. Además, se introducen los términos y conceptos que se emplearán en el resto de la memoria.

Posteriormente se presenta un estudio sobre las Herramientas de Autor que existen en este ámbito. La relación coste-eficacia en la creación de SIIAA es un tema que se encuentra a la orden del día y por lo tanto, el análisis de las Herramientas de Autor resulta imprescindible.

El estudio de los SIIAA para el entrenamiento de destrezas físicas y de las Herramientas de Autor existentes para las mismas sirve para obtener una perspectiva global del alcance de los objetivos de este proyecto te tesis y para aportar una visión general de los problemas a los que se pretende dar respuesta.

1.1 SISTEMAS INTELIGENTES DE AYUDA AL APRENDIZAJE: PERSPECTIVA HISTÓRICA

Los ordenadores y la Inteligencia Artificial han sido utilizados con objetivos educativos desde la década de los 60. Estos objetivos han ido cambiando a lo largo de la historia y la naturaleza de los contextos de aprendizaje donde se aplican se ha diversificado considerablemente. De esta manera, han ido surgiendo distintos paradigmas de desarrollo a lo largo de la historia.

Los Sistemas de Enseñanza asistida por Ordenador (*Computer-Aided Instruction, CAI*) dieron origen a los distintos grupos de tecnologías relevantes que existen hoy en día (Brusilovsky & Peylo, 2003): los Sistemas Inteligentes de Enseñanza asistida por ordenador (*Inteligent Computer-Aided Intruction, ICAI o AI-ICAI*) y los Sistemas Tutores Inteligentes (*Intelligent Tutoring Systems*). Estos dos tipos de sistemas se diferenciaron por las técnicas que emplean para asistir al estudiante, pero ambos evolucionaron de los CAI por una razón: ambos se centraron en representar el conocimiento del estudiante en el sistema. En este momento fue cuando se hizo uso por primera vez del término *modelo de estudiante* (Self et al., 1999)

Los ICAI se centraron principalmente en la planificación instruccional (Brown et al., 1973; Wescourt et al., 1977) y en el análisis inteligente de soluciones (Self, 1974). La idea de los CAI /ICAI era sustituir totalmente o parcialmente la enseñanza clásica en aula. Sin embargo, a finales de la década de los 70, esta idea se fue abandonando y se avanzó hacia el paradigma de los Sistemas Tutores Inteligentes. Esta propuesta, sugería la construcción de sistemas más flexibles que los CAI/ICAI, dotados de iniciativa y que favorecieran la interacción con el alumno para ofrecerle tutorización individualizada en su proceso de aprendizaje (Burton & Brown, 1979; Clancey, 1979).

1.2 PARADIGMAS DE PROGRAMACIÓN DE SISTEMAS TUTORES INTELIGENTES

Los Sistemas Tutores Inteligentes (STI) emergieron en las décadas de los 80 y 90, siendo su objetivo ofrecer las ventajas de la tutorización *one-on-*

one sin requerir la presencia de un tutor real para cada estudiante. Los STIs se caracterizan por compartir una arquitectura basada en cuatro módulos principales: módulo de dominio, módulo pedagógico, modelo de estudiante y módulo de diálogo (Wenger, 1987) (Figura 1). Los STIs permiten la creación de sistemas muy flexibles dependiendo de la inteligencia que se le quiera dar a cada uno de estos cuatro módulos. Por ejemplo, un STI que se centra en la *inteligencia* del modelo de dominio, podrá generar soluciones diversas y complejas para que los estudiantes siempre tengan nuevos problemas con los que practicar. A su vez, la inteligencia que se le quiera dar a cada módulo, tendrá un impacto directo en la dificultad para crear el STI.

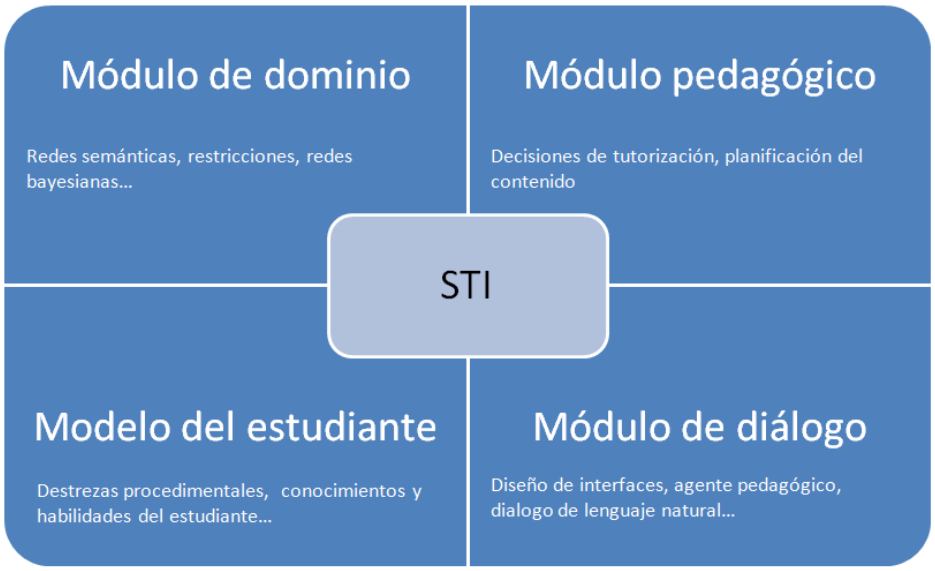


Figura 1: Módulos que componen un STI

Los STI también se pueden clasificar teniendo en cuenta los algoritmos en los que basan su funcionamiento. En este aspecto, los STI se pueden dividir en tres grandes grupos: Tutores cognitivos (J. Anderson & Corbett, 1995), tutores basados en restricciones (*Constraint-based* tutors)(Mitrovic et al., 2009) y tutores Example-tracing (Aleven et al., 2009):

- Tutores cognitivos: Un tutor cognitivo es un tipo de STI que utiliza un modelo cognitivo para dar feedback a los estudiantes que

trabajan con problemas de ejemplo. Ese modelo cognitivo se basa en las teorías cognitivas ACT* (Anderson, 1983) y ACT-R (Anderson, 1993), las cuales se basan en la afirmación de que las habilidades cognitivas se pueden entender como un conjunto de reglas de producción *if-then*. Estos tutores se utilizan para hacer el seguimiento de modelos o *model-tracing* o para hacer el seguimiento sobre el conocimiento del alumno o *knowledge-tracing*. Básicamente, estas técnicas consisten en hacer un seguimiento de la actividad del alumno y compararla paso a paso con la planificación realizada por el tutor. Por este motivo, este tipo de tutores suelen implementar técnicas de diagnóstico basadas en reconocimiento de planes (Carberry, 2001).

- Tutores basados en restricciones: Estos tutores se crean utilizando el modelado basado restricciones (*Constraint-Based Modelling*, CBM)(Ohlsson, 1992;Mitrovic, 2011). Con este paradigma se definen restricciones sobre las acciones que pueden realizar los alumnos al resolver un problema para modelar las soluciones posibles. En los tutores *model-tracing* es necesario representar todas las posibles soluciones a un problema, y esto puede ser en un problema en dominios pobremente definidos (*ill-defined domains*) ya que es muy difícil o imposible abordar todas las posibilidades. En cambio, el modelado basado en restricciones permite definir solo las acciones interesantes, incluso permite la definición de reglas para identificar errores específicos. Aprovechando estas características, el modelado basado en restricciones permite agrupar las acciones que violan los mismos principios del dominio.
- Tutores *example-tracing*: Estos tipos de tutores se asemejan mucho a los tutores cognitivos (incluso pueden llegar a ser iguales (Aleven et al., 2009)) y se crearon con la intención de agilizar la creación de los tutores cognitivos. Así como los tutores *model-tracing* siguen e interpretan el comportamiento del estudiante en base a un modelo cognitivo, los tutores *example-tracing* interpretan el comportamiento del estudiante en base a unos *ejemplos* específicos sobre cómo debe solucionarse un problema. Estos ejemplos resumen las distintas soluciones aceptables que el alumno puede dar.

Los tutores cognitivos fueron utilizados satisfactoriamente en las aulas (Koedinger et al., 1997), pero su éxito fue mitigado por el coste que suponía crear éste tipo de tutores (Murray, 2003). Los tutores basados en restricciones, los tutores *model-tracing* y los *example-tracing* fueron creados con el objetivo de reducir ese coste. Sin embargo, generar dominios completos, modelar a los estudiantes con la precisión necesaria y desarrollar las estrategias de enseñanza componen una labor de gran complejidad, por lo que el coste de generar un Sistema Tutor Inteligente sigue siendo elevado. Para minimizar este coste, en las últimas décadas se han desarrollado múltiples herramientas de autor que se discutirán en más profundidad en la sección 1.5.

1.3 SISTEMAS DE AYUDA AL APRENDIZAJE DE TAREAS PROCEDIMENTALES

Aprender una tarea procedimental exige diferentes tipos de capacidades humanas y, dependiendo de las destrezas que se quieran entrenar, los Sistemas de Ayuda al aprendizaje varían mucho respecto al tipo de asistencia que ofrecen al alumno. En esta sección, en primer lugar se analiza la importancia de cada tipo de destreza y se hace una clasificación ilustrativa de algunos dominios procedimentales en función de la predominancia de las destrezas que exigen. A continuación, se describen los distintos tipos de SIIAA que existen en base al dominio procedimental al que pertenecen.

1.3.1 Destrezas requeridas en las tareas procedimentales

En general, una tarea procedimental se define como aquella que se resuelve siguiendo una secuencia de pasos. En este contexto, resolver un problema matemático y aparcar un coche serían tareas procedimentales, aunque requieren destrezas muy diferentes. Otros autores acotan más la definición, y consideran que una tarea procedimental es aquella que para realizarla requiere la combinación de dos clases de capacidades humanas: capacidades intelectuales y capacidades motoras (Gagné, 1965). Si nos atenemos a la definición de Gagné, la resolución de un problema matemático no sería explícitamente una tarea procedimental, ya que esa tarea no requiere ninguna destreza psicomotora. Aun así,

muchos autores tratan dominios del estilo de la matemática como procedimentales, centrándose en la identificación de pasos que hacen falta para resolver los problemas y las relaciones que se dan entre ellos.

La Figura 2 ilustra a modo de ejemplo general las diferencias de carga intelectual y psicomotora que demandan algunas tareas procedimentales. Resolver un problema matemático o físico y diseñar una base de datos son tareas que requieren destrezas intelectuales diferentes. Sin embargo, estas tareas coinciden en que implican el uso de destrezas puramente intelectuales, y ninguna psicomotora. En otro nivel se pueden encontrar las tareas de mantenimiento de maquinaria. Estas tareas suelen requerir por lo común una menor exigencia intelectual que las anteriores, y a su vez exigen ciertas habilidades psicomotoras: percepción espacial para identificar o saber dónde buscar los mecanismos de la maquinaria relevantes en cada momento, capacidad de navegación si los desplazamientos tienen importancia, capacidad de manipulación de elementos, etc. Obviamente, el nivel de capacidad intelectual varía dependiendo de la complejidad de uso de la maquinaria y montar piezas exigirá mayor destreza psicomotora que la monitorización de una máquina. Si se avanza a la parte derecha del diagrama, otro ejemplo relevante es el dominio de la conducción, que requiere una destreza significativa tanto intelectual como psicomotora. En el mismo grupo, el pilotaje de un avión demanda mayor destreza intelectual que conducir un coche, y ambas tareas requieren grandes habilidades de percepción espacial.

En el último grupo se encuentran las actividades físicas, como la danza, el tenis o los deportes en general. Estas requieren mayor destreza motora que las anteriores, aunque las destrezas intelectuales también influyan. Por ejemplo, a la hora de hacer un saque de tenis, hay que saber cuál es la secuencia correcta de pasos que hay que llevar a cabo además de tener una técnica depurada. Más aún, durante el juego, el tenista debe mostrar su capacidad de análisis de situaciones y de poner en marcha estrategias efectivas para ganar el partido.

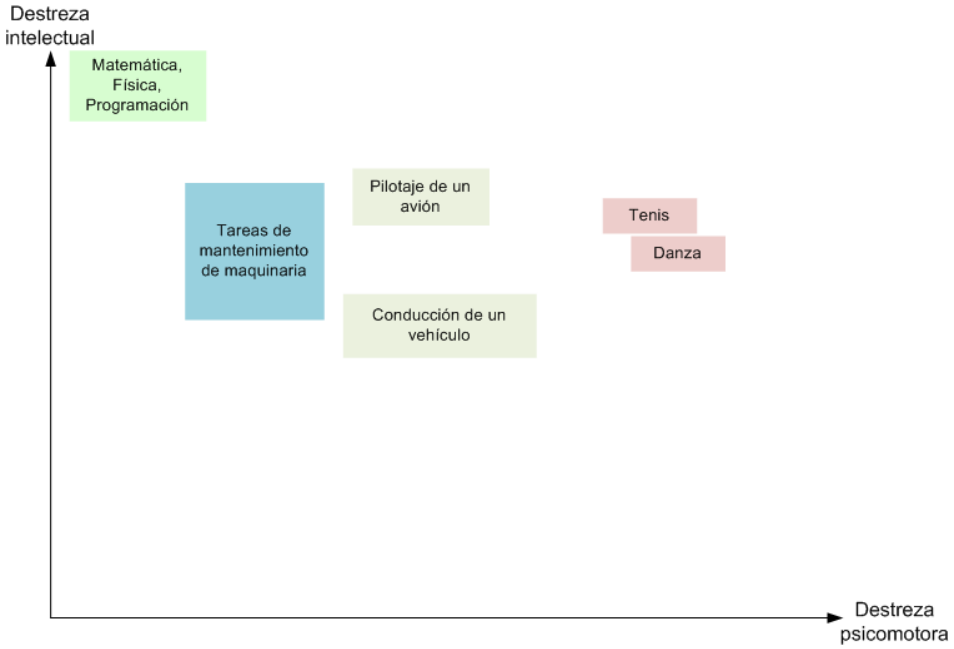


Figura 2: Clasificación de los dominios procedimentales basándose en la carga intelectual y psicomotora que requieren.

Dadas las diferencias entre tareas procedimentales, incluyendo sus posibles definiciones, en el resto del capítulo se presentan sistemas de ayuda al aprendizaje y tecnologías asociadas distinguiendo el tipo de destrezas a las que dan prioridad de aprendizaje.

1.3.2 Sistemas Interactivos Inteligentes de Ayuda al Aprendizaje

Cuando entran en juego tanto las destrezas intelectuales como las psicomotoras, los sistemas orientados a la práctica (*performance-oriented systems)* adquieren mayor importancia y los Sistemas Interactivos (SI) basados en simulación resultan indispensables. El entrenamiento de personal es una de las aplicaciones prácticas más claras de los SI, por lo que la lista de sistemas que incorporan algún tipo de funcionalidad educativa puede ser interminable. Sin embargo, la lista se reduce mucho cuando se trata de funcionalidades avanzadas que ayuden en el aprendizaje de tareas procedimentales.

Dependiendo del dominio procedimental donde se quiera aplicar la enseñanza, un SIIAA tiene que ser capaz de lidiar con acciones continuas, streams y tipos de datos complejos provenientes del SI: sistemas de captura de movimiento, simulador de conducción etc. Cuando se gestionan este tipo de datos, la extracción de la semántica de estas acciones difiere considerablemente de sistemas donde las acciones son representables mediante simples eventos. Por ejemplo, un SIIAA para aprender matemáticas difiere mucho de un SIIAA para el aprendizaje en el dominio de la conducción. En el dominio de las matemáticas, un dato de entrada puede ser un texto o un click para seleccionar un texto de la aplicación, mientras que en el dominio de la conducción hay que gestionar un gran número de variables en tiempo real: todos los mandos del vehículo, el estado de los objetos con los que el vehículo interactúa etc. Queda claro, que en el segundo caso la gestión de los datos de entrada es mucho más complicada. Además, normalmente los problemas que se plantean en el dominio de las matemáticas suelen estar bien estructurados, es decir, los pasos a seguir para encontrar la solución a un problema están bien definidos. Sin embargo, en un entrenamiento de conducción, la incertidumbre a la que hay que hacer frente es mucho mayor.

Siguiendo con el tema del dominio, existe otro término al que en los últimos años se le está dando mucha importancia a la hora de diseñar un ITS: el nivel de definición de la tarea (*ill-definedness*)(Fournier-viger et al., 2010; Mitrovic & Weerasinghe, 2009). Según Simon (Simon, 1978), un problema pobremente estructurado (*ill-structured problem*) es un problema complejo de resolver, con múltiples puntos de partida y múltiples y discutibles soluciones o estrategias para buscar soluciones difusas. Los dominios que incluyen este tipo de problemas son denominados como dominios pobremente definidos (*ill-defined domains*). El tipo de estructuración del problema cobra mucha importancia en el mundo de los STI, ya que dependiendo de esta estructuración se eligen los enfoques (paradigmas) sobre los que se van a construir los STI. Los tutores *model-tracing* no se adaptan bien al aprendizaje de tareas pobremente definidas, ya que en estos casos no suele existir una estrategia clara para encontrar la solución a una tarea. Sin embargo, funcionan muy bien en las tareas bien definidas, ya que la

precisión del seguimiento que se le hace al modelo cognitivo les permite sugerir los pasos que deben seguir los alumnos, dar demostraciones, evaluar el nivel de conocimiento que están adquiriendo e inferir los objetivos de aprendizaje alcanzados. En cambio, los tutores que utilizan el modelado basado en restricciones no son adecuados para hacer ese seguimiento al estudiante en las tareas bien estructuradas. El modelado basado en restricciones se centra en especificar cuál es el comportamiento correcto (o incorrecto) en vez de especificar un "camino" a seguir para dar solución a la tarea, por lo que funcionan bien en tareas pobremente estructuradas.

1.3.3 Entrenamiento de destrezas en las tareas procedimentales

Existen muchos Sistemas de Ayuda al Aprendizaje (SAA) para el entrenamiento de tareas procedimentales. Muchos de ellos se utilizan en dominios que no requieren ninguna destreza física. Otros consideran tanto las destrezas psicomotoras como las intelectuales, aunque hay que destacar que la ayuda inteligente del SAA va disminuyendo a medida que nos acercamos a dominios donde la evaluación de destrezas psicomotoras se hace imprescindible. En la actualidad, no existen muchos SAA que asistan a los estudiantes en el entrenamiento de destrezas físicas, y la evaluación y asistencia que ofrecen es menor que la que se ofrece en otros SAA. En las siguientes secciones se describen los sistemas de cada uno de estos grupos.

1.3.3.1 Entrenamiento de destrezas intelectuales

Los SAA que tratan el entrenamiento de destrezas intelectuales no son un claro ejemplo para el aprendizaje de tareas procedimentales, ya que solo tratan la parte intelectual de la tarea. Además, en general no suelen ofrecer una gran interacción con el usuario y suelen tratar acciones que son representables mediante simples eventos (p. ej. clicks de ratón). Sin embargo, las técnicas o paradigmas de programación que ofrecen estos tipos de sistemas se pueden utilizar (y se han utilizado) satisfactoriamente en otros SAA de tareas procedimentales. Asimismo, destacan por el tipo de asistencia que ofrecen a los alumnos y el seguimiento que hacen sobre los mismos.

SQL-Tutor (Mitrovic, 1998) es un ejemplo significativo de los STI de este grupo que se emplea para el aprendizaje del diseño de bases de datos relacionales. Este tutor selecciona problemas individualizados para los estudiantes basándose en sus modelos. Dependiendo de los errores que cometan al resolver los problemas, el tutor genera feedback adaptativo y actualiza el modelo del estudiante. EER-Tutor (Mitrovic et al., 2007) y su predecesor KERMIT (Suraweera & Mitrovic, 2002) son otros dos tutores de similares características cuyo objetivo es también enseñar a diseñar bases de datos relacionales. La principal característica de estos sistemas es que son capaces de tutorizar tareas pobremente definidas, para lo que utilizan un modelado basado en restricciones como paradigma de programación. Andes (VanLehn et al., 2005) es otro STI para el aprendizaje de problemas de física que se podría clasificar junto a los anteriores. Tiene una interfaz gráfica (Figura 3) que permite al usuario leer un problema físico, definir variables para resolverlo y dibujar vectores. Este STI lleva a cabo la tutorización dando un feedback inmediato a los errores cometidos por los estudiantes y generando pistas sobre los siguientes pasos que deben dar para solucionar el problema que se les ha presentado. Asimismo, tiene una base de conocimiento con reglas de física definidas por profesores en ese dominio. Teniendo en cuenta estas reglas, Andes genera automáticamente unas redes Bayesianas basándose en el output de su solver. Estas redes Bayesianas generan el grafo de soluciones, donde se representan todas las posibles soluciones al problema. Por último, se podrían citar los tutores cognitivos que se han creado para enseñar programación, geometría y matemáticas (Ritter et al., 2007). Las herramientas que se utilizan en estos tutores suelen ser parecidas a la de Andes en cuanto a interfaz, pero al ser dominios bien definidos utilizan el paradigma *model-tracing* para hacer el seguimiento del modelo de estudiante.

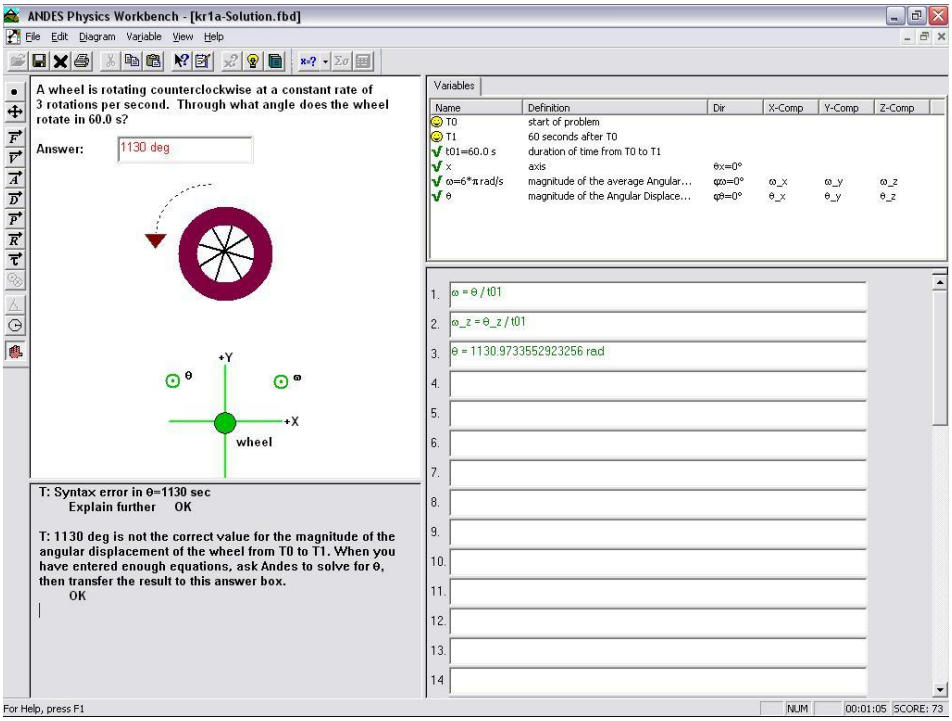


Figura 3: Interfaz del STI Andes (obtenido de www.learnlab.org).

1.3.3.2 Entrenamiento de destrezas INTELECTUALES y físicas

Los SAA que tratan tareas procedimentales que requieren destrezas intelectuales y físicas se suelen clasificar como sistemas orientados a la práctica. Estos sistemas, se centran más en proveer un entorno de aprendizaje completo donde los estudiantes aprenden las tareas mediante la práctica, mientras se les ofrece guiado y feedback. Dentro de esta clasificación, se pueden encontrar los SAA basados en simulación, o como se han denominado en este trabajo, los SIIAA.

Entre los SIIAA más antiguos podemos encontrar los tutores generados con la herramienta de autor **XAIDA** (Wenzel et al., 1998). Esta herramienta genera tutores para el entrenamiento de tareas de mantenimiento y tienen una marcada orientación práctica. Básicamente toman en cuenta las características físicas de la maquinaria, su teoría operativa, los procedimientos operativos y de mantenimiento y las soluciones para los problemas que puedan surgir con la maquinaria. La asistencia que ofrecen estos sistemas respecto a la destreza física es

mínima, ya que no son la determinante para que los alumnos aprendan a realizar las tareas. En su lugar, se centran en que los alumnos aprendan las distintas partes de los sistemas o los pasos que hay que seguir dentro de un procedimiento.

STEVE (Soar Training Expert for Virtual Environments)(Rickel & Johnson, 1999) es el nombre de un agente pedagógico animado 3D. Este agente está enfocado al entrenamiento, sobre todo, en control de maquinaria. Más concretamente, STEVE ha sido utilizado para entrenar a operarios de navíos en el uso de la consola de control de turbinas o en el manejo de las válvulas y compresores de aire de los motores (Stiles, 1999). STEVE también ha sido utilizado para el entrenamiento de tareas de equipo (McCarthy et al., 1999) o para la toma de decisiones en escenarios militares (Rickel et al., 2001). Entre las capacidades de STEVE se encuentran: demostrar cómo se realiza un procedimiento paso a paso en el entorno virtual, comunicarse con el alumno mediante síntesis y reconocimiento de voz, contestar a preguntas, adaptar los procedimientos del dominio a eventos inesperados y recordar acciones pasadas.

VIVIDS (Munro & Surmon, 1997) es el STI sobre el que funciona STEVE y utiliza una representación de red procedimental. Esta red define cuales son los pasos que representan la tarea procedimental y el orden correcto para llevar a cabo esos pasos. Sus autores justifican que este tipo de representación es mucho más adaptable a distintos dominios y tutores que, por ejemplo, las reglas de producción que se utilizan en los tutores cognitivos para representar el modelo cognitivo. En la representación de red procedimental se definen planes jerárquicos, los cuales están compuestos por pasos (ej. pulsar un botón) o acciones compuestas (ej. un subplan). Además, estos pasos pueden contener restricciones de orden y prerrequisitos. Los prerrequisitos están representados por vínculos causales; cada uno de estos vínculos especifica cuál es paso que hace que un objetivo de la tarea se cumpla y que a su vez puede ser una precondición para que otro paso comience (o acabe la tarea). Además del dominio procedimental, STEVE también gestiona el conocimiento perceptual de la simulación El conocimiento perceptual transmite a STEVE la información sobre los objetos en el

entorno virtual (personas, el propio STEVE y otros objetos de la simulación), sus atributos de simulación y sus propiedades espaciales. Esta información está representada por una lista de pares atributo-valor (ej. *estado_luz1: off*) que se actualiza periódicamente. Esta manera de representar el conocimiento del dominio requiere tener un conocimiento muy detallado de lo que puede hacer el alumno. Si esto es una gran ventaja a la hora de enseñar tareas bien definidas como el mantenimiento de maquinaría, se puede convertir en una pesadilla en tareas pobremente definidas como la conducción.

En este contexto, cabe destacar el proyecto europeo SKILLS. Este proyecto aborda el desarrollo de nuevas metodologías de adquisición, interpretación, almacenamiento y simulación de destrezas procedimentales y motoras mediante el uso de interfaces multimodales y entornos virtuales. En este proyecto se han desarrollado demostradores para diferentes dominios (Gutierrez et al., 2008), entre ellos uno específico para tareas de mantenimiento y ensamblaje industriales. El objetivo de este trabajo es la transferencia de destrezas motoras y procedimentales. En lo referente a las destrezas motoras, se definen dos métricas: la ejecución óptima de operaciones básicas y la correcta ejecución de los movimientos. En cuanto a las destrezas procedimentales se tienen en cuenta: la secuencia de pasos, el conocimiento sobre los movimientos a llevar a cabo, y la elección de herramientas correctas. Aunque se definen metricas para la transferencia de destrezas motoras, este trabajo se centra en la parte procedimental de la tarea. El sistema de entrenamiento propuesto detecta errores en la secuencia de pasos, y ofrece guiado y feedback para las tareas procedimentales de mantenimiento (Gutiérrez, et al. , 2010). Sin embargo, no ofrece información sobre la causalidad de los errores cometidos durante la tarea.

Stottler Henke Associates (www.stottlerhenke.com) es una compañía estadounidense que está especializada en la aplicación de técnicas de Inteligencia Artificial al desarrollo de sistemas de entrenamiento. Han desarrollado varios SIIAA para el entrenamiento de tareas procedimentales, entre los que se encuentran uno para el entrenamiento de operarios de la NASA (Ong et al., 2000) y otro para el

pilotaje de helicópteros. (Remolina et al., 2004). Este último resulta más relevante para este proyecto de tesis ya que el pilotaje se trata de un dominio pobremente definido y además requiere mayor destreza psicomotora que las tareas de entrenamiento que se han descrito hasta el momento.

El **IFT** (Intelligent Flight Trainer) es un STI para pilotos de helicóptero que consiste de un simulador de vuelo integrado con un STI. El STI está compuesto de dos sistemas diferentes: el ayudante y el asesor (*advisor*), los cuales ofrecen un comportamiento adaptativo dependiendo del modelo del estudiante. El primero ayuda en funciones como corregir el vuelo del helicóptero, lo que equivaldría a sujetar la bici mientras aprendemos a montar para no caernos. Dependiendo del nivel del estudiante se ajusta la ayuda que recibe durante el vuelo. El sistema asesor se comunica verbalmente con el alumno y tiene cuatro funciones principales: instruye a los alumnos con procedimientos básicos, monitoriza la ejecución de la tarea (frases como: "vigila tu velocidad"), monitoriza la actividad de los controles y por último hace sugerencias a los aprendices de vuelo.

El STI de IFT requiere que se le asignen objetivos de aprendizaje del tipo *aprender el control cíclico a la derecha, disminuir altura* o *aumentar velocidad*. Teniendo en cuenta estos objetivos, el sistema da un feedback visual y verbal dependiendo del comportamiento del estudiante. Por ejemplo, si el estudiante pierde altitud, se le enseña el procedimiento para solucionar ese problema. Por otra parte, el sistema también es capaz de evaluar la automaticidad de las acciones. Es decir, el STI evalúa si el sistema es capaz de llevar a cabo tareas principales (ej.: mantener altura) mientras se le generan tareas secundarias (ej: pulsar botones).

El conocimiento procedimental se modela mediante *comportamientos*, los cuales especifican los procedimientos que se deben llevar a cabo en cada situación o condición. Estos comportamientos se modelan como máquinas de estado finitas. La lógica de las condiciones se evalúa bajo un orden establecido. Para que las máquinas de estados pasen de un estado a otro, se utilizan 4 primitivas:

- Acciones: Definen todas las posibles acciones que el sistema puede ejecutar.

- Comportamientos: Ligan las acciones con la lógica condicional.
- Predicados: Establecen las condiciones para que cada acción y comportamiento ocurran.
- Conectores: Controlan el orden en el que se tienen que evaluar las condiciones.

Estos cuatro elementos permiten modelar las actividades de los estudiantes, que pueden oscilar desde secuencias simples hasta lógica condicional compleja. En lo referente a la evaluación de los procedimientos, se llevan a cabo a través de máquinas de evaluación (*evaluation machines*). Estas máquinas de evaluación lanzan eventos cuando se cumple alguna condición durante el ejercicio, por ejemplo cuando se está volando demasiado alto, o cuando la palanca de mando se ha movido como se esperaba. Si se generan muchos eventos se priorizan para dar un feedback. Además, gracias a las máquinas de evaluación se sabe en todo momento cuál es el estado en el que se encuentran los procedimientos, definiendo así el contexto donde se están llevando a cabo las acciones. Gracias a ello, se puede dar un feedback contextualizado.

En general, el STI utilizado en el IFT da una asistencia completa al estudiante. Esta asistencia se centra mucho en la secuencia de pasos que debe dar en las diferentes situaciones que se puede encontrar el piloto. Además de la secuencia de acciones, también se tiene en cuenta el tiempo que pasa entre una acción y otra o la duración de las acciones, por lo que de alguna manera, el sistema es capaz de enseñar a coordinar distintas acciones. Por último, el sistema también es capaz de evaluar si las acciones se están ejecutando con la cautela adecuada, por ejemplo si se está ascendiendo o descendiendo bruscamente.

1.3.3.3 Entrenamiento de destrezas FÍSICAS e intelectuales

Los SIIAA descritos en los dos apartados anteriores se centran principalmente en enseñar el orden en el que se lleva a cabo el procedimiento, lo que puede ser válido en los entornos de mantenimiento de maquinaria o pilotaje de vehículos. En estos SIIAA la destreza física está presente pero no es la más importante del aprendizaje. Hay otros dominios donde analizar la secuencia de las

acciones no es suficiente, también puede hacer falta evaluar características como la coordinación o la corrección de los movimientos. Dependiendo del dominio, a cada sistema le conviene ofrecer asistencia de distinto tipo en lo referente a la destreza física que se esté tratando. Mientras un SIIAA para el dominio de la cirugía debe dar asistencia para mejorar la precisión de los movimientos o evaluar la velocidad con la que se ejecutan las acciones, un tutor para la enseñanza de danza debe también ayudar a coordinar los movimientos que se llevan a cabo.

En la actualidad no existen muchos SIIAA que se centren en el entrenamiento de destrezas físicas. Los SIIAA que más aportan se reducen básicamente a los dominios de la cirugía, ejercicio físico y manejo de maquinaria (por ej.: excavadora, brazos robóticos). A continuación se detalla el trabajo que se ha hecho en cada dominio y cómo se gestiona la asistencia referente a las destrezas físicas en cada uno de ellos.

- **Dominio de operaciones de maquinaria**

Dentro de este dominio existen dos trabajos que sobresalen respecto a los demás. CanadarmTutor (Kabanza & Nkambou, 2005) y un STI para la evaluación de destrezas en el manejo de maquinaria parcialmente automatizada (Tervo et al., 2010).

CanadarmTutor es un STI basado en simulación para enseñar a utilizar Canadarm2, un brazo robótico de 7 grados de libertad que se utiliza en la Estación Espacial Internacional. Su acción principal es la de enseñar a mover el brazo robótico desde un punto hasta otro. A la hora de realizar esa trayectoria hay muchas posibilidades diferentes para mover el brazo: hay que esquivar distintos objetos, hay que tener en cuenta la vista del brazo que tiene el usuario (puede que no haya visibilidad suficiente dependiendo de donde se encuentre el brazo), hay que evitar singularidades y hay que tener en cuenta cómo de familiarizados están los estudiantes con los movimientos que pueden ejecutar. En consecuencia, se puede decir que se trata de un dominio pobremente definido.

En un principio, CanadarmTutor fue programado con un sistema experto. Sus autores integraron un *path-planner* que creaba los

movimientos correctos que debía seguir el brazo teniendo en cuenta su visibilidad respecto a las cámaras. Sin embargo, los trazados creados por el path-planner no eran realistas o fáciles de seguir, y cómo no había ninguna representación de las destrezas era imposible hacer un seguimiento al alumno. Por esta razón, se optó por el paradigma *model-tracing*, pero éste también resulto ser fallido. Los modelos de tareas funcionaban si se especificaban los pasos básicos a seguir, pero resultó imposible refinar el modelo con mayor detalle: no llegaron a poder especificar todas las rotaciones de las articulaciones del brazo robótico que una persona podía ejecutar.

Para evitar estas limitaciones, utilizaron un nuevo paradigma basado en minería de datos (Fournier-viger et al., 2011). Este método consiste en grabar las soluciones que los usuarios dan a una tarea y aplicarle técnicas de minería de datos para extraer partes de soluciones que ocurren frecuentemente. De esta manera, aunque no se defina todo el espacio de soluciones, el sistema es capaz de sugerir los pasos que se pueden dar para llegar al objetivo, por ejemplo cómo deben rotar la articulación del brazo robótico en una situación específica. El problema principal de utilizar sólo este enfoque radica en que no se pueden ofrecer soluciones a los estudiantes si siguen caminos que no se han seguido previamente. En vista de estos problemas, la última versión de CanadarmTutor utiliza un paradigma híbrido (Fournier-Vige et al., 2013). Si el usuario necesita ayuda, se hacen sugerencias sobre el camino que deben seguir con el plan que ofrece el modelo cognitivo y se hacen sugerencias sobre las rotaciones de las articulaciones utilizando el modelo de tareas parcial. Si no se puede ofrecer ayuda con estos dos métodos, se genera un plan con el path-planner.

Tervo et al. (2010) presentaron un framework general para la evaluación de destrezas que emplea un Hidden Markov Model (HMM) para reconocer secuencias de tareas y subtareas. El trabajo está evaluado en un entorno de maquinaria parcialmente automatizada (excavadora equipada con motosierra o segadora) y tiene en cuenta cuatro métricas para medir la destreza humana:

- *Eficiencia de la tarea:* Se trata la capacidad que tiene el operario de realizar la tarea sin derrochar recursos. Es decir, cuando se compara

a un experto con un principiante es muy posible que el principiante consuma más recursos que el experto. En este trabajo la eficiencia es medida por el tiempo que se tarda para realizar la tarea.

- *Complejidad de la secuencia de tareas:* Este enfoque asume que un operario experto consigue la misma meta que un principiante con movimientos menos bruscos y con menor número de acciones.
- *Habilidad para planificar y tomar decisiones:* Cuando se trabaja con tareas complejas en entornos dinámicos con restricciones de tiempo los expertos eligen intuitivamente la solución correcta y después ejecutan la tarea con decisión. En cambio un operario puede llevar a cabo la misma acción varias veces hasta que le salga bien. Varios cambios de dirección o paradas en el camino indicarían indecisión y que tiene pocas habilidades respecto a la toma de decisiones.
- *Dificultad de la tarea:* Se toma en cuenta que para llevar a cabo tareas con la mayor eficiencia posible es imprescindible manipular funciones distintas de la maquinaria a un ritmo alto y solapando las funciones en vez de llevarlas a cabo por separado. Por lo tanto, el número total de funciones controladas en paralelo y la media de tiempo que se tarda en cada función se utiliza como medida de la dificultad de la tarea.

La metodología que presentan evalúa las destrezas de los operarios teniendo en cuenta las señales de entrada de los controles que utilizan los operarios: botones y palancas de movimientos. Para llevar a cabo esta evaluación utilizan un HMM definiendo sus estados explícitamente. El rol de la estructura del HMM es intentar reconocer los intentos del operario que están causados por las acciones resultantes. En este caso las acciones resultantes serían las de control de maquinaría como pulsar botones o controlar la palanca de movimientos. El entrenamiento de modelos estadísticos como HMM para representar diferentes niveles de destrezas es un modelo popular (Yang et al., 1997), (Solis & Takanishi, 2008). Estos tipos de modelos estadísticos suelen hacer una comparación entre la representación de una destreza del experto y la actividad llevada a cabo por un aprendiz. El resultado que suelen dar es una probabilidad, la medida en la que la actividad llevada a cabo por el estudiante se parece a la del experto, pero no ofrecen resultados sobre la

causalidad. Por tanto, es imposible saber por qué los alumnos no llegan al nivel de destreza deseado. Además, si existen múltiples maneras de llevar a cabo una tarea de manera satisfactoria, se puede volver imposible o muy poco práctico definir modelos para la evaluación de destrezas.

Viendo este problema, Tervo et al. ofrecen otra perspectiva para evaluar destrezas mediante la utilización de HMM. Dividen el trabajo que hay que hacer en una tarea en distintos ciclos de trabajo o subtareas. Cada estado de HMM corresponde a una fase operacional del trabajo (ej: pisar embrague). Además el HMM almacena la secuencia de estados, que se refiere a la secuencia de subtareas individuales que se necesita para llegar al objetivo. Es decir, primero se detecta cuál es la acción que probablemente se está llevando a cabo y después se intenta inferir cuál es la secuencia de pasos más probable.

Para utilizar los HMM con este enfoque es necesario que la tarea sea una tarea repetitiva y cada tarea tiene que estar compuesta por un número pequeño de subtareas. Si se usaran muchas clases, la capacidad de generalización del clasificador HMM se reduce demasiado (Theodoridis & Koutroumbas, 1999). Además, como se ha citado previamente, si la secuencia de tareas a realizar es muy larga o existen varias posibilidades para solucionar la tarea, el uso de HMM para este objetivo no resulta práctico. O dicho de otra manera, este enfoque no es apropiado para tareas pobremente definidas.

- **Dominio de la cirugía**

Los problemas que se han citado en el dominio de las operaciones de maquinaria son trasladables al dominio de la cirugía. En este ámbito muchas veces los parámetros para valorar las destrezas de un futuro cirujano no suelen ser evidentes, y por ello se tiende a desarrollar soluciones específicas para cada tipo de operación quirúrgica. Existen algunos sistemas que hacen un análisis de bajo nivel de las posiciones, fuerzas y tiempos grabados durante los entrenamientos en simuladores y sistemas de teleoperación (Cotin et al., 2002; Rosen et al., 2002; Vemer et al., 2003; Yamauchi et al., 2002).

Murphy et al. (2003) hacen una evaluación más exhaustiva de las tareas dinámicas en el ámbito de la cirugía mediante clasificadores HMM. Entrenan cada HMM de forma separada para cada movimiento basándose en los gestos extraídos del estado dinámico del sistema. Estos movimientos constituyen una red de HMMs. El movimiento más significativo, en cada instante de tiempo, es reconocido evaluando la probabilidad de la secuencia de gestos para cada modelo de movimientos. La evaluación de la destreza se basa en el número total de movimientos para realizar la tarea y el porcentaje de tiempo utilizado para cada movimiento. En (Rosen et al., 2003) y (Rosen et al., 2001) las tareas se modelan definiendo secuencias de subtareas que se modelan mediante HMMs. Las fuerzas y torques aplicadas por las herramientas son medidas en tres dimensiones y los estados de los HMMs están definidos para que coincidan con las tareas que un cirujano ejecuta durante una operación. La evaluación de las destrezas se hace en base a la distancia estadística al modelo del experto, teniendo en cuenta tiempos de ejecución de las tareas, la frecuencia de las tareas ejecutadas y la medición de fuerzas y torques durante las tareas.

- **Dominios de ejercicio físico**

Los SIIAA que se presentan en este apartado son los que más se centran en la parte física de la tarea procedimental entre los SIIAA que se han descrito hasta el momento. Estos SIIAA suelen estar equipados con diferentes tipos de sistemas de captura basados en cámaras (con marcadores o sin marcadores), sistemas inerciales o magnéticos, etc. (Medved, 2001). Estos tipos de SIIAA normalmente utilizan técnicas de reconocimiento de acciones para identificar la actividad del estudiante, las cuales siguen tres fases principales (Weinland et al., 2011): extracción de puntos característicos (*feature extraction*), segmentación de la acción y el aprendizaje y clasificación de la acción. Sin embargo, dependiendo del tipo de acciones que se quieran reconocer (puntuales o cortas, periódicas y compuestas), se utilizan distintas técnicas de reconocimiento de acciones (p.j. Unzueta, et al., 2008). De hecho, la mayoría de las técnicas de reconocimiento de acciones dependen de las propiedades del dominio donde se ejecutan las acciones. Esto es una gran desventaja, ya que si el tipo de acción que se quiere entrenar se modifica, el

rendimiento de las técnicas estadísticas utilizadas para la clasificación puede empeorar mucho.

Existen algunos trabajos que van más allá del simple reconocimiento de acciones. RVT (Reactive Virtual Trainer)(Ruttkay & Welbergen, 2008; Ruttkay & Zwiers, 2006) es un agente virtual inteligente que es capaz de representar ejercicios físicos llevados a cabo por una persona, monitorizar a los estudiantes y proporcionar feedback a diferentes niveles. Este sistema se centra sobre todo en el aspecto psicológico que debe mostrar el avatar, como generar feedback empático o inferir sobre el estado físico y emocional del estudiante. RVT está diseñado solo para el entrenamiento de ejercicios repetitivos, como por ejemplo ejercicios de fitness o psicoterapia. Para ello modelan el movimiento del agente virtual especificando tres parámetros del movimiento: el *tempo*, el cual indica el ritmo a seguir en cada repetición, la amplitud del movimiento y el esfuerzo, medido a través de la aceleración del movimiento. Asimismo, el RVT necesita que se le indiquen las trayectorias que deben seguir las articulaciones del cuerpo utilizando lo que denominan "movimientos básicos". Estos movimientos básicos están compuestos de al menos dos posturas claves (el inicio y final de una pose) y también permiten especificar poses intermedias. Otra manera de definir el camino que tienen que seguir los marcadores es a través de splines Hermite.

Aunque la definición del movimiento que debe llevar a cabo el agente virtual es elaborada, a la hora de evaluar las destrezas físicas del alumno no utilizan toda esa información para corregirle: solo tienen en cuenta el *tempo* o la sincronización de sus movimientos respecto al agente virtual. Basándose en las trayectorias de las articulaciones definidas en el RVT (extremidades), el sistema normaliza los datos capturados por el sistema de captura (posiciones de los marcadores) y genera una *localización permitida* para cada marcador. De esta manera, se comparan las posiciones actuales y las permitidas generando un diagnóstico de tipo "correcto", "despacio", "rápido" e "incorrecto".

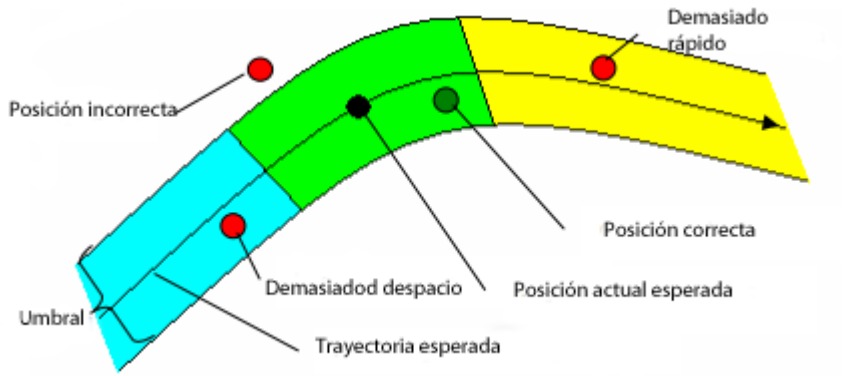


Figura 4: Supuesta trayectoria de un punto del cuerpo, con posiciones rápidas, lentas, correctas e incorrectas.

Existen otros SIIAA específicos para el entrenamiento de baile. En algunos de estos sistemas el entrenamiento se centra en que el estudiante debe imitar los movimientos de una entidad virtual (Davcev et al., 2003; Hachimura et al., 2004; Soga et al., 2009). Otros sistemas de entrenamiento de danza hacen un análisis más profundo del movimiento del estudiante utilizando la técnica de *motion matching*. El objetivo de esta técnica es emparejar movimientos de una plantilla con los movimientos capturados. Hachimura (Hachimura et al., 2005) propone el uso de LMA (Laban Movement Analysis) para el análisis de los movimientos. Este análisis incluye la energía cinética, espacio, tiempo y forma de movimiento como características a analizar. El espacio muestra la dirección del movimiento utilizando como centro de coordenadas el centro de la pelvis. El tiempo expresa la aceleración de cada parte del cuerpo y la forma del movimiento indica la postura global del cuerpo durante un movimiento. Estas características son extraídas cuando los movimientos son comparados y las diferencias entre los movimientos pueden ser computados en cada instante de tiempo. Este método puede emparejar posturas globalmente pero no puede definir errores locales.

Otros autores utilizan los ángulos entre los segmentos que unen las articulacione a la hora de emparejar los movimientos y detectar errores locales o específicos. Qian et al.(Qian et al., 2004) hacen el emparejamiento de movimientos midiendo la distancia de Mahalanobis entre dos series de ángulos de articulaciones, mientras que Kwon et al.

(Kwon & Gross, 2005) lo hacen a través de la distancia euclídea. El SIIAA creado por Chan y Leung. (Chan & Leung, 2011) es capaz de corregir errores en posturas y de sincronización mediante la muestra de dos avatares virtuales diferentes a los estudiantes. Uno de ellos está formado por cilindros y muestra el movimiento que lleva a cabo el estudiante. Dependiendo de la corrección del movimiento, el cilindro correspondiente se muestra de un color que oscila desde el blanco hasta el rojo. A su vez, el otro avatar hace demostraciones al usuario con los movimientos correctos. El cálculo del nivel de corrección de las posturas se hace mediante la distancia euclídea entre las posiciones de las articulaciones reales y la postura de la plantilla. El sistema solo da el nivel de corrección de los movimientos, por lo que tanto el entrenador como el estudiante deben analizar las grabaciones de los movimientos ejecutados para deducir la manera de mejorar la realización de los movimientos.

Duan et al. (Duan et al., 2008) ofrecen otra visión diferente para extraer y transferir destrezas físicas a un estudiante. Clasifican la destreza humana en otros dos tipos de destreza diferentes: destreza de control y destreza de movimiento. La destreza de control se refiere al proceso de toma decisiones para elegir los movimientos adecuados en diferentes circunstancias. Por otro lado, la destreza de movimiento mide la habilidad que una persona tiene para moverse una vez que se sabe qué movimiento se debe ejecutar. Duan et al. ponen como ejemplo la evaluación de un pase de futbol para verificar su trabajo. Utilizan 6 cámaras y el software BhViewer (www.eca-robotics.com) para transformar el movimiento en datos angulares de las articulaciones y de esta manera crean un avatar para reproducir los movimientos llevados a cabo por el estudiante.

Para validar las destrezas de control del estudiante proponen un circuito con dos pilares y una portería, siendo el objetivo sortear los dos pilares con el balón y chutar a portería. Para medir la destreza de control tienen en cuenta el número de toques que se le dan al balón y la velocidad del balón en cada toque, teniendo en cuenta si el control ha sido *suave* o *fuerte*. En cuanto a la destreza del movimiento, tienen en

cuenta la amplitud de los movimientos y la aceleración de las articulaciones. El análisis de la tarea se lleva a cabo a posteriori.

1.3.4 Conclusión

En este apartado se ha visto como los SIIAA ofrecen menos asistencia cuanto más se acerquen a la evaluación de tareas que son básicamente físicas, como por ejemplo la danza (ver Figura 5). Para empezar, existe una amplia gama de métodos para tutorizar en los SAA que se basan en enseñar destrezas intelectuales. En lo referente a los SAA que se centran en asistir tanto destrezas intelectuales como físicas, no centran mucha atención en la parte física de la tarea. Sin embargo, los paradigmas que se utilizan en estos SAA podrían ser perfectamente aplicables a la tutorización de tareas físicas.

Actualmente no existen muchos SAA que asistan a los estudiantes en el entrenamiento de destrezas físicas. Aunque las bases para analizar los movimientos estén asentadas, existe una falta de metodologías de manipulación de datos para poder hacer un diagnóstico automático de las destrezas físicas de una persona. Generalmente, los métodos existentes hacen una comparación entre una plantilla predefinida de movimientos y los movimientos de los alumnos, dando como resultado de diagnóstico un porcentaje de similitud entre ambos movimientos. Además, en muchos de estos tipos de SAA se ha apreciado que el aprendizaje se basa en aprender distintos movimientos a través de la imitación a un avatar virtual y las correcciones que se ofrecen son poco detalladas. Dadas estas razones, el análisis de destrezas físicas aún se sigue haciendo a posteriori y de una manera manual en actividades como el deporte profesional.

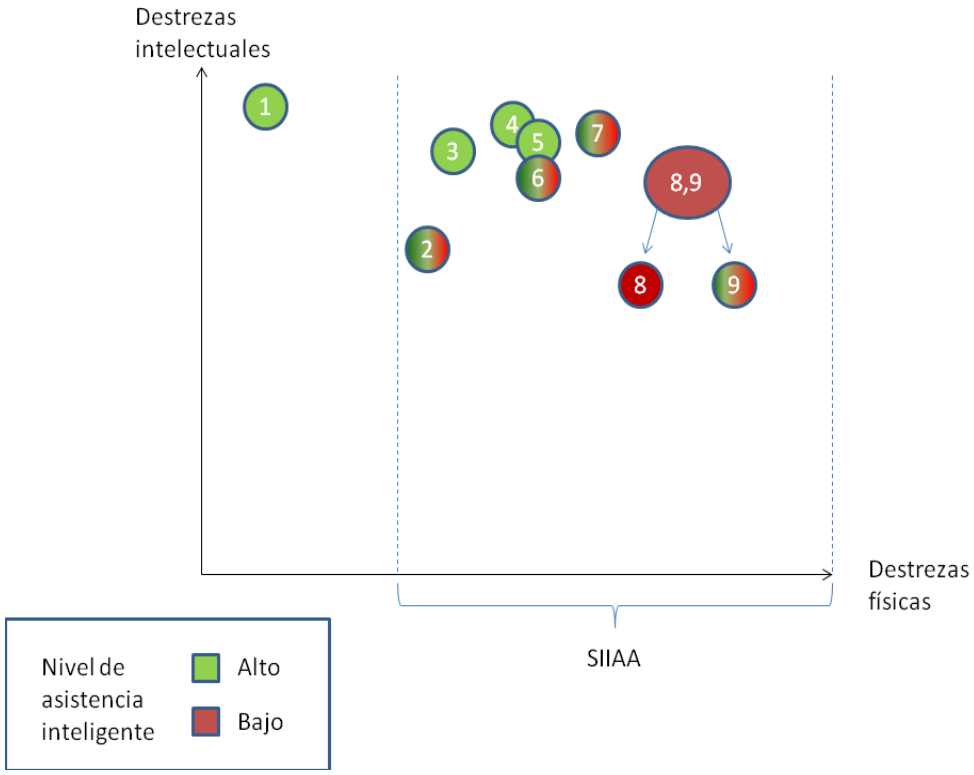


Figura 5: Clasificación de los SIIAA basada en el dominio en el que ofrecen asistencia y el nivel de asistencia que ofrecen. 1) SQL-Tutor, EER-TUTOR, KERMIT, Andes 2) XAIDA 3)STEVE-VIVIDS 4)IFT 5) CandarmTutor 6) Partly Automated Mobile Working Machines 7)SAA dominio de la cirugia 8)RTV 9)Sistemas que utilizan emparejamiento de movimientos.

1.4 MODELADO DE DESTREZAS FÍSICAS

El diseño de la interacción basada en movimiento a través de la tecnología es un área emergente que se centra en el cuerpo humano y en su capacidad de movimiento. La variedad de dispositivos de entrada que existen para capturar movimientos humanos (por ejemplo: MoCap con marcadores o sin marcadores, incluidos los de bajo coste como el sensor Microsoft Kinect) permiten tipos de interacción más complejas que las que pueden ofrecer un joystick, un ratón o un teclado. Teniendo en cuenta estos avances, las notaciones para definir el movimiento han adquirido un mayor interés. Entre las notaciones que existen, la notación

de Laban o labanotación (Loke et al., 2005) es la que ha cobrado más importancia. La labanotación es un sistema para analizar y grabar el movimiento y fue diseñado originalmente por Rudolf Laban en la década de los 20. Utiliza una notación simbólica parecida a la musical y cada símbolo representa un cambio; es decir, un movimiento. El *pentagrama* (Figura 6) está dividido en diferentes columnas, donde cada columna representa una parte del cuerpo: piernas, cuerpo, brazos y cabeza. Los movimientos se entienden como pasos o posturas, donde un paso se refiere a un movimiento que requiere una transferencia de peso mientras que la postura no la requiere.

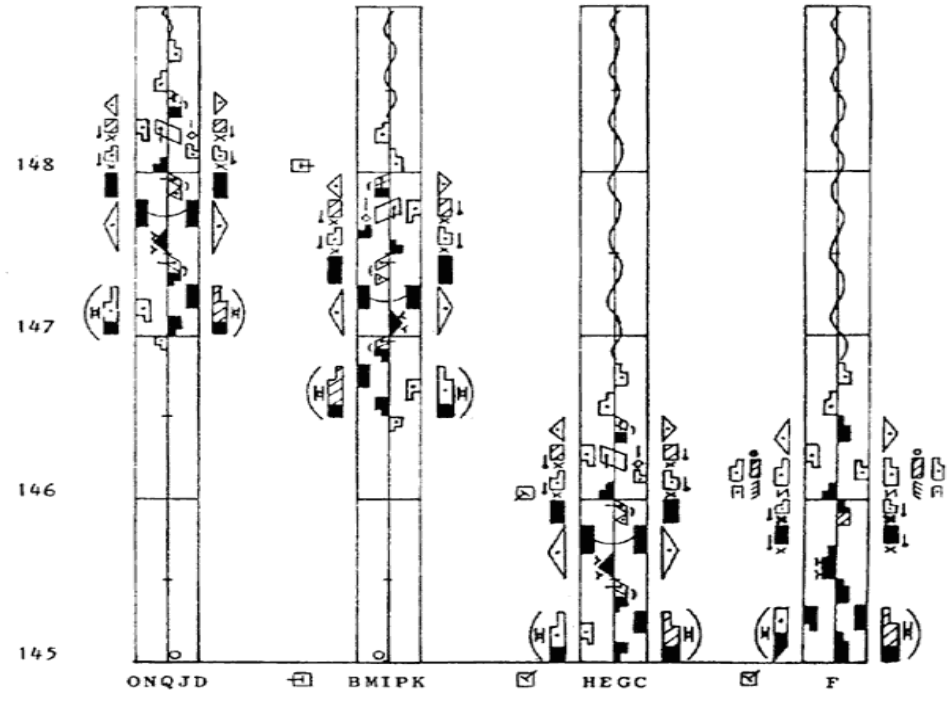


Figura 6: Ejemplo de labanotación

En la labanotación básicamente se modelan tres conceptos básicos: la descripción del motivo, la descripción del esfuerzo y forma, y la descripción estructurada. En la descripción del motivo sólo se indica el motivo o la característica más importante del movimiento. El esfuerzo indica la dinámica cuantitativamente mediante distintos símbolos de esfuerzo (Figura 7).

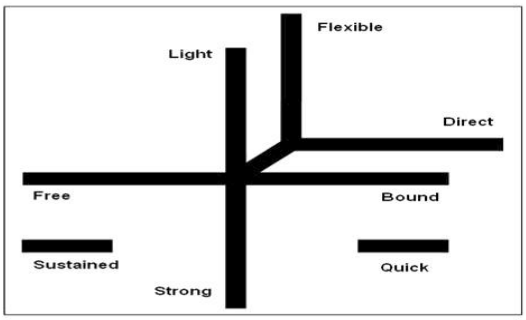


Figura 7: Gráfica de esfuerzo de Laban

La forma indica la manera en la que el cuerpo cambia de postura durante el movimiento (forma de los gestos y maneras de cambiar posturas). Por último, la característica estructural hace la descripción más detallada del movimiento mediante términos claramente definidos y medibles: el cuerpo y sus partes, espacio (dirección, nivel, distancia, grado de movimiento), tiempo y dinámica (calidad o textura, por ejemplo: fuerte, elástico, acentuado, con énfasis). Normalmente, la descripción estructural suele hacer referencia a la dirección donde el cuerpo o sus partes se tienen que dirigir. La Figura 8 muestra la representación que describen los movimientos en el plano horizontal y plano frontal.

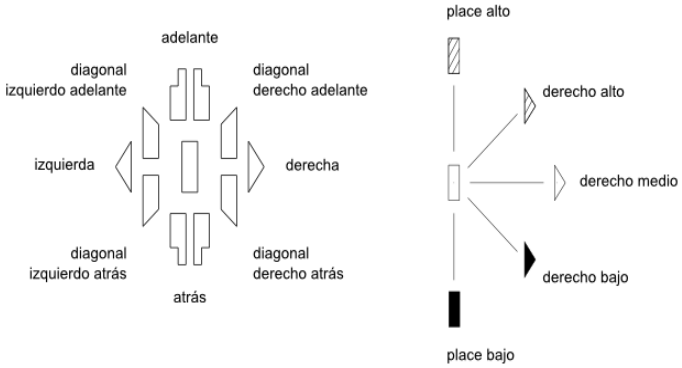


Figura 8: Los símbolos que describen las direcciones en el plano horizontal y vertical

Dada la importancia que ha adquirido esta notación, actualmente existen muchas herramientas de autor para la labanotación como las herramientas para Windows LabaNotator (http://labanotator.kepha.si),

Calaban (http://web.bham.ac.uk/calaban/frame.html) o las herramientas para Mac LabanWriter y LabanReader (www.dance.osu.edu). Normalmente, la labanotación se utiliza para dar vida a agentes virtuales animados (por ejemplo (Davcev et al., 2003), pero la forma de representar el movimiento usado en esta notación también ha sido utilizada en varios trabajos para analizar y evaluar los movimientos de los estudiantes. Por ejemplo, Hachimura (Hachimura et al., 2005) utiliza las características de la notación de Laban para enseñar la danza japonesa Furi mediante su sistema. RVT también se basa en los parámetros de Laban de esfuerzo y forma para modelar el movimiento.

Otro método más reciente para modelar movimientos es el lenguaje de scripting GESTYLE (Noot & Ruttkay, 2003). Este lenguaje surgió para definir los movimientos de agentes virtuales conversacionales y permite definir expresiones de movimientos complejos –basados en movimientos básicos- utilizando composiciones tanto paralelas como secuenciales de una forma embebida. Por ejemplo, dar una palma con los brazos encima de la cabeza se definiría como una ejecución paralela de dos movimientos (una con cada brazo) encima de la cabeza en el plano corporal, cada uno basado en dos movimientos básicos, uno para el movimiento del brazo y otro para la orientación de la palma. Este lenguaje de scripting se ha utilizado en RVT (Ruttkay & Zwiers, 2006), en el cual solo se modelan ejercicios repetitivos. La definición de una palmada se definiría de esta forma en GESTYLE:

```
<DefGest Name="LeftHand_ClapAboveHead"> <PAR>
<UseGest Name="LeftHand_Lift" /> <UseGest
Name="LeftWrist_Rot"/></PAR> </DefGest><DefGest
Name="RightHand_ClapAboveHead"> <PAR> <UseGest
Name="RightHand_Lift" /> <UseGest
Name="RightWrist_Rot"/></PAR> </DefGest>
<DefGest Name="ClapAboveHead"> <PAR> <UseGest
Name=" LeftHand_ClapAboveHead" /> <UseGest Name="
RightHand_ClapAboveHead"/> </PAR> </DefGest>
```

Los movimientos compuestos y los ejercicios más largos pueden ser definidos como nuevos elementos de más alto nivel. En cuanto a las relaciones temporales, se pueden utilizar duraciones del tipo "wait", lo que permite sincronizar distintos movimientos. En los ejercicios repetitivos, el GESTYLE permite definir el tempo o el ritmo, pero

también pueden expresar parámetros como el esfuerzo o la amplitud de los movimientos, parámetros similares a la labanotación. Este lenguaje es completo y permite definir movimientos con bastante precisión, pero es un lenguaje demasiado técnico para los autores a los que está destinada la herramienta, como fisioterapeutas o entrenadores personales (Ruttkay & Zwiers, 2006).

1.5 HERRAMIENTAS DE AUTOR PARA LA ADQUISICIÓN DE CONOCIMIENTO PARA SIIAA

Generar información para los módulos de un STI, por ejemplo, crear dominios completos, modelar estudiantes con la precisión suficiente y desarrollar estrategias de aprendizaje es una tarea compleja (Munro et al., 1997). Por esta razón, surgió la necesidad de crear herramientas de autor para que la creación fuera eficiente. En la actualidad, existen dos clasificaciones importantes sobre las herramientas de autor (HA), una de Murray (Murray, 2003) y otra actualización de la misma elaborada por Nkambou y Bourdeau (Nkambou & Bourdeau, 2010). Ambos coinciden en que existe una falta de métodos y herramientas estándar con la que se pueda facilitar el proceso de autoría de los STI. Igualmente, clasifican los usuarios interesados en crear tutores inteligentes en dos tipos: usuarios con y sin conocimientos de programación. Mientras que los primeros pueden usar plantillas, reutilizar código o utilizar APIs sin que requieran una interfaz intuitiva, los segundos necesitan herramientas que sean fáciles de utilizar y donde puedan reflejar su modelo mental del sistema que quieren construir. Por esta razón, clasifican las herramientas de autor en dos tipos: herramientas basadas en ventanas (shells) y herramientas de autor para los no-programadores.

Otro tipo de clasificación es la que se hace en base a los tipos de sistemas que generan las HA, donde se encuentran las HA basadas en simulaciones. En este último grupo se encontrarían los SIIAA. Una característica importante de estas HA es la capacidad para crear tutores y el modelo de simulación que soportan. Igualmente, estas herramientas de autor permiten llevar a cabo diferentes tipos de *adquisición de conocimiento.* Por ejemplo, algunas herramientas se centran en asistir en el modelado del dominio conceptual procedimental. En este último caso

la manera de adquirir el conocimiento puede variar, lo cual puede afectar a la integración de la HA en algunos entornos. Por otra parte, una HA debe tener en cuenta la problemática relacionada con del dominio de la tarea donde se lleva a cabo el entrenamiento. Por ejemplo, hay que tener en cuenta si el SIIAA es capaz de soportar dominios pobremente definidos y con un alto grado de incertidumbre. Además, la representación del conocimiento y las variables que forman parte de la simulación determinan la relación entre la HA y el sistema interactivo. Si la representación de las tareas o conceptos de aprendizaje es muy dependiente del sistema interactivo, el manejo del dominio puede resultar muy difícil. A esto hay que añadirle que la probabilidad de generar un dominio incompleto crece considerablemente. Por último, otro punto importante son las herramientas o métodos que una HA ofrece para hacer el seguimiento sobre el progreso y las destrezas que van adquiriendo los estudiantes.

Existen múltiples HA que tienen en cuenta las características que se han mencionado en las anteriores líneas. Algunas de ellas tienen escasa capacidad de simulación. Uno de estos sistemas es XAIDA (Hsieh et al., 1999; Wenzel et al., 1998), una herramienta para el desarrollo de SAA para el entrenamiento de mantenimiento de maquinaria (Figura 9). Esta HA tiene en cuenta las características del dispositivo a mantener, su teoría operacional, los procedimientos de operación, mantenimiento y solución de averías. Su sistema de adquisición del conocimiento del dominio permite adquirir conocimiento en diferentes dominios, pero en la práctica solo se ha utilizado para enseñar información fáctica sobre el sistema: partes de sistemas mecánicos, eléctricos o hidráulicos, o los pasos a seguir en un procedimiento.

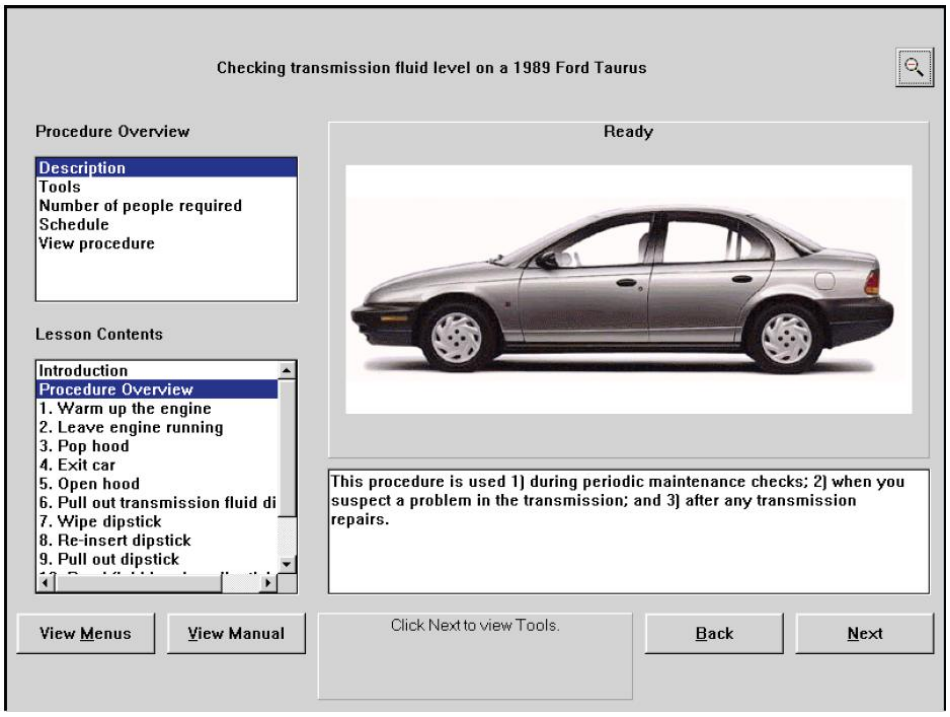


Figura 9: Captura de la HA XAIDA

SIMQUEST (Van Joolingen et al., 1997), RIDES y VIVIDS (Munro, A., Johnson, M.C., Pizzini, Q.A., Surmon, D.S., Towne, D.M., Wogulis, 1997) son otros sistemas de autor para crear simulaciones embebidas en un entorno instruccional. En comparación a RIDES y VIVIDS, SIMQUEST soporta un modelo cognitivo más detallado porque modela el dominio como una entidad autónoma y describe las relaciones conceptuales a un nivel abstracto. Más aún, no muestra la interfaz como parte del modelo, sino como una vista del mismo. Por último, RIDES y VIVIDS están dirigidos a modelar y asistir en el procedimiento a enseñar, mientras que el objetivo de SIMQUEST se centra en asistir los procesos de descubrimiento del aprendizaje. Estas tres herramientas tienen una fuerte orientación práctica, pero su capacidad de simulación es limitada.

Diligent (Angros et al., 2002) es la HA diseñada para adquirir el conocimiento procedimental necesario para STEVE. En Diligent el experto demuestra cómo se debe llevar a cabo un procedimiento y el agente aprende las reglas observando los pasos ejecutados por el experto

durante el procedimiento. Diligent ejecuta internamente el procedimiento para intentar entenderlo e intenta llegar al mismo resultado que el instructor. En ciertos puntos, el instructor puede hacer modificaciones o correcciones a la descripción procedimental generada por Diligent. Además, la autoría en Diligent incluye la adquisición de conocimiento lingüístico, lo que permite a STEVE hablar sobre el dominio. Todo este proceso reduce considerablemente el esfuerzo que hay que hacer en la adquisición de conocimiento, pero esta herramienta sufre algunas limitaciones a la hora de integrarlo con simuladores en algunos entornos. Diligent no es adecuado para dominios en los que las consecuencias de las acciones no son claramente observables. De modo que no es adecuado para entornos pobremente definidos.

Demonstr8 (Blessing, 1997), aunque no esté basado en simulaciones, comparte algunas características con Diligent. Al igual que en esta última HA, el autor demuestra el procedimiento para resolver la tarea resolviendo tareas de ejemplo y el sistema deduce automáticamente cuáles son los pasos a seguir para resolver esa tarea. Una vez que se ha generado el procedimiento, todas las reglas de producción generadas son mostradas al autor para que las revise y/o modifique. Para ello, es indispensable que el autor tenga conocimientos de model-tracing. Adicionalmente, los autores pueden asignar metas y destrezas a las reglas para que el sistema pueda distinguir acciones particulares en diferentes contextos. El sistema también permite establecer niveles de criterio en diferentes destrezas y el modelo del estudiante es actualizado dependiendo de las destrezas adquiridas. Como última característica, cabe señalar que los SAA creados con Demonstr8 permiten monitorizar el progreso de los estudiantes en cada paso que se haga al resolver la tarea, y también es capaz de dar feedback cuando se requiere o cuando sea solicitado. Sin embargo, la principal limitación de Demostr8 reside en que sólo es válido para dominios aritméticos.

Flexitrainer (Ramachandran et al., 2004) es una HA creada por StottlerHenke Associates. Se ha utilizado para crear diversos tutores, sobre los que destaca IFT (sección 1.3.3.2). Esta HA se utiliza para definir una base de datos con la red semántica que representa el dominio del

STI y también para definir el dominio procedimental. La herramienta está compuesta de cuatro editores diferentes: el editor de principios (*task principle editor*) de la tarea, el editor de ejercicios, el editor del modelo de estudiante y el editor del comportamiento del tutor. El editor de principios recoge las definiciones sobre el conocimiento que hay que enseñar: las soluciones de las tareas que el STI necesita para hacer un seguimiento del progreso, destrezas y principios del estudiante. En el editor de ejercicios se definen distintas variables: los prerrequisitos que el estudiante debe satisfacer para superar un nivel, variables de simulación que se pueden utilizar para evaluar las ejecuciones de los estudiantes (por ejemplo: altura o velocidad), comportamientos que describen los procedimientos que se deben llevar a cabo, comportamientos que describen cuando empieza o acaba cada procedimiento, los comportamientos de los evaluadores, la lista de todos los eventos a los que el tutor debería de reaccionar etc. En resumen, las tareas son descritas en el editor de ejercicios con máquinas de estados finitos que definen las situaciones a los que los estudiantes deben hacer frente. La Figura 10 muestra la edición de un comportamiento en FlexiTrainer.

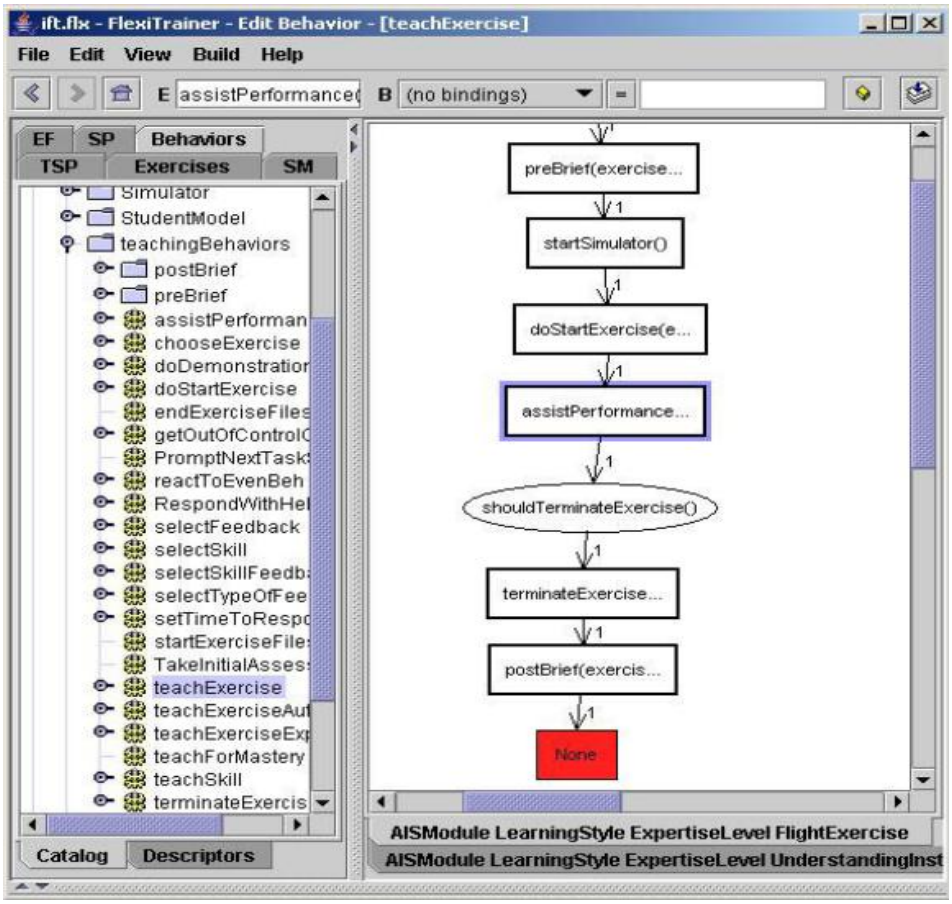


Figura 10: Comportamiento que define la lógica a alto nivel sobre cómo debe actuar el tutor a la hora de enseñar un ejercicio.

En lo referente al editor del modelo de estudiante, se utiliza para definir el modelo de estudiante y el tutor lo actualiza automáticamente según los estudiantes van adquiriendo los distintos objetivos de aprendizaje. Y por último, el editor de comportamientos del tutor se usa para especificar cómo hay que evaluar y enseñar al estudiante. Todas estas herramientas son muy potentes, pero a su vez requieren conocimientos de programación. En la Figura 10 se puede apreciar cómo se crean los diagramas de estados. Para ello, solamente hay que arrastrar los scripts del panel izquierdo de la herramienta a la parte derecha. Pero para poder hacer esto, los scripts deben ser programados previamente. Por otro lado, existe una estrecha relación entre la HA y el simulador. Esto

significa que la información de bajo nivel del simulador tiene que ser especificada, y como esta característica puede complicar la gestión del dominio, se incrementa la probabilidad de generar un dominio incompleto.

Todas las herramientas presentadas hasta el momento son HA de STIs clásicos, pero en los últimos años se han creado herramientas más potentes que muestran buenas prácticas de autoría de STIs. Un esfuerzo importante en este campo es el Cognitive Tutor Authoring Tools (CTAT) (Aleven et al., 2006), diseñado para crear tutores cognitivos y tutores Example-Tracing. Los tutores cognitivos comparan las acciones de los estudiantes respecto al modelo cognitivo definido y tienen la ventaja de que pueden trabajar con múltiples problemas simultáneamente. Con los tutores Example-Tracing (Aleven et al., 2009), el autor demuestra al sistema como se debería resolver un problema particular, por lo que cada solución tiene que ser creada por separado. CTAT contiene herramientas para definir interfaces de estudiante e interfaces para añadir ejemplos correctos e incorrectos para tutores Example-Tracing. La herramienta Behaviour Recorder graba los ejemplos en modo de grafos (*behaviour graph)* y ofrece así una forma para planear, editar y testear los modelos cognitivos. Este grabador permite obtener instantáneas para cada estado de un problema y se puede utilizar para indicar si los resultados generados son correctos. Además, CTAT contiene otra herramienta para asistir en la búsqueda de errores en un modelo cognitivo. Sin embargo, añadir ejemplos para un tutor Example-Tracing puede convertirse en una tarea costosa, sobre todo si el tutor requiere que se le demuestren todas las posibles soluciones para un problema. Para solucionar este problema, Machuda presentó SimStudent (Matsuda et al., 2007), una HA donde el autor demuestra cómo se resuelve una tarea y el sistema generaliza esta demostración, por lo que se reduce significativamente el tiempo que se necesita para desarrollar un Example-Tracing tutor en CTAT.

En el área de los tutores basados en el paradigma de modelado mediante restricciones, la HA más importante es ASPIRE (Mitrovic et al., 2009). Se trata de un sistema de autoría para tareas procedimentales y no procedimentales que está basada en WETAS (Martin & Mitrovic,

2003), una HA de tipo shell. ASPIRE está compuesto de un servidor de autoría y de un servidor de tutorización. El primero asiste al autor en el desarrollo del STI y el segundo implanta los sistemas desarrollados. Como se trata de una herramienta independiente del dominio, ASPIRE ofrece un entorno de trabajo donde se pueden generar ontologías del dominio y definir la estructura del problema y la solución general de la estructura basándose en ellas. Ya que ASPIRE está orientado a tutores basados en restricciones, ofrece utilidades para generar la sintaxis y semántica de las restricciones que se necesitan, basándose en la estructura de la solución y las ontologías definidas previamente. Teniendo en cuenta esta característica de ASPIRE, en un principio los autores no necesitan tener ningún conocimiento de IA o programación, ya que las restricciones se generan automáticamente. ASPIRE oculta las restricciones para los autores novatos y no programadores, pero permite tanto crear como modificar restricciones a otros desarrolladores. ASPIRE se ha utilizado satisfactoriamente en la creación de varios tutores, siendo Capital Investment Tutor (Mitrovic et al., 2008) uno de los más importantes. ASTUS (Lebeau et al., 2009) es un framework para crear STIs independientes del dominio y para dominios bien definidos. Este framework se basa en el clásico arquetipo de STI de cuatro módulos, e incluye importantes facilidades para la representación del conocimiento. Sus principales objetivos son minimizar el esfuerzo requerido en la creación de tutores model-tracing y ofrecer un framework modular para la creación de STIs. Los tutores generados con ASTUS tienen la capacidad de reconocer la composición de los errores, generar feedback inmediato y feedback para los errores específicos, generar sugerencias sobre los pasos a seguir y demostrar cómo hay que resolver un paso. También da cierto control a los estudiantes para elegir el tipo de ayuda que necesitan. Estás características de los tutores son posibles gracias a un modelado complejo de la tarea que se hace mediante grafos de soluciones, una mezcla entre los grafos que se usan en CTAT y Andes. Sin embargo, se necesitan conocimientos de programación para definir la información necesaria para algunos módulos, por ejemplo para definir las sugerencias que le van a dar al alumno. Además sus autores admiten que puede ser difícil crear tutores para dominios pobremente definidos con ASTUS (Paquette et al., 2010).

En lo que se refiere a SIIAA para el entrenamiento de destrezas físicas, normalmente suelen optar por solucionar problemas específicos a la hora de crear un sistema entrenamiento, por lo que no suelen ofrecer HA y los que lo ofrecen suelen tener una capacidad limitada de autoría. Una excepción es la herramienta de autor ofrecida por RVT. Permite definir la trayectoria de los movimientos de cada parte del cuerpo mediante la especificación de diferentes puntos que definan el trazado de la trayectoria. Esta trayectoria puede ser dada en base a splines Hermite. De esta manera, se pueden crear movimientos básicos que son reutilizables en movimientos más complejos. Como se ha citado anteriormente, la parametrización de los movimientos se hace a través del lenguaje GESTYLE, y aunque se trate de un lenguaje de scripting de alto nivel sigue siendo muy técnico para personas ajenas a la programación. Por otro lado, la HA de RVT permite establecer el tempo del ejercicio mediante distintas señales acústicas y a su vez permite especificar la secuencia de movimientos a seguir en un ejercicio.

En los trabajos de Ruttkay (Ruttkay & Welbergen, 2008; Ruttkay & Zwiers, 2006) se cita que RVT posee una herramienta de edición "simple", pero no se muestran las interfaces ni se puede deducir hasta qué nivel llega la autoría de ejercicios.

En resumen, se puede decir que ocurre un problema similar al analizado en la sección de entrenamiento para las tareas procedimentales. Las HA que se centran en dar asistencia para las tareas procedimentales intelectuales contienen una gran variedad de funcionalidades para crear los modelos de tareas. Sin embargo, esto no es aplicable a las HA para SIIAA para el entrenamiento de destrezas físicas. Existen pocas HA en este ámbito, y las que existen suelen optar por solucionar problemas específicos del sistema de entrenamiento y no tratan los problemas relacionados con la creación de tareas para sistemas de tutorización.

CAPÍTULO 2

SIIAA BASADOS EN LA PLATAFORMA OLYMPUS

OLYMPUS (Aguirre et al., 2012) es una plataforma genérica para crear SIIAA para el entrenamiento de tareas procedimentales. El objetivo general de los SIIAA generados con OLYMPUS es transferir el conocimiento de un experto a un alumno, por lo que la plataforma abarca tanto el proceso de aprendizaje como el de adquisición de conocimiento mediante la captura de la actividad de expertos. Para lograr estas capacidades, la plataforma OLYMPUS ofrece varias herramientas y componentes reutilizables, con los que se facilita la adquisición de conocimiento, se monitoriza la actividad de los estudiantes y se modela el feedback que el SIIAA tiene que dar al estudiante. OLYMPUS se sustenta en el framework ULISES (Lozano-Rodero, 2009), el cual está compuesto por una serie de módulos genéricos que se encargan de comunicar un Sistema Interactivo con un Sistema de Ayuda al Aprendizaje. El framework ejecuta un proceso con tres pasos principales: primero se **observa** lo que está pasando en el entorno virtual; a continuación se **interpretan** las observaciones capturadas, con lo que se obtiene una representación a alto nivel de la actividad del estudiante; y por último, la actividad del alumno es **diagnosticada** para detectar errores y posibles lagunas de conocimiento.

Los resultados de diagnóstico generados se publican para que los puedan utilizar otros módulos de la plataforma (u otros externos) para generar feedback, informar al instructor u otras funcionalidades educativas.

En este capítulo se hace una descripción general de los componentes que forman OLYMPUS y los procesos que definen su uso, teniendo en cuenta el proceso que va desde la adquisición de conocimiento hasta la asistencia en tiempo real a los estudiantes. Posteriormente se describe el framework ULISES, y después se describe el resto de herramientas que forman parte de la plataforma OLYMPUS.

2.1 DESCRIPCIÓN GENERAL

OLYMPUS modela de forma genérica el proceso para crear un SIIAA capaz de diagnosticar, dar un feedback adecuado y monitorizar la actividad de los alumnos entre otras funciones. No fuerza a utilizar una estrategia de aprendizaje específica, pero parte de la base de que el SAA tiene que tomar decisiones basadas en el nivel de conocimiento del alumno durante cada fase del aprendizaje. Ese nivel de conocimiento puede ser inferido por las acciones que lleva a cabo el estudiante mientras resuelve tareas (Bauer, 2002; Shute, 2004), de lo cual se encarga el proceso de diagnóstico cognitivo. Este planteamiento fundamenta la arquitectura de OLYMPUS (Figura 11) que, aunque permite integrar módulos independientes, parte de inicio con una serie de componentes y herramientas independientes del dominio:

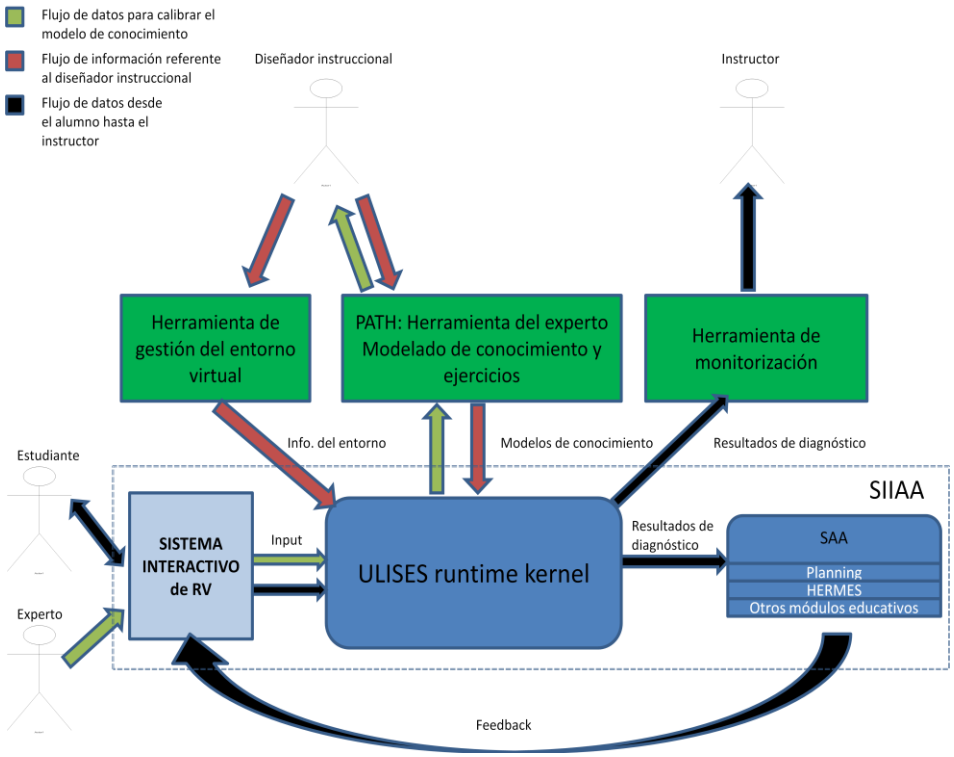


Figura 11: Arquitectura de la plataforma OLYMPUS

- Sistema Interactivo (SI): Sistema, por lo general basado en RV-RM, que los estudiantes utilizan para practicar tareas procedimentales. El SI debe permitir que las interacciones entre el estudiante y el entorno virtual sean capturadas para que se pueda llevar a cabo la comunicación con el runtime kernel de ULISES.
- Runtime kernel de ULISES: módulo de ULISES que transforma los datos capturados del SI en resultados de diagnóstico válidos para los SAA.
- Sistema de Ayuda al Aprendizaje (SAA): conjunto de módulos que proporcionan las funcionalidades educativas del SIIAA basándose en los resultados de diagnóstico generados por ULISES.
- Herramienta de gestión del entorno virtual (sección 2.3.1): Herramienta para modelar el conocimiento asociado a los objetos observables del escenario virtual donde se van a ejecutar las tareas. Por ejemplo, en un entorno de simulación de conducción, con esta

herramienta se pueden definir elementos como los carriles, las propiedades del vehículo que se conduce (velocidad, aceleración etc.), señales de tráfico, etc.

- Herramienta de autor PATH (sección 2.3.2): Herramienta que permite la creación de los modelos necesarios para el framework ULISES basándose en la actividad de los expertos en el SI.
- Subsistema de feedback HERMES (sección 2.3.3): Se trata de un SAA integrado en OLYMPUS cuyo objetivo es ofrecer un feedback adaptable a las características del alumno y del contexto de la tarea (Lopez-Garate, Lozano-Rodero, Matey, 2008).
- Herramienta de monitorización (sección 2.3.4): Herramienta que permite monitorizar gráficamente y en tiempo real la actividad del estudiante.

OLYMPUS define el flujo de uso de sus herramientas considerando la interacción entre los siguientes roles que pueden tomar parte en el proceso educativo:

- Estudiante: usuario que usa el SIIAA para aprender.
- Diseñador instruccional: profesional de la educación que diseña las tareas de entrenamiento. No tiene por qué ser un profesional experto del dominio que se pretende enseñar.
- Experto: es el profesional del dominio que el diseñador instruccional captura para definir los modelos de conocimiento que se necesitan en OLYMPUS.
- Instructor: es la persona que supervisa a los estudiantes durante las sesiones de entrenamiento.

En la Figura 11 se muestran con flechas de distinto color los flujos de información que se generan entre roles en función del proceso que se esté llevando a cabo:

1. Se comienza por diseñar las tareas que los instructores van a proponer a los estudiantes. Para ello, los diseñadores instruccionales deben definir la información observable de los escenarios virtuales mediante la herramienta de gestión del entorno virtual.

2. Los diseñadores instruccionales definen los ejercicios para los estudiantes mediante la herramienta de autor PATH. Con esta herramienta los diseñadores instruccionales pueden capturar la actividad de los expertos mientras llevan a cabo una tarea en el SI. Es decir, los modelos de conocimiento necesarios para ULISES se modelan basándose en el conocimiento del experto.
3. ULISES genera resultados de diagnóstico que pueden ser utilizados por los SAA, por ejemplo, por HERMES. Estos resultados de diagnóstico también se envían a la Herramienta de Monitorización para que los instructores hagan el seguimiento de la actividad del alumno en las sesiones de entrenamiento.

2.2 EL FRAMEWORK ULISES

El framework ULISES (Lozano-Rodero, 2009) es la base en la que se sustenta OLYMPUS. La metodología que define este framework para comunicar un SI con un SAA se inspira en los mismos pasos que lleva a cabo un instructor real inconscientemente cuando tutoriza una sesión de entrenamiento. En primer lugar, observa qué está pasando en el entorno, luego intenta interpretar las acciones del estudiante y después las diagnostica. A partir de este punto, su estrategia educativa decidirá qué hacer con esa información: hacer una demostración, ofrecer asistencia etc. En definitiva, lo que considere más conveniente para lograr que los alumnos aprendan.

Esta aproximación se traslada al framework ULISES mediante tres subsistemas que representan ese mismo proceso (Figura 12). Estos subsistemas forman el runtime kernel de ULISES. Cada uno de ellos recibe los datos generados por el subsistema anterior en tiempo real, los analiza y los transforma con el fin de generar la información educativa adecuada para los componentes del SAA que la requieran. Los subsistemas se fundamentan en un metamodelo estructurado en tres niveles lógicos: observación, interpretación y diagnóstico que se describen en el siguiente apartado.

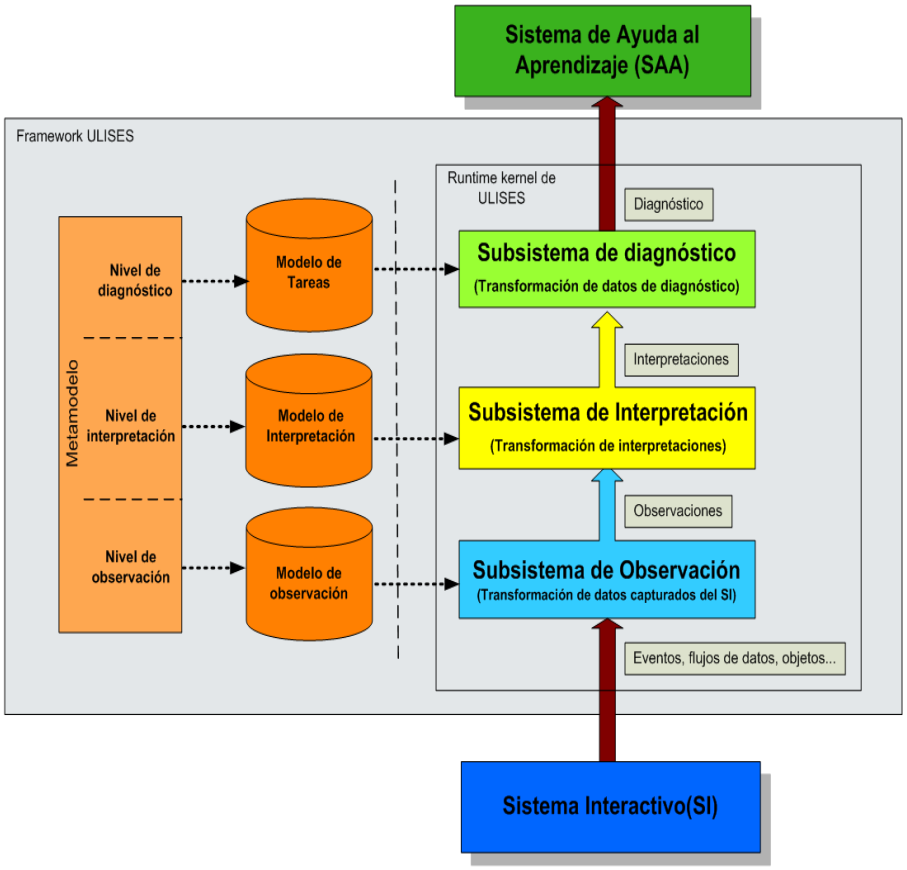


Figura 12: Arquitectura general del framework ULISES

2.2.1 Metamodelo de ULISES

El metamodelo de ULISES está definido por tres niveles de abstracción que describen genéricamente los elementos que tienen que ser particularizados para poder describir la información necesaria para cada subsistema de ULISES (Figura 13). Estos elementos describen cómo son los modelos que hay que generar para cada subsistema al construir un SIIAA, es decir, indican:

- Cómo describir lo que hay que observar en el SI
- Cómo describir lo que hay que interpretar y cómo se interpreta
- Cómo describir las tareas que se quieren diagnosticar

Estos elementos son particularizados para el dominio concreto, y esa particularización da lugar al modelo específico de Observación, Interpretación y Tareas. De esta manera, se consigue una manera automática de comunicar un SI con un SAA. La comunicación con el SI queda predeterminada por el Modelo de Observación, mientras que el de Tareas establece la comunicación con el SAA.

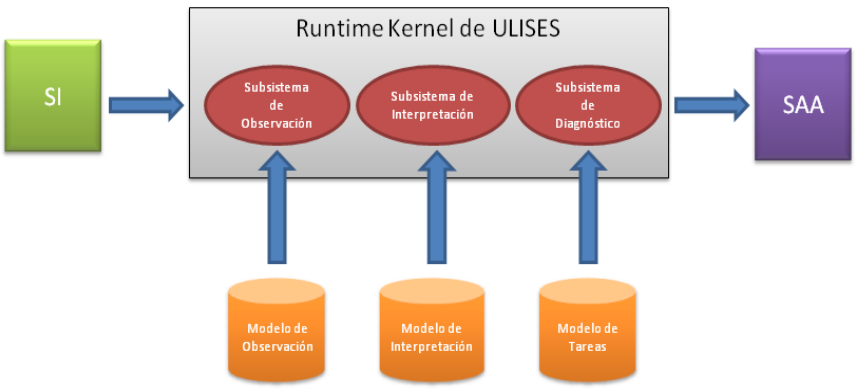


Figura 13: Composición del runtime kernel de ULISES

2.2.2 Runtime kernel de ULISES

El runtime kernel de ULISES está diseñado siguiendo una arquitectura multiagente. Los sistemas multiagente han sido empleados como un enfoque alternativo en la construcción de STIs, por ejemplo, (Vicari & Gluz, 2007) (R. Nkambou & Kabanza, 2001) (Vassileva et al., 2003). ULISES emplea el estándar de interoperabilidad FIPA para comunicar a sus agentes (Bellifemine et al., 2001). La utilización de este estándar hace posible la comunicación entre los tres subsistemas de ULISES, las herramientas de OLYMPUS y otras aplicaciones externas. Por lo general, dicha comunicación se basa en los protocolos estándar de suscripción, solicitud y consulta, con los que los agentes ofrecen y piden la información que necesitan sólo cuando es requerida.

En las siguientes líneas se describe la relación entre los agentes de ULISES y el resto de agentes que forman un SIIAA (Figura 14):

- Agentes de entrada-salida: Permiten que el runtime kernel se comunique con interfaces de usuario u otros servicios, por ejemplo, con PATH, la herramienta de autor para adquisición de conocimiento de OLYMPUS.
- Agentes de escucha: Se ocupan de la comunicación entre el SI y el agente observador. Pueden existir varios agentes de escucha en caso de que el SI tenga una arquitectura distribuida.
- Agente observador: Este agente encapsula el subsistema de observación. Se encarga de recoger los datos de simulación provenientes de los agentes de escucha y de transformar esos datos en observaciones para los subsistemas de interpretación y diagnóstico.
- Agente de interpretación: Encapsula el subsistema de interpretación. Recibe las observaciones a las que se ha suscrito y determina qué actividad está realizando el alumno utilizando el Modelo de Interpretación.
- Agente de diagnóstico: Encapsula el subsistema de diagnóstico, encargado de coordinar el diagnóstico de la actividad que recibe desde el agente de interpretación. El proceso de diagnóstico puede ser llevado a cabo internamente (con componentes de diagnóstico específicos integrados en el subsistema de diagnóstico) o externamente (por ejemplo con DETECTive (Ferrero, 2004)). En ambos casos, el agente de diagnóstico se encarga de comunicar los resultados de diagnóstico a los componentes o herramientas que lo requieran.

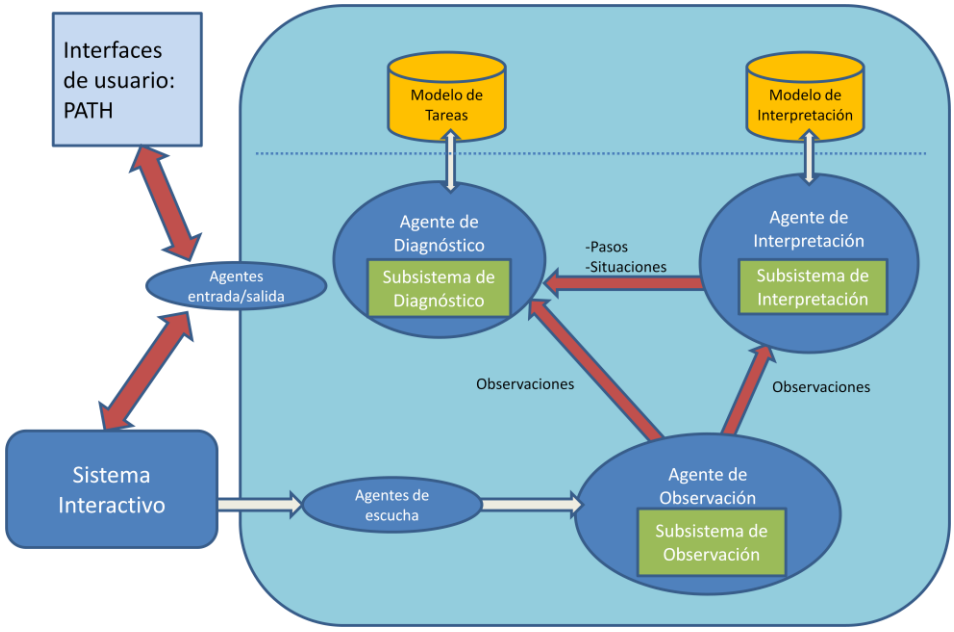


Figura 14: Agentes que forman el runtime kernel de ULISES

2.3 HERRAMIENTAS QUE COMPONEN LA PLATAFORMA OLYMPUS

Actualmente, OLYMPUS cuenta con una herramienta de gestión del entorno virtual, el sistema de autoría PATH, una herramienta de monitorización de la actividad en el SIIAA, un generador de informes de evaluación y el subsistema de feedback HERMES.

2.3.1 Herramienta de gestión del entorno virtual

El objetivo de esta herramienta es generar interactivamente el conocimiento asociado a los objetos 3D de los escenarios virtuales de entrenamiento. Es decir, hay que crear la información necesaria para que el subsistema de Observación de ULISES pueda generar observaciones sobre los elementos del entorno. Esta información podría definirse ad hoc en cada Sistema Interactivo, pero mediante esta herramienta se agiliza el proceso. Por ejemplo, en un sistema de entrenamiento para la conducción, es necesario definir conocimiento sobre los carriles, semáforos, vehículos que forman parte del escenario, el propio vehículo

etc. Para conseguir este objetivo, se sigue el siguiente proceso que se detalla a continuación: (1) se define la ontología del escenario, (2) se crea la malla de conocimiento y (3) se valida el conocimiento observable.

1. Para empezar, se especifican los elementos del escenario que ULISES necesitará observar. Estos objetos son descritos en la *ontología de la escena*, la cual está compuesta por clases y sus correspondientes propiedades. Por ejemplo, si se necesitan observar los carriles de las carreteras, se creará una clase Carril con sus correspondientes propiedades: velocidad máxima del carril, tipo de carril etc.
2. En el segundo paso, se identifica la representación física de los objetos en el entorno 3D. La definición se lleva a cabo asignando geometrías simplificadas, denominadas como *mallas de conocimiento*, a los objetos 3D que se visualizarán en la simulación. Además, se definen observadores en el entorno virtual. Su función es capturar mediante sensores virtuales las mallas de conocimiento del entorno durante la sesión de entrenamiento. Mediante esta representación el observador captura los objetos 3D observables buscando colisiones de sus sensores con las mallas de conocimiento en lugar de tener que hacerlo con los objetos 3D reales, lo que implicaría mayor complejidad y tiempo de cómputo.
3. Por último, la herramienta ofrece la posibilidad de validar si los elementos generados son consistentes y si se capturan los objetos deseados y sus propiedades sin tener que esperar a probarlo en la simulación del estudiante.

Figura 15: Diferentes pantallas de la herramienta de gestión del entorno virtual

2.3.2 PATH: La herramienta de autor para adquisición de conocimiento procedimental

El objetivo principal de esta herramienta es dar expresión al conocimiento del experto sin que se requieran conocimientos de programación. La creación de los modelos necesarios para el framework ULISES es una tarea costosa y requiere una comprensión profunda de ULISES. Para minimizar el coste, OLYMPUS ofrece la herramienta de autor PATH, que permite al diseñador instruccional crear el Modelo de Tareas y el Modelo de Interpretación basándose en el conocimiento del experto. Para ello, ofrece funcionalidades de captura y análisis de la actividad del experto cuando éste resuelve una tarea en el SI.

PATH y la metodología subyacente es una de las aportaciones principales de este proyecto de tesis, por lo que se ahondará más sobre esta herramienta en el capítulo 4 de esta memoria.

2.3.3 Subsistema de feedback HERMES

HERMES es un sistema de feedback independiente del dominio cuyo comportamiento es adaptable a las características del alumno y del contexto de la tarea (Lopez-Garate, 2011). Además, es capaz de aprovechar las capacidades multimodales de los Sistemas Interactivos. HERMES selecciona el feedback basándose en los resultados de diagnóstico generados por ULISES. Este proceso de selección de feedback infiere las acciones más importantes entre las que se están llevando a cabo por el estudiante en un momento concreto. Es más, también determina cuál es el mensaje más apropiado para el alumno teniendo en cuenta sus características. El algoritmo de selección de feedback tiene en cuenta tanto la asimilación de los mensajes como los factores educativos. Por último, este algoritmo hace posible que el comportamiento de HERMES sea configurado para las características concretas de un dominio y a las preferencias de un experto.

2.3.4 Herramientas de monitorización y generación de informes

Visualizar adecuadamente los resultados de diagnóstico es tan importante como generar un diagnóstico preciso. Por esa razón, OLYMPUS ofrece dos recursos visuales: una herramienta de monitorización de la actividad y un generador de informes de evaluación. La herramienta de monitorización permite a los instructores monitorizar en tiempo real la actividad de los estudiantes en el SI, estructurándola en forma de situaciones y pasos definidos en el Modelo de Interpretación (los detalles de la estructura de la actividad se encuentran en la sección 3.4). Asimismo, la herramienta muestra visualmente los resultados de diagnóstico de cada paso (Figura 16).

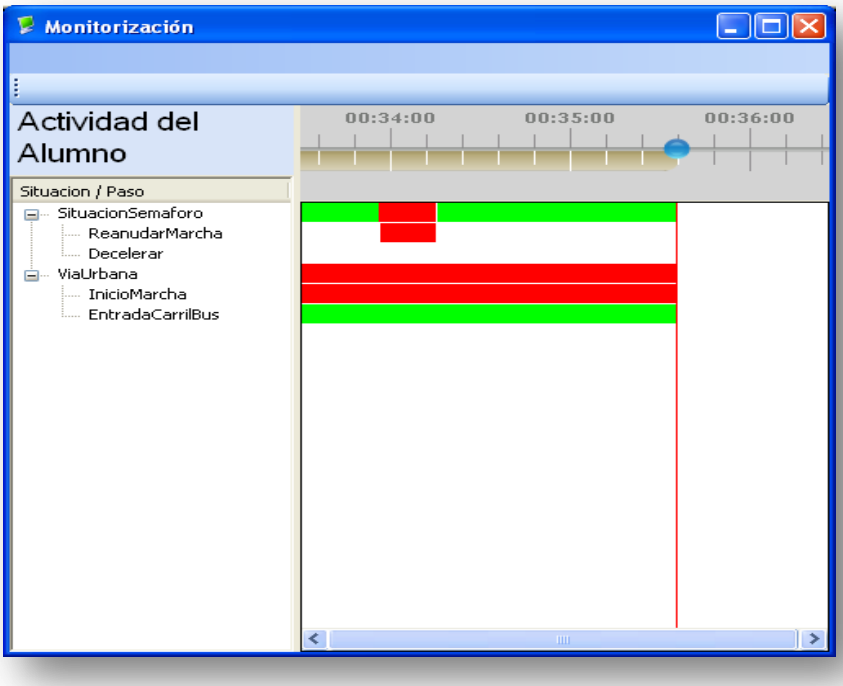


Figura 16: Herramienta de monitorización

Además de ofrecer resultados de diagnóstico en tiempo real, cuando un ejercicio se da por acabado se muestran los resultados de diagnóstico más detalladamente en un informe de evaluación (Figura 17). Este informe permite consultar las notas que se han obtenido en casa paso y situación, así como el grado de adquisición de los objetivos de aprendizaje que se asignan en el Modelo de Tareas. Todos los resultados de los ejercicios son guardados en una BBDD para que el instructor pueda hacer un seguimiento del progreso del estudiante a lo largo del tiempo.

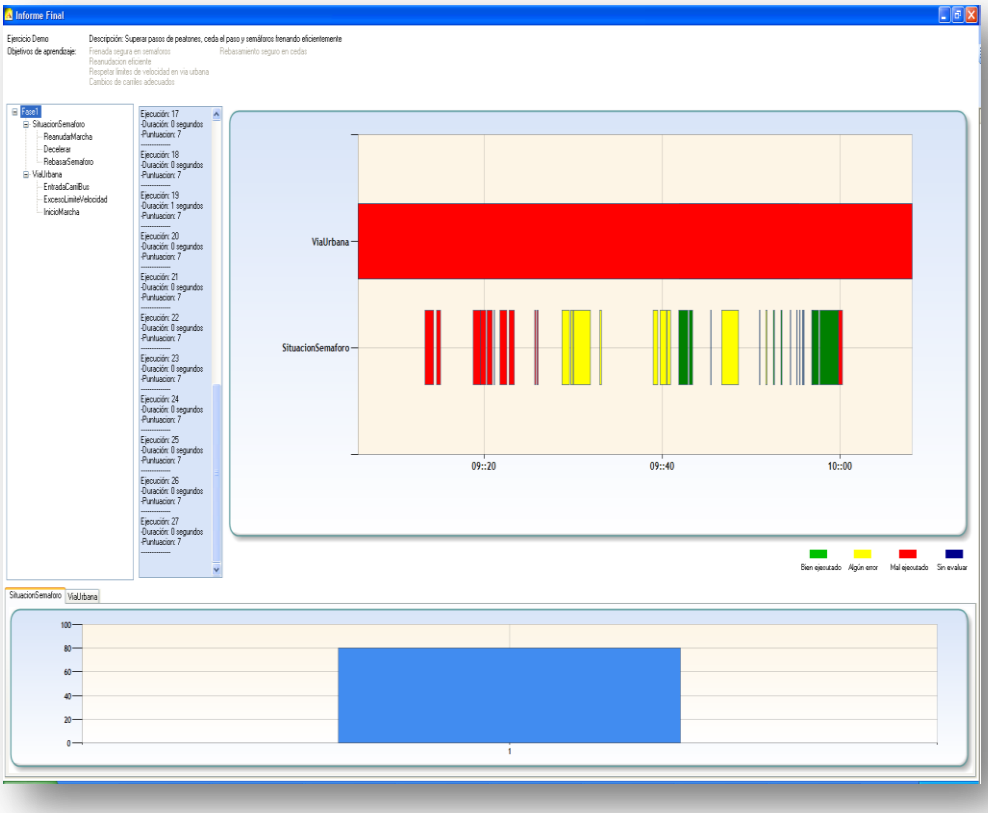


Figura 17: Ejemplo de informe de evaluación

2.4 DISCUSIÓN GENERAL

El framework ULISES ofrece una metodología que es independiente del SI y de las funcionalidades del SAA que se quieran integrar, lo que hace posible que dicha metodología sea independiente del dominio de aprendizaje. De la misma manera, el subsistema de diagnostico de ULISES permite la integración de distintas técnicas o herramientas externas de diagnóstico, lo que lo convierte en adecuado tanto para dominios bien definidos como para dominios pobremente estructurados.

Sin embargo, esta genericidad se ve limitada a la hora de tratar con tareas que requieren gestionar la información teniendo en cuenta la incertidumbre. Este problema es aplicable a las tareas para el aprendizaje de destrezas físicas. En este ámbito, la incertidumbre se presenta en el mismo momento que un alumno empieza a hacer un movimiento. En ese instante, un instructor no sería capaz de discernir el tipo de movimiento que el alumno está intentando realizar, por lo que debería esperar un periodo de tiempo hasta poder descartar o afirmar qué tipo de movimiento puede estar haciendo el alumno.

El framework ULISES no fue diseñado con esta finalidad y solo procesa la información que captura en cada ciclo de ejecución. Dada esta problemática, se vio la necesidad de dotar a este framework con la capacidad de tratar con la incertidumbre, para poder así extender la utilización de OLYMPUS a otros dominios más abiertos como el entrenamiento de destrezas físicas.

CAPÍTULO 3

EVALUACIÓN DE DESTREZAS MOTORAS CON ULISES

En este capítulo se presentan las aportaciones de este proyecto de tesis al framework ULISES para dotarle de capacidades de evaluación de destrezas motoras y, en general, de evaluación bajo incertidumbre.

El capítulo comienza con el análisis de la problemática abordada. A continuación, se presenta la descripción general de la solución propuesta y en las secciones siguientes se detallan el diseño e implementación llevados a cabo en cada nivel de ULISES, incluyendo los elementos definidos, el funcionamiento de cada subsistema y su forma de gestionar la incertidumbre. Posteriormente, se explica la aplicación del framework para el aprendizaje de destrezas motoras en el dominio del tenis. El capítulo finaliza con la validación del framework y con las conclusiones obtenidas sobre la evaluación de las destrezas motoras con ULISES.

3.1 ANÁLISIS DEL PROBLEMA Y ALCANCE

En última instancia, el objetivo de un SIIAA para el entrenamiento de destrezas físicas es conseguir que el alumno transfiera con éxito las

habilidades adquiridas durante el entrenamiento asistido por computador al entorno real. Para ello, se deben diseñar tareas que permitan observar la destreza del alumno a medida que intenta dominar cada objetivo de aprendizaje del dominio, lo cual es algo complicado cuando intervienen destrezas motoras. Como se ha citado en el capítulo de Análisis de Antecedentes, se suelen utilizar diferentes medidas objetivas para valorar la destreza de los alumnos. Por ejemplo, CanadarmTutor (sección 1.3.3.3) evalúa si el alumno es capaz de seguir una secuencia de pasos correcta moviendo el brazo robótico. En los entornos de maquinaria parcialmente automatizada se estima la eficiencia del alumno al resolver la tarea, la complejidad de la secuencia de pasos del procedimiento, la habilidad para tomar decisiones y la dificultad de la tarea. En cambio, en el dominio de la cirugía se suelen medir las posiciones, fuerzas y tiempos empleados al llevar a cabo los movimientos del instrumental médico.

En general, dependiendo del dominio se evalúan las destrezas utilizando medidas particularizadas, pero ni siquiera dentro del mismo dominio suele existir un consenso sobre cuales son las medidas objetivas más adecuadas. Como indica Tervo (2010), la principal causa de este problema radica en que existe una falta de métodos de manipulación de datos que revelen los distintos niveles de destreza. Teniendo esto en cuenta, en este trabajo se han definido cuatro características del movimiento que sirven para llevar a cabo un diagnóstico de las destrezas motoras de un estudiante: coordinación, mantenimiento de posturas, seguimiento de trayectorias y conocimiento del procedimiento o secuencia de movimientos. En este capítulo se demuestra que estas cuatro características del movimiento se pueden usar para evaluar una destreza física y que sirven para revelar distintos grados de destreza en un alumno.

Por otro lado, cualesquiera que sean las características del movimiento que se tengan en cuenta para evaluar la destreza física del alumno, el SIIAA tendrá que transformarlas en mensajes comprensibles para el estudiante o el instructor. La transformación de flujos de datos computacionales en variables lingüísticas ha sido objeto de estudio desde los inicios de los computadores (Zadeh, 1975), y en el caso de los

SIIAA sigue siendo de interés. Por ejemplo, cuando un experto va a corregir a un estudiante utiliza descripciones cualitativas (Panjkota et al., 2009) como: "tienes que extender más el brazo", "no has girado el hombro lo suficientemente rápido", etc. Teniendo en cuenta esta característica del entrenamiento, la transformación de las variables cuantitativas del movimiento a un dominio cualitativo implica que el movimiento sea descompuesto en partes (Zadeh, 1997) y que la representación sea intuitiva o comprensible desde el punto de vista didáctico. Asimismo, además de tener en cuenta la incertidumbre, la transformación del movimiento en variables lingüísticas también debe tener en cuenta que los términos son *difusos*: no existen claros límites establecidos que diferencien un movimiento rápido de uno lento, o un movimiento recto de uno diagonal.

La *incertidumbre* también afecta a la representación del movimiento: cuando un estudiante empieza a hacer un movimiento, es probable que el tutor no lo identifique hasta que ocurra un cambio de dirección o hasta que transcurra un periodo de tiempo. Por ejemplo, si el estudiante empieza a mover su mano hacia arriba, el tutor no sabrá si el estudiante está intentando mover el brazo diagonalmente o en línea recta. La Figura 18 muestra cuatro trayectorias diferentes que comienzan en la misma dirección. Sin embargo, hasta que se recorre una parte de la trayectoria no resulta evidente cuál es el movimiento que se está intentando llevar a cabo.

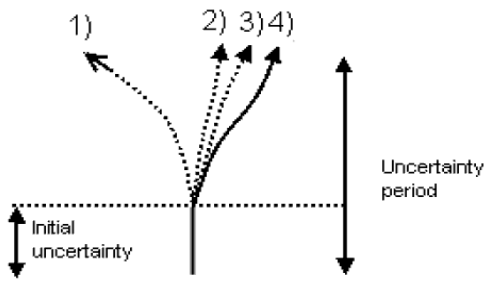


Figura 18: Cuatro trayectorias y su periodo de incertidumbre

Por último, ULISES también debe tener en cuenta la *intencionalidad* de las acciones del alumno: cuando una persona aprende un nuevo movimiento, no suele ejecutar el movimiento exacto que tiene en mente, sino uno imperfecto (Figura 18, trayectorias 2, 3 y 4). Para que el SIIAA pueda corregirle debe interpretar lo que intenta hacer en lugar de lo que ha hecho, lo que añade dificultad al proceso de interpretación.

3.2 DESCRIPCIÓN GENERAL

Con el fin de que ULISES sea capaz de tratar con la problemática definida en el apartado anterior, se han añadido nuevas funcionalidades a los tres niveles que componen el framework. Además, se ha modificado el protocolo de gestión y comunicación de datos entre los tres subsistemas del runtime kernel de ULISES que le permiten mantener la coherencia de datos en tiempo real. A continuación se describen las aportaciones generales en cada nivel para conseguir que ULISES evalúe destrezas motoras:

- **Nivel de Observación**: En este nivel se ha especificado una representación del movimiento que permite al Nivel de Interpretación extraer la semántica de los movimientos. Esta representación permite definir observaciones que están afectadas por la incertidumbre y transformar las variables cuantitativas del movimiento en variables cualitativas válidas para el diagnóstico. Basándose en esta especificación, se ha modificado el Subsistema de Observación para que sea capaz de extraer las observaciones definidas en el Modelo de Observación y se las comunique al Subsistema de Interpretación.
- **Nivel de Interpretación**: El Nivel de Interpretación describe los elementos necesarios para que se puedan identificar las acciones de los alumnos que son interesantes para el diagnóstico a partir de las observaciones modeladas en el Nivel de Observación. Se ha adaptado el algoritmo de interpretación y los elementos asociados a este nivel para permitir la interpretación bajo incertidumbre. Además se tiene en cuenta el problema que conlleva determinar la intencionalidad de los movimientos del alumno.

- **Nivel de Diagnóstico**: Se ha diseñado un algoritmo de diagnóstico capaz de tratar con la incertidumbre generada en los niveles anteriores. Los resultados de diagnóstico que genera son válidos para transformarlos en variables lingüísticas y mantienen las propiedades de genericidad que les permiten ser el nexo de unión entre el Sistema de Ayuda al Aprendizaje y el framework ULISES.

A continuación se presentan estos tres niveles en mayor profundidad.

3.3 OBSERVACIÓN

Observar lo que está pasando en el Sistema Interactivo es el primer paso que hay que dar para poder hacer una evaluación de la actividad del alumno. Para ello, el nivel de Observación contiene los elementos genéricos que permiten especificar cómo hay que transformar un flujo de datos proveniente del SI en entidades significativas (**observaciones)** que describen lo percibido durante un intervalo de tiempo. Estos elementos son particularizados para el dominio concreto del SIIAA, dando lugar al Modelo de Observación. Este modelo se utiliza para abastecer al Subsistema de Observación, el cual es el encargado de extraer observaciones a partir de los datos recibidos del Sistema Interactivo (Figura 19).

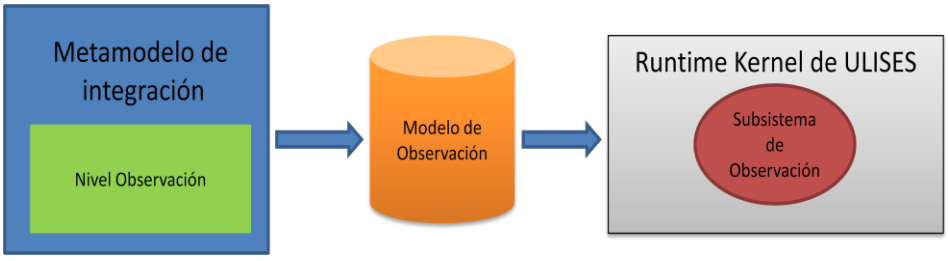


Figura 19: Relación entre Nivel, Modelo y Subsistema de Observación.

En primer lugar se hace una descripción general del Nivel de Observación del metamodelo de integración de ULISES. A continuación se detallan los elementos definidos en el nivel para especificar cómo hay que observar movimientos y finalmente se explican las aportaciones realizadas en el Subsistema de Observación, considerando tanto los

elementos de su modelo dinámico como el proceso de generación de observaciones.

3.3.1 Descripción del Nivel de Observación

Este nivel contiene los elementos necesarios para especificar los hechos interesantes que están ocurriendo en el Sistema Interactivo desde un punto de vista educativo. Estos hechos servirán de primitivas para describir la actividad del estudiante en los demás niveles: interpretación y diagnóstico. Por ejemplo, si en el Nivel de Interpretación se quisiera describir que un tenista va a golpear la pelota, habría que observar el movimiento de la raqueta, el movimiento de la pelota y la distancia entre ambos elementos.

Representar observaciones correctas y completas en este nivel es de suma importancia, ya que los vacíos en el Nivel de Observación pueden llevar a malinterpretaciones en los niveles superiores. Por esta razón, el Nivel de Observación tiene la flexibilidad suficiente para especificar observaciones a partir de datos de distinta naturaleza como eventos, flujos de datos u objetos complejos que contienen cualquier tipo de información. Asimismo, tiene la capacidad de definir observaciones tanto puntuales (ej: el estudiante está accionando un botón) como continuas (ej: el estudiante está moviendo su brazo hacia arriba).

La evaluación de las destrezas físicas parte de la representación del movimiento que se especifica en el Nivel de Observación. Para ello, tal y como se detalla a continuación, el Nivel de Observación permite modelar la observación de *trayectorias* de puntos característicos (articulaciones del cuerpo, el utensilio que se está manejando, etc.) en forma de una secuencia de arcos caracterizados con lógica difusa (Figura 20):

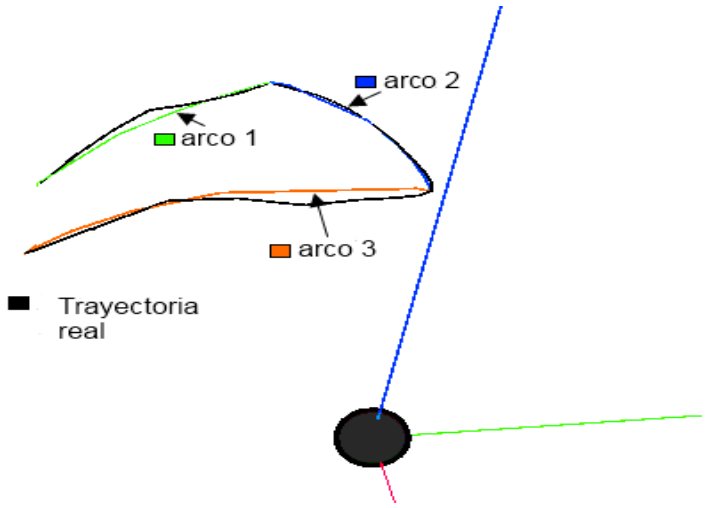


Figura 20: Ejemplo que ilustra la representación mediante arcos

Para llevar a cabo esta representación, en este trabajo se asume que el SIIAA es capaz de capturar y hacer el seguimiento de los puntos característicos que interesan para hacer la evaluación. La *trayectoria* que forma el movimiento de un punto característico se segmenta en partes más pequeñas y cada segmento se representa de forma simplificada mediante un arco de circunferencia que lo aproxima. La decisión de utilizar arcos se ha fundamentado desde el punto de vista educativo. Se considera que cada parte del movimiento tiene que ser representada de una manera suficientemente simple para que las correcciones sobre cada una se puedan expresar verbalmente. A la vez que simple, debe contener la información necesaria para poder obtener datos significativos semánticamente. Los arcos cumplen dichas características: asociando propiedades al arco como su dirección, velocidad, aceleración y la concavidad o convexidad, es posible corregir un movimiento mediante el uso de expresiones verbales. En cambio, representaciones de curvas más complejas, como las splines, tendrían que dividirse para poder referirse verbalmente a partes específicas de la curva.

En este trabajo las trayectorias se segmentan cuando se detecta un cambio de dirección "significativo", siendo significativos los cambios que superen un umbral configurable en función de la precisión de movimientos que haya que diagnosticar. Cuanto menor sea el umbral

más segmentos se generarán y los arcos reproducirán la trayectoria original con mayor exactitud, pero a costa de complicar el Modelo de Interpretación. Cada segmento se representa mediante una observación individual y, dado que cada segmento está expresado con un arco, el resultado generado por el Subsistema de Observación será un conjunto de arcos relacionados temporalmente.

Además de los arcos, es necesario que la representación del movimiento tenga en cuenta el problema de la intencionalidad. Para ello, se ha incorporado en este nivel la *especificación de observaciones difusas*, con las que se le indica al Subsistema de Observación que las observaciones (internamente arcos) que genere, tienen que incluir la representación difusa de su dirección de movimiento. Se definen 26 direcciones principales de movimiento separadas por 45º (Figura 21), de forma similar a la representación de Labanotación (Loke et al., 2005):

$$Classes = \{P = (i,j,k) \mid P \neq (0,0,0)\ \forall\ i,j,k\ \in I\} \quad I = \{1,0,-1\}$$

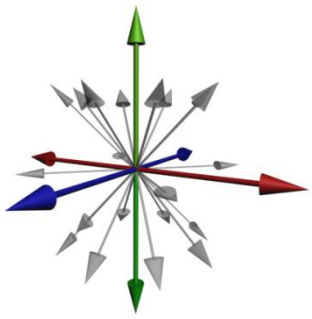


Figura 21: Las 26 clases que se usan para representar el movimiento en una dirección.

Dada esta especificación, cuando el Subsistema de Observación lleva a cabo la segmentación, determina la dirección de movimiento más aproximada del segmento que lo representa y determina el grado de pertenencia de la observación a cada una de las 26 clases. De esta manera, el Subsistema de Interpretación puede relajar o tensar el criterio por el que vaya a considerar que el alumno está realizando un movimiento esperado, es decir, controla hasta qué punto es capaz de

comprender la intencionalidad del alumno. Tal y como se detalla en la sección 3.4, estableciendo restricciones sobre el grado de pertenencia de una observación difusa a cada clase se consigue detectar situaciones como esta: "El movimiento que ha hecho el alumno se parece mucho a este otro movimiento, puede que esté intentando hacer este movimiento".

En resumen, combinando la lógica difusa con arcos a la hora de representar sus direcciones, se establecen las bases para obtener una descripción cualitativa del movimiento.

3.3.2 Elementos del Nivel de Observación

La *especificación de observación* (EspObservacion) es el elemento básico de este nivel. Se emplea para representar de forma genérica cómo debe tratar el Subsistema de Observación los datos de entrada del Sistema Interactivo para determinar que una observación está sucediendo *durante un intervalo de tiempo*. Es decir, las especificaciones de observaciones permiten modelar como una variable lingüística un conjunto de datos de entrada del SI que individualmente no tienen significado. Por ejemplo, en un simulador de conducción, la distancia del vehículo del alumno a los carriles contiguos permite observar la invasión del carril, lo que a su vez permitirá interpretar que se está ejecutando el procedimiento de cambio de carril.

Teniendo esto en cuenta, el elemento EspObservacion contiene los atributos que especifican qué tiene que hacer el runtime kernel de ULISES para extraer observaciones a partir de los datos de entrada obtenidos del SI. Éstos son los atributos que componen una EspObservacion (Figura 22):

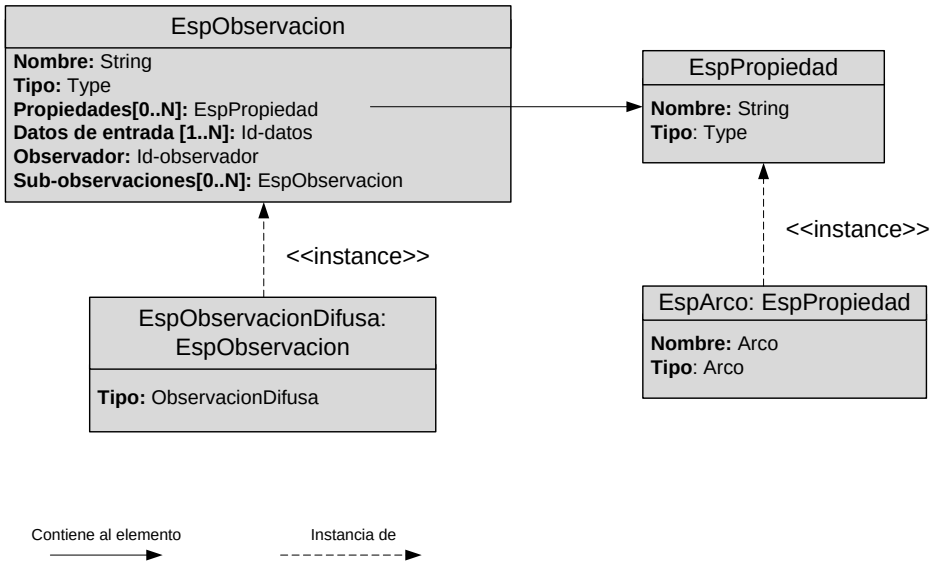


Figura 22: Especificación de una Observación, de una Observación Difusa y una Propiedad

- *Nombre:* Texto que identifica la observación.
- *Tipo:* Especifica el tipo de observación. En caso de tratarse del tipo *ObservacionDifusa*, el runtime kernel de ULISES sabrá que tiene que crear una instancia de observación de tipo difuso y calcular el grado de pertenencia del movimiento a cada una de las 26 direcciones principales.
- *Propiedades:* Lista de especificaciones de propiedades (EspPropiedad) asociadas a la observación. Las observaciones pueden incluir opcionalmente una serie de propiedades que describan o cuantifiquen la observación. Por ejemplo, cuando un alumno lleva a cabo un movimiento, el arco observado podrá incluir propiedades como la velocidad o la aceleración del movimiento en cada instante.
- *Datos de entrada:* Lista de identificadores de los datos provenientes del Sistema Interactivo que los agentes de escucha enviarán al Subsistema de Observación. A partir de estos datos el runtime kernel de ULISES deberá generar la observación. Puede identificar un evento, un flujo de datos como las posiciones de una trayectoria, un objeto estructurado etc.

- *Observador:* Identificador del módulo Observador que el Subsistema de Observación tiene que ejecutar para determinar si una observación está ocurriendo en un instante determinado a partir de los *Datos de entrada*. El observador también calcula el valor de las propiedades de cada observación.
- *Sub-observaciones:* Lista de EspObservacion que solo es posible extraer si la observación que la contiene está ocurriendo. Por ejemplo, en un partido de tenis a dobles, no tiene sentido observar si el compañero está golpeando la pelota a no ser que el compañero esté en el campo de visión del jugador.

Las *Propiedades* que se enumeran dentro de una EspObservación se especifican mediante el elemento EspPropiedad. Este elemento consta de dos atributos:

- *Nombre:* Indica el nombre de la propiedad.
- *Tipo*: Especifica el tipo de dato (número, texto, objeto, arco, etc.) y sirve para que el Subsistema de Observación sepa calcular y asignar el valor de las propiedades a partir de los datos de entrada y para restringir las operaciones entre propiedades que el runtime kernel vaya a realizar.

Las observaciones que representen un segmento de un movimiento, tendrán que definir una propiedad de tipo de arco, que el Subsistema de Observación utilizará para calcular los atributos del elemento Arco definido en la sección 3.3.3.2.

La creación del Modelo de Observación consiste en crear todas las instancias de EspObservacion con sus correspondientes EspPropiedad que sean necesarias para interpretar y diagnosticar la actividad del alumno. El Modelo de Observación se almacena de forma persistente en formato XML. Por ejemplo:

```
<ObservationModel Name="KinectOM.xml">
  <EspObservations>
   <EspObservation Name="LeftArmFlexion" Spy-
Id="KinectSpy" Description="">
       <InputData>
       <IData Id="Skeleton" />
       <IData Id="TrackingState" />
       <IData Id="FinKinect" />
       <IData Id="Train" />
```

```
        <IData Id="Save" />
        </InputData>
        <Properties>
        <EspProperty Name="Angle" type="float" />
        </Properties>
        <SpyData>
        <EvaluationClassName
Name="KinectObservators.KinectObserver" Descrip-
tion="..."/>
        </SpyData>
        <SubObservations></SubObservations>
   </EspObservation>
  </EspObservations>
</ObservationModel>
```

El Modelo de Interpretación es la entrada al Subsistema de Observación que se detalla en el siguiente apartado.

3.3.3 Subsistema de Observación

El Subsistema de Observación es el encargado de extraer las observaciones especificadas en el Modelo de Observación a partir de los datos que le llegan en tiempo real desde el Sistema Interactivo. En esta sección se describe el subsistema en tres apartados. En primer lugar, se describe la arquitectura de este subsistema. Seguidamente se detalla el Modelo Dinámico de Observación y por último, el proceso de extracción observaciones y su gestión a lo largo del tiempo.

3.3.3.1 Arquitectura

El Subsistema de Observación está compuesto por el Gestor de Observaciones y los Observadores. El Gestor de Observaciones se encarga de crear y destruir los Observadores basándose en el Modelo de Observación. Las observaciones son generadas por los Observadores especificados en el Modelo y se envían al Gestor de Observaciones para que mantenga la coherencia temporal de las mismas (sección 3.3.3.3). Una vez que se han gestionado las observaciones, los cambios que se producen en ellas son comunicados al resto de agentes a través del Agente de Observación. La siguiente figura ilustra la arquitectura del Subsistema de Observación:

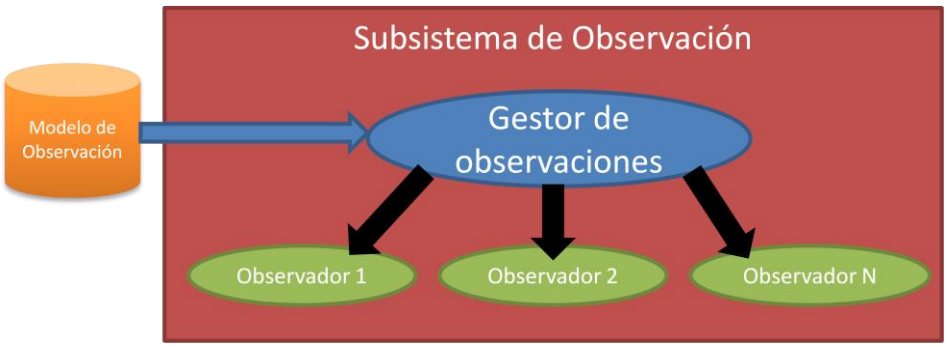


Figura 23: Arquitectura del Subsistema de Observación

3.3.3.2 Modelo Dinámico de Observación bajo incertidumbre

El Modelo Dinámico de Observación contiene los elementos Observación y ObservacionDifusa, con los que se representan las observaciones que se van extrayendo durante una sesión de entrenamiento. A medida que se van extrayendo, estas observaciones son utilizadas por los Subsistemas de Interpretación y Diagnóstico. Sus definiciones son:

- Observación: Intervalo de tiempo compuesto de dos subintervalos consecutivos. El subintervalo inicial representa que el Observador ha confirmado que el hecho especificado en la EspObservacion se está produciendo, mientras que el segundo está bajo incertidumbre, es decir, que el Observador no ha sido capaz de concluir si el hecho especificado se está produciendo realmente o no.
- ObservacionDifusa: Observación que representa un segmento de movimiento descrito de forma difusa y que, por tanto, contiene el grado de pertenencia de la observación a cada una de las clases que representan las 26 direcciones de movimiento posibles.

Hay que destacar que las observaciones se van construyendo a lo largo del intervalo de tiempo en que se están produciendo, sin necesidad de esperar a que terminen. Es decir, el Observador especificado en la EspObservacion devuelve una respuesta en cada ciclo de observación. El Subsistema de Observación es capaz de gestionar distintos tipos de observaciones: observaciones instantáneas y observaciones continuas, estando algunas de ellas afectadas por la incertidumbre.

Si además representan un movimiento, se incluye su representación difusa generando una ObservacionDifusa. Contiene los mismos elementos que la Observación y además contiene una lista (*ListaDifusa*) para registrar el grado de pertenencia a las direcciones de movimiento. Los atributos completos son los siguientes:

- *Nombre:* Identificador que corresponde con el nombre de la EspObservación que lo especifica.
- *Inicio:* Marca temporal que representa el instante en el que la observación ha comenzado.
- *Abierto:* Sus posibles valores son verdadero o falso. Si su valor es verdadero indica que la observación no ha acabado y por lo tanto se puede seguir observando en ciclos futuros.
- *Fin:* Marca temporal que representa el instante en el que la observación ha finalizado. En el caso de que *Abierto* sea verdadero, esta marca temporal indica el instante actual de simulación.
- *Estado:* Indica el estado en el que se encuentra una Observación. La observación puede estar dándose (estado *true*), no existir (estado *false*) o se puede encontrar en un estado de incertidumbre (estado *wait*).
- *ConfirmadoHasta:* Marca temporal que indica el instante hasta el que se ha confirmado la observación, es decir, marca el final del primer subintervalo y el inicio del segundo. En las observaciones donde no exista la incertidumbre esta marca siempre coincidirá con la marca de *Fin* y ocurre lo mismo cuando una observación ha finalizado.
- *Propiedades*: Lista de objetos que contienen los valores recogidos para cada Propiedad durante el intervalo *Inicio-ConfirmadoHasta*. La lista contiene tantos elementos como Propiedades especificados en EspObservación.

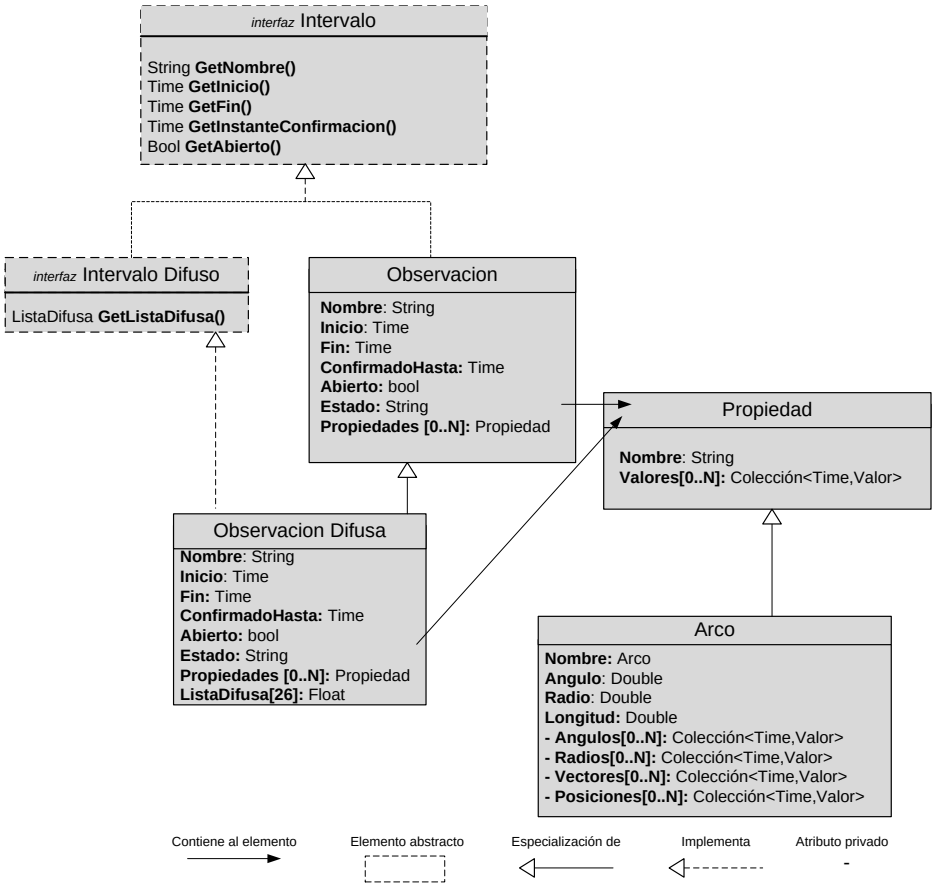


Figura 24: Definición de Observación, Observación Difusa, Propiedad y Arco.

El elemento Propiedad recoge los valores de la propiedad especificada por EspPropiedad en la duración del subintervalo *Inicio-ConfirmadoHasta* y viene especificado por estos atributos:

- *Nombre* de la propiedad.
- *Valores:* Lista de valores asociados a los instantes temporales en los que han sido calculados.

Por último, el Arco es la propiedad que utiliza la ObservaciónDifusa para representar de forma aproximada un segmento de un movimiento. Los atributos del Arco se calculan en un sistema de coordenadas local que se obtiene de la siguiente manera: el origen del sistema de

coordenadas se sitúa en el punto de inicio del segmento (Figura 25). El vector v_{12} es el vector que define la dirección del movimiento (desde el punto inicial del movimiento al punto final) y representa el eje X del sistema de coordenadas local. El eje Y es obtenido basándose en la proyección del eje X en el plano horizontal del sistema de coordenadas globales. De esta manera, todos los movimientos tienen una misma referencia estática.

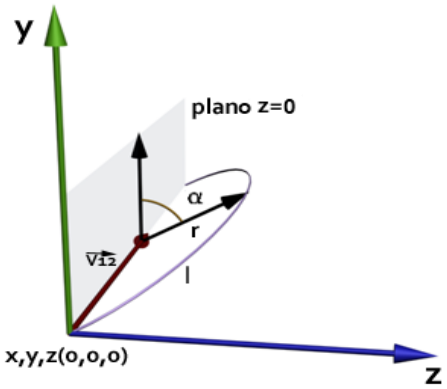


Figura 25: El vector que representa la dirección del movimiento

Con este sistema de referencia como base, estos son los elementos que definen un Arco:

- *Ángulo:* Representa el ángulo (α en la Figura 25) entre el plano Z=0 del sistema de coordenadas local y el movimiento.
- *Radio:* Representa el radio del arco (r en la Figura 25).
- *Longitud:* Representa la longitud del arco (l en la Figura 25).
- *Valores:* Almacenan la trayectoria el arco para cada ciclo de simulación. Estos valores posibilitan el cálculo de las propiedades ángulo, radio y longitud del arco.

Estos atributos permiten definir el arco de una manera simple y a su vez contienen la información suficiente para que el Subsistema de Diagnóstico genere resultados referidos a segmentos de una trayectoria compleja expresables verbalmente.

3.3.3.3 Proceso de extracción de observaciones

El Subsistema de Observación ejecuta en cada ciclo de simulación el proceso de extracción de observaciones. Este proceso recorre la lista de especificaciones contenida en el Modelo de Observación y llama a sus Observadores para comprobar si se están produciendo en ese instante. En el siguiente apartado se describe el proceso de actualización de observaciones a lo largo del tiempo teniendo en cuenta la gestión del estado de incertidumbre que puede afectarlas. El apartado posterior detalla el procedimiento con el que se extraen observaciones difusas y se calculan sus propiedades.

- **Proceso de actualización de observaciones bajo incertidumbre**

El Subsistema de Observación se encarga de crear Observaciones cuando se detecta una nueva, de extender la duración de los intervalos ya existentes y de terminarlas cuando dejan de observarse. Cuando se trata de una Observación afectada por incertidumbre también se encarga de confirmarlas y descartarlas.

En la siguiente figura se representan con líneas verticales discontinuas todos los instantes o ciclos de simulación en los que los Observadores generan tres ejemplos de observaciones: una instantánea, una continua sin incertidumbre y una observación bajo incertidumbre (Figura 26). A continuación se describe la respuesta de los observadores durante varios instantes:

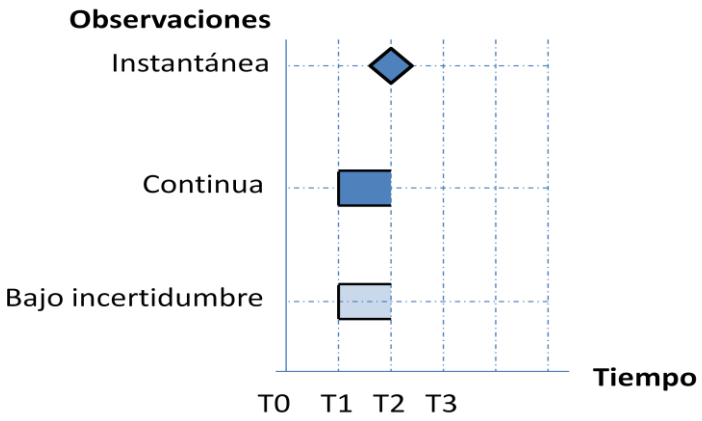


Figura 26: Detección de observaciones. Fase 1

- T0: Comienzo del proceso de observación.
- T1: Instante donde comienzan la observación continua y la observación bajo incertidumbre.
- T2: En este instante se ha detectado una observación instantánea y la observación continua se alarga hasta este instante. La observación bajo incertidumbre también se ha alargado hasta este instante, pero aún no se puede confirmar que ese hecho esté ocurriendo. Hay que esperar para tener certeza sobre si la observación está ocurriendo o no. Como esta observación no está confirmada, no se pueden calcular sus propiedades.

Después de que hayan pasado unos instantes, el cronograma tendría la siguiente forma y éstas serían las respuestas de los observadores en esos instantes:

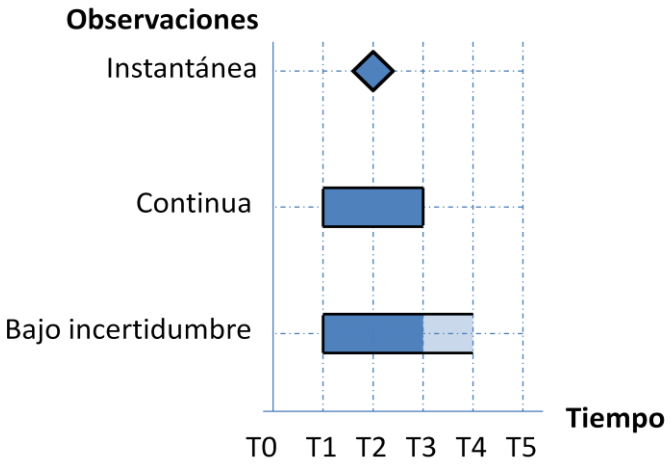


Figura 27: Detección de observaciones. Fase 2

- T3: En este instante se deja de detectar la observación continua. Además, se confirma que la observación bajo incertidumbre estaba ocurriendo hasta este instante y se registra ese instante. Se lleva a cabo el cálculo de sus propiedades hasta T3.
- T4: La observación bajo incertidumbre está confirmada hasta el instante T3, pero en el instante T4 no hay certeza de que la observación siga ocurriendo o de que haya dejado de ocurrir.

Por lo tanto, una Observación contendrá la información sobre el momento que comenzó a observarse, el momento hasta el que se ha confirmado la observación y se indica si ha finalizado. Cuando termina, la Observación está descrita por los instantes en que comenzó y dejo de observarse, ya que la incertidumbre debe haberse resuelto obligatoriamente. Si la Observación no está sujeta a incertidumbre, su instante de confirmación siempre coincidirá con el instante final de la Observación.

Si el instante inicial coincide con el instante de confirmación, significará que aún no hay certeza de que la observación se está dando (parte izquierda Figura 28). Cuando el observador confirma una observación con incertidumbre por primera vez, existen dos posibilidades: que la confirmación sea positiva o negativa. Si la confirmación es positiva, se actualiza el dato donde se indica que la

observación se confirma hasta este instante. Si ocurre lo contrario, el Observador correspondiente se deshace de la observación.

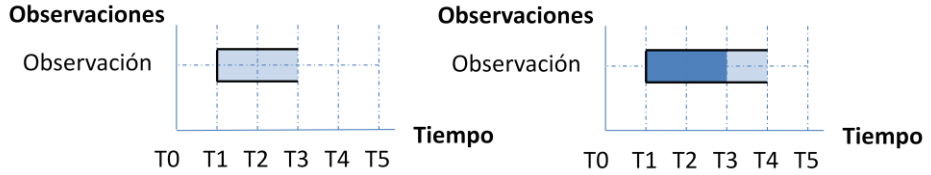


Figura 28: Posible desarrollo de una observación con incertidumbre durante el tiempo

En caso de que la confirmación haya sido positiva la primera vez, un escenario posible sería el que se muestra en la parte derecha de la Figura 28. La observación ha sido confirmada hasta el instante T3, y en este momento (instante T4) se podrían dar tres escenarios posibles:

1. Se confirma la observación positivamente: La observación se alarga hasta el instante T4 y se recalculan las propiedades con los datos almacenados hasta este instante.
2. Se confirma la observación negativamente: Se indica que el instante final de la observación es el instante T3 y se desechan los datos guardados entre el instante T3 y T4.
3. No llega confirmación: Hay que esperar a ciclos futuros para ver si la observación se alarga o no.

Además de actualizar las observaciones bajo incertidumbre, el Subsistema de Observación se encarga de notificar al Subsistema de Interpretación y Subsistema de Diagnóstico sobre los cambios relevantes en las Observaciones. Estos son los distintos mensajes que se mandan a estos subsistemas:

- Finalizar Observación: Notifica la finalización de una Observación.
- Nueva Observación: Indica que una nueva Observación ha comenzado.
- Observación en espera: Indica que una Observación se encuentra en estado de incertidumbre.
- Cancelar Observación: Indica que una Observación que se encontraba en estado de incertidumbre ha sido cancelada.

- Actualizar Propiedades: Mensaje que indica que las propiedades de una Observación han cambiado.
- Actualizar Propiedades en espera: Este mensaje indica la confirmación de una Observación. En la notificación también se incluye las propiedades actualizadas de la Observación, las cuales se han calculado al confirmarse el intervalo.
- Continuar Observación: Indica la continuación de una Observación confirmada.

➢ **Proceso de generación de observaciones difusas**

El Observador de movimientos se encarga de extraer observaciones difusas a partir de un flujo de posiciones 3D que forman la trayectoria de un punto característico. El proceso se describe en el siguiente pseudocódigo:

```
1   Observaciones[26]=ExtraerObservacionesConIncertidumbre(
2   umbralTiempo,umbralCambioDireccion,Trayectoria
3   instanteActual)
4   {
5       obsAux[26]; //lista con las 26 observaciones
6       correspondientes a las 26 direcciones de movimientos
7
8       segmento=Segmentar(umbralCambioDireccion,
9       Trayectoria,instanteActual,umbralTiempo);
10
11      Si (segmento != nulo) //existe un nuevo segmento
12      {
13          claseMov = Clasificar(segmento, out listaDifu-
            sa);
14
15          Para cada clase en ListaClasesMov
16          {
17
18              Si claseMov == clase
19              {
20                  obsAux[clase].Estado = "true";
21              }
22
23              Si no //se marcan las observaciones a
24              //descartar
25              {
26                  obsAux[clase].Estado = "false";
27              }
```

```
28          }
29        }
30        Si no //no se sabe cuál de los 26 tipos de
31        //movimientos se está llevando a cabo
32        {
33            Para cada clase en ListaClasesMov
34            {
35                obsAux[clase].Estado = "wait";
36            }
37        }
38   return obsAux;
39   }
40
41   //función de segmentación del movimiento
42  Segmento= Segmentar(umbralTiempo, umbralCambio-
43  Direccion, Trayectoria, instanteActual)
44  {
45  segmento= nulo;
46      transcurrido=CalcularPeriodo(instanteActual);
47
48      bool cambioDir=DetectarCambioDireccion(
49      umbralCambioDireccion,Trayectoria, out comienzo,
50      out fin);
51
52      Si cambioDir o transcurrido > umbralTiempo
53      {
54          segmento=Segmentar(comienzo,fin,Trayectoria);
55      }
56  return segmento;
57  }
```

Este proceso de extracción de observaciones ejecuta los siguientes pasos que se han detallado en el pseudocódigo:

1. *Generación de lista de observaciones*: Por cada movimiento en cada ciclo, el subsistema tiene que decidir cuál de los 26 movimientos se está llevando a cabo. Cada uno de estos 26 tipos de movimientos está ligado a una observación. Por esta razón, el Observador genera una lista con las 26 observaciones correspondientes a los 26 tipos de movimientos y asigna el estado a cada uno de ellos: *true, false* o *wait* (línea 5)
2. *Segmentación* (llamada a la función *Segmentar* de la línea 8, implementación en línea 42): El objetivo de la segmentación es generar segmentos para que se pueda llevar a cabo una

aproximación de la trayectoria a una secuencia de arcos. La segmentación se hace en base a dos criterios: la detección de cambios de dirección y tiempo transcurrido desde el final de la detección del último segmento.

2.1. *Calculo de tiempo transcurrido* (línea 46): El algoritmo de extracción de observaciones establece un umbral de tiempo para segmentar el movimiento. Si discurre un periodo de tiempo considerable sin que se haya detectado un cambio de dirección, se lleva a cabo la segmentación. De esta manera, se asegura que el subsistema va a seguir generando observaciones aunque el movimiento se esté ejecutando lentamente.

2.2. *Detección de cambio de dirección* (línea 48): Los cambios de dirección se detectan si hay un cambio de dirección en cualquiera de los ejes (X, Y o Z) del sistema de coordenadas local. Se define un umbral que establece qué es un cambio de dirección significativo, el cual se calibra dependiendo del nivel de precisión que se necesite en la evaluación de la tarea.

3. *Clasificación* (línea 13)*:* Si se detecta un nuevo segmento (línea 11), se procede a la clasificación de dicho segmento. Esta clasificación se ejecuta utilizando una función difusa y sirve para tratar la imprecisión que un alumno puede tener al realizar un movimiento. Por esta razón, cuando el segmento es clasificado, dicho segmento pertenece a una sola clase, pero también se calcula la distancia al resto de clases para deducir la similitud entre el movimiento ejecutado y las clases. El primer paso en el proceso de clasificación es el cálculo del ángulo (β) entre el vector del movimiento (Figura 25) y cada una de las clases de la (Figura 29):

$$\beta = \cos^{-1}(\overrightarrow{v12}\ x\ \overrightarrow{vClass})$$

Seguidamente se calcula el grado de pertenencia (distancia) del movimiento a cada clase basándose en esta función difusa:

$$f(class) = \begin{cases} 0, & \beta \geq \frac{\pi}{4} \\ \frac{\frac{\pi}{4} - \beta}{\frac{\pi}{4}}, & 0 \leq \beta < \frac{\pi}{4} \end{cases} \quad (3)$$

Esto indica que el movimiento se clasificará en la clase que más se asemeje y los resultados de grados de semejanza se almacenarán en la ListaDifusa de la Observación Difusa.

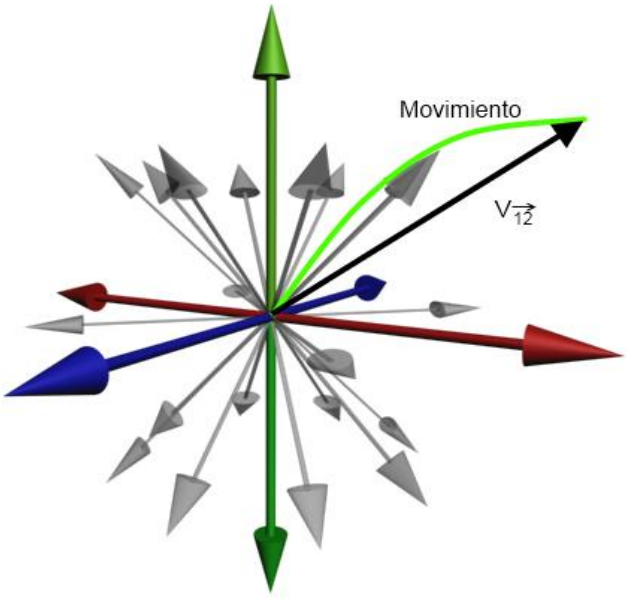


Figura 29: Las 26 clases utilizadas para clasificar el movimiento.

4. *Gestión de los estados de la observación* (líneas 15 -38): La detección de un segmento indica el fin de la incertidumbre respecto a un movimiento. Por lo tanto, una vez que se ha llevado a cabo la clasificación, el método del observador asigna el estado de la Observación relacionada al movimiento a "true" (línea 20), e indica que el resto de observaciones que se barajaban como hipótesis eran falsas (línea 26). Sin embargo, si no existe un cambio de dirección considerable ni ha transcurrido el tiempo suficiente para determinar qué movimiento se está llevando a

cabo (línea 30), esto significa que la Observación Difusa se encuentra en un estado de incertidumbre. Es decir, no existe información suficiente para determinar qué movimiento está llevando a cabo el estudiante. Debido a ello, el método indicará al Subsistema de Observación esta circunstancia (línea 35) asignando el estado de cada una de las 26 observaciones a "wait".

Una vez se ha determinado el estado de cada una de las 26 Observaciones Difusas posibles, el Subsistema de Observación se encarga de mantener la coherencia temporal de estas observaciones (ver Figura 30). Durante los instantes T1 y T3, el método ExtraerObservacionesConIncertidumbre devolvería la lista de las 26 observaciones en estado "wait". Durante este periodo de incertidumbre, el subsistema se encarga de acumular las propiedades de la observación que interesan para que cuando se detecte un nuevo segmento (instante T4) se pueda hacer el cálculo del Arco. En ese mismo instante el Subsistema se deshace de las Observaciones Difusas que aún no habían sido confirmadas y confirma la observación que se ha detectado.

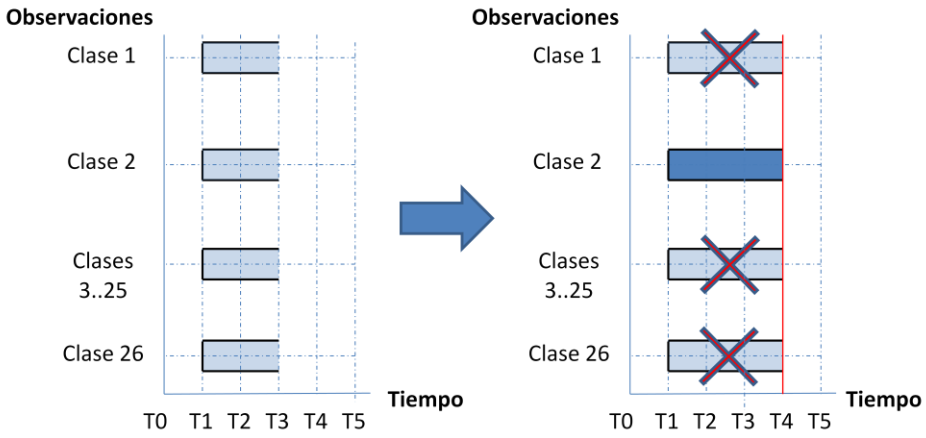


Figura 30: Coherencia temporal en el Subsistema de Observación

En el caso de la Observación Difusa que se ha confirmado, el Subsistema de Observación procede al cálculo de las propiedades del Arco y para ello se utilizan las posiciones del punto característico que se han

almacenado durante el periodo de incertidumbre. Para cada punto, se representa un vector definido por el origen del movimiento y el punto, y se obtienen el ángulo y la distancia respecto al vector que define el movimiento en el sistema de coordenadas local. La media de estos valores permite la obtención de las propiedades ángulo, radio y longitud del Arco que define cada segmento del movimiento.

Cuando se detecta un nuevo segmento, también existe la posibilidad de que el nuevo segmento sea de la misma clase que el segmento anterior. En este caso, el Subsistema de Observación alarga la observación hasta el instante donde se ha detectado el nuevo segmento y se vuelve a hacer el cálculo de las propiedades del Arco con los datos del arco entero (arco anterior más nuevo arco).

3.4 INTERPRETACIÓN

Una vez que se ha observado lo que está pasando en el Sistema Interactivo, el siguiente paso consta en interpretar esa actividad. Para ello, el Nivel de Interpretación describe de forma genérica cómo reconocer esa actividad del alumno. Concretamente, los elementos utilizados en este nivel se utilizarán para describir la actividad que vaya a ser diagnosticada, formando el Modelo de Interpretación. La información proporcionada por este modelo es utilizada por el Subsistema de Interpretación, que determina la actividad del alumno y comunica al Subsistema de Diagnóstico las conclusiones a las que va llegando mientras interpreta (Figura 31).

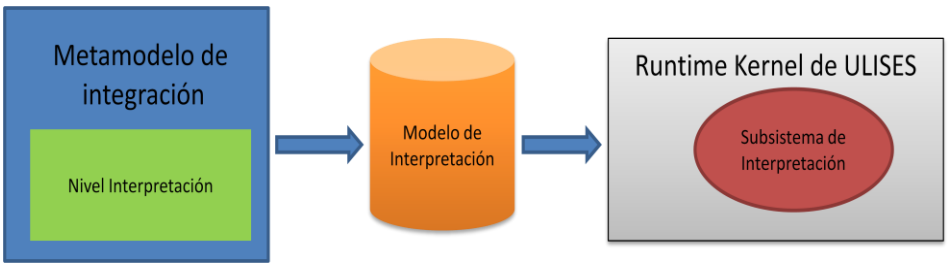


Figura 31: Relación entre Nivel, Modelo y Subsistema de Interpretación.

3.4.1 Descripción del Nivel de Interpretación

El Nivel de Interpretación describe genéricamente cómo reconocer la actividad del estudiante en el SI, es decir, expresa digitalmente lo que está ocurriendo en el entorno virtual. Así como los instructores realizan interpretaciones subjetivas basándose en lo que perciben del entorno, ULISES imita ese comportamiento a la hora de interpretar las observaciones del Sistema Interactivo. Es decir, pueden existir varias interpretaciones válidas para la misma actividad del alumno. De todas formas, la interpretación será válida o suficientemente completa si sirve para obtener resultados de diagnóstico ajustados al criterio del instructor.

Por lo tanto, se puede decir que el primer objetivo de este nivel es la creación de información útil y válida para el diagnóstico. Por otro lado, la interpretación tiene que conseguir gestionar tanto la incertidumbre temporal, como la intencionalidad del alumno:

- En primer lugar, la interpretación tiene que ser capaz de gestionar la incertidumbre. Como se ha detallado en el Nivel de Observación, existen periodos de tiempo donde todavía el Subsistema de Observación no puede determinar qué es lo que está siendo observado. Esta misma incertidumbre se extiende al Nivel de Interpretación. Por ejemplo, si observáramos un conjunto aleatorio de movimientos de tenis ¿cuándo es posible saber que un alumno está llevando a cabo un saque de tenis? Probablemente antes de que el movimiento se haya acabado, un instructor sería capaz de decir con certeza que el alumno está intentando hacer un saque e incluso, descartar otras posibles acciones que el instructor tuviera en mente sobre las acciones del aprendiz. Sin embargo, hasta que la interpretación sepa con certeza de qué acción se trata, no puede perder los datos pasados ni descartar acciones hasta que la interpretación sea fiable.
- En segundo lugar, el Nivel de Interpretación tiene que ofrecer la flexibilidad suficiente para que se pueda modelar el comportamiento de un estudiante. Cuando se trata de aprender una nueva destreza, es muy probable que el estudiante no pueda llevar a cabo la misma acción que tiene en mente, sino una parecida. Por esta razón, este

nivel tiene que ofrecer los elementos necesarios para modelar la intencionalidad que tiene un alumno cuando lleva a cabo una acción.

En base a estos requisitos, el Nivel de Interpretación describe los elementos necesarios para que se puedan reconocer la actividad del estudiante en su contexto, los Pasos y las Situaciones. Esta actividad se modela utilizando restricciones, los cuales permiten establecer relaciones temporales u otro tipo de relaciones entre observaciones y que además permiten modelar la intencionalidad que los movimientos. En cuanto al Subsistema de Interpretación, se encarga de evaluar las restricciones definidas para discernir la actividad del alumno teniendo en cuenta la incertidumbre de las observaciones generadas en el Subsistema de Observación.

3.4.2 Elementos del Nivel de Interpretación

El Nivel de Interpretación contiene los elementos necesarios para especificar cómo reconocer las acciones del estudiante y el contexto en el que las realizan a partir de las observaciones especificadas en el Nivel de Observación, que son: la especificación del paso (EspIPaso) y de la situación (EspISituacion). El primero contiene los atributos necesarios para que el Subsistema de Interpretación sepa cómo identificar las acciones del alumno. El segundo contiene los atributos necesarios para definir los contextos donde se pueden llevar a cabo los pasos.

Estos dos elementos se modelan utilizando el paradigma del modelado por **restricciones.** Contextualizando esta técnica en ULISES, el modelado de estos elementos consiste en definir relaciones complejas entre especificaciones de intervalos de tiempo: EspObservacion y sus derivados (EspObservacionDifusa), EspIPaso y EspISituacion. Los elementos para definir restricciones se definen en el siguiente apartado, resaltando las restricciones temporales para el modelado de destrezas físicas.

3.4.2.1 Elementos de restricción

El Nivel de Interpretación especifica acciones continuas, con o sin incertidumbre, además de con acciones puntuales. Para ello se emplean distintos tipos de restricciones: Restricciones temporales para relacionar cualitativa y cuantitativamente los intervalos, Restricciones Lógicas para

combinar restricciones, Restricciones sobre Propiedades para acotar el valor de las propiedades de las observaciones y Restricciones sobre Movimientos que combinan varias restricciones sobre los atributos de las observaciones difusas y/o sobre los arcos que puedan contener:

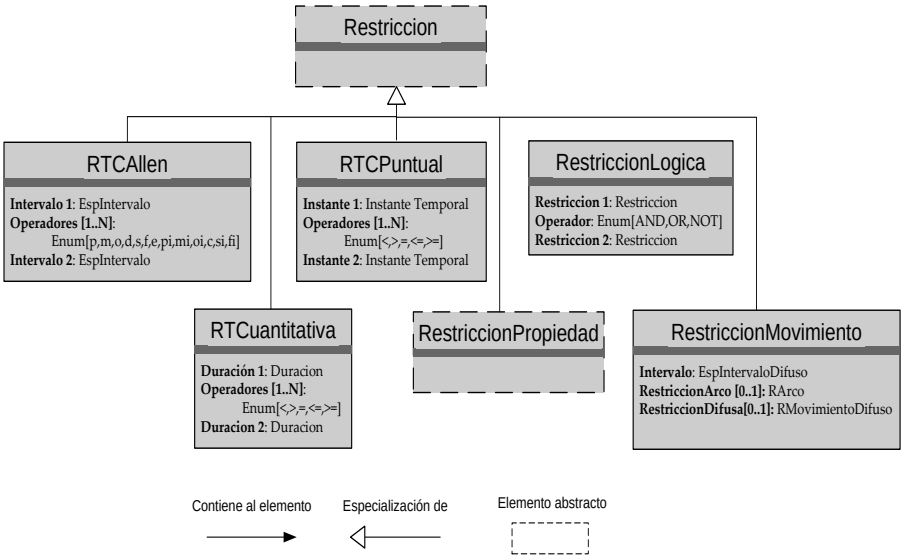


Figura 32: Tipos de restricciones

En el Anexo A se incluyen las reglas de evaluación para las restricciones que se definen en este apartado.

- **Restricciones temporales:**

Existen dos tipos de restricciones temporales: las Restricciones Temporales Cualitativas y las Restricciones Temporales Cuantitativas. A su vez, las Restricciones Temporales Cualitativas se pueden distinguir dos tipos de restricciones:

- Restricciones Temporales Cualitativas de Allen (RTCAllen): Se utilizan para definir relaciones temporales entre intervalos de tiempo sin la necesidad de especificar numéricamente los instantes de inicio, final o la duración de los intervalos (Allen, 1984). La Figura

33 muestra las 13 relaciones básicas que se pueden establecer utilizando el álgebra de intervalos de Allen:

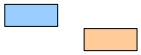
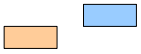
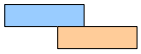
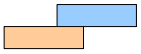
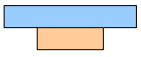
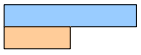
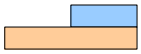
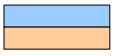
Relación	Simb			Simb	Relación inversa
Precede *Precedes*	*p*			*pi*	Es precedido por *Is preceded by*
Seguido de *Meets*	*m*			*mi*	Sigue a *Is met by*
Se solapa con *Overlaps*	*o*			*oi*	Es solapado por *Is overlapped by*
Durante *During*	*d*			*c*	Contiene *Contains*
Comienza *Starts*	*s*			*si*	Es comenzado por *Is started by*
Finaliza *Finishes*	*f*			*fi*	Es finalizado por *Is finished by*
Igual *Equals*	*e*				

Figura 33: Relaciones básicas del álgebra de Allen.

- Restricciones Temporales Cualitativas Puntuales (RTCPuntual): Definen relaciones cualitativas temporales entre los instantes de inicio o fin de dos intervalos. El siguiente ejemplo ilustra una restricción temporal de este tipo, donde el instante final del intervalo LevantarBrazo es anterior al inicio del intervalo SubirPierna:

  LevantarBrazo.Fin<SubirPierna.Inicio

  Los posibles valores de los operadores en este tipo de restricciones son: "<" (menor), "=" (igual), ">" (mayor), "<=" (menor o igual), ">=" mayor o igual).

Por último, las Relaciones Temporales se completan con las:

- Restricciones Temporales Cuantitativas (RTCuantitativa): Estas restricciones se utilizan para establecer restricciones sobre la duración de un intervalo o la diferencia de duración entre dos intervalos. A continuación se muestra un ejemplo donde se indica que tienen que pasar menos de 1 segundo entre el inicio del intervalo LevantarBrazo y el inicio del intervalo SubirPierna:

  Duracion(LevantarBrazo.Inicio, SubirPierna.Inicio) < 1000 ms

- **Restricciones sobre Propiedades**

Estas restricciones sirven para establecer condiciones sobre los valores de las propiedades de las observaciones. Por ejemplo, un profesor de tenis puede observar que el jugador se acerca a la pelota. Si la distancia es pequeña el jugador estará en situación de llevar cabo un golpeo, mientras que si se encuentra lejos de la pelota puede que los movimientos del tenista no tengan interés para el profesor. Para definir una situación de golpeo se podría definir una Restricción sobre la propiedad "Distancia" de la EspObservación "movimiento jugador":

$$MovimientoJugador.Distancia < 3m$$

- **Restricciones Lógicas**

Se utilizan para establecer relaciones lógicas entre intervalos mediante el uso de los operadores AND, OR y NOT. Ejemplo:

$$Duracion(LevantarBrazo) > 400ms \ AND \ Duracion(LevantarPierna) < 2000ms \ AND \ LevantarBrazo.Velocidad < 3 \ m/s$$

Además, estas restricciones permiten relacionar todo tipo de restricciones y relacionar sucesivamente varias restricciones. Por ejemplo, para representar que "el movimiento C comienza justo cuando finaliza el movimiento B y este comienza después del movimiento A":

A[Precedes]B AND B[Meets]C

- **Restricciones sobre Movimientos**

Las Restricciones sobre Movimientos se definen para establecer Restricciones sobre las Observaciones que impliquen un movimiento. Por un lado permiten establecer condiciones sobre las propiedades que se definen en un Arco del movimiento: longitud, radio y arco. Por otro lado, también se pueden establecer condiciones sobre el grado de pertenencia a las 26 direcciones de movimiento posibles que se definen en el atributo ListaDifusa de una ObservaciónDifusa.

En el siguiente ejemplo se muestran dos restricciones aplicadas sobre la observación RodillaDerechaDireccion010, la cual es observada cuando la rodilla derecha se levanta. La primera parte de la restricción

(Arco) restringe la concavidad, curvatura y la longitud del movimiento. La segunda parte (Fuzzy) restringe la dirección del movimiento, indicando el grado de similitud del movimiento respecto a sus clases de movimiento vecinas. Estas últimas restricciones están representadas por 26 restricciones lógicas y cada una de ellas restringe la similitud a una clase de movimiento mediante un umbral (en porcentaje).

*RodillaDerechaDireccion010.**Arc**(>90,<0.3,<0.5).**Fuzzy** (>= 0.0, >=0.0,>=0.0, >=0.0 ,>=0.0, >=0.0, >=0.0, >=0.0 ,>=0.0, >=0.0, >=0.0,>=0.0,>=0.0,>=0.0,>=0.0,>=0.0,>=0.0, >=0.0,>=0.0,>=0.0,>=0.0,>=0.0,>=0.0,>=0.0 AND <100.0,>=50.0)*

Por lo tanto, como se aprecia en la Figura 34 una Restricción de Movimiento está compuesta por tres atributos: un atributo *Intervalo:* que contiene una instancia de EspIntervaloDifuso para identificar la especificación de observación sobre la que se aplica la restricción, una Restricción de Arco (RArco) y una Restricción Difusa sobre el Movimiento (RMovimientoDifuso).

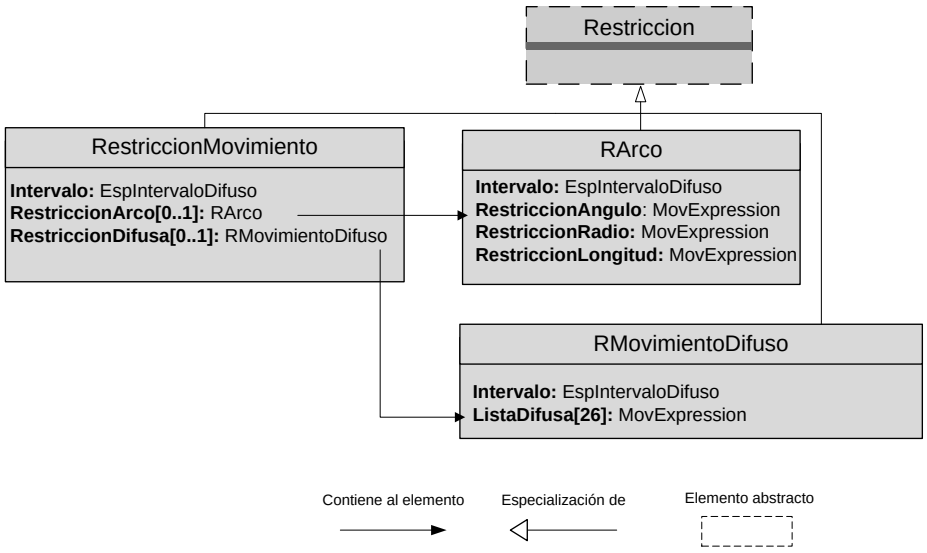


Figura 34: Especificación de las Restricciones de Movimiento.

Estas dos restricciones hacen uso de las expresiones de tipo MovExpression. Esta expresión permite añadir flexibilidad a la definición de Restricciones sobre Movimientos: permite utilizar operadores relacionales y operadores lógicos para definir restricciones complejas. Una MovExpressión contiene una o dos condiciones relacionadas mediante un operador lógico. Estos son los atributos que definen un MovExpression (Figura 35):

- *Operador relacional 1:* Operador relacional (<, >, =, <=, >=) que restringe la primera condición.
- *Valor 1:* Valor sobre el que se quiere aplicar la primera condición.
- *Operador lógico:* Operador lógico (AND, OR,) opcional para relacionar los pares *Operador1-Valor1* y *Operador2-Valor2*.
- *Operador relacional 2:* Operador relacional (<, >, =, <=, >=) que restringe la segunda condición. Solo se utilizará si se ha definido un operador lógico a la primera condición.
- *Valor 2:* Valor sobre el que se aplica la segunda condición. Solo se utilizará si se ha definido un operador lógico a la primera condición.

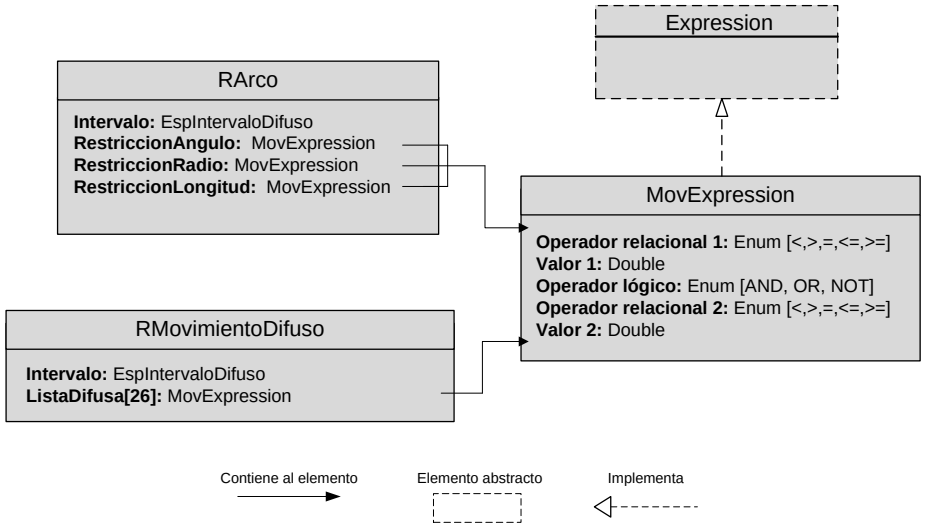


Figura 35: Especificación de RArco, RMovimientoDifuso y MovExpression.

En lo que se refiere a la Restricción de Arco, este elemento permite establecer condiciones sobre las propiedades ángulo, radio y longitud de un Arco. Estas propiedades se definen basándose en el sistema de referencia de la Figura 25 y las condiciones sobre ellas se establecen utilizando expresiones del tipo MovExpression. A continuación se describen en más detalle los atributos que contiene la Restricción de Arco:

- *Intervalo*: Contiene la instancia de EspIntervaloDifuso al que se refiere el movimiento.
- *Restricción sobre ángulo*: Restricción para establecer condiciones sobre el ángulo del Arco.
- *Restricción sobre radio*: Restricción para establecer condiciones sobre el radio del Arco.
- *Restricción sobre longitud*: Restricción para establecer condiciones sobre la longitud del Arco.

El último elemento que compone la RestricciónMovimiento es la RMovimientoDifuso, el cual contiene los siguientes atributos:

- Intervalo: Contiene la instancia de EspIntervaloDifuso al que se refiere el movimiento.
- ListaDifusa: Contiene una lista de 26 expresiones del tipo MovExpression. Como existen 26 clases de movimientos definidos en el Modelo de Observación, en cada MovExpression se especifica cómo se tiene que parecer el movimiento ejecutado a la clase de movimientos correspondiente.

Cabe decir que pueden existir casos donde solo interese establecer solo la Restricción sobre el Arco, o solo la Restricción Difusa sobre el movimiento, por lo que la Restricción de Movimiento permite definir solo un tipo de Restricción o incluir ambas Restricciones.

3.4.2.2 Especificación de pasos y situaciones

Los elementos EspIPaso y EspISituacion son los dos elementos de interés de cara al Nivel de Diagnóstico. Estos dos elementos describen mediante

restricciones cómo se van a interpretar las acciones del estudiante y el contexto en el que se llevan a cabo.

El elemento **EspIPaso** representa una acción que tiene lugar durante un intervalo de tiempo y que después será diagnosticado. El modelo de un paso contiene los atributos necesarios para que el runtime kernel de ULISES interprete cuando un paso está siendo llevado a cabo partiendo de las observaciones existentes. La especificación de un paso se describe mediante los siguientes atributos (Figura 36):

- *Nombre:* Identificafor del EspIPaso.
- *Restricciones generales:* Restricciones entre intervalos que al detectarse indicarán que el paso se está llevando a cabo. Cuando estas restricciones dejan de cumplirse se da el paso por finalizado.
- *Restricciones de inicio:* Restricciones entre intervalos que al detectarse indicarán que un paso ha comenzado. Si no se especifican estas restricciones el paso comenzará cuando se cumplan las restricciones generales.
- *Restricciones de ejecución:* Cuando las restricciones generales se cumplen, el Intérprete solicita el diagnóstico de paso. Sin embargo, las restricciones de ejecución permiten solicitar el diagnóstico de un paso cuando se satisfacen dichas restricciones.
- *Restricciones de fin*: Restricciones entre intervalos que al detectarse darán por finalizado un paso. Si estas restricciones no son especificadas, el paso no finalizará mientras se cumplan las restricciones generales.
- *Unidad de tiempo*: Si alguna restricción incluye algún dato temporal, este atributo indicará las unidades en las que tiene que expresarse. La unidad de tiempo predefinida es el milisegundo.

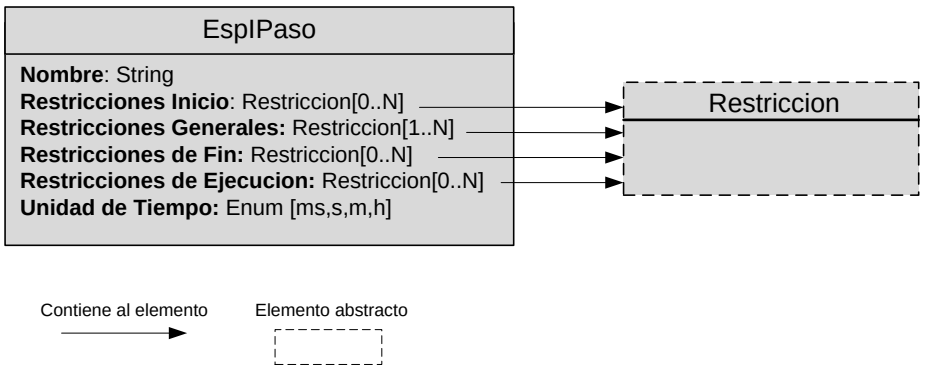


Figura 36: Especificación de paso en el Nivel de Interpretación

Es importante subrayar que un paso no debe ser interpretado basándose en la corrección o incorrección de una acción. Cuando se define un paso, hay que definirlo con el objetivo de identificar el paso independientemente del nivel de corrección del paso.

El último elemento del Nivel de Interpretación es el **EspISituación.** Una situación representa el contexto de acción específico que es relevante para el diagnóstico. Una situación es identificada cuando existen una serie de factores que determinan los pasos que los estudiantes deben ejecutar. Por ejemplo, en el dominio del tenis, si se quiere diagnosticar un golpeo, primero se deberá discernir si el jugador se encuentra en una situación de saque o si se encuentra disputando un juego. Cuando un jugador se encuentra disputando un juego ejecuta varios golpeos, pero estos golpeos no serían relevantes si lo que se está diagnosticando es un saque.

Para especificar una situación en el Nivel de Interpretación se emplea el elemento EspISituacion (Figura 37), y está descrito por los siguientes atributos:

- *Nombre*: Identificador de EspISituacion.
- *Restricciones generales*: Restricciones entre intervalos que al detectarse indicarán que la situación se está llevando a cabo. Cuando estas restricciones dejan de cumplirse se da la situación por finalizada.
- *Restricciones de inicio*: Restricciones entre intervalos que al detectarse indicarán que una situación ha comenzado. Si no se especifican estas

restricciones la situación comenzará cuando se cumplan las restricciones generales.

- *Restricciones de fin*: Restricciones entre intervalos que al detectarse darán una situación por finalizada. Si estas restricciones no son especificadas, la situación no finalizará mientras se cumplan las restricciones generales.
- *Unidad de tiempo*: Si alguna restricción incluye algún dato temporal, este atributo indicará las unidades en las que tiene que expresarse. La unidad de tiempo predefinida es el milisegundo.

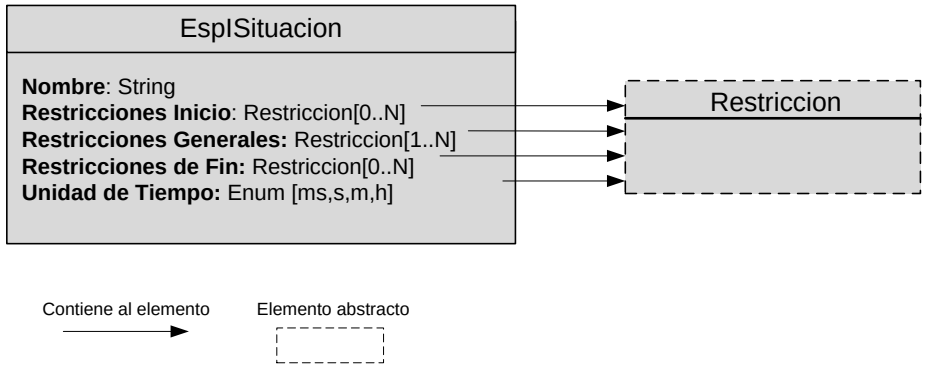


Figura 37: Especificación de situación en el Nivel de Interpretación

3.4.3 Subsistema de Interpretación

El Subsistema de Interpretación o Intérprete es el encargado de determinar qué pasos y situaciones que se han definido en el Modelo de Interpretación se están llevando a cabo en el Sistema Interactivo. Para ello utiliza las observaciones que recibe del Subsistema de Observación. Algunas de estas observaciones estarán bajo incertidumbre, por lo que el Intérprete debe gestionar observaciones sin saber si estas se van a confirmar y en caso de que las observaciones existan el Intérprete deberá trabajar con ellas sin saber si van a continuar. A su vez, este subsistema comunica los resultados de interpretación al Subsistema de Diagnóstico para que esté pueda generar un diagnóstico.

En este apartado se describen los elementos Paso y Situación que se generan en este subsistema y a continuación se describe el proceso de interpretación bajo incertidumbre.

3.4.3.1 Modelo Dinámico de Interpretación bajo incertidumbre

El Modelo Dinámico de Interpretación contiene los elementos Paso y Situación. Estos elementos se van generando a medida que el proceso de interpretación decide si los pasos y situaciones han comenzado, continúan o han finalizado.

Estos dos elementos implementan la interfaz Intervalo y contienen los siguientes atributos (Figura 38):

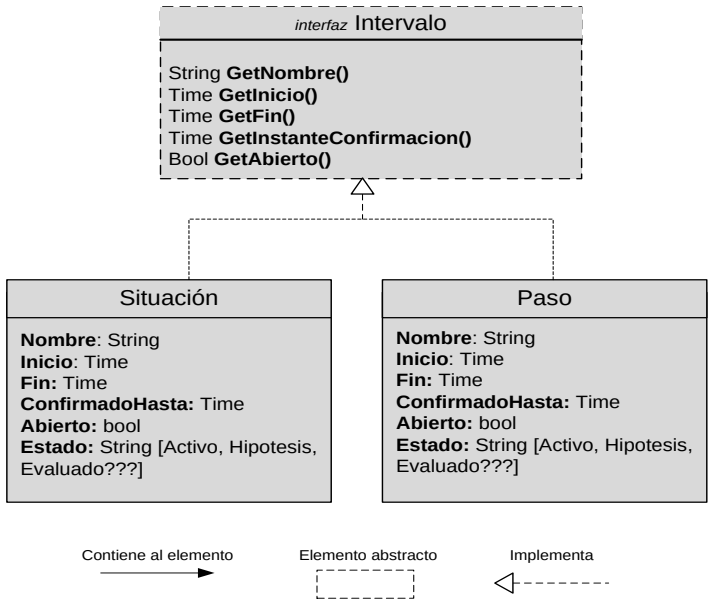


Figura 38: Definición de Situación y Paso

- *Nombre:* Identificador que coincide con el nombre del EspIPaso o EspISituación que se especifica.
- *Inicio:* Marca temporal que representa el instante en el que la observación ha comenzado.
- *Fin:* Marca temporal que representa en instante en el que la observación ha finalizado. En el caso de que *Abierto* sea verdadero, esta marca temporal indica el instante actual de simulación.
- *ConfirmadoHasta:* Marca temporal que indica el instante hasta el que se ha confirmado el paso o la situación. En los intervalos donde no

exista la incertidumbre esta marca siempre coincidirá con la marca de *Fin* y ocurre lo mismo cuando una observación ha finalizado.

- *Abierto:* Sus posibles valores son verdadero o falso. Si su valor es verdadero indica que el paso o la situación no ha acabado y por lo tanto se puede seguir interpretándose en ciclos futuros.
- *Estado:* Sus valores posibles son "Activo" o "Hipótesis". Cuando hay indicios de que un Paso o Situación pueden estar ocurriendo el proceso de interpretación crea una instancia de estos elementos en estado de "Hipótesis". Mientras exista incertidumbre los Pasos y Situaciones se mantendrán en este estado. Cuando acabe la incertidumbre el Intérprete confirmará (cambia el estado a "Activo") o descartará (cambia el estado a "Inactivo") el Paso o la Situación correspondiente.

3.4.3.2 Proceso de interpretación

El Subsistema de Interpretación decide qué pasos está llevando cabo el estudiante y en qué situaciones las está ejecutando partiendo de las observaciones extraídas por el Subsistema de Observación. Para ello, el proceso evalúa en cada ciclo las Restricciones que contienen cada EspIPaso y EspISituación teniendo en cuenta las Observaciones, Pasos y Situaciones que contiene el Intérprete en el instante correspondiente. Una vez que se han obtenido los resultados de evaluación de las Restricciones, el proceso tiene que actualizar el estado de cada EspIPaso y EspISituación, es decir, decide cuáles son aquellos pasos y situaciones que se están llevando a cabo.

- **Evaluación de restricciones**

Dado un conjunto de Observaciones, Pasos y Situaciones, se hace la evaluación de las restricciones para determinar cuanto antes si se están cumpliendo o no. Éstos son los valores posibles del resultado de una evaluación:

- Verdadero: La Restricción se cumple por completo.
- Falso: No se cumple la Restricción.

- Esperar_Datos: No hay suficientes datos (Observaciones, Pasos o Situaciones) para evaluar la Restricción. Los datos existentes no impiden que la Restricción se cumpla.
- Esperar_Final: No es posible saber si la restricción se cumple hasta que la Observación/Paso/Situación finalice o el Subsistema de Observación resuelva la incertidumbre.
- No_Datos: Ninguno de los elementos existentes (Observación, Paso o Situación) son referenciados en la evaluación de la Restricción actual.
- Abortar: La evaluación de la Restricción tiene que demorarse porque se referencian intervalos (Observaciones, Pasos o Situaciones) que aún no se han evaluado. Para poder llevar a cabo la evaluación de una Restricción, primero se tienen que haber evaluado todos los intervalos que se referencian.

Si se aprecian estos resultados de evaluación se puede observar que la evaluación de Restricciones no es inmediata. En el caso de que una restricción contenga instancias de observaciones que se ven afectadas por la incertidumbre, el proceso de evaluación de restricciones tiene que gestionar el mantenimiento de las instancias de esas observaciones, es decir, mientras las referencias existentes de observaciones permitan que una restricción se pueda cumplir, el proceso de evaluación de la restricción mantiene dichas instancias de las observaciones. En el siguiente diagrama se refleja el proceso que se sigue al evaluar una restricción:

Figura 39: Gestión de observaciones en la evaluación de una restricción.

El siguiente ejemplo ilustra la necesidad de gestionar adecuadamente las instancias de observaciones en memoria. Imaginemos que se define una restricción *A [Precedes] B*, donde *A* y *B* son observaciones afectadas por la incertidumbre. Esta restricción indica que la observación *A* debe finalizar antes de que comience la observación *B*. En la Figura 40 se muestran tres periodos consecutivos del proceso de evaluación de la restricción:

a) Al evaluar la restricción en el instante T3, la observación B no permite que se cumpla la restricción. A estas alturas de la evaluación no es posible que A preceda a B, se confirmen las observaciones o no. Por lo tanto el proceso evaluador de la restricción se deshace de la instancia de la observación B. Sin embargo, no se deshace del intervalo A porque existe la posibilidad de que aun preceda a otra instancia de B.

b) Después de deshacerse de la instancia B_1 de la observación B, el proceso ha readquirido otra instancia de la observación B. En el instante T4 el resultado de la evaluación de la restricción es "Falso". La instancia B_2 de la observación B impide que la restricción se cumpla. Por lo tanto, el proceso se deshace de la observación B y adquirirá una nueva instancia en el siguiente ciclo.

c) En este caso el proceso no se deshará de ninguna de las observaciones ya que aún existe la posibilidad de que A preceda a B: la observación A puede quedar confirmada solo hasta el instante T4 y B puede confirmarse en los siguientes ciclos.

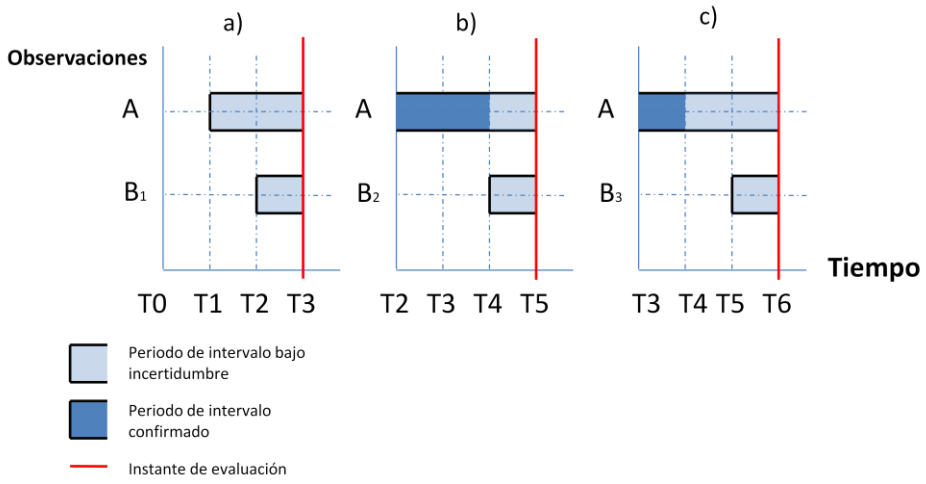


Figura 40: Ejemplo de gestión de las observaciones

➢ **Interpretación de Pasos y Situaciones**

El objetivo de evaluar las restricciones en los Pasos y Situaciones es la de inferir el estado de estos dos elementos. El proceso se asemeja al proceso que sigue un instructor real. Si un estudiante está realizando un movimiento, el instructor puede pensar que el estudiante no está llevando a cabo el movimiento que unos milisegundos atrás parecía ejecutar. En este contexto, el instructor puede desechar ese movimiento de su mente. Asimismo, se puede dar la situación contraria. El instructor puede confirmar que el estudiante está ejecutando ese movimiento que sospechaba que podía estar ocurriendo. Teniendo en cuenta este esquema, el proceso de interpretación decide el estado de los Pasos y Situaciones. Para ello el Intérprete se basa en los resultados de evaluación de las Restricciones contenidas en los EspIPaso y EspISituacion: restricciones de inicio, restricciones generales y restricciones de fin. Los posibles estados en los que se pueden encontrar los Pasos y las Situaciones son:

- Inactivo: El Paso o la Situación no se está produciendo.
- Activo: El Paso o la Situación se está produciendo.
- Hipótesis: Las Observaciones que se van extrayendo, los Pasos o las Situaciones existentes no permiten confirmar que el Paso o la Situación esté ocurriendo, pero en ciclos posteriores podría hacerlo.

Dependiendo del estado en el que se encuentren el paso o la situación se comprueban las distintas restricciones y se decide el estado al que tienen que pasar dependiendo del resultado de evaluación de esas restricciones: "Verdadero", "Falso", "Esperar_Datos", "Esperar_Final", "No_Datos". La siguiente figura muestra las diferentes transiciones de estados de los pasos y las situaciones:

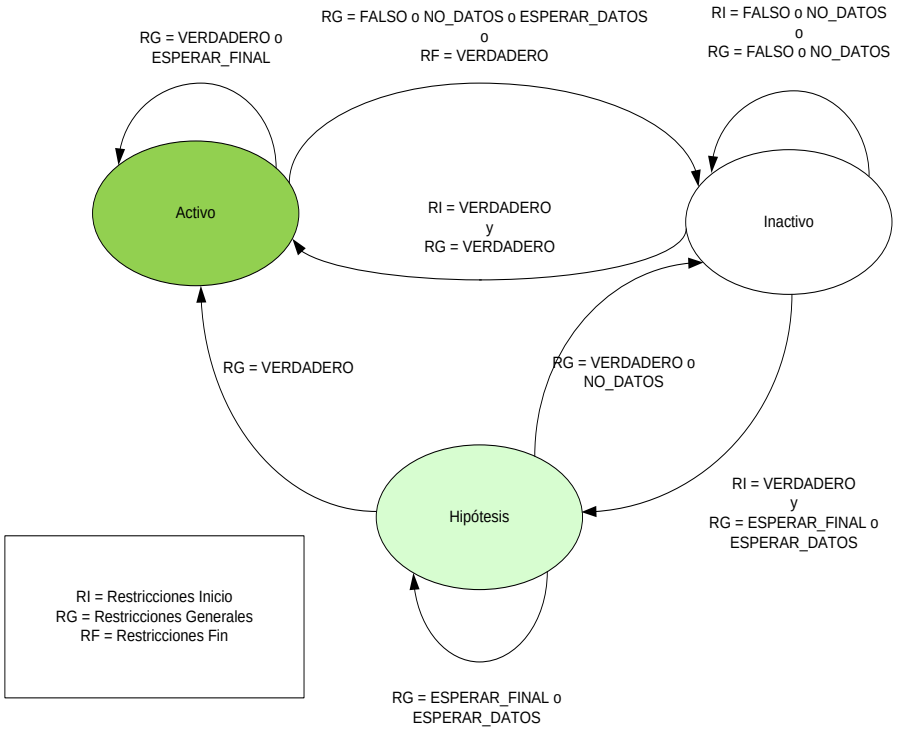


Figura 41: Diagrama de estados de la interpretación

En el siguiente pseudocódigo se muestra el proceso que se lleva a cabo en cada ciclo del Intérprete. En el primer paso se actualizan las

observaciones que el Intérprete tiene almacenados en memoria. Para ello, este subsistema utiliza los mensajes de notificación recibidos desde el Subsistema de Observación: "Nueva Observación", "Observación en espera", "Cancelar Observación", "Actualizar Propiedades", "Actualizar Propiedades en espera", "Finalizar Observación" y "Continuar Observación". A continuación se revisan las Situaciones y Pasos Activos, Hipotéticos y Nuevos en este orden de preferencia. En cada una de estas funciones se evalúan las restricciones correspondientes de cada Paso y Situación para actualizar sus estados. Como las restricciones pueden estar constituidas por otros pasos y situaciones (además de observaciones), existe la posibilidad de que cuando se el intérprete quiera evaluar una restricción todavía no se hayan evaluado los elementos que componen esa restricción. Cuando esto ocurre, la evaluación de restricciones devuelve un resultado del tipo "Abortar", indicando que el paso o la situación deben esperar a evaluarse. Esta reevaluación se lleva a cabo (línea 11 y línea 14) una vez que el resto de Pasos y Situaciones han sido evaluados.

```
1   ActualizarObservaciones(instante);
2   RevisarSituacionesActivas(instante, situacionesEspera);
3   RevisarPasosActivos(instante, pasosEspera);
4   RevisarSituacionesHipoteticas(instante,
5   situacionesEspera);
6   RevisarPasosHipoteticos(instante, pasosEspera);
7   RevisarNuevasSituaciones(instante);
8   RevisarNuevosPasos(instante);
9
10  Mientras (pasosEspera.Count > 0)
11      pasosEspera = resolverPasosEspera();
12
13  Mientras (situacionesEspera.Count > 0)
14      situacionesEspera = resolverSituacionesEspera();
```

Este proceso que se acaba de citar se lleva a cabo en todos los ciclos, pero existe un momento crítico donde el proceso de interpretación varía su comportamiento: el final de incertidumbre marcado por el Subsistema de Observación. La importancia de este instante reside en que en que el ciclo de interpretación se lleva a cabo en dos fases. Esto es debido a la

gestión de la incertidumbre por parte del Subsistema de Observación (Figura 42): cuando finaliza el periodo de incertidumbre, el Subsistema de Observación indica si las observaciones que se encontraban en estado de incertidumbre se confirman o se cancelan. Seguidamente, el Subsistema de Observación indica el comienzo de las nuevas observaciones bajo incertidumbre.

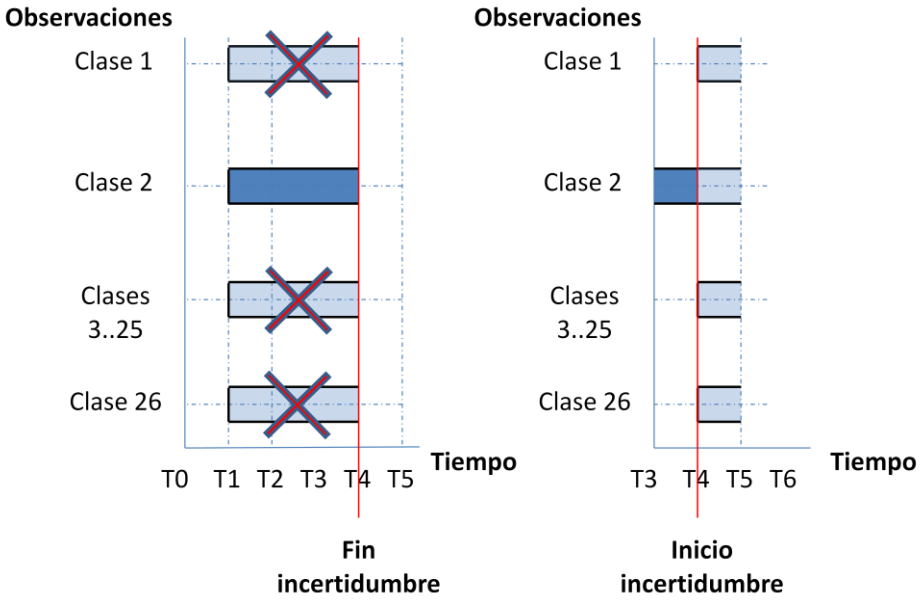


Figura 42: Momento de fin y nuevo periodo de incertidumbre

Debido a esta doble notificación, el ciclo de interpretación se hace en dos fases:

1. En la primera fase, el Intérprete resuelve la incertidumbre. Para ello, actualiza el estado de las observaciones que estaban bajo incertidumbre (las cuales tendrán las propiedades recalculadas) y lleva a cabo los pasos que se encuentran entre la línea 2 y 8 del pseudocódigo. De esta manera, los pasos o las situaciones que dependían de intervalos que se encontraban bajo incertidumbre actualizan sus estados.
2. En la segunda fase, se vuelve evaluar el estado de los pasos y las situaciones llevando a cabo el proceso que se ejecuta en ciclo común (líneas 1-14).

Esta interpretación en dos fases es imprescindible, ya que si no se llevará a cabo el Intérprete no podría adquirir las nuevas instancias de las observaciones bajo incertidumbre, y por lo tanto no podría mantener la cohesión temporal de los Pasos y Situaciones.

3.5 DIAGNÓSTICO

La generación de un diagnóstico de la actividad del alumno es el objetivo final de ULISES. Para ello el Nivel de Diagnóstico define los elementos que tienen que ser particularizados al construir un SIIAA para generar el Modelo de Tareas. Este modelo describe las tareas que los estudiantes van a llevar a cabo de tal forma que puedan ser diagnosticados automáticamente. El punto de partida del Nivel de Diagnóstico es DETECTive, un sistema de diagnóstico genérico para tareas procedimentales (Ferrero, 2004).

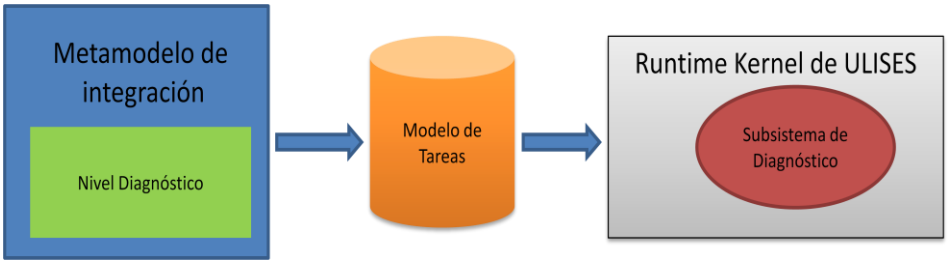


Figura 43: Relación entre el Nivel de Diagnóstico, Modelo de Tareas y subsistema de diagnóstico.

3.5.1 Descripción del Nivel de Diagnóstico

El objetivo de este nivel es proporcionar un conjunto de elementos que permitan la encapsulación del Modelo de Tareas, siguiendo los principios establecidos por DETECTive: genericidad, usabilidad y capacidad de generar diagnósticos apropiados y suficientes.

- Genericidad: Como en los dos anteriores niveles, hay que mantener la genericidad evitando dependencia del dominio.
- Usabilidad: Se favorece la usabilidad mediante una herramienta de autor para la generación del Modelo de Tareas. El capítulo 4 describe

en detalle la creación del Modelo de Tareas mediante la herramienta de autor.

- Capacidad de diagnóstico apropiado y suficiente: Al igual que en los niveles anteriores, el Nivel de Diagnóstico tiene que ser capaz de lidiar con la incertidumbre. Sin embargo, la capacidad de generar un diagnóstico apropiado y suficiente es específico de este nivel, porque depende de la técnica de diagnóstico utilizada. Como se ha citado anteriormente, ULISES permite la utilización de distintas técnicas de diagnóstico dependiendo del dominio donde se quiera integrar el SIIAA. Sea cual sea la técnica de diagnóstico utilizada, el Nivel de Diagnóstico tiene que ofrecer los elementos necesarios para encapsular el Modelo de Tareas adecuado para cada técnica.

3.5.2 Elementos del Nivel de Diagnóstico

El Nivel de Diagnóstico define los elementos para componer y resolver una tarea. Estos elementos son abstractos y acotan las características de los Modelos de Tareas que se pueden encapsular en ULISES. Cuando se integra una técnica que de diagnóstico en ULISES, estos elementos se especializan incorporando los atributos adicionales que la técnica necesita. Los elementos abstractos que este nivel define son el EspDPaso, EspDSituación, EspSolución, Tareas y Restricción. A continuación se detalla cada uno de estos elementos.

Un EspDPaso representa la unidad diagnosticable mínima en este nivel. Cuando el alumno lleva a cabo un movimiento, éste formará parte de un paso afectado por la incertidumbre. El paso será parte de una situación concreta y con ello contribuye en la solución de una Tarea. En el Nivel de Interpretación, un EspIPaso contiene los atributos necesarios para saber si un alumno está ejecutando un paso. En cambio, en este nivel, la especificación de paso define los atributos que sirven para determinar su corrección. Una especificación (EspDPaso) está compuesta (Figura 44) por los siguientes atributos:

- *Nombre*: Identificador único del EspDPaso. Este identificador tiene que coincidir con el identificador del EspIPaso que se quiera diagnosticar.

- *Observaciones:* Identificadores de EspObservaciones (que pueden incluir EspObservacionesDifusas) que se utilizan para determinar la corrección del paso especificado.

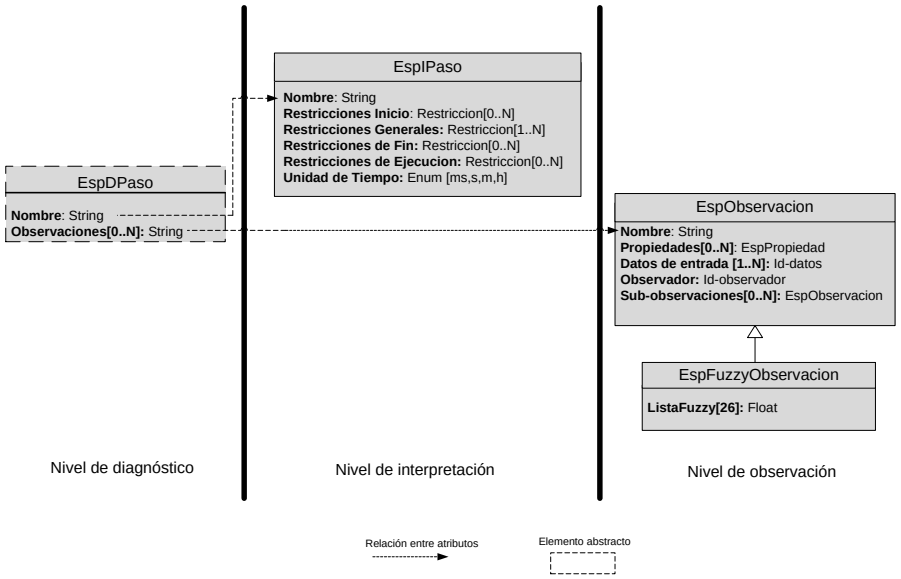


Figura 44: Especificación de Paso en el Nivel de Diagnóstico

El segundo elemento del Nivel de Diagnóstico es la EspDSituación, la cual representa el contexto donde se van a diagnosticar los pasos llevados a cabo por el alumno. Dado que los pasos y las observaciones relacionadas al movimiento estarán afectados por la incertidumbre, ésta se traslada también a la situación. Gracias a este elemento se puede discernir la corrección o la incorrección de un paso, ya que el significado del paso cambia dependiendo del contexto donde se esté ejecutando.

Además, hay que tener en cuenta que una situación puede resolverse de distintas maneras, por esta razón, una situación puede estar asociada a varias EspSoluciones, tanto correctas como incorrectas.

La EspDSituación, como elemento abstracto, está formada por los siguientes atributos (Figura 45):

- *Nombre:* Identificador único de la situación. Tiene que coincidir con el nombre de alguna de las EspISituaciones definidas en el Modelo de Interpretación.

- *Soluciones:* Lista de EspSoluciones correctas o incorrectas que se utilizan para diagnosticar las acciones del alumno dentro de una situación concreta.

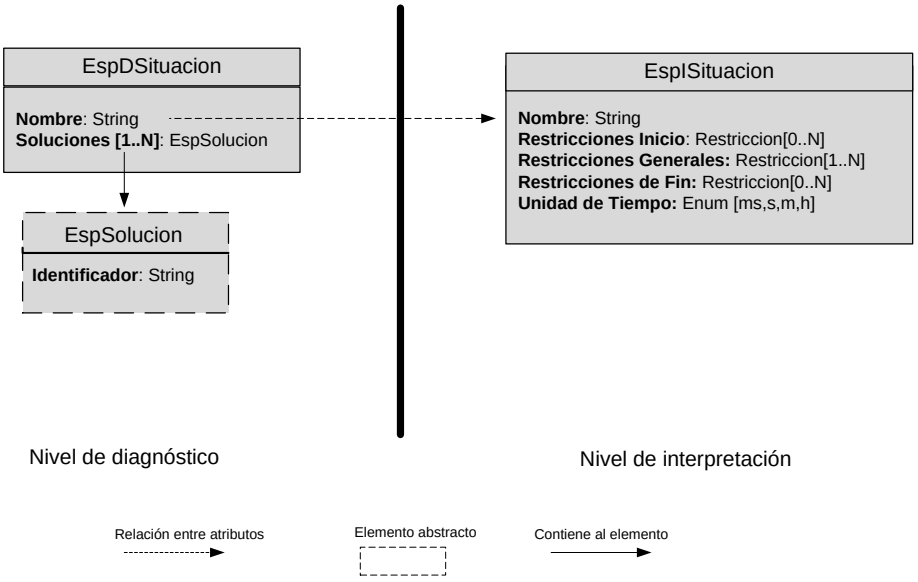


Figura 45: Especificación de Situación en el Nivel de Diagnóstico

En lo referente a la EspSolucion, se trata de un elemento que encapsula una forma de resolver una situación utilizando EspDPasos. Estas soluciones pueden ser tanto correctas como incorrectas. En algunos casos, conviene definir soluciones incorrectas, por ejemplo para representar errores típicos que un alumno pueda cometer. Por otro lado, cada EspSolucion puede estar relacionado con un módulo de diagnóstico diferente, por lo que se pueden utilizar múltiples técnicas de diagnóstico al mismo tiempo para distintas soluciones a la hora de evaluar una situación. Sin embargo, sea cual sea la técnica utilizada, la solución siempre estará compuesta por EspDPasos, aunque la estructura de la solución varíe de una técnica a otra.

El último elemento y principal elemento de este nivel es la Tarea. Cuando un profesor plantea una tarea, se le asocian varios objetivos de aprendizaje a dicha tarea. Después, en función de cómo haya sido resuelta la tarea, se podrá determinar el nivel de aprendizaje de un

estudiante. Para que esto sea posible, la especificación de una Tarea incluye la información necesaria para diagnosticar la actividad del alumno. La tarea está compuesta por las situaciones que el alumno debe afrontar y las correspondientes soluciones a estas situaciones. A continuación se describen los atributos que componen el elemento Tarea (Figura 46):

- *Nombre* de la tarea.
- *Enunciado*: Planteamiento de la tarea que se le comunica al alumno.
- *Descripción*: Descripción de la tarea que el alumno tiene que llevar a cabo.
- *Dificultad* del ejercicio.
- *Objetivos de aprendizaje*: Lista de objetivos de aprendizaje asociados a la tarea que han sido definidos por el instructor.
- *Situaciones*: Lista de EspDSituaciones a la que un alumno debe hacer frente para resolver la tarea.

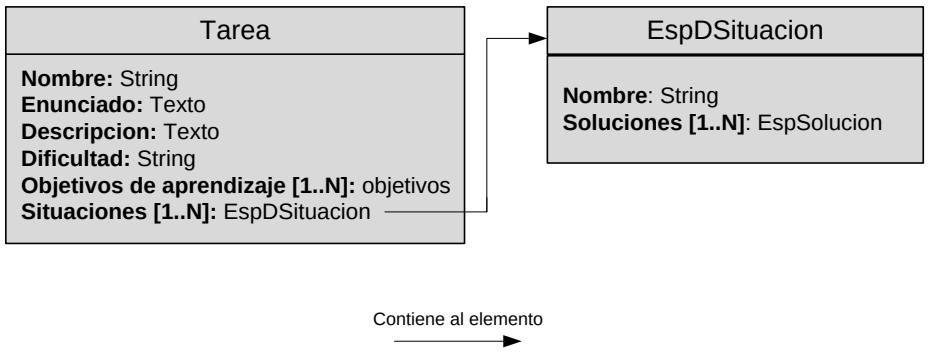


Figura 46: Especificación de la Tarea

3.5.3 Subsistema de Diagnóstico

El Subsistema de Diagnóstico es el encargado de detectar los errores que un alumno comete mientras lleva a cabo una tarea. Este subsistema recoge las observaciones generadas por el Subsistema de Observación y los pasos y las situaciones que se interpretan desde el Intérprete. Esta información se envía al módulo de diagnóstico correspondiente (dependiendo de la técnica que se utilice en cada solución) para que

lleve a cabo el diagnóstico. En este caso, dada la naturaleza del dominio de las destrezas físicas, se ha utilizado una técnica de diagnóstico basada en satisfacción de restricciones.

3.5.3.1 Proceso del diagnóstico

Independientemente de la técnica de diagnóstico utilizada para el diagnóstico, el Subsistema de Diagnóstico lleva a cabo un proceso común. Al igual que el Subsistema de Interpretación, este subsistema mantiene actualizadas las observaciones que necesita gracias a los mensajes de notificación que manda el Subsistema de Observación: "Nueva Observación", "Observación en espera", "Cancelar Observación", "Actualizar Propiedades", "Actualizar Propiedades en espera", "Finalizar Observación" y "Continuar Observación". Este mismo proceso ocurre con los pasos y situaciones. El Intérprete notifica a este subsistema todos los cambios de estado que sufren tanto los pasos como situaciones: paso de estado inactivo a hipotético, hipotético a activo y el final del paso o la situación. Del mismo modo, el Intérprete notifica al Subsistema de Diagnóstico si los pasos que se encuentran activos están bajo incertidumbre en cada uno de los ciclos.

Después de actualizar la información referente a las observaciones, pasos y situaciones, el proceso de diagnóstico distingue los pasos activos que se encuentran bajo incertidumbre y los que no. En el primer caso, el proceso de diagnóstico marca el paso para que la técnica de diagnóstico correspondiente no tenga en cuenta ese paso al llevar a cabo el diagnóstico. A diferencia del Intérprete, un paso sólo se debe diagnosticar cuando hay certeza de que está ocurriendo. Es decir, sólo se emitirá un diagnóstico válido en los ciclos que no exista incertidumbre.

Una vez que se ha llevado a cabo este marcado, el Subsistema de Diagnóstico manda a diagnosticar las soluciones de todas las situaciones que se encuentran activas. Cuando la situación finaliza, el subsistema se deshace de todas las soluciones y pasos contenidos en la situación.

Por último, el proceso de diagnóstico se encarga de recoger los resultados de diagnóstico y procesar esa información para que pueda ser utilizada por los Sistemas de Ayuda al Aprendizaje que compongan el SIIAA (por ejemplo HERMES).

3.5.3.2 Elementos de la técnica basada en satisfacción de restricciones

Los dominios donde se lleva a cabo el diagnóstico de destrezas motoras se consideran dominios pobremente definidos, ya que el grado de impredictibilidad de las acciones que lleva a cabo un alumno es alto. Como se ha comentado en el análisis del estado del arte de este trabajo, en este tipo de dominios las técnicas basadas en el modelado de restricciones son más adecuadas que otras como el reconocimiento de planes (Mitrovic & Weerasinghe, 2009). En el caso del diagnóstico de destrezas motoras el modelado basado en restricciones ofrece una clara ventaja: en vez de definir una manera de resolver un problema, permite definir cómo se deberían llevar a cabo las acciones con el objetivo de detectar errores en la ejecución de los mismos. Por lo tanto, cuando una restricción no se cumple, se detecta un error. Asimismo, el modelado basado en restricciones permite agrupar las acciones que violan el mismo principio del dominio. En la implementación de la técnica que se ha hecho en ULISES, además de esa agrupación se distingue el contexto donde se ha producido el error. Esta distinción es muy importante, ya que existen muchos casos donde el mismo error puede obtener un significado diferente dependiendo del contexto donde haya ocurrido.

Para que esto sea posible, el elemento EspSolucionRestriccion contiene una lista de pasos que pueden darse cuando el alumno afronta una situación (Figura 47).

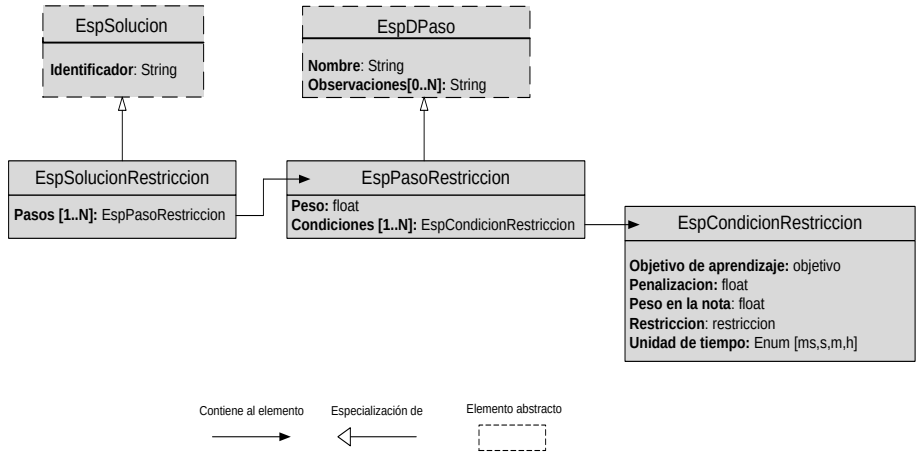


Figura 47: Definición de los elementos EspSolucioRestriccion, EspPasoRestriccion y EspCondicionRestriccion

En cuanto al EspPasoRestriccion, contiene los siguientes atributos:

- *Peso*: Define el peso que tiene un paso dentro de una situación. Si por ejemplo se está evaluando una situación de saque tenis, el paso de golpear la pelota tendrá mayor peso que el paso de lanzamiento de la pelota.
- *Condiciones*: Cada paso contiene una lista de condiciones. El cumplimiento de estas condiciones será la que determine la corrección de un paso.

Por último, el EspCondiciónRestricción queda definido de esta manera:

- *Objetivo de aprendizaje*: Cada condición está relacionada con un objetivo de aprendizaje que se ha definido en la Tarea. El cumplimiento de las condiciones permite al Subsistema de Diagnóstico discernir en qué medida se está obteniendo un objetivo de aprendizaje por parte del alumno.
- *Penalización:* Cuando la condición no se cumpla, el valor marcado por este atributo será el que penalicé el objetivo de aprendizaje.
- *Peso en la nota*: El no cumplimiento de una condición penalizará el paso con el valor contenido en este atributo.
- *Restricción*: Representa la restricción que debe satisfacer en la condición actual. Los elementos de restricción que se utilizan en las

condiciones son los mismos que los que se han definido en el Nivel de Interpretación. Sin embargo, en este nivel su objetivo es la detección de errores del alumno.

- *Unidad de tiempo*: Si alguna restricción incluye algún dato temporal, este atributo indicará las unidades en las que tiene que expresarse. La unidad de tiempo predefinida es el milisegundo

3.5.3.3 Proceso de evaluación de restricciones

En cada ciclo de diagnóstico, por cada situación que se encuentre activa, se llama a la función *Diagnosticar()* (Figura 48) que implementa la técnica especifica de la solución. Por ejemplo, una técnica basada en la satisfacción de restricciones. A su vez, la solución verificará qué pasos han sido marcados por el proceso de diagnóstico de ULISES para que no sean diagnosticados por su estado de incertidumbre actual. Excepto en estos casos, la solución indicará a los pasos que evalúen sus condiciones. Aunque no haya que diagnosticar un paso, la técnica de diagnóstico basada en restricciones gestiona las observaciones que forman parte de las condiciones contenidas en el paso. El proceso de gestión de observaciones es el mismo que se lleva a cabo en la interpretación: mientras una observación no impida que una restricción se cumpla no se descartará. De la misma manera, cuando el Subsistema de Observación indique que una observación ha sido confirmada, el proceso de evaluación de la técnica se encarga de actualizar las instancias de esa observación que se hayan referenciado en las restricciones.

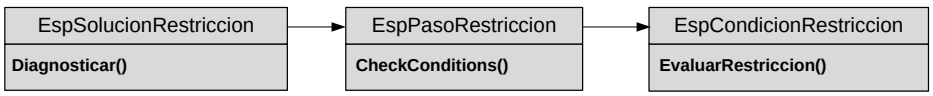


Figura 48: Llamadas en el proceso de diagnóstico de la técnica basada en la satisfacción de restricciones

Una vez gestionadas las observaciones, se evalúan las restricciones de las condiciones de cada paso. Los posibles resultados de evaluación de una restricción son: "Verdadero", "Falso", "Esperar_Datos" y "Esperar_Final". Después de completar el proceso de cada ciclo, el proceso de la solución específica devuelve los resultados de cada Paso

diagnosticado al Subsistema de Diagnóstico para que gestione los resultados.

3.6 APLICACIÓN DE LA METODOLOGÍA EN UN DOMINIO PARA EL APRENDIZAJE DE HABILIDADES MOTORAS

En este capítulo se presenta la aplicación y experimentos realizados para validar la evaluación de destrezas motoras con ULISES. La validación busca demostrar que ULISES es capaz de generar un diagnóstico apropiado para las distintas características de una destreza física. Es decir, el SIIAA tiene que ser capaz de detectar errores cometidos y de sugerir correcciones. Para ello el sistema tiene que ser capaz de extraer la semántica de los movimientos y también de lidiar con la incertidumbre de esos movimientos. En primer lugar se presentan los criterios de validación y posteriormente la aplicación realizada para diagnosticar destrezas físicas en el dominio del tenis. Por último se presenta el experimento que se ha llevado a cabo para la validación, seguido de los resultados obtenidos y una discusión sobre ellos.

3.6.1 Criterios de validación

Se considera que en la evaluación de destrezas físicas hay que ir más allá de recomendar una secuencia de pasos o de calcular la distancia a la solución óptima. En la sección 1.3.3.3 de este trabajo de tesis se estudiaban las fortalezas y las carencias de los diagnósticos generados por distintos SIIAA para el entrenamiento de destrezas físicas. En base a ese estudio, el SIIAA se centra en diagnosticar estas cuatro características de los movimientos:

1. Coordinación: La coordinación es obtenida cuando diferentes movimientos del cuerpo son combinados de una manera ordenada.
2. Pose: Se refiere a la configuración intencionada del cuerpo humano.
3. Trayectoria del movimiento: La trayectoria que un punto característico (elemento sobre el que se hace el seguimiento o tracking) sigue durante un periodo de tiempo.

4. Procedimiento de una secuencia de movimientos: Especifica cómo se tienen que secuenciar los pasos llevados a cabo.

Teniendo en cuenta estas características, se han seleccionado diferentes movimientos de tenis para validar la metodología de evaluación de destrezas físicas con ULISES. En la siguiente sección se describe el estudio práctico para la evaluación de destrezas de tenis.

3.6.2 Estudio de caso práctico: Diagnóstico de destrezas en el tenis

En esta sección se describen los modelos de observación, interpretación y tareas para diagnosticar dos destrezas de tenis: el servicio y el golpeo de derecha. En ambos casos, solo te han tenido en cuenta los movimientos de la parte alta del cuerpo (de cintura hacia arriba).

El sistema de captura utilizado ha sido el sensor Microsoft Kinect. El objetivo de este estudio consiste en demostrar que ULISES es capaz de observar, interpretar y diagnosticar destrezas físicas y no en realizar un análisis cuantitativo exhaustivo de los movimientos. Por lo tanto, se ha considerado que el sensor Kinect posee la precisión suficiente para el cometido.

La creación de modelos se ha hecho utilizando la herramienta de autor PATH, la cual es parte de la plataforma OLYMPUS. En este aspecto, el primer paso para crear los modelos necesarios de ULISES es la definición de la Tarea, donde se deciden las situaciones y los pasos que se van a diagnosticar. Estas son las situaciones que se han definido para diagnosticar las dos destrezas citadas anteriormente:

- *Situación de juego*: Es la situación en la que se encuentra un jugador de tenis inmediatamente después de ejecutar un saque.
- *Situación de saque*: Se trata del golpeo que da comienzo a un punto.

Y estos son los pasos que componen estas situaciones:

- *Golpeo de derecha con efecto* (situación de juego): Se trata de un golpeo especial para añadir efecto a la pelota. Cuanta más grande sea la curvatura de la trayectoria de la raqueta, mayor será el efecto añadido a la pelota.

- *Swing delantero* (situación de saque): Este es un movimiento donde la raqueta se mueve por delante del cuerpo. En un saque correcto la raqueta debería empezar su trayectoria por detrás del cuerpo. Si en cambio, el comienzo del movimiento de la raqueta se hace en la parte anterior del cuerpo, este movimiento será considerado incorrecto.
- *Lanzamiento de bola* (situación de saque): Este paso representa el lanzamiento al aire de la pelota de tenis. Los alumnos deberían de empezar a mover el brazo lanzador al mismo tiempo que empiezan a mover la raqueta. Además, cuando se va a soltar la pelota al aire, el brazo debe estar extendido.
- *Follow through* (situación de saque): Representa el movimiento que se ejecuta después de golpear la pelota. La raqueta debe seguir moviéndose hasta el lado del brazo con el que se ha lanzado la pelota. A consecuencia de la inercia rotacional del tronco, cuando se lleva a cabo este movimiento el cuerpo debe acabar paralelo a la red.
- *Swing posterior* (situación de saque): El brazo que ejecuta el golpeo empieza su trayectoria con la raqueta apuntando al suelo y es levantada por la parte posterior del cuerpo. Cuando comienza este movimiento el cuerpo debe estar en perpendicular a la red. De esta manera el jugador podrá rotar el cuerpo hacia adelante mientras se ejecuta el golpeo, añadiendo así mayor velocidad a la raqueta.
- *Alzar brazo lanzador* (situación de saque): Al lanzar la pelota la mano debe elevarse como mínimo a la altura de la cabeza.
- *Postura trofeo* (situación de saque): Este paso es el paso final de la preparación del saque. Un jugador se encuentra en esta posición cuando su brazo se encuentra extendido verticalmente y la raqueta y el brazo que sujeta la raqueta forman una "L". La excesiva prolongación temporal de este paso provoca una pérdida de energía elástica. Además, cuando el brazo esta en forma de "L" el codo no debe posicionarse a una altura demasiada baja (para evitar lesiones).

3.6.2.1 Modelo de Observación

El Modelo de Observación define las observaciones que se necesitan en el Modelo de Interpretación y en el Modelo de Tareas. Estas

observaciones son calculadas en base al esqueleto compuesto por 20 articulaciones que viene calculado por el sensor Kinect. La siguiente figura muestra las articulaciones sobre los que el Kinect hace el seguimiento (tracking):

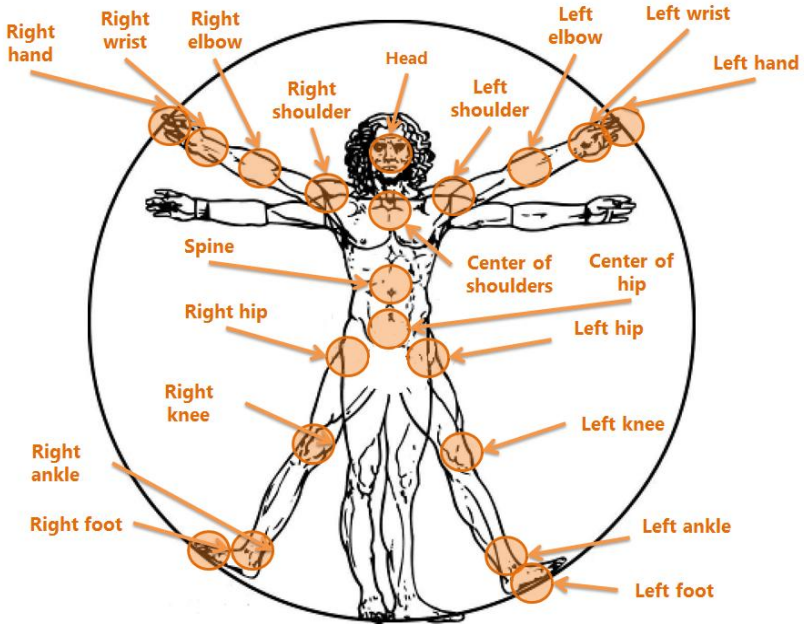


Figura 49: Las 20 articulaciones que forman el esqueleto calculado por Kinect

Basándose en el en tracking de estas 20 articulaciones, en el Modelo de Observación se han definido tanto EspObservaciones como EspObservacionesDifusas. El Subsistema de Observación hace el cálculo de las observaciones difusas basándose en un sistema de coordenadas cuyo origen se encuentra posicionado en la articulación del centro de la pelvis. En la siguiente tabla se describen las observaciones que forman parte del Modelo de Observación:

Tabla 1: Modelo de Observación

Observación	Propiedades	Description
MovimientosMano-Izquierda	• PosZ • PosY • PosX • Arco	Estas observaciones difusas son utilizadas para representar las 26 clases de movimientos que se pueden hacer con la mano izquierda: {MovimientoManoIzquierda001...MovimientoManoIzquierda111} Las propiedades de cada observación son las posición de la mano izquierda y el arco que describe el movimiento.
(Difuso) MovimientosManoDerecha	• PosZ • PosY • PosX • Arco	Estas observaciones difusas son utilizadas para representar las 26 clases de movimientos que se pueden hacer con la mano derecha.
FlexionBrazo-Izquierdo	• Ángulo	Esta observación indica que el brazo izquierdo está flexionado. Su propiedad indica el ángulo de flexión del brazo.
FlexionBrazoDerecho	• Ángulo	Esta observación indica que el brazo derecho está flexionado. Su propiedad indica el ángulo de flexión del brazo.
Cuerpo	• Ángulo	The angle property of the body observation indicates the body`s angle relative to the Kinect. This is used to know if the trainee is parallel or perpendicular to the net.
Cabeza	• Altura	Esta observación representa la posición de la cabeza.
BrazoLanzadorAlto	• Altura	Esta observación se genera cuando la mano que lanza la pelota se encuentra más alta que la altura de la cabeza.
CodoIzquierdo	• Ángulo	La propiedad Ángulo de esta observación representa el ángulo formado entre el brazo izquierdo y el cuerpo. Cuanto más cerca esté el codo del cuerpo el ángulo será menor.
ManoIzquierda	• PosX • PosY • PosZ	Indica la posición de la mano izquierda cuando la mano izquierda es observada.
Tracking		Esta observación indica que el alumno está siendo trackeado, es decir, la información de las articulaciones está siendo calculada.

3.6.2.2 Modelo de Interpretación

En este modelo se definen todos los pasos que son necesarios para llevar a cabo la interpretación. Algunos pasos son definidos con el objetivo de ser diagnosticados (los pasos definidos al comienzo de la sección 3.6.2) y

otros son definidos porque son parte de la definición de otros pasos. La Tabla 2 describe la definición de cada paso del Modelo de Interpretación. Por cada paso se muestran las restricciones que componen cada paso, con las observaciones e instancias necesarias para definirlas.

Tabla 2: Definición de los pasos del Modelo de Interpretación

Step	Observations	Instances	Constraints	Description
BrazoDetras	ManoIzquierda	hl	*General:* hl.PosZ<0.0	Este paso se da cuando la mano izquierda se encuentra detrás del cuerpo.
Comenzar Lanzamiento	FUZZY::ManoDerecha010 FUZZY:: ManoDerecha 011 FUZZY:: ManoDerecha -111	handUp handFU handRFU	*General:* handUp.Arco(>0.0, < 1.0, >0.4) OR handFU.Arco(>0.0, < 1.0, >0.4) OR handRFU.Arco(>0.0, < 1.0, >0.4)	Este paso comienza cuando el alumno comienza el movimiento para lanzar la bola. La propiedad Arco(>0.0, < 1.0, >0.4) indica que el radio del movimiento tiene que ser menor que 1 metro y que la distancia linear del arco tiene que ser mayor de 0.4 metros. Este último parámetro permite despreciar los movimientos insignificantes. *Nota: El prefijo FUZZY se utiliza para indicar que la observación es de tipo difuso.*
PosturaTrofeo	ManoIzquierda FlexionBrazoIzquierdo	hp laf	*General:* laf AND hp.PosZ < 0.0	La postura de trofeo se detecta cuando el brazo está flexionado (~90º) y detrás del cuerpo.
BrazoLanzador Alto	TossHandHigh	hbh	*General:* hbh	Este paso se detecta cuando la mano del brazo lanzador se encuentra más alta que la cabeza..
BrazoDelante	HandLeftPosition	hl	*General:* hl.PosZ > 0.0	Este paso se detecta cuando la mano está delante del cuerpo.
FollowThrough	FUZZY::ManoIzq0-10 FUZZY:: ManoIzq -1-11 FUZZY:: ManoIzq 0-11 FUZZY:: ManoIzq -1-10	b c d e	*General:* b.Arco(>0.0, < 1.0, >0.28) OR c.Arco(>0.0, < 1.0, >0.28) OR d.Arco(>0.0, < 1.0, >0.28) OR e.Arco(>0.0, < 1.0, >0.28) *Accion: b OR c OR d OR e*	Como este paso es el último que se realiza en un saque, se define una restricción de acción para que el paso se diagnostique antes de que la situación acabe.
SwingPosterior	FUZZY::ManoIzq 11-1 FUZZY::ManoIzq 110 FUZZY::ManoIzq 01-1 FUZZY::ManoIzq -11-1 FUZZY::ManoIzq 010 ManoIzquierda	hlUB hlUF hlUN hlUB2 hlU hp	*General:* (hlUB OR hlUF OR hlUN OR hlU OR hlUB2) AND hp.PosZ < 0.0	El SwingPosterior comienza cuando la raqueta empieza a moverse hacia atrás..
SwingAnterior	FUZZY::ManoIzq 010 FUZZY::ManoIzq 011 ManoIzquierda	handUp handFU hl	*General:* handUp.Arco(>0.0, < 1.0, >0.3) OR handFU.Arco(>0.0, < 1.0, >0.3)	Este paso se puede ejecutar por delante o por detrás del cuerpo. Aunque la ejecución de este paso delante del cuerpo es considerada errónea, tiene que ser interpretado sin considerar su corrección.
GolpeoDerecha	FUZZY::ManoIzq -111 FUZZY::ManoIzq 011 FUZZY::ManoIzq -101 FUZZY::ManoIzq 001	hl hl2 hl3 hl4	*General:* hl1.Arco(>0.0,>0.0,>0.35) OR hl2.Arco(>0.0,>0.0,>0.35) OR hl3.Arco(>0.0,>0.0,>0.35) OR hl4.Arco(>0.0,>0.0,>0.35)	La propiedad Arco(>0.0,>0.0,>0.35) indica que su distancia linear tiene que ser mayor de 0.35 metros. De esta manera se ignoran movimientos insignificantes.

3.6.2.3 Modelo de Tareas

En el Modelo de Tareas se describen las dos situaciones que componen la tarea: el juego y el saque. Cada situación está compuesta por una solución basada en restricciones. En aras de la simplicidad, en la Tabla 3 no se muestran las soluciones completas. Solo se muestran los pasos, las condiciones que componen los pasos, sus correspondientes restricciones y una breve descripción de lo que se quiere corregir con cada paso. En la descripción también se indica el tipo de característica en la destreza que se diagnostica: coordinación, trayectoria pose o procedimiento. El paso GolpeoDerecha pertenece a la solución correspondiente a la situación de juego, mientras que el resto de pasos pertenecen a la solución de la situación de saque.

Tabla 3: Definición de las situaciones y los pasos que componen sus soluciones

Situation: Step	Conditions	Observations:instances	Constraints	Description
Juego: GolpeoDerecha	Trayectoria raqueta	FUZZY::ManoIzq-111 : hl1 FUZZY::ManoIzq011 : hl2 FUZZY::ManoIzq-101 : hl3 FUZZY::ManoIzq001 : hl4	hl1.Arco(>90.0, > 0.1, >0.0) OR hl2.Arco(>90.0, > 0.1, >0.0) OR hl3.Arco(>90.0, > 0.1, >0.0) OR hl4.Arco(>90.0, > 0.1, >0.0)	Para añadir el efecto suficiente a la pelota, el arco del movimiento efectuado tiene que ser mayor de 0.1 metros. Además, el movimiento tiene que ser cóncavo. *Características:* Trayectoria
Saque: SwingDelantero	Posición brazo	ManoIzquierda: hp	hp.PosZ < 0.0	La raqueta se tiene que mover por la parte posterior del cuerpo hasta que golpee la pelota. Si el paso SwingDelantero se da cuando la raqueta se encuentra en frente del cuerpo el paso será considerado incorrecto. *Características:* Trayectoria
Saque: PosturaTrofeo	1)Altura codo 2)Duración	1)CodoIzquierdo: le 2)FlexionBrazoIzquierdo:laf	1)le.Angulo > 80.0 2)Duracion(laf) < 500	Cuando el cuerpo se encuentra en la PosturaTrofeo, el codo no debe estar demasiado bajo porque se pierde eficiencia y puede causar lesiones. Si el brazo permanece demasiado tiempo en esta postura se pierde energía elástica. *Características:* Postura, trayectoria.
Saque: Lanzamiento	1)Coordinación	1) STEPS::Lanzamiento: rtb STEPS::SwingPosterior: bss	1) Duracion(rtb.Start,bss.Start) < 200 OR Duracion(bss.Start,rtb.Start) < 200	Este paso tiene que comenzar a la vez que el SwingPosterior. *Características:* Coordinación *Nota: El prefijo STEPS es utilizado para indicar que la instancia utilizada en la restricción es un paso, no una observación.*

Saque: FollowThrough	1)Posición cuerpo 2)Posición raqueta 3)Levantar brazo 4)Procedimiento	1)Cuerpo: ba 2) FUZZY::ManoIzq0-11: hl FUZZY::ManoIzq-1-11: hlb 3) STEPS::AlzarBrazo: ma STEPS::FollowThrough: ft 4)STEPS::PosturaTrofeo: tp STEPS::FollowThrough: ft	1) ba.Angulo < 150.0 AND ba.Angulo > 100.0 2) hl(>= 0.0,>=0.0,>=0.0, >=0.0 ,>=0.0, >=0.0, >=0.0, >=0.0, >=0.0 ,>=0.0, >=0.0, >=0.0, >=0.0, >=0.0, >=0.0, >=0.0, >=0.0, >=0.0, >=0.0, >=0.0, >=0.0, >=0.0, >=0.0,>=0.0, >=0.0,>=0.0, >=0.0 AND <100.0, >=50.0) OR hlb 3) ma [Precede] ft 4) tp [Precede] ft	1) Cuando se ejecuta este paso el cuerpo tiene que estar más o menos paralelo a la red. 2) El movimiento de la raqueta deber continuar hasta llegar al lado del brazo lanzador. Esto significa que el movimiento es diagonal (observación clase ManoIzq-1-11) o similar: la observación ManoIzq0-11 tiene que ser similar a ManoIzq-1-11 en más del 50%. 3) Antes de ejecutar el paso FollowThrough, el paso AlzarBrazo debe haberse ejecutado. 4) Antes del paso FollowThrough, se tiene que haber ejecutado la PosturaTrofeo. *Características:* Postura, procedimiento, trayectoria. *Note: El prefijo FUZZY es utilizado para indicar que las instancia es una instancia de observación difusa.*
Saque: SwingPosterior	1)Posición cuerpo 2)Coordinación	1) Cuerpo: ba 2) STEPS::Lanzamiento: rtb STEPS::SwingPosterior: bss	1)ba.Angulo > 195.0 AND ba.Angulo < 225.0 2)Duracion(rtb.Start,bss.Start) < 200 OR Duracion(bss.Start,rtb.Start) < 200	1) Cuando comienza el saque (comienza con SwingPosterior) el cuerpo tiene que estar colocado perpendicularmente a la red. 2) Este paso tiene que coordinarse con el lanzamiento de la pelota. *Características:* Postura, coordinación.
Saque: AlzarBrazo	1)Extender brazo	1) FlexionBrazoDerecho: rea	1) rea.Angulo > 140.0	Cuando se va a lanzar la pelota el brazo debería estar estirado. *Características:* Postura

3.6.2.4 Experimento

El caso de estudio práctico presentado en esta sección ha sido validado con un experimento. En este experimento se ha realizado un estudio del rendimiento del sistema en el diagnóstico de saques de tenis. En este experimento utilizó un sensor de captura Microsoft Kinect para capturar el movimiento de los sujetos. El experimento estuvo formado por 15 participantes diestros con bajas destrezas de tenis, cuyas estaturas comprendían entre 1,60m y 1,94m (Tabla 4). Cada uno de ellos llevó a cabo 10 saques de tenis satisfactorios con la mano izquierda. Esta última condición fue impuesta con el objetivo de igualar el nivel de destreza entre los 15 participantes. Además, se proporcionó una red de tenis a los sujetos para que tuvieran una referencia estática a la hora de realizar los saques.

Tabla 4: Estatura de los participantes

	Altura
Sujeto 1	1,79
Sujeto 2	1,65
Sujeto 3	1,78
Sujeto 4	1,80
Sujeto 5	1,61
Sujeto 6	1,75
Sujeto 7	1,74
Sujeto 8	1,73
Sujeto 9	1,79
Sujeto 10	1,77
Sujeto 11	1,72
Sujeto 12	1,70
Sujeto 13	1,78
Sujeto 14	1,75
Sujeto 15	1,94

Como el objetivo del experimento es evaluar la capacidad de diagnóstico de los saques de tenis, solo se han tenido en cuenta los saques satisfactorios de los sujetos:

- El sujeto ha llevado a cabo sólo movimientos relacionados con el saque. Se descartan aquellos saques en los que, por ejemplo, el sujeto salta, hace movimientos de calentamiento, se posiciona para realizar el saque…
- El sensor Kinect ha sido capaz de llevar a cabo el tracking de todas las articulaciones necesarias para la correcta interpretación de los pasos que componen el saque.

Para llevar a cabo la validación del sistema, se han medido el número de interpretaciones y diagnósticos correctos por cada paso que compone la situación de saque (Tabla 3).

3.6.2.5 Resultados experimentales

Los resultados experimentales se han obtenido siguiendo este procedimiento:

1. Se grabaron todos los saques, guardando los datos del esqueleto generado por el sensor Kinect.
2. Se reprodujeron todos los saques y se interpretaron y diagnosticaron manualmente. Los criterios para hacerlo se obtuvieron en el sitio web de ITF Coaching[1], asumiendo así el rol de experto. Por ejemplo, el experto registró todos los pasos llevados a cabo por los sujetos y si estos pasos fueron llevados a cabo correcta o incorrectamente. El experto aplicó el mismo criterio que se usó para definir el Modelo de Interpretación y el Modelo de Tareas. Se ignoraron otras apreciaciones subjetivas que no se incluyeron en los modelos, aunque pudieran haberse modelado igualmente. Estos resultados fueron comparados con los resultados de evaluación generados por ULISES (Figura 50). Los resultados de evaluación se obtienen a través de la herramienta de monitorización que es parte de la plataforma OLYMPUS.

En esta herramienta se muestra un cronograma interactivo con todos los pasos ejecutados, representando su corrección mediante distintos colores. Además, clickando en cada paso se pueden ver las condiciones respetadas, no respetadas y la causa de esos errores. La herramienta

[1] (http://en.coaching.itftennis.com/home.aspx)

también muestra las notas de cada paso y la nota de la situación, pero en estas notas no han sido calibradas ni validadas ya que el objetivo del experimento se centra en la detección de errores.

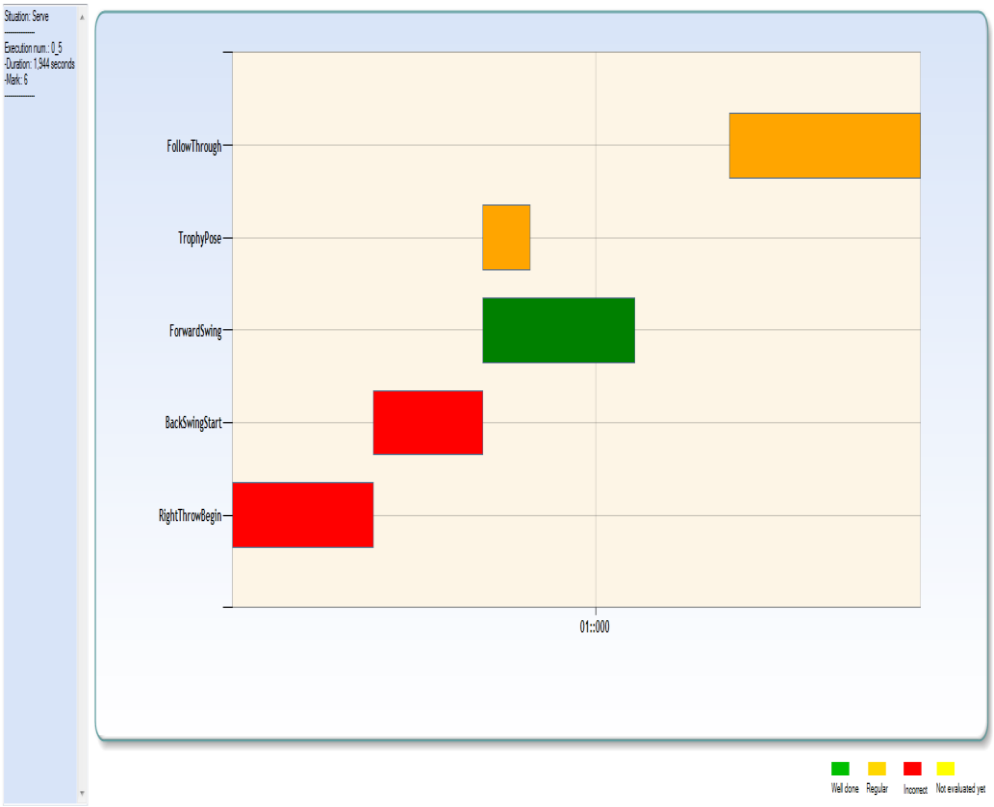


Figura 50: Vista parcial de los resultados de evaluación generados con la herramienta de monitorización de OLYMPUS

La Tabla 5 muestra los resultados experimentales para cada uno de los pasos que tenían que ser diagnosticados dentro de la situación de saque. La precisión de interpretación expresa el porcentaje de pasos interpretados correctamente sobre el total de pasos que los sujetos han ejecutado. La precisión de diagnóstico representa el número de pasos diagnosticados que coincide con el experto real sobre el total de pasos diagnosticados. El total de pasos diagnosticados toma en cuenta solo aquellos pasos que han sido interpretados correctamente.

Tabla 5: Resultados del experimento para cada paso

Nombre de paso	Número de ejecuciones del paso	Número de ejecuciones donde del paso ha sido diagnosticado como incorrecto	Precisión de interpretación	Precisión de diagnóstico
Swing Posterior	128	50	97.3%	100%
Postura Trofeo	109	73	96.3%	98.5%
Swing Delantero	101	51	100%	100%
Lanzamiento	155	117	96.9%	94%
AlzarBrazo	101	87	100%	100%
Follow Through	147	140	98%	98.52%
GLOBAL	741	518	98.1%	98.5%

Asimismo, los resultados han sido desglosados dependiendo de la característica de la destreza que se quería evaluar (Tabla 6):

Tabla 6: Resultados del experimento para cada característica de la destreza física

Nombre de la característica	Número de veces que se ha diagnosticado la característica	Número de veces en la que la característica ha sido evaluada como incorrecta	Precisión de diagnóstico
Coordinación	156	119	92.5%
Postura	473	299	99.7%
Trayectoria	332	147	99.4%
Procedimiento	293	99	99.3%
GLOBAL	1254	664	97.7%

Los resultados experimentales muestran que es posible interpretar (98.1% de precisión) y diagnosticar (98.5% de precisión) los pasos que se han modelado para evaluar un saque de tenis utilizando la técnica de diagnóstico basada en restricciones. Los pasos *PosturaTrofeo* y *Lanzamiento* han sido los pasos con menor precisión de interpretación. La restricción de interpretación no fue modelada con la precisión suficiente, por lo que el paso fue detectado más veces de lo debido. En cambio, el problema con el paso *Lanzamiento* ha sido que algunas veces no se pudo

detectar el paso o que el paso fue detectado de manera tardía. En este caso el error se produjo a la hora de gestionar las observaciones en el nivel de observación.

En lo que se refiere a la precisión del diagnóstico, el paso *Lanzamiento* fue el que obtuvo la menor precisión. Este paso fue modelado para diagnosticar la coordinación entre el paso *Lanzamiento* y el paso *SwingPosterior*; sin embargo, el modelo de diagnóstico no contempla todas las posibles relaciones entre estos dos pasos. Por ejemplo, los casos donde cada paso ha sido ejecutado más de una vez. Por lo tanto, esta circunstancia también ha influenciado en los resultados de diagnóstico del paso *Lanzamiento*, lo que a su vez ha influido negativamente en los resultados de diagnóstico de la característica coordinación (92.5 % de precisión, Tabla 6).

Por otra parte, los resultados muestran que el paso mejor ejecutado fue el paso *SwingPosterior* (50 ejecuciones incorrectas sobre 128 ejecuciones), mientras que el paso que peor se ejecutó fue el paso *FollowThrough.* Este dato es lógico, ya que para llevar a cabo el paso *FollowThrough* correctamente se deben satisfacer cuatro condiciones. Por lo tanto, en comparación con los otros pasos, es más fácil que los sujetos cometieran al menos un error por ejecución en este paso.

Los resultados presentados fueron obtenidos gracias a un proceso iterativo de refinamiento de los modelos de interpretación y tareas. Este proceso de refinamiento fue llevado a cabo basándose en el conocimiento del experto y utilizando las herramientas ofrecidas por OLYMPUS. El experto obtuvo los requisitos necesarios para los modelos de la página de ITF Coaching y los modelos fueron calibrados hasta que se obtuvieron resultados satisfactorios. Los modelos de Interpretación y Tareas fueron definidos para poder evaluar solamente los pasos que se llevan a cabo al ejecutar un saque. Otros movimientos que no eran interesantes para el análisis (por ejemplo posicionamiento antes de hacer el saque) no se contemplaron, porque serían innecesarios y harían que los modelos fueran más complejos. Por lo tanto, los movimientos que no eran contemplados por los modelos pero se interpretaban cómo tal fueron ignorados (en total 49 veces). Si se quisiera, los modelos pueden extenderse para tener en cuenta este tipo de movimientos. Además de

los pasos ignorados por el posicionamiento (u otros movimientos no relacionados con el saque), se ignoraron otros 75 pasos debido a las oclusiones generadas por el Kinect. La mayoría de estas oclusiones ocurrieron a causa del posicionamiento incorrecto de los sujetos, cuando el brazo que sujetaba la raqueta se encontraba detrás el cuerpo, fuera del campo de visión del Kinect. Estas oclusiones no supondrían un problema, por ejemplo, en un juego para el Kinect, pero lo son para un sistema cuyo objetivo es generar interpretaciones y diagnósticos satisfactorios. Una posible solución a este problema consistiría en usar más de un sensor Kinect para conseguir mejor campo de visión o utilizar otro sistema MoCap con cámaras rodeando el campo de captura completo.

Como se ha ido comentando a lo largo de este trabajo de tesis, detectar la intencionalidad de los movimientos llevados a cabo por un alumno es una parte clave de este trabajo. Esta intencionalidad es detectada por el subsistema de interpretación, cuyo resultado (98.1% de precisión) indica que el sistema llevado cabo funciona correctamente. Los pasos *Lanzamiento,FollowThrough* y *SwingPosterior* fueron modelados utilizando la propiedad arco de las observaciones, y han sido útiles para detectar la intencionalidad (ver restricciones en la Tabla 3).

Además, estableciendo restricciones en la longitud del arco, el Intérprete ha sido capaz de ignorar los movimientos que eran insignificantes. En el paso *SwingPosterior*, la intencionalidad fue modelada definiendo los posibles movimientos que un alumno puede llevar a cabo cuando intenta llevar a cabo ese paso. En este caso, la detección del paso también es satisfactoria (97.1% de precisión). Éstas son dos maneras diferentes de detectar la intencionalidad en los movimientos, pero definir el Modelo de Interpretación es una tarea subjetiva que depende en el criterio del experto. Utilizando este criterio, existen numerosas maneras de detectar la intencionalidad y el modelado basado en restricciones utilizado en la técnica de diagnóstico ofrece la suficiente flexibilidad para modelarla.

En lo que se refiere al diagnóstico, los resultados muestran que el sistema ha sido capaz de corregir y detectar errores de los cuatro tipos de características de la destreza física (la última columna de la Tabla 3

describe las características que son diagnosticadas en cada paso). Para empezar, la coordinación fue diagnosticada estableciendo relaciones temporales entre pasos: el paso *Lanzamiento* requiere que *SwingPosterior* haya comenzado casi al mismo tiempo. La característica del procedimiento a seguir en el saque fue diagnosticado en el paso *FollowThrough* con una precisión del 99.3%. En este paso se verifica que el brazo lanzador ha sido levantado hasta la altura correspondiente y que el sujeto haya ejecutado la *PosturaTrofeo* antes de que el saque finalice. El diagnostico correcto de estos tres pasos demuestra que la incertidumbre ha sido gestionada adecuadamente, ya que para diagnosticar la coordinación y procedimiento se requiere gestionar las relaciones entre pasos con incertidumbre. Además, las posturas fueron diagnosticadas correctamente en los pasos *SwingDelantero* (verifica que la mano ejecute los movimientos por la parte trasera del cuerpo), *PosturaTrofeo* (verifica la correcta posición "L" del brazo durante el movimiento), *FollowThrough* (verifica si el brazo lanzador ha sido levantado correctamente), *SwingPosterior* y *AlzarBrazo* (verifican el correcto posicionamiento de las partes del cuerpo).

Además de diagnosticar los pasos correctamente en lo referente a las cuatro características asociadas al movimiento, el experimento ha demostrado que los resultados generados por ULISES pueden ser expresados cualitativamente. La Tabla 7 muestra los diferentes mensajes de feedback generados para los pasos:

Tabla 7: Lista de feedback generados a partir de los resultados de diagnóstico

Situación: Paso	**Feedback**
Juego: GolpeoDerecha	Añade más efecto a la pelota, levanta más la raqueta.
Servicio: SwingDelantero	Servicio incorrecto. Tienes que mover la raqueta hacia atrás, no hacia la parte de adelante.
Servicio: PosturaTrofeo	1)Has bajado demasiado el codo al flexionar tu brazo. 2)Has perdido demasiada energía elástica mientras preparabas el golpeo
Servicio: Lanzamiento	Tienes que empezar a lanzar la pelota al mismo tiempo que mueves la raqueta hacia atrás.
Servicio: FollowThrough	1)Tienes que acabar el servicio estando paralelo a la red. 2)Tienes que acabar el servicio correctamente. Debes que llevar a cabo el movimiento complete de la raqueta, llevándola hasta el lado derecho de tu cuerpo. 3)No has levantado la mano lo suficiente cuando has lanzado la pelota. 4)No has flexionado el brazo lo suficiente antes de golpear la pelota.

Servicio: SwingPosterior	Tienes que empezar a lanzar la pelota al mismo tiempo que mueves la raqueta hacia atrás.
Servicio: LevantarBrazo	Tienes que estirar más el brazo cuando vayas a lanzar la pelota..

El objetivo de este experimento era demostrar que el SIIAA implementado es capaz de discernir las acciones que son llevadas a cabo y corregir esas acciones desde un punto de vista educativo. Aunque los resultados del experimento han sido satisfactorios, se deben llevar a cabo otros experimentos para demostrar cuánto contribuye ULISES en el entrenamiento de destrezas físicas.

Por otro lado, se considera que el coste de modelar movimientos se puede reducir. Como se ha citado anteriormente, la definición de los modelos se ha llevado a cabo utilizando la herramienta de autor PATH. Los modelos de Interpretación y de Tareas se han modelado mediante un proceso iterativo, hasta conseguir la precisión de interpretación y diagnóstico suficientes para la validación. Se considera que el coste de modelar restricciones de Arcos y Listas Difusas es alto, sobre todo si los movimientos a ejecutar son muy complejos. Una posible solución a este problema es utilizar un sistema de generación de modelos semiautomático. Un sistema parecido fue propuesto en el trabajo de Fournier-Viger y Nkambou (Fournier-Viger & Nkambou, 2009),pero no lo aplicaron en el ámbito de evaluación de destrezas motoras.

3.7 DISCUSIÓN

En este capítulo se ha presentado una metodología para definir SIIAA basados en ULISES con la capacidad dar asistencia a los estudiantes en el aprendizaje de destrezas físicas (en https://sites.google.com/site/olympusulises/services) se puede encontrar una demostración del trabajo realizado). Esta metodología ha demostrado ser útil a la hora de abordar las particularidades de la evaluación de las destrezas físicas. Por un lado, se ha gestionado satisfactoriamente la incertidumbre relacionada con el movimiento. Además, el modelado del Modelo de Interpretación basado en restricciones junto con la definición de Observaciones Difusas también ha servido para este propósito, ya que de esta manera se permite reflejar

la intencionalidad que un alumno pueda tener cuando ejecuta un movimiento que está aprendiendo.

Por otro lado, el coste de definir un modelo, y modificar o añadir nuevos movimientos es menor que en otras técnicas basadas en el reconocimientos de acciones y movimientos. Estas técnicas suelen utilizar clasificadores, por lo que cuando se redefine o se define un nuevo movimiento, los clasificadores tienen que soportar todo el proceso de aprendizaje otra vez. Es más, a diferencia de ese enfoque, el sistema presentado en este capítulo no depende de las propiedades del dominio donde las acciones se llevan a cabo.

En este capítulo también se ha citado la importancia de transformar las variables cuantitativas del movimiento a un dominio cualitativo para generar resultados de diagnóstico que puedan ser usados otros componentes educativos. Esta transformación también ha resultado satisfactoria gracias a la definición de Arcos en el nivel de interpretación. Gracias a este elemento, es posible corregir un movimiento mediante el uso de elementos expresables verbalmente. Esto es posible porque los arcos permiten definir un movimiento de una manera lo suficientemente simple para que se puedan hacer correcciones verbales. En este trabajo, se han definido las variables ángulo, longitud y radio para definir un Arco y establecer restricciones sobre el mismo, pero se podrían añadir nuevas variables si fuera necesario.

Las características del movimiento que se han utilizado para la evaluación de destrezas físicas (coordinación, procedimiento, trayectoria y postura) también se pueden expandir. Se considera que estas cuatro características son básicas para evaluar una destreza física, pero la flexibilidad y la manera subjetiva en la que se define el Modelo de Interpretación permite la adición de nuevas características.

CAPÍTULO 4

PROCESO DE ADQUISICIÓN DE CONOCIMIENTO PARA ULISES

En el capítulo anterior se ha presentado el metamodelo de ULISES con el que se especifican los Modelos de Observación, Interpretación y Tareas que sirven de entrada al runtime kernel de ULISES. La definición de estos modelos requiere un proceso de adquisición de conocimiento complejo en el que toman parte varios roles. Para hacer frente a este proceso, en este proyecto de tesis se ha definido un proceso de adquisición de conocimiento y se ha diseñado e implementado la herramienta de autor PATH para ejecutarlo.

El proceso de adquisición de conocimiento comienza con la definición de las tareas procedimentales a las que los estudiantes tendrán que enfrentarse. El diseño de estas tareas no es un paso obvio, ya que existen muchos aspectos relevantes que los diseñadores tienen que tener en cuenta para conseguir que sean efectivas educativamente. Cuanto más metodológico sea el proceso de adquisición de conocimiento, los diseñadores tendrán menos preocupaciones técnicas (por ejemplo de programación) relacionados con la creación del SIIAA. Esto significa que pueden centrar su esfuerzo en identificar y diseñar los aspectos educativos de tareas.

Teniendo en cuenta este argumento, PATH ha sido organizado desde su origen de tal manera que ayude a entender la conceptualización del framework ULISES y la plataforma OLYMPUS. Es más, este sistema de autoría introduce en el proceso de adquisición a los expertos en el dominio de aprendizaje cuyas destrezas hay que transferir a los alumnos. Para ello, PATH permite capturar y analizar la actividad de los expertos mientras resuelven la tarea que se va a presentar a los alumnos. De hecho, esta capacidad es fundamental para modelar destrezas físicas. Asimismo, la metodología de adquisición de conocimiento se refleja en PATH para guiar a los diseñadores en el proceso de creación de tareas.

4.1 DESCRIPCIÓN GENERAL

PATH ha sido diseñada para facilitar un proceso de creación de tareas multiusuario, el cual implica a varios roles. Por un lado, los expertos en el dominio (no tienen responsabilidades de enseñanza) y los instructores (dirigen las sesiones de entrenamiento) trabajan con el Sistema Interactivo y no tienen necesidad de conocer el modelado interno de un Sistema de Ayuda al Aprendizaje. Por otro lado, los diseñadores instruccionales (diseñan las tareas para los estudiantes) no necesitan entender el funcionamiento del Sistema Interactivo.

En este contexto, los diseñadores instruccionales inician el diseño de tareas identificando los *objetivos de aprendizaje* que se quieren trabajar con ellas. Desde este punto de partida, tienen que idear las situaciones a las que los estudiantes van a hacer frente y las soluciones que deberán dar a esas situaciones para obtener evidencias de sus progresos respecto a cada objetivo de aprendizaje.

Este proceso podría escribirse en una hoja de papel, pero no sería un proceso inmediato, ni mucho menos. La ventaja de utilizar PATH reside en que promueve la reflexión y planificación cuando los autores están definiendo un ejercicio. Tal como indica Sowa (2011), "*parte del conocimiento en las mentes de la gente puede ser representado mediante proposiciones, la mayoría en forma de imágenes, y el resto en hábitos, vagas intuiciones y en presentimientos que nunca son verbalizados o visualizados*". Contextualizando esta cita en PATH, los instructores saben cómo

interpretar y diagnosticar las acciones de los estudiantes, pero si les preguntamos sobre cómo lo hacen, no siempre tendrán una respuesta evidente para ello. Para solucionar este problema, PATH proporciona mecanismos con los que ayudar a los diseñadores a formalizar y representar las destrezas y el conocimiento del experto.

Asimismo, PATH también aborda algunos principios de diseño que cualquier herramienta de autor debe tratar:

1. Utilización de métodos eficientes, con una relación coste-eficacia baja y que ayuden a ahorrar tiempo.
2. Reducir el conocimiento técnico y disminuir el umbral de destrezas requeridas (destrezas de programación e Inteligencia Artificial).
3. Ofrecer una interfaz de usuario amigable.
4. Ayuda para adaptar el material de enseñanza a necesidades específicas.
5. Diseño de métodos para testear el sistema creado. En el caso de PATH, ofrece una serie de herramientas para testear la corrección de los modelos definidos.

4.2 METODOLOGÍA DE DISEÑO PARA LA CONSTRUCCIÓN DE PATH

Esta herramienta ha sido desarrollada siguiendo una metodología de trabajo centrada en el usuario (*User-centered Design)*(Abras, Maloney-Krichmar, & Preece, 2004). En este tipo de metodología los usuarios finales influyen en el proceso de diseño del producto. La siguiente figura ilustra el proceso que se sigue en las metodologías de diseño y evaluación centradas en el usuario:

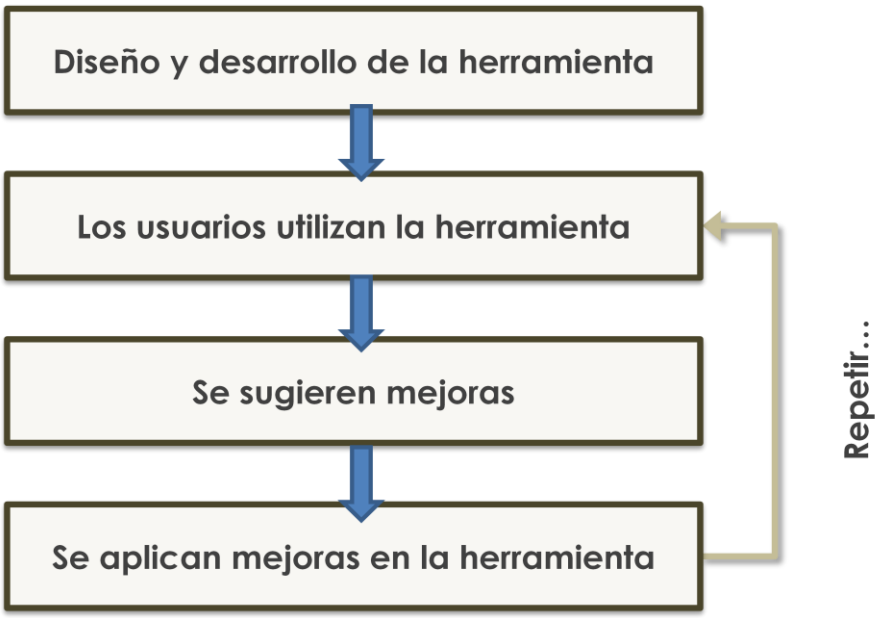


Figura 51: Proceso seguido en el diseño centrado en el usuario

En este caso el papel del usuario lo desempeñaron los diseñadores instruccionales de la empresa Lander Simulation & Training Solutions (http://www.landersimulation.com) dentro del proyecto INTRASIM. Además, se realizaron pruebas de validación adicionales con cuatro sujetos ajenos a dicho proyecto con la intención de aplicar nuevas mejoras a la herramienta.

Esta metodología de diseño y evaluación fue llevada a cabo iterativamente hasta que se logró un resultado satisfactorio.

4.3 PROCESO DE CREACIÓN DE TAREAS MEDIANTE PATH

La creación de tareas mediante PATH está ligada directamente con el framework de ULISES. Es decir, PATH guía a los autores en la creación de los modelos necesarios para ULISES: Modelo de Tareas, Modelo de Interpretación y Modelo de Observación consecutivamente. La creación de las tareas siguiendo ese orden establecido, ayuda a los autores a adquirir los conceptos de la creación de la tarea de una manera más fácil. Por ello, PATH define y promueve la utilización de una

metodología para seguir ese orden. Aunque exista un orden preestablecido, también se debe tener en cuenta que los diseñadores de un ejercicio pueden ir hacia atrás y adelante en el proceso de creación de los modelos. De esta manera los autores pueden mantener la consistencia de un modelo sobre los otros modelos.

Por otra parte, la definición de los modelos de Interpretación y Tareas no es inmediata. Como se ha explicado en el capítulo anterior, las observaciones son la base sobre las que se definen estos dos modelos, pero la extracción de las relaciones entre observaciones y sus valores no es obvia. Esas relaciones no se obtienen a simple vista mientras se observa a un experto resolver una tarea en el Sistema Interactivo. Para resolver este problema, PATH asiste a los autores con una herramienta de captura/monitorización cuyo objetivo es capturar la actividad de los expertos en el Sistema Interactivo. De esta manera, los diseñadores pueden visualizar y analizar el comportamiento de las observaciones generadas por ULISES en los contextos de entrenamiento en los que evaluarán a los alumnos. Mediante este mecanismo, los diseñadores instruccionales pueden definir y calibrar los modelos de Interpretación y Tareas. Este proceso se muestra en la siguiente figura:

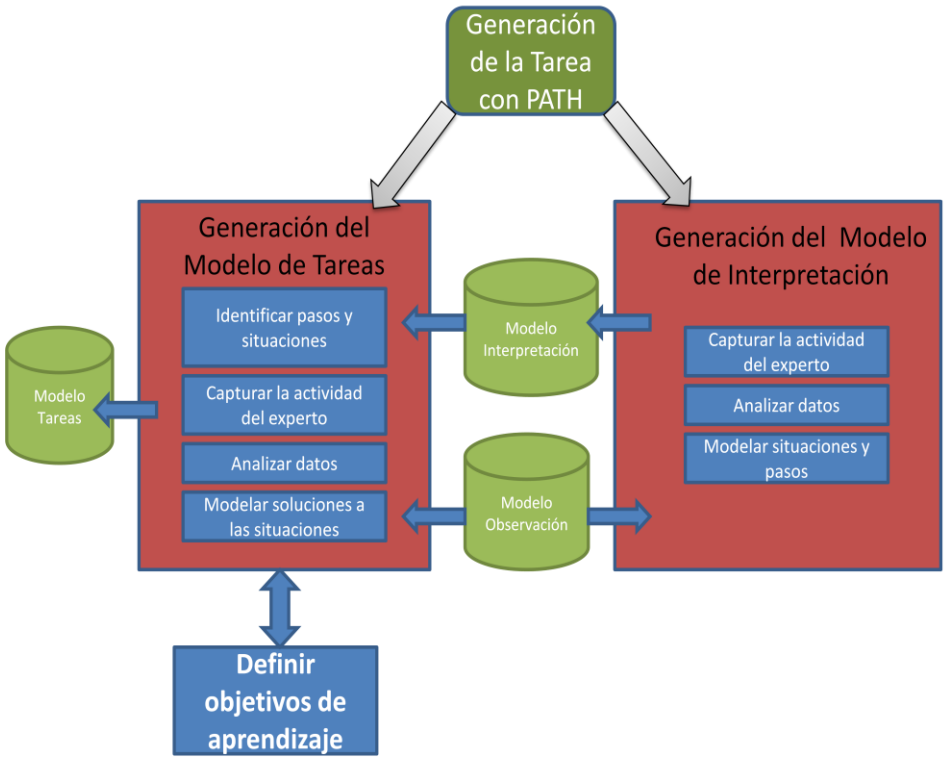


Figura 52: Proceso de creación de Tareas mediante PATH.

Además de ayudar a definir y calibrar los modelos, PATH también ayuda a los autores en la creación de los modelos indicando inconsistencias entre los modelos de Interpretación y Tareas y mostrando los pasos que quedan por completar. Además, PATH ayuda a agilizar este proceso permitiendo la reutilización de los elementos que se definen en los modelos. Por último, la herramienta permite verificar si la tarea creada realiza las funciones para las que fue diseñada correctamente, mostrando para ello los resultados de interpretación y diagnóstico en tiempo real.

En las siguientes líneas se describen los objetivos y las relaciones entre las distintas fases de la creación de una tarea. Asimismo se presentan los procesos asociados para cada uno de los modelos.

4.3.1 Generación del Modelo de Tareas

La generación del Modelo de Tareas es la particularización de los elementos del Nivel de Diagnóstico en el dominio en el que se quiere construir el SIIAA. Por lo tanto, las subfases de este proceso dependen de la técnica (o técnicas) de diagnóstico que se vayan a utilizar. Como las técnicas de diagnóstico específicas no están predefinidas en ULISES, PATH permite una integración fácil de nuevas interfaces para los métodos de diagnóstico que se vayan a utilizar. La compatibilidad de un método de diagnóstico con ULISES está predeterminada por los elementos que componen el Nivel de Diagnóstico. Partiendo de esta restricción, se siguen algunas fases comunes en todo proceso de creación de una tarea: la creación de situaciones, la definición de soluciones para cada situación y la definición de los pasos que llevaran a cabo los estudiantes para resolver una situación.

Seguir el orden citado en la creación de una tarea hace reflexionar a los diseñadores instruccionales sobre las situaciones y los pasos que necesitarán para diagnosticar a los estudiantes, y por lo tanto podrán inferir las situaciones y los pasos que son necesarios también en el Modelo de Interpretación (Figura 53).

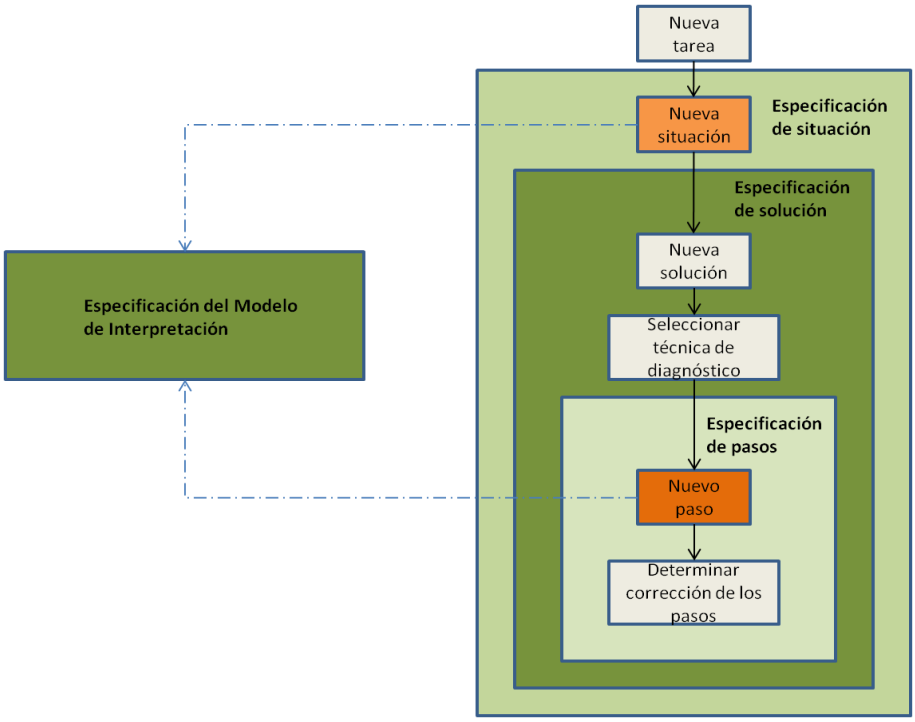


Figura 53: El procedimiento para crear un nuevo Modelo de Tareas

Sin embargo, antes de pensar en crear una situación, el diseñador instruccional tiene que identificar los objetivos de aprendizaje que los estudiantes tienen que abordar. Una tarea no puede ser definida aislando los objetivos de diferentes tipos. En un proceso complejo de aprendizaje, los objetivos de diferente tipo están entremezclados, y por ello una estrategia efectiva debe proponer tareas que las integren y coordinen (van Merriënboer, 1997). Para conseguir esto, PATH permite a los autores la creación de diferentes objetivos de aprendizaje y asignarlos a las tareas. Así, se ofrece a los instructores un medio para saber si un estudiante está progresando en lo referente a un objetivo de aprendizaje específico, que puede servir como base en la definición de nuevas tareas. Cuando se crea un nuevo objetivo de aprendizaje con PATH, se le asigna un nombre, una descripción y el nivel de dificultad para conseguir el objetivo (Figura 54). Además, todos los objetivos de aprendizaje son almacenados genéricamente, por lo que cuando se crea una nueva tarea pueden ser importados y reutilizados.

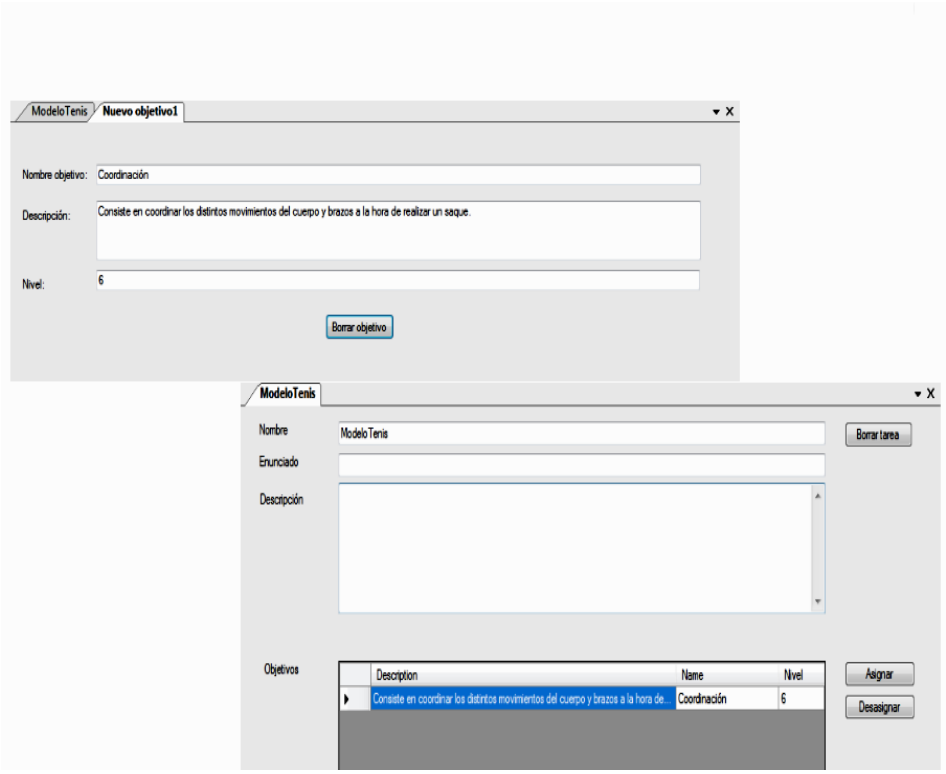


Figura 54: Creación y asignación de un objetivo de aprendizaje

4.3.1.1 Creación de situaciones

Una vez que se han creado y asignado objetivos de aprendizaje a la Tarea, el siguiente paso es crear las situaciones a las que el estudiante se tiene que enfrentar. Más adelante, a una situación habrá que añadirle las posibles soluciones, la cual puede resultar en una tarea costosa y compleja. Por ello, PATH permite reutilizar (importar/exportar) situaciones y soluciones que han sido utilizadas anteriormente. Si ninguno de los elementos importables encaja en la tarea, el diseñador instruccional deberá crear nuevas situaciones, que también serán reutilizables en el futuro. Esta característica de reusabilidad permite a los autores transformar copias de una situación completa (con sus soluciones) en una alternativa deseada. De la misma forma, una solución siempre estará compuesta por pasos, y la especificación de cada paso

puede ser una tarea tediosa. Por esta razón, estos elementos reutilizables pueden reducir significativamente la carga de trabajo de los autores.

Por otro lado, PATH también ayuda a mantener la coherencia entre los modelos de Interpretación y Tareas. Cuando se crea una nueva situación en el Modelo de Tareas, ésta se añade directamente en el Modelo de Interpretación. De esta manera, cuando los autores completen el Modelo de Interpretación tendrán una base para empezar a especificar el modelo.

La Figura 55 muestra una interfaz del Modelo de Tareas, donde se pueden editar y crear nuevas situaciones. Estas son las diferentes áreas:

1. El panel izquierdo permite la navegación entre los modelos de Observación, Interpretación y Tarea. En la imagen se encuentra seleccionada la pestaña del Modelo de Tareas, donde se presenta un árbol con todos los Modelos de Tareas disponibles. A través de este árbol se pueden seleccionar tareas, situaciones y soluciones. A su vez, también se utiliza esta visualización de árbol para importar nuevas situaciones o soluciones.
2. Esta lista desplegable muestra las técnicas de diagnóstico disponible. En este caso se ha creado una técnica basada en el modelado por restricciones (sección 3.5.3) para la solución de la situación de saque (mostrado en la figura como Serve).
3. Interfaz específica para la técnica basada en la satisfacción de restricciones. En esta interfaz el usuario tiene que definir los pasos y sus correspondientes condiciones para la solución de la situación seleccionada. Cuando se editan las restricciones, el usuario tiene la posibilidad de validar si las restricciones son correctas (sintáctica y semánticamente).
4. Se muestran dos tipos de pestañas. Las extensiones ".TAR" (Modelo de Tareas) y ".INT" (Modelo de Interpretación) que se ven en las pestañas identifican el modelo que está siendo editado.

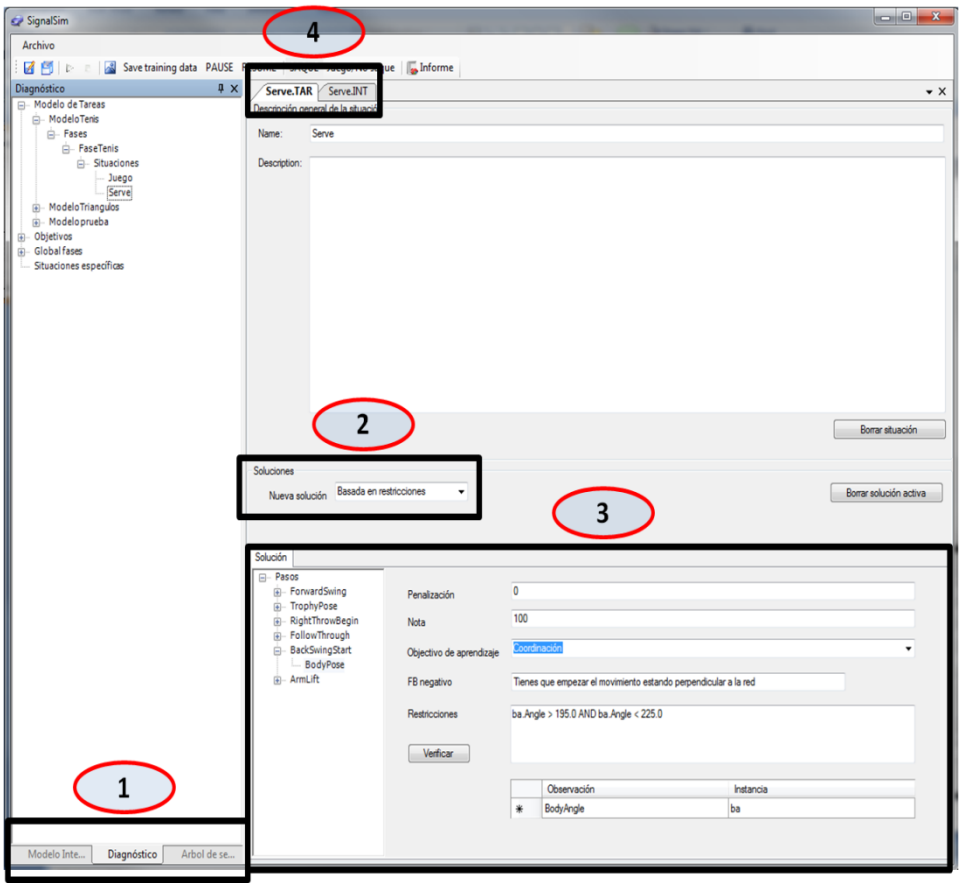


Figura 55: Las zonas que componen la interfaz para completar el Modelo de Tareas.

4.3.1.2 Creación de soluciones

Cuando se crea una situación, el siguiente paso consiste en crear su solución o soluciones. El framework de ULISES permite la integración de diferentes técnicas de diagnóstico. Por esta razón, PATH permite la integración de nuevas interfaces de técnicas de diagnóstico mediante plugins. Estas interfaces tienen que implementar la interfaz específica de la técnica para PATH (Figura 55, Área 3), por lo que cuando son utilizadas (Figura 55, Área 2) la interfaz de la solución es creada dinámicamente. De esta manera, cuando se integra una nueva interfaz para una técnica de diagnóstico, la arquitectura y las interfaces generales de PATH permanecen intactas.

Al pensar en una solución a la situación, inicialmente el diseñador instruccional debe identificar los pasos que los alumnos podrían ejecutar cuando afrontan la situación. Como se ha citado en el capítulo anterior, estos pasos son las primitivas de diagnóstico definidas en ULISES, independientemente de la técnica de diagnóstico que se emplee. Hay que recordar, que la estructura interna de cada solución se encuentra más allá de las especificaciones de ULISES. En vista de esta característica, el procedimiento subsecuente depende de la técnica que se utilice y en la interfaz específica implementada en PATH.

PATH ha sido utilizado en la creación de tareas en tres SIIAA de diferentes dominios:

1. SIIAA para conductores de camión profesionales.

Figura 56: Vista del simulador de camión

2. SIIAA para el entrenamiento de personas con discapacidades cognitivas en tareas de jardinería.

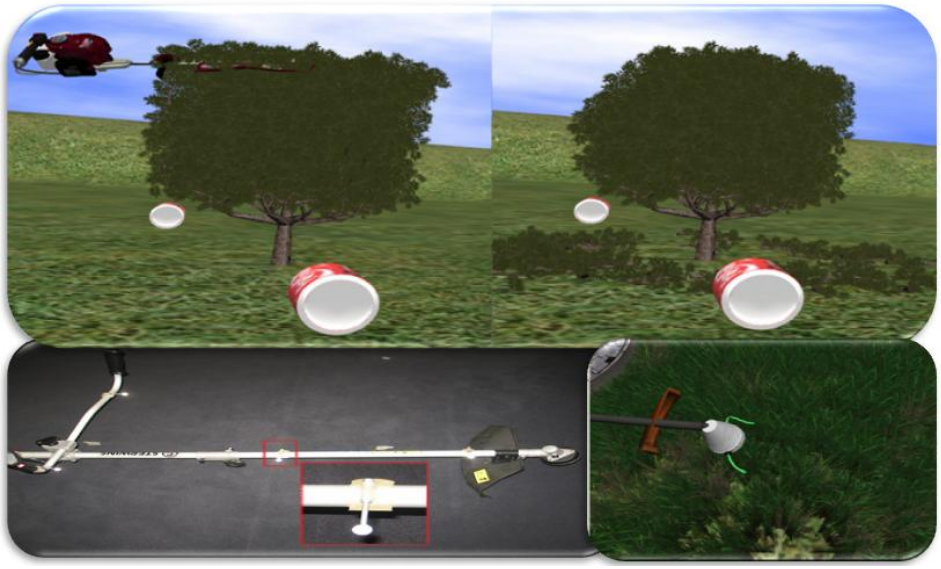


Figura 57: Distintas capturas del SIIAA para tareas de jardinería

3. SIIAA para el entrenamiento de destrezas físicas relacionadas con el tenis.

Figura 58: Capturas de ejemplo del SIIAA para el entrenamiento de destrezas físicas en el tenis

En estos tres SIIAA se optó por utilizar la técnica de diagnóstico basada en restricciones por las características de los dominios en los se emplearon. Por esta razón, hasta el momento ésta es la única interfaz que se ha implementado en PATH.

En esta interfaz se definen los atributos que se han descrito en el capítulo anterior y que componen una EspSolucionRestriccion: los pasos (con su nota) que componen las situación y las condiciones que hay respetar en cada paso para obtener los objetivos de aprendizaje asociados a la tarea. Las condiciones están compuestas por los atributos el objetivo de aprendizaje, penalización, nota y restricción (ver Figura 47 en la sección 3.5.3.2). Al igual que ocurre con las situaciones, cuando se añade un nuevo paso, este paso se vuelca al Modelo de Interpretación para facilitar la definición de los modelos.

Por otro lado, la definición de las restricciones de las condiciones de los pasos requiere cierto grado de experiencia en el uso de las restricciones. Por esta razón PATH ofrece asistencia a la hora de crear restricciones, con el objetivo de minimizar errores y de agilizar proceso de aprendizaje. La herramienta verifica si una restricción es correcta semánticamente y si las instancias utilizadas (observaciones, pasos o situaciones) son correctas y coherentes con los modelos de Observación y/o Interpretación.

4.3.2 Captura de la actividad del experto

Dar expresión al conocimiento del experto del dominio es uno de los objetivos principales de PATH. En este aspecto, el uso de restricciones para definir tanto el Modelo de Tareas como el de Interpretación es un serio obstáculo para los diseñadores instruccionales. La dificultad de definir restricciones en el modelado basado en restricciones es un tema conocido (Fournier-Viger & Nkambou, 2009). En el caso de un SIIAA basado en el framework ULISES, la definición de restricciones para evaluar pasos y situaciones puede ser costosa, ya que en algunos casos se deberán establecer relaciones complejas entre observaciones.

Para facilitar el proceso de definición de restricciones, PATH permite capturar la actividad de los expertos en el Sistema Interactivo y monitorizar los valores de las observaciones y sus propiedades. De esta

manera, los diseñadores instruccionales pueden analizar el comportamiento de las observaciones y obtener patrones y relaciones entre observaciones capturadas en una situación.

La herramienta de captura de PATH permite la creación de distintos tipos de gráficas, donde se pueden arrastrar las observaciones deseadas para visualizar el valor de sus propiedades mientras se captura la actividad del experto en el Sistema Interactivo. El primer tipo de gráfica es 2D y se emplea para monitorizar valores numéricos durante el tiempo. En la Figura 59 se muestra la captura de la actividad de un experto mientras conduce un simulador de camión. Como se puede apreciar, se permite la adición de múltiples gráficas alineadas al mismo tiempo, ya que de esta manera se pueden apreciar mejor las relaciones entre distintas observaciones. En este caso, la gráfica superior muestra los valores de las propiedades de velocidad del camión y el porcentaje de acelerador pisado, las cuales pertenecen a la observación *Vehiculo*.

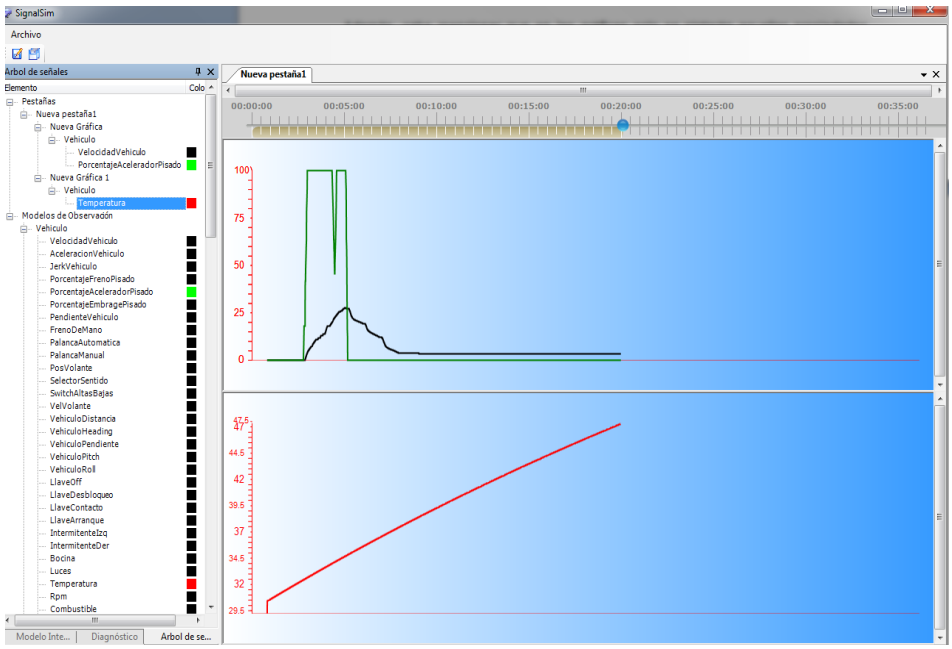


Figura 59: Captura de pantalla de dos gráficas 2D

El segundo tipo de gráfica es una gráfica 3D donde se pueden monitorizar las propiedades de un movimiento, como pueden ser la

posición de una articulación o la aproximación del movimiento a un Arco. Asimismo, gracias a esta gráfica se puede saber qué clase de movimiento está llevando a cabo el experto en todo momento, lo cual es indispensable a la hora de crear restricciones relacionadas con el movimiento.

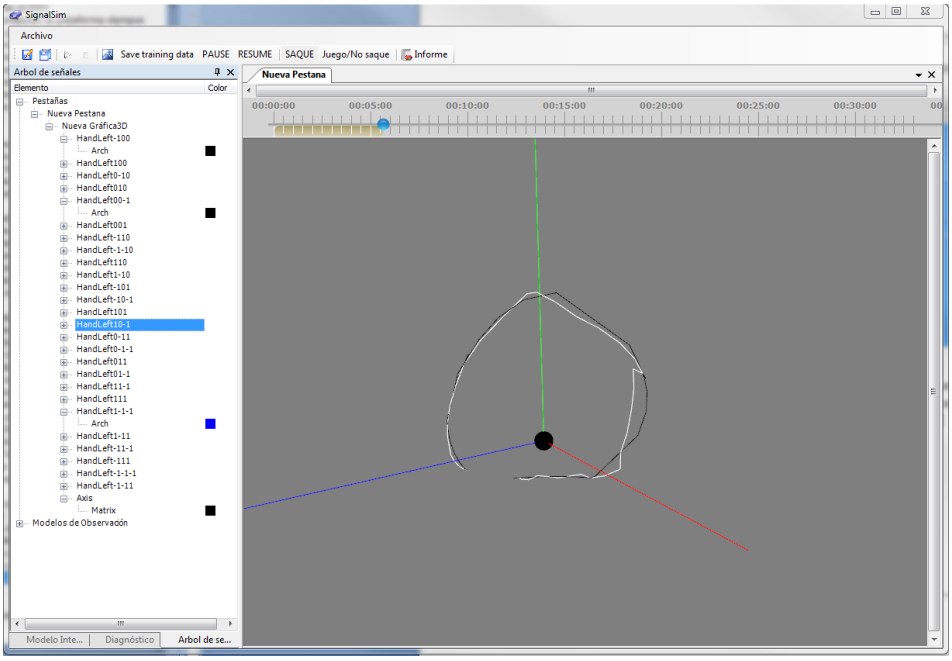


Figura 60: Gráfica 3D para la captura de movimientos.

La tercera y última gráfica sirve para mostrar el esqueleto del experto mientras ejecuta los movimientos. Gracias a este esqueleto virtual se pueden relacionar las observaciones con los movimientos que se están llevando a cabo en todo momento (Figura 61).

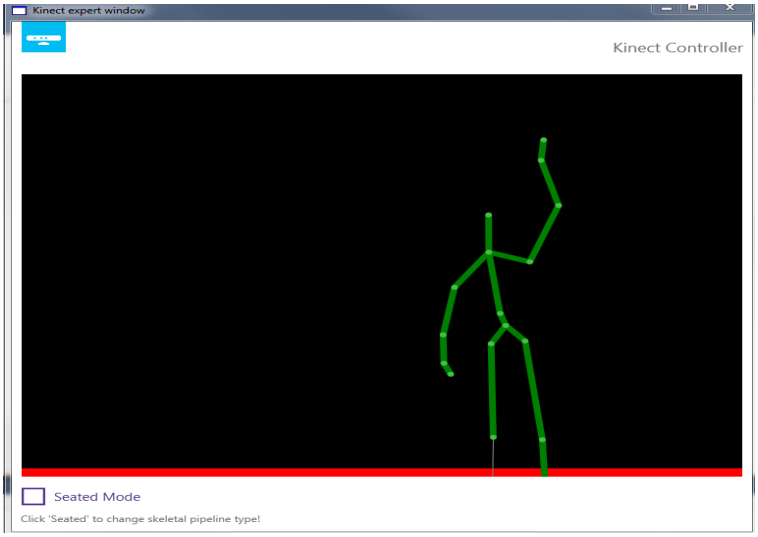


Figura 61: Interfaz para capturar los movimientos del experto

Además de poder mostrar la actividad del experto en el Sistema Interactivo de diferentes formas, PATH permite guardar las sesiones de captura para que los diseñadores puedan reproducirlas más tarde. De esta manera pueden analizar la actividad del experto más exhaustivamente sin que los expertos tengan que repetir las capturas continuamente.

4.3.3 Generación del Modelo de Interpretación

La creación del Modelo Tareas es el paso anterior a la definición del Modelo de Interpretación. Gracias a este primer paso, los diseñadores instruccionales reflexionan sobre los pasos y las situaciones que se necesitan modelar en una Tarea. Una vez que se ha llevado a cabo este proceso, el siguiente paso corresponde a la definición del Modelo de Interpretación. En este modelo se deberán modelar los pasos y situaciones que hay que interpretar para que se puedan diagnosticar los pasos y situaciones definidos en el modelo anterior.

4.3.3.1 Definición de pasos y situaciones

PATH ofrece un guiado sencillo en la definición de pasos y situaciones. Como se ha citado anteriormente, al crear un paso o situación en el Modelo de Tareas, estos pasos son creados automáticamente en el

Modelo de Interpretación. Además, PATH los marca como "incompletos" (Figura 62). Lo mismo ocurre con los pasos y situaciones que son creados manualmente por el usuario. Así, se consigue que la primera tarea que haga el instructor (después de definir el Modelo de Tareas) al acceder a la interfaz del Modelo de Interpretación sea definir los pasos y situaciones que están incompletos. Esto permite saber a los usuarios de PATH cómo empezar con el Modelo de Interpretación, lo que ayuda a adquirir las bases de la metodología propuesta por PATH y ULISES.

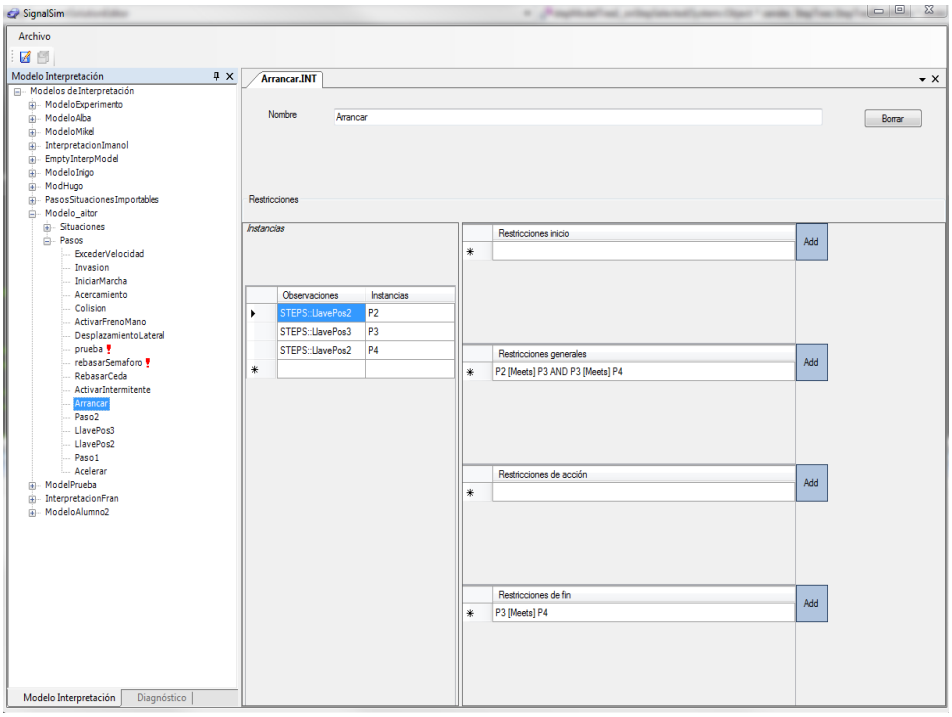


Figura 62: Interfaz del Modelo de Interpretación

Por otra parte, todos los pasos y situaciones que se crean están disponibles para reutilizarlos en futuros modelos. Estos dos elementos pueden ser usados como plantillas, al igual que las situaciones en el Modelo de Tareas. En la Figura 62 se muestra la definición del paso *Arrancar*. En la parte derecha de la pestaña *Arrancar.INT* se definen las instancias de observaciones, pasos o situaciones necesarias, los cuales se utilizan en la definición de las restricciones del paso. El árbol que se

encuentra en el panel izquierdo muestra los Modelos de Tareas disponibles para la edición, con sus correspondientes pasos y situaciones. Los pasos o situaciones que necesitan completarse son mostrados con un signo de exclamación.

4.3.4 Validación de los modelos de Interpretación y Tareas

Un paso necesario en el proceso de creación de una Tarea es la validación de las soluciones que se han definido. Para este propósito PATH ofrece una herramienta de monitorización que monitoriza los pasos y situaciones que se ejecutan en el Sistema Interactivo en tiempo real. En otras palabras, PATH es capaz de monitorizar los resultados de diagnóstico en tiempo real. Esto significa que tanto los instructores como los diseñadores instruccionales pueden visualizar los pasos y situaciones que están siendo detectados en el Sistema Interactivo y también verificar si se están ejecutando correcta o incorrectamente. Esta corrección o incorrección deberá encajar con la visión del experto sobre cómo hay que afrontar una situación.

Por último, hay que destacar que no es necesario acabar toda la tarea para verificar los pasos y situaciones que se han creado; este testeo se puede llevar a cabo cuando los diseñadores instruccionales lo deseen. En la siguiente figura se muestra la herramienta de monitorización durante la ejecución de una tarea de conducción. Los intervalos de pasos y situaciones que se están llevando a cabo se muestran en verde cuando son correctos, y en rojo cuando son incorrectos:

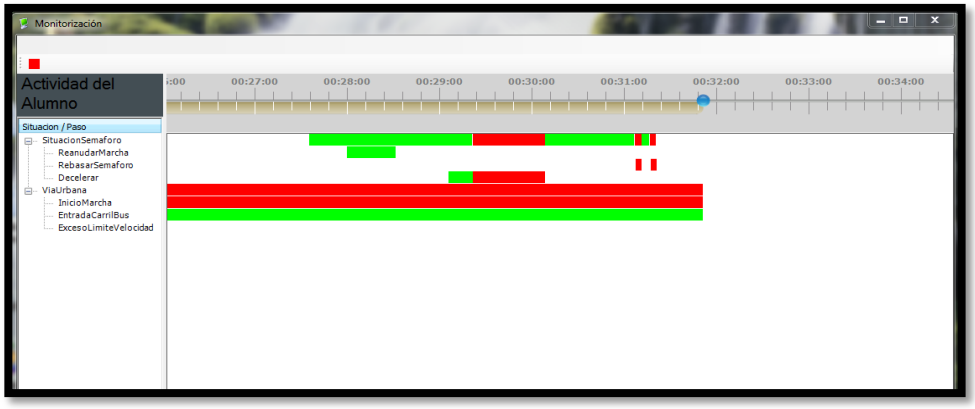


Figura 63: Herramienta de monitorización de PATH.

Una vez que se ha detallado el proceso de creación de tareas con PATH, se va a proceder a describir la validación de la herramienta de autoría.

4.4 VALIDACIÓN DE PATH

PATH ha sido utilizado en el desarrollo de tres SIIAA diferentes: un SIIAA para el entrenamiento de conductores de camión profesionales, otro SIIAA para el entrenamiento de personas discapacitadas en tareas de jardinería (Ostiategui et al., 2010) y un último SIIAA para el entrenamiento de destrezas en el tenis. Diferentes grupos de personas tomaron parte en la creación de cada SIIAA y en los tres SIIAA se obtuvieron resultados satisfactorios.

Con el objetivo de obtener impresiones sobre la creación de tareas con PATH y obtener conclusiones, se han llevado a cabo dos tipos de evaluaciones. En la primera evaluación, se estudian las lecciones aprendidas en los tres SIIAA donde se ha hecho uso de PATH. De su uso se sacan conclusiones y una idea general sobre la usabilidad de PATH en proyectos específicos de distintos dominios. Sin embargo, en esos proyectos no se pudo medir si en realidad la herramienta de autor ayuda a adquirir la metodología propuesta en ULISES. Por lo tanto, se diseñó un experimento para demostrar esta característica de PATH. En los siguientes subapartados se describen tanto las lecciones aprendidas como el experimento realizado.

4.4.1 Lecciones aprendidas

A continuación se describe la implementación de los tres SIIAA con las lecciones que se aprendieron en el desarrollo de cada uno de ellos.

4.4.1.1 Implementación del SIIAA para entrenamiento de conductores de camión

El SI utilizado para el entrenamiento de conductores de camión se trata de un simulador inmersivo altamente realista (Bilbao et al., 2011). PATH se utilizó para crear tareas relacionadas con la seguridad vial y la eficiencia en la conducción. En estas tareas los conductores eran evaluados en diferentes escenarios: puertos de montaña, vías urbanas, autopistas etc. Para llevar a cabo esta evaluación se necesitaron todo tipo

de observaciones: observaciones sobre el vehículo (todas las señales que provienen de la cabina, así como la velocidad, posición de los pedales, estado de los indicadores luminosos…) y otras observaciones sobre los objetos en el entorno virtual (carriles, viandantes, señales de tráfico, vehículos…). La información necesaria para generar estas observaciones con ULISES fue definida utilizando la herramienta de modelado de conocimiento de escenarios 3D que forma parte de OLYMPUS (sección 2.3.1).

En este proyecto se pudo comprobar que sin PATH, la programación a medida para detectar errores de forma precisa resulta costosa. Además, la programación se complica en situaciones que implican relaciones temporales entre acciones, por ejemplo, el mantenimiento de la distancia de seguridad. De esta manera, se comprobó que PATH ahorra gran parte de esfuerzo de programación. Antes de la utilización de OLYMPUS y PATH los programadores tenían que crear soluciones ad-hoc para cada acción que se quería detectar. Es más, no podían reutilizar el mismo código para situaciones similares donde la evaluación de las acciones variaba. Este esfuerzo se ahorra reutilizando las especificaciones de pasos y situaciones.

El trabajo de los diseñadores instruccionales también se agilizaba gracias a PATH ya que no tenían por qué conocer ni entender la programación interna y el funcionamiento del simulador. Además, el uso de las herramientas de captura y monitorización fue de gran ayuda a la hora de definir y refinar los modelos de interpretación y tareas. Antes de la implementación de PATH, el proceso de refinado de los "modelos de programación" era un proceso de prueba y error. Al llevar a cabo el proceso desde la herramienta de autor utilizando sesiones de experto capturadas previamente, la definición y el refinamiento de los modelos resultó mucho más ágil. En lo referente a la creación de las tareas, no supusieron la definición de un gran número de pasos y situaciones, básicamente porque muchos pasos y situaciones se pudieron reutilizar en las diferentes tareas que se crearon.

Por otro lado, las características del dominio de la conducción hicieron que la definición de las restricciones necesarias resultara compleja. En este dominio los conductores interactúan con un gran

número de agentes y objetos. Por ejemplo, si se quiere evaluar si un conductor respeta los límites de velocidad, no es suficiente con verificar si el conductor va a una velocidad menor que el límite establecido por la vía: las restricciones también tienen que tener en cuenta, por ejemplo, las distintas señales que se pueden encontrar en la carretera, vehículos adyacentes o la presencia de viandantes.

En total, la creación de este SIIAA duró cinco meses, teniendo en cuenta la definición de conocimiento en los escenarios, la creación del modelo de observación (integración con ULISES) y los procesos de creación de las tareas. Una vez completada la integración con ULISES, la definición de todas las tareas se completo en dos semanas de trabajo.

4.4.1.2 Implementación del SIIAA para el entrenamiento de personas discapacitadas en tareas de jardinería

El segundo SIIAA fue desarrollado con el objetivo de entrenar alumnos discapacitados en tareas de jardinería. En este caso el SI está compuesto por maquinaria de jardinería real equipada con sensores y un sistema de captura del movimiento para capturar la posición de la maquinaria. Los usuarios de este sistema portan un HMD (Head Mounted Display) que simula interacciones reales. Como el HMD también está equipado con sensores, el sistema de captura de movimiento también deduce la dirección a la que los alumnos dirigen la mirada. En lo que se refiere a los escenarios de simulación, en este caso son mucho más simples que los del simulador de conducción, ya que el área de trabajo de los estudiantes está bastante limitada. Los escenarios son jardines donde los estudiantes pueden moverse libremente mientras llevan a cabo tareas de jardinería como desbrozado, recorte de ramas, podado o eliminación de malas hierbas.

En este proyecto se crearon en total 45 tareas, 56 pasos y 22 situaciones. Teniendo en cuenta todas las tareas, se reutilizaron el 100% de las situaciones y cada situación se reutilizó una media de dos veces. Además se reutilizó el 73% de los pasos, con una media de utilización de 4 veces por cada paso. Los alumnos fueron evaluados en situaciones donde puede existir algún tipo de peligro: trabajo en bordes, con viandantes alrededor, con desbrozadora etc. Si se compara con el SIIAA para el entrenamiento de conductores, el número de observaciones

totales que se definió fue menor. El número de pasos a diagnosticar era mayor, pero las acciones eran más simples de evaluar, ya que muchos pasos estaban destinados a diagnosticar acciones peligrosas a los que los estudiantes debían hacer frente. La simplicidad de estos pasos radica en que la mayoría de las restricciones eran restricciones cuantitativas sobre un solo intervalo, restricciones sobre propiedades y restricciones lógicas. Las siguientes restricciones ilustran la clase de restricciones que se utilizaron en este proyecto:

- Maquinaria.Altura < 0.4 AND Maquinaria.Altura > 0.1 AND Localizacion.AreaID < > 3.0
- Duracion(Aceleracion) < 3000

La primera restricción se utiliza para discernir si el estudiante está trabajando en un área correcta y también para verificar si la altura de la maquinaria es correcta cuando se corta el césped. La segunda restricción se definió para las situaciones donde el alumno no estuviera mirando al césped. Si el alumno no está mirando césped mientras acelera la maquinaria, significa que no está prestando atención a la actividad. Dada la simplicidad de las restricciones, los diseñadores no utilizaron la herramienta de monitorización en exceso, ya que los modelos fueron fáciles de refinar. Cabe mencionar, que este tipo de restricciones son más fáciles de modelar que, por ejemplo, restricciones donde se toman en cuenta múltiples intervalos que están relacionados temporalmente (por ejemplo, el tipo de restricciones utilizadas en el SIIAA para el entrenamiento de conductores). Tanto la creación del Modelo de Observación como la creación de los modelos necesarios para interpretar y diagnosticar las tareas duraron dos semanas cada uno.

4.4.1.3 Implementación del SIIAA para el entrenamiento de destrezas físicas en el tenis

El tercer y último SIIAA se trata de un sistema para el entrenamiento de destrezas motoras del tenis, el cual ha sido ampliamente descrito en el capítulo anterior de este trabajo de tesis. En este SIIAA se creó una tarea, compuesta por 2 situaciones y 12 pasos. Estos pasos se centraron en evaluar la coordinación, procedimientos, posturas, y trayectorias de distintos puntos característicos del cuerpo. En este caso, la mayoría de

restricciones tienen en cuenta restricciones temporales entre intervalos y en ellas también se modelan movimientos. Estos últimos se definen añadiendo restricciones sobre las propiedades de los movimientos (arcos y listas difusas). Se observó que la creación de estos tipos de restricciones tuvo un alto coste, ya que algunos movimientos que se querían diagnosticar eran complejos, al igual que las relaciones entre los intervalos. Dada esta complejidad, el proceso de refinación de los modelos fue largo ya que hubo que capturar y monitorizar muchas veces al experto. En su estado actual, la creación del SIIAA duró unos 40 días. La creación del modelo de observación fue más rápida que en los dos anteriores SIIAA (3 semanas), pero la creación de los modelos duró 20 días.

4.4.1.4 Conclusiones

En vista de los datos obtenidos en estos tres SIIAA, se puede afirmar que el esfuerzo de crear una tarea con PATH depende de varios factores. El primer factor que afecta es el tamaño de los modelos. Normalmente, cuanto mayor sea el tamaño de los modelos, el número de restricciones que hay que definir será mayor. Igualmente, las características del dominio donde se va a construir el SIIAA afectan a la complejidad del proceso de creación de tareas. Estas características establecen la dificultad de modelar un dominio formalmente. En lo que se refiere al modelado de restricciones, está directamente influenciado por las características del dominio y por la precisión que se necesita para el diagnóstico. Si la evaluación de una acción necesita ser muy detallada, se requerirá la definición de más restricciones, por lo tanto el coste de generar la tarea será más alto. Por otro lado, la especificación de las tareas es subjetiva, es decir, existen muchas maneras de definir los pasos y las situaciones. Por ello, algunos diseñadores instruccionales tardarán más que otros al definir una tarea, dependiendo de su experiencia y habilidad.

Al margen de la complejidad de definir una tarea, su proceso de creación es acelerado gracias a PATH. Para empezar, PATH abstrae al usuario de las variables de simulación y del funcionamiento interno de los simuladores y evita la programación en la creación de los modelos de interpretación y tareas. Además, PATH permite la reutilización de pasos

y situaciones en diferentes tareas. Esta reutilización dependerá de las similitudes que compartan las diferentes tareas que vayan a resolver los alumnos. Por último, gracias a las herramientas de captura y monitorización el proceso de refinado de los modelos es acelerado en comparación con los métodos clásicos de prueba y error que se usan en la programación de simuladores con fines educativos.

Debido a todos estos factores, no es factible obtener unas métricas objetivas sobre el coste general de la creación de tareas con PATH, ya que la creación de cada SIIAA tiene su propia problemática. A pesar de ello, PATH ha demostrado ser útil en tres SIIAA de diferentes dominios y en caso del SIIAA para el entrenamiento de conductores de camión, se ha constatado que PATH acelera el proceso de creación de tareas. De todas formas, en estos tres SIIAA no se pudo medir si en realidad PATH ayuda a los diseñadores instruccionales novatos en la metodología de ULISES a asimilar la metodología propuesta por ULISES. Por esta razón, se propuso el experimento que se describe en las siguientes líneas.

4.4.2 Experimento

Se llevó a cabo un experimento para investigar el tipo de problemas que pueden surgir en la utilización de PATH y para evaluar cómo ayuda en el entendimiento general de la metodología propuesta con ULISES. Para este propósito, se diseñó un experimento en el que participaron 16 ingenieros; los 16 compartían las siguientes características:

- No tenían conocimientos sobre el modelado basado en restricciones
- No tenían experiencia previa en la representación de conocimiento de SIIAA
- No conocían la arquitectura ni la metodología de ULISES
- No estaban familiarizados con simuladores de entrenamiento ni entrenamiento de personas
- Todos tenían un nivel profesional similar como ingenieros de distintas especialidades

El experimento consistió en modelar una tarea para evaluar a los conductores en un simulador de camión. En otras palabras, los sujetos tomaron el rol de diseñadores instruccionales. Además, se les

proporcionó un experto para que pudieran capturar su actividad cuando lo requirieran. Los sujetos se dividieron en dos grupos: ocho utilizaron PATH para definir la tarea y los otros ocho definieron la tarea en papel. Los 16 participantes recibieron una serie de especificaciones que la tarea tenía que cumplir para evaluar una serie de destrezas. Con estas especificaciones, el objetivo de los sujetos era crear los Modelos de Interpretación y Tareas, definiendo los pasos, situaciones y sus soluciones correspondientes. Como se ha citado anteriormente, PATH permite incluir en el modelo los objetivos de aprendizaje y el cálculo de notas para evaluar los pasos y situaciones. Pero en este experimento no se exigió que llegaran a tal nivel de profundidad, ya que sólo se quería evaluar si los participantes eran capaces de crear el "esqueleto" principal de una tarea siguiendo la metodología. En resumen, se quería demostrar que PATH ayuda a materializar los niveles del metamodelo de ULISES y que también ayuda a interiorizar las bases de la metodología.

Las especificaciones de la tarea que modelaron para el experimento eran:

1. Evaluación de colisiones con vehículos u otros objetos
2. El conductor deber parar cuando se encuentra con un semáforo en ámbar o rojo
3. Se deben pasar las señales de ceda el paso a una velocidad máxima de 10 km/h
4. El conductor debe respetar los límites de velocidad de cada vía
5. No se recomienda acelerar si el freno de mano se encuentra accionado
6. Si el vehículo se encuentra a menos de 30 metros de un semáforo, el conductor no debería acelerar
7. Si existe niebla, el conductor debe activar las luces antiniebla
8. El vehículo debe calentar el motor (al menos 60 segundos después de que se haya arrancado el vehículo) antes de iniciar la conducción
9. Un cambio de carril no puede producirse en menos de 3 segundos. De lo contrario se considerará que se ha llevado a cabo un cambio de carril brusco

Estas especificaciones fueron diseñadas especialmente para este experimento. Algunas requieren destrezas de modelado de restricciones básicas y otras tienen un mayor grado de dificultad, y para completar los modelos se requiere una buena comprensión de los conceptos de Paso y Situación. A todos los participantes del experimento se les dio una clase teórica de 45 minutos, donde se les explicaron las bases de ULISES y OLYMPUS, la metodología a seguir y una guía rápida de los tipos de restricciones existentes. Después de la sesión teórica, los sujetos que iban a usar PATH utilizaron la herramienta de autor durante 30 minutos para familiarizarse con la interfaz visual. Además, ambos grupos fueron provistos de un ejemplo de la creación de un paso y una situación completa tanto en el Modelo de Interpretación como en el Modelo de Tareas. No se estableció ningún límite de tiempo, por lo que los sujetos fueron libres de continuar hasta que consideraron que los modelos estaban completos o que no podían seguir completándolos. Es importante subrayar que durante el experimento se permitieron preguntas relacionadas sobre la teoría o sintaxis de las restricciones, ya que el objetivo del experimento no era medir el conocimiento de los sujetos sobre el modelado basado en restricciones.

A continuación se describe el análisis del experimento, el cual se divide en dos partes. Primeramente se comparan los resultados obtenidos por los sujetos que utilizaron PATH con los sujetos que no lo utilizaron. A partir de ahora se referirá al grupo que utilizó PATH como GP y al grupo que no utilizó PATH como NGP.

4.4.2.1 Comparación de los grupos

Se encontraron varias diferencias entre el grupo GP y NGP. La Tabla 8 muestra las diferencias entre el número total de pasos creados y el número total de pasos válidos creados en los modelos de interpretación y tareas. El *número total de pasos* representa a todos los pasos que se crearon durante el proceso, formaran parte de la solución final o no, mientras que los *pasos válidos* representan a los pasos que dejaron finalmente en el modelo. Se efectuó una prueba-T para dos muestras independientes (Two-Sample T-Test) en ambos grupos para determinar si el uso de la herramienta de autor afectaba en la creación de pasos

válidos en los modelos de tareas e interpretación. Para este estudio se adoptó un nivel de significancia del 95%.

Los resultados de la prueba-T en la diferencia de los pasos totales y los pasos válidos en el Modelo de Tareas indican que el factor de grupo (T-Valor = 2.01, p-valor = 0.085, GL = 7) no es lo suficientemente significativo (nivel de confidencia 95%) para concluir que la utilización de PATH optimiza la creación de pasos en el Modelo de Tareas, pero el resultado sugiere que existe una tendencia a cometer menos errores en la creación de pasos del Modelo de Tareas. Más aún, en algunos usuarios de NGP la diferencia entre los pasos creados y los pasos válidos era pequeña, pero el número de especificaciones que consiguieron fue muy bajo (Tabla 9). Por lo tanto, se puede concluir que su rendimiento en el proceso de creación del Modelo de Tareas fue bajo, por lo que una pequeña diferencia entre el número de total de pasos creados y el número de pasos válidos no es significativo en este caso.

En lo referente a los resultados de la prueba-T en el Modelo de Interpretación (T-Valor = 3.46, p-valor = 0.009, GL = 8), muestran que PATH optimiza el proceso de creación de pasos en este modelo. En general, los sujetos de NGP crearon más pasos que los necesarios y a medida que acababan la tarea se iban dando cuenta de que estaban cometiendo errores en la creación de ambos modelos. En otras palabras, adoptaron estrategias inefectivas y tuvieron que reorganizar los modelos más veces que los usuarios de PATH. Además, sus acciones les guiaron a más inconsistencias entre los dos modelos porque tuvieron dificultades a la hora de identificar los pasos que faltaban para completar los modelos. Sin embargo, los usuarios de PATH no tuvieron problemas para identificar los pasos que les faltaban. En lo que concierne a la metodología, algunas personas del grupo NGP confundieron los modelos de interpretación y tareas, y además tuvieron dificultades de seguir completando el experimento una vez que acababan el Modelo de Tareas. Estas diferencias entre los dos grupos sugieren que PATH promueve la planificación del proceso de adquisición de conocimiento.

Tabla 8. Resultados de los sujetos que tomaron parte en el experimento

		Grupo1: Sin PATH									Grupo2: con PATH								
		1	2	3	4	5	6	7	8	Media	1	2	3	4	5	6	7	8	Mean
Modelo de Tareas	Número total de pasos creados	17	12	11	16	15	12	9	12	13	11	9	11	11	10	10	9	11	10.25
	Número de pasos válidos	9	8	9	9	11	9	9	8	9	9	7	8	9	8	9	7	8	8.125
	Diferencia entre pasos totales y pasos válidos (media)	4									2.125								
Modelo de Interpretación	Número total de pasos creados	17	16	15	16	17	11	9	14	14.375	15	13	12	13	12	13	13	12	12.875
	Número de pasos válidos	13	13	11	12	12	10	9	9	11.125	14	12	12	12	11	12	11	12	12
	Diferencia entre pasos totales y pasos válidos (media)	3.25									0.875								

4.4.2.2 Importancia de las herramientas de captura y monitorización

El número de especificaciones obtenidas muestra la importancia de las herramientas de captura y monitorización de la actividad. En este caso, el análisis de la Prueba-T (T-Valor = 12.11, p-valor = 0.000, GL = 11) demuestra que la utilización de PATH ayudó a los sujetos en la cumplimentación de las especificaciones. Los sujetos de NGP no tenían manera de inferir los posibles valores de las observaciones, por lo que se les dieron los valores que necesitaban. Sin embargo, no fueron capaces de establecer las relaciones necesarias entre observaciones.

Por otro lado, los usuarios de PATH hicieron uso de la herramienta de captura para intentar completar todas las restricciones. La Tabla 9 ilustra la diferencia entre los dos grupos de sujetos. Mientras el GP fue capaz de completar la mayoría de las especificaciones, el NGP se quedo muy lejos del objetivo. Los usuarios de PATH completaron todas las restricciones necesarias, aunque algunas restricciones no consiguieron representar su propósito. Al contrario, los sujetos de NGP no pudieron definir algunas restricciones porque no sabían cómo hacerlo, básicamente porque sin capturar la actividad del experto en el simulador no eran capaces ni de establecer una hipótesis válida.

Tabla 9 Número de especificaciones de evaluación completadas en el proceso de creación de la tarea

	Sujetos NGP								Sujetos GP							
Número de sujetos	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
Especificaciones de evaluación conseguidas (sobre 9)	3	2	3	2	2	0	0	1	7	7	9	8	8	7	8	7

4.4.2.3 Resultados y análisis de los usuarios de PATH

Ninguno de los sujetos que utilizaron PATH tuvo problemas con la herramienta o con la metodología que debían aplicar. Como muestra la Tabla 9, los sujetos completaron la mayoría de las especificaciones, en parte por el buen uso que dieron a las herramientas de monitorización y captura. Los sujetos hicieron uso de la herramienta de monitorización

una media de 4.12 veces, y mostraron ser capaces de identificar errores de representación y de redefinir las especificaciones de la tarea por sí mismos. Sin embargo, la mayoría tuvo problemas a la hora de definir restricciones. Todos entendieron como usar y relacionar observaciones, pero la mayoría de ellos olvidó la palabra clave (STEPS::) para instanciar pasos en una restricción. Asimismo, los usuarios de PATH afirmaron que no tuvieron problemas para generar restricciones semántica y sintácticamente correctas, gracias a la asistencia que les ofreció la herramienta de verificación de restricciones de PATH.

Por otro lado, los ocho participantes tuvieron dificultades definiendo las restricciones, especialmente en la definición del paso *Cambio de Carril*. Cabe destacar que este paso era el paso más difícil de modelar. Ninguno de los participantes, excepto el sujeto número 3, fue capaz de abstraer el significado del paso *Cambio de Carril.* Definieron correctamente dicho paso en el Modelo de Tareas, pero no pudieron definir las restricciones necesarias en el Modelo de Interpretación. En lo que se refiere a las especificaciones conseguidas, se considera que si hubieran pasado más tiempo refinando el modelo, todos ellos habrían completado las especificaciones. Como se ha citado anteriormente, la creación de una tarea y su proceso de aprendizaje es cuestión de pocas semanas. El modelado de tareas complejas requiere actividad continua con PATH, y como en el caso del experimento, la experiencia no se puede adquirir en unas pocas horas. De todas formas, a pesar de que los sujetos no tenían conocimientos previos sobre restricciones y la metodología a utilizar, consiguieron crear la tarea propuesta parcialmente (media = 7.62, sobre 9 especificaciones), lo cual da una idea de la sencillez de la utilización de PATH y de la metodología propuesta.

En el Anexo B del documento se muestran los Modelos de Tareas e Interpretación en los que se baso la evaluación de los modelos definidos por los estudiantes.

- **Resultados del cuestionario a los usuarios de PATH**

En esta parte de la evaluación se estudió la opinión de los usuarios de PATH respecto a la usabilidad de la herramienta y su forma de representar el conocimiento. El cuestionario estaba compuesto por 9

preguntas (Tabla 10), las cuales fueron respondidas una vez que los sujetos acabaron de modelar la tarea. Todas las preguntas fueron respondidas en una escala Likert de siete puntos, siendo 1 la puntuación mínima (en total desacuerdo) y 7 la máxima (completamente de acuerdo).

Los resultados del cuestionario muestran que la creación de tareas mediante PATH fue bien aceptada por los participantes del estudio. En general los participantes piensan que PATH fue fácil de utilizar (mediana = 6, media = 6.13, DE = 0.64) y no tuvieron problemas encontrando la información que necesitaban (mediana = 1, media = 1.38, DE = 0,51). Los sujetos creen que si hubieran pasado más tiempo aprendiendo la teoría, la definición de las tareas hubiera sido más sencilla (mediana = 6, media = 5.5, DE = 0.76), pero no piensan que si hubieran pasado más tiempo con la herramienta su rendimiento habría mejorado (mediana = 1, media = 1.63, DE = 0.92). Como se ha citado en las líneas anteriores, la mayoría de las dificultades con las que se encontraron los sujetos estaban relacionadas con las restricciones, lo cual es un problema reconocido en la literatura. Por lo tanto, los problemas que tuvieron fueron causados por falta de conocimiento de las restricciones y en parte también de la metodología de ULISES, pero no por la usabilidad de la herramienta de autor. Es más, los sujetos apreciaron que el proceso de aprendizaje de la herramienta de autor fue relativamente sencillo (mediana = 6, media = 5.62, DE = 0.52). Parte de este éxito radica en la organización visual de la herramienta. El establecimiento de diferentes módulos para cada concepto de aprendizaje (Modelo de Tareas, Modelo de Interpretación y Modelo de Observación) (mediana = 5.5, media = 5.63, DE = 0.74) y la organización de la información en la interfaz (mediana = 5.5, media = 5.25, DE = 0.89) ayudó a los usuarios a interiorizar y entender los conceptos de la metodología de ULISES.

Tabla 10: Resultados de la encuesta. Los resultados están obtenidos en una escala Likert de siete puntos: marca mínima 1 y marca máxima 7

Pregunta	***Mediana***	***Media***	***DE***
1.-¿Cómo de fácil fue aprender a utilizar la herramienta?	6	5.63	0.52
2.-¿ Estaba la información bien organizada?	6	5.75	0.7
3.-¿Tuviste algún problema a la hora de encontrar la información que necesitabas?	1	1.38	0.51
4.-¿Crees que la creación de la tarea sería más rápida si hubieras tenido más tiempo de entrenamiento con la herramienta?	1	1.63	0.92
5.-¿Crees que la creación de la tarea sería más rápida si hubieras tenido más entrenamiento teórico?	6	5.5	0.76
6.-¿Crees que los diferentes módulos de conocimiento para cada concepto del metamodelo de tareas te ha ayudado a entender y utilizar los Modelos de Tareas e Interpretación?	5.5	5.63	0.74
7.-¿Crees que la manera de navegar en la herramienta te ha facilitado entender la arquitectura de ULISES?	5.5	5.25	0.89
8.-¿Crees que la herramienta ha sido fácil de utilizar?	6	6.13	0.64
9.-¿Cómo de satisfecho estás con el sistema en general?	6	6	0.53

4.5 DISCUSIÓN GENERAL Y LÍNEAS FUTURAS

El objetivo de PATH es simplificar y guiar a los autores en el proceso de generación de tareas para un amplio abanico de SIIAA. Asimismo, tiene por objeto ayudar a los diseñadores instruccionales a dar expresión al conocimiento de los expertos del dominio en entornos "learn by doing". PATH tiene numerosas ventajas, pero también existen varias desventajas que se han encontrado a la hora de evaluar si se consiguen los objetivos establecidos.

A diferencia de ASTUS, ASPIRE o SimStudent, PATH no ofrece una generación automática de los modelos, como pueden ser la generación automática de restricciones o generalizaciones a través de las demonstraciones de los expertos. Como ha ocurrido en el caso de la creación de tareas para evaluación de destrezas físicas, la ausencia de

este tipo de sistemas puede ser una desventaja cuando se implementan restricciones complejas con PATH. Se considera que un sistema de este tipo aceleraría notablemente el proceso de creación de tareas y también el proceso de aprendizaje relacionado al modelado de restricciones. En vez de centrarse en automatizar la creación de tareas, PATH ha enfocado sus esfuerzos en acelerar el proceso de creación de tareas mediante el impulso a la reflexión y la planificación. Tanto en los sistemas de Model Tracing como en los sistemas basados en el modelado por restricciones se requiere una alta carga cognitiva. En el primero, se deben definir todas las posibles soluciones a un problema, mientras que en el segundo hay que componer las restricciones que son necesarias. Además de automatizar estos procesos, se cree que los diseñadores instruccionales deben hacer un trabajo de reflexión previo a la tarea, algo en lo que PATH ofrece una ayuda considerable.

En lo referente a la ayuda que se ofrece a los estudiantes, se considera que no existe un enfoque perfecto que sea adecuado para cualquier dominio. Por esa razón, en vez implementar una sola técnica de diagnóstico, ULISES ofrece un subsistema de diagnóstico que es independiente de la técnica de diagnóstico específica que se integre en el SIIAA. Por esta razón, PATH permite la integración sencilla de interfaces diferentes para cada técnica de diagnóstico que se integre con ULISES. Esta es una ventaja importante respecto a sistemas como ASTUS o CTAT. Estos sistemas no son adecuados para dominios pobremente definidos; el modelado basado en restricciones ofrece mejores resultados en estos dominios. En otras situaciones, también puede ser ventajoso utilizar otras técnicas de diagnóstico o integrar técnicas de descubrimiento del conocimiento que generen tareas parciales. Por esta razón, se puede concluir que dependiendo del dominio en el que se quiera crear un SIIAA, algunas técnicas de diagnóstico serán más adecuadas que otras. Además, dependiendo del paradigma que se utilice, la generación de una tarea puede implicar más o menos trabajo (Mitrovic, Koedinger, & Martin, 2003). No obstante, en algunos casos la integración de varios sistemas de diagnóstico puede ser interesante (Fournier-Viger et al., 2010), lo cual permite obtener las ventajas de las cada técnica de diagnóstico. Tanto ULISES como PATH permiten la integración de múltiples técnicas de diagnóstico. Además, se

está trabajando en el desarrollo de un módulo de generación semiautomática del conocimiento para ULISES que sea capaz de generar soluciones parciales basándose en los ejemplos de un experto en el dominio.

Por último, hay que destacar que PATH ha sido empleado satisfactoriamente en la creación de tareas para tres SIIAA diferentes: un SIIAA para entrenar conductores de camión profesionales, otro SIIAA para entrenar personas discapacitadas en tareas de jardinería y un último SIIAA para el entrenamiento de destrezas físicas. En estos tres proyectos los diseñadores instruccionales fueron capaces de crear tareas para evaluar a los estudiantes y los resultados mostraron que las tareas cumplían con los requisitos especificados por los expertos de cada dominio. Es más, los experimentos llevados a cabo con PATH han demostrado que gente sin experiencia previa en la definición de modelos de diagnóstico y programación con simuladores fueron capaces de aprender la metodología propuesta en ULISES. Estos sujetos fueron capaces de crear tareas con un corto periodo de entrenamiento, lo cual se puede considerar como un éxito. Se ha demostrado que la metodología que se utiliza en PATH puede ser rápidamente adquirida, con la ventaja de que los diseñadores instruccionales pueden definir tareas sin tener conocimiento del funcionamiento del sistema interactivo. Sin embargo, es inevitable que estos diseñadores tengan que aprender las bases de cada técnica de diagnóstico que se implemente en PATH. Aún así, con la creación de los tres SIIAA, ha quedado demostrado que PATH acelera el proceso de creación de las tareas y que ayuda a comprender la metodología propuesta en ULISES en un corto periodo de tiempo.

CAPÍTULO 5

CONCLUSIONES Y LÍNEAS FUTURAS DE INVESTIGACIÓN

En este capítulo se resume el trabajo realizado, presentando las principales aportaciones y resultados obtenidos en este trabajo de tesis. Además se exponen las posibles líneas de continuación al trabajo presentado.

5.1 RESUMEN

En este proyecto de tesis se han abordado dos temas principales. Por un lado se ha profundizado en la problemática referente a la evaluación de destrezas motoras en los Sistemas Interactivos Inteligentes de Ayuda al Aprendizaje basados en el framework ULISES. Por otro lado, este proyecto se ha centrado en la investigación y desarrollo de una herramienta de autor que minimice el coste de adquisición de conocimiento de los SIIAA basados en ULISES. La creación de los modelos necesarios para ULISES puede ser un trabajo costoso y complejo, y por esta razón se vio la necesidad de disponer de herramientas que ayuden a generarlo.

Para empezar, se ha realizado un estudio sobre los SIIAA focalizado en las tareas procedimentales y en los métodos de evaluación

de destrezas motoras que utilizan. A partir de este estudio se han analizado en profundidad las ventajas y desventajas de los distintos paradigmas que se utilizan a la hora de evaluar destrezas físicas. Éste análisis ha permitido ver las limitaciones de los métodos existentes y diseñar un sistema que mejore las prestaciones de los mismos. Asimismo, se han estudiado las diferentes herramientas de autor que existen en los SIIAA más destacados.

Una vez concluido el análisis de los antecedentes, se ha descrito la arquitectura de la plataforma OLYMPUS y el framework ULISES, cuyo análisis es indispensable a la hora entender este trabajo de tesis. El framework ULISES determina los nexos entre un Sistema Interactivo y un Sistema de Ayuda al aprendizaje, que se fundamentan en el proceso de *observar, interpretar y diagnosticar*.

A continuación, se ha analizado la problemática clave que surge al evaluar destrezas físicas con el framework ULISES. Dadas las características especiales que surgen a la hora de tratar las destrezas físicas, se han añadido nuevas capacidades al framework ULISES. Esto significa que los distintos subsistemas de ULISES tienen que tener la capacidad de trabajar con incertidumbre temporal para así poder llevar a cabo el diagnóstico de destrezas físicas.

Por último, considerando las conclusiones extraídas de las herramientas de autor para SIIAA existentes y las necesidades específicas del framework ULISES, se ha descrito el diseño y desarrollo de la herramienta de autor PATH, el cual forma parte de la plataforma OLYMPUS. Esta herramienta ha demostrado que acelera el proceso de creación de tareas y ayuda a los instructores a adquirir los principios de la metodología utilizada en ULISES.

5.2 PRINCIPALES RESULTADOS Y APORTACIONES

Los objetivos planteados inicialmente en este trabajo han sido satisfechos, permitiendo extraer las siguientes conclusiones:

- Se ha propuesto un método basado en el framework ULISES para evaluar niveles de destrezas físicas. El método se ha centrado en el diagnóstico de cuatro características fundamentales: coordinación, posturas, trayectorias y procedimiento de la secuencia de movimientos.
- Se ha adaptado el framework ULISES para que sea capaz de generar resultados de diagnóstico sobre destrezas motoras. Para ello, se han extendido los tres niveles de este framework (**Observación, Interpretación y Diagnóstico**) para que sean capaces de gestionar la incertidumbre temporal que se da durante el diagnóstico de movimientos en tiempo real.
- La metodología propuesta se ha enfocado desde un punto de vista cualitativo para que los resultados de diagnóstico permitan generar un feedback que imite el que da un instructor real. Es decir, el sistema diseñado es capaz de transformar las variables cuantitativas de un movimiento a un dominio cualitativo.
- El movimiento se ha representado como una serie de arcos caracterizados con lógica difusa que aproximan las trayectorias originales de un conjunto de puntos característicos. Los arcos se han integrado en el Modelo de Observación de ULISES, permitiendo especificar relaciones temporales entre los mismos en el Modelo de Interpretación para definir movimientos complejos.
- Los arcos han permitido describir la dirección, velocidad, aceleración y la concavidad o convexidad de cada segmento de interés de una trayectoria. Con estas variables, ha sido posible corregir un movimiento mediante el uso de elementos expresables verbalmente. Aunque solo se hayan utilizado estos parámetros para modelar los arcos, el Modelo de Observación permite incorporar nuevos parámetros que permitan el diagnóstico de otros aspectos de un movimiento.
- El modelado del movimiento mediante restricciones, con arcos y lógica difusa ha demostrado ser válido para representar la intencionalidad de los alumnos cuando llevan a cabo un movimiento.

- Esta metodología es independiente de los dispositivos de captura del movimiento utilizados, requiriendo únicamente que el Sistema Interactivo pueda hacer el seguimiento de puntos característicos en tiempo real con la precisión que exija el dominio de aprendizaje.
- La metodología permite representar de forma genérica actividad continua que no se ve afectada por la incertidumbre, actividad continua bajo incertidumbre y actividad instantánea, pudiéndose dar de forma simultánea.
- La metodología diseñada ha sido probada satisfactoriamente en el dominio del tenis. El experimento realizado ha permitido verificar la validez de la propuesta.
- Vista la complejidad y el coste de crear generar modelos de tareas para el framework ULISES, se ha desarrollado la herramienta de autor PATH para adquirir el conocimiento necesario para ULISES.
- El objetivo principal de esta herramienta es dar expresión al conocimiento del experto sin que se requieran conocimientos de IA Para ello, PATH distingue los roles de experto y diseñador de la tarea. El diseñador instruccional se basa en la experiencia del experto para definir las tareas. De esta manera, el experto se abstrae de la creación de tareas y el diseñador instruccional no se tiene que preocupar sobre los aspectos educativos de la tarea.
- PATH promueve una metodología de adquisición de conocimiento que hace reflexionar a los diseñadores instruccionales sobre el modelado formal de las tareas con ULISES. De esta manera, los diseñadores interiorizan la metodología de una manera más fácil y además se acelera el proceso de creación de la tarea.
- PATH ha sido evaluado en tres dominios diferentes, con tareas pobremente definidas y que implican un alto grado de incertidumbre en las acciones de los estudiantes. Además se ha llevado a cabo un experimento para verificar la validez de la herramienta de autor.
- La herramienta de autor elimina la necesidad de programación en los modelos de Interpretación y Tareas, lo que acelera notablemente el proceso de creación de tareas.

- La utilización de PATH en la creación de tareas abstrae a los diseñadores de entender el funcionamiento interno del simulador.
- Gracias al experimento realizado, se ha demostrado que PATH ayuda a adquirir la metodología propuesta en ULISES. Todas las tareas creadas en los experimentos fueron creadas en cuestión de horas y con 45 minutos de entrenamiento. Además, los Modelos de ULISES para los tres SIIAA se han desarrollado en cuestión de semanas. Estos datos demuestran la sencillez de uso de PATH.

5.3 LÍNEAS FUTURAS DE INVESTIGACIÓN

Teniendo en cuenta los resultados obtenidos en este trabajo de investigación, se proponen tres líneas principales de investigación: nuevos Sistemas Interactivos y paralelización en la generación de Observaciones, generación de tareas parciales en base a demostraciones e integración de nuevas técnicas de diagnóstico.

- **Nuevos Sistemas Interactivos y aplicación de la metodología en otros dominios**

Esta línea contempla la creación de nuevos SIIAA en otros dominios donde el entrenamiento de destrezas físicas cobre importancia. Un dominio interesante (y que cuenta con muchas posibilidades de investigación) sería el dominio de entrenamiento de tareas de cirugía. Como se ha citado en el análisis de los antecedentes, generalmente en este dominio la evaluación de las destrezas se hace en base a la distancia estadística al modelo del experto, teniendo en cuenta tiempos de ejecución de las tareas, la frecuencia de las tareas ejecutadas y la medición de fuerzas y torques durante las tareas. Por lo tanto, el trabajo presentado aportaría un nuevo punto de vista en lo que refiere a la evaluación de destrezas motoras en el ámbito de la cirugía. Para ello, se podría crear un SIIAA utilizando interfaces hápticas. Este tipo de tecnología ya ha sido utilizada en nuestro grupo de trabajo con fines educativos (Echegaray-López, 2013).

Además de en el dominio de la cirugía, esta tecnología puede ser utilizada en innumerables campos de alto potencial de investigación con el fin de entrenar destrezas físicas: serious games para la recuperación de pacientes que han sufrido derrames cerebrales, entrenamiento de deportes, corrección de posturas en la cadena de producción etc.

- **Extensión del Nivel de Observación**

Esta línea busca mejorar el rendimiento del Subsistema de Observación. En este trabajo de investigación se ha abordado la captura de movimiento del cuerpo completo, utilizando como sistema de captura el sensor Kinect de Microsoft. Sin embargo, en el SIIAA desarrollado únicamente se tuvieron en cuenta los nodos de la parte superior del cuerpo a la hora de evaluar las destrezas del alumno. Se considera que la gestión de las observaciones de todos los puntos interesantes del cuerpo significaría un aumento considerable en el coste de generación de observaciones. Dado que el Nivel de Observación abastece los niveles superiores, ese cuello de botella podría significar la perdida de generación de resultados de diagnóstico en tiempo interactivo.

Una manera de mejorar la eficacia de este nivel consistiría en la paralelización de la generación de observaciones. El Subsistema de Observación permite un alto grado de paralelización, ya que la gestión de cada observación se hace de manera independiente. Por esta razón, sería interesante investigar la paralelización de este nivel mediante distintas estrategias: paralelización en la GPU, sistema de procesamiento multiprocesador o procesamiento multicomputador.

- **Generación de tareas parciales en base a demostraciones**

Esta línea futura se centra en extender la herramienta de autor PATH para que se puedan generar tareas parciales a través de las demostraciones de un experto en el dominio. Actualmente el sistema permite capturar y monitorizar la actividad del experto, pero los resultados de investigación han mostrado que el modelado de los movimientos puede convertirse en una tarea muy costosa. El objetivo sería conseguir la generación automática de restricciones o generalizaciones a través de las demostraciones de los expertos.

ASTUS, ASPIRE, SimStudent y el trabajo de Fournier-Viger (Fournier-Viger et al., 2013) representan investigaciones similares a las que se quieren llevar a cabo. Actualmente, otra tesis doctoral que se está llevando a cabo coordinado con este trabajo, investiga para conseguir dicha meta. Para ello se está integrando el trabajo presentado en esta disertación con técnicas de minería de datos para que se puedan generar modelos de Interpretación y Tareas parciales. Se considera que un trabajo de dichas características aceleraría notablemente el proceso de creación de tareas. Asimismo, la generación automática de restricciones sería de gran ayuda para los diseñadores sin previa experiencia con el modelado de restricciones: además de acelerar el proceso de creación de modelos, les ayudaría a aprender la técnica con "ejemplos" reales creados a partir de las demostraciones de los expertos.

- **Integración de nuevas técnicas de diagnóstico**

En este trabajo de tesis se ha demostrado que la utilización de la técnica basada en restricciones es adecuada para la evaluación de destrezas físicas. Sin embargo, en algunos dominios de entrenamiento de destrezas físicas un SIIAA podría beneficiarse de combinar varias técnicas de diagnóstico: por ejemplo la combinación entre la técnica basada en restricciones y una técnica basada en el reconocimiento de planes. Esta problemática ha sido abordada en otros trabajos de investigación como en (Fournier-Viger et al., 2010). Lozano-Rodero también trató este tema en su trabajo de investigación (Lozano-Rodero, 2009). Como resultado, optó por incluir en el framework ULISES un módulo de control de estrategia que permite la integración de varias técnicas de diagnóstico.

ANEXO A

REGLAS DE EVALUACIÓN DE RESTRICCIONES

En este anexo se presentan las reglas definidas para evaluar Restricciones. El anexo comienza con la descripción gráfica de las todas las relaciones temporales posibles que se pueden dan entre dos intervalos. Estas posibilidades contemplan que uno de los dos intervalos o los dos aún estén abiertos. Además, en caso de que algún intervalo se encuentre abierto, éste puede encontrarse bajo incertidumbre. De la misma manera, se puede dar la posibilidad de que uno de los dos intervalos aún no se haya observado. Las siguientes secciones describen las Relaciones Temporales Cualitativas de Allen, las Relaciones Temporales Cualitativas Puntuales, las Restricciones Lógicas y por último se presentan las reglas de evaluación de las Restricciones Temporales Cuantitativas junto a su algoritmo de cálculo de duración de los intervalos.

A.1 REPRESENTACIÓN GRÁFICA DE RELACIONES ENTRE INTERVALOS

Las figuras Figura 64, Figura 65, Figura 66 y Figura 67 representan las relaciones temporales cualitativas que se pueden dar entre dos intervalos (verde y azul). Los intervalos se representan de la siguiente manera:

- El intervalo abierto se representa mediante un rectángulo que no tiene cerrado su lado derecho.
- Si un intervalo se encuentra abierto, puede encontrarse bajo incertidumbre. La parte que no se encuentra bajo incertidumbre se muestra de color verde o azul fuerte, mientras que la parte del intervalo que se encuentra bajo incertidumbre se muestra de un color verde o azul claro.
- Los intervalos cerrados se muestran mediante un rectángulo, cuyo lado derecho está cerrado.

En las tablas de las secciones posteriores se utiliza la misma numeración empleada en esta sección para numerar las distintas relaciones entre intervalos.

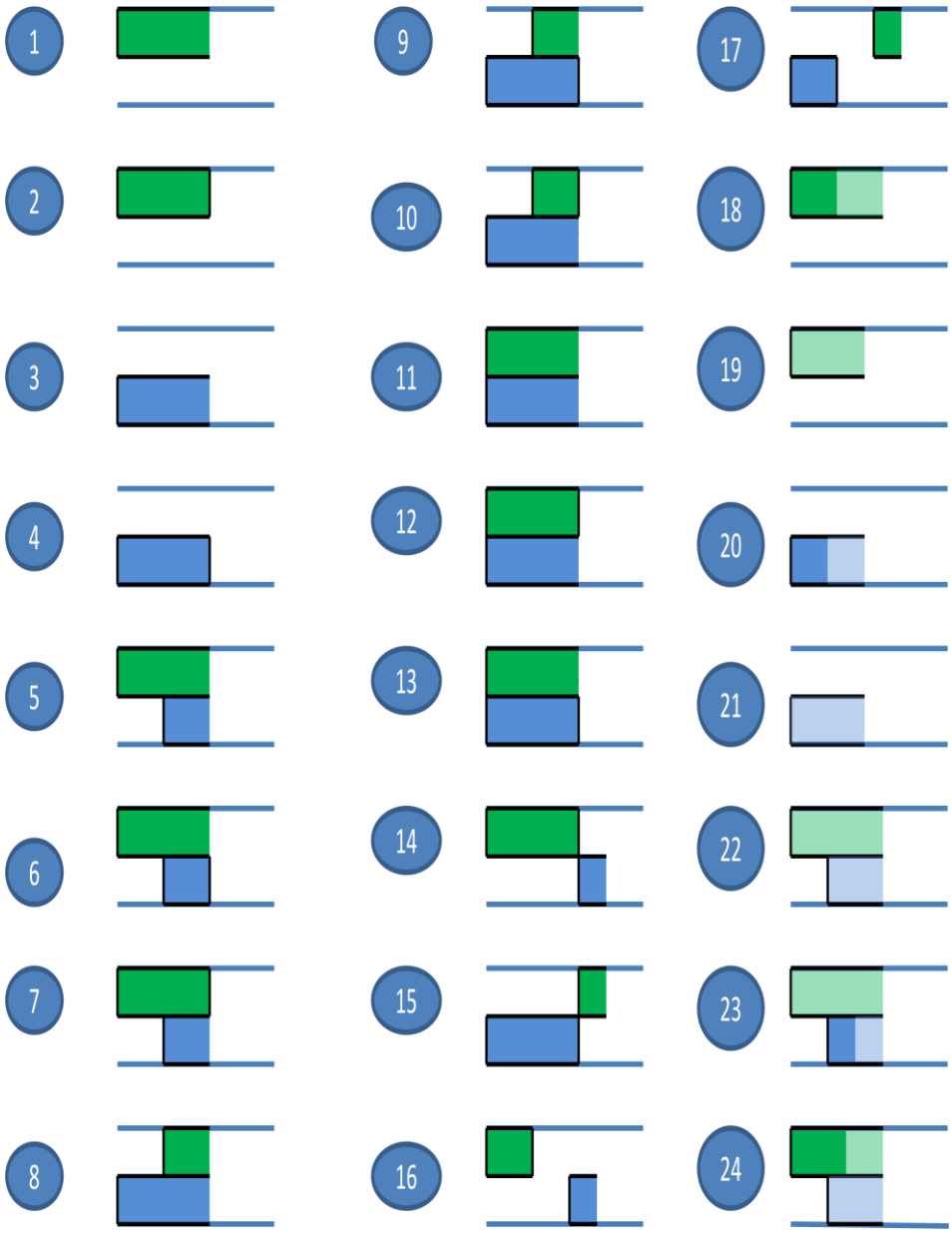


Figura 64: Relaciones temporales entre intervalos (relaciones 1-24)

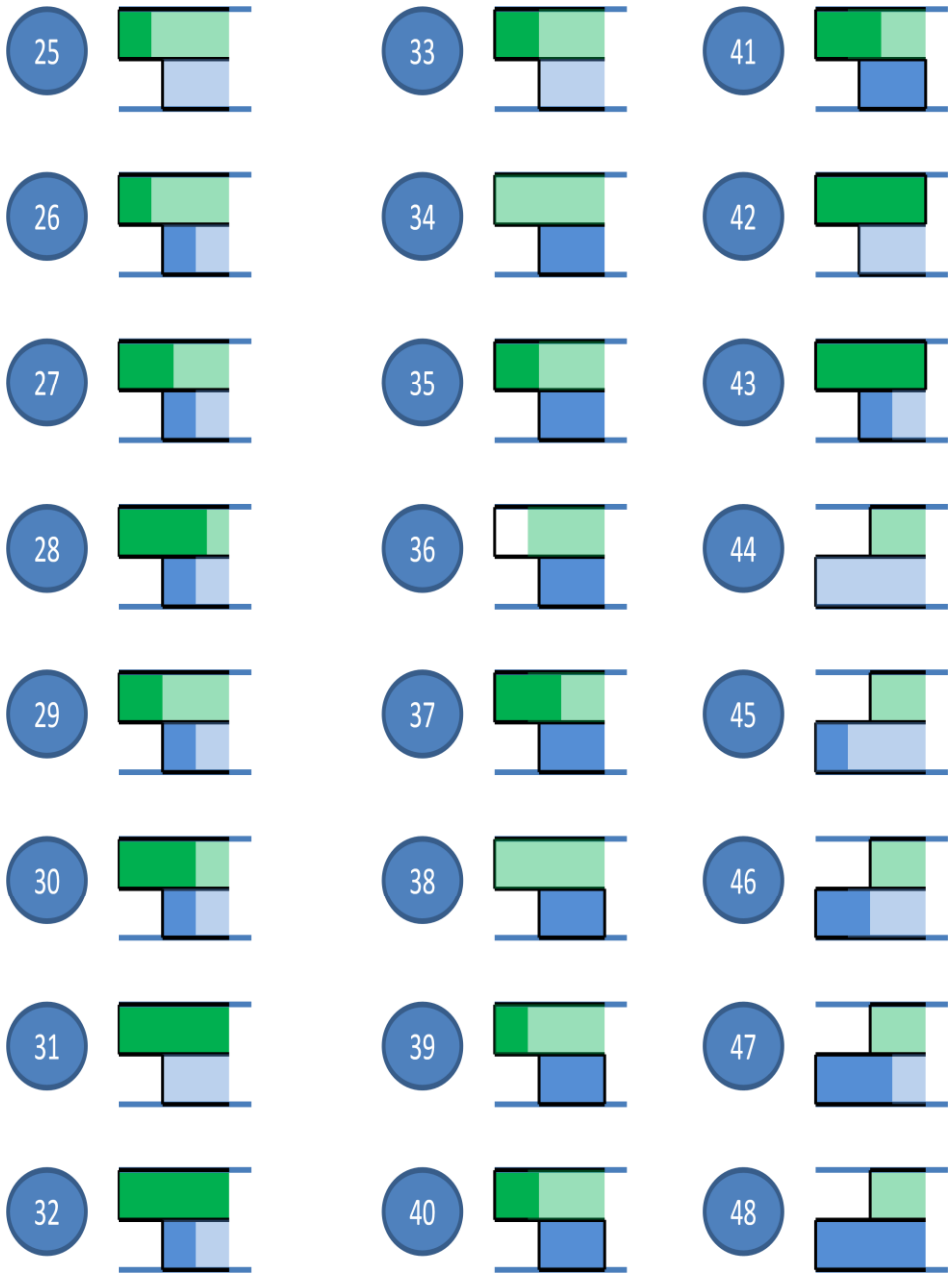


Figura 65: Relaciones temporales entre intervalos (relaciones 25-48)

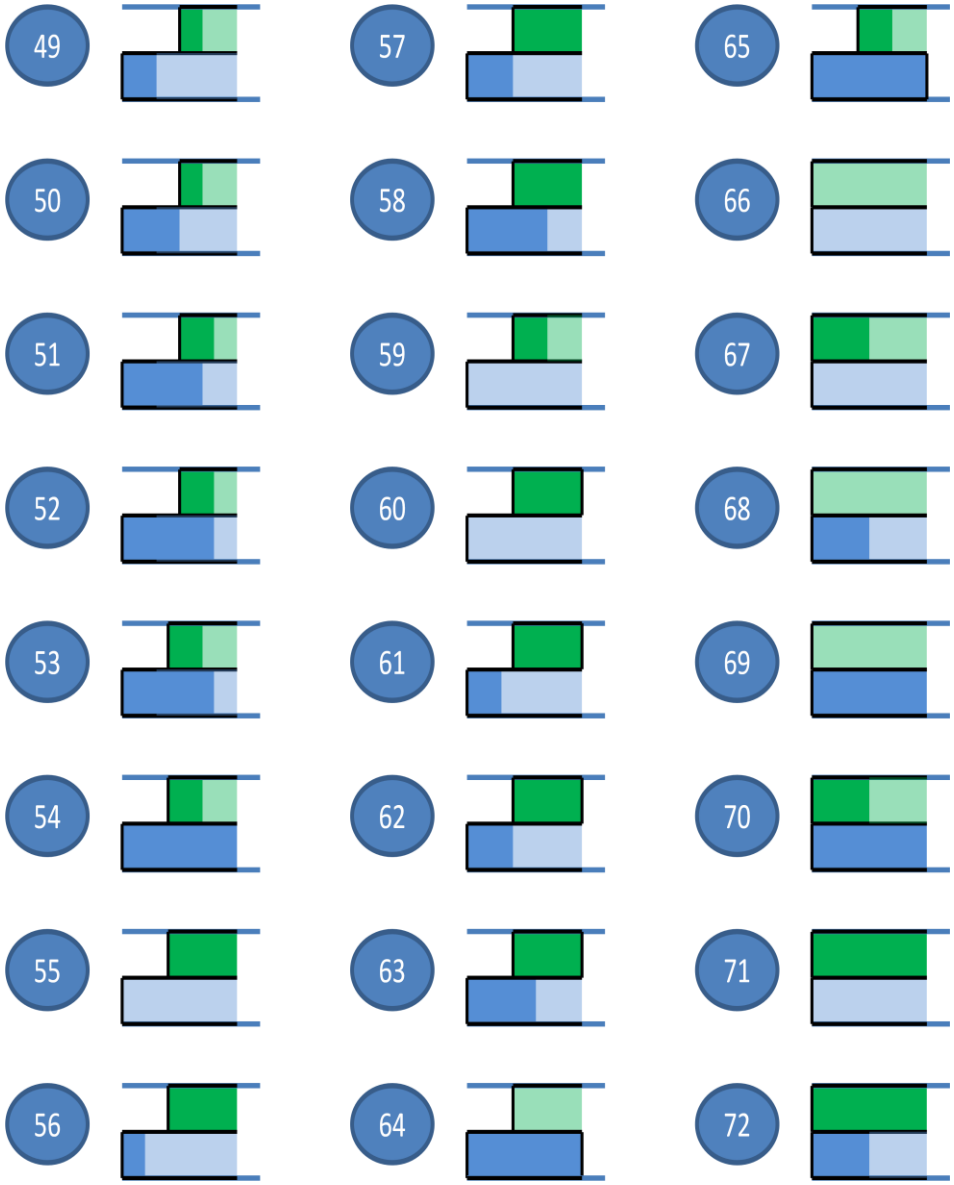


Figura 66: Relaciones temporales entre intervalos (relaciones 49-72)

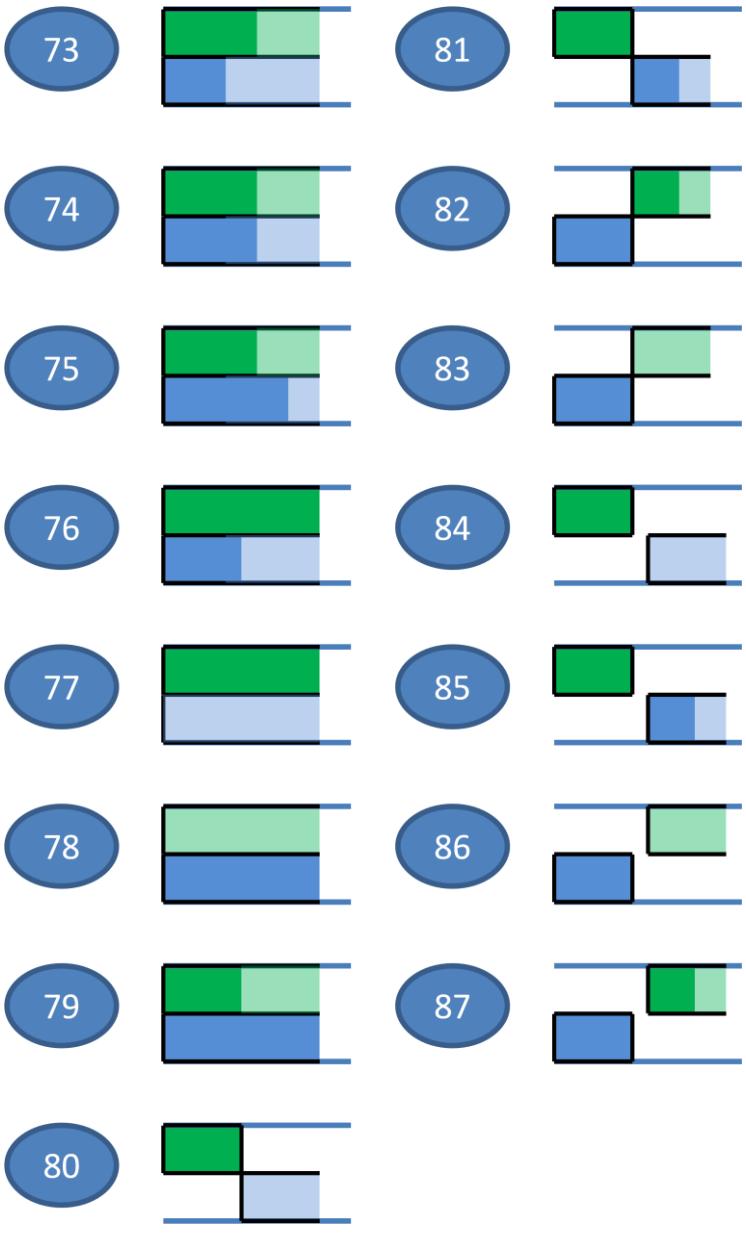


Figura 67: Relaciones temporales entre intervalos (relaciones 73-87)

A.2 EVALUACIÓN DE LAS RESTRICCIONES TEMPORALES CUALITATIVAS DE ALLEN

En las siguientes tablas se muestran los resultados de las reglas de evaluación de las RTCAllen para todas las posibles relaciones temporales. Cada columna muestra una relación temporal, siguiendo la enumeración de la sección anterior. Cada fila muestra un operador básico de Allen.

Tabla 11: Reglas de evaluación de RTCAllen (relaciones 1-17)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
precedes(p)	ED	ED	F	F	F	F	F	F	F	F	F	F	F	F	F	V	F
overlaps(o)	ED	F	F	F	EF	F	V	F	F	F	F	F	F	F	F	F	F
meets(m)	ED	F	F	F	F	F	F	F	F	F	F	F	F	V	F	F	F
starts(s)	F	F	F	F	F	F	F	F	F	F	EF	V	F	F	F	F	F
finishes(f)	F	F	ED	F	F	F	F	EF	F	F	F	F	F	F	F	F	F
equals(e)	F	F	F	F	F	F	F	F	F	F	EF	F	F	F	F	F	F
during(d)	F	F	ED	F	F	F	F	EF	F	V	F	F	F	F	F	F	F
contains (D)	ED	F	F	F	EF	V	F	F	F	F	F	F	F	F	F	F	F
is finished by(F)	ED	F	F	F	EF	F	F	F	F	F	F	F	F	F	F	F	F
is started by(S)	F	F	F	F	F	F	F	F	F	F	EF	F	V	F	F	F	F
is overlapped by(O)	F	F	ED	F	F	F	F	EF	V	F	F	F	F	F	F	F	F
is met by (M)	F	F	ED	F	F	F	F	F	F	F	F	F	F	F	V	F	F
is preceded by(P)	F	F	ED	ED	F	F	F	F	F	F	F	F	F	F	F	F	V

V	Verdadero	ED	Esperar_Datos
F	Falso	EF	Esperar_Final

Tabla 12: Reglas de evaluación de RTCAllen (relaciones 18-34)

	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
precedes(p)	ED	ED	F	F	F	F	EF	EF	EF	F	F	F	F	F	F	EF	F
overlaps(o)	ED	ED	F	F	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF
meets(m)	ED	ED	F	F	F	F	F	F	F	F	F	EF	F	F	F	EF	F
starts(s)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
finishes(f)	F	F	ED	ED	F	F	F	F	F	F	F	F	F	F	F	F	F
equals(e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
during(d)	F	F	ED	ED	F	F	F	F	F	F	F	F	F	F	F	F	F
contains (D)	ED	ED	F	F	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF
is finished by(F)	ED	ED	F	F	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF
is started by(S)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
is over-lapped by(O)	F	F	ED	ED	F	F	F	F	F	F	F	F	F	F	F	F	F
is met by (M)	F	F	ED	ED	F	F	F	F	F	F	F	F	F	F	F	F	F
is preceded by(P)	F	F	ED	ED	F	F	F	F	F	F	F	F	F	F	F	F	F

V	Verdadero	ED	Esperar_Datos
F	Falso	EF	Esperar_Final

Tabla 13: Reglas de evaluación de RTCAllen (relaciones 35-51)

	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
precedes(p)	F	EF	F	F	EF	F	F	F	F	F	F	F	F	F	F	F	F
overlaps(o)	EF	EF	EF	F	F	F	EF	EF	EF	F	F	F	F	F	F	F	F
meets(m)	EF	F	F	F	F	EF	F	F	F	F	F	F	F	F	F	F	F
starts(s)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
finishes(f)	F	F	F	F	F	F	F	F	F	EF	EF	EF	EF	EF	EF	EF	EF
equals(e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
during(d)	F	F	F	F	F	F	F	F	F	EF	EF	EF	EF	EF	EF	EF	EF
contains (D)	EF	EF	EF	EF	EF	EF	EF	F	EF	F	F	F	F	F	F	F	F
is finished by(F)	EF	EF	EF	F	F	F	F	F	F	F	F	F	F	F	F	F	F
is started by(S)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
is over-lapped by(O)	F	F	F	F	F	F	F	F	F	EF	EF	EF	EF	EF	EF	EF	EF
is met by (M)	F	F	F	F	F	F	F	F	F	F	F	EF	F	F	F	EF	F
is preceded by(P)	F	F	F	F	F	F	F	F	F	F	EF	F	F	F	EF	F	F

V	Verdadero	ED	Esperar_Datos
F	Falso	EF	Esperar_Final

Tabla 14: Reglas de evaluación de RTCAllen (relaciones 52-68)

	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68
precedes(p)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
overlaps(o)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
meets(m)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
starts(s)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	EF	EF	EF
finishes(f)	EF	EF	EF	EF	EF	EF	EF	EF	F	F	F	F	F	F	F	F	F
equals(e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	EF	EF	EF
during(d)	EF	EF	EF	EF	EF	EF	EF	EF	EF	V	V	V	F	EF	F	F	F
contains (D)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
is finished by(F)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
is started by(S)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	EF	EF	EF
is over-lapped by(O)	EF	EF	EF	EF	EF	EF	EF	EF	F	F	F	EF	EF	EF	F	F	F
is met by (M)	F	F	F	F	F	EF	F	F	F	F	EF	F	F	F	F	F	F
is preceded by(P)	F	F	F	F	EF	F	F	F	F	EF	F	F	F	F	F	F	F

V	Verdadero	ED	Esperar_Datos
F	Falso	EF	Esperar_Final

Tabla 15: Reglas de evaluación de RTCAllen (relaciones 69-84)

	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84
precedes(p)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	EF
overlaps(o)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
meets(m)	F	F	F	F	F	F	F	F	F	F	F	EF	V	F	F	F
starts(s)	EF	EF	EF	EF	EF	EF	EF	EF	EF	F	EF	F	F	F	F	F
finishes(f)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
equals(e)	EF	EF	EF	EF	EF	EF	EF	F	F	F	F	F	F	F	F	F
during(d)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
contains (D)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
is finished by(F)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
is started by(S)	EF	EF	EF	EF	EF	EF	EF	EF	F	EF	EF	F	F	F	F	F
is over-lapped by(O)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
is met by (M)	F	F	F	F	F	F	F	F	F	F	F	F	F	V	EF	F
is preceded by(P)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F

Verdadero		ED	Esperar_Datos
Falso		EF	Esperar_Final

Tabla 16: Reglas de evaluación de RTCAllen (relaciones 85-87)

	85	86	87
precedes(p)	V	F	F
overlaps(o)	F	F	F
meets(m)	F	F	F
starts(s)	F	F	F
finishes(f)	F	F	F
equals(e)	F	F	F
during(d)	F	F	F
contains (D)	F	F	F
is finished by(F)	F	F	F
is started by(S)	F	F	F
is overlapped by(O)	F	F	F
is met by (M)	F	F	F
is preceded by(P)	F	EF	V

V	Verdadero	ED	Esperar_Datos
F	Falso	EF	Esperar_Final

A.3 EVALUACIÓN DE RESTRICCIONES TEMPORALES CUALITATIVAS PUNTUALES

Las tablas que se exponen en esta sección muestran las reglas de evaluación de RTCPuntual para todas las relaciones enumeradas en la sección A.1. Las filas muestran el resultado de evaluar todas las relaciones temporales posibles con los instantes de inicio y fin de los dos intervalos. Los instantes temporales de los intervalos se representan de la siguiente manera:

- I1 = Inicio del intervalo verde (*ver sección A.1 para los colores de los intervalos)
- I2 = Inicio del intervalo azul
- F1 = Final de intervalo verde
- F2 = Final de intervalo azul

Tabla 17: Reglas de evaluación de RTCPuntual (relaciones 1-17)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
I1 <= I2	ED	ED	F	F	V	V	V	F	F	F	V	V	V	V	F	V	F
I1 < I2	ED	ED	F	F	V	V	V	F	F	F	F	F	F	V	F	V	F
I1 = I2	F	F	F	F	F	F	F	F	F	F	V	V	V	F	F	F	F
I1 >= I2	F	F	ED	ED	F	F	F	V	V	V	V	V	V	F	V	F	V
I1 > I2	F	F	ED	ED	F	F	F	V	V	V	F	F	F	F	V	F	V
I1 <= F2	ED	ED	ED	ED	V	V	V	V	V	V	V	V	V	V	V	V	F
I1 < F2	ED	ED	ED	F	V	V	V	V	V	V	V	V	V	V	F	V	F
I1 = F2	F	F	ED	ED	F	F	F	F	F	F	F	F	F	F	V	F	F
I1 >= F2	F	F	ED	ED	F	F	F	F	F	F	F	F	F	F	V	F	V
I1 > F2	F	F	ED	ED	F	F	F	F	F	F	F	F	F	F	F	F	V
F1 <= I2	ED	ED	F	F	F	F	F	F	F	F	F	F	F	V	F	V	F
F1 < I2	ED	ED	F	F	F	F	F	F	F	F	F	F	F	F	F	V	F
F1 = I2	ED	ED	F	F	F	F	F	F	F	F	F	F	F	V	F	F	F
F1 >= I2	ED	F	ED	ED	V	V	V	V	V	V	V	V	V	V	V	F	V
F1 > I2	ED	F	ED	ED	V	V	V	V	V	V	V	V	V	F	V	F	V
F1 <= F2	ED	ED	ED	F	EF	F	V	EF	F	V	EF	V	F	V	F	V	F
F1 < F2	ED	ED	ED	F	EF	F	V	EF	F	V	EF	V	F	V	F	V	F
F1 = F2	ED	F	ED	F	EF	F	F	EF	F	F	EF	F	F	F	F	F	F
F1 >= F2	ED	F	ED	ED	EF	V	F	EF	V	F	EF	F	V	F	V	F	V
F1 > F2	ED	F	ED	ED	EF	V	F	EF	V	F	EF	F	V	F	V	F	V

V	Verdadero	**ED**	Esperar_Datos
F	Falso	**EF**	Esperar_Final

Tabla 18: Reglas de evaluación de RTCPuntual (relaciones 18-34)

	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
I1 <= I2	ED	ED	F	F	EF	EF	EF	EF	EF	V	V	V	V	EF	V	EF	EF
I1 < I2	ED	ED	F	F	EF	EF	EF	EF	EF	V	V	V	V	EF	V	EF	EF
I1 = I2	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
I1 >= I2	F	F	ED	ED	F	F	F	F	F	F	F	F	F	F	F	F	F
I1 > I2	F	F	ED	ED	F	F	F	F	F	F	F	F	F	F	F	F	F
I1 <= F2	ED	ED	ED	ED	EF	EF	EF	EF	EF	V	V	V	V	EF	V	EF	EF
I1 < F2	ED	ED	ED	ED	EF	EF	EF	EF	EF	V	V	V	V	EF	V	EF	EF
I1 = F2	F	F	ED	ED	F	F	F	F	F	F	F	F	F	F	F	F	F
I1 >= F2	F	F	ED	ED	F	F	F	F	F	F	F	F	F	F	F	F	F
I1 > F2	F	F	ED	ED	F	F	F	F	F	F	F	F	F	F	F	F	F
F1 <= I2	ED	ED	F	F	F	F	F	EF	EF	F	F	EF	F	EF	F	EF	F
F1 < I2	ED	ED	F	F	F	F	F	EF	EF	F	F	F	F	F	F	F	F
F1 = I2	ED	ED	F	F	F	F	F	F	F	F	F	EF	F	F	F	EF	F
F1 >= I2	ED	ED	ED	ED	EF	EF	EF	EF	EF	V	V	EF	V	EF	V	EF	EF
F1 > I2	ED	ED	ED	ED	EF	EF	EF	EF	EF	V	V	EF	V	EF	V	EF	EF
F1 <= F2	ED	ED	ED	ED	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF
F1 < F2	ED	ED	ED	ED	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF
F1 = F2	ED	ED	ED	ED	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF
F1 >= F2	ED	ED	ED	ED	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF
F1 > F2	ED	ED	ED	ED	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF

V	Verdadero	**ED**	Esperar_Datos
F	Falso	**EF**	Esperar_Final

Tabla 19: Reglas de evaluación de RTCPuntual (relaciones 35-52)

	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52
I1 <= I2	V	V	V	EF	V	V	V	EF	V	F	F	F	F	F	F	F	F	F
I1 < I2	V	V	V	EF	V	V	V	EF	V	F	F	F	F	F	F	F	F	F
I1 = I2	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
I1 >= I2	F	F	F	F	F	F	F	F	F	EF	EF	EF	EF	EF	V	V	V	V
I1 > I2	F	F	F	F	F	F	F	F	F	EF	EF	EF	EF	EF	V	V	V	V
I1 <= F2	V	V	V	EF	V	V	V	EF	V	EF	EF	EF	EF	EF	EF	EF	V	EF
I1 < F2	V	V	V	EF	V	V	V	EF	V	EF	EF	EF	EF	EF	EF	EF	V	EF
I1 = F2	F	F	F	F	F	F	F	F	F	F	F	EF	F	F	F	EF	F	F
I1 >= F2	F	F	F	F	F	F	F	F	F	F	EF	EF	F	F	F	F	F	F
I1 > F2	F	F	F	F	F	F	F	F	F	F	EF	EF	F	F	F	F	F	F
F1 <= I2	EF	EF	F	F	EF	EF	F	F	F	F	F	F	F	F	F	F	F	F
F1 < I2	F	EF	F	F	EF	F	F	F	F	F	F	F	F	F	F	F	F	F
F1 = I2	EF	F	F	F	F	EF	F	F	F	F	F	F	F	F	F	F	F	F
F1 >= I2	EF	EF	EF	EF	EF	EF	V	EF	V	EF	EF	EF	EF	EF	V	V	V	V
F1 > I2	EF	EF	EF	EF	EF	EF	V	EF	V	EF	EF	EF	EF	EF	V	V	V	V
F1 <= F2	EF	EF	EF	F	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF
F1 < F2	EF	EF	EF	F	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF
F1 = F2	EF	EF	EF	F	F	F	F	F	F	EF	EF	EF	EF	EF	EF	EF	EF	EF
F1 >= F2	EF	EF	EF	EF	EF	EF	EF	F	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF
F1 > F2	EF	EF	EF	EF	EF	EF	EF	F	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF

V	Verdadero	**ED**	Esperar_Datos
F	Falso	**EF**	Esperar_Final

Tabla 20: Reglas de evaluación de RTCPuntual (relaciones 53-70)

	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70
I1 <= I2	F	F	F	F	F	F	F	F	F	F	F	F	F	EF	EF	EF	EF	V
I1 < I2	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
I1 = I2	F	F	F	F	F	F	F	F	F	F	F	F	F	EF	EF	EF	EF	V
I1 >= I2	V	V	EF	V	V	V	EF	EF	V	V	V	EF	V	EF	EF	EF	EF	V
I1 > I2	V	V	EF	V	V	V	EF	EF	V	V	V	EF	V	F	F	F	F	F
I1 <= F2	V	V	EF	EF	EF	V	EF	EF	EF	EF	V	EF	V	EF	EF	EF	EF	V
I1 < F2	V	V	EF	EF	EF	V	EF	EF	EF	EF	V	EF	V	EF	EF	EF	EF	V
I1 = F2	F	F	F	F	EF	F	F	F	F	EF	F	F	F	F	F	F	F	F
I1 >= F2	F	F	F	EF	EF	F	F	F	EF	EF	F	F	F	F	F	F	F	F
I1 > F2	F	F	F	EF	F	F	F	F	EF	F	F	F	F	F	F	F	F	F
F1 <= I2	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
F1 < I2	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
F1 = I2	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
F1 >= I2	V	V	EF	V	V	V	EF	EF	V	V	V	EF	V	EF	EF	EF	EF	V
F1 > I2	V	V	EF	V	V	V	EF	EF	V	V	V	EF	V	EF	EF	EF	EF	V
F1 <= F2	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF
F1 < F2	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF
F1 = F2	EF	EF	EF	EF	EF	EF	EF	F	F	F	F	F	F	EF	EF	EF	EF	EF
F1 >= F2	EF	EF	EF	EF	EF	EF	EF	F	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF
F1 > F2	EF	EF	EF	EF	EF	EF	EF	F	EF	EF	EF	EF	EF	EF	EF	EF	EF	EF

V	Verdadero	**ED**	Esperar_Datos
F	Falso	**EF**	Esperar_Final

Tabla 21: Reglas de evaluación de RTCPuntual (relaciones 71- 87)

	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87
I1 <= I2	EF	V	V	V	V	V	EF	EF	V	EF	V	F	F	EF	V	F	F
I1 < I2	F	F	F	F	F	V	EF	EF	V	EF	V	F	F	EF	V	F	F
I1 = I2	EF	V	V	V	V	V	EF	EF	V	F	F	F	F	F	F	F	F
I1 >= I2	EF	V	V	V	V	V	EF	EF	V	F	F	F	EF	F	F	EF	V
I1 > I2	F	F	F	F	F	F	F	F	F	F	F	V	EF	F	F	EF	V
I1 <= F2	EF	V	V	V	V	V	EF	EF	V	EF	V	V	EF	EF	V	F	F
I1 < F2	EF	V	V	V	V	V	EF	EF	V	EF	V	F	F	EF	V	F	F
I1 = F2	F	F	F	F	F	F	F	F	F	F	F	V	EF	F	F	F	F
I1 >= F2	F	F	F	F	F	F	F	F	F	F	F	V	EF	F	F	EF	V
I1 > F2	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	EF	V
F1 <= I2	F	F	F	F	F	F	F	F	F	EF	EF	F	F	EF	V	F	F
F1 < I2	F	F	F	F	F	F	F	F	F	F	F	F	F	EF	V	F	F
F1 = I2	F	F	F	F	F	F	F	F	F	EF	V	F	F	F	F	F	F
F1 >= I2	EF	V	V	V	V	V	EF	EF	V	EF	EF	V	EF	F	F	EF	V
F1 > I2	EF	V	V	V	V	V	EF	EF	EF	F	F	V	EF	F	F	EF	V
F1 <= F2	EF	EF	EF	EF	EF	EF	EF	F	EF	EF	V	F	F	EF	V	F	F
F1 < F2	EF	EF	EF	EF	EF	EF	EF	F	EF	EF	V	F	F	EF	V	F	F
F1 = F2	EF	EF	EF	EF	EF	F	F	F	F	F	F	F	F	F	F	F	F
F1 >= F2	EF	EF	EF	EF	EF	EF	F	EF	EF	F	F	V	EF	F	F	EF	V
F1 > F2	EF	EF	EF	EF	EF	EF	F	EF	EF	F	F	V	EF	F	F	EF	V

V	Verdadero	**ED**	Esperar_Datos
F	Falso	**EF**	Esperar_Final

A.4 EVALUACIÓN DE RESTRICCIONES LÓGICAS

A continuación se presentan las reglas para evaluar las restricciones lógicas. Se presentan tres tablas, una tabla para cada operador lógico (AND, OR, NOT). Cada una muestra el resultado de evaluar la restricción lógica por cada posible resultado de evaluación.

Además de los acrónimos ED, EF, ND, V y F, en esta sección se presentan los acrónimos EDC (Esperar desde resultado confirmado) y EDNC (esperar desde resultado no confirmado). El primero indica que al evaluar la restricción el intervalo evaluado se encontraba confirmado en algún punto, mientras que el segundo indica que todavía no se ha confirmado que el intervalo esté ocurriendo.

Tabla 22: Reglas de evaluación de Restricción Lógica (AND)

AND	ED	EF	ND	V	F	EDC	EDNC
ED	ED	ED	F	ED	F	ED	ED
EF	ED	EF	ED	EF	F	EF	EF
ND	ED	ED	ND	ED	F	ED	ED
V	ED	EF	ED	V	F	EF	EF
F	F	F	F	F	F	F	F
EDC	ED	EF	ED	EF	F	EF	EF
EDNC	ED	EF	ED	EF	F	EF	EF

V	Verdadero	**ED**	Esperar_Datos	ND	NoDatos
F	Falso	**EF**	Esperar_Final		

Tabla 23: Reglas de evaluación de Restricción Lógica (OR)

OR	ED	EF	ND	V	F	EDC	EDNC
ED	ED	EF	ED	V	ED	EF	EF
EF	EF	EF	EF	V	EF	EF	EF
ND	ED	EF	ND	V	ND	EF	EF
V	V	V	V	V	V	V	V
F	ED	EF	ND	V	F	EF	EF
EDC	ED	EF	ED	V	F	EF	EF
EDNC	ED	EF	ED	V	F	EF	EF

V	Verdadero	**ED**	Esperar_Datos	ND	NoDatos
F	Falso	**EF**	Esperar_Final		

Tabla 24: Reglas de evaluación para Restricción Lógica (NOT)

NOT	ED	EF	ND	V	F	EDC	EDF
	ED	EF	V	F	V	EF	EF

V	Verdadero	ED	Esperar_Datos
F	Falso	EF	Esperar_Final

A.5 EVALUACIÓN DE RESTRICCIONES TEMPORALES CUANTITATIVAS

El proceso de evaluación de las RTCuantitativas se lleva a cabo en dos fases. En la primera fase se intentan calcular las duraciones entre los distintos instantes por separado. En este cálculo, los resultados posibles son Verdadero, Falso, No_Datos, Esperar_Datos, Esperar_Final, EDC (esperar desde resultado confirmado) y EDNC (esperar desde resultado no confirmado). En los casos en los que el resultado del cálculo de la duración sea Verdadero, Esperar_Final, EDC y EDNC, el resultado vendrá acompañado de la duración calculada.

En la segunda fase se evalúa la RTCuantitativa en base a los resultados de las duraciones obtenidas. En esa evaluación se utilizan las reglas de evaluación que se muestran en las siguientes tablas. Se muestra una tabla por cada operador (<, >, <=, >=, =). Cada tabla muestra el resultado de evaluar la RTCuantitativa *Duracion1[operador]Duracion2* según el resultado de cálculo de las duraciones de cada intervalo.

Tabla 25: Reglas de evaluación de RTCuantitativa (operador <)

<	V	F	ND	ED	EF	EDC	EDNC
V	d1<d2	F	ED	ED	d2==null->EF (d1<d2)->V !(d1<d2)->EF	d2==null->EF (d1<d2)->V !(d1<d2)->EF	EF
F	F	F	F	F	F	F	F
ND	ED	F	ND	ED	ED	ED	ED
ED	ED	F	ED	ED	ED	ED	ED

EF	d1==null->EF (d1<d2)->EF !(d1<d2)->F	F	ED	ED	EF	EF	EF
EDC	d1==null->EF (d1<d2)->EF !(d1<d2)->F	F	ED	ED	EF	EF	EF
EDNC	EF	F	ED	ED	EF	EF	EF

ND	No_Datos	**ED**	Esperar_Datos
F	Falso	**EF**	Esperar_Final

Tabla 26: Reglas de evaluación de RTCuantitativa (operador >)

>	**V**	**F**	**ND**	**ED**	**EF**	**EDC**	**EDNC**
V	d1>d2	F	ED	ED	d2==null->EF (d1>d2)->EF !(d1>d2)->F	d2==null->EF (d1>d2)->EF !(d1>d2)->F	EF
F	F	F	F	F	F	F	F
ND	ED	F	ND	ED	ED	ED	ED
ED	ED	F	ED	ED	ED	ED	ED
EF	d1==null->EF (d1>d2)->V !(d1>d2)->EF	F	ED	ED	EF	EF	EF
EDC	d1==null->EF (d1>d2)->V !(d1>d2)->EF	F	ED	ED	EF	EF	EF
EDNC	EF	F	ED	ED	EF	EF	EF

ND	No_Datos	**ED**	Esperar_Datos
F	Falso	**EF**	Esperar_Final

Tabla 27: Reglas de evaluación de RTCuantitativa (operador <=)

<=	**V**	**F**	**ND**	**ED**	**EF**	**EDC**	**EDNC**
V	d1<=d2	F	ED	ED	d2==null->EF (d1<=d2)->V !(d1<=d2)->EF	d2==null->EF (d1<=d2)->V !(d1<=d2)->EF	EF
F	F	F	F	F	F	F	F
ND	ED	F	ND	ED	ED	ED	ED
ED	ED	F	ED	ED	ED	ED	ED
EF	d1==null->EF (d1<=d2)->EF !(d1<=d2)->F	F	ED	ED	EF	EF	EF
EDC	d1==null->EF (d1<=d2)->EF	F	ED	ED	EF	EF	EF

	!(d1<=d2)->F						
EDNC	EF	F	ED	ED	EF	EF	EF

ND	No_Datos	**ED**	Esperar_Datos
F	Falso	**EF**	Esperar_Final

Tabla 28: Reglas de evaluación de RTCuantitativa (operador >=)

>=	V	F	ND	ED	EF	EDC	EDNC
V	d1>=d2	F	ED	ED	d2==null->EF (d1>=d2)->EF !(d1>=d2)->F	d2==null->EF (d1>=d2)->EF !(d1>=d2)->F	EF
F	F	F	F	F	F	F	F
ND	ED	F	ND	ED	ED	ED	ED
ED	ED	F	ED	ED	ED	ED	ED
EF	d1==null->EF (d1>=d2)->V !(d1>=d2)->EF	F	ED	ED	EF	EF	EF
EDC	d1==null->EF (d1>=d2)->V !(d1>=d2)->EF	F	ED	ED	EF	EF	EF
EDNC	EF	F	ED	ED	EF	EF	EF

ND	No_Datos	**ED**	Esperar_Datos
F	Falso	**EF**	Esperar_Final

Tabla 29: Reglas de evaluación de RTCuantitativa (operador =)

=	V	F	ND	ED	EF	EDC	EDNC
V	d1=d2	F	ED	ED	EF	EF	EF
F	F	F	F	F	F	F	F
ND	ED	F	ND	ED	ED	ED	ED
ED	ED	F	ED	ED	ED	ED	ED
EF	EF	F	ED	ED	EF	EF	EF
EDC	EF	F	ED	ED	EF	EF	EF
EDNC	EF	F	ED	ED	EF	EF	EF

ND	No_Datos	**ED**	Esperar_Datos
F	Falso	**EF**	Esperar_Final

ANEXO B

MODELOS PATH

En este anexo se muestran el Modelo de Tareas y el Modelo de Interpretación que se definieron para el experimento realizado con los usuarios de PATH. Estos modelos se utilizaron para compararlos con los modelos creados por los estudiantes y poder realizar así realizar una evaluación de dichos modelos. Cabe destacar que la información referente a los modelos se encuentra almacenada en un sistema de gestión de BBDD relacional SQL Server y en ficheros XML. El anexo muestra la definición de los modelos en XML.

B.1 MODELO DE TAREAS

A continuación se muestran las soluciones a las 3 situaciones que había que definir en el experimento: situación de ceda del paso, situación general y la situación semáforo:

```
<Solution SituationName="SitCeda">
        <Steps>
          <Step Name="RebasarCeda">
            <Weight>0</Weight>
            <FeedbackPos>
            </FeedbackPos>
            <Conditions>
              <Condition Name="Condicion">
                <Declarations>
                  <Declaration Name="Vehiculo">
                    <IntervalInstances>
                      <IntervalInstance>V</IntervalInstance>
                    </IntervalInstances>
                  </Declaration>
                </Declarations>
```

```
                    <Constraint><![CDATA[V.VelocidadVehiculo <10.0]]></Constraint>
                    <Penalize>0</Penalize>
                    <MarkWeight>0</MarkWeight>
                    <FeedbackNeg>
                    </FeedbackNeg>
                    <LearningObjective>Frenada segura en semafo-
          ros</LearningObjective>
                  </Condition>
                </Conditions>
              </Step>
            </Steps>
</Solution>

<Solution SituationName="SitSemaforo">
            <Steps>
              <Step Name="Rebasar">
                <Weight>0</Weight>
                <FeedbackPos>
                </FeedbackPos>
                <Conditions>
                  <Condition Name="Condicion">
                    <Declarations>
                      <Declaration Name="Semaforo">
                        <IntervalInstances>
                          <IntervalInstance>S</IntervalInstance>
                        </IntervalInstances>
                      </Declaration>
                    </Declarations>
                    <Constraint><![CDATA[S.EstadoSemaforo == 2.0]]></Constraint>
                    <Penalize>0</Penalize>
                    <MarkWeight>0</MarkWeight>
                    <FeedbackNeg>
                    </FeedbackNeg>
                    <LearningObjective>Frenada segura en semafo-
          ros</LearningObjective>
                  </Condition>
                </Conditions>
              </Step>
              <Step Name="Acelerar">
                <Weight>0</Weight>
                <FeedbackPos>
                </FeedbackPos>
                <Conditions>
                  <Condition Name="Condicion">
                    <Declarations>
                      <Declaration Name="Semaforo">
                        <IntervalInstances>
                          <IntervalInstance>S</IntervalInstance>
                        </IntervalInstances>
                      </Declaration>
                    </Declarations>
                    <Constraint><![CDATA[(S.EstadoSemaforo == 2.0) OR
          (S.DistanciaSemaforo>30.0)]]></Constraint>
                    <Penalize>0</Penalize>
                    <MarkWeight>0</MarkWeight>
                    <FeedbackNeg>
                    </FeedbackNeg>
                    <LearningObjective>Frenada segura en semafo-
          ros</LearningObjective>
                  </Condition>
                </Conditions>
              </Step>
            </Steps>
  </Solution>

  <Solution SituationName="General">
            <Steps>
```

```
    <Step Name="Choque">
      <Weight>0</Weight>
      <FeedbackPos>
      </FeedbackPos>
      <Conditions>
        <Condition Name="Condicion">
          <Declarations>
            <Declaration Name="ColisionVehiculo">
              <IntervalInstances>
                <IntervalInstance>CV</IntervalInstance>
              </IntervalInstances>
            </Declaration>
          </Declarations>
          <Constraint><![CDATA[CV.VehiculoColisiona < 1.0]]></Constraint>
          <Penalize>0</Penalize>
          <MarkWeight>0</MarkWeight>
          <FeedbackNeg>
          </FeedbackNeg>
          <LearningObjective>Frenada segura en semafo-
ros</LearningObjective>
        </Condition>
      </Conditions>
    </Step>
    <Step Name="Circular">
      <Weight>0</Weight>
      <FeedbackPos>
      </FeedbackPos>
      <Conditions>
        <Condition Name="Condicion">
          <Declarations>
            <Declaration Name="Vehiculo">
              <IntervalInstances>
                <IntervalInstance>V</IntervalInstance>
              </IntervalInstances>
            </Declaration>
          </Declarations>
          <Constraint><![CDATA[V.VelocidadVehiculo < 40.0]]></Constraint>
          <Penalize>0</Penalize>
          <MarkWeight>0</MarkWeight>
          <FeedbackNeg>
          </FeedbackNeg>
          <LearningObjective>Frenada segura en semafo-
ros</LearningObjective>
        </Condition>
        <Condition Name="Condicion2">
          <Declarations>
            <Declaration Name="Vehiculo">
              <IntervalInstances>
                <IntervalInstance>V</IntervalInstance>
              </IntervalInstances>
            </Declaration>
          </Declarations>
          <Constraint><![CDATA[V.LucesCruce > 0.0]]></Constraint>
          <Penalize>0</Penalize>
          <MarkWeight>0</MarkWeight>
          <FeedbackNeg>
          </FeedbackNeg>
          <LearningObjective>Frenada segura en semafo-
ros</LearningObjective>
        </Condition>
      </Conditions>
    </Step>
    <Step Name="Acelerar">
      <Weight>0</Weight>
      <FeedbackPos>
      </FeedbackPos>
      <Conditions>
```

```
            <Condition Name="Condicion">
              <Declarations>
                <Declaration Name="Vehiculo">
                  <IntervalInstances>
                    <IntervalInstance>V</IntervalInstance>
                  </IntervalInstances>
                </Declaration>
              </Declarations>
              <Constraint><![CDATA[V.FrenoDeMano == 0.0]]></Constraint>
              <Penalize>0</Penalize>
              <MarkWeight>0</MarkWeight>
              <FeedbackNeg>
              </FeedbackNeg>
              <LearningObjective>
              </LearningObjective>
            </Condition>
          </Conditions>
        </Step>
        <Step Name="IniciarMarcha">
          <Weight>0</Weight>
          <FeedbackPos>
          </FeedbackPos>
          <Conditions>
            <Condition Name="Condicion">
              <Declarations>
                <Declaration Name="STEPS::IniciarMarcha">
                  <IntervalInstances>
                    <IntervalInstance>IM</IntervalInstance>
                  </IntervalInstances>
                </Declaration>
                <Declaration Name="STEPS::Arrancar">
                  <IntervalInstances>
                    <IntervalInstance>A</IntervalInstance>
                  </IntervalInstances>
                </Declaration>
              </Declarations>
              <Constraint><![CDATA[Duration (A.End, IM.Start)
>60000]]></Constraint>
              <Penalize>0</Penalize>
              <MarkWeight>0</MarkWeight>
              <FeedbackNeg>
              </FeedbackNeg>
              <LearningObjective>
              </LearningObjective>
            </Condition>
          </Conditions>
        </Step>
        <Step Name="CambioCarril">
          <Weight>0</Weight>
          <FeedbackPos>
          </FeedbackPos>
          <Conditions>
            <Condition Name="Condicion">
              <Declarations>
                <Declaration Name="STEPS::CambioCarril">
                  <IntervalInstances>
                    <IntervalInstance>CC</IntervalInstance>
                  </IntervalInstances>
                </Declaration>
              </Declarations>
              <Constraint><![CDATA[Duration (CC) > 3000]]></Constraint>
              <Penalize>0</Penalize>
              <MarkWeight>0</MarkWeight>
              <FeedbackNeg>
              </FeedbackNeg>
              <LearningObjective>
              </LearningObjective>
```

```
                </Condition>
              </Conditions>
            </Step>
          </Steps>
  </Solution>
```

B.2 MODELO DE INTERPRETACIÓN

Esta sección muestra los pasos y las situaciones definidos en el Modelo de Interpretación.

- **Situaciones: Situación ceda, general y semáforo**

```
<Situation Name="SitCeda">
            <Declarations>
              <Declaration Name="CedaAlPaso">
                <IntervalInstances>
                  <IntervalInstance>C</IntervalInstance>
                </IntervalInstances>
              </Declaration>
            </Declarations>
            <Constraints>
              <Start>
              </Start>
              <General><![CDATA[C]]></General>
              <End>
              </End>
            </Constraints>
            <TimeUnit>ms</TimeUnit>
</Situation>
<Situation Name="General">
            <Declarations>
              <Declaration Name="Carriles">
                <IntervalInstances>
                  <IntervalInstance>C</IntervalInstance>
                </IntervalInstances>
              </Declaration>
            </Declarations>
            <Constraints>
              <Start>
              </Start>
              <General><![CDATA[C]]></General>
              <End>
              </End>
            </Constraints>
            <TimeUnit>ms</TimeUnit>
</Situation>
<Situation Name="SitSemaforo">
            <Declarations>
              <Declaration Name="Semaforo">
                <IntervalInstances>
                  <IntervalInstance>S</IntervalInstance>
                </IntervalInstances>
              </Declaration>
            </Declarations>
            <Constraints>
              <Start>
              </Start>
              <General><![CDATA[S]]></General>
              <End>
              </End>
            </Constraints>
            <TimeUnit>ms</TimeUnit>
```

```
</Situation>
```

- **Pasos: IniciarMarcha, Arrancar, Rebasar, Acelerar, Circular, Choque, Off, RebasarCeda, Desbloquear, Contacto, CambioCarril, Arranque**

```
<Step Name="IniciarMarcha">
            <Declarations>
              <Declaration Name="Vehiculo">
                <IntervalInstances>
                  <IntervalInstance>V</IntervalInstance>
                </IntervalInstances>
              </Declaration>
            </Declarations>
            <Constraints>
              <Start>
              </Start>
              <General><![CDATA[V.VelocidadVehiculo > 0.0]]></General>
              <Action>
              </Action>
              <End>
              </End>
            </Constraints>
            <TimeUnit>ms</TimeUnit>
</Step>

<Step Name="Arrancar">
            <Declarations>
              <Declaration Name="Vehiculo">
                <IntervalInstances>
                  <IntervalInstance>V</IntervalInstance>
                </IntervalInstances>
              </Declaration>
              <Declaration Name="STEPS::Contacto">
                <IntervalInstances>
                  <IntervalInstance>C</IntervalInstance>
                </IntervalInstances>
              </Declaration>
              <Declaration Name="STEPS::Arranque">
                <IntervalInstances>
                  <IntervalInstance>A</IntervalInstance>
                </IntervalInstances>
              </Declaration>
              <Declaration Name="STEPS::Desbloquear">
                <IntervalInstances>
                  <IntervalInstance>D</IntervalInstance>
                </IntervalInstances>
              </Declaration>
            </Declarations>
            <Constraints>
              <Start><![CDATA[D [Meets] C [Meets] A]]></Start>
              <General><![CDATA[C [Meets] A]]></General>
              <Action>
              </Action>
              <End><![CDATA[C]]></End>
            </Constraints>
            <TimeUnit>ms</TimeUnit>
</Step>

<Step Name="Rebasar">
            <Declarations>
              <Declaration Name="Semaforo">
                <IntervalInstances>
                  <IntervalInstance>S</IntervalInstance>
                </IntervalInstances>
```

```
              </Declaration>
            </Declarations>
            <Constraints>
              <Start>
              </Start>
              <General><![CDATA[(S.DistanciaSemaforo >2.0) AND (S.DistanciaSemaforo
         < 5.0)]]></General>
              <Action>
              </Action>
              <End>
              </End>
            </Constraints>
            <TimeUnit>ms</TimeUnit>
</Step>

<Step Name="Acelerar">
            <Declarations>
              <Declaration Name="Vehiculo">
                <IntervalInstances>
                  <IntervalInstance>V</IntervalInstance>
                </IntervalInstances>
              </Declaration>
            </Declarations>
            <Constraints>
              <Start>
              </Start>
              <General><![CDATA[V.PorcentajeAceleradorPisado > 0.0]]></General>
              <Action>
              </Action>
              <End>
              </End>
            </Constraints>
            <TimeUnit>ms</TimeUnit>
</Step>
<Step Name="Circular">
            <Declarations>
              <Declaration Name="Vehiculo">
                <IntervalInstances>
                  <IntervalInstance>V</IntervalInstance>
                </IntervalInstances>
              </Declaration>
            </Declarations>
            <Constraints>
              <Start>
              </Start>
              <General><![CDATA[V]]></General>
              <Action>
              </Action>
              <End>
              </End>
            </Constraints>
            <TimeUnit>ms</TimeUnit>
</Step>
<Step Name="Choque">
            <Declarations>
              <Declaration Name="ColisionVehiculo">
                <IntervalInstances>
                  <IntervalInstance>CV</IntervalInstance>
                </IntervalInstances>
              </Declaration>
            </Declarations>
            <Constraints>
              <Start>
              </Start>
              <General><![CDATA[CV.VehiculoColisiona == 1.0]]></General>
              <Action>
              </Action>
```

```
              <End>
              </End>
            </Constraints>
            <TimeUnit>ms</TimeUnit>
</Step>
<Step Name="Off">
            <Declarations>
              <Declaration Name="Vehiculo">
                <IntervalInstances>
                  <IntervalInstance>V</IntervalInstance>
                </IntervalInstances>
              </Declaration>
            </Declarations>
            <Constraints>
              <Start>
              </Start>
              <General><![CDATA[V.LlaveOff > 0.0]]></General>
              <Action>
              </Action>
              <End>
              </End>
            </Constraints>
            <TimeUnit>ms</TimeUnit>
</Step>
<Step Name="RebasarCeda">
            <Declarations>
              <Declaration Name="CedaAlPaso">
                <IntervalInstances>
                  <IntervalInstance>CP</IntervalInstance>
                </IntervalInstances>
              </Declaration>
            </Declarations>
            <Constraints>
              <Start>
              </Start>
              <General><![CDATA[CP.DistanciaCeda > 2.0 AND CP.DistanciaCeda
         <5.0]]></General>
              <Action>
              </Action>
              <End>
              </End>
            </Constraints>
            <TimeUnit>ms</TimeUnit>
</Step>
<Step Name="Desbloquear">
            <Declarations>
              <Declaration Name="Vehiculo">
                <IntervalInstances>
                  <IntervalInstance>V</IntervalInstance>
                </IntervalInstances>
              </Declaration>
            </Declarations>
            <Constraints>
              <Start>
              </Start>
              <General><![CDATA[V.LlaveDesbloqueo > 0.0]]></General>
              <Action>
              </Action>
              <End>
              </End>
            </Constraints>
            <TimeUnit>ms</TimeUnit>
</Step>
<Step Name="Contacto">
            <Declarations>
              <Declaration Name="Vehiculo">
                <IntervalInstances>
```

```
                <IntervalInstance>V</IntervalInstance>
              </IntervalInstances>
            </Declaration>
          </Declarations>
          <Constraints>
            <Start>
            </Start>
            <General><![CDATA[V.LlaveContacto > 0.0]]></General>
            <Action>
            </Action>
            <End>
            </End>
          </Constraints>
          <TimeUnit>ms</TimeUnit>
</Step>
<Step Name="CambioCarril">
          <Declarations>
            <Declaration Name="AcercamientoCarrilIzquierdo">
              <IntervalInstances>
                <IntervalInstance>ACI</IntervalInstance>
              </IntervalInstances>
            </Declaration>
            <Declaration Name="InvasionCarrilIzquierdo">
              <IntervalInstances>
                <IntervalInstance>ICI</IntervalInstance>
              </IntervalInstances>
            </Declaration>
          </Declarations>
          <Constraints>
            <Start><![CDATA[ACI]]></Start>
            <General><![CDATA[ACI [Meets] ICI]]></General>
            <Action>
            </Action>
            <End><![CDATA[ICI]]></End>
          </Constraints>
          <TimeUnit>ms</TimeUnit>
</Step>
<Step Name="Arranque">
          <Declarations>
            <Declaration Name="Vehiculo">
              <IntervalInstances>
                <IntervalInstance>V</IntervalInstance>
              </IntervalInstances>
            </Declaration>
          </Declarations>
          <Constraints>
            <Start>
            </Start>
            <General><![CDATA[V.LlaveArranque > 0.0]]></General>
            <Action>
            </Action>
            <End>
            </End>
          </Constraints>
          <TimeUnit>ms</TimeUnit>
</Step>
```

ANEXO C

ACRÓNIMOS

- AI – ICAI: Véase ICAI
- BBDD: Base de datos
- CAI: Computer-Aided Instruction
- CBM: Constraint-Based Modelling
- CTAT: Cognitive Tutor Authoring Tools
- DE: Desviación estándar
- EDC: Esperar Desde resultado Confirmado
- EDNC: Esperar Desde resultado No Confirmado
- FIPA: Foundation for Intelligent Physical Agents
- GL: Grados de Libertad
- GP: Grupo PATH
- GNP: Grupo No PATH
- HA: Herramienta de Autor
- HMD: Headed Mounted Display
- HMM: Hidden Markov Model
- ICAI: Intelligent Computer-Aided Instruction
- IFT: Intelligent Flight Instructor
- LMA: Laban Movement Analysis
- MAS: Multi-agent system

- MoCap: Motion Capture
- RM: Realidad Mixta
- RV: Realidad Virtual
- RVT: Reactive Virtual Trainer
- SAA: Sistema de Ayuda al Aprendizaje
- SI: Sistema Interactivo
- SIIAA: Sistema Interactivo Inteligente de Ayuda al Aprendizaje
- STEVE: Soar Training Expert for Virtual Environments
- STI: Sistema Tutor Inteligente

ANEXO D

PUBLICACIONES

En este anexo se indican las publicaciones y comunicaciones a congreso que se han generado durante este trabajo de tesis.

D.1 ARTÍCULOS EN REVISTA

Aguirre, A., Lozano-Rodero, A., Matey, L., Villamañe, M. & Ferrero, B. PATH: An authoring tool to model the knowledge for Intelligent Interactive Learning Systems. *Submitted to User modeling and user-adapted interaction*, 2013 (Indexado).

Aguirre, A., Lozano-Rodero, A., Matey, L., Villamañe, M. & Ferrero, B. A novel approach to the diagnosis of motor skills. *Submitted to Learning Technologies, IEEE Transactions on* , 2013 (Indexado).

D.2 COMUNICACIONES A CONGRESOS

Aguirre, A., Lozano-Rodero, A., Villamañe, M., Ferrero, B., & Matey, L. (2012). OLYMPUS : An Intelligent Interactive Learning Platform for Procedural Tasks. In *Proceedings of the International Conference on Computer Graphics Theory and Applications (GRAPP 2012) and of the International Conference on Information Visualization Theory and Applications (IVAPP 2012)* (pp. 543–550). Rome.

BIBLIOGRAFÍA

Abras, C., Maloney-Krichmar, D., & Preece, J. (2004). User-centered design. *Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications*, *37*(4), 445–456.

Aguirre, A., Lozano-Rodero, A., Villamañe, M., Ferrero, B., & Matey, L. (2012). OLYMPUS : An Intelligent Interactive Learning Platform for Procedural Tasks. In *Proceedings of the International Conference on Computer Graphics Theory and Applications (GRAPP 2012) and of the International Conference on Information Visualization Theory and Applications (IVAPP 2012)* (pp. 543–550). Rome.

Aleven, V., Mclaren, B. M., Sewall, J., & Koedinger, K. R. (2006). The Cognitive Tutor Authoring Tools (CTAT): Preliminary Evaluation of Efficiency Gains.

Aleven, V., Mclaren, B. M., Sewall, J., & Koedinger, K. R. (2009). Example-Tracing Tutors : A New Paradigm for Intelligent Tutoring Systems. *International Journal of Artificial Intelligence in Education*, (19(2)), 105–154.

Allen, J. F. (1984). Towards a general theory of action and time. *Artificial Intelligence*, *23*(2), 123–154. doi:10.1016/0004-3702(84)90008-0

Anderson, J., & Corbett, A. (1995). Cognitive tutors: Lessons learned. *The journal of the learning sciences*, *4.2*, 167–207. Retrieved from http://www.tandfonline.com/doi/full/10.1207/s15327809jls0402_2

Anderson, J. R. (1983). The architecture of cognition. Cambridge: MA: Harvard University Press.

Anderson, J. R. (1993). Rules of the mind. Hillsdale: Erlbaum.

Angros, R., Johnson, W. L., Rickel, J., & Scholer, A. (2002). Learning Domain Knowledge for Teaching Procedural Skills. In *Proceeding of the first international joint conference on Autonomous agents and multiagent systems: part3*. ACM.

Bauer, M. (2002). Using Evidence-Centered Design to Align Formative and Summative Assessment". In *Evidence-centered design approach to creating diagnostic e-assessments (ITS'2002)* (pp. 87–96).

Bellifemine, F., Poggi, A., & Rimassa, G. (2001). Developing multi-agent systems with a FIPA-compliant agent framework. *Software Practice and Experience*, *31(2)*, 103–128.

Bilbao, A., Brazalez, A., Garcia, I., Tybel, M., & Aguiriano, N. (2011). Virtual Instrument Cluster for enhancing the configurability of an automotive simulator. *Simulation*, *88*(8), 957–971. doi:10.1177/0037549711425050

Blessing, S. B. (1997). A Programming by Demonstration Authoring Tool for Model – Tracing Tutors. *International Journal of Artificial Intelligence in Education*, *8*, 233–261.

Bossard, C., Querrec, R., & Chevaillier, P. (2010). PEGASE : A Generic and Adaptable Intelligent System for Virtual Reality Learning Environments, *9*(2), 1–13.

Brown, J. S., Burton, R. R., & Zdybel, F. (1973). A model-driven question-answering system for mixed-initiative computer-assisted construction. *Systems, Man and Cybernetics, IEEE Transactions on*, (3), 248–257.

Brusilovsky, P., & Peylo, C. (2003). Adaptive and Intelligent Web-based Educational Systems. *International Journal of Artificial Intelligence in Education*, *13*, 159–172.

Burton, R. R., & Brown, J. S. (1979). An investigation of computer coaching for informal learning activities. *International Journal of Man-Machine Studies*, *11*(1), 5–24.

Carberry, S. (2001). Techniques for Plan Recognition. *User modeling and user-adapted interaction*, *11(1)*, 31–48.

Chan, J., & Leung, H. (2011). A virtual reality dance training system using motion capture technology. *Learning Technologies , IEEE Transactions on*, *4*(2), 187–

195. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5557840

Clancey, W. J. (1979). Tutoring rules for guiding a case method dialogue. *International Journal of Man-Machine Studies*, *11*(1), 25–49.

Cotin, S., Stylopoulos, N., Ottensmeyer, M., Neumann, P., Rattner, D., & Dawson, S. (2002). Metrics for laparoscopic skills trainers: the weakest link! In *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2002* (pp. 35–43). Springer.

Davcev, D., Trajkovic, V., Kalajdziski, S., & Celakoski, S. (2003). Augmented reality environment for dance learning. In *Information Technology: Research and Education, 2003. Proceedings. ITRE2003. International Conference on* (pp. 189–193).

Duan, F., Zhang, Y., Pongthanya, N., Watanabe, K., Yokoi, H., & Arai, T. (2008). Analyzing human skill through control trajectories and motion capture data. *2008 IEEE International Conference on Automation Science and Engineering*, 454–459. doi:10.1109/COASE.2008.4626426

Echegaray-López, G. (2013). *Rigid and Deformable Collision Handling for a Haptic Neurosurgery Simulator*. University of Navarra.

Ferrero, B. (2004). *DETECTive: un entorno genérico e integrable para diagnóstico de actividades de aprendizaje. Fundamento, diseño y evaluación*. University of the Basque Country UPV/EHU.

Fournier-Viger, P., & Nkambou, R. (2009). Exploiting Partial Problem Spaces Learned from Users ‟ Interactions to Provide Key Tutoring Services in Procedural and Ill-Defined Domains. In *Artificial Intelligence in Education: Building Learning Systems that Care: From Knowledge Representation to Affective Modelling 200* (p. 383).

Fournier-Viger, P., Nkambou, R., & Nguifo, E. M. (2010). ITS in Ill-Defined Domains : Toward Hybrid Approaches An Hybrid Model in CanadarmTutor. In *Intelligent Tutoring Systems* (pp. 318–320). Springer Berlin / Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-13437-1_57

Fournier-Viger, P., Nkambou, R., & Nguifo, E. M. (2011). Learning Procedural Knowledge from User Solutions to Ill-Defined Tasks in a Simulated Robotic Manipulator I . The CanadarmTutor Tutoring System. In R. et al. (Ed.), *Handbook of Educational DataMining* (pp. 451–465).

Fournier-Viger, P., Nkambou, R., Nguifo, E. M., Mayers, A., & Faghihi, U. (2013). A Multi-Paradigm Intelligent Tutoring System for Robotic Arm Training. *IEEE Transactions on Learning Technologies*, 1–1. doi:10.1109/TLT.2013.27

Gagné, R. M. (1965). The conditions of learning. Holt, Rinehart and Winston New York.

Gutierrez, T., Casado, S., & Carrillo, A. (2008). Examples of skills capturing and transfer systems in different domain such as the industrial maintenance field. In *Proceedings of IDMME Virtual Concept*.

Gutiérrez, T., Rodríguez, J., Vélaz, Y., Suescun, S. C. A., & Sánchez, E. J. (2010). IMA-VR : A Multimodal Virtual Training System for Skills Transfer in Industrial Maintenance and Assembly Tasks. In *RO-MAN, 2010 IEEE* (pp. 428–433).

Hachimura, K., Kato, H., & Tamura, H. (2004). A prototype dance training support system with motion capture and mixed reality technologies. In *Robot and Human Interactive Communication, 2004. ROMAN 2004. 13th IEEE International Workshop on* (pp. 217–222).

Hachimura, K., Takashina, K., & Yoshimura, M. (2005). Analysis and evaluation of dancing movement based on LMA. In *Robot and Human Interactive Communication, 2005. ROMAN 2005. IEEE International Workshop on* (pp. 294–299).

Hsieh, P. Y., Halff, H. M., Redfield, C. L., & Division, T. T. (1999). Four Easy Pieces : Development Systems for Knowledge-Based Generative Instruction. *International Journal of Artificial Intelligence in Education*, *10*, 1–45.

Kabanza, F., & Nkambou, R. (2005). Path-planning for autonomous training on robot manipulators in space. In *IJCAI* (pp. 1729–1731). Morgan Kaufmann Publishers Inc. Retrieved from http://www.ijcai.org/Past Proceedings/IJCAI-05/PDF/post-0130.pdf

Koedinger, K. R., Anderson, J. R., Hadley, W. H., & Mark, M. A. (1997). Intelligent Tutoring Goes To School in the Big City. *International Journal of Artificial Intelligence in Education (IJAIED) 8*, *8*, 30–43.

Kwon, D. Y., & Gross, M. (2005). Combining body sensors and visual sensors for motion training. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology* (pp. 94–101).

Lebeau, J., Fortin, M., Paquette, L., & Mayers, A. (2009). From Cognitive to Pedagogical Knowledge Models in Problem-Solving ITS Frameworks. In

Artificial Intelligence in Education: Building Learning Systems that Care: From Knowledge Representation to Affective Modelling (pp. 731–733). IOS Press.

Loke, L., Larssen, A., & Robertson, T. (2005). Labanotation for design of movement-based interaction. *Proceedings of the second Australasian conference on Interactive entertainment*, *2005*. Retrieved from http://dl.acm.org/citation.cfm?id=1109197

Lopez-Garate, M. (2011). *Sistema de Selección de Feedback Adaptativo y Configurable para Sistemas Interactivos Inteligentes de Ayuda al Aprendizaje*. University of Navarra.

Lopez-Garate, M., Lozano-Rodero, A., & Matey, L. (2008). AN ADAPTIVE AND CUSTOMIZABLE FEEDBACK SYSTEM FOR VR-BASED TRAINING SIMULATORS (Short Paper). In *7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*.

Lozano-Rodero, A. (2009). *Metodología de Desarrollo de Sistemas Interactivos Inteligentes de Ayuda al Aprendizaje de Tareas Procedimentales basados en Realidad Virtual y Mixta*. University of Navarra. Retrieved from http://hdl.handle.net/10171/27734

Martin, B., & Mitrovic, A. (2003). Domain Modelling: Art or Science? In *11th Int. Conference on Artificial Intelligence in Education* (pp. 183–190). Sidney.

Matsuda, N., Cohen, W. W., Sewall, J., Lacerda, G., & Koedinger, K. R. (2007). Predicting Students ' Performance with SimStudent : Learning Cognitive Skills from Observation 1. *Learning*.

McCarthy, L., Stiles, R., Johnson, W. L., & Rickel, J. (1999). Human-systems interaction for virtual environment team training. *Virtual Reality*, *4*(1), 38–48.

Medved, V. (2001). *Measurement of human locomotion*. CRC PressI Llc.

Mitrovic, A. (1998). Experiences in Implementing Constraint-Based modeling in SQL-TUTOR. In *Intelligent Tutoring Systems* (pp. 414–423). Springer Berlin / Heidelberg.

Mitrovic, A. (2011). Fifteen years of constraint-based tutors: what we have achieved and where we are going. *User Modeling and User-Adapted Interaction*, *22*(1-2), 39–72. doi:10.1007/s11257-011-9105-9

Mitrovic, A., Koedinger, K., & Martin, B. (2003). *A Comparative Analysis of Cognitive Tutoring and Constraint-Based Modeling*. *Inteligent Tutoring Systems* (pp. 313–322). Springer Berlin / Heidelberg.

Mitrovic, A., Martin, B., & Suraweera, P. (2007). Intelligent tutors for all: Constraint-based modeling methodology, systems and authoring.

Mitrovic, A., Martin, B., Suraweera, P., Zakharov, K., & Mcguigan, N. (2009). ASPIRE : An Authoring System and Deployment Environment for Constraint-Based Tutors. *International Journal of Artificial Intelligence in Education*, *19*, 155–188.

Mitrovic, A., & Weerasinghe, A. (2009). Revisiting Ill-Definedness and the Consequences for ITSs. *Artificial Intelligence in Education: Building Learning Systems That Care: from Knowledge Representation to Affective Modelling*, *200*, 375.

Mitrovic, A., McGuigan, N., Martin, B., Suraweera, P., Milik, N., Holland, J. (2008). Authoring constraint-based systems in ASPIRE: a case study of a capital investment tutor. In *ED-MEDIA 2008* (pp. 4607–4616).

Munro, A., & Surmon, D. (1997). Primitive simulation-centered tutor services. In *Proceedings of the AI-ED Workshop on Architectures for Intelligent Simulation-Based Learning Environments*.

Munro, A., Johnson, M.C., Pizzini, Q.A., Surmon, D.S., Towne, D.M., Wogulis, J. L. (1997). Au thoring simulation centered tutors with RIDES. *International Journal of Artificial Intelligence in Education 8(3-4)*, 284–316.

Murphy, T. E., Vignes, C. M., Yuh, D. D., & Okamura, A. M. (2003). Automatic Motion Recognition and Skill Evaluation for Dynamic Tasks. In *Eurohaptics* (pp. 363–373).

Murray, T. O. M. (2003). AN OVERVIEW OF INTELLIGENT TUTORING SYSTEM AUTHORING TOOLS : Updated Analysis of the State of the Art. *Authoring Tools for Advanced Technology Learning Environments*, 493–546.

Nkambou, R., & Kabanza, F. (2001). Designing Intelligent Tutoring Systems: A multiagent Planning Approach. *ACM SIGCUE Outlook 27(2)*, 46–60.

Nkambou, Roger, & Bourdeau, J. (2010). Building Intelligent Tutoring Systems : An Overview, 361–375.

Noot, H., & Ruttkay, Z. (2003). The GESTYLE Language. In *International workshop on gesture and sign language based human–computer interaction*. Genova, Italy.

Ohlsson, S. (1992). Constraint-based student modeling. *Artificial Intelligence and Education 3 (4)*, 429–447.

Ong, J. C., Noneman, S. R., & Group, O. T. (2000). Intelligent tutoring systems for procedural task training of remote payload operations at NASA. In *The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC)* (Vol. 2000).

Ostiategui, F., Amundarain, A., Lozano-Rodero, A., & Matey, L. (2010). Gardening Work Simulation Tool in Virtual Reality for Disabled People Tutorial. In *Proceedings of Integrated Design and Manufacturing - Virtual Concept (IDMME'10)*. Bordeaux, France.

Panjkota, A., Stančić, I. V. O., & Šupuk, T. (2009). Outline of a Qualitative Analysis for the Human Motion in Case of Ergometer Rowing. In *WSEAS International Conference Proceeding. Mathematics and Computers in Science and Engineering. WSEAS*.

Paquette, L., Lebeau, J., & Mayers, A. (2010). Authoring Problem-Solving Tutors : A Comparison between ASTUS and CTAT. *Advances in Intelligent Tutoring Systems*, 377–405.

Qian, G., Guo, F., Ingalls, T., Olson, L., James, J., & Rikakis, T. (2004). A gesture-driven multimodal interactive dance system. In *Multimedia and Expo, 2004. ICME'04. 2004 IEEE International Conference on* (Vol. 3, pp. 1579–1582).

Ramachandran, S., Remolina, E., & Fu, D. (2004). FlexiTrainer : A Visual Authoring Framework for Case- Based Intelligent Tutoring Systems FlexiTrainer : A Visual Authoring Framework for Case-Based Intelligent. *Intelligent Tutoring Systems*, 59–80.

Remolina, E., Ramachandran, S., Fu, D., Stottler, R., & Howse, W. R. (2004). Intelligent Simulation-Based Tutor for Flight Training. In *The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC)*. National Training Systems Association.

Rickel, Je, & Johnson, W. L. (1999). Virtual Humans for Team Training in Virtual Reality. In *Proceedings of the ninth international conference on artificial intelligencen in education* (p. 585).

Rickel, Jeff, Gratch, J., HIll, R., Marsella, S., & Swartout, W. (2001). Steve Goes to Bosnia : Towards a New Generation Virtual Humans for Interactive Experiences. In *AAAI spring symposium on artificial intelligence and interactive entertainment*.

Ritter, S., Anderson, J. R., Koedinger, K. R., & Corbett, A. (2007). Cognitive Tutor: Applied research in mathematics education. *Psychonomic bulletin & review*, *14*(2), 249–255.

Rosen, J., Chang, L., Brown, J. D., Hannaford, B., Sinanan, M., & Satava, R. (2003). Minimally invasive surgery task decomposition-etymology of endoscopic suturing. *Studies in Hehtmlalth Technology and Informatics*, 295–301.

Rosen, J., Hannaford, B., Richards, C. G., & Sinanan, M. N. (2001). Markov modeling of minimally invasive surgery based on tool/tissue interaction and force/torque signatures for evaluating surgical skills. *Biomedical Engineering, IEEE Transactions on*, *48*(5), 579–591.

Rosen, J., Solazzo, M., Hannaford, B., & Sinanan, M. (2002). Task decomposition of laparoscopic surgery for objective evaluation of surgical residents' learning curve using hidden Markov model. *Computer Aided Surgery*, *7*(1), 49–61.

Ruttkay, Z., & Welbergen, H. Van. (2008). Elbows higher! performing, observing and correcting exercises by a virtual trainer. *Intelligent Virtual Agents*. Retrieved from http://link.springer.com/chapter/10.1007/978-3-540-85483-8_41

Ruttkay, Z., & Zwiers, J. (2006). Towards a reactive virtual trainer. *Intelligent Virtual Agents*. Retrieved from http://link.springer.com/chapter/10.1007/11821830_24

Self, J. A. (1974). Student models in computer-aided instruction. *International Journal of Man-machine studies*, *6*(2), 261–276.

Self, J., Based, C., Unit, L., & Ls, L. (1999). The defining characteristics of intelligent tutoring systems research : ITSs care , precisely. *International Journal of Artificial Intelligence in Education*, *10*(1998), 350–364.

Shute, V. J. (2004). Towards Automating ECD-Based Diagnostic Assessments". *Technology, Instruction, Cognition and Learning (TICL)*, *2*.

Simon, H. A. (1978). Information-processing theory of human problem solving. *Handbook of learning and cognitive processes*, *5*, 271–295.

Soga, A., Umino, B., & Hirayama, M. (2009). Automatic composition for contemporary dance using 3D motion clips: Experiment on dance training and system

evaluation. In *CyberWorlds, 2009. CW'09. International Conference on* (pp. 171–176).

Solis, J., & Takanishi, A. (2008). Enabling autonomous systems to perceptually detect human performance improvements and their applications. In *Automation Science and Engineering, 2008. CASE 2008. IEEE International Conference on* (pp. 259–264).

Sowa, J. F., Grove, P., Cole, C. A. B., & Shapiro, S. C. (2011). Book reviews. *The journal of the American Academy of Psychoanalysis and Dynamic Psychiatry*, *39*(4), 753–72. doi:10.1521/jaap.2011.39.4.753

Stiles, R. (1999). Virtual Environments for Training Final Report. *Report No. CDRL A0002 Final Report). Palo Alto, CA: Advanced Technology Center, Lockheed Martin Missiles and Space*.

Suraweera, P., & Mitrovic, A. (2002). KERMIT: a constraint-based tutor for database modeling. In *Intelligent Tutoring Systems* (pp. 377–387).

Tervo, K., Palmroth, L., & Koivo, H. (2010). Skill Evaluation of Human Operators in Partly Automated Mobile Working Machines. *Automation Science and Engineering, IEEE transactions on*, *7*(1), 133–142.

Theodoridis, S., & Koutroumbas, K. (1999). Pattern Recognition. *CA: Academic*.

Unzueta, L., Mena, O., Sierra, B., & Suescun, Á. (2008). Kinetic pseudo-energy history for human dynamic gestures recognition. In *Articulated Motion and Deformable Objects* (pp. 390–399). Springer.

Van Joolingen, W.R., King, S., de Jong, T. (1997). The SimQuest authoring system for simulation- based discovery learning. *Artificial intelligence and education: Knowledge and media in learning systems. IOS Press, Amsterdam*.

Van Merriënboer, J. J. G. (1997). Training Complex Cognitive Skills: A Four-Component Instructional Design Model for Technical Training. *Educational Technology Publications*.

VanLehn, K., Lynch, C., K., S., Shapiro, J. A., Shelby, R., Taylor, L., … Wintersgill, M. (2005). The Andes physics tutoring system: lessons learned. *International Journal of Artificial Intelligence in Education 15(3)*, 147–204. Retrieved from http://www.dtic.mil/dtic/tr/fulltext/u2/a521924.pdf

Vassileva, J., Calla, G. M. C., & Greer, J. I. M. (2003). Multi-Agent Multi-User Modeling in I-Help. *User modeling and user-adapted interaction*, 179–210.

Vemer, L., Oleynikov, D., Holtmann, S., Haider, H., & Zhukov, L. (2003). Measurements of the Level of Surgical Expertise Using Flight Path Analysis from da Vinci™ Robotic Surgical System. *Medicine Meets Virtual Reality 11: NextMed: Health Horizon*, *94*, 373.

Vicari, R. M., & Gluz, J. G. (2007). An Intelligent Tutoring System (ITS) view on AOSE. *International Journal of Agent-Oriented Software Engineering 1(3-4)*, 295–333.

Weinland, D., Ronfard, R., & Boyer, E. (2011). A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding*, *115*(2), 224–241. doi:10.1016/j.cviu.2010.10.002

Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems*. *Morgan Kaufmann Publishers Inc*. Morgan Kaufmann.

Wenzel, B. M., Dirnberger, M. T., Hsieh, P. Y., Chudanov, T. J., Halff, H. M. (1998). Evaluating Subject Matter Experts' Learning and Use of an ITS Authoring Tool. In *Proceedings of the 4th International Conference on Intelligent Tutoring Systems*.

Wescourt, K. T., Beard, M., & Gould, L. (1977). Knowledge-based adaptive curriculum sequencing for CAI: Application of a network representation. In *Proceedings of the 1977 annual conference* (pp. 234–240).

Yamauchi, Y., Yamashita, J., Morikawa, O., Hashimoto, R., Mochimaru, M., Fukui, Y., … Yokoyama, K. (2002). Surgical skill evaluation by force data for endoscopic sinus surgery training system. In *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2002* (pp. 44–51). Springer.

Yang, J., Xu, Y., & Chen, C. S. (1997). Human action learning via hidden Markov model. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, *27*(1), 34–44.

Zadeh, L. A. (1975). The Concept of a Linguistic Variable and its Application to Approximate Reasoning-I. In *Information Sciences 8 (3)* (Vol. 249, pp. 199–249).

Zadeh, L. A. (1997). Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. *Fuzzy sets and systems*, *90*(2), 111–127.