TESIS DOCTORAL

Notación gráfica para la representación de dominios de planificación jerárquica orientada al uso comercial

Universidad de Granada

Departamento de Ciencias de la

Computación e Inteligencia Artificial

Autor: Francisco Carlos Palao Reinés.

Directores: Dr. Luis Castillo Vidal, Dr. Juan Fernandez Olivares.

Granada, Marzo de 2011

Editor: Editorial de la Universidad de Granada
Autor: Francisco Carlos Palao Reinés
D.L.: GR 3135-2011
ISBN: 978-84-694-3561-8

La memoria titulada "Notación Gráfica para la representación de dominios de planificación jerárquica orientada al uso comercial" que presenta D. Francisco Carlos Palao Reinés para optar al grado de doctor, ha sido realizada dentro del programa de doctorado "Diseño, Análisis y Aplicaciones de Sistemas Inteligentes" del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada así como en el seno de la spin-off de la IActive Intelligent Solutions bajo la dirección de los Doctores D. Luis Castillo Vidal y D. Juan Fernandez Olivares.

Granada, Marzo de 2010

El Doctorando Los Directores

Fdo: Francisco Carlos Palao Reinés Fdo: Luis Castillo Vidal Fdo: Juan Fernandez Olivares

Tesis Doctoral financiada por los siguientes programas:

- Red Nacional de Planificación, Scheduling y Razonamiento Temporal, RNPST, TIN2004-20168E.
- 2003 – 2006: Proyecto SIADEX. NET033957/1. Proyecto financiado por la Consejería de Medio Ambiente de Andalucía para el Desarrollo de un Sistema Inteligente de Ayuda a la Toma de Decisiones en Extinción de Incendios Forestales.
  Proyecto desarrollado en el ámbito de la Universidad de Granda.
- 2007 – 2010: Programa Torres Quevedo. Ministerio de Ciencia e Innovación.
  Proyecto desarrollado en el ámbito de IActive Intelligent Solutions.
- 2007 – 2011: Proyecto NEOTEC (CDTI). Herramientas para el Desarrollo de Sistemas Inteligentes basados en Tecnologías del Conocimiento.
  Proyecto desarrollado en el ámbito de IActive Intelligent Solutions.

Índice

Agradecimientos

Esta tesis no habría podido ser una realidad sin el apoyo y la colaboración que me han dado mis directores de tesis, mis compañeros de trabajo desde la Universidad de Granada y desde IActive, y por supuesto, mi familia y amigos.

Agradezco a Luis Castillo y a Juan Fernandez, mis directores de tesis, el sobreesfuerzo que han tenido que realizar para dirigir la escritura de esta tesis y compaginarla con mi complicada agenda. Y sobre todo, les agradezco que hayan tenido la visión de apoyar una tesis con un enfoque de transferencia de tecnología, que no deja de ser un enfoque diferente a las tesis que habitualmente suelen escribirse, aunque yo particularmente estoy totalmente convencido de que es la mejor manera de maximizar las posibilidades de que los resultados de investigación lleguen al mercado y a la sociedad para su beneficio.

Agradezco a Óscar García, a Tomás Garzón, a Raúl Raya, a Eva Hidalgo y a todo el Departamento de Tecnología de IActive el trabajo que realizan en la empresa y que ha hecho realidad los planteamientos de esta tesis.

Agradezco a mis amigos, a los que tanto valoro y aprecio por que siguen tratándome y teniéndome tan en cuenta siempre como si nos viésemos a diario, a pesar que nos vemos con muy poca frecuencia por mi constante nivel de trabajo.

Agradezco a mi "abuelucha" Emilia por enseñarme lo importante que es cuidar de los demás y a ser una persona prudente y cariñosa. También agradezco a mis abuelos, que desafortunadamente ya no están aquí pero que han sido, junto con mis padres, los cimientos de lo que hoy en día soy. A mi abuelo Juan por su carácter emprendedor que sé que he heredado, a mi abuela Carmen por enseñarme lo importante que es creer ciegamente en algo y a mi abuelo Amadeo por enseñarme a tener paciencia mientras pescábamos en la playa y a saber aprovechar las oportunidades.

Y por último, y más importante, agradezco a mi familia. A mi mujer María del Mar por estar siempre cuando la necesito, por su enorme paciencia y por haberme enseñado a querer de verdad; a mi padre Francisco por haberme enseñado a ser una persona organizada y luchadora, gracias a la educación en el deporte que siempre nos ha transmitido; a mi madre Emilia por haberme enseñado a ser una persona cariñosa y respetuosa con los demás; y a mi hermano Jorge por haberme enseñado que la familia es lo más importante y que cuando hay problemas, una risa siempre te hacer ver las cosas de un modo diferente. Espero poder recompensaros algún día y que pronto podamos disfrutar de muchos más momentos en compañía de la familia que, al fin y al cabo, es lo más importante en este mundo.

A todos vosotros y a las personas que por mi (habitual) falta de memoria no haya mencionado, GRACIAS.

Contribuciones de la Tesis

La tesis desarrollada presenta un proceso de transferencia tecnológica que comienza a partir de una necesidad de mercado y es cubierta mediante los resultados de investigación básica en el área de Inteligencia Artificial Planning & Scheduling, pasando por las etapas de investigación aplicada mediante el desarrollo de prototipos no comerciales y dando como resultado aplicaciones posicionadas en el mercado de alto valor añadido que presentan un fuerte grado de innovación gracias al avance científico y tecnológico en el que están basadas.

El proceso de transferencia tecnológica descrito, está ampliamente reconocido con las siglas I+D+i (Investigación + Desarrollo + innovación). **A continuación se describen las distintas aportaciones de la tesis, clasificándolas en las distintas etapas del proceso de I+D+i**:

- **Investigación (2003-2007)**

  El trabajo de investigación comenzó con el desarrollo de un planificador jerárquico [1] que inicialmente fue aplicado para la planificación automática de operaciones de extinción de incendios como sistema de ayuda en la toma de decisiones en incendios forestales en el proyecto SIADEX [2].

  Para obtener el rendimiento adecuado del planificador, fue necesario el uso y adaptación de técnicas de extracción y representación del conocimiento, de forma que el planificador tomase decisiones adecuadas, tal y cómo lo haría un experto en extinción [3].

  También fue necesario dotar al planificador de capacidad de razonamiento temporal difuso de manera que permitiese la monitorización de la ejecución del plan, dando respuesta de forma flexible a los retrasos y posibles desfases en la ejecución de las tareas del mismo [4].

  Por último, se hizo un especial esfuerzo en la integración del planificador en una arquitectura compuesta por servicios web [5], de este modo se facilitó la conexión del planificador en distintos entornos [6] y permitió su aplicación a distintos problemas.

  Estos avances, especialmente las capacidades de gestión temporal del planificador, como se verá a lo largo de la memoria, fueron claves para desarrollar sofisticadas tecnologías que representan un avance considerable en la oferta tecnológica actual del mercado.

  Es importante destacar que, además de los artículos publicados en revista de índice de impacto, los capítulos de libro y los artículos en la conferencia más importante a nivel internacional en el área (ICAPS), el trabajo desarrollado en el proyecto SIADEX obtuvo el primer premio a la aplicación en el congreso ICAPS de 2006 y a la transferencia tecnológica en el CEDI 2005.

- **Desarrollo (2007 - 2010)**.

  A partir de los resultados de la etapa anterior, tras hacer un análisis de mercado para detectar nichos en los que poder posicionar productos comerciales, se creó una spin-off, llamada IActive, para transferir al mercado los resultados de investigación desarrollados.

  Tal y cómo se describirá en esta memoria, se planteó el desarrollo de una herramienta que permitiera a Ingenieros Informáticos que no tuvieran experiencia previa en el área científica de Planificación Automática, desarrollar sistemas basados en el planificador automático desarrollado previamente. Para ello fue necesario el desarrollo de una notación gráfica que permitiese a los usuarios de la herramienta modelar el conocimiento necesario para la adaptación del planificador a distintos entornos y problemas [7].

  El producto desarrollado se denominó IActive Knowledge Studio y, tras menos de un año desde su lanzamiento, ha conseguido que el número de usuarios del planificador inteligente se haya multiplicado considerablemente y hoy en día sea usado por varias compañías privadas que lo están integrando en sus productos para aportar valor añadido a los mismos.

  Durante esta etapa también se han obtenido distintos premios empresariales de reconocido prestigio nacional por el valor añadido que presenta la empresa al mercado gracias a la investigación previa en la que se basan sus productos. Algunos ejemplos son el prestigioso premio "Emprendedor XXI" (2010) otorgado por La Caixa y el premio "Empresas de Base Tecnológica" (2011) otorgado por Bancaja.

- **Innovación (Actualidad)**.

  Se entiende por innovación cualquier cambio o avance en productos existentes o desarrollo de nuevos productos que el mercado está dispuesto a adquirir.

  En este sentido, gracias a los productos desarrollados, se ha tenido un resultado en innovación real mediante productos y soluciones de alto valor añadido [8] que están siendo adquiridas por importantes clientes. Además de la positiva respuesta del mercado y del aumento del volumen de negocio de la compañía, importantes analistas de tendencias tecnológicas como Gartner han reconocido el importante elemento de innovación que caracteriza a los productos y soluciones que desarrolla la empresa [9].

  Los avances desarrollados por IActive no sólo están permitiendo el incremento de su volumen de negocio sino que también están facilitando la captación de financiación por parte de Capital Riesgo. En 2009 la compañía obtuvo una primera ronda de inversión de 500.000 € y a comienzos de 2011 una segunda de 3 millones de euros para el desarrollo comercial y el posicionamiento en el mercado de sus innovadores productos desarrollados a partir de los resultados de investigación que describe esta memoria.

I. Memoria

1. Motivación de la tesis

1.1 Océanos Azules y Océanos Rojos

El término de *Océano Azul*[10] hace referencia a los nuevos mercados que son creados, gracias a la existencia de un nicho sin cubrir o en áreas que no están explotadas en la actualidad, y que generan oportunidades de crecimiento rentable y sostenido a largo plazo.

La búsqueda permanente de Océanos Azules es una labor obligatoria y totalmente necesaria para la continuidad de cualquier compañía en el mercado.

De hecho, se denomina *Océano Rojo* a aquel mercado en el que el nivel de competencia es fuerte y la rentabilidad de las compañías posicionadas en él se reduce drástica y continuamente ya que conforme el mercado madura, la estrategia que adoptan los distintos competidores es la de reducción de precios y minimizar costes.

Las compañías que se posicionan en mercados maduros tienen un alto riesgo de desaparecer si estos se convierten en *Océanos Rojos* y previamente no han sido capaces de diseñar y ejecutar una estrategia que les haya llevado a *Océanos Azules*.

Océano Roj
Mercado satura
altamente compe
Océano Azul
Nuevo nicho de Mercado

En la próxima sección analizaremos el mercado relativo a las tecnologías de automatización de procesos de negocio y ayuda en la toma de decisiones en empresas.

Entendemos por tecnologías de automatización de procesos de negocio aquellas que permiten a las entidades dar soporte a la coordinación entre las distintas actividades y operaciones, relacionadas unas con otras, para cumplir los objetivos establecidos.

Este tipo de tecnologías están, por norma general, muy asociadas a tecnologías de ayuda a la toma de decisiones ya que, éstas últimas, ayudan al personal de las compañías a tomar las decisiones adecuadas para, no sólo automatizar los procesos, sino decidir cuál es el proceso más adecuado en cada ocasión.

1.2 Mercado actual de las tecnologías de automatización de procesos

A continuación analizaremos el estado actual del mercado, en relación a su oferta, de las tecnologías de automatización de procesos de negocio y soporte a la ayuda en la toma de decisiones para entornos de gestión empresarial, dejando fuera del entorno competitivo definido la oferta en entornos industriales.

Existen distintas tecnologías de uso comercial para la automatización de procesos y dar soporte a la toma de decisiones para la gestión empresarial**.** Algunas de ellas son capaces de automatizar procesos, generando una secuencia de acciones a ejecutar para alcanzar los objetivos planteados. Mientras que, otras tecnologías existentes, aunque no generan secuencias de acciones de forma automática, sí prestan una ayuda directa al usuario para que él mismo diseñe el proceso necesario para alcanzar los objetivos establecidos.

A continuación se describen las tecnologías del entorno competitivo de automatización de procesos de negocio y soporte a la ayuda en la toma de decisiones:

- **BPM (Business Process Management):** Basada en tecnologías de automatización de procesos y workflow [11][12], se ha desarrollado una categoría comercial denominada BPM que consiste en una metodología de definición de procesos [13] que cuenta con numerosos entornos de modelado y motores de ejecución comerciales [14].

  La metodología BPM cuenta con cuatro fases:

  1) Diseño de los procesos de negocio: En la que se diseñan los procesos que pretenden automatizarse.
  2) Modelado de los procesos de negocio: Se modelan mediante una herramienta de modelado que permitirá describirlos en la notación BPMN [15].
  3) Automatización de procesos de negocio: Un motor de ejecución automatiza los procesos de negocio y ofrece la secuencia de pasos necesaria para alcanzar los objetivos marcados basados en los procesos previamente definidos.
  4) Optimización continua: Tras la ejecución del proceso de negocio o simulación del mismo en el entorno de modelado, se modifica el propio proceso para optimizar la automatización del mismo.

- **BRMS (Business Rules Management System):** Se han desarrollado motores de reglas [16] que hoy en día constituyen una categoría comercial denominada BRMS que consiste en componentes software que permiten la automatización de decisiones complejas cuando el número de variables a tener en cuenta es alto. Este tipo de sistemas no están destinados a sintetizar soluciones, es decir, no ofrecen decisiones compuestas de múltiples acciones. Al contrario, están basados en un proceso analítico que devuelve como resultado una única acción acompañada de los parámetros necesarios para optimizar el resultado y alcanzar el objetivo planteado.

  Actualmente, estos sistemas se están integrando en suites de BPM para dotar a las ramas condicionales de los procesos BPMN de una capacidad mayor a la hora de elegir cuál es el camino correcto, mediante la evaluación de un gran número de variables y una lógica de decisión más compleja, durante la ejecución del proceso.

- **BI (Business Intelligence):** Basados en tecnologías de minería de datos [17] [18], se ha creado una categoría comercial denominada BI que consiste en sistemas que facilitan la extracción, procesamiento y conversión de datos a información y conocimiento útil para la ayuda en la toma de decisiones. Mediante este tipo de sistemas, los usuarios pueden visualizar, normalmente de un modo gráfico mediante la incorporación de gráficas, la información procedente de múltiples fuentes lo que la facilita la toma de decisiones.

- **Otros sistemas sustitutivos de ayuda a la toma de decisiones para cada entorno**: Nos encontramos que para cada aplicación o entorno concreto, distintos sistemas software podrían facilitar la ayuda en la toma de decisiones a los usuarios. Por ejemplo, en el entorno de ayuda en la toma de decisiones para la extinción de incendios forestales, un GIS (Geographic Information System) facilita la toma de decisiones mediante la presentación en pantalla de toda la información necesaria para la toma de decisiones representada en un mapa geográfico. Aunque es el usuario quien tiene que calcular y tomar la decisión desde cero, la realidad es que este tipo de sistemas presenta una gran ayuda para realizar dicha labor.

Con el objetivo de realizar una comparación entre las prestaciones y necesidades que cubre cada una de las tecnologías enumeradas, se va a realizar **un análisis de fortalezas y debilidades**.

BPM (Business Process Management)	
Fortalezas	**Debilidades**
• Ofrece una representación gráfica y concreta de los procesos lo que ayuda a definirlos, entenderlos y comunicarlos. • Al definir el proceso queda una imagen muy clara de cómo podría ser la ejecución final del sistema ya que contempla ramas condicionales y, en definitiva, explicita todos los caminos posibles • La ejecución se realiza sobre el proceso definido con ramas condicionales. • Es más intuitivo para el experto de negocio. • Suelen estar bien integrados con arquitecturas SOA para la ejecución del proceso. • Es posible realizar simulaciones sobre el propio proceso definido para optimizarlo. • Cuentan con muchas herramientas de modelado de procesos. • Agilidad en tiempo de respuesta para la automatización y ejecución del proceso.	• No maneja adecuadamente entornos con gran número de variables ya que aumenta la complejidad de los modelos al contemplar variables de entorno. • No permite el modelado de procesos dinámicos ya que aumenta la complejidad de los modelos al definir múltiples ramas condicionales (código espagueti). • Las ejecuciones finales son muy estáticas ya que siempre se realizan sobre el esquema determinado por el proceso definido previamente. • Sistema reactivo por pasos. No es capaz de seleccionar acciones en base a una planificación previa sino que ejecuta y decide paso a paso. • Los procesos complejos y flexibles los envían a los usuarios para que ellos los desempeñen. Está orientado a automatizar procesos de negocio estáticos. • Requieren mucha infraestructura y esfuerzo de implantación. • No ofrecen información temporal mediante una métrica establecida sobre las actividades que forman el proceso.

BRMS (Business Rules Management System)	
Fortalezas	**Debilidades**
• Permiten tener en cuenta una gran cantidad de variables para automatizar decisiones y optimizar su resultado. • Se reducen costes ya que extraen la lógica del software del código interno y para cambiar el comportamiento de los sistemas y no es necesario reprogramar las aplicaciones. • Facilitan el entendimiento de las acciones tomadas, ya que se almacenan todas las decisiones tomadas y sus parámetros de forma que después puede saberse porqué se tomaron estas decisiones. • Incluyen entornos de desarrollo para expertos de negocio (no tan cercanos como BPM) y para técnicos. • Eficiencia en tiempo de respuesta.	• El resultado es limitado a una acción o decisión: Son sistemas de ayuda a la toma de decisiones que son capaces de tomar una decisión concreta en función a multitud de parámetros pero no dan como resultado varias acciones. • Cuando se integran en un BPM, deciden paso a paso: no tienen en cuenta las decisiones que tomarán más adelante en todo el proceso por lo que no aseguran obtener el objetivo global.

BI (Business Intelligence)	
Fortalezas	**Debilidades**
• Son capaces de integrarse con una gran cantidad de fuente de datos. • Permiten procesar una gran cantidad de datos para facilitar su entendimiento. • Aportan entornos gráficos para la representación de los datos analizados. • Existen una gran cantidad de herramientas comerciales que dan soporte al desarrollo de módulos BI en sistemas software. • Eficiencia en el tiempo de respuesta.	• No generan automáticamente decisiones, ni únicas ni compuestas por distintas acciones. • Sólo sirven de apoyo en la toma de decisiones para obtener información a partir de distintas fuentes de datos.

Una vez analizadas las debilidades y fortalezas de cada una de las tecnologías que hoy en día son utilizadas para la automatización de procesos de negocio y ayuda en la toma de decisiones en la gestión empresarial, se realizará un **análisis competitivo de la oferta tecnológica** en relación a las capacidades de las distintas tecnologías y partiendo de necesidades del mercado.

Actualmente, el mercado demanda tecnologías que sean capaces de automatizar procesos en entornos dinámicos y muy cambiantes. Hasta día de hoy se ha dado soporte para la automatización de procesos estáticos o de gestión de negocio y cada vez más se demandan tecnologías que den soporte para actividad relativa a procesos dinámicos, cambiantes y dependientes del conocimiento de los trabajadores expertos o Knowledge Workers [19].

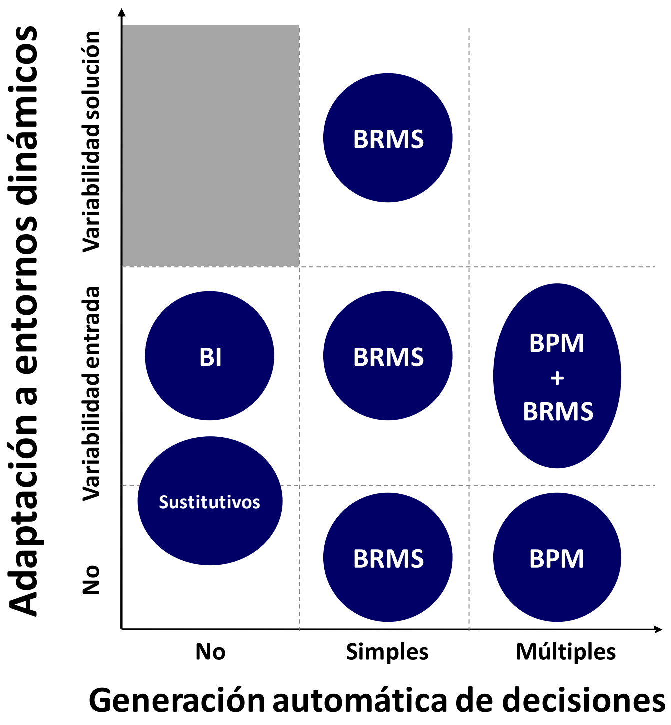
Peter F Drucker, el padre de la gestión empresarial y gurú de tendencias de mercado, auguraba hace unos años la importancia de lo anteriormente comentado: "*Knowledge worker productivity is the biggest of the 21st century management challenges. In the developed countries it is their first survival requirement. In no other way can the developed countries hope to maintain themselves, let alone to maintain their leadership and their standards of living.*"

Para realizar el análisis competitivo de la oferta tecnológica se clasificarán las distintas tecnologías descritas en función de dos variables:

- **La capacidad de generar decisiones de manera automática.** Recordemos que algunas de las tecnologías analizadas sólo son útiles como ayuda para la toma de decisiones por parte del usuario mientras otras son capaces de generar decisiones simples (compuestas por una única acción) o múltiples (compuestas por una serie de acciones).
- **La capacidad de adaptación a entornos dinámicos.** Entendemos por entornos dinámicos en los que se dan alguno de los siguientes factores:
  - En los que el número de variables que condicionan las actividades a realizar o los planes de acción para alcanzar los objetivos son muy elevado.

- En los que las actividades a realizar o los planes de acción para alcanzar los objetivos tienen una gran variabilidad por lo que, dependiendo del estado inicial o del objetivo marcado, los planes necesarios pueden ser muy distintos unos de otros.

A continuación, **se muestra un mapa de posicionamiento competitivo,** basado en las variables de referencia, que nos ayudará a entender la oferta actual en el mercado.



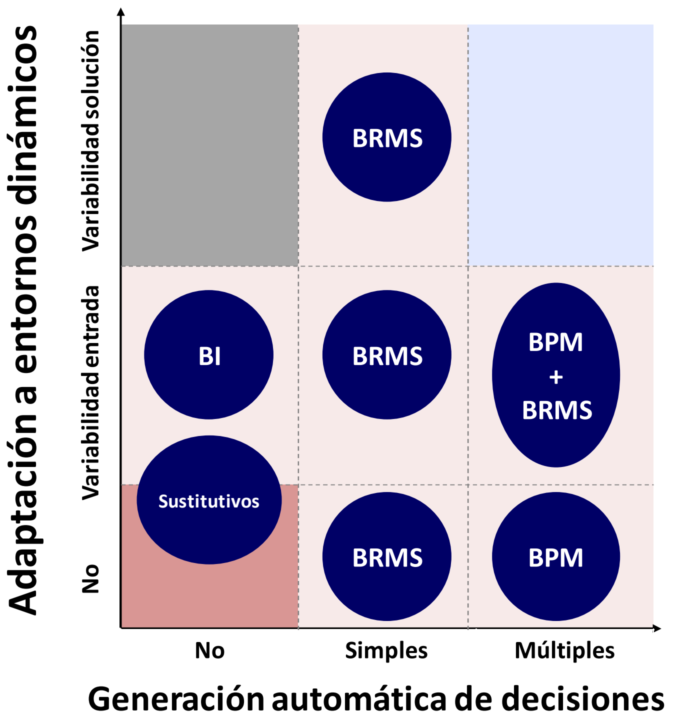
En relación a la primera variable analizada, tanto los sistemas BRMS como los sistemas BPM son capaces de generar automáticamente decisiones, aunque solo los sistemas BPM son capaces de generar decisiones compuestas por múltiples acciones (procesos completos).

Sin embargo, tal y cómo se ha analizado en el análisis de fortalezas y debilidades, los sistemas BPM no se adaptan adecuadamente a entornos dinámicos [20]. Veíamos, y del mismo modo se refleja en el gráfico anterior, que la unión de sistemas BPM junto con tecnología BRMS permite a este tipo de sistemas analizar múltiples variables en sus entornos de ejecución y mejorar sus nodos condicionales en los procesos definidos [21].

Además, este tipo de sistemas siguen teniendo una carencia importante y es que las decisiones tomadas para diseñar el proceso definitivo se realizan en tiempo de ejecución, son sistemas reactivos, por lo que pueden no garantizar la resolución del problema en entornos dinámicos que requieran de una gran variabilidad en las soluciones, tal y cómo se muestra en el gráfico.

Por lo tanto, observamos que los sistemas BPM tan sólo son válidos para automatizar procesos previamente predefinidos y que, por consiguiente, tienen carencias a la hora de ser utilizados en entornos dinámicos.

El aumento y la saturación de la oferta de las tecnologías comentadas para la automatización de procesos de gestión empresarial y ayuda en la toma de decisiones, crean un océano rojo que obliga a la búsqueda de nuevas tecnologías que den soporte a las necesidades de mercado aún no satisfechas.

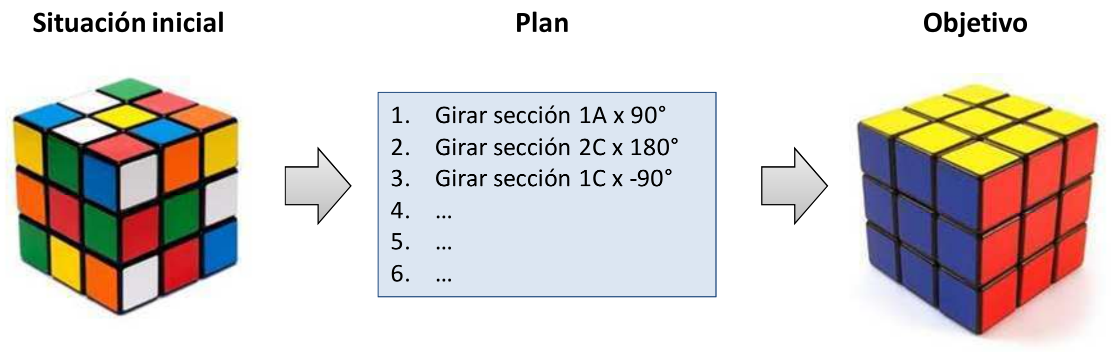


En la siguiente sección se realizará una introducción a las tecnologías de planificación automática como posible oferta para crear un océano azul en el mercado analizado.

1.3 Planificación automática

La planificación automática es el área de la Inteligencia Artificial que trata de construir planes de actuación usando programas de ordenador llamados planificadores automáticos.

Entendemos como plan de actuación el conjunto de acciones necesarias para alcanzar un objetivo a partir de una situación inicial. Por lo tanto, un planificador automático será capaz de generar el plan de actuación adecuado para lograr el objetivo definido.


Situación inicial
Plan
Objetivo
1. Girar sección 1A x 90°
2. Girar sección 2C x 180°
3. Girar sección 1C x -90°
4. …
5. …
6. …

La planificación automática es un problema complejo debido a varias circunstancias:

- **Encontrar un plan es un problema NP-Completo**, o lo que es lo mismo, encontrar una secuencia de acciones se convierte fácilmente, con un número medio de acciones y posibilidades entre las mismas, en un problema combinatorio intratable con las técnicas de búsqueda clásicas.
- **Dificultad para modelar el conocimiento** necesario para la planificación. La planificación clásica se centraba en resolver problemas sencillos, tipo puzzle, con un conjunto pequeño de acciones y objetos diferentes con los que tratar. En problemas reales, para resolver un plan, hay que modelar cientos de acciones y miles de objetos. Y además de la cantidad de conocimiento a representar, la variabilidad del mismo también es un problema de modelado que es necesario abordar para representar todo el conocimiento necesario para poder resolver los problemas planteados [22].
- **La gestión del tiempo**, ya que no es únicamente necesario que las acciones se ejecuten en un determinado orden, sino que se ejecuten en un instante de tiempo determinado según las posibles interrelaciones con otras acciones, sincronizaciones de inicio y fin, etc..
- **Un planificador automático generalmente no es un componente aislado** y cerrado, sino que forma parte de un sistema mucho más complejo, en donde tiene que interactuar con otros actores y con fuentes de datos heterogéneas [23], ya sean humanos o programas software.

Para dar soporte a esta problemática, a día de hoy se han desarrollado planificadores automáticos que han sido aplicados con éxito en entornos reales, como la robótica [24] sistemas de manejo de emergencias (inundaciones [25], operaciones de evacuación y rescate [26][27], incendios forestales [28] [2], sistemas de manufacturación [29], o incluso en misiones espaciales [30] [31] [32].

Este éxito de la planificación automática para resolver problemas reales está haciendo crecer el interés en el mercado de las tecnologías basadas en planificación automática[1]**.** Esto está motivando a los investigadores en el área a construir planificadores más amigables, que resuelvan un mayor número de problemas, y para reducir los costes de adaptación e implantación, de ahí la gran actividad investigadora en el área con la aparición continua de nuevas técnicas y teorías.

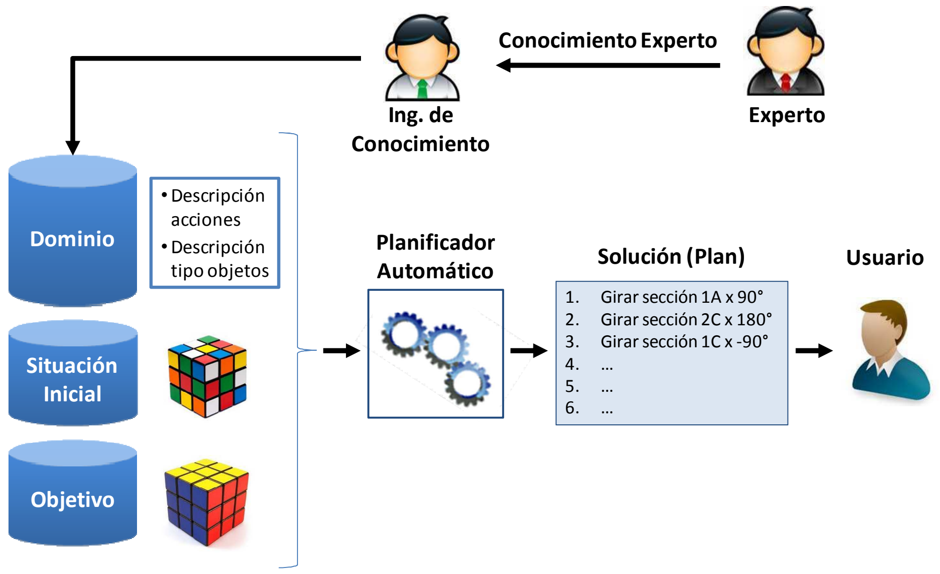
Desarrollar un planificador automático específico para cada una de los posibles dominios de aplicación de esta tecnología es una labor ardua y costosa. Además cuando quisiéramos adaptar el planificador para resolver un tipo de problema distinto o bien cuando las condiciones del dominio para el cual el planificador fue diseñado cambiasen habría que rehacer el planificador parcial o completamente.

Por ello desde los inicios de la planificación se han tratado de desarrollar algoritmos de planificación independientes del dominio de aplicación. Esta separación requiere que, por un lado dispongamos de unos modelos generales que nos permitan almacenar la información del contexto con el que estamos trabajando, por otro lado es necesario almacenar el conocimiento sobre el tipo de acciones que se podrán emplear en el plan buscado y por último es necesario diseñar los algoritmos que operen sobre estas estructuras o lo que es lo mismo, el planificador automático.

A continuación se describen los elementos sobre los que se plantea un problema de planificación a un algoritmo de planificación automática. Un planificador automático deberá ser capaz de comprender el modelo de representación del conocimiento y ser capaz de razonar sobre él, pero sin estar especializado en el dominio en concreto. **Un problema de planificación se formula en base a tres elementos:**

1. **Dominio de planificación:** Una descripción del tipo de objetos existentes en el entorno y del conjunto de acciones que podemos aplicar sobre ellos. Los objetos del entorno tendrán una serie de propiedades y las acciones a aplicar tendrán una serie de efectos sobre los objetos y de precondiciones que deberán satisfacerse para poder aplicar las acciones.
2. **Estado Inicial:** Una representación del estado inicial en el que se encontrarán los objetos del entorno antes de que empecemos la ejecución de nuestro plan.
3. **Objetivo:** Una representación de la meta o del objetivo que deseamos cumplir tras la ejecución de nuestro plan.

[1] www.siri.com, www.iactivecompany.com



Un planificador automático genera el plan para alcanzar el objetivo, partiendo del estado inicial, utilizando las acciones y objetos definidos en el dominio. Estas representaciones deben tener una sintaxis y una semántica bien definida además de ser lo suficientemente expresivas para poder resolver una gran cantidad de problemas en diversos dominios de aplicación. Existe un lenguaje considerado estándar en el ámbito de la planificación automática, llamado PDDL (Planning Domain Description Language) [33]

La investigación en el área de planificación automática, no se reduce exclusivamente a desarrollar teorías y algoritmos de planificación más eficientes y expresivos, sino que también en realizar labores de Ingeniería del Conocimiento con el fin de modelar, adquirir y representar el conocimiento del dominio. Además, es también objeto de estudio del área de IC para Planificación, el desarrollo de herramientas que permitan la comprensión del plan resultante y facilitar la interacción y comunicación con el mismo por parte de los usuarios. Como se puede apreciar en la ilustración anterior, hay varios actores involucrados en el proceso de planificación:

- **Expertos**: Son las personas que tienen el conocimiento del entorno y las medidas a tomar para alcanzar los objetivos. A partir de su experiencia se extrae la información y conocimiento necesarios para construir el dominio de planificación.
- **Ingenieros del Conocimiento**: Son personas con un fuerte conocimiento en planificación automática y, en particular, en el modelado de dominios que realizan partiendo del conocimiento del Experto, mediante un proceso puro de ingeniería del conocimiento.
- **Usuarios finales**. Son las personas utilizan el planificador automático para resolver un problema. En muchas ocasiones los usuarios son los propios expertos que utilizan planificadores automáticos para obtener soluciones a problemas que ellos mismos podrían resolver, de un modo más rápido y seguro.

1.4 Un nuevo Océano Azul: Los planificadores automáticos

Al igual que se desarrolló para el resto de tecnologías estudiadas, a continuación se muestra un análisis de fortalezas y debilidades de los Planificadores Automáticos para la automatización de procesos de negocio y ayuda a la toma de decisiones.

Planificadores Automáticos	
Fortalezas	**Debilidades**
• Reducen costes ya que "extraen" la lógica del software del código interno y para cambiar el comportamiento de los sistemas y no es necesario reprogramar las aplicaciones • Generación de planes adaptados a entornos dinámicos con gran número de variables de entrada. • Generación de planes muy dinámicos adaptados a entornos que requieran gran variabilidad en las decisiones tomadas. • Crea el proceso en su conjunto centrado en el objetivo y no ejecuta paso a paso como un BPM. Los planificadores automáticos son capaces de generar propuestas de proceso (planes) antes de comenzar su ejecución y conocer, de antemano, si se cumplirá el objetivo definido. • Dan soporte a las características de razonamiento de los Knowledge worker que no cubren los BPM [19]: Generar procesos fiables, no repetitivos, que manejen incertidumbre y cambiantes durante su ejecución. • Son capaces de ofrecer planes con información temporal, en cuanto a la duración de las actividades, tiempos de inicio y fin.	• La escritura de los dominios de planificación requiere fuertes conocimientos en este tipo de tecnologías y de los lenguajes de escritura de dominios específicos como PDDL. • Optimización de recursos. • Incorporación de información en tiempo de ejecución del plan. • Coste computacional más elevado.

Tras el análisis realizado se observa que, gracias a la capacidad con la que cuentan los planificadores automáticos de generar de modo dinámico un plan o proceso para alcanzar un objetivo determinado dependiendo de las variables del entorno (situación inicial) [34], **los planificadores automáticos encuentran un nicho de mercado todavía sin explotar para dar soporte a la automatización de procesos en entornos altamente dinámicos.**

Por lo tanto, los planificadores automáticos representan el ingrediente necesario para crear un nuevo *Océano Azul* en el mercado, basado en un proceso de innovación que aportará una ventaja competitiva para la automatización de procesos.

No obstante, la oferta al mercado de este tipo de tecnologías requerirá de herramientas comerciales que permitan y faciliten su uso por compañías privadas. En la siguiente sección se enumerarán los objetivos planteados en esta tesis en este sentido.

1.5 Objetivos de la tesis

Durante los años de desarrollo de esta tesis, el doctorando ha realizado distintas aportaciones al área de la planificación automática mediante el desarrollo de un planificador jerárquico propio dotado de capacidades de gestión del tiempo [4], así como en su integración en arquitecturas software basadas en servicios web [35] y en su aplicación a problemas reales [36] [3].

No obstante, según lo visto en las secciones anteriores, para hacer llegar todos estos avances al mercado se requieren aportaciones adicionales. Para adaptar un planificador automático a un nuevo sector o problema determinado, es necesario modelar el dominio partiendo del conocimiento de un experto que puede provenir de diversas fuentes (como documentación en modo texto o la propia mente del experto) [3] [37].

Sin embargo, un importante inconveniente, que dificulta la adopción y uso extendido de las tecnologías de planificación inteligente, es que hoy en día sólo un reducido número de personas, normalmente investigadores en el área de planificación automática, tienen los conocimientos necesarios para modelar dominios orientados a resolver problemas de planificación automática en distintas áreas de aplicación.

El objetivo de esta tesis es definir una notación gráfica para representar dominios de planificación jerárquica, dado que este tipo de planificadores como se verá en la memoria tienen una estructura para representar el conocimiento más intuitiva, que permita no sólo a investigadores del área sino también a otros profesionales del sector de la tecnología modelar dominios de planificación, de un modo sencillo y de forma parecida a otras notaciones gráficas existentes en el mercado [7].

Para abordar el objetivo planteado, el trabajo se centrará en el área Knowledge Engineering for planning (KEP) que, además de centrarse en los aspectos de modelado, adquisición y representación de conocimiento, también se centra en el diseño y desarrollo de las herramientas necesarias para la integración y despliegue de planificadores automáticos, cubriendo el ciclo completo de vida de un proyecto de ingeniería del software: modelado, adquisición, representación, integración, despliegue y validación.

Gracias al trabajo planteado se romperán las barreras actuales que impiden la adopción y uso extendido de las técnicas de Planificación automática, permitiendo el uso de este tipo de tecnologías en compañías privadas, aumentando considerablemente el número de usuarios y **potenciando el uso de planificadores automáticos en el mercado** ya que el número de personas capaces de modelar dominios de planificación aumentará considerablemente. Además, la notación gráfica definida facilitará el modelado de los dominios a aquellas personas que actualmente tienen conocimientos sobre el modelado de dominios de planificación.

1.6 Estructura de la tesis

La tesis está dividida en tres bloques, siendo el primero de ellos el principal en el que se desarrolla el trabajo, el siguiente consiste en una recopilación de los artículos publicados por el doctorando que soportan esta tesis y el último las referencias bibliográficas.

En el primero de los bloques se sitúa el problema abordado y la solución al mismo, dentro de este bloque encontramos cuatro secciones. En la primera sección, dónde nos encontramos ahora, se ha realizado un análisis del mercado y se han identificado nuevas necesidades que serán abordadas por esta tesis, se ha realizado una introducción a la planificación automática y se ha definido el objetivo de la tesis dentro del mercado analizado. En la segunda sección se describirá la planificación jerárquica como un tipo especial de planificación y los lenguajes actuales para la representación de dominios para este tipo de planificadores. En la tercera sección se estudiarán las distintas alternativas que han sido planteadas para alcanzar el objetivo de la tesis. Por último, en la cuarta sección se describirá la solución y los avances aportados consistentes en la notación gráfica para la representación de dominios de planificación jerárquica y un producto comercial denominado *IActive Knowledge Studio* basado en ella.

2. Problemática abordada

2.1 Introducción a los planificadores de redes de tareas jerárquicas (HTN)

La planificación de redes de tareas jerárquicas o HTN (Hierarchical Task Networks) tiene su origen en la necesidad de acercar el modelo de representación de acciones a la representación con distintos niveles de abstracción que el ingeniero del conocimiento utiliza a la hora de modelar un dominio. De hecho está demostrado que la expresividad del modelo HTN es mayor que la del modelo de STRIPS [36].

Aunque los orígenes de la planificación jerárquica son antiguos [38][39], incluso ya existiendo algunas aplicaciones notables [40], como por otro lado ocurre habitualmente en el área de planificación, las bases teóricas no fueron establecidas hasta algún tiempo después [41]. Aún así todavía hoy en día no hay un modelo común y unificado, aunque todos comparten una serie de características muy similares.

Cada planificador HTN tiene sus propios algoritmos de búsqueda y su propio lenguaje para la descripción de dominios. Incluso en la IPC[2] no hay todavía una sección exclusiva dedicada a planificadores HTN, sólo se realizó en una ocasión en la AIPS 2002, durante la 3ª IPC. Ha habido, no obstante, esfuerzos e intentos notables para desarrollar un lenguaje de representación de dominios unificado [42], sin que hayan tenido éxito debido principalmente a esta heterogeneidad y complejidad en los sistemas.

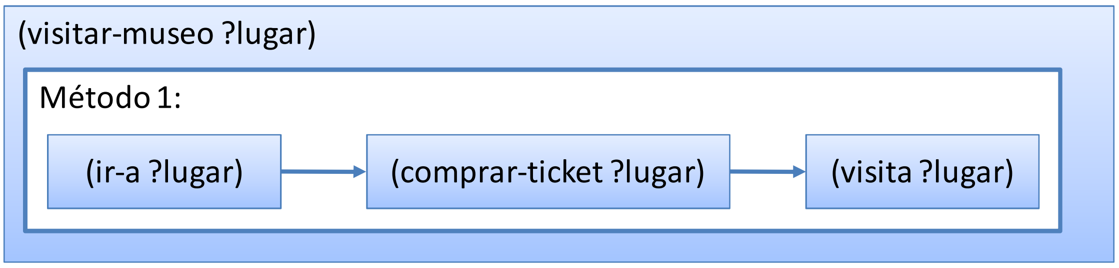
Otra razón por la que los planificadores HTN han tenido tanto éxito es debido a su eficiencia al enfrentarse a problemas reales. La expresividad del HTN y la forma en que este modelo realiza la búsqueda, permiten que el diseñador del dominio tenga un mayor control sobre el proceso de búsqueda, pudiendo este optimizar el dominio para evitar realizar búsqueda innecesaria, incluyendo en el dominio el conocimiento necesario para guiar correctamente el proceso de búsqueda, evitando zonas del espacio de búsqueda que estén prohibidas y forzando a pasar por otras zonas según el conocimiento que se dispone sobre el problema.

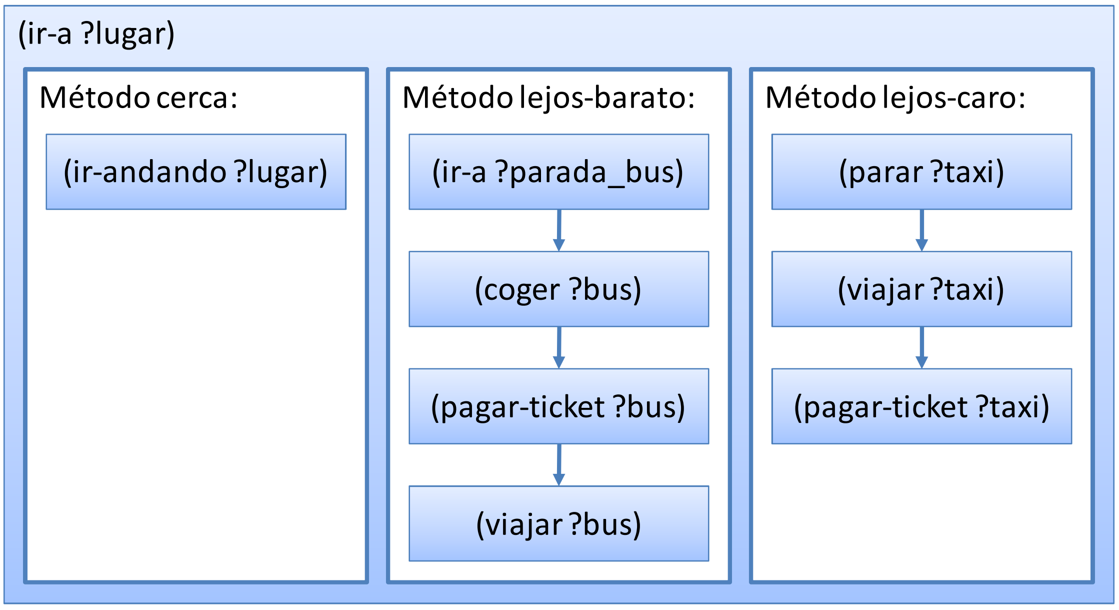
El modelo de HTN que se explica está basado en el modelo de UMCP (Universal Method Composition Planner) [41] y SHOP2 [43]. **En este modelo encontramos los siguientes elementos:**

- **Acciones primitivas u operadores primitivos, base u hoja:** Los operadores primitivos son muy similares a las acciones usadas en el modelo de STRIPS. Representan acciones que finalmente son ejecutadas y producen cambios en el estado del mundo (representado internamente en el planificador) y no se pueden descomponer.

[2] International Planning Competition http://ipc.icaps-conference.org/

- **Tareas o acciones compuestas o abstractas:** Una tarea compuesta representa una actividad que puede ser llevada a cabo de varias formas alternativas. La manera de representar cada una de estas alternativas está basada en esquemas de descomposición. Un esquema de descomposición consiste básicamente en un grafo dirigido acíclico cuyos nodos representan a su vez tareas abstractas, operadores primitivos, o bien una mezcla de ambos y cuyos arcos representan relaciones de orden
- **Métodos: Una tarea abstracta puede tene**r varios esquemas de reducción distintos, que definen la forma en que ésta se sustituye durante el proceso de planificación. Para decidir que esquema de reducción es aplicable en cada momento o es el más adecuado se introduce el concepto de método, que establece una serie de condiciones previas a la aplicación de la reducción.





Esta forma de organizar el conocimiento procedural en base a tareas de más alto y más bajo nivel, es mucho más cercana a la forma de resolver problemas de los seres humanos, que solemos marcar una serie de tareas generales que es necesario realizar y las vamos refinando conforme vamos definiendo el plan.

Como se ha señalado, el modelo de planificación HTN es similar al modelo de STRIPS [44] en algunos aspectos. Primero ambos tienen una representación común del espacio de estados. Por otro lado la mayoría de los planificadores HTN también presuponen un conocimiento completo del entorno. Y finalmente la representación de los operadores primitivos es similar, con unas precondiciones, y unos efectos, que en STRIPS están divididos en una lista de adición y una lista de supresión. La principal diferencia radica en qué se considera un objetivo y cómo se consigue. Para STRIPS un objetivo es alcanzar un estado meta, en HTN el objetivo es aplicar los esquemas de descomposición sobre todas las tareas abstractas de la red de tareas de partida, hasta tener una red de tareas válida, únicamente compuesta de operadores primitivos, que será finalmente el plan resultante.

Supongamos el ejemplo de la figura anterior en el que se muestra una tarea abstracta para visitar un museo. Esta tarea tiene un único método que establece cual es la red por la que sustituimos la tarea abstracta, que está compuesta de tres tareas que se realizan en secuencia. Primero debemos desplazarnos al lugar de visita, después hay que adquirir un ticket de entrada y por último realizamos la visita. En el ejemplo también se muestra la estructura de la tarea abstracta ir-a, que dispone de tres posibles métodos de descomposición. Cada uno de estos métodos puede llevar una serie de precondiciones asociadas para seleccionar en cada momento cual es la descomposición más adecuada. Por ejemplo si el lugar de destino está a una distancia menor de un umbral predeterminado se puede ir andando, en otro caso se puede coger un taxi o un autobús. Obsérvese que para coger el autobús previamente hay que desplazarse hasta la parada, para lo cual podemos aplicar el mismo método ir-a. Cualquiera del resto de tareas que no se muestran en el ejemplo como pagar-ticket o viajar, pueden ser otras tareas abstractas u operadores primitivos. El planificador va construyendo mediante la sustitución de tareas abstractas por su correspondiente red de tareas, un árbol de expansión de tareas que constituye su espacio de búsqueda, distinto del espacio de búsqueda de los planificadores basados en estados que es un grafo de estados con transiciones de uno a otro estado, y también del espacio de búsqueda de planes de UCPOP ya que las tareas son introducidas en el plan candidato siguiendo una filosofía distinta, cuyo espacio de búsqueda es un espacio de planes en el que cada acción introduce un refinamiento sucesivo del mismo. El árbol de expansión de tareas puede verse como un árbol Y/O donde los nodos O son las distintas métodos alternativos que tenemos para expandir una tarea compuesta, y los nodos Y serían las distintas tareas codificadas en la red de tareas que es necesario expandir.

Las tareas abstractas permiten al diseñador de dominios escribir el dominio con diferentes y múltiples niveles de abstracción, pudiendo utilizar un enfoque top-down en donde partimos de tareas muy abstractas y las vamos especificando cada vez más en tareas de más bajo nivel o bien un enfoque bottom-up en donde partimos de los operadores primitivos y vamos construyendo las tareas abstractas que lo asocian.

Así pues se puede describir un algoritmo de descomposición de tareas HTN muy básico (hacia adelante y con estados) de la siguiente forma [41]:

- Sea ε_o el estado inicial.
- D: el dominio de planificación.
- T_o: la red de tareas inicial que deseamos descomponer.
- π: el plan en construcción que es una lista de tareas primitivas, inicialmente vacío.
- α_i la agenda de tareas pendientes que inicialmente se construye de la siguiente forma: $\alpha_0 = U_{i:(1..n)}\ t_i \in T_0$, y que se utiliza en la llamada al primer ciclo HTN.

```
HTN (<$\varepsilon_i$, $\Gamma_i$, $\pi_i$, $\alpha_i$>, $\mathcal{D}$)
1.  if empty($\alpha_i$) then: return $\pi_i$
2.  while escoger no determinísticamente una tarea: $\rho \in \alpha_i$
3.      if primitive($\rho$) then:
4.          if preconditions($\rho$, $\varepsilon_i$) == true:
5.              $\varepsilon_{i+1}$ = $\varepsilon_i$ + add_list($\rho$) - delete_list($\rho$)
6.              $\pi_{i+1}$ = $\pi_i$ + $\rho$
7.              return HTN (<$\varepsilon_{i+1}$, $\Gamma_i$, $\pi_{i+1}$, $\alpha_i$>, $D$)
8.          else: backtrack
9.      elif compound($\rho$) then:
10.         Escoger un método $\mu$ válido o backtrack
11.         $\Gamma_{i+1}$ = $\Gamma_i$ - $\rho$ + expand($\rho$, $\mu$)
12.         $\alpha_{i+1}$= update($\alpha_i$ , $\Gamma_{i+1}$)
13.         return HTN(<$\varepsilon_i$, $\Gamma_{i+1}$, $\pi_i$, $\alpha_{i+1}$>, $\mathcal{D}$)
14. return FAIL
```

Cuando la expansión de una tarea abstracta falla, porque alguna de las acciones que componen la red de tareas en la que se descompone no se puede aplicar, se produce un backtracking o vuelta atrás, en la cual el algoritmo de planificación debe explorar otras alternativas (otros métodos u otra forma de unificar las variables). El mecanismo de bactracking es el que permite al planificador adaptar el plan al estado del entorno, pero al mismo tiempo es lo que consume más tiempo y recursos durante la planificación. Todos los planificadores jerárquicos y no jerárquicos tratan de encontrar formas de evitar el backtracking, pero quizá en los planificadores jerárquicos es más sencillo [45].

2.2 Representación de dominios en planificadores HTN

Un aspecto muy importante para cualquier modelo de planificación es definir cómo el algoritmo es informado de la situación en la que se encuentra su entorno y del problema que tiene que resolver. La mayoría de los algoritmos de planificación son independientes del dominio, es decir, son genéricos y en principio podrían utilizarse en cualquier dominio planificación, por lo tanto **se hace necesario disponer de un lenguaje de representación de dominios que el planificador pueda procesar**.

Existe un lenguaje considerado estándar en el ámbito de la planificación automática, llamado PDDL (Planning Domain Description Language) [33]. PDDL nace en 1998 como lenguaje para describir los problemas de la IPC de ese mismo año. Tiene en su haber dos grandes ventajas que lo han hecho tan usado. Primera, es el lenguaje usado en la IPC, cualquier planificador que quiera participar en esta competición, que por otro lado es bastante prestigiosa, debe ser capaz de procesarlo. Segunda más importante, al ser muchos los autores que colaboraron en su especificación, supo unificar las características principales de los lenguajes de los planificadores más importantes en la época. Su base fue el lenguaje ADL [46], pero también recogió formalismos de plataformas de planificación como PRODIGY [47] o SIPE-2 [40] [48], planificadores de orden parcial como UCPOP [49], Unpop [50], o jerárquicos como UMCP [41], aunque el modelo jerárquico propuesto en PDDL no ha sido muy aceptado como discutiremos más adelante.

PDDL es un lenguaje muy expresivo, de hecho hay muchos planificadores que no pueden soportarlo plenamente, por ello incorpora un mecanismo de requerimientos sobre el planificador, como una serie de flags, que el planificador debe soportar para operar sobre el dominio. En la versión 1.0 de PDDL [33] su expresividad era muy similar a la de ADL con la incorporación de la posibilidad de definir dominios jerárquicos. Afortunadamente PDDL es un lenguaje vivo que va evolucionando con cada edición de la IPC. En la edición del 2002 se presentó una nueva versión del lenguaje la 2.1 [42] que incorporaba importantes novedades. PDDL 2.1 se organizaba en 4 niveles con expresividad creciente. En el nivel 1 tenía la misma expresividad que PDDL 1.0. En el nivel 2 se incluye una mayor expresividad para el tratamiento de expresiones numéricas o fluents. En el nivel 3 se introduce temporización en las acciones, con precondiciones y efectos at-start, at-end o overall. El nivel 4 permite la definición de acciones con efectos continuos en el tiempo. La última versión oficial de PDDL es la 3.0 [51] presentada en la IPC del 2006, esta versión recoge la posibilidad de introducir preferencias de usuario y restricciones en el dominio de planificación.

Existen otras extensiones de PDDL no presentadas en la IPC. Por ejemplo hay otras propuestas para dar soporte jerárquico a HTN [52], o la propuesta de un quinto nivel [53] para modelar procesos continuos que van cambiando con el paso tiempo los valores numéricos.

2.3 Problemática de la representación de dominios mediante PPDL

Para adaptar un planificador automático a un nuevo sector o problema determinado, es necesario modelar el dominio partiendo del conocimiento de un experto o de la documentación adecuada.

Para ello, es necesario utilizar lenguajes textuales de representación de dominios como PDDL que requieren de un conocimiento específico muy elevado.

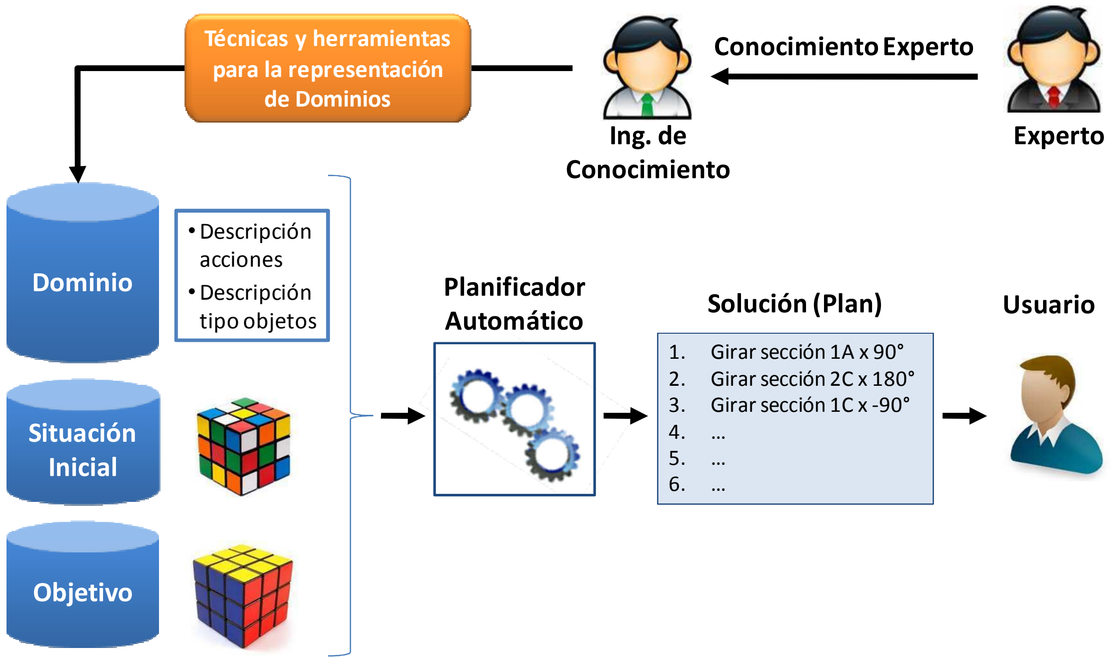
Normalmente sólo los investigadores en el área de planificación automática con fuertes conocimientos en ella, conocen PPDL para modelar dominios de planificación.

Una de las causas por las que PDDL no es un lenguaje extendido en el mercado es porque, como se verá en las soluciones aportadas, no cuentan con una estructura ni nomenclatura que siga estándares ni prácticas habituales de mercado.

Por lo tanto, si se pretende aumentar el uso de planificadores automáticos en el mercado, será necesario desarrollar lenguajes de modelado comerciales y herramientas que faciliten la representación de dominios de planificación a programadores e ingenieros informáticos que no sean necesariamente expertos en Inteligencia Artificial y planificación automática.

3. Alternativas de Solución

Tal y cómo veíamos en la sección anterior y como se observa en la siguiente figura, para adaptar un planificador automático a un nuevo sector o problema determinado, es necesario modelar el dominio partiendo del conocimiento de un experto o de la documentación adecuada.


Técnicas y herramientas para la representación de Dominios
Conocimiento Experto
Ing. de Conocimiento
Experto
Dominio
• Descripción acciones
• Descripción tipo objetos
Situación Inicial
Objetivo
Planificador Automático
Solución (Plan)
1. Girar sección 1A x 90°
2. Girar sección 2C x 180°
3. Girar sección 1C x -90°
4. …
5. …
6. …
Usuario

Para realizar esta labor, se hacen necesarias herramientas, más o menos sofisticadas, basadas en técnicas de representación de la información y el conocimiento que permitan al Ingeniero de Conocimiento representar dominios interpretables por el algoritmo de planificación.

Partiendo de los avances en Lenguajes de Programación y en Ingeniería del Conocimiento, podemos plantear distintos enfoques para satisfacer la necesidad identificada:

a. **Utilizar lenguajes textuales de descripción de dominios.**
La comunidad científica en planificación automática ha utilizado desde los años 60 lenguajes de descripción de dominios. El primero de ellos fue el de STRIPS [44] . Una de las posteriores evoluciones con mayor aceptación, que ya ha sido mencionada en esta memoria, fue el lenguaje PDDL [33] que ha sido continuamente evolucionado y adaptado para soportar los avances realizados por los algoritmos de planificación automática [42][53]. De hecho, una de las aportaciones del trabajo previo a esta tesis fue la evolución del lenguaje PDDL para soportar la escritura de dominios jerárquicos HTN-PDDL [54]. Esta aportación fue un avance muy importante, como se verá más adelante, para permitir la escritura de dominios de planificación jerárquica siguiendo lenguajes estándares y para el posterior desarrollo de una notación gráfica.

b. **Utilizar notación gráfica para describir dominios.**
Durante los años 90 se empieza a trabajar en ontologías para representar planes y facilitar el trabajo entre distintos grupos de investigación en el área [55] [56]. A final de la década surgen importantes proyectos que intentan consolidar los trabajos previos como es el caso de SPAR (Shared Planning and Activity Representation) [57].

Recientemente, se han desarrollado entornos para soportar parte de las fases del ciclo del vida de Ingeniería del conocimiento, como GIPOII [58], orientado al modelado, adquisición y representación del conocimiento basado en transciones de objetos, o itSIMPLE[59], destinado a facilitar la ingeniería de dominios PDDL no jerárquicos, utilizando conceptos estándar de ingeniería del software como: modelado de datos basado en UML y de acciones usando diagramas de transición UML.

Sin embargo, ninguna de estas iniciativas consigue desarrollar un estándar de uso comercial con aceptación en el mercado. No obstante, sí que han conseguido prosperar algunas iniciativas de otras áreas relacionadas con la planificación y procesos, para representación gráfica de sus modelos de información y conocimiento, como es el caso de BPMN (Business Process Model and Notation) [15]. Estas iniciativas están avaladas, al igual que otras ampliamente conocidas para la representación de modelos de datos como UML (Unified Modeling Language) [60], por OMG (Object Management Group) [61].

Antes de apostar por un planteamiento concreto para dar solución al problema abordado, es necesario que realicemos un análisis de ventajas e inconvenientes de cada una de los enfoques estudiados.

a. **Utilizar lenguajes textuales de descripción de dominios.**
<u>Ventajas</u>

- Este tipo de lenguajes cuenta con una gran expresividad, normalmente basada en lógica de predicados.
- Existen estándares, como PDDL, ampliamente aceptados por la comunidad científica.

<u>Inconvenientes</u>

- Requiere de un profundo conocimiento en planificación automática y en lógica de predicados de primer orden para la representación de dominios basados en él.
- Dificulta la comprensión de la estructura de los dominios de planificación al no tener ningún apoyo visual de representación gráfica.
- Como se verá más adelante, la estructura en la que se representa la información y el conocimiento de los dominios de planificación en el lenguaje PDDL presenta problemas en la integración de los planificadores automáticos con otros sistemas software.

- Para representar dominios de planificadores jerárquicos no sería valido un estándar de planificación como PDDL, sino que habría que utilizar alguna variante como la HTN-PDDL mencionada anteriormente.

b. **Utilizar notación gráfica para describir dominios.**

Ventajas

- Facilita, mediante la representación gráfica, la comprensión visual de los dominios de planificación y ayuda al Ingeniero de Conocimiento a entender el paradigma para resolver los problemas que utilizan los planificadores automáticos.
- Sigue las tendencias del mercado en cuanto a herramientas comerciales para el uso de nuevas tecnologías, como es el caso de BPM, que están basadas en notaciones gráficas.
- Existen estándares, como BPMN para la representación de procesos y UML para la representación de bases de datos, ampliamente aceptados por el mercado en los que pueden basarse este tipo de lenguajes.

Inconvenientes

- Complejidad de alcanzar el mismo nivel de expresividad para la definición de dominios de planificación que mediante un lenguaje de presentación de dominios.

4. Solución adoptada

Tal y cómo se ha descrito en la memoria hasta el momento, la presente tesis plantea el objetivo de llevar al mercado los planificadores jerárquicos para cubrir el nicho detectado y, para ello, es preciso dar una respuesta a la barrera existente a la hora de facilitar la representación de dominios de planificación al personal técnico de las compañías de software.

No obstante, los primeros pasos para cubrir esta necesidad de mercado no han estado centrados en facilitar la representación de dominios, sino que **previamente se han tenido que desarrollar avances sobre el área de la planificación automática** que realmente den cobertura a las necesidades de mercado detectadas.

En este sentido, el trabajo de investigación comenzó con el desarrollo de un planificador jerárquico [1] que fue fundamental para el resto de desarrollos que han soportado esta tesis.

Las primeras aplicaciones reales del planificador, que validaron la tecnología desarrollada, fue un sistema de ayuda a la toma de decisiones en extinción de incendios forestales, denominado SIADEX [2] en el que se tuvieron unas primeras experiencias sobre la representación del conocimiento experto para la resolución de problemas reales [3].

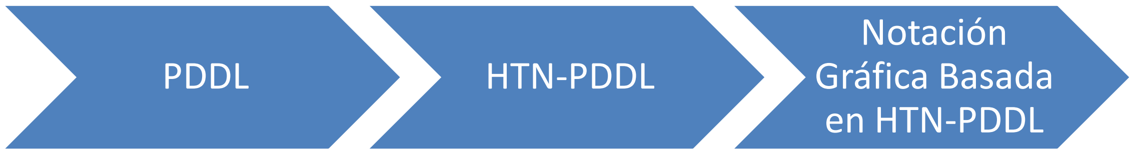
Tal y cómo se ha analizado en el estudio de mercado, la capacidad de representación y razonamiento temporal es fundamental para dar soporte a dominios de aplicación real. En este sentido, también fue necesario dotar al planificador de capacidad de razonamiento temporal difuso de manera que permitiese la monitorización de la ejecución del plan teniendo en cuenta retrasos y posibles desfases en la ejecución de las tareas del mismo [4].

Previamente al desarrollo de las herramientas que faciliten el uso de planificadores en entornos no-académicos, también se hizo un especial esfuerzo en la integración del planificador en una arquitectura compuesta por servicios web [5], ya que son las arquitecturas que hoy en día están más extendidas en el entorno comercial, de este modo se facilitó la conexión del planificador en distintos entornos [6] y permitir su aplicación a distintos problemas.

A partir de los avances y el planificador desarrollado, que fueron fundamentales como los primeros pasos en el desarrollo de esta tesis, **se realizó el análisis de alternativas** visto en la sección anterior, para el desarrollo de una herramienta que permitiera a Ingenieros Informáticos que no tuvieran experiencia previa en el área científica de Planificación Automática, desarrollar sistemas basados en el planificador automático desarrollado previamente.

También se analizaron lenguajes existentes de representación del conocimiento para planificación [62] [63] que no han sido adoptados por el mercado, tal y cómo es el objetivo de este trabajo, debido a que no estaban basados en estándares comerciales.

Atendiendo al objetivo marcado de facilitar el uso comercial de planificadores jerárquicos, y tras el análisis desarrollado, se pone de manifiesto la necesidad de desarrollar una notación gráfica para la representación de dominios basada en un lenguaje textual para la escritura de dominios de planificadores jerárquicos, como es el caso de HTN-PDDL [54].



No obstante, **la solución adoptada no estará basada exclusivamente en el desarrollo de una notación gráfica** para el desarrollo de los dominios de planificación, sino que será necesario un planteamiento de solución más sofisticada y completa para facilitar su uso comercial [7].

La solución planteada está basada en los siguientes elementos:

- **Definición de una notación gráfica**, basada en la notación de representación de procesos BPMN ampliamente conocida por el mercado, de los siguientes elementos de los dominios de planificación jerárquica HTN-PDDL:
  - Acciones primitivas.
  - Tareas compuestas.
  - Métodos.
  - Objetivos.

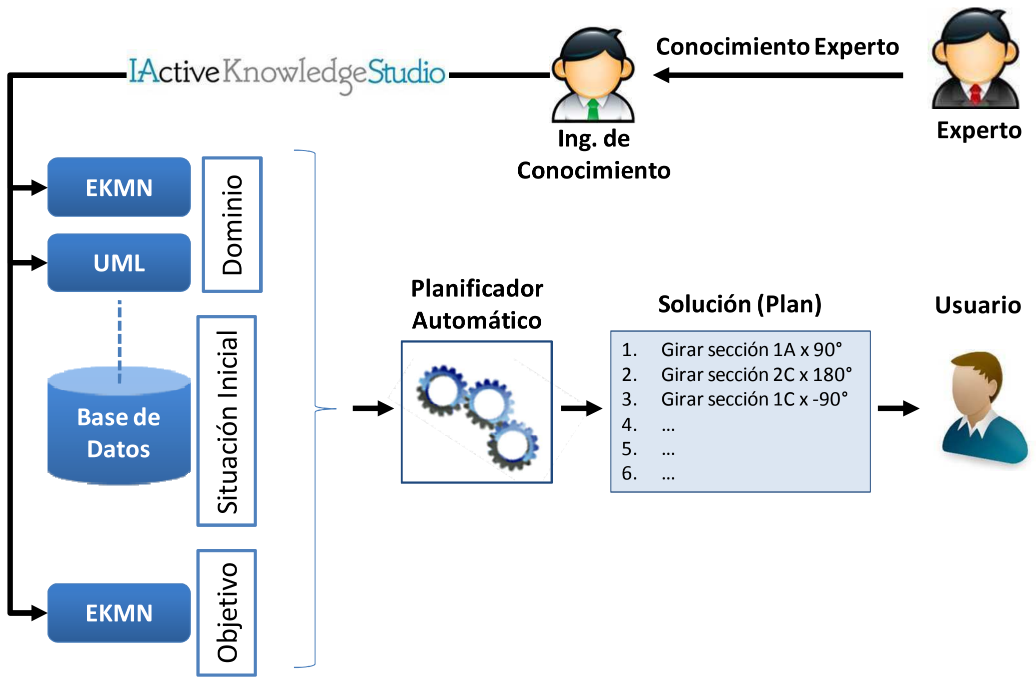
  A esta notación gráfica, que será descrita en detalle más adelante, se le ha denominado **EKMN** (Expert Knowledge Model and Notation).

- **Utilización de la notación gráfica UML ya existente** y ampliamente utilizada en el mercado, para la representación gráfica de los siguientes elementos de los dominios:
  - Representación de los tipos y modelos de datos utilizados por los elementos anteriores: Acciones primitivas, Tareas compuestas, Métodos y Objetivos.
  - Representación de los objetos, pertenecientes a tipos definidos, utilizados en la situación inicial del algoritmo de planificación.

  Gracias al uso de UML para representar los objetos utilizados en la situación inicial del algoritmo de planificación, facilitamos al planificador automático la conexión e integración con Bases de Datos externas de las que obtener dicha información.

- **Utilización de un lenguaje de lógica de predicados que complementa la carencia de expresividad de las notaciones gráficas**, , continuando la tradición de PDDL y HTN-PDDL, basado en los lenguajes de programación de objetos como JAVA para hacerlo más familiar al Ingeniero de Conocimiento, que permitirá la descripción de los siguientes elementos:
    - ➢ Precondiciones, condiciones e información sobre los elementos temporales de las Acciones primitivas, Tareas Compuestas y de los Métodos.

A continuación se ilustran los distintos elementos de la solución planteada:



En la siguiente sección, se describe con detalle la notación gráfica definida EKMN para la representación de dominios de planificación jerárquica.

4.1 Descripción de la notación gráfica planteada

En esta sección se describe la notación gráfica planteada, que ha sido denominada **EKMN (Expert Knowledge Model and Notation)** [7], basada en la notación gráfica de procesos de negocio BPMN (Business Process Management and Notation) y desarrollada a partir del lenguaje textual de representación de dominios jerárquicos HTN-PDDL. Además, para aumentar la expresividad de la notación definida, se ha complementado con un lenguaje de lógica de predicados denominado **EKL (Expert Knowledge Language)** con el que se describirán los atributos de cada elemento del dominio de planificación jerárquica.

Antes de describir en detalle EKMN, merece la pena aclarar la estructura con la que ha sido definido, ya que es distinta a la estructura con la que se plantea un dominio en HTN-PDDL, para facilitar, como se verá más adelante, la integración del planificador con otros sistemas,.

EKMN distingue los siguientes elementos como inputs al planificador:

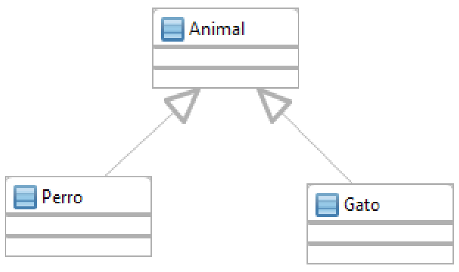
1) **Modelo de Conceptos**: Serán las estructuras y los tipos de objetos que serán tratados por el planificador durante la búsqueda de una solución a un problema dado. Si, por ejemplo, el problema tratado es sobre planificación en extinción de incendios, los conceptos modelados serán, por ejemplo, "Vehículo aéreo", "Vehículo terrestre", "Incendio", etc.
2) **Conocimiento Experto**: Serán las posibles tareas a realizar así como su relación entre ellas a través de métodos. En un ejemplo de planificación en extinción de incendios, en este apartado encontraremos tareas como "Atacar Fuego", "Desplazar vehículos" con sus correspondientes condiciones y efectos y relaciones entre ellas de orden, etc.
3) **Datos del Problema**:
    a. **Incoming Data**: Objetos concretos que, siguiendo una estructura según el modelo de conceptos definido, tome como situación inicial el planificador automático para alcanzar el objetivo determinado. Estos datos son leídos desde una base de datos externa con la que se integrará el planificador y que deberá seguir la estructura definida en el modelo de conceptos.
    b. **Local Data**: Objetos concretos que, siguiendo una estructura según el modelo de conceptos definido, tome como situación inicial el planificador automático para alcanzar el objetivo determinado. Estos datos son definidos durante la escritura del dominio de planificación. Estos objetos, en PDDL, corresponden a constantes definidas en el propio dominio y en EKMN se separan de la escritura del dominio o, lo que es lo mismo, del conocimiento experto para incluirla en un elemento correspondiente a información inicial de entrada.
    c. **Incoming Goal**: Será el objetivo que, siguiendo la definición dada por el conocimiento experto, tome como objetivo el planificador.

Una vez descrita la estructura con la que serán definidos los dominios de planificación en EKMN, a continuación se describe su correspondencia con los elementos de un dominio HTN-PDDL cuya sintaxis es la siguiente:

```
<domain>:
                                        (define (domain <symbol>)
                                                      [<types-def>]
                                                  [<constants-def>]
                                                 [<predicates-def>]
                                                  [<functions-def>]
                                                [<structure-def>*])
```

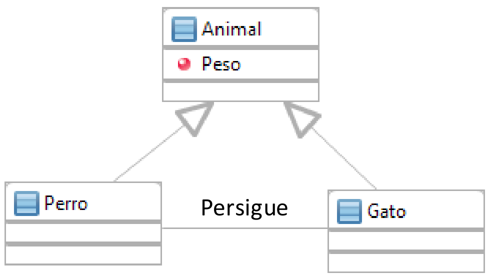
- La declaración de tipos en EKMN se realiza con notación UML en el Modelo de Conceptos:

```
(:types
        Animal – object
        Perro – Animal
        Gato – Animal
)
```



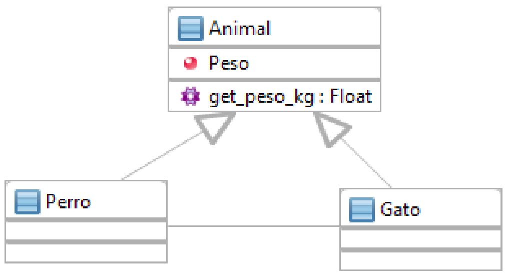
- La declaración de constantes en EKMN se realiza en el Local Data mediante instancias de clases definidas previamente en el Concepto de Modelos.
- La declaración de predicados en EKMN se realiza como atributos de clases o como relación entre clases en notación UML dentro del Modelo de Conceptos:

```
(:predicates

   (peso ?a – Animal ?x -number)

   (persigue ?p – Perro ?g – Gato)

)
```



- La declaración de funciones en EKMN se realiza como operaciones de clases en notación UML dentro del Modelo de Conceptos:

```
(:predicates

(peso ?x) – float )


(: functions

 (get_peso ?x)

 {

 import math

 return (?x1 * 1000)
})
```



En cuanto a la estructura principal, **a continuación se muestran los distintos elementos representados por la notación gráfica EKMN dentro del Conocimiento Experto**, junto con los atributos que serán necesarios describir en notación EKL y la denominación comercial con el que han sido definidos.

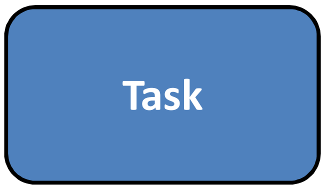
```
<structure-def>:

                                        <action-def>
                               | <durative-action-def>
                                       | <derived-def>
                                      | <htn-task-def>
```

4.1.1 Acciones Primitivas (Task)

A continuación se muestra la descripción BNF de las acciones primitivas en HTN-PDDL:

```
<action-def>:(:action <name>

                                    :parameters (<typed-variable>*)
                                                       [<metags>]
                                             [<preconditions-def>]
                                                  [<effect-def>])
<preconditions-def>:                :precondition <goal-def>
<effect-def>:                       :effect <effect>
<metatags>:                         :meta (<tag>*)
```

La denominación comercial, basada en BPMN, para este elemento es *Task*. Ya que el concepto que representa este elemento en un dominio de planificación jerárquica coincide con el concepto que representa en el modelo BPMN.

Notación Gráfica (EKMN)	Atributos HTN-PDDL	Atributos EKL
	Name	Name
	Parameters	Parameters
	Precondition	Conditions
	Effect	Effects
	Temporal constraints	Temporal constraints
	Metags	Metadata
		Actors [(*)]
		Activity [(*)]

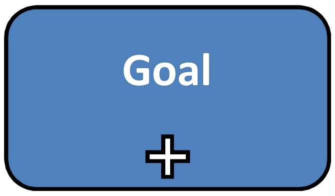
[(*)] Atributos basados en BPMN que han sido añadidos al modelo.

4.1.2 Tareas Compuestas y Objetivos (Goal)

A continuación se muestra la descripción BNF de las tareas compuestas en HTN-PDDL:

```
<htn-task-def>:
                                   (:task <name>
                                   :parameters (<typed-variable>*)
                                   [<meta-tags>]
                                   <method-def>)
<method-def>:

                                   <method>*

                                   | (!<method>*)
                                   <method>: (:method <name>
                                   [<meta-tags>]
                                   [<preconditions-def>]
                                   :tasks <task-network>)
<task-network>:

                                   <task-structure>

                                   | ( )
 <task-structure>:
                                    <task-def>
                                   | ! task_def
                                   | \<<dur-constraints> <task-
                                   structure>+\>
                                   | \<<task-structure>+\>
                                   | (<dur-constraints> <task-
                                   structure>+)
                                   | (<task-structure>+)
                                   | [<dur-constraints> <task-structure>]
                                   | [<task-structure>]
```

La denominación comercial para ambos elementos es la de Goal ya que los objetivos en planificación jerárquica son representados, a su vez, por una tarea compuesta**. La notación gráfica coincidirá con los sub-procesos, formados por varias tareas, de BPMN.**

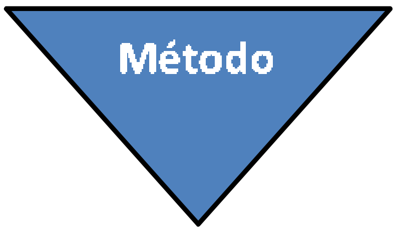
Notación Gráfica (EKMN)	Atributos HTN-PDDL	Atributos EKL
	Name	Name
	Parameters	Parameters
	Method list	Method list

4.1.3 Métodos (Methods)

A continuación se muestra la descripción BNF de los métodos en HTN-PDDL:

```
<method>: (:method <name>
                              [<meta-tags>]
                              [<preconditions-def>]
                              :tasks <task-network>)
```

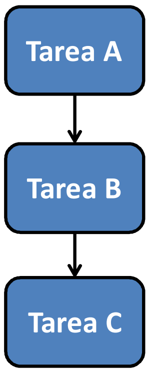
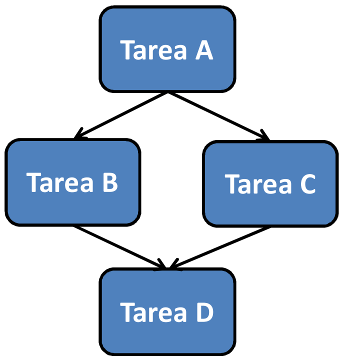
La denominación comercial para este elemento se mantiene. Ya que el concepto de método no existe en BPMN. En cuanto a su representación gráfica, se ha utilizado un elemento geométrico básico no existente en BPMN.

Notación Gráfica (EKMN)	Atributos HTN-PDDL	Atributos EKL
	Name	Name
	Parameters	Parameters
	Preconditions	Conditions
	TaskNetwork	

Como puede observarse, se han eliminado atributos que no son necesarios como la red de tareas ya que estos elementos son descritos directamente mediante la notación gráfica.

4.1.4 Representación de redes de tareas

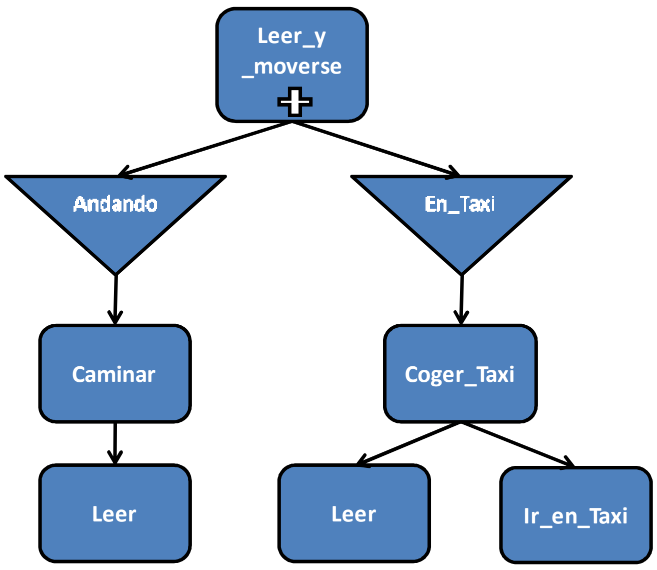
A continuación se especifica cómo se representan las redes de tareas en EKMN en relación a cómo se hace en HTN-PDDL:

• **Tareas secuenciales:** Son aquellas que, en HTN-PDDL, se ejecutan en el mismo orden en el que son expresadas y pueden agruparse entre paréntesis para forzar su ejecución en orden secuencial. En EKMN su ejecución se realiza siguiendo el orden abajo-arriba y el camino que marcan las flechas que asocian a unas tareas con otras. La representación de la figura de la derecha, en HTN-PDDL, se escribiría del siguiente modo: ((Tarea_A) (Tarea_B) (Tarea_C))	
• **Tareas paralelas:** Son aquellas que, en HTN-PDDL, se pueden ejecutar en orden diferente en el que son expresadas o en paralelo, y son representadas agrupadas entre corchetes. En EKMN su ejecución se realiza siguiendo el orden abajo-arriba y cuando encontramos tareas en paralelo son representadas a la misma altura. La representación de la figura de la derecha, en HTN-PDDL, se escribiría del siguiente modo: ((Tarea_A) [(Tarea_B) (Tarea_C)] (Tarea_D))	

4.1.5 Ejemplo

A continuación se muestra un ejemplo en HTN-PDDL y su representación en EKMN:

```
(:task leer_y_moverse
  :parameters (?quien ?origen ?destino ?texto)
  ((:method andando
     :precondition (and (bind ?donde (posicion ?quien)) (<= (distancia ?donde ?destino) (limite_andando ?quien)))
     :tasks ((leer ?quien ?texto)(caminar ?quien ?origen ?destino)))
   (:method en_taxi
     :precondition ()
     :tasks ((coger_taxi ?quien ?taxi) [(leer ?quien ?libro) (ir_en_taxi ?quien ?taxi ?destino)]))))

(:action caminar
  :parameters (?quien ?origen ?destino ?camino)
  :precondition (camino ?origen ?destino ?camino)
  :effect (assign (posicion ?quien) ?destino))

(:action ir_en_taxi
  :parameters (?quien ?origen ?destino ?taxi)
  :precondition (en_taxi ?quien ?taxi)
  :effect (and (assign (posicion ?quien) ?destino) (not (en_taxi ?quien ?taxi)))

(:action coger_taxi
  :parameters (?quien ?taxi)
  :precondition (and (bind ?donde (posicion ?quien)) (bind ?donde (posicion ?taxi)))
  :effect ((en_taxi ?quien ?taxi)))

(:action leer
  :parameters (?quien ?texto)
  :precondition ()
  :effect (assign (leido ?texto ?quien)))
```



4.2 IActive Knowledge Studio: Herramienta de desarrollo de sistemas basados en planificadores jerárquicos basada en notación EKMN.

IActive Knowledge Studio es una herramienta software que permite la representación de gráfica de dominios mediante EKMN para adaptar e integrar un planificador jerárquico HTN [54], que ha sido denominado IActive Decisor, a distintos entornos.

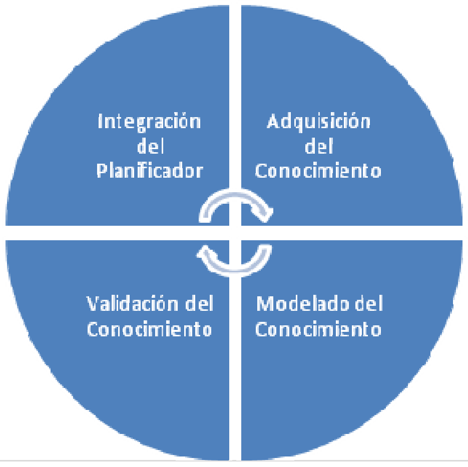
IActive Knowledge Studio

La herramienta IActive Knowledge Studio no sólo ha sido diseñada para facilitar la representación de dominios de planificación jerárquica a personal no experto en planificación, sino que ha sido diseñada para cubrir todo el ciclo de vida de desarrollo de un sistema basado en planificación automática [64] [65]. Existen otras herramientas para la representación de dominios de planificación pero, al contrario que el IActive Knowledge Studio, no cubren todo el ciclo de vida de desarrollo del sistema [66][59] teniendo en cuenta la fase de integración.

Las etapas del ciclo de vida de desarrollo de un sistema basado en planificación automática son las siguientes:

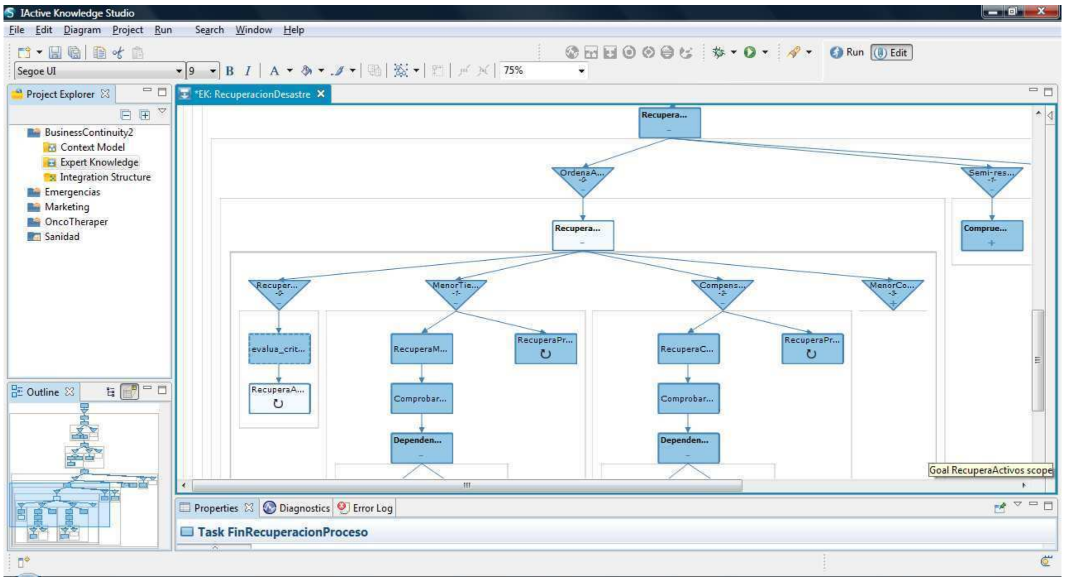
- **Adquisición del conocimiento**: Entrevistas con expertos y análisis de la documentación para obtener el know-how necesario para representar en el dominio.
- **Modelado del conocimiento**: Representación del dominio de planificación basada en el conocimiento adquirido en la etapa previa.
- **Validación del conocimiento**: Validar, con la ayuda de un experto mediante la ejecución de ejemplos, que el conocimiento modelado es correcto.
- **Integración**: Conexión del planificador inteligente con el resto de sistemas software con los que deberá de interactuar para cumplir su función en el sistema desarrollado.

Estas etapas podrán repetirse iterativamente de manera continua para seguir mejorando los resultados obtenidos por el planificador inteligente, ya que su integración es posible que se detecten nuevos requisitos o errores de modelado, así como para desarrollar nuevas versiones del sistema completo.

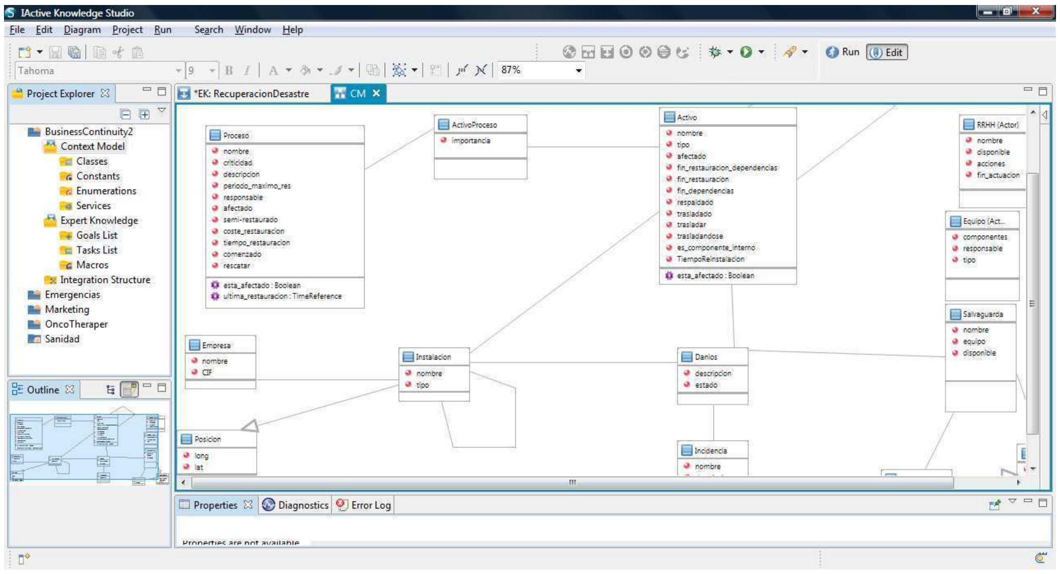

Integración del Planificador
Adquisición del Conocimiento
Validación del Conocimiento
Modelado del Conocimiento

Para dar soporte a estas cuatro fases del ciclo de vida de desarrollo de sistemas basados en planificación automática, la herramienta IActive Knowledge Studio cuenta con cuatro entornos de trabajo:

- **Entorno de edición**: Desde este entorno se pueden representar dominios de planificación jerárquica mediante la notación EKMN para describir la estructura de los dominios de planificación: Es decir, es posible modelar el ***Modelo de Conceptos*** mediante la notación UML y el ***Conocimiento Experto*** mediante la notación EKMN, tal y cómo puede observarse en las siguientes ilustraciones:

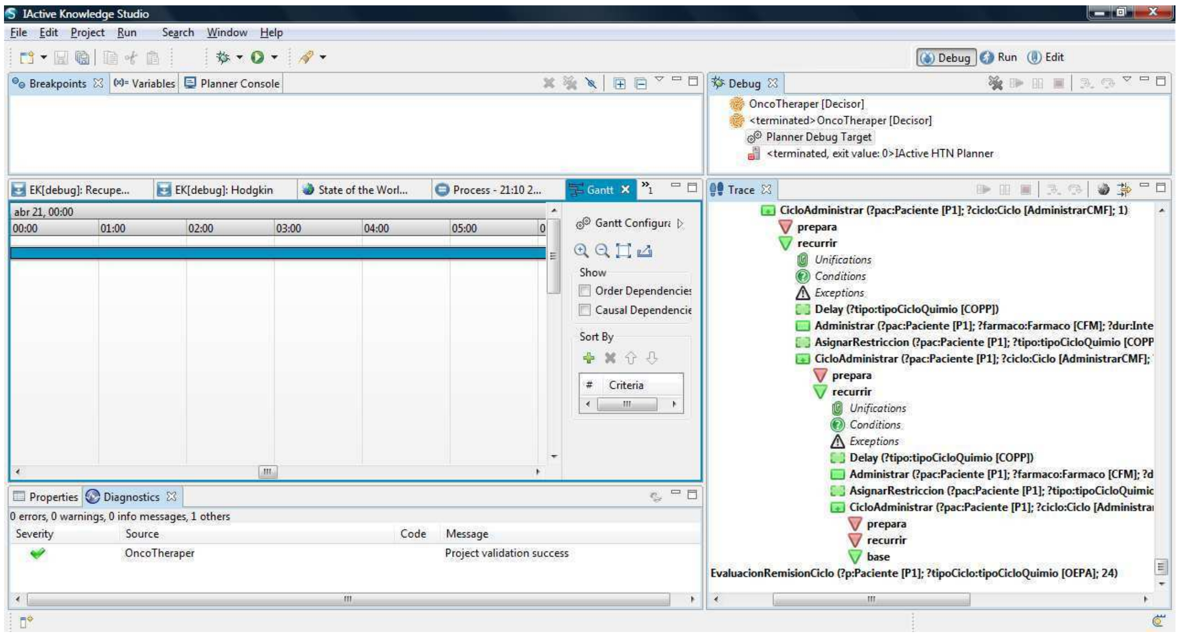


Conocimiento experto (EKMN)



Modelo de Conceptos (UML)

- **Entorno de depuración**: Mediante este entorno permite la ejecución paso a paso del planificador con un dominio y un problema concretos y hacer un trazado de la ejecución del algoritmo de planificación para el dominio y el problema planteados. Esta es una herramienta de gran utilidad para que los Ingenieros del Conocimiento puedan desarrollar los dominios según las especificaciones marcadas por los expertos.



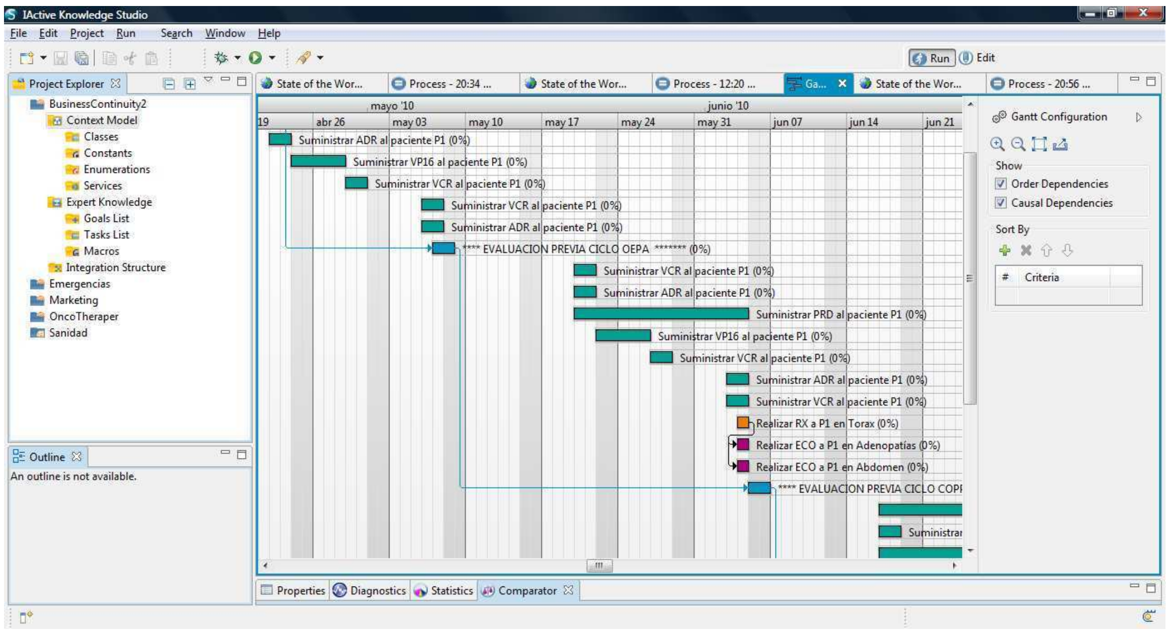
OncoTheraper [Decisor]
<terminated>OncoTheraper [Decisor]
Planner Debug Target
<terminated, exit value: 0>IActive HTN Planner

```
CicloAdministrar (?pac:Paciente [P1]; ?ciclo:Ciclo [AdministrarCMF]; 1)
  prepara
  recurrir
    Unifications
    Conditions
    Exceptions
    Delay (?tipo:tipoCicloQuimio [COPP])
    Administrar (?pac:Paciente [P1]; ?farmaco:Farmaco [CFM]; ?dur:Inte
    AsignarRestriccion (?pac:Paciente [P1]; ?tipo:tipoCicloQuimio [COPP
    CicloAdministrar (?pac:Paciente [P1]; ?ciclo:Ciclo [AdministrarCMF];
      prepara
      recurrir
        Unifications
        Conditions
        Exceptions
        Delay (?tipo:tipoCicloQuimio [COPP])
        Administrar (?pac:Paciente [P1]; ?farmaco:Farmaco [CFM]; ?d
        AsignarRestriccion (?pac:Paciente [P1]; ?tipo:tipoCicloQuimic
        CicloAdministrar (?pac:Paciente [P1]; ?ciclo:Ciclo [Administrar
          prepara
          recurrir
          base
EvaluacionRemisionCiclo (?p:Paciente [P1]; ?tipoCiclo:tipoCicloQuimio [OEPA]; 24)
```

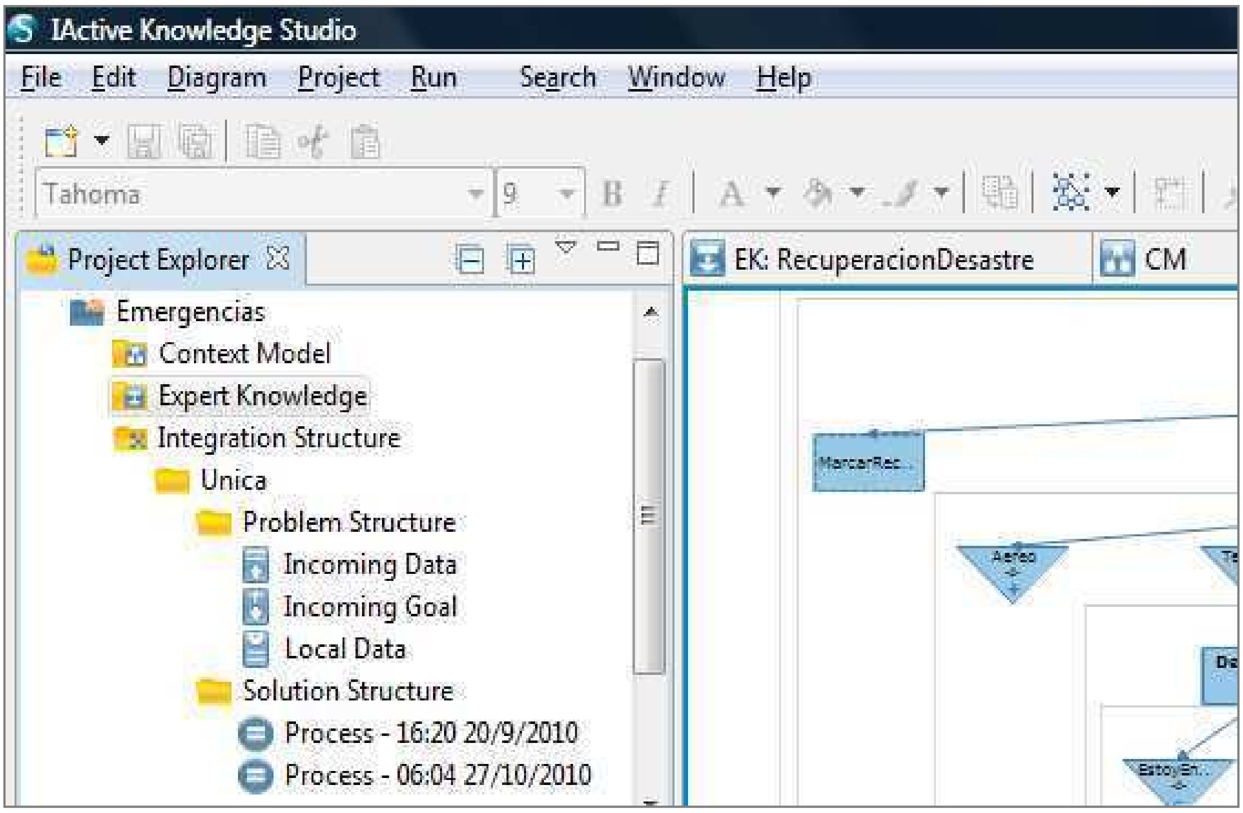
0 errors, 0 warnings, 0 info messages, 1 others

Severity	Source	Code	Message
✓	OncoTherapers		Project validation success

- **Entorno de ejecución**: Mediante este entorno se pueden visualizar gráficamente, mediante un diagrama de Gantt, los planes resultantes de los problemas planteados después de haber realizado una representación del dominio. De este modo, es posible realizar la validación con un experto de los planes que se obtienen y, por tanto, del dominio que se está modelando..


IActive Knowledge Studio
File Edit Project Run Search Window Help
Run Edit
Project Explorer
BusinessContinuity2
Context Model
Classes
Constants
Enumerations
Services
Expert Knowledge
Goals List
Tasks List
Macros
Integration Structure
Emergencias
Marketing
OncoTherapers
Sanidad
Outline
An outline is not available.
State of the Wor... Process - 20:34 ... State of the Wor... Process - 12:20 ... Ga... State of the Wor... Process - 20:56 ...
mayo '10
junio '10
19 abr 26 may 03 may 10 may 17 may 24 may 31 jun 07 jun 14 jun 21
Suministrar ADR al paciente P1 (0%)
Suministrar VP16 al paciente P1 (0%)
Suministrar VCR al paciente P1 (0%)
Suministrar VCR al paciente P1 (0%)
Suministrar ADR al paciente P1 (0%)
**** EVALUACION PREVIA CICLO OEPA ******* (0%)
Suministrar VCR al paciente P1 (0%)
Suministrar ADR al paciente P1 (0%)
Suministrar PRD al paciente P1 (0%)
Suministrar VP16 al paciente P1 (0%)
Suministrar VCR al paciente P1 (0%)
Suministrar ADR al paciente P1 (0%)
Suministrar VCR al paciente P1 (0%)
Realizar RX a P1 en Torax (0%)
Realizar ECO a P1 en Adenopatías (0%)
Realizar ECO a P1 en Abdomen (0%)
**** EVALUACION PREVIA CICLO COPP
Suministrar
Gantt Configuration
Show
Order Dependencies
Causal Dependencies
Sort By
# Criteria
Properties Diagnostics Statistics Comparator

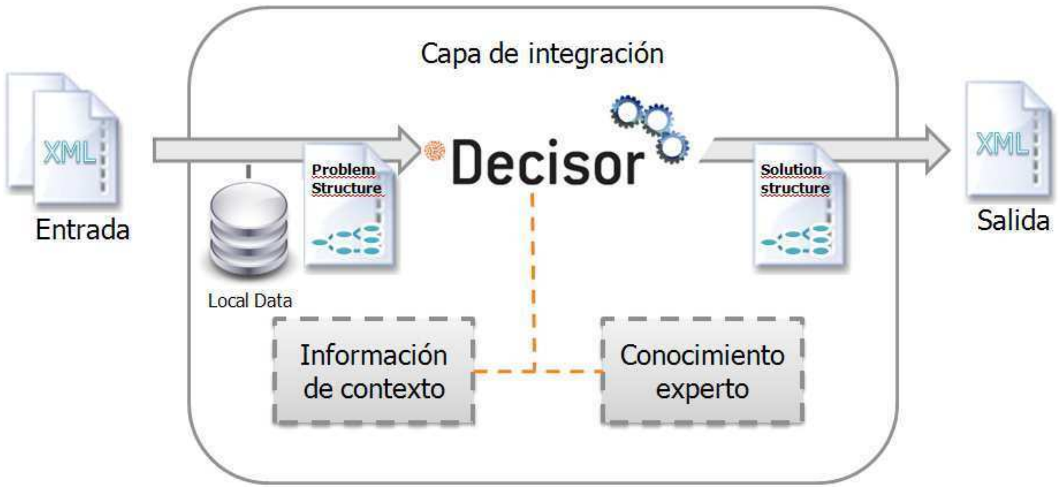
- **Entorno de integración**: Mediante este entorno se facilita la integración del planificador jerárquico, junto con el dominio representado, en un sistema externo que cumpla con los modelos de datos representados en el *Modelo de Conceptos*. Deste este entorno, se define el conjunto de objetos instanciados del ***Local Data***, una estructura de integración de entrada en XML mediante un DTD (Document Type Definiton) que definirá la estructura con la que deben de facilitarse los datos para el ***Incoming Data,*** *una estru*ctura de integración de entrada en XML mediante un DTD (Document Type Definition) que definirá la estrucutra con la que deben de facilitarse los objetivos a resolver para el ***Incoming Goal*** y una estructura de integración de salida en XML mediante un DTD que definirá la estructura, denominada ***Solution Structure***, con la que se facilitarán los planes obtenidos por el planificador automático.



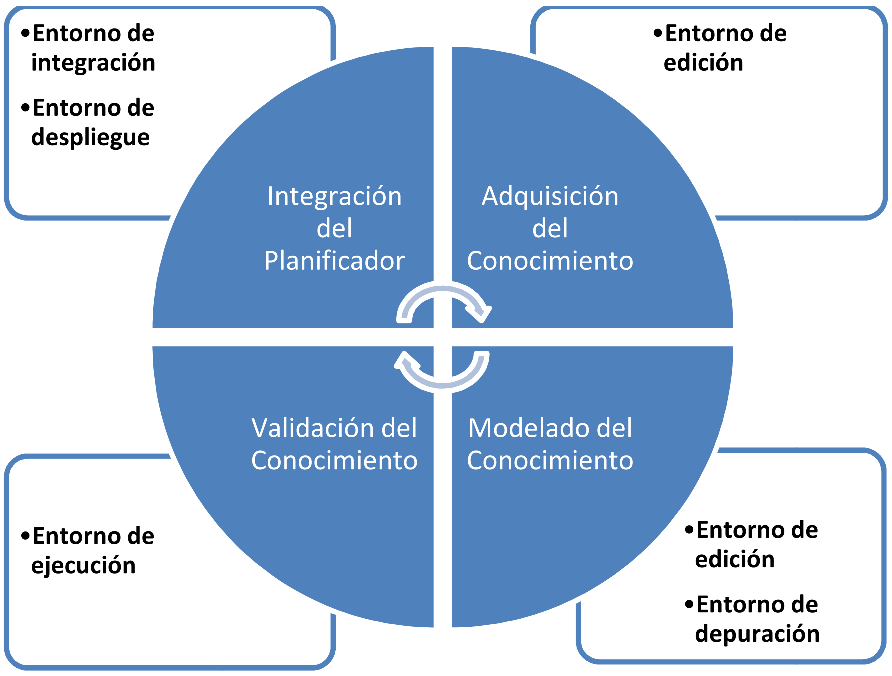
Emergencias
- Context Model
- Expert Knowledge
- Integration Structure
  - Unica
    - Problem Structure
      - Incoming Data
      - Incoming Goal
      - Local Data
    - Solution Structure
      - Process - 16:20 20/9/2010
      - Process - 06:04 27/10/2010

Detalle de los elementos de integración.

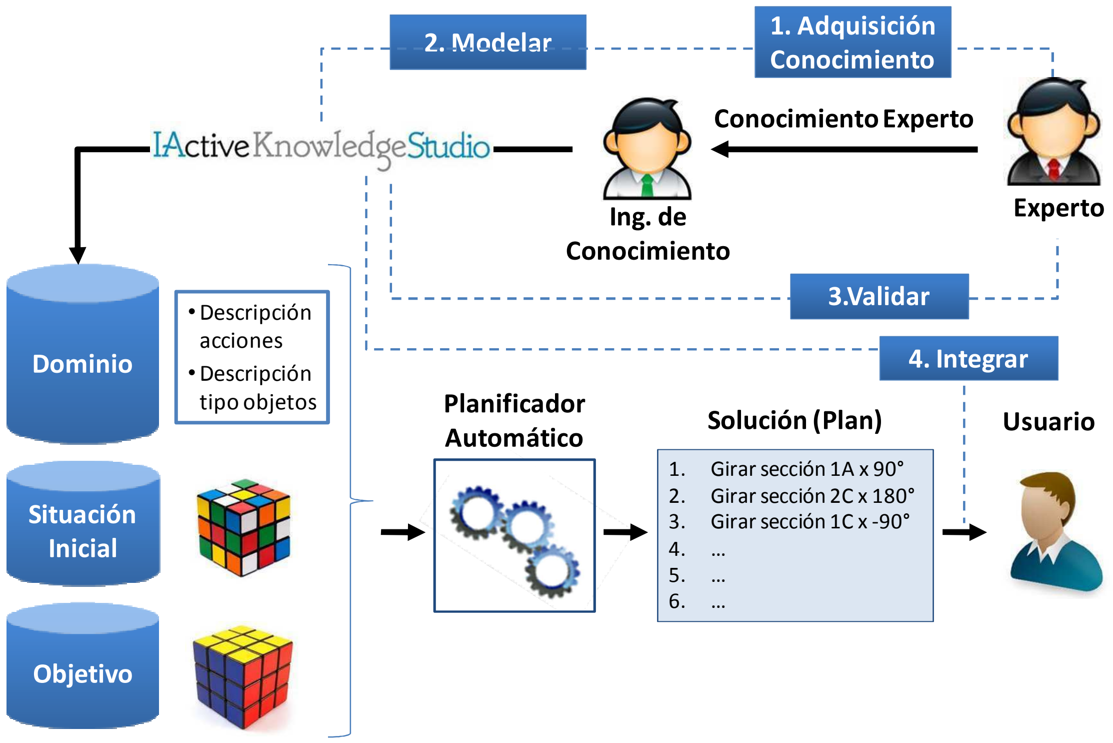
- **Entorno de despliegue**: Funcionalidades que permiten generar un componente software que incorpora el planificador inteligente junto a los elementos definidos del dominio de planificación (Información de Contexto, Conocimiento Experto, Local Data) y las APIs de entrada y de salida necesarias para su integración siguiendo las estructuras de integración definidas (Incoming Data, Incoming Goal y Solution Structure).



A continuación se muestra una imagen que muestra cómo los distintos entornos del IActive Knowledge Studio cubren las distintas fases del ciclo de vida del desarrollo de un sistema basado en un planificador automático.



En definitiva, gracias al IActive Knowledge Studio no sólo se facilita la representación de dominios para planificadores jerárquicos a Ingenieros de Conocimiento de otras compañías, sino que también se han aportado avances a la hora de integrar planificadores en otros sistemas software e, incluso, facilita la labor de validación de los dominios de planificación con los expertos ya que pueden visualizar gráficamente los resultados y hasta comprender la estructura básica de los dominios descritos en EKMN.



5. Avances aportados al mercado

Gracias al diseño de la notación gráfica para la representación de dominios de planificación jerárquica EKMN y la herramienta para el modelado de dominios basados en dicha notación IActive Knowledge Studio, **se han aportado principalmente tres tipos de avances al mercado** que se listas a continuación y se describirán con más detalle en las siguientes secciones:

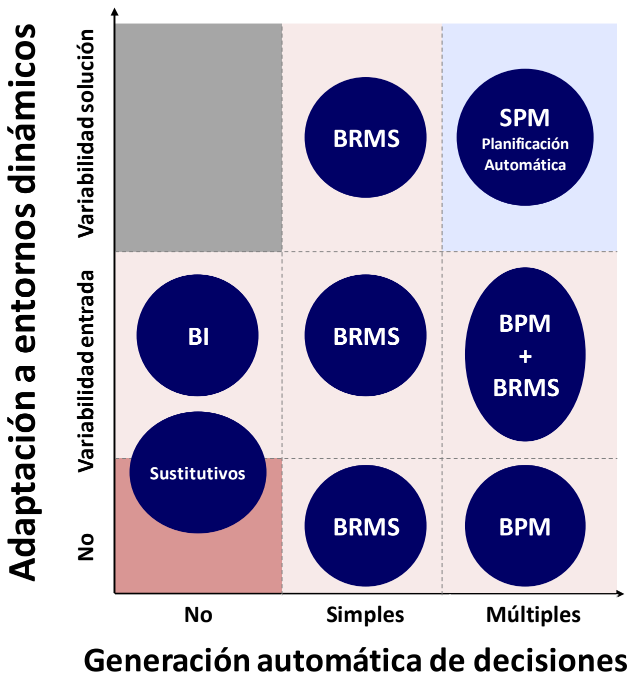
1) **Posicionamiento de los planificadores jerárquicos en un nuevo nicho de mercado:** Se ha demostrado con datos la entrada de los planificadores jerárquicos al mercado, cubriendo un nuevo nicho de mercado en las tecnologías de automatización de procesos y de ayuda a la toma de decisiones.
2) **Aumento del uso comercial de planificadores jerárquicos:** Ha aumentado el número de Ingenieros Informáticos que es capaz de modelar dominios de planificación jerárquica. Tal y cómo se describió al comienzo de este documento, este era el objetivo principal del trabajo y, como se detallará más adelante, veremos que el número de Ingenieros Informáticos que actualmente utilizan planificadores automáticos se ha elevado considerablemente.
3) **Desarrollo de aplicaciones comerciales mediante el IActive Knowledge Studio:** Se han desarrollado aplicaciones comerciales basadas en el planificador jerárquico utilizando el IActive Knowledge Studio que, gracias a la incorporación de técnicas de planificación automática, han permitido a las empresas proveedoras de dichas aplicaciones conseguir una fuerte diferenciación en el mercado respecto a la competencia.

5.1 Posicionamiento de planificadores jerárquicos en nuevos nichos de mercado

Tal y cómo se describió en las motivaciones de la sección inicial de este documento, la tecnología en planificación automática cubre la actual necesidad de mercado de dar soporte a la automatización de procesos de negocio en entornos dinámicos [65] y también incluyendo capacidades de razonamiento temporal que no habían sido cubiertas hasta ahora [4].

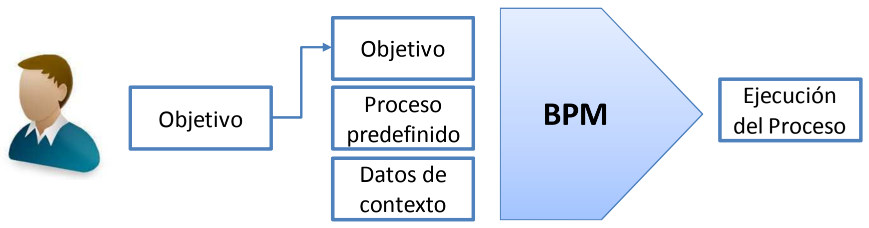
Gracias al desarrollo de la notación gráfica EKMN y a la herramienta IActive Knowledge Studio, se permite el acceso y el posicionamiento de los planificadores jerárquicos en el mercado dentro de un nuevo *Océano Azul* o nicho de mercado.

Para posicionar adecuadamente a los planificadores jerárquicos dentro del mercado, se ha definido la siguiente denominación de mercado: **SPM (Smart Process Management)** [34].

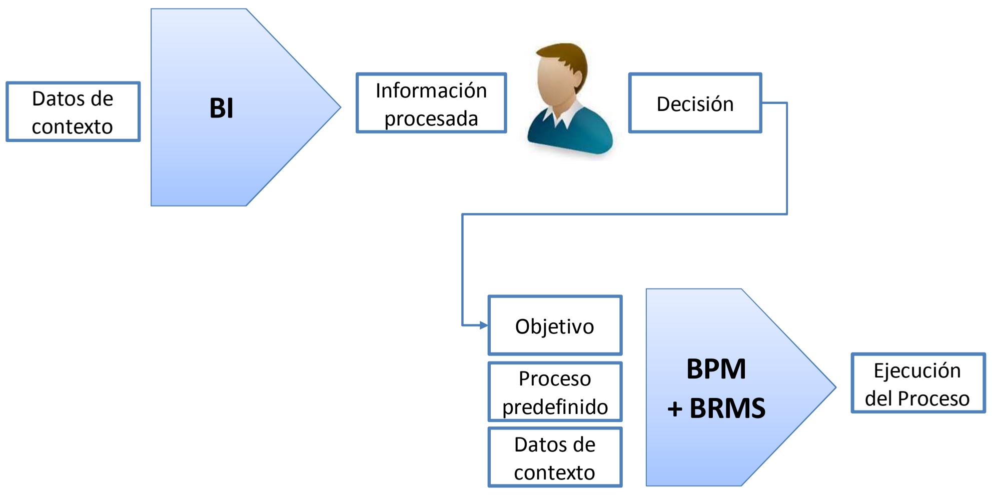


Según lo descrito anteriormente, a continuación se muestran posibles configuraciones y relaciones entre distintas tecnologías para dar soporte a la ayuda en la toma de decisiones para la automatización de procesos de negocio en distintos tipos de entornos.

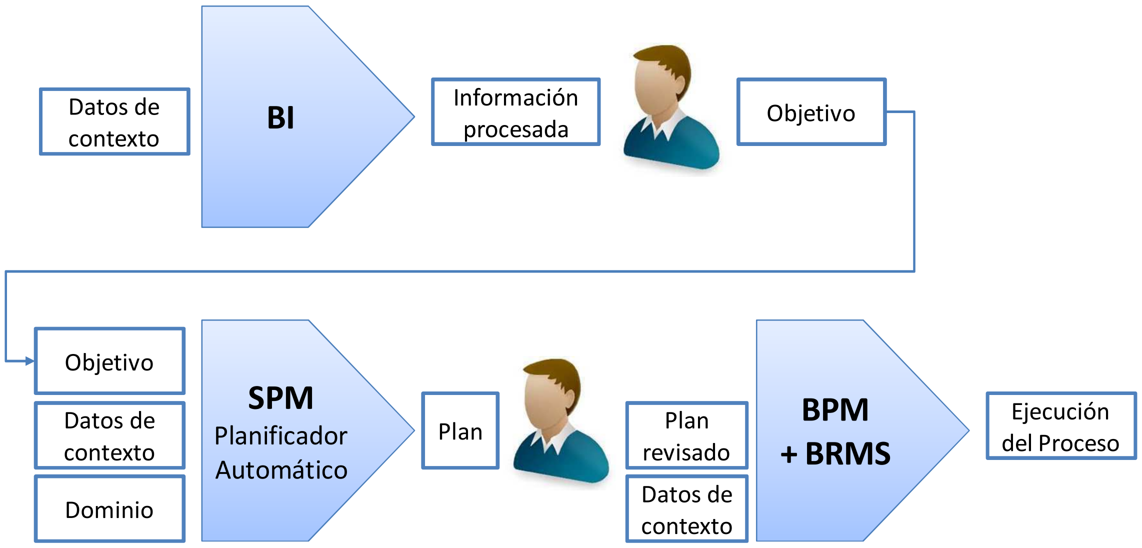
Automatización de procesos en un entorno no dinámico mediante un BPM:



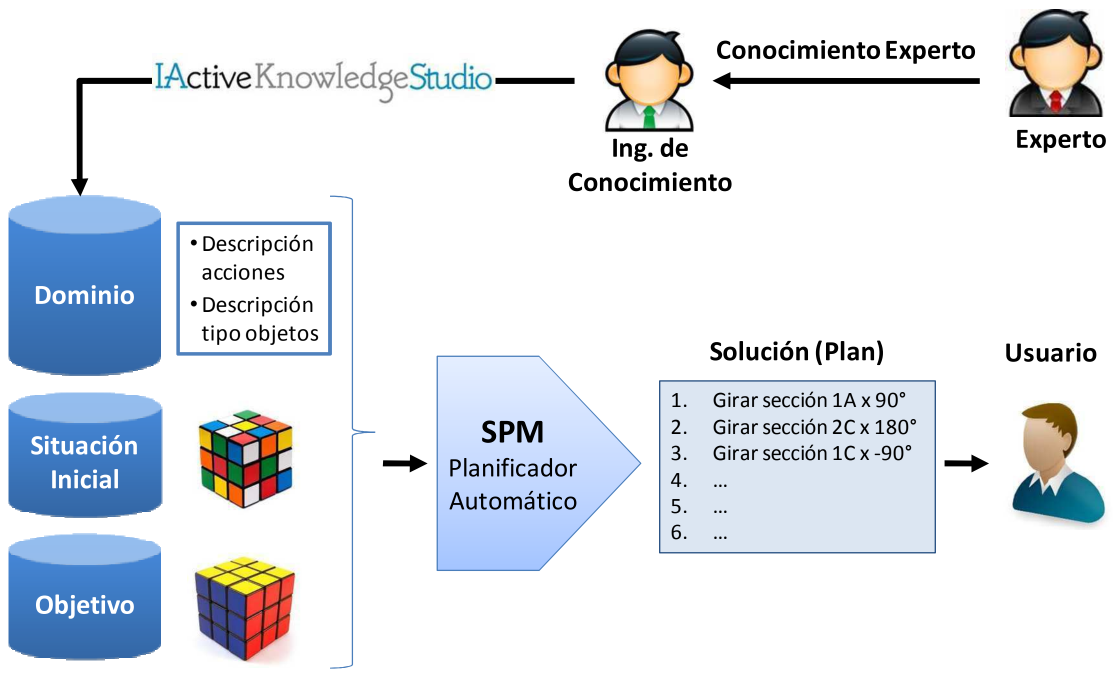
Automatización de procesos en un entorno dinámico, con múltiples variables apoyada con un BI para procesar la información, mediante un motor BPM complementado con BRMS:



Automatización de procesos en un entorno dinámico, con múltiples variables de entorno y gran variabilidad en el proceso, apoyada con un BI para procesar la información y con un planificador automático para sugerir el proceso adecuado, mediante en un motor BPM complementado con BRMS:



Merece la pena aclarar que, aunque los planificadores automáticos puedan integrarse con sistemas BI y BPM para satisfacer una necesidad de mercado no cubierta por estos sistemas en cuanto a automatización de procesos se refiere, **los planificadores automáticos también son utilizados por si solos como tecnología de ayuda a la toma de decisiones** [2] [8] para dotar a las aplicaciones software de la capacidad de generar planes de acciones de forma automática que serán ofrecidos al usuario como sugerencia de decisiones en su propia actividad. Algunos ejemplos de este tipo de aplicaciones serán vistos en la siguiente sección.

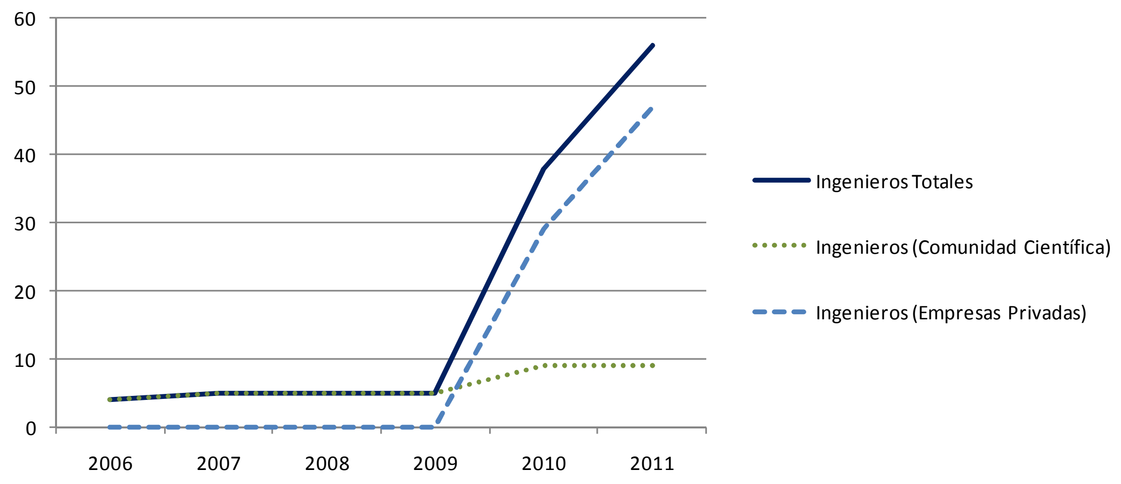


5.2 Aumento del uso comercial de planificadores jerárquicos

Como ha sido descrito en este documento, en el año 2005 se desarrolló un planificador jerárquico propio [2], que fue utilizado por sus creadores y ocasionalmente por algún miembro de la comunidad científica en el desarrollo de aplicaciones reales utilizando el lenguaje HTN-PDDL [54]. En 2009 fue definida la notación gráfica EKMN [7] y en 2010 se desarrolló la herramienta IActive Knowledge Studio, que dio soporte a la utilización de dicha notación para el modelado de dominios del planificador jerárquico previamente desarrollado.

A continuación, se pueden observar los datos relativos al uso del planificador jerárquico y, como gracias a la definición de la notación gráfica y al desarrollo de la herramienta IActive Knowledge Studio, se ha conseguido aumentar la adopción y el uso extendido de las técnicas de planificación, ha aumentado considerablemente el uso del planificador no sólo por miembros de la comunidad científica sino también por Ingenieros Informáticos de compañías privadas.

Número de Usuarios del Planificador Jerárquico



Nº Usuarios Planificador Jerárquico	**2006**	**2007**	**2008**	**2009**	**2010**	**2011 (*)**
Ingenieros (Comunidad Científica)	4	5	5	5	9	9
Ingenieros (Empresas privadas)	0	0	0	0	29	47
Ingenieros Totales	**4**	**5**	**5**	**5**	**38**	**56**

(*) Previsión con datos reales desde 28 de febrero de 2011

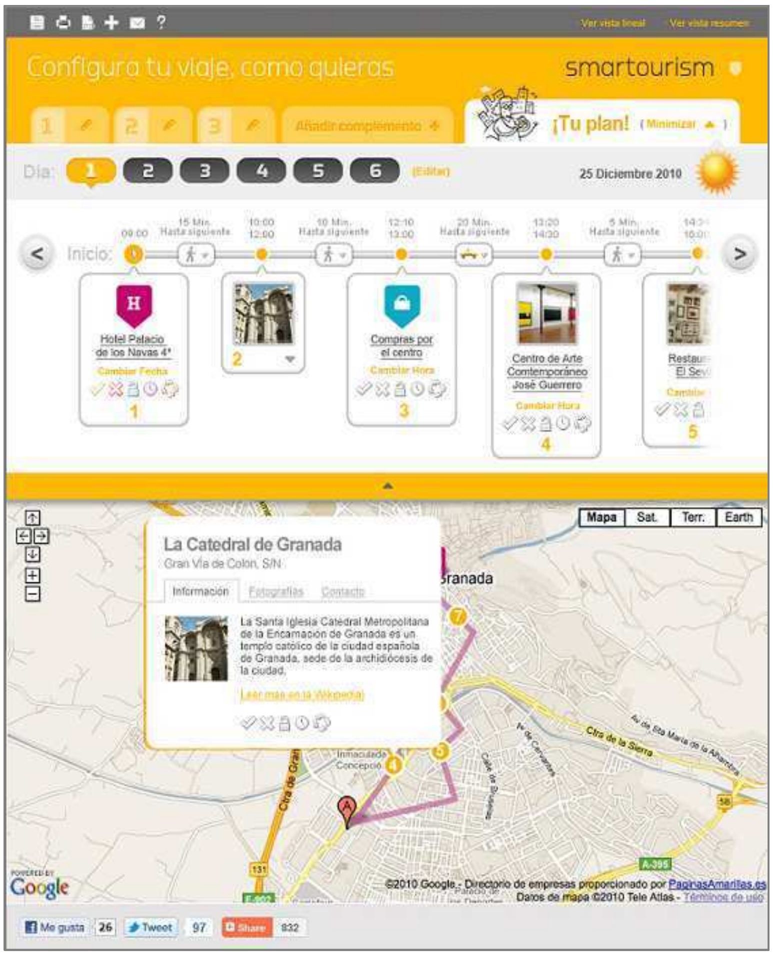
5.3 Aplicaciones comerciales desarrolladas con IActive Knowledge Studio

Desde su lanzamiento al mercado, IActive Knowledge Studio ha sido utilizado para desarrollar distintas aplicaciones de ayuda a la toma de decisiones para distintos sectores de actividad. Algunas de ellas han sido desarrolladas por la propia compañía IActive y otras de ellas han sido desarrolladas mediante otras compañías tecnológicas.

A continuación se describen dos de las aplicaciones desarrolladas hasta la fecha y que presentan como mayor innovación la incorporación de un planificador automático para ayudar en la toma de decisiones al usuario.

5.3.1 Smartourism: Sistema inteligente para personalizar visitas turísticas

Este sistema ha sido desarrollado directamente por la compañía IActive y consiste en una aplicación en entorno web que ayuda al turista a decidir qué visitar y qué hacer durante su estancia en la ciudad [8].



La Catedral de Granada
Gran Vía de Colon, S/N

Información | Fotografías | Contacto

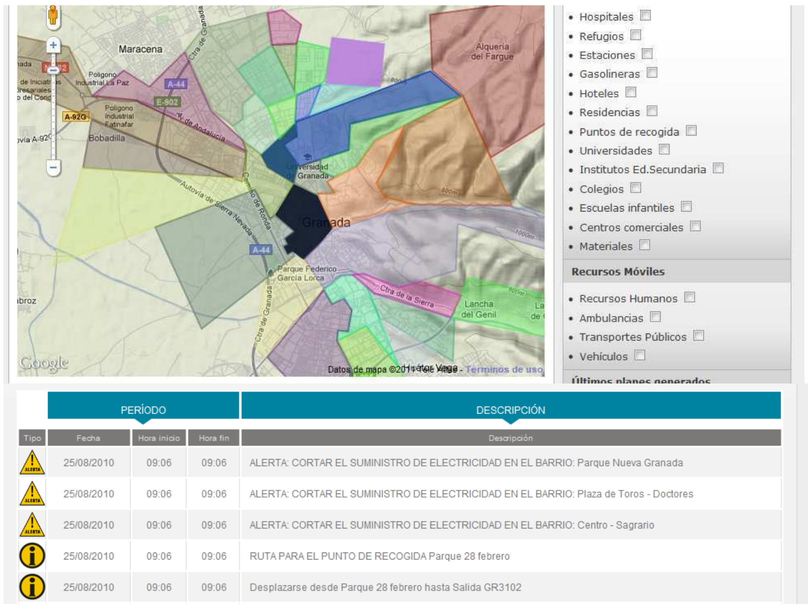
La Santa Iglesia Catedral Metropolitana de la Encarnación de Granada es un templo católico de la ciudad española de Granada, sede de la archidiócesis de la ciudad.

Leer más en la Wikipedia

El sistema está actualmente implantado en el **Ayuntamiento de Granada** y se está trabajando para implantarlo en distintos servicios de turismo pertenecientes a otros ayuntamientos y comunidades autónomas.

5.3.2 Gestión de emergencias y evacuaciones

Cuando ocurre una gran catástrofe que afecta a un núcleo urbano, como puede ser una inundación, terremoto, etc. es necesario actuar con rapidez y seguridad para poner en marcha operaciones de evacuación de la población. Basado en el planificador automático, se ha desarrollado un sistema que es capaz de generar planes de evacuación de civiles para núcleos urbanos teniendo en cuenta los protocolos de evacuación, los recursos disponibles y la información del contexto.



	PERÍODO			DESCRIPCIÓN
Tipo	Fecha	Hora inicio	Hora fin	Descripción
⚠	25/08/2010	09:06	09:06	ALERTA: CORTAR EL SUMINISTRO DE ELECTRICIDAD EN EL BARRIO: Parque Nueva Granada
⚠	25/08/2010	09:06	09:06	ALERTA: CORTAR EL SUMINISTRO DE ELECTRICIDAD EN EL BARRIO: Plaza de Toros - Doctores
⚠	25/08/2010	09:06	09:06	ALERTA: CORTAR EL SUMINISTRO DE ELECTRICIDAD EN EL BARRIO: Centro - Sagrario
ℹ	25/08/2010	09:06	09:06	RUTA PARA EL PUNTO DE RECOGIDA Parque 28 febrero
ℹ	25/08/2010	09:06	09:06	Desplazarse desde Parque 28 febrero hasta Salida GR3102

El sistema se ha desarrollado para la **Unidad Militar de Emergencias del Ministerio de Defensa** de España y ha sido reconocido por la prestigiosa analista tecnológica de mercado Gartner [9], como un caso de estudio.

5.3.3 Business Continuity Planning

Este sistema ha sido desarrollado en colaboración con la consultora tecnológica de Ernst&Young y consiste en una aplicación que realiza la automatización de la actuación necesaria cuando ocurre algún incidente en los activos tecnológicos de una gran compañía, de manera que genera y podría ejecutar, con la ayuda de un BPM, las acciones necesarias para minimizar los riesgos y los daños ocurridos por la incidencia así como para que el negocio retome su actividad normal en el menor tiempo posible.

El sistema se está actualmente implantando en **Bankinter** y, tras su finalización, se ofrecerá a otras entidades público y privadas que requieran de sistemas de continuidad de negocio para dar mayor seguridad a su actividad.

6. Conclusiones y trabajo futuro

6.1 Conclusiones

Durante todo el proceso de transferencia tecnológica descrito en la tesis, se han obtenido **las siguientes conclusiones** que serán de gran utilidad no sólo para el estado del arte del área de Planning & Schedulig sino también para futuros procesos de I+D+i:

- **La tecnologías BPM y de Planificación Automática resuelven problemas de automatización diferentes** por lo que no son competencia directa sino complementarias. Además, las tecnologías de Planificación Automática permiten la creación de un nuevo *Océano Azul* al soportar la generación de procesos en entornos altamente dinámicos [8][20].

- **A la hora de definir notaciones gráficas para representar el conocimiento**, con el objetivo de hacerlas llegar y de utilidad para el máximo número de personas, se han de seguir estándares conocidos y los nuevos elementos necesarios basarlos en estos estándares. Es necesario basarse en estándares de mercado para facilitar el acceso de algoritmos y de avances científicos a Ingenieros que desarrollan su actividad en empresas privadas [7].

- **Al comienzo de la etapa de investigación es útil realizar un estudio de mercado basado en necesidades** ya que orientará los objetivos y los desarrollos realizados en esta y en las siguientes fases de transferencia tecnológica lo que reducirá esfuerzos y aumentará la probabilidad de éxito futuro en el mercado. Para este estudio de mercado es muy útil realizar un proceso de prototipado rápido que pueda ser validado por usuarios finales reales y cuyo feedback pueda ser utilizado dentro de la fase de investigación aplicada.

  En este sentido, para este trabajo ha sido especialmente importante dar respuesta a las necesidades de razonamiento temporal [4] y a las de integración en arquitecturas SOA que existen dentro del mercado [6].

- **Al comienzo de la etapa de desarrollo es útil realizar un estudio de competencia**, basado en productos, para plantear enfoques de innovación en valor basado en características diferenciales y no en aumento de prestaciones que deteriore el margen de los productos.

6.2 Trabajo Futuro

Todavía quedan etapas por recorrer para seguir acercando el modelado de dominios a cualquier Ingeniero Informático de un modo sencillo, tal y cómo ocurre al trabajar con la mayoría de los lenguajes de programación comerciales.

Para seguir alcanzando este objetivo, el próximo paso es evolucionar el lenguaje EKL (Expert Knowledge Language) para que tenga una sintaxis más parecida a los lenguajes actuales orientados a objetos.

Por otro lado, en lo que a los planificadores automáticos respecta, una debilidad que poseen que los hace poco eficientes en entornos comerciales es la gran necesidad de información inicial que requieren para llevar a cabo el proceso de planificación. Un planificador automático requiere recibir como entrada el conjunto de información del contexto inicial para definir un plan que alcance el objetivo a partir de dicho contexto. Sin embargo, en muchas ocasiones la información está dispersa en distintas bases de datos o, aunque esté en una única base de datos, el coste que tiene recoger toda la información inicialmente es demasiado elevado. Por lo tanto, se hace necesario evolucionar los algoritmos de planificación para que no requieran toda la información de contexto inicialmente.

En estas dos líneas de trabajo, entre otras, avanzará la compañía IActive junto con el Grupo de Investigación Sistemas Inteligentes, siendo estas necesidades de mercado inputs de las próximas fases de investigación aplicada que se desarrollen. Lo cual, tal y cómo comentábamos en la sección anterior, aportará información útil a las próximas labores de investigación y, con total seguridad, aumentará el valor de los resultados obtenidos en el futuro.

II. Publicaciones: Trabajos publicados, aceptados y sometidos, y otros méritos

1. Artículos publicados durante el desarrollo de la tesis

Revistas

1. M. de la Asunción, L. Castillo, J. Fdez-Olivares, O. García-Pérez, A.González, F. Palao. SIADEX: an interactive artificial intelligence planner for decision support in forest fire fighting. Artificial Intelligence Communications (ISSN 0921-7126),Volumen: 18 (4) Páginas, 257-268, 2005. (JCR 2009: 0.755)
2. M. de la Asunción, L. Castillo, J. Fdez-Olivares, O. García-Pérez, A.González, F. Palao.Título: Handling fuzzy temporal constraints in a planning environment. Annals of Operations Research “Special Issue on Personnel Scheduling and Planning” (ISSN: 0254-5330) Volumen: 155(1) Páginas: 391-415. 2007(JCR 2009: 0.961)

Capítulos de Libro

1. M. de la Asunción, L. Castillo, J. Fdez-Olivares, O. García-Pérez, F. Palao. Knowledge and plan execution management in planning fire fighting operations. Planning, Scheduling and Constraint Satisfaction: From theory to practice, L. Castillo, D. Borrajo, M.A. Salido, A. Oddi (Editors) (ISBN 1-58603-484-7) Páginas: 149-158. 2005. IOS Press, Holanda.
2. Juan Fernandez Olivares, Tomás Garzón, Luis Castillo Vidal, Óscar García Pérez, Francisco Palao. A Middleware for the automated composition and invocation of semantic web services based on HTN planning techniques. Current topics in Artificial Intelligence (CAEPIA 2007). Springer LNAI 4788, Volumen: 4788, Página: 70-79.2007. Awarded as the best AI technology transfer application during the 1st Spanish Conference on Computer Science 2005 (CEDI 2005).
3. Marc de la Asuncion, Óscar García Pérez, Francisco Palao. SIADEX: A Real World Planning Aproach for Forest Fire Fighting. STAIRS 2004. ISBN:1-58603-451-0. IOS Press.

Congresos

1. L. Castillo, J. Fdez.-Olivares, O. García-Pérez, F. Palao, T.Garzon, R.Raya, E.Hidalgo. EKDL: A graphic notation for modeling hierarchical planning domains. ICAPS 2011, Knowledge Engineering for Planning and Scheduling Workshop.

2. L. Castillo, J. Fdez.-Olivares, O. García-Pérez, F. Palao, T.Garzon, R.Raya, E.Hidalgo. Smartourism: An Intelligent Systems for personalizing Sightseeing trip. IJCAI Intelligent Planet Workshop
3. L. Castillo, J. Fdez-Olivares, O. García-Pérez, A. Gonz\'alez, F. Palao. Reducing the impact of AI Planning on end users. ICAPS 2007, Workshop on Moving Planning and Scheduling Systems into the Real World, 2007.
4. F. Palao-Reinés, Improving Planning Techniques for Web Services, 2006, Doctoral Consortium ICAPS 06.
5. L. Castillo, J. Fdez.-Olivares, O. García-Pérez, F. Palao. Bringing users and planning technology together. Experiences in SIADEX. 16th International Conference on Automated Planning and Scheduling (ICAPS 2006). Awarded as the "Best Application Paper" of the 2006 edition
6. L. Castillo, J. Fdez.-Olivares, O. García-Pérez, F. Palao. Efficiently handling temporal knowledge in an HTN planner. 16th International Conference on Automated Planning and Scheduling (ICAPS 2006).
7. L. Castillo, J. Fdez.-Olivares, O. García-Pérez, F. Palao. SIADEX. An integrated planning framework for crisis action planning. International Conference on Automated Planning and Scheduling, ICAPS 2005, Software Demonstrations Track.
8. L. Castillo, J. Fdez.-Olivares, O. García-Pérez, F. Palao. Sistema Inteligente de Ayuda a la Decisión para el diseño de Planes de Extinción: SIADEX. Incendios Forestales Magazine, Spring 2005.
9. M. de la Asunción, L. Castillo, J. Fdez.-Olivares, O. García-Pérez, A. González, F. Palao. SIADEX: an interactive artificial intelligence planner for decision support in forest fire fighting. Workshop on binding Environmental Sciences and Artificial Intelligence (BESAI 2004), European Conference on Artificial Intelligence, 2004.
10. M. de la Asunción, L. Castillo, J. Fdez.-Olivares, O. García-Pérez, A. González, F. Palao. Knowledge and plan execution management in planning fire fighting operations. Workshop on Planning and Scheduling: Bridging Theory to Practice. European Conference on Artificial Intelligence, 2004
11. M. de la Asunción, L. Castillo, J. Fdez.-Olivares, O. García-Pérez, A. González, F. Palao. Local (human-centered) Replanning in the SIADEX Framework. X Conference of the Spanish Association for Artificial Intelligence, Workshop on Planning, Scheduling and Temporal Reasoning, 11 November, San Sebastian, Spain.

SIADEX: an interactive knowledge-based planner for decision support in forest fire fighting

Marc de la Asunción and **Luis Castillo** and **Juan Fernández-Olivares** and **Oscar García-Pérez** and **Antonio González** and **Francisco Palao** [1]

Abstract. SIADEX is a complex framework that integrates several AI techniques able to design fighting plans against forest fires. It is based on four main components, a web server, that centralizes all the flow of information between the system and the user, the ontology server, that is the cornerstone of the architecture as the basis for knowledge sharing and exchange between all the components, and the planning and monitoring servers that are offered as intelligent services through the web server. This perspective also allow to view SIADEX as a collaborative working environment where it provides two basic functionalities: the intelligent services offered to the user (the ontology, planning and monitoring modules) and a middleware level that interfaces these back-end services to the front-end software of the user (a web browser) achieving several valuable goals like transparent access of the user to a distributed architecture, an ubiquitous access to the services that allow the mobility of the user and his/her independence of the access device.

Keywords: planning, knowledge representation, uncertainty management.

1 Introduction

The management of crisis episodes usually follows a sequence of stages like that shown in Figure 1, in which several stages of situation assessment, intervention planning, plan dispatching and execution and plan execution monitoring are followed up to completely overcome the crisis. Due to the dinamicity and the many uncertainties of real world, a revision process of the intervention plans might be carried out, triggering again a new cycle.

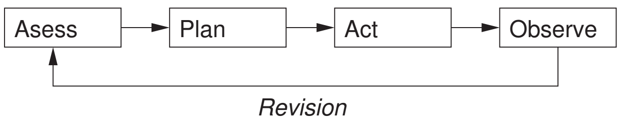


Figure 1. Lifecycle of crisis management

This cycle usually relies in a chain of decision making stages from high skilled staff (in charge of assessing the situation, defining the goals of the intervention, and the strategies to achieve these goals) to ground operators (in charge of carrying out activities). Nowadays, the increasing use of more sofisticated tools and the extensive presence of computers at every decision making stage, produces an overwhelming amount of information to be processed by technical staff, making their job even more stressing and difficult. It is clear in the literature that the use of decision support systems based on a variety of artificial intelligence techniques may help to analyze this information, track it during the evolution of the episode and even to share part of the decisions.

Particularly, AI planning techniques have shown to perform very efficiently in providing technical staff with valuable tentative plans and strategies either in military domains (operations planning[21], air campaign design[16] or noncombatant evacuation operations[18]) or in civil domains (oil spills[3], floods [4] or forest fires[10, 1, 19]). These AI planning techniques are only the core of complex architectures that may also involve other AI techniques (like deductive frameworks, uncertainty propagation, systems modeling, etc) and their plans might be directly scheduled for real execution or adapted by hand by technical staff before its execution.

However, for these AI planning techniques to be successful, several issues must be seriously taken into account.

- Easy access to knowledge for end users. AI planning techniques strongly relies in a detailed representation of the knowledge of the problem and this knowledge must be easily accessible for technical staff since they will be responsible of using the system, stating goals, modify plans, etc. The fact is that these people are not required to know about AI planning techniques or formal representations, so the access to the knowledge must be transparent for the end users. Additionally, the mobility of the user should be considered since technical staff might access from any place and from any device and platform (laptop, PDA, desktop computer).
- Integration with legacy software and context awareness. Part of the knowledge required to react to some crisis episodes is not given by end users, but it is stored in already existing systems in the host institution like for example geographic information systems (GIS) or meteorological conditions and forecasts and the current state of resources, etc. This implies a large amount of information that nobody wants to re-type by hand since it is already available in electronic format like files and databases. Therefore, in order to succeed, these systems must have the capability of accessing these files and databases and translating them into the knowledge representation framework suitable for AI planning techniques.
- Explicit support for flexibility and dinamicity. Knowledge of real world problems is always faulty due to incompleteness or unpredictability. Therefore, any real application of these systems must explicitly deal with uncertainty in any of its forms.

[1] Dpt. of Computer Science and AI, ETSI Informatica, Universidad de Granada, Spain email: siadex@decsai.ugr.es

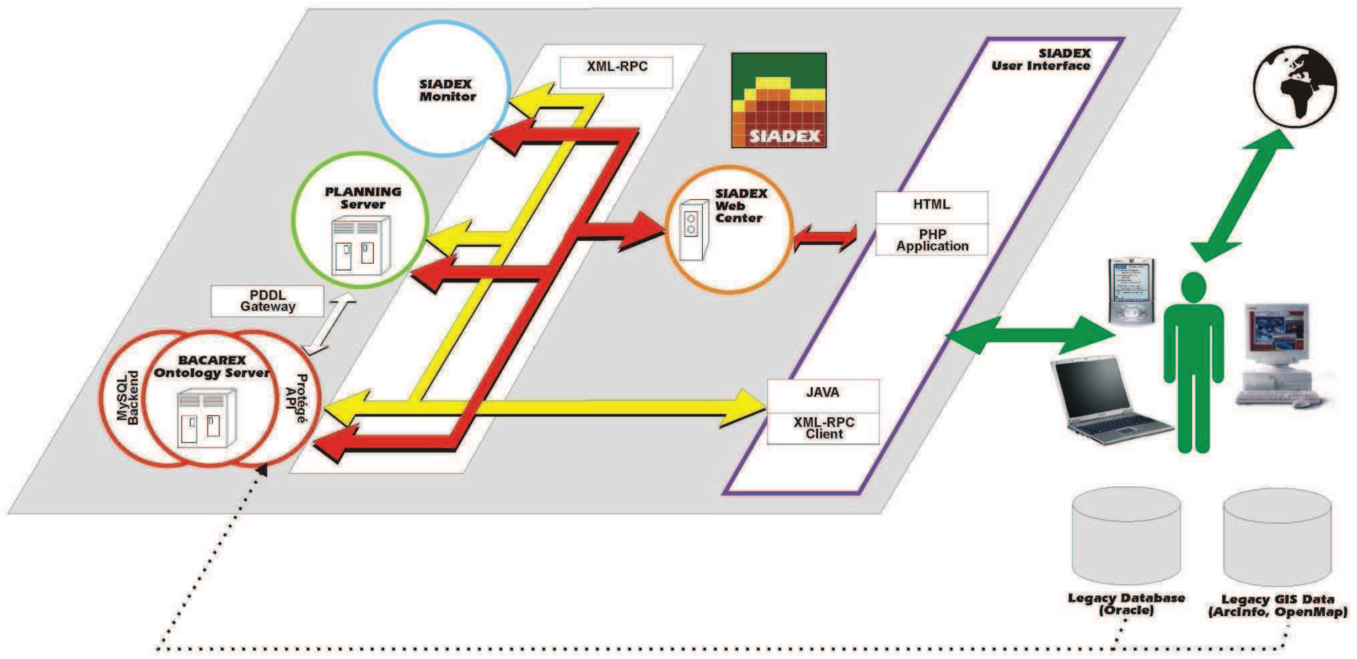


Figure 2. Overall architecture of SIADEX

In the following, all these questions are sketched explaining the whole architecture of SIADEX, an intelligent distributed digital assistant that is being developed by the authors for the assisted design of forest fire fighting in the Andalusian Regional Government for the Environment.

2 Architecture

The whole architecture of SIADEX is outlined in Figure 2 although it is not fully developed yet. Nevertheless, most part of the techniques explained in this section have already been implemented and tested in previous projects, in industrial settings [6, 8, 9, 12], and now are being integrated into a unique system.

Figure 2 shows a distributed architecture composed of several modules (surrounded with a circle) that may run in different machines so that the load of the processes could be distributed. All the processes communicate to each other following the XML-RPC protocol[20], so that visibility is guaranteed in any condition that supports TCP/IP. The user interface has been designed to give full accessibility in almost any condition since staff might be accessing either from their office (with a warm environment, a desktop computer, a broad band connection, etc) or from the countryside (with a hostile environment, a laptop or a PDA with, possibly, a low band connection). Therefore, two access methods were designed, one of them based on web browsing (html and javascript), so that users may access from almost any computer-based means or any operating system with web browsing capabilities (MS Windows, Linux, Palm OS, Pocket PC). The other one is based on a Java application whose hardware and software requirements are slightly higher. In any case, both interfaces access to an XML-RPC gateway that connects the most important modules of SIADEX: the knowledge base BACAREX, the planner and the monitor.

2.1 The ontology server

The backbone of the architecture is BACAREX, the knowledge base. It is an ontology of planning objects designed with Protégé 2000[2] that contains resources (persons, squads, vehicles, facilities, ...), activities (transport, water spraying, refueling, ...) and additional data (GIS information, weather forecasting, ...) what configures a standalone module: *the ontology server*.

It is, thus, responsible of providing the knowledge required by the planning process, but it is also an open platform to a continuous update and query of the technical staff. The ontology server has a back-end in MySQL so that the flexibility and efficiency of data storage is guaranteed and multiple users/processes in parallel are also allowed. It also provides both offline and online access facilities. Offline access is done by the standard Protégé framework, so that the development team may access the knowledge and carry out maintenance and validity checking operations with full operability. On the contrary, online access is done through the Protégé API on which both a Java application and a web access service have been built. This online access is devoted to staff users that do not have skills on knowledge representation for planning but may painlessly access the knowledge in a web browsing fashion by means of a hierarchy of objects and activities close to their understanding of the problem. Instances in the ontology server have been extracted from legacy software and extended to the level of detail required by SIADEX by plugins and mappers from different applications so that they do not need to be re-typed again: most information about objects is automatically extracted from databases in Oracle (used by the technical staff for resource management) and cartographic information (geography, weather forecasts) is also extracted from Arc Info. This last input is specially important since staff is more skilled with this type of software and even the definition of planning problems (the description of a crisis episode or the situation assessment) has been adapted to be

[2] http://protege.stanford.edu

introduced by this means.

However, the knowledge stored in the ontology server is not directly recognizable by the planner module. It only recognizes problems described in the Planning Domain Description Language PDDL [15]. In order to do that, a PDDL gateway has been designed to translate the knowledge in the ontology into PDDL format as will be explained later.

2.2 The planning server

The planning module is able to analyze the knowledge stored in the ontology server and to design (or redesign) an attack plan. It is currently under development but it is an extension of two previously built systems of the same authors [7, 8] with the following features

- It is a hierarchical HTN planner[14, 17] that follows a top-down procedure for the extraction of plans (first obtains a general plan, and then it refines it into a more specific plan, and so on until it obtains a grounded plan). This top-down procedure is adapted to the own hierarchical structure of the ontology so both modules are closely related.
- It allows to represent and handle uncertain temporal knowledge [9] and soft temporal constraints like deadlines, execution times and durations.
- It is a planning server, that is, it has been designed like a plug and play module of the architecture so that it may be either easily used as a standalone planning service through the XML-RPC protocol or easily updated just by replacing the current planning algorithm by any PDDL compliant planning algorithm.
- The output of the planning module is a temporal attack plan, that is, a chronologically ordered sequence of actions where every action has a time stamp that refers to the time at which it should be executed.
- This temporal attack plan might be translated into a fire simulator called CARDIN [5] and submitted to the evaluation of technical staff.

2.3 The Monitor module

Once this temporal plan has been accepted for execution, it is supervised (observed) in real time by the monitor module that tracks all the changes produced by the execution of the actions in the plan, the time they take to execute and it updates the ontology accordingly so that these changes are publicly available for any query at any time. It is also able to detect any perturbation in its execution [12] and, following the loopback of Figure 1, it asks for a plan repairing (replanning) and revision procedure. This is one of the most important features of the system as it allows to respond to the uncertainty in the execution of a plan in a highly dynamic world. This error recovering procedure allows two types of repairs to the plan being executed [12]. On the one hand, it allows autonomous repairs without any intervention of the technical staff and on the other hand, it allows the technical staff to interactively redesign a new plan for the newly detected situation by giving advice to the planner on how to solve some goals.

This distributed architecture positions SIADEX as a collaborative working environment [11] where the knowledge is shared and exchanged between several heterogeneous entities, either human (technical staff) or programs (the planning server and the monitor) thanks to the services provided by the web server and the ontology server, providing several valuable advantages:

- Ubiquity. It currently supports full user mobility since the access to the web server is platform independent and technical staff may access either from their offices or from the countryside, either with a laptop or a PDA with a GPRS connection.
- Context awareness. SIADEX is sensible to the fire scenario in many different ways since it is able to represent many contextual information that can influence the reasoning process of the planning module. For example, geographical conditions of the environment, weather forecast, exact position of resources given by GPS devices, etc, may be obtained from technical staff, seamlessly introduced in the ontology server and used by the planning module.
- Service oriented architecture. The modularity of the architecture, as shown in Figure 2, allows for the interoperability of independent and heterogeneous processes and their communication through standard protocols like XML-RPC[20] and PDDL [15] in a plug a play fashion where every server might be easily replaced or updated.
- User friendliness. The centralization of the interface through the web server allows technical staff to interact with the system with a low training period. All the knowledge available in the system (classes, instances, actions or plans) are easily handled in a web browsing fashion.

The following sections give a more detailed overview of some of these modules.

3 Details of the ontology

As said before, BACAREX is an ontology of planning objects and activities related to the forest fighting plan in the Andalusian regional government that has been designed in Protégé 2000 with more than 130 classes and more than 2000 instances only for planning objects and without taking into account the representation of activities. The slots stored for every object are both operational (needed by the own reasoning process of the planner, say the geographic coordinates of the object) and informational (not needed by the planner but that may be required by the technical staff during the development of an episode, say the radio channel of the responsible of a sector). The most relevant features of this ontology are described in the following.

3.1 The taxonomy of domain objects

The most important part of the ontology is devoted to the fire fighting resources, which are modeled either as material resources or human resources (Figure 3). Material resources may be facilities like operation bases, airports, etc and they represent static objects from the point of view of the planning process since most of their attributes will remain unchanged although they are very important like for example, the geographic position of an airport, the availability of refueling facilities, etc. The remaining resources, either vehicles or human workers, are very dynamic since many of their attributes change rapidly during the execution of plans, for example their geographic position, their state of availability, the work they are carrying out, etc.

3.2 Legal information about the use of resources

Legal issues about the use of resources are a very important feature that the planning module has to take into account if one wants the plans to be of practical use. These legal issues refer mainly to temporal constraints that must be met and that are fixed by the law or any type of contracting agreement:

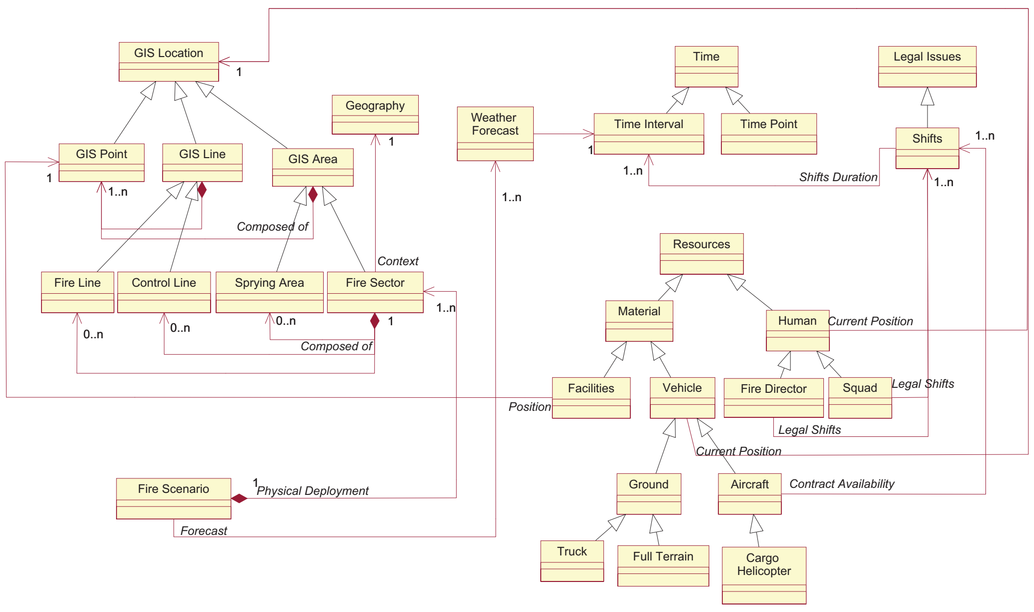


Figure 3. A UML representation of a simplified part of the ontology of SIADEX

- The maximum duration of the shifts of the squads or the fire directors (*Legal Shifts* in Figure 3)
- The periods of availability of resources since the use of some planes or helicopters is subject to contract and they are only available during some periods (*Contract Availability* in Figure 3).

3.3 The fire scenario and situation assessment

Contrary to instances of resources, that remain from one episode to other, instances of fire scenarios are volatile, that is, they are created at the beginning of a fire incident, but they disapear as the attack plan evolves and at the end of the episode. Once a forest fire has been detected, the technical staff proceed by the following steps. Firstly, they gather all the information of the environment (geography -orientation of the terrain, slope-, forest fuel, weather forecasts -temperature, humidity, wind speed and direction-) and starts a fire simulation [5] to predict the behaviour of the fire.

After the simulation, they define the operational units of the forest fire: the sectors. These are relevant areas of the environment where the attack will be focused. Every fire scenario may have several sectors and every sector may be composed into operational targets like fire lines, control lines and spraying areas. Instances of operational targets are created very dynamically and they might be modified after every revision cycle (Figure 1). The fire director assigns a numeric value of threat intensity and a set of tasks to be accomplished in every operational target. The set of tasks determine the type of resources that will be needed at every operational target and the intensity of the threat will be used by the planner to size the number of resources to be assigned to that target (the larger the intensity the larger the number of resources) and the set of tasks. Therefore, the state of resources and the definition of the fire scenario constitute the starting point of the planning module.

3.4 Tasks and operating procedures

Knowledge about tasks and fighting protocols (operating procedures) are represented in a HTN-style [14, 17], that is, as a hierarchy of compound tasks, that may be decomposed into simpler tasks, and primitive tasks, that are non decomposable like that shown in Figure 4. Higher level compound tasks represent strategic activities and low level primitive tasks represent actions to be performed by any of the resources. This part of the knowledge base is the most important one and the most fragile with respect to the performance of the planner since it encodes the knowledge used by the planner about how to build plans.

3.5 Constraints

There are many types of constraints that affect to the knowledge stored in the ontology and that are taken into account by the planner module. The following presents some of the most relevant constraints.

Knowledge constraints

Some constraints have been added to dynamically validate the knowledge stored in the ontology and for an early detection of inconsistencies (it must be taken into account that the ontology may be modified by the monitoring module during the execution of the plan but also by the technical staff to introduce or modify part of the knowledge about resources or the episode and this manual process might introduce some inconsistencies in the knowledge base that

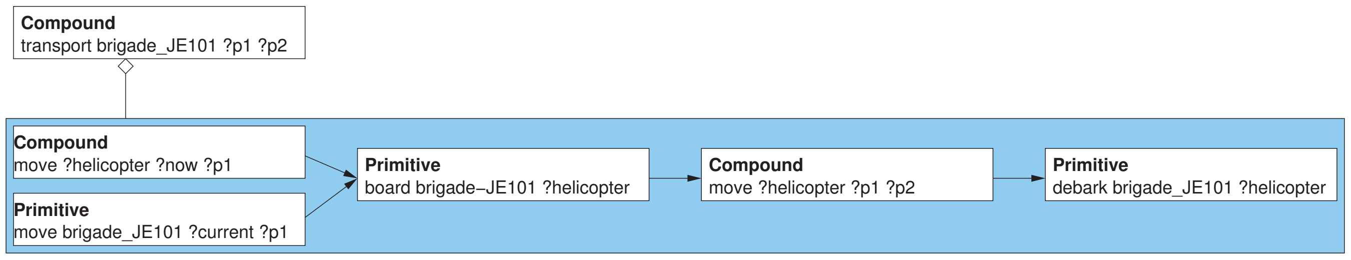

Compound
transport brigade_JE101 ?p1 ?p2
Compound
move ?helicopter ?now ?p1
Primitive
move brigade_JE101 ?current ?p1
Primitive
board brigade–JE101 ?helicopter
Compound
move ?helicopter ?p1 ?p2
Primitive
debark brigade_JE101 ?helicopter

Figure 4. Compound tasks, decompositions and primitive tasks. The compound task to transport the brigade `brigade_JE101` from point `p1` to point `p2` is decomposed into simpler tasks: (1) move an available helicopter and the brigade to point `p1`, once there, board the brigade, move the helicopter to point `p2` and debark the brigade into point `p2`. The action to board the brigade is primitive and it may not be decomposed further. The action to move the brigade to the meeting point is compound and might be decomposed further since the brigade might need to find a full terrain vehicle to get to the meeting point.

could produce the planner to fail or to obtain non-sense plans). These constraints have been designed by using the Protégé PAL Constraints Language and are checked offline by the development team[3] like the one shown in Figure 5.

```
(forall ?pt (and (> (UTM_Zone ?pt) 28)
                 (< (UTM_Zone ?pt) 31)
                 (> (UTM_X ?pt) 0)
                 (> (UTM_Y ?pt) 0)))
```

Figure 5. A PAL constraint in Protégé about the coordinates of objects: every objects must be situated between Zones 29 and 30 (i.e., the region of Andalusia) and coordinates X and Y must be defined and greater than zero.

Temporal constraints

These are the most important type of constraints and they mainly influence the ordering between the actions in a plan. The first source of temporal constraints is related to the duration of actions, and therefore, to the final schedule of the plan. Although the details are explained in [8] it may be summarized as follows. Every primitive action a has a start point $start(a)$ and an end point $end(a)$. All the effects caused by a occur somewhere between both points at different time points

$$\forall e \in effects(a), \quad start(a) \leq start(a) + delay(e) \leq end(a)$$

This is represented in the ontology for every primitive task (that is tasks that are not decomposable). Let us suppose that the effect e of action a is used to satisfy a requirement of a primitive task b which has to be executed necessarily at time $start(b)$. Then, the execution of action a is restricted to be

$$start(a) \leq start(b) - delay(e)$$

The representation of these constraints in the ontology is the basis of temporal constraint posting and propagation in the planning module of SIADEX. Other sources of temporal constraints that also affect the planning process is the use of numeric variables. Let say that a vehicle v has a cruise speed $cruise(v)$, then the task $a \equiv move(v, p1, p2)$ to move the vehicle v from point $p1$ to point $p2$ is constrained to be

$$end(a) - start(a) = \frac{distance(p1, p2)}{cruise(v)}$$

[3] PAL Constraints may be checked dynamically from the user interface, but this feature has not been implemented yet.

Other temporal constraints like legal duration of shifts of squads or the contract availability of aircrafts are easily represented in the ontology like intervals associated to a certain condition [4] and handled accordingly by the planning module

Constraints on operating procedures

The capabilities of the resources is not encoded in the own resources, but in the ontology of tasks and operating procedures by incorporating typed arguments to their representation like in the PDDL formalism [15]. Let us consider for example a compound task `(drive ?v - vehicle ?p1 ?p2 - GIS_Point)` to drive a vehicle `?v` from one point to another. This means that the movement task might be carried out naturally by any instance of the class `vehicle`. However one might also have a compound task like `(fast_drive ?v - aircraft ?p1 ?p2 - GIS_Point)` where the movement can only be done by an aircraft and the remaining vehicles are immediately discarded to perform this task.

3.6 Representation of uncertainty

The application of AI techniques to crisis management, like to many real world applications as well, requires an explicit consideration of uncertainty since knowledge tends to be faulty (due to incompleteness, imprecision or unpredictability) or there may be exogenous events out of the control of the agents involved in the problem. From the knowledge representation point of view, uncertainty is considered in the following terms.

- Temporal uncertainty. It is clear that in a real world problem, most part of temporal knowledge may not be perfectly known like the duration of an action, the arrival times of a vehicle or deadlines for satisfying a certain goal. In our ontology, every temporal reference like deadlines or makespans are stored as an interval $[t1, t2]$ (Figure 3) so that precise references may be stored like “exactly t time units” $[t, t]$ but also some uncertain references like “between $t1$ and $t2$ time units” $[t1, t2]$ or “more than t time units” $[t, +\infty)$. Thanks to the use of Temporal Constraint Networks [13, 8] we are able to handle and propagate successfully uncertain temporal constraints like these shown before.
- Resource uncertainty. The use of an ontology structured in classes and subclasses of entities allow us to easily abstract the usage of resources in the description of actions and protocols without having to specify completely how that resource should be used. For example, let us consider the part of an ontology devoted to vehicles that is shown in Figure 3. A transportation request

[4] These constraints are named *timed initial conditions* in PDDL 2.2.

for the `brigade_JE101` from `point1` to `point2` by using `truck-101` might be represented like this activity

```
(move brigade_JE101 truck101 point1 point2)
```

but if we don't know exactly the resource that will be used for this activity, then it might be represented as

```
(move brigade_JE101 ?v - vehicle point1 point2)
```

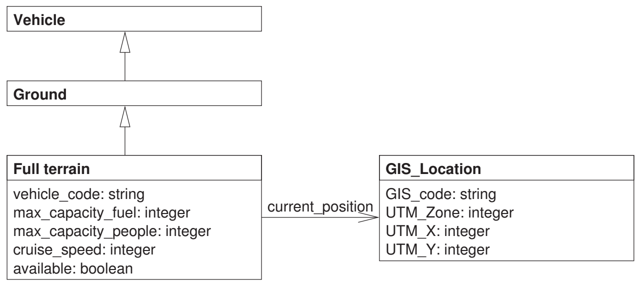
where the variable `?v` represents any instance of the class `vehicle`, that is, any available vehicle able to transport people.

- Location uncertainty. In optimal conditions, every mobile resource would be endowed with a GPS device so that its position would be perfectly known.
  In real world this is not always possible so that we must maintain the track of the location of a resource as precise as we can for safety and efficiency reasons and not always is possible to locate a resource in a precise point. In order to do that, the current positions for some objects in the ontology are described under the class `GIS Location` as shown in Figure 3 so that we allow to specify literals like `(current_position brigade_JE101 ?position - GIS_Location)` where `?position` is an instance of class `GIS Location`, either a point, a line or an area. In the case that the position of a resource is perfectly known and fixed, like instances of class facility, literals of the form `(current_position ?facility ?position - GIS_Point)` are used.

4 The planner

The planning module is still under development so we may not show performance or efficiency data here but it is a planner based on the HTN paradigm [14] over the ideas previously developed in our non-hierarchical temporal planner MACHINE [8] and our hybrid hierarchical planer HYBIS [7]. An HTN planning problem, like most planning problems, is stated in terms of three components: the initial state of the world, the set of available actions, that is also known as the domain of the problem, and the goal.

- The initial state is usually expressed as a conjunction of literals that represent the set of facts that are known to be true. It may be obtained by querying the ontology about the main slots of the instances of classes like facilities, human resources, vehicles, water points, etc. The representation of the initial state is based on the closed world assumption.
- The domain is also extracted from the ontology, from the knowledge stored in the actions and protocols section. It must be said that the planner is compliant with the syntax and semantics of PDDL, the planning domain description language [15] so both the initial state and the domain must be expressed in this formalism. In order to do that, a PDDL gateway has been designed so that, at the beginning of a planning episode, this gateway iterates over the whole ontology and translate the content of the main instances into PDDL in a process whose main features are the following ones:
  - Predicates are easily translated since every slot of the ontology whose content might be relevant for the planning process has a special slot that contains a template on how to translate the content of the slot into a PDDL literal. Binary slots of the form `Instance.slot=value` are easily translated into PDDL literals of the form `(slot instance value)`. Non binary predicates or slots that represent references to other instances have a similar translation process, and even some of the slots may produce multiple literals (see Figure 6).



```
(current_position vehicle_code GIS_code) and
(coordinates GIS_code UTM_Zone UTM_X UTM_Y)
```

Figure 6. Translation of some references in the ontology

  - The classes hierarchy is also translated as a hierarchy of types in the PDDL domain description.
  - Slots that contain numerical values that may change during a planning episode, mainly numerical resources like the level of fuel of a vehicle, are declared like fluents in the PDDL domain, so that the use of arithmetic operations and functions are permitted over them and they allow the planer to reason about the use of resources.
  - The temporal knowledge related to durations and delays of activities, like subgoals with deadlines, the maximum makespan allowed for a plan, durations of actions or any other temporal constraint defined between actions, is expressed in the formalism described in [8], that extends the expressiveness of the level 3 of PDDL [15] devoted to durative planning actions.
- The goal is defined as a situation assessment of the episode made up of instances of the ontology that describe the desires of the technical staff. Then, the goal of a problem is defined in the following terms (by extending the PDDL representation of goals):
  - Geographical deployment of the episode (GIS information): situation of the episode, main fire lines, main focuses, water discharging areas, defense lines (grouped by sectors each of which has a fighting director that is responsible of it), command area and waiting areas.
  - All the items related to fire fighting activity have a slot that defines its intensity, that is, a numeric evaluation of the power of resources that should be devoted to it and that is fixed by hand by the fire fighting director. From the intensities, the planner may pre-select several combinations of resources, that are submitted to the fight director who finally selects one of them. Later, the planner will select a specific set of instances of resources that fits within the selected combination of resources.
  - The fire director also assigns different tasks to be carried out at each item with fire fighting activity (among the available tasks in the ontology) and, since every task is preconditioned with the type of resource that is able to develop it, the planner automatically assigns resources to tasks.

Finally, thanks to the expressive power of using Temporal Constraint Networks as the underlying formalism to represent temporal knowledge, the planner is able to obtain approximate temporal plans, that is plans whose timeline is flexible and may be scheduled in several different ways to adapt to unexpected delays, but this is discussed in the following section (for more details, see [8, 9]).

5 Monitoring and Replanning

Up to now, we have described a batch process that starts with the knowledge stored in the ontology server and ends with the raising of one (or more) approximate temporal plans ready for execution. This section briefly describes the modules that allow the closing loopback shown in Figure 1 and a revision process of plans. These modules are the monitoring process and the replanning process (the role of the replanner is also played by the same planning module since it is an incremental planner able to plan an replan over the same episode [12]) whose interaction is sketched in Figure 7 and described below.

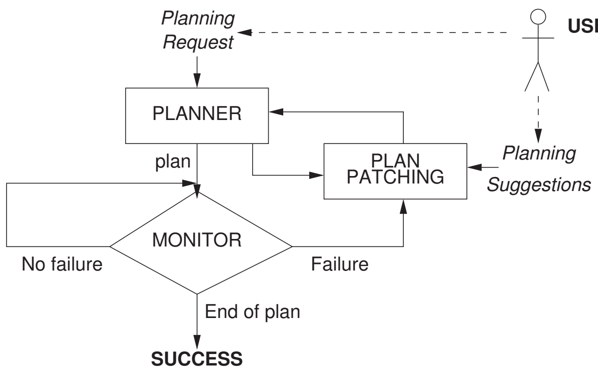


Figure 7. The monitoring user-centered replanning processes

The monitoring algorithm [8, 12] is a real time algorithm that follows the execution of the temporal plan at the highest level of detail since temporal plans still represent the time at which every effect of every action is achieved. It checks that everything executes as predicted, otherwise, it may detect and, in some cases repair, some of the problems[5]. The type of problems and their possible solutions are the following ones:

- Unexpected delays. These are detected when an action exceeds the time predicted to achieve one of its effects and there may be three different situations.
  - Local delays. They are delays that only affect locally to an isolated branch of the temporal plan without affecting the remaining actions. In these cases, a new reschedule may be found only for the actions of the affected branch leaving the remaining schedule unaltered.
  - Global delays. They are more important since they might affect all of the remaining actions and deadline goals or makespans might also be affected and hence, a whole new reschedule might be needed.
  - Infeasible delays. When an action a_j has been delayed beyond its limits, then no reschedule is possible and the possibility of replanning must be considered.
- Execution failures. These situations mean a real fail of execution of an action and they are very serious since, as a consequence of the failure, the cause/effect relationships between actions could have been damaged and the plan could fail to achieve its goal. Therefore, they are situations that necessarily imply some replanning decisions to be taken and that may be as easy as re-executing the failed action or as complex a re-designing a whole branch of the plan. These execution failures may be due to one of the following reasons:
  - Missing condition. A condition that was previously achieved by the execution of a previous action has disapeared. For example, a defense line that was open by a bulldozer to protect an area but that the fire has jumped over it and now the protected area is threatened.
  - Missing effect. An effect of an action under execution would never be achieved. For example, when an aircraft breaks down and it is not able to drop a discharge of water that had been previously scheduled.
  - Unexpected condition. A condition that was not previously known suddenly raises. For example an unforeseen increase in the speed of the wind that impedes the flight of helicopters.

  In any of these cases, the revision and redesign of the failed branch is carried out in a close interaction between the technical staff and the planning module in a very interesting "user centered" plan patching episode [12] like that outlined in Figure 7. Its main features are the following ones.
  On the one hand, this episode allows the interaction with the user, that is, during this plan patching episode, the user may edit the plan and either delete and suggest conditions or actions to reflect his strategy to resume the execution of the plan or even define a new assessment of the situation by deleting goals and introducing new goals for the planner. This is a very important point since in critic situations, where there may be lives in danger, a completely automated process is not very realistic and the skills of human operators could not be substituted. After the patching of the failed plan, the own planning algorithm is able to regenerate a new plan adapted to these changes of the user and then submitted again to the technical staff for consideration until an appropriate solution is found and scheduled for execution, restarting the loop of Figure 1.
  On the other hand, this regeneration process is strongly based on local changes made on the failed plan, so that no radical changes are introduced and only the failed branches are redesigned, leaving the remaining of the plan unaltered. This issue is very important since otherwise, a global redesign of the plan could produce dramatic changes on the resources and their tasks and a chaotic migration from the older plan to the new one that would be unrealistic.

In summary, after this replanning episode, the complete cycle of the crisis management shown in Figure 1 is completely covered by the architecture of SIADEX.

6 Related work

Regarding general crisis situations SIPE [3] has been used for obtaining plans of attack for oil spill threats in the sea. Although its architecture is very similar to SIADEX, its main drawback is that it requires end users to have a deep knowledge of planning techniques either to interact or to provide knowledge for the system.

Another system used in civil crises was that described in [4] where hybrid hierarchical techniques were developed to obtain plans of re-

[5] Clearly, the user may also interrupt the execution of the plan at any moment.

sponse for floods in Germany, but as far as authors know, it does not have monitoring or replanning capabilities, what impedes severely its use in real environments.

Specifically in the field of forest fire fighting there are several approaches in the literature. PHOENIX [10] (1989-1993) is a hierarchical multi-agent system that incorporate a distributed, adaptative and real-time planner, designed to control simulated forest fires. It is based on a skeletal plan paradigm so there is no generative capability, instead, plans are firstly extracted from a library of skeletal plans and refined for the current episode. The aim of PHOENIX is not to solve fire extinction problems in real domains, nor assist an expert in such task, but use the domain simulation for making proofs. Because of that it has significant weaknesses like little interaction with user and simple replanning and monitoring techniques.

CHARADE [1] (1992-1995) is an interactive case-based planning system. Its aim is to assist an expert in situation assessment and first attack to forest fire. The weaknesses of this system are the lack of a deliberative process to modify and refine the plan, and the absence of replanning and monitoring techniques, so that the only generative capability resides is restricted to the user.

The CARICA [2] (1995-1997) project tried to adapt the CHARADE system to the experts requirements, who wanted a tool for learning to plan fire forest attacks. Thus the initial application was reoriented to finally become an environment for testing and learning to extinct forest fires.

Another work is that presented in [19] where authors explore the possibility of having a multiagent system based on Case Based Reasoning and Constraint Reasoning that allow the retrieval of predefined skeletons of plans and their adaptation to the current situation.

The main drawbacks of these approaches are:

- The lack of deliberative planning techniques able to flexibly generate new plans for a situation without the requirement of having a predefined skeleton or library of cases of plans previously stored.
- No monitoring of plan execution or replanning techniques to allow flexibility and responsiveness in the execution of the plan.
- They consider user interaction only at edition level, not allowing users and the planner to collaborate in the resolution of the same problem.
- An insufficient treatment of the uncertainty about the temporal and spatial knowledge or the use of resources.

7 Final Remarks

In summary, this paper has outlined the architecture of SIADEX, an integrated system in development under a research contract with the Andalusian Regional Ministry of Environment. It is based on several AI techniques and it is intended to serve as an intelligent decision support system for the design of forest fire fighting plans. The main advantages of SIADEX are straightforward. Firstly, it has been developed as an easy to use tool for the technical staff of forest fire fighting, who are not experts in AI techniques, but that may access the system painlessly through a web service. Secondly, it is interactive, so that it allows the user to suggest part of the solution or to suggest changes in the design of a plan that would be recognized by the planner. Thirdly, it is flexible both in the representation of knowledge and in its use, so that many types of uncertainty (temporal, spatial and about resources) are explicitly dealt with at different stages of the problem, from knowledge representation to plan execution.

This project has just entered its second year of development out of three years of duration, most part of the architecture (the ontology server, most part of the interface and the underlying communication technology, and the monitor module have already been developed, but the planner is still in development) and it is obtaining a good feedback from the technical staff. We intend to make a first demonstration of the system by the autumn of 2004.

REFERENCES

[1] P. Avesani, A. Perini, and F. Ricci, ‘Interactive case-based planning for forest fire management’, *Applied Intelligence*, **13**(1), 41–57, (2000).

[2] P. Avesani, A. Perinni, and F. Ricci, ‘CBET: a case-based exploration tool’, in *Fifth Congress of the Italian Association for A.I.*, (1997).

[3] M. Bienkowski, ‘Demonstrating the Operational Feasibility of New Technologies: The ARPI IFDs’, *IEEE Expert*, **10**(1), 27–33, (1995).

[4] S. Biundo and B. Schattenberg, ‘From abstract crisis to concrete relief - a preliminary report on combining state abstraction and htn planning’, in *6th European Conference on Planning (ECP-01)*, (2001).

[5] D. Caballero, J. Martinez-Millan, J. Martos, and S. Vignote, ‘Cardin 3.0, a model for forest fire spread and fire fighting simulation’, in *2st International Conference on Forest Fire Research Vol.1*, (1994).

[6] L. Castillo, J. Fdez-Olivares, and A. González, ‘Mixing expresiveness and efficiency in a manufacturing planner’, *Journal of Experimental and Theoretical Artificial Intelligence*, **13**, 141–162, (2001).

[7] L. Castillo, J. Fdez-Olivares, and A. González, ‘On the adequacy of hierarchical planning characteristics for real world problem solving’, in *Proc. of VI European Conference of Planning*, (2001).

[8] L. Castillo, J. Fdez-Olivares, and A. González, ‘A temporal constraint network based temporal planner’, in *Workshop of the UK Planning and Scheduling Special Interest Group, PLANSIG 2002*, pp. 99–109, (2002).

[9] L. Castillo, J. Fdez-Olivares, and A. González, ‘Some issues on the representation and explpoitation of imprecise temporal knowledge in an AI planner’, in *Knowledge-Based Intelligent Information and Engineering Systems*, Lecture Notes in Artificial Intelligence, LNAI-2774, pp. 1321–1328. Springer-Verlag, (2003).

[10] P.R. Cohen, M.L. Greenberg, D.M. Hart, and A.E. Howe, ‘Trial by fire: understanding the design requirements for agents in complex environments’, *AI Magazine*, **10**(3), 32–48, (1989).

[11] European Commission, ‘Next generation collaborative workig environments 2005-2010. report of the expert group Collaboration@WORK’. Brussels, 2004.

[12] Marc de la Asunción, Luis Castillo, Juan Fernández-Olivares, Oscar García-Pérez, Antonio González, and Francisco Palao, ‘Local (human-centered) replanning in the siadex framework’, in *Conference of the Spanish Association for Artificial Intelligence, II Workshop on Planning, Scheduling and Temporal Reasoning*, eds., L. Castillo and M.A. Salido, pp. 79–88, (2003).

[13] R. Dechter, I. Meiri, and J. Pearl, ‘Temporal constraint networks’, *Artificial Intelligence*, **49**, 61–95, (1991).

[14] K. Erol, J. Hendler, and D. Nau, ‘UMCP: A sound and complete procedure for hierarchical task-network planning’, in *AIPS-94*, (1994).

[15] D. Long and M. Fox, ‘PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains’, *Journal of Artificial Intelligence Research*, **20**, 61–124, (2003).

[16] K. L. Myers, ‘CPEF: A continuous planning and execution framework’, *AI Magazine*, **20**(4), 63–69, (1999).

[17] D. Nau, T.C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman, ‘SHOP2: An HTN Planning System’, *Journal of Artificial Intelligence Research*, **20**, 379–404, (2003).

[18] H. Mu noz Avila, D. W. Aha, L. Breslow, and D. Nau, ‘HICAP: An interactive case-based planning architecture and its application to noncombatant evacuation operations’, in *Ninth Conference on Innovative Applications of Artificial Intelligence*, pp. 879–885. AAAI Press, (1999).

[19] E. Rollon, D. Isern, A. Agostini, and U. Cortés, ‘Towards the distributed management of emergencies: forest fires case study’, in *IJCAI Workshop on Environmental Decision Support Systems*, (2003).

[20] UserLand Software Inc., ‘XML-RPC: Remote Procedure Call’. http://www.xmlrpc.com, 2004.

[21] D. E. Wilkins and R. V. Desimone, ‘Applying an AI plannet to military operations planning’, in *Intelligent Scheduling*, eds., M. Zweben and M. S. Fox, Morgan Kaufmann, (1994).

Handling fuzzy temporal constraints in a planning environment[*]

Marc de la Asunción Luis Castillo
Juan Fernández-Olivares Oscar García-Pérez

Antonio González

Francisco Palao

Contact:
Luis Castillo
Dpto. Ciencias de la Computación e Inteligencia Artificial
ETSI Informática, Universidad de Granada
18071-Granada, Spain
email:siadexwww@decsai.ugr.es

September 19, 2005

[*]This work has been partially supported under the project MCyT TIC2002-04146-C05-2 and the contract NET033957 with the Andalusian Regional Government

Abstract

An interleaved integration of the planning and scheduling process is presented with the idea of including soft temporal constraints in a partial order planner that is being used as the core module of an intelligent decision support system for the design forest fire fighting plans. These soft temporal constraints have been defined through fuzzy sets. This representation allows us a flexible representation and handling of temporal information. The scheduler model consists of a fuzzy temporal constraints network whose main goal is the consistency checking of the network associated to each partial order plan. Moreover, we present a model of estimating this consistency, and show the monitoring and rescheduling capabilities of the system. The resulting approach is able to tackle problems with ill defined knowledge, to obtain plans that are approximately consistent and to adapt the execution of plans to unexpected delays.

1 Introduction

Artificial Intelligence Planning techniques have shown very useful in the resolution of different problems related to logistics and workflow domains, just to cite a few (Biundo et al., 2003). However, many real world problems require the inclusion of an implicit representation and handling of time for the plans to be correctly executed so that some scheduling techniques must be imported into a planning framework. But on the other hand, many realistic scheduling problems still require the inclusion of some planning techniques to work properly (Bartk and Mecl, 2003). The work presented in this paper is based on one of these interdisciplinary domains devoted to mobile workforce management in crisis situations: the SIADEX project (de la Asunción et al., 2003; de la Asunción et al., 2005). Although the techniques being developed in this project are devoted to the assisted design of forest fire fighting plans, they are general enough so as to be of application to other crisis management domains either in military frameworks (operations planning (Wilkins and Desimone, 1994), air campaign design (Myers, 1999) or noncombatant evacuation operations (noz Avila et al., 1999)) or in civil frameworks (oil spills (Bienkowski, 1995), floods (Biundo and Schattenberg, 2001) or forest fires (Avesani et al., 2000; Cohen et al., 1989)). In these real domains one must consider many different techniques but this paper focuses on two of them that are particularly relevant.

On the one hand, there is a need to integrate planning and scheduling techniques so that intelligent decision support systems are able not only to reason about action and changes, but also about time and resources.

On the other hand, in order to solve this kind of problems it is necessary to consider a flexible representation and handling of time. Temporal planners (Bacchus and Ady, 2001; Do and Kambhampati, 2001; Haslum and Geffner, 2001; Smith and Weld, 1999) use a rather rigid notion of time in the sense that time and durations are assigned and compared taking into account only strict equality. Hence these temporal planners may only be applied to problems where temporal knowledge is sharply defined and resulting temporal plans have to be executed exactly in the time line defined by the planner since, otherwise, the plan will fail. Let us suppose that in a forest fire episode, two fire fighting brigades are expected to arrive to the waiting area exactly at 10:00 am, but they arrive at 10:05 am and 10:10 am respectively. Is the plan still valid?

In many real applications the temporal bounds of activities, like deadlines or durations, are implicitly considered as flexible bounds at some extent, that is, they are not a rigid matter because of incomplete or vague knowledge or preferences, unpredictable behaviors or execution errors. Therefore this paper presents Machine^{TF}, an extension of a previous work of the authors (Castillo et al., 2001), based both on the use of an interleaved integration of the planning and scheduling processes and on the representation and handling of soft temporal constraints. For example, when one makes a plan in order to transport by truck a fire fighting brigade to a forest fire, one doesn't know the exact time needed to activate the members of the brigade, nor the exact time that the drive takes to complete the route by truck, nor the exact time of arrival at the waiting area. Instead, all these temporal constraints are roughly defined, but these plans can easily be designed by an expert. So this paper explains how to represent these soft temporal constraints and how to include successfully soft constraint handling procedures into a planning framework.

Soft constraints have been defined by means of fuzzy sets (D. Dubois and Prade, 2003), a very expressive formalism to represent imprecision and preference of the user when he/she uses soft temporal constraints. Hence, the basic representation of the temporal knowledge will be a Fuzzy Temporal Constraint Network (FTCN (Vila and Godo, 1994b; Marín et al., 1997)). This also allows the planner to build plans based on vague temporal constraints of the form: "the action of driving the truck will take about 100 time units" or "action *a* must be delayed more or less between 20 and 30 time units after action *b*". This type of "soft constraints" provide several advantages:

- It is possible to model domains in which temporal knowledge is vaguely defined and to obtain temporal plans of practical use for these domains.
- It becomes a very interesting approach in time critical problems, like crisis management, space missions or complex scheduling problems, where execution times are very tight and perhaps a solution that completely meets the temporal constraints is not possible and, instead, a relaxation of the constraints is feasible and an approximate solution could be found.
- A temporal plan built on top of a FTCN is not a single solution, but a class of possible solutions, i.e., a set of solutions for which the set of actions is always the same, but their temporal interleaving might slightly change from each other. In fact, slightly different temporal schedules of the same set of actions might be obtained by a solution extraction procedure as explained in the paper. This flexible representation of plans would allow to adapt to possible delays during the execution of the plan without the need to re-plan a new sequence: let us suppose that a sequence of actions has been designed and that it has already started its execution, if an unexpected delay occurs, it is still possible to obtain a new time line for the remaining of the plan just by changing to an alternative schedule able to adapt to the existing delay.

The basic process of MACHINE^{TF}is a partial order model in which at every step the plan being designed by the planner is passed to the scheduler module in order to determine its consistency. An inconsistent plan is immediately detected, even prior to complete all its actions, and rejected. In the case of consistent plans, since the consistency in a FTCN is a degree matter the degree of temporal consistency is used by the heuristic of the partial order planner in order to search plans appropriate to the preferences of the user. The problem of an efficient estimation of the consistency is addressed through the calculus of optimistic and pessimistic bounds of the consistency.

The paper is organized as follows. It does not discuss the domain of crisis management (forest fires) in detail, instead, it is focused on the techniques underlying the approach to integrate planning and scheduling and to represent and handle soft temporal constraints. For a more general description of SIADEX, see (de la Asunción et al., 2003; de la Asunción et al., 2005). After giving a description of the integration of planning and scheduling we have used in section 2, we first describe in section 3 how the fuzzy sets are used as a model of knowledge representation, and then turn in section 4 to the description of the scheduler process based on the use of FTCN and the problem of estimation of the consistency for these networks is addressed. Section 5 shows the monitoring and rescheduling capacities of MACHINE^{TF}and an example of the process. Section 6 presents a monitoring example and section 7 concludes.

2 Integration of planning and scheduling

The main goal of this work is the representation and handling of soft temporal constraints in a partial order planner. Moreover, we are interested in the application of the model to real problems, like the assistance to experts in the task of designing forest fire extinction plans described above in the context of the SIADEX project. Therefore, a flexible enough handling of time in the complete process is needed.

Thus, in order to successfully deal with temporal constraints, the planning algorithm should be extended to cope with some temporal reasoning capabilities that traditionally belong to scheduling systems like constraint posting and propagation, consistency checking or solution extraction. In the literature, there are mainly two families of architectures for the integration of planning and scheduling. On the one hand (Figure 1) a planner and a scheduler execute one after the other in a waterfall model. This model is easier to implement but has many drawbacks mainly related to backtracking points between both isolated systems.

Instead, we have followed an interleaved integration (Figure 2) where part of the decisions of the scheduler have been introduced intimately into the planning engine.

The main idea of this integration model is to allow the collaboration between the planner and the scheduler in order to search the best plan for the problem. After every step of the planning engine the current plan under construction is sent to the scheduler. It checks the temporal consistency of the constraints associated to this current plan, and this information is used by the planner to reject the plan, in the case of inconsistency, or to follow with the refinement of this plan, in the case of a consistent plan. Therefore, the integration between planner and scheduler overcomes the backtracking problems of the former approach so that, for example, an inconsistency of the constraints makes the planner to backtrack immediately.

Another important point is the inclusion in the partial order planner of temporal information in the way of soft temporal constraints. Some examples of these constraints are the statement of a makespan, the duration of actions, the temporal distance between pair of actions or deadlines goals.

In the next section we propose an extension of the knowledge representation of our planner MACHINE (Castillo et al., 2001), a partial order causal link based planner (Weld, 1994), able to deal with soft temporal constraints.

3 Extending the knowledge representation

The starting point to be addressed when extending a planning model to account for new knowledge is its model of actions, that is, the way the planning framework represent changes in the world. Other issues to be defined are the definition of problems and the definition of plans. In this case, we have extended the basic model of action of our planner MACHINE (Castillo et al., 2001) (basically the same than PDDL 2.1 (Long and Fox, 2003b)), to allow the representation of fuzzy temporal constraints.

3.1 Representing and handling fuzzy temporal constraints

Planning for real problems is nothing without a valid execution of the plan, and this often implies the need to deal with temporal constraints if one wants to obtain a realistic solution. There are several examples in the literature that have defined approaches to deal with these constraints (Do and Kambhampati, 2001; Haslum and Geffner, 2001; Smith and Weld, 1999), however, in many real problems, some (or many) of these constraints are not rigidly defined. This lack of rigidity appears in several different ways.

Let us consider that we have designed a plan to carry a fire fighting brigade from one location to another including transportation and loading and unloading their tools. Let us suppose that the maximum duration for the unloading operation is 60 time units and the whole makespan is 240 time units maximum. A rigid interpretation of temporal constraints would imply a chain of execution failures if the unloading of tools and material takes 61 time units and the makespan finally grows up to 243 time units, and also, in the case of more complex plans, it will surely raise important questions about the causal correctness of the remaining plan. In real life, and depending on acceptance criteria, this relative delay could be acceptable if nothing else can be done and the goal is finally achieved. Perhaps an interval based representation could be appropriate and we could accept that the duration of the unloading could be represented like $[60-\delta_1, 60+\delta_1]$ and the makespan like $[0, 240+\delta_2]$ where δ_1, δ_2 depend on acceptance criteria. This interval based representation has been used in the literature as a valid means to deal with temporal imprecision or temporal preference in planning systems (Laborie and Ghallab, 1995; Muscettola, 1994), however it presents some drawbacks that need to be clarified. Let us suppose in the previous example that $\delta_1 = 2$ and $\delta_2 = 4$. This would make the execution of the previous plan acceptable, but it would also allow the planner to accept a longer plan with a makespan of say 244.

However, this does not reflect exactly the desires of the user when he relaxed the restrictions. A relaxation of the temporal constraints is not only a widening of the bounds of the constraint, but also a distribution of his preference. In the example, he accepts a makespan between 240 and 244 but, and this is the most important, not all of them are equally preferred, the planner should give priority to plans whose makespan is closer or under 240 time units. At this point it is clear that an interval based representation is not able to represent this information and, therefore, it does not seem expressive enough to represent appropriately the semantics of a relaxation of the constraints in a planning and scheduling framework.

In order to solve this representation problem, we propose the use of fuzzy intervals to model the concept of soft temporal constraints in a planning and scheduling framework as a more expressive formalism able to represent the preferences of the user when he relaxes a temporal constraint, so that the initial constraint represents what the user *desires*, with the highest priority, and the relaxed constraint represents what the user *would admit* in the case that his desires cannot be satisfied and giving priority to the values closer to the initial satisfiability. A fuzzy interval, like that shown in Figure 3.a, is able to represent this information by means of its membership function μ, i.e., the degree of satisfaction of a temporal constraint over the time line (Dubois et al., 1993). Initial satisfiability is represented as the set of values t such that $\mu(t) = 1$, that is, the strict desires of the user. The soft constraint is represented as the set of values t such that $0 < \mu(t) < 1$, i.e., those values that are admissible and that are ordered in preference to the initial satisfiability, and finally, those values t such that $\mu(t) = 0$ are not admissible at all, that is, the user cannot admit these values in any case. Therefore, the preferences of the user regarding a relaxed and flexible temporal constraint C are perfectly represented by the following membership function

$$\forall t \in T, \mu_C(t) = \alpha$$

where $\alpha \in [0, 1]$ represents the degree of user satisfaction of the constraint C. This model of soft temporal constraints implies a certain degree of controllability of the duration of actions, since during their execution, the executive (human or computer) may be required to shorten or enlarge its duration to better fit the overall temporal constraints. However, the model is also subject to the occurrence of exogenous events that might modify the duration of an action. Exogenous events are not controllable, that is, they fall outside of the scope of the executive, but the approach described in this paper is able to monitor these exogenous changes and to suggest the executive to modify the duration of subsequent actions in order to accommodate to possible delays while maintaining consistency.

This idea of constraint relaxation is well known in many constraints satisfaction problems approaches (Dechter, 2003) and it is the main motivation of this paper for dealing with fuzzy sets. In particular, the remaining of this paper assumes the use of convex trapezoidal fuzzy subsets like the fuzzy subset F shown in Figure 3.b that will be noted as tuple with their respective points $F = (a, b, c, d)$. Obviously, this fuzzy notation may also be used to represent crisp subsets, i.e. subsets with no relaxation at all, for example, the tuple (b, b, c, c) represents the classic strict interval $[b, c]$.

In particular, the different types of fuzzy temporal constraints that will be used are shown in Figures 4, 5 and 6, where in all the cases the value δ encodes the admissibility limits for each constraint, and it may also be different for each side of the fuzzy set.

3.2 Actions, preconditions and effects

An action a in MACHINE^{TF} is represented taking into account the following issues.

- It is represented by means of two time points, namely $start(a)$ and $end(a)$.
- An action has effects, that is, changes in its environment and every effect takes some time to be achieved. This time may not be precisely known, so it is represented as the fuzzy temporal constraint "about t_f time units" (Figure 4) or "more or less between t_1 and t_2 time units" (Figure 5) that must be between $start(a)$ and $end(a)$. This knowledge represents a flexible constraint where the initial constraint has been relaxed through a parameter δ. Obviously, if the delay is strictly bounded and the user doesn't permit the relaxation then $\delta = 0$.
- An action has preconditions, that is, conditions that have to be made true by the effect of another action prior to its execution. Since effects take some time to be achieved this is one of the main source of (soft) temporal constraint posting between actions. Let us consider that the condition f of action a is satisfied by the effect f of action b that takes "about tf time units" to be achieved. Then there must be a (soft) temporal constraint that enforces $start(a)$ to be "more than t_f time units" after $start(b)$ (Figure 6).
- Actions have a duration that may be either unlimited (although it is calculated during the planning process) or bounded, that is, they may have a maximum duration (*maxbound*) allowed specified in terms of "less than *maxbound* time units" (Figure 6). All the actions have an minimum duration, that is, the duration required to obtain all its effects.

Then, the domain of a planning problem is the set of actions available to solve that problem.

Example 1 *This example, inspired in zeno problems of the international planning competition (Penberthy and Weld, 1994; Long and Fox, 2003a), presents a transport problem in a crisis situation. The problem, depicted in Figure 7 consists in carrying two fire fighting brigades from their respective bases (i.e., city-a and and city-b) to the crisis scenario (city-c). In order to do that, a helicopter able to transport one brigade must be used. This helicopter has its base at city-a and must pick up brigade-1 and brigade-2 and carry them to city-c. The helicopter may fly at two different speeds: a slow speed (named fly) that takes "more or less" 10000 time units to travel from city-a to city-c and a fast speed (named zoom) that takes "more or less" 7000 time units. In addition to* *this, brigade-1 must reach city-c before 7500 time units from the begining of the plan (fuzzy temporal reference (0,0,7500,7600)), but brigade-2 does not have any deadline.*

In this case, the domain representing all possible actions is represented in Figure 8. The operation of the helicopter is represented as a finite state automaton at the top of the figure, showing all possible states for the helicopter. The figure also shows the representation of actions fly and refuel with their respective preconditions and effects.

As can be seen, the helicopter must fly at its maximum speed to meet the given constraints to reach city-c since flying at the minimum speed would be too slow to meet the constraint imposed on the transportation of brigade-1. This is depicted in Figure 9 where the initial part of the plan devoted to carry brigade-1 is shown. There may be seen that the refuelling action may be executed in parallel with the boarding of the brigade and its equipment and also that the plan finally ends a bit earlier than required (7200, 7401, 7500, 7600), that is, the earliest ending time is 7401 time units although, in the case of an unexpected delay, it could be delayed up to 7500 time units with the highest possibility degree.

3.3 Problems

Problems are stated like a conjunction of subproblems (literals) that must be solved, so the planner must design a sequence of actions from the domain that achieve these subproblems. In the domain of Example 1 a possible goal would be `(at brigade-1 city-c)` to find a plan for a trip from `city-a` to `city-c`. However, a problem may also include soft temporal constraints.

- Deadlines. Some of the literals have to be achieved at a given time. Deadlines are expressed as a soft constraint in any of the forms shown in Figures 4, 5 or 6. For example the deadline goal

  `(at brigade-1 city-c) AT (11pm-`δ`,11pm,11pm,11pm+`δ`)`

  might be used to require `Brigade-1` to arrive in `city-c` “more or less at 11 pm”.

- Makespans. The total length of the plan may also be restricted by means of a soft temporal constraint like any of the ones shown in Figures 4, 5 or 6. An example could be to find a plan with ”approximately less than or equal to 7500 time units”.

3.4 Plans

Plans are a partially ordered sequence of actions and in Machine^{TF} they are deployed over a Fuzzy Simple Temporal Constraint Network.

Definition 1 *A FTCN $\mathcal{N} = \langle \mathcal{X}, \mathcal{C} \rangle$ is composed of a set of variables $\mathcal{X} = \{X_0, X_1, \ldots, X_{n+1}\}$ and a set of fuzzy binary temporal constraints defined between them $\mathcal{C} = \{C_{ij} |\ 0 \leq i, j \leq n+1\}$ (Marín et al., 1997).*

Every variable X_i is a crisp variable whose domain is the real time scale T. Variables X_0 and X_{n+1} are two dummy variables used to represent the beginning and the end of the network. Every fuzzy binary constraint C_{ij} restricts the possible relative values of X_i and X_j, i.e., $X_j - X_i \leq C_{ij}$. Every constraint C_{ij} is defined as any of the types shown in Figures 4, 6 or 5 is represented by a possibility distribution π_{ij} over the continuous time scale T. In the framework of Machine^{TF}, a FTCN is used to represent a fuzzy temporal plan where every variable represents the execution time of one of the actions of the plan and every fuzzy binary temporal constraint is posted during the resolution process every time that an action solves a subgoal or when actions are reordered to avoid interferences between them, as will be explained later. Figure 9 shows a soft temporal plan where the annotations under each action a represent the soft constraint defined for $start(a)$, i.e., $C_{start(a),0}$ in the FTCN.

4 The scheduler

The scheduler module we used in Figure 2 takes as input a partial plan. This partial plan is the current plan under construction by the planner, that can be even an plan with some actions missing, and the main idea is to check the temporal consistency of this plan. That is, the main goal of the scheduler is to study if there are solutions to the constraints set associated with the partial plan. Therefore, in a first step the scheduler generates the FTCN of the partial plan, analyzes and propagate all the constraints, and finally it gives as output information about the consistency of such network.

4.1 Fuzzy temporal constraint network concepts

In this process it is very important to take into account the fuzzy representation of the temporal information we have used. Thus, we can see how in a soft temporal plan, like the one shown in Figure 9, we don't represent a unique solution but a set of solutions such that all of them have the same actions, but they might be scheduled for execution in different orders, all of them consistent with the FTCN of the temporal plan. This is very important since during the design of the plan, Machine^{TF} does not commit to assign a precise execution time for every action like most temporal planners (Do and Kambhampati, 2001; Haslum and Geffner, 2001; Smith and Weld, 1999). This excess of commitment would make the planner to increase the number of backtracks. Instead, Machine^{TF} maintains only the consistency of the soft temporal plan without committing to a crisp solution. Only at the end of the planning process, a ground solution is found and scheduled for execution.

In other words, a crisp solution to a FTCN is a tuple of crisp values $s = (x_0, \ldots, x_{n+1})$ which represents an assignment of the form $X_i = x_i, x_i \in T$. In the framework of Machine^{TF}, a solution is an schedule of its actions, that is, an assignment of a crisp execution time for every action of the plan that may be used to execute the plan in practice. The set of solutions of a FTCN may be easily obtained by a solution extraction procedure (Figure 11) and every solution has a degree of consistency which quantifies how accurate the solution is with respect to the constraints of the FTCN.

Definition 2 *A σ-possible solution of a FTCN $\mathcal{N}$ is a tuple $s = (x_0, x_1, \ldots, x_{n+1})$ that verifies that $\pi_S(s) = \sigma$ and*

$$\pi_S(s) \;=\; \min_{0 \le i,j \le n+1} \pi_{ij}(x_j \;-\; x_i) \tag{1}$$

Additionally, the most accurate solution that could be obtained from the set of solutions to a FTCN is given by the following value α, that may be used as a measure of the "goodness" of the FTCN (the fuzzy temporal plan).

Definition 3 *A FTCN is α-consistent if the set of possible solutions $S \subseteq \mathcal{R}^{n+2}$ verifies*

$$\sup_{s \in S} \pi_S(s) = \alpha \tag{2}$$

Example 2 *Table 1 shows a possible solution for the fuzzy plan of Figure 9. This solution is 1-consistent, meaning that all the constraints have been completely satisfied.*

Table 2 shows an alternative solution of the same plan but now the consistency is 0.73, meaning that some constraints have not been completely verified. In this case there is only one constraint that is not completely satisfied, and this one is the constraint between actions OD (Open Debarking) and CD (Close Debarking). This solution could have been obtained after the happening of some delay that make impossible the previous solution (in this case it is produced by a delay of action OD).

The equivalent to the classical (or not fuzzy) situation is the 1-consistency, or we can say only consistency. The inconsistency is related to the absence of solutions ($\alpha = 0$). When the network is 1-consistency then the distribution π_S is normalized, that is, there is at least a solution that completely verifies the desires of the user, although there may also be solutions with a different and intermediate consistency value.

In the framework of MACHINE^{TF}, the meaning of the value α is a generalization of the motivation for using fuzzy sets and it needs a further explanation. MACHINE^{TF} is deterministic, that is, all the plans found are valid to solve the problem, so the main difference between all possible valid plans is the degree of satisfaction of the soft temporal constraints α. Those valid plans with the highest value, $\alpha = 1$, are those plans such that there is at least a solution (schedule) that satisfy completely the desires of the user. On the contrary, those plans such that $0 < \alpha < 1$ are admissible plans, that is, plans that contain at least one admissible solution (schedule) that doesn't satisfy the original desires of the user but they still satisfy the constraints of the user at some degree greater than 0.

In this way, the consistency can be considered as a quality measure of the partial plan since reflects the desires of the user. In order to study the consistency of a network, next we present some previous concept and results described in (Marín et al., 1997).

Definition 4 *Two FTCN $\mathcal{N}$ and $\mathcal{M}$ with the same number of variables are equivalent if and only if every σ-possible solution of one of them is also a σ-possible solution of the other, that is,*

$$\pi_S^N(s) = \pi_S^M(s)$$

where π_S^N and π_S^M are the distribution associated with the fuzzy sets of the possible solutions of the FTCN $\mathcal{N}$ and $\mathcal{M}$ respectively.

An FTCN $\mathcal{M}$ is said to be a *minimal network* if its constraints are minimal, regarding inclusion, with respect to all its equivalent FTCNs $\mathcal{N}$. The constraints M_{ij} of the minimal network are obtained by means of an exhaustive propagation of constraints, through the following expression:

$$M_{ij} = \bigcap_{k=0}^{n+1} L_{ij}^k,$$

where L_{ij}^k is the constraint induced by all the paths of length k that connect variables X_i and X_j:

$$L_{ij}^k = \cap\, C_{i_o,i_1,\ldots,i_k}^k \;\; i_1 \ldots i_{k-1} \leq n+1,\; i_0 = i,\; i_k = j$$

$$C_{i_o,i_1,\ldots,i_k}^k = \sum_{p=1}^{k} L_{i_{p-1},i_p}.$$

The network $\mathcal{N}$ is inconsistent if and only if its minimal constraint is the empty distribution, that is $\pi_\emptyset(x) = 0 \;\forall x \in \mathcal{R}$. The network is consistent (or 1-consistent) if and only if the constraints M_{ij} are normalized. The degree of consistency of the network is calculated by

$$\alpha = \underset{s \in R^{n+2}}{sup}\; \pi_S(s) \;=\; \underset{s \in R^{n+2}}{sup}\; \underset{0 \leq i,j \leq n+1}{min}\; \pi_{ij}(x_j - x_i),$$

where each π_{ij} is associated the minimal constraint between the variables X_i and X_j.

The minimal network always verifies

$$M_{ij} \subseteq M_{ik} \oplus M_{kj},\; i,j,k \leq n+1.$$

This means that a new constraint propagation process would not provide any additional information on M_{ij}. In our case, the minimal network will be used to detect the consistency or inconsistency of the network. The algorithm to calculate the minimal network (see Figure 10) is a fuzzy generalization (Marín et al., 1997; Vila and Godo, 1994a) of the shortest-path algorithm proposed in (Dechter et al., 1991).

As the constraints have been defined as trapezoidal fuzzy numbers, we can easily manipulate them using the well known arithmetic described in (Dubois and Prade, 1978). Thus, by using normalized trapezoidal fuzzy numbers this minimization algorithm is executed in polynomial time. The fuzzy operation required are:

$$(a, b, c, d) \oplus (e, f, g, h) = (a + e, b + f, c + g, d + h)$$

$$(a, b, c, d) \cap (e, f, g, h) = (max\{a, e\}, max\{b, f\}, min\{c, g\}, min\{d, h\}).$$

Finally, other important concept that we use in the interpretation of the use of the FTCN in the planning process is the independence between network variables.

Definition 5 *Two variables X_i and X_j belonging to a FTCN $\mathcal{N}$ are independent if and only if the minimal constraints M_{ij}, M_{i0} and M_{0j} verify $M_{ij} = M_{i0} \oplus M_{0j}$ $\forall\, i, j$.*

When the variables are independent then the constraints M_{i0} and M_{j0} contains all the needed information to make variable assignment without consider the rest of constraints.

These results will be used in the next subsection to interpret and estimate the consistency of a network.

4.2 Consistency checking

The main goal of the scheduler module is determining the consistency of the network associated with the partial plan analyzed by the planner. In the fuzzy case, the consistency of the network is a degree value. Obviously, a consistency value $\alpha = 0$ implies the inconsistency of the network and there are no possible solutions. In other case, the scheduler returns to the planner a real value between 0 and 1. This value represents the degree in which the desires of the users are satisfied by the best solution of the network, and therefore it can be interpreted as a quality measure of the solutions associated with the network. The idea is to integrate this value in the heuristic function of the search process of the planner. Therefore, the first step is the calculus of the consistency of the network.

This calculus may seem easy but it poses a difficult question. Given Equation 2, the value of the maximum degree of consistency α of a FTCN might be a very good source of information for decision making, but up to now, it cannot be obtained analytically.

In order to obtain the consistency value, we focus on the minimal network M of the original one. Thus, when all of the fuzzy temporal constraints in the minimal network are normalized, i.e.,

$$hgt(\pi_{ij}) = 1$$

where

$$hgt(\pi) = \underset{t \in T}{max} \ \ \pi(t),$$

the degree of consistency α is equal to 1, that is, there is at least one solution that completely meets the constraints (Marín et al., 1997)representing the initial desires of the user, otherwise $0 \leq \alpha < 1$ and thus, it is not possible to find a solution that verifies completely all the initial desires of the user but in any case verifies the soft constraints in a degree. In these cases some search algorithm like simulated annealing or genetic algorithms could be used to find that value, but this will degrade severely the performance of MACHINE^{TF} since this process should be launched to evaluate every possible plan being built.

In the framework of MACHINE^{TF}, although the value of α is not known, it may be bounded and used to guide the search of the planning process. Thus, the idea is to set bounds to the value of the consistency, and to use these bounds to estimate it.

The algorithm (Marín et al., 1997) described in Figure 11 allow us to obtain solutions for a minimal FTCN once the consistency value is a known value.

In any case, we don't know the consistency value α, and therefore this procedure cannot be used directly. With the aim of obtaining a pessimistic estimation of the consistency we propose a greedy algorithm that consists in an iterative procedure that encapsulates the solution extraction algorithm.

Initially, we take $\sigma = 0$, and therefore we work on the support sets[1] of the fuzzy constraints of the minimal network. Following the algorithm of solution extraction we select a value of the support set for each one of the F sets. The set F contains in each step the possible values of each variable, taking into account the own constraints of the variable and the new constraints generated by the new assignments made by the algorithm. The selection procedure is not determined in the algorithm, therefore we have checked different possibilities (to select the minimum value of the support set, the center point of the support set,...). The best results have been obtained when we select the minimum value of the mode, i.e., the core set of values of a fuzzy set $m(A) = \{u, \mu_A(u) = 1\}$ (Figure 12).

Once this value has been selected for each variable, we have a solution that can be evaluated through equation 1 obtaining a value σ_s. Now, we know that the value of the consistency of the network is equal or greater that σ_s since the consistency is the greatest π_S for all the possible solutions.

The next step consists in using again the algorithm of solution extraction but now with σ_s as new σ parameter. In this case we have selected a random value of the σ_s-cut of each F set. Following the same steps as above we obtain a new solution of the minimal network, and we repeat the process until we obtain $\sigma_s = 1$ or alternatively during a fixed number of runs. The infeasible solutions ($\sigma_s = 0$) are rejected, and the remaining ones are used to give a pessimistic estimation of the consistency:

$$\delta_l = \underset{i}{max} \ \pi_S(sol_i) \ i = 1, \ldots, k$$

where sol_i are the set of solutions considered by the previous process and k is the number of runs. Clearly,

$$\delta_l \leq \alpha.$$

On the other hand, let us consider the following value

$$\delta_u = \underset{i}{min}\ hgt(\pi_{0i})$$

for a minimal FTCN. Now, we are going to prove that this value is an optimistic estimation of the consistency, that is, $\alpha \leq \delta_u$.

Proposition 1 *The consistency value α is always equal or less than the value $\underset{i}{min}\ hgt(\pi_{0i})$, that is, $\alpha \leq \delta_u$.*

Proof: Let $\delta_i = hgt(\pi_{0i})$ the height of the fuzzy number π_{0i}, since π_{0i} are trapezoidal fuzzy numbers

$$\exists x_i^* \mid \pi_{0i}(x_i^*) = \delta_i.$$

We can take as solution

$$s^* = (x_0^*, x_1^*, \ldots, x_{n+1}^*).$$

Because of the selection process of each x_i^*

$$\pi_{0i}(x_i) \leq \pi_{0i}(x_i^*)$$

for any x_i, then

$$\pi_S(s) = \underset{i,j}{min}\ \pi_{ij}(x_j\ -\ x_i) \leq\ \underset{i}{min}\ \pi_{0i}(x_i) \leq\ \underset{i}{min}\ \pi_{0i}(x_i^*) = \underset{i}{min}\ \delta_i = \delta_u$$

and therefore

$$\alpha = \underset{s}{sup}\ \pi_S(s) \leq \delta_u.$$

Moreover, this result can be improved when the variables are independent. For each constraint M_{0i} we consider its modal interval defined by

$$L_i = mod(M_{0i})$$

corresponding to all the elements of the domain whose membership function is equal to the height of the fuzzy constraints. We define

$$L = L_0 \times L_1 \times \ldots \times L_{n+1}.$$

When the variables are independent all the element of L are solutions of the network and the consistency coincide with the value of δ_u.

Proposition 2 *When the variables of the FTCN are all independent then every element s of L is a δ_u-possible solution of the network, and*

$$\alpha = \delta_u.$$

Proof: For independent variable we know that

$$M_{ij} = M_{i0} \oplus M_{0j},$$

and therefore

$$M_{ij} = M_{0j} \ominus M_{0i}.$$

By taking a solution

$$s = (x_1, x_2, \ldots, x_{n+1}) \in L$$

then $x_i \in L_i$ *y* $x_j \in L_j$*, and*

$$(x_j - x_i) \in mod(M_{0j} \ominus M_{0i})$$

and

$$\pi_{ij}(x_j - x_i) = min\{\delta_i, \delta_j\}$$

then

$$\underset{i,j}{min}\ \pi_{ij}(x_j, x_i) = min\{\underset{i}{min}\ \pi_{0i}(x_i), \underset{i \neq 0,j}{min}\ \pi_{ij}(x_j, x_i)\} = \delta_u$$

and obviously taking into account the selection of the elements of the solution

$$\alpha = \delta_u.$$

For independent variable the estimation of the consistency is really simple, since the modal interval of the constraints M_{0i} define completely the set of solutions. In the general case, the situation is more complex since the rest of binary constraints need to be taken into account, and the solution can be obtained outside the set L. In any case, the previous result gives us an interesting interpretation of the estimation δ_u. We can say that this is a rather good estimation since it has been obtained from a simplified model, that is, a model in which the variables have been considered as independent ones.

In any case, both estimation values δ_l and δ_u bounds the unknown value of α

$$\alpha \in [\delta_l, \delta_u].$$

Since the exact value of α is unknown, the centroid of this interval is used to approximate it. This value is then used by MACHINE^{TF} as a secondary ranking criterion during the search process, where the primary ranking criterion is the heuristic evaluation function of MACHINE, that has proven to be very useful in several domains (Castillo et al., 2001). This implies a deeper integration of planning and scheduling techniques since now the heuristic evaluation that guides the steps of the planner also takes into account the degree of temporal consistency in such a way that, given two plans with the same solving power, MACHINE^{TF} will choose that with the higher temporal consistency. Next section shows how the planner handles all the soft temporal constraints.

4.3 Handling soft temporal constraints during planning

The main loop of the planning algorithm is a best first search guided by a ranking function that takes into account the usefulness of the plan being constructed (Castillo et al., 2001) and the soft consistency of temporal constraints, as explained before. At every step, a pending subgoal is selected for its resolution so that either a new action or a existing one are used to solve it. In both cases new soft temporal constraints are posted to ensure a correct ordering of actions, propagated by means of an all-pairs polynomial algorithm (Figure 10), and the soft temporal consistency of the resulting plan is evaluated and taken into account for its ranking. Eventually, if some interferences between concurrent actions is detected, this is corrected by adding new soft temporal constraints to produce a reordering of the actions that avoids the interference.

Goal Satisfaction Let us suppose that an action a with an effect f that takes "about t_f time units", solves a precondition of an action b. Then the following soft temporal constraint

$$start(b) \geq start(a) \oplus (t_f - \delta, t_f, t_f, t_f + \delta)$$

is posted and propagated (Figure 10). The satisfaction of a subgoal is recorded in a structure called causal link in order to avoid that no action that deletes f could overlap in the interval between $start(a)$ and $start(b)$.

Duration of actions For all actions, either with maximum duration $maxbound$ or not, we first calculate the following values:

$$dur^{min}(a) = \max_{f' \ in\ effects(a)} (t_{f'} - \delta, t_{f'}, t_{f'}, t_{f'} + \delta)$$

$$dur^{max}(a) = \begin{cases} (-\infty, -\infty, +\infty, +\infty) \text{ (unspecified)} \\ (maxbound - \delta, maxbound, maxbound, maxbound + \delta) \text{ (bounded)} \end{cases}$$

and next we post and propagate the following constraint:

$$start(a) \oplus dur^{max}(a) \geq end(a) \geq start(a) \oplus dur^{min}(a)$$

Deadline goals For every deadline goal g, with deadline $(t_{min}-\delta, t_{min}, t_{max}, t_{max}+\delta)$, meaning that goal g must be achieved "more or less between times t_{min} and t_{max}", which has been solved by an action a with its effect f that takes "about t_f time units", the following soft constraint is posted and propagated.

$$start(a) = (t_{min} - \delta, t_{min}, t_{max}, t_{max} + \delta) \ \ominus (t_f - \delta, t_f, t_f, t_f + \delta)$$

Concurrent actions and threats Say that action c with the effect $(not\ f)$ that takes "about $t_{not(f)}$ time units" overlaps with a casual link from actions a to b with respect to the effect f. Then, the overlapping must be avoided by reordering the threatening action c by promotion (putting c after all the effects of b)

$$start(c) \geq start(b) \oplus dur^{min}(b)$$

or demotion (putting a after the negative effect of c)

$$start(a) \geq start(c) \oplus (t_{not(f)} - \delta, t_{not(f)}, t_{not(f)}, t_{not(f)} + \delta)$$

With these new capabilities for handling soft temporal constraints, Machine^{TF} is able to obtain soft temporal plans like that shown in Figure 9 by means of the solution extraction procedure shown in Figure 11 and execute the plan. However, there are some additional features that must be mentioned.

5 Monitoring and rescheduling

One of the advantages of Machine^{TF} is that the underlying FTCN in a fuzzy temporal plan may be used to monitor its execution and to reschedule part of the plan in the case that a delay has occurred during its execution but maintaining the causal structure of the plan. In the case of Machine^{TF} there may be three different types of delays.

1. Local delays. They are delays that only affect locally to an isolated branch of the temporal plan without affecting the remaining actions. This is the easiest case since it could not affect deadline goals or makespans which depend on more global temporal constraints. In these cases, a new reschedule may be found only for the actions of the affected branch leaving the remaining schedule unaltered.

2. Global delays. They are more important since they may affect all of the remaining actions and deadline goals or makespans might also be affected and hence, a whole new reschedule might be needed.

3. Infeasible delays. When an action a_j has been delayed more than it is acceptable even taking into account the existence of soft temporal constraints,i.e., $\pi_{0j}(delay) = 0$, then no reschedule is possible and the possibility of re-planning should be considered.

The algorithm to monitor the execution of fuzzy temporal plans is shown in Figure 13. It uses the set of actions of the plan still to be executed Π, a set of already executed actions C and a queue of delayed actions Q, that is, actions that should have executed before but they are not able to execute because, for some reason, some of its preconditions have not been achieved yet by their producing actions. The variable $TIME$ is used to track the evolution of the schedule. It works as follows. The monitoring procedure defines a tentative schedule by obtaining a solution with maximal consistency. If nothing goes wrong ($Q = \emptyset$), every action whose execution time equals to $TIME$ and whose preconditions have already been satisfied, is executed and the variable $TIME$ is increased to the earliest execution time of the remaining actions in Π.

However a delay (either with local scope or global scope) might occur at time $TIME$, that is, some of the effects of an action might take longer than expected producing the delay of all of the actions that had been scheduled to time $TIME$ but that have that missing effect in their preconditions. In this case, delayed actions are included in the queue Q and variable $TIME$ is continuously increased by a minimum $\Delta TIME$ producing, in every iteration, a re-computation of the schedule for actions either in Q or Π. This reschedule of the remaining actions in the plan is needed to propagate the delay of the actions in Q to any future action that could have any temporal constraint relative to them. This procedure allows to readapt the schedule to the detected delay. It must be said that, in the case of delays, since new schedules are being continuously obtained to fit the delay, the consistency σ of the schedules may decrease due to the violation of any fuzzy temporal constraint but it will still be acceptable whenever $\sigma > 0$. The extreme case is when the accumulated delay is completely unacceptable and the consistency of the schedule is 0. This is detected in step 4 when the variable $TIME$ takes an infeasible value for some action a_j such that $\pi_{0j}(TIME) = 0$.

This capability for monitoring fuzzy temporal plans is very useful in realistic domains. One could have argued that this could have also been achieved by obtaining a rigid temporal plan and executing it in a flexible manner, however this would pose severe questions on the consistency of the explicit delay of an action since it might produce unsolvable flaws with respect to future actions causally dependent of the delayed action or produce any unexpected interference with the effects of other concurrent actions. In the case of MACHINE^{TF}, the delay of actions in order to flexibly modify a previous schedule is a safe process since it is based on the fuzzy temporal constraints explicitly included in the plan, which have been obtained taking into account the existence of threats and causal relations between actions, as explained in previous sections. Hence, no feasible delay nor reschedule obtained on the basis of the FTCN of the plan could produce an unexpected interference between parallel actions, either on local delays or global delays.

6 An example

The following example shows the monitoring of the plan in Figure 9 and simulates the occurrence of several unexpected delays as well as the corresponding reschedule of actions. The simulation is shown in Table 3.

The first row is the initial schedule with $\sigma = 1$ and it is shown in Figure 14.a). Once the execution starts, four delays have been simulated.

1. The first one (`Stop-Refuel`) is a local delay that only affects one action.

The remaining delays are global ones.

2. It may also be seen that the first global delay (`ZOOM`) only produces a reschedule of the actions without affecting the consistency (quality) of the plan ($\sigma = 1$, Figure 14.b).

3. However the second global delay (`STOP`) degrade a little the consistency of the plan, that is, it produces a delay that does not completely satisfy the strict makespan, but it might be acceptable in the terms of the fuzzy temporal constraints represented in the domain ($\sigma = 0.73$). It is worth noting that the solution extraction procedure also suggest reducing the expected duration of action `Close-Debark` (`CD`) in order to maintain the highest consistency. The duration of action `CD` encoded in the domain is the fuzzy subset $\pi_{OD-CD} \equiv (150\ 200\ 300\ 350)$, its duration in every previous schedule had been fixed to 200 time units ($\pi_{OD-CD}(200) = 1$) but after this delay, the solution extraction procedure suggests to fix it to 186.6 time units ($\pi_{OD-CD}(186.6) = 0.73$).

4. However, this is not finally possible and there is a new delay of action `CD` that ends the plan with a consistency of $\sigma = 0.6$.

7 Some experiments

This section shows the performance of MACHINE^{TF} in several realistic problems and one problem based on the zeno travel domain shown along the paper. The description of these problems is as follows.

Problem # 1. It is the problem in the the zeno travel domain, as seen up to now with fuzzy durations.

Problem # 2. It is a sample problem in the fire fighting scenario.

Problem # 3. MACHINE^{TF} is also able to deal with problems from other domains, like manufacturing systems . This third problem is a real-life example of batch manufacturing for a dairy products problem (Castillo et al., 2000; Castillo et al., 2001) with fuzzy due dates. It consists of 48 different possible actions.

The results of the execution of MACHINE^{TF} in these domains are shown in Table 4. In realistic domains with a complex knowledge representation, run times do not seem to scale as efficiently as many of the high performance planners that take part in the International Planning Competition (Long and Fox, 2003a) but, on the other hand, its temporal expressiveness enable it to deal with realistic problems where these other planners may not succeed, that is, problems with ill defined temporal knowledge, to obtain approximately consistent temporal plans and to adapt the execution of these plans to unexpected delays during the execution.

8 Final remarks

This paper has outlined MACHINE^{TF}, a temporal planner based on the representation of fuzzy temporal knowledge able to build fuzzy temporal plans. The main contributions of MACHINE^{TF} are the following.

- It is able to obtain plans in domains in which time is not precisely known. This is very important in real world problems and mainly with respect to classic temporal planners and schedulers, which need a rigid representation of time and, therefore, they lose much of the temporal expressiveness of these domains.

- It is also able to obtain approximate solutions where exact solutions are not feasible, given a soft interpretation of temporal constraints. This is also very important in hard temporal problems since it transforms the inconsistency in a matter of preference and degrees, more in accordance with most usual constraints in real life.

- And finally, fuzzy temporal plans obtained provide a flexible time line for its execution. Given that a fuzzy temporal plan may be scheduled in different time lines, depending on the solution that has been extracted, if an unexpected delay occurs (due to an error, a delay in the execution of an action, etc), a new schedule adapted to the new situation may be re-extracted without the need to re-plan.

To illustrate the results of these techniques, some examples extracted from the domain of crisis management have been shown. These examples are inspired in the work being done by the authors under a research contract with the Andalusian Regional Ministry of Environment (de la Asunción et al., 2003; de la Asunción et al., 2005) devoted to an intelligent decision support system for the assisted design of forest fire fighting plans.

However there are some open problems that need further work. The first one is the need to improve the featuring of FTCNs in order to obtain a better estimation of the global consistency α. The second one is related to the monitoring procedure. As explained before, in the case of delays ($Q \neq \emptyset$) the procedure continuously reschedules the remaining actions propagating the delay along the existing constraints. This is mainly done by a shortest path algorithm whose efficiency is $O(n^3)$ where n is the number of time points (actions) of the FTCN. This means that, if the number of actions is very large and the needed temporal resolution to advance $TIME$ is very small, there might be no time to complete the propagation. Therefore, a post-processing of the FTCN that could restrict the propagation of constraints to make the reschedule faster might be used. In fact, a reformulation like the one in proposed in (Morris and Muscettola, 2000) defined on TCNs could be adapted to the case of fuzzy constraints to cope with degrees of consistency and applied to the fuzzy temporal plans of MACHINE^{TF} before the monitoring procedure.

9 Related work

Machine^{TF} is a deterministic planner and uncertainty is restricted only to the time of occurrence of events so it is a classical goal-oriented planner leaving out of its scope other approaches to handle uncertainty that involve nondeterminism and a planning process focused on optimizing a plan utility function like in MDPs (Blythe, 1999; Bresina et al., 2002). However it still faces a very important problem in deterministic domains, that is, how to obtain and execute a plan when part (or the whole) of the temporal knowledge is uncertain.

The need to represent and reason about uncertain temporal knowledge has been a very active field in the literature devoted to real problems. The approach presented in this paper is based on the use of possibility distributions (Zadeh, 1978) to model either uncertainty or imprecision or a preference criterion in same the sense of (Khatib et al., 2001) (in fact fuzzy temporal constraints are a special case of this approach) and not necessarily related to probabilistic uncertainty like in (Bresina et al., 2002). There are also other approaches that deal with temporal uncertainty that have been built over STN (Dechter et al., 1991) like STNUs, simple temporal networks under uncertainty, (Vidal and Fargier, 1999; Morris and Muscettola, 2000). These approaches are based on the use of contingent (uncertain) constraints defined on pairs of temporal points to represent uncertain temporal knowledge. Contingent temporal constraints do no explicitly affect the consistency of the STNU, that is still a crisp result in $\{0,1\}$, although it does affect the schedule of a solution. The duration of a contingent constraint cannot be predicted before execution, and variables related to a contingent constraint must be maintained uninstantiated until the execution of the plan[2]. In the case of Machine^{TF}, fuzzy temporal constraints provide a little more of expressiveness and they define a possibility measure over these contingent links to represent degrees of uncertainty or preference so final solutions will also have associated a σ-consistency varying softly in $[0,1]$. Although Machine^{TF} can make predictions, with maximal possibility, on these contingent links during the monitoring of the plan, there is no way to guarantee that this prediction will be finally obtained in a $\sigma-$consistent real schedule nor that the final schedule will be σ-consistent, $\sigma > 0$.

There are also works on scheduling temporal plans either on STNs (Muscettola et al., 1998) or STNUs (Morris and Muscettola, 2000) but in all of them any schedule may suddenly change from consistency to inconsistency. In Machine^{TF} there are degrees of consistency allowing for approximate plans or schedules when completely consistent ones are not possible.

Notes

[1]The support of a fuzzy set A is the set $s(A) = \{u, \mu_A(u) > 0\}$

[2]This leads to the definition of several types of controllabilities (strong, dynamic and weak) depending on how one may ensure, before execution, that a consistent solution may be finally scheduled.

References

Avesani, P., Perini, A., and Ricci, F. (2000). Interactive case-based planning for forest fire management. *Applied Intelligence*, 13(1):41–57.

Bacchus, F. and Ady, M. (2001). Planning with resources and concurrency. a forward chaining approach. In *IJCAI'01*.

Bartk, R. and Mecl, R. (2003). Integrating planning into production scheduling: Visopt shopfloor system. In Kendall, G., Burke, E., and Petrovic, S., editors, *Proceedings of the 1st Multidisciplinary International Conference on Scheduling:Theory and Applications (MISTA)*, pages 259–278.

Bienkowski, M. (1995). Demonstrating the Operational Feasibility of New Technologies: The ARPI IFDs. *IEEE Expert*, 10(1):27–33.

Biundo, S., Aylett, R., Beetz, M., Borrajo, D., Cesta, A., Grant, T., McCluskey, L., Milani, A., and Verfaillie, G. (2003). PLANET technological roadmap on AI planning and scheduling. Electronically avaliable at `http://www.planet-noe.org/service/Resources/Roadmap/Roadmap2.pdf`.

Biundo, S. and Schattenberg, B. (2001). From abstract crisis to concrete relief - a preliminary report on combining state abstraction and htn planning. In *6th European Conference on Planning (ECP-01)*.

Blythe, J. (1999). Decision-theoretic planning. *AI Magazine*, 20(2):37–54.

Bresina, J., Dearden, R., Meuleanu, N., Ramakrishnan, A., Smith, D., and Washington, R. (2002). Planning under continuous time and resource uncertainty: A challenge for AI. In *Proc. UAI*.

Castillo, L., Fdez-Olivares, J., and González, A. (2000). Automatic generation of control sequences for manufacturing systems based on nonlinear planning techniques. *Artificial Intelligence in Engineering*, 4(1):15–30.

Castillo, L., Fdez-Olivares, J., and González, A. (2001). Mixing expresiveness and efficiency in a manufacturing planner. *Journal of Experimental and Theoretical Artificial Intelligence*, 13:141–162.

Cohen, P., Greenberg, M., Hart, D., and Howe, A. (1989). Trial by fire: understanding the design requirements for agents in complex environments. *AI Magazine*, 10(3):32–48.

D. Dubois, H. F. and Prade, H. (2003). Fuzzy scheduling: modelling flexible constraints vs. coping with incomplete knowledge. *European Journal of Operational Research*, 147:231–252.

de la Asunción, M., Castillo, L., Fdez-Olivares, J., García-Pérez, O., González, A., and Palao, F. (2003). SIADEX: Assisted Design of Forest Fire Fighting Plans by Artificial Intelligence Planning techniques. `http://siadex.ugr.es`.

de la Asunción, M., Castillo, L., Fdez-Olivares, J., García-Pérez, O., González, A., and Palao, F. (2005). Siadex: an interactive artificial intelligence planner for decision support and training in forest fire fighting. *Artificial Intelligence Communications*, 18(4).

Dechter, R. (2003). *Constraint processing*. Morgan Kaufmann.

Dechter, R., Meiri, I., and Pearl, J. (1991). Temporal constraint networks. *Artificial Intelligence*, 49:61–95.

Do, M. and Kambhampati, S. (2001). SAPA: a domain-independent heuristic metric temporal planner. In *European Conference on Planning*, pages 109–120.

Dubois, D., Fargier, H., and Prade, H. (1993). The use of fuzzy constraints in job-shop scheduling. In *Proc. of IJCAI-93/SIGMAN Workshop on Knowledge-based Production Planning, Scheduling and Control*, Chambery, France.

Dubois, D. and Prade, H. (1978). Operations on fuzzy numbers. *International Journal of Systems Science*, 9:613–626.

Haslum, P. and Geffner, H. (2001). Heuristic planning with time and resources. In *European Conference on Planning*, pages 121–132.

Khatib, L., Morris, P., Morris, R., and Rossi, F. (2001). Temporal constraint reasoning with preferences. In *IJCAI 2001*, pages 322–327.

Laborie, P. and Ghallab, M. (1995). Planning with sharable resource constraints. In *IJCAI'95*, pages 1643–1649.

Long, D. and Fox, M. (2003a). The 3rd international planning competition: Results and analysis. *Journal of Artificial Intelligence Research*, 20:1–59.

Long, D. and Fox, M. (2003b). PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research*, 20:61–124.

Marín, R., Cárdenas, M., Balsa, M., and Sánchez, J. (1997). Obtaining solutions in fuzzy constraint networks. *International Journal of Approximate Reasoning*, 16:261–288.

Morris, P. and Muscettola, N. (2000). Execution of temporal plans with uncertainty. In *AAAI 2000*, pages 491–496.

Muscettola, N. (1994). HSTS: integrating planning and scheduling. In Zweben, M. and Fox, M., editors, *Intelligent scheduling*, pages 169–212. Morgan Kaufmann.

Muscettola, N., Morris, P., and Tsamardinos, I. (1998). Reformulating temporal plans for efficient execution. In *6th Conf. on Principles of Knowledge Representation and Reasoning*, pages 444–452.

Myers, K. L. (1999). CPEF: A continuous planning and execution framework. *AI Magazine*, 20(4):63–69.

noz Avila, H. M., Aha, D. W., Breslow, L., and Nau, D. (1999). HICAP: An interactive case-based planning architecture and its application to noncombatant evacuation operations. In *Ninth Conference on Innovative Applications of Artificial Intelligence*, pages 879–885. AAAI Press.

Penberthy, J. and Weld, D. (1994). Temporal planning with continous change. In *AAAI'94*, pages 1010–1015.

Smith, D. and Weld, D. (1999). Temporal planning with mutual exclusion reasoning. In *IJCAI'99*, pages 326–337.

Vidal, T. and Fargier, H. (1999). Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental and Theoretical A.I.*, 11:23–45.

Vila, L. and Godo, L. (1994a). On fuzzy temporal constraint networks. *Mathware and Soft Computing*, 3:315–334.

Vila, L. and Godo, L. (1994b). Query answering in fuzzy temporal constraint networks. In *IJCAI'95: Proceedings of the Intl. Joint Conference on Atificial Intelligence*, volume 3, pages 315–334.

Weld, D. (1994). An introduction to least commitment planning. *AI Magazine*, 15(4):27–61.

Wilkins, D. E. and Desimone, R. V. (1994). Applying an AI plannet to military operations planning. In Zweben, M. and Fox, M. S., editors, *Intelligent Scheduling*. Morgan Kaufmann.

Zadeh, L. (1978). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1:3–28.

Tables

OB	CB	REFUEL	STOP-REFUEL	ZOOM	STOP	OD	CD	END	σ
1	201	1	101	201	7201	7201	7401	7401	1

Table 1: A consistent solution

OB	CB	REFUEL	STOP-REFUEL	ZOOM	STOP	OD	CD	END	σ
1	201	1	180	300	7340	7340	7526.6	7526.6	0.73

Table 2: A 0.73-consistent solution

TIME	OB	CB	REFUEL	STOP-REFUEL	ZOOM	STOP	OD	CD	END	Detected Delay	σ
0	1	201	1	101	201	7201	7201	7401	7401	-	1
100	[1]	201	[1]	101	201	7201	7201	7401	7401	-	1
101	[1]	201	[1]	(101)	201	7201	7201	7401	7401	STOP-REFUEL	1
180	[1]	201	[1]	[180]	201	7201	7201	7401	7401	-	1
201	[1]	[201]	[1]	[180]	(201)	7201	7201	7401	7401	ZOOM	1
300	[1]	[201]	[1]	[180]	[300]	7300	7300	7500	7500	-	1
7300	[1]	[201]	[1]	[180]	[300]	(7300)	7300	7500	7500	STOP	1
7340	[1]	[201]	[1]	[180]	[300]	[7340]	[7340]	7526.6	7526.6	-	0.73
7526.6	[1]	[201]	[1]	[180]	[300]	[7340]	[7340]	(7526.6)	7526.6	CD	0.73
7540	[1]	[201]	[1]	[180]	[300]	[7340]	[7340]	[7540]	[7540]	-	0.6

Table 3: An example of the execution of the plan in Figure 9 which simulates several delays and their corresponding reschedules. Times in [brackets] represent already executed actions and times in (parentheses) represent delayed actions

Problem	**DS**	**PL**	**NE**	**CT**
P#1	9	18	28	2.1
P#2	17	19	337	27
P#3	48	40	1913	1619.7

Table 4: Experimental results of MACHINE^{TF} in several domains. (**DS**) Domain Size: number of possible actions. (**PL**) Plan length: number of actions included in the plan. (**NE**) Search effort: nodes of the search space explored by the planning algorithm. (**CT**) CPU time in seconds, running CLISP and Linux on a Pentium IV 1,6GHz.

Figure legends

List of Figures

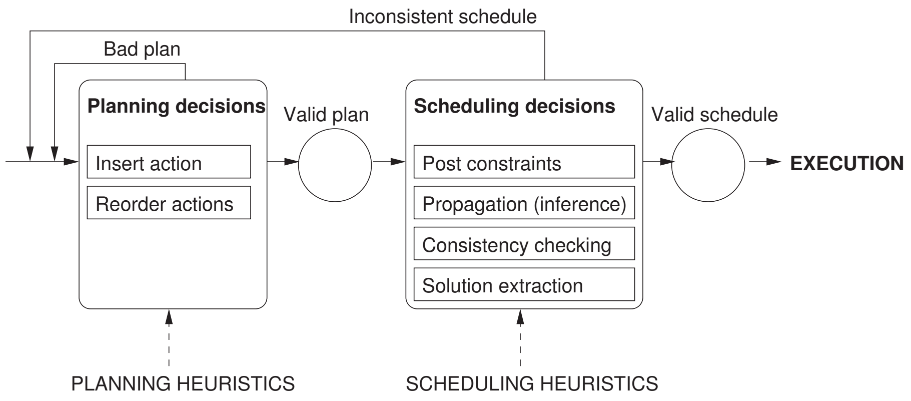


Figure 1: A batch model of planning and scheduling.

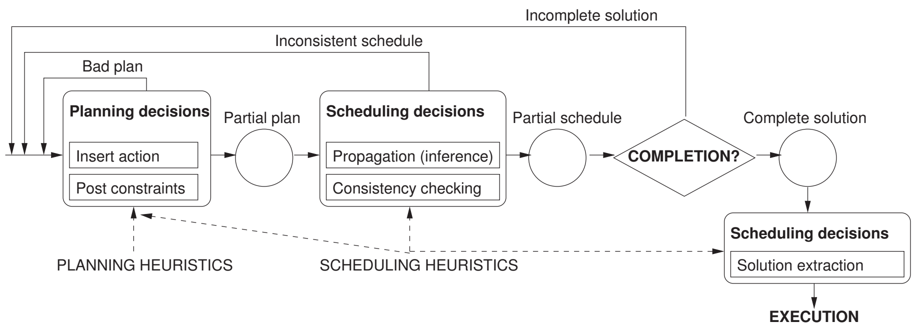


Figure 2: An integrated model of planning and scheduling

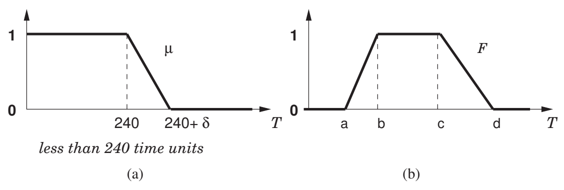


Figure 3: Two fuzzy temporal intervals

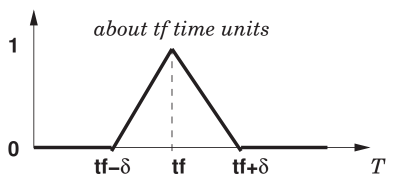


Figure 4: A fuzzy temporal constraints representing ”about tf time units”.

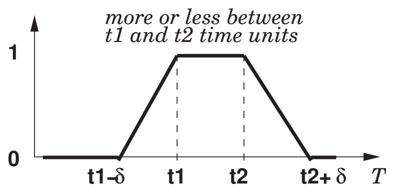


Figure 5: A fuzzy temporal constraints representing ” more or less between t1 and t2 time units.

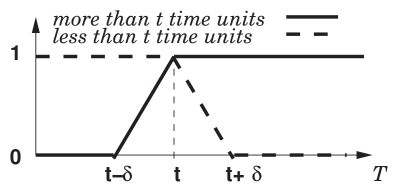


Figure 6: Two fuzzy temporal constraints representing ”more than t time units” or ” less that t time units”.

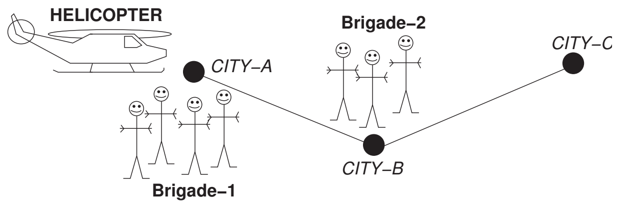
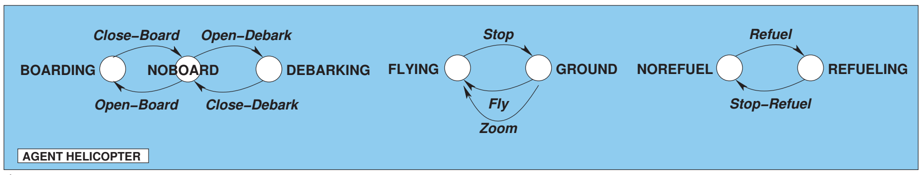


Figure 7: A transport domain inspired in the domain Zeno.



```
(:agent helicopter1
   (:states (ground flying norefuel refueling
             noboard boarding debarking))
   (:action fly
    :parameters (?city1 ?city2)
    :max-duration (6900 7000 7000 7100)
    :pre-condition
       (and (helicopter-at helicopter1 ?city1)
            (refueled helicopter1)
            (state helicopter1 ground))
    :sim-condition
       nil
    :effects
       (and ((not (helicopter-at helicopter1 ?city1)) (0 0 0 0))
            ((not (state helicopter1 ground)) (0 0 0 0))
            ((not (refueled helicopter1)) (6900 7000 7000 7100))
            ((helicopter-at helicopter1 ?city2)(6900 7000 7000 7100))
            ((state helicopter1 flying) (0 0 0 0))))
   (:action refuel
    :parameters nil
    :max-duration (75 100 100 225)
    :pre-condition
       (state helicopter1 norefueling)
    :sim-condition
       (state helicopter1 ground)
    :effects
       (and ((not (state helicopter1 norefuel)) (0 0 0 0))
            ((refueled helicopter1) (75 100 200 225))
            ((state helicopter1 refueling) (0 0 0 0))))
```

Figure 8: The structure and part of the domain of the travel domain of Example 1. All the available actions for the helicopter are shown. Note that there are two modes of flight depending on their speed: a fast flight (zoom), that takes shorter, and a slow flight (fly), that takes longer.

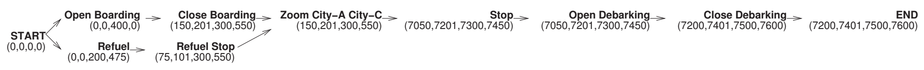


Figure 9: A soft temporal plan for part of problem of Example 1 under the deadline constraint (0,0,7500,7600) for brigade-1 to be at city-c. Every action is labeled with its respective execution time as a fuzzy temporal reference.

```
for k=0 to n+1
  for i=0 to n+1
    for j=0 to n+1
      $M_{ij} = M_{ij} \cap (M_{ik} \oplus M_{kj})$
  If $M_{ij} = \pi_\emptyset$ then the network is inconsistent
```

Figure 10: Constraints propagation

```
**Algorithm Solution Extraction**
let $\langle X, M \rangle$ a minimal FTCN and $\sigma \in [0,1], \sigma < \alpha$

$X_0 = 0$
for i=1 to n+1
    $F = \bigcap_{j<i} (x_j \oplus M_{ji})$
    Select $x_i$ such that $\pi_F(x_i) \geq \sigma$
    $X_i = x_i$
$s = (x_0, x_1, \ldots, x_{n+1})$
Return s, $\pi_S(s) \geq \sigma$
```

Figure 11: Solution extraction

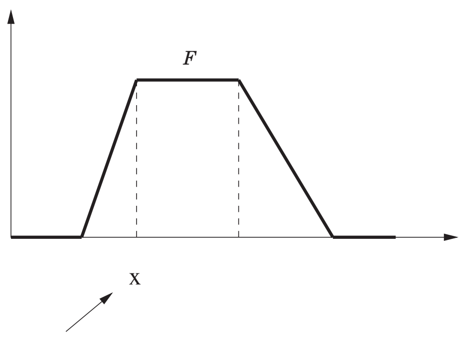
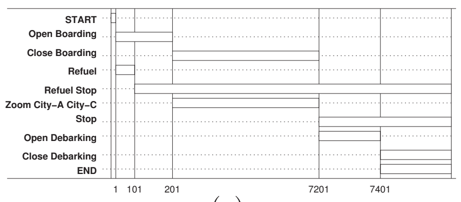


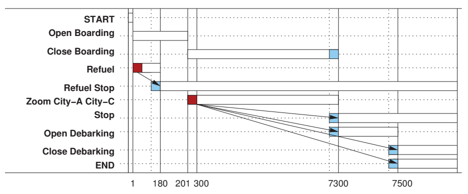
Figure 12: The minimum value of the mode of a fuzzy set

```
1. Initialization
    • Given $\Pi = \{a_0, a_1, \ldots, a_{n+1}\}$ a fuzzy temporal plan
    • $C = \emptyset$ and $Q = \emptyset$
    • Initialize $TIME$
2. Design a possible time line $T$
    • Fix the execution time for every action in $C$ and consider $TIME$ to be the
      execution time for every action in $Q$
    • Use the solution extraction procedure to obtain a time line $s = \{x_i | a_i \in \Pi \cup Q\}$
      with a degree of consistency $\sigma = \pi_S(s)$ given by equation 1 so that action $a_i$ is
      scheduled to be executed at time $x_i$.
3. Monitoring the execution
    (a) If $\Pi$ and $Q$ are empty then **SUCCESS**
    (b) For every action $a_j \in \Pi$ such that $x_j = TIME$
        • Extract $a_j$ from $\Pi$
        • If $preconds(a_j)$ are true in the environment, then execute $a_j$ ($C = C \cup \{a_j\}$)
        • Otherwise delay $a_j$ ($Q = Q \cup \{a_j\}$)
    (c) Advance $TIME$ in $\Delta TIME$
4. If $Q \neq \emptyset$ then monitor the queue.
    (a) For every action $a_j \in Q$
        • If $\pi_{0j}(TIME) = 0$ then **ERROR: TIME OUT**
        • If $preconds(a_j)$ are true in the environment, then extract $a_j$ from $Q$ and
          execute $a_j$ ($C = C \cup \{a_j\}$)
    (b) Go to Step 2
5. Go to Step 3
```

Figure 13: Monitoring algorithm for fuzzy temporal plans



(a)



(b)

Figure 14: Two different schedules obtained from the plan in Figure 9. (a) The original schedule with $\sigma = 1$. (b) The reschedule after the second delay with $\sigma = 1$

Knowledge and plan execution management in planning fire fighting operations

Marc de la Asunción, Luis Castillo, Juan Fdez-Olivares,
Oscar García-Pérez, Antonio González, Francisco Palao
siadex.dev@decsai.ugr.es
Departmento de Ciencias de la Computación e Inteligencia Artificial.
E.T.S. Ingenieria Informatica.
Universidad de Granada (Spain)

Abstract. This work introduces SIADEX [1], a planning framework intented to assist human experts in the design of forest fire fighting plans. Issues about how to engineer planning knowledge for such a system, how to monitor the execution of fighting plans and how to patch unfeasible plans are discussed in detail.

1 Introduction

AI planning techniques have shown to perform very efficiently in providing human experts with valuable tentative plans and strategies either in military [15] or civil [2] domains. These AI planning techniques are only the core of more complex architectures that may also involve knowledge management techniques and their plans might be directly scheduled for real execution or adapted by domain experts before their execution. The proposed system, SIADEX [7], is a plannig framework that falls into this category of AI planning systems. It is being developed under a research contract with the Andalusian Regional Government (Regional Ministry of Environment) [7]. It is conceived as a distributed planning application and it is intended to assist technical staff in the design of forest fire fighting plans.

Thus, in order to achieve an adequate development of such a system, apart from the development of an adequate core planning engine, one has to focus on other planning related techniques. On the one hand, it is necessary to previously perform a knowledge engineering process in order to define an operational framework where to model and represent the knowledge managed by domain experts, planning experts and the own planning engine. In this sense, we have established a methodology that allows to reach "planner-readable" planning domain and problems from the knowledge currently managed by experts or knowledge engineers (planning experts). This methodology assumes that the modeling language (a high-level language, able to support a user-friendly introduction close to the expert understanding) is different from the language used by the planner (low-level language).

On the other hand, in order to obtain a useful system able to be applied in the real world, techniques about the execution and monitoring of temporal plans in a real framework, and techniques devoted to manage the response to unexpected situations have to be developed.

[1]This work has been partially supported by the Spanish MCyT under project TIC2002-04146-C05-02 and the Andalusian Regional Ministry of Environment contract n. NET033957/1

Thus, we also describe a monitoring and replanning process able to (1) reschedule part of a temporal plan in the case that a delay ocurred during its execution, but maintaining the causal structure of the plan; and (2) in the case of an unfeasible delay, patch the plan in a human centered approach.

2 Overall Architecture

The core of the SIADEX architecture, (Figure 1), is a *planning server* that obtains and serves temporally extended plans following a hierarchical planning algorithm. The input to the planning server comes from the *ontology server* (called BACAREX), that is capable of integrating and "distilling" knowledge from different formats and sources (legacy GIS and external databases). It also serves the necessary knowledge (world objects and their properties, world states, temporal constraints, actions and tasks), represented in a standard (PDDL-compliant) planning language, in order to generate fire fighting plans: a chronologically ordered sequence of actions, where every action has a *time stamp* that refers to the time at which it should be executed. These plans are executed under the supervision of the *monitoring module* and a human operator. The communication between these modules is performed using the XML-RPC Protocol[2], a standard protocol that allows complex data structures to be transmitted, processed and returned using XML as encoding and HTTP as transport layer.

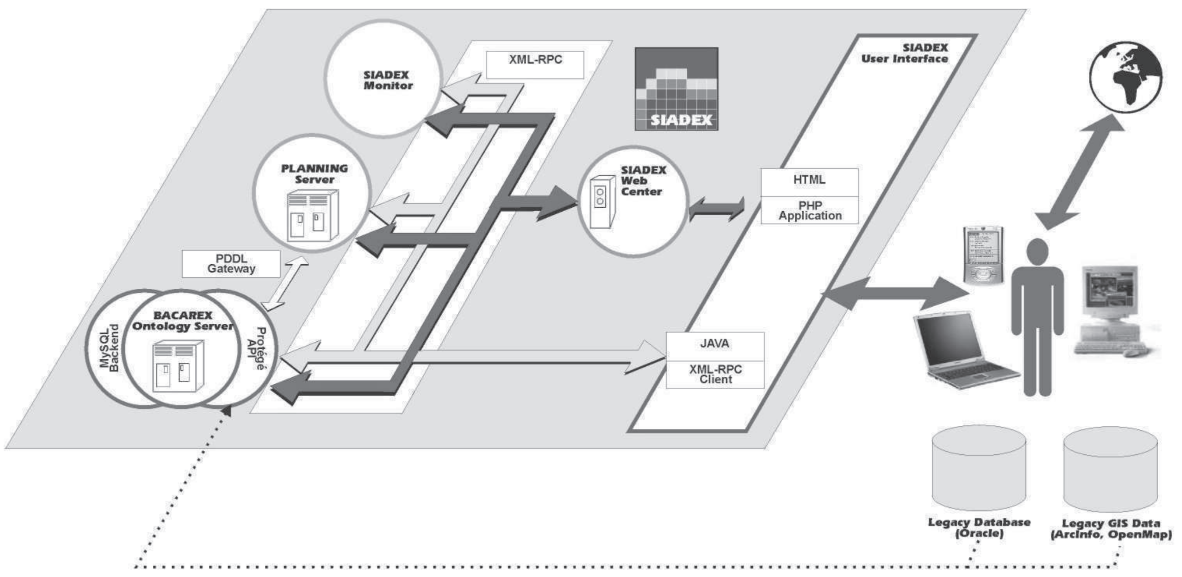


Figure 1: The architecture of SIADEX

This also allows the user to communicate with the system under a TCP/IP connection (OS independent), through a personal computer, a laptop or even a PDA device. All these operations are coordinated by the *WebCenter* module that interfaces all of the interactions between SIADEX and the outer world: it receives planning requests by the user (through the User Interface), and ask the planning server for new plans ; it gathers information on

[2]XML: Extensible Markup Language http://www.w3.org/XML. XML-RPC: XML Remote Procedure Call http://www.xmlrpc.com

the execution of the plan, and it launches execution orders, or raises alerts about possible execution failures to a human operator (upon notification of the monitor).

As can be seen, this architecture aims to be a distributed, stand-alone planning application, that implies the development of different planning techniques (hierarchical planning, plan monitoring and replanning, user interaction) that have to be integrated with time and resources management (scheduling). In addition, a stand-alone application as SIADEX cannot be built without developing appropiated planning and scheduling knowledge engineering techniques. All these issues wil be discussed in the following sections.

3 BACAREX: the knowledge base of SIADEX

BACAREX is an ontology of planning objects conceived as a standalone module. It is, thus, responsible of providing the knowledge required by the planning process, but it is also an open platform to a continuous update and query by (domain or planning) experts. The ontology has an interface with legacy databases so that the flexibility and efficiency of data storage is guaranteed and multiple users/processes in parallel are also allowed. It also provides both offline and online access facilities. Offline access allow planning experts accessing the knowledge and carry out maintenance and validity checking operations with full operability. Online access is done through a web access service, and it is devoted to domain experts that do not have skills on knowledge representation for planning but may painlessly access the knowledge in a web browsing fashion by means of a hierarchy of objects and activities close to their understanding of the problem.

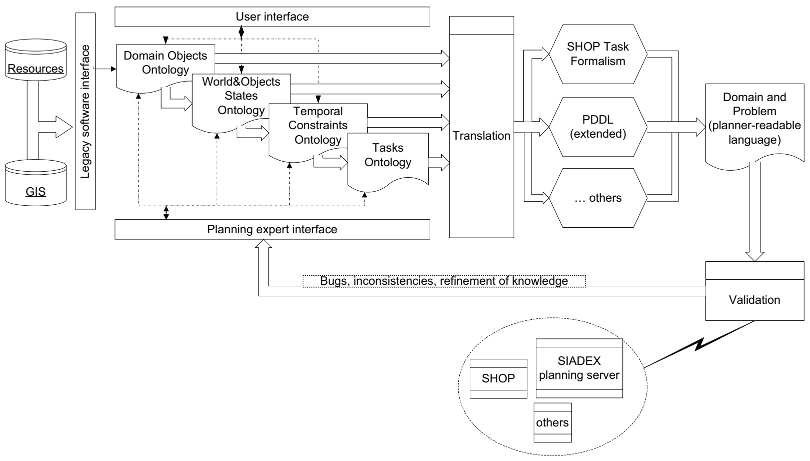
The operating requirements of such a knowledge base, the great amount of knowledge managed and its different categories, requires to consider methodologies and tools as standard and planner-independent as possible. We have realized that it is possible to adapt CommonKADS [16] to model and represent some parts of the knowledge base (later detailed). With respect to choosing a modeling language, this depends also on the existence of a tool for representing and editing the planning knowledge. Although some planning-specific tools and modeling languages can be found in the literature [13, 15, 17] they are either not planner-independent [15, 17], or they do not fit to real-world expressiveness requirements [13]. Thus, for the sake of standarization we have opted for Protégé [3], a widely recognized tool in the field of knowledge based systems development, and we have defined our planning modeling language and planning knowledge acquisition and validation process on this framework.

Thus, the knowledge engineering process that we have carried out has lead to the definition of an ontology of planning knowledge (BACAREX) and a validation process for such ontology. In addition, different actors in the management of this knowledge have been considered: the user (usually a domain expert), the knowledge engineer (a planning expert) and the planner; and different techniques have been developped according to every actor needs and operating requirements.

3.1 Engineering planning knowledge

In the development of BACAREX (see Figure 2) we have had to faced with the modeling, representation and management of several categories of knowledge.

[3]http://protege.stanford.edu. It also allows to easily develop online and offline acces to the knowledge base.



Resources
GIS
Legacy software interface
User interface
Domain Objects Ontology
World&Objects States Ontology
Temporal Constraints Ontology
Tasks Ontology
Planning expert interface
Translation
SHOP Task Formalism
PDDL (extended)
... others
Domain and Problem (planner-readable language)
Validation
Bugs, inconsistencies, refinement of knowledge
SHOP
SIADEX planning server
others

Figure 2: The modeling, representation and validation process followed in SIADEX

Domain objects. Objects (like geographical points, squads, vehicles, etc.) , their properties (like coordinates, number of components, avalaibility) and relations between objects (position, components, etc.) have to be identified, modeled and represented. In addition additional and relevant data like GIS information or weather forecasting already exists in legacy databases (Oracle databases for resource management, and GIS like ArcInfo for cartographic information, are used by technical staff) and they have to be incorporated also as domain objects. This knowledge is modeled and representend in the *domain obects ontology*, a part of the ontology that structures domain objects as a hierarchy of classes and instances. There are standard methodologies, like CommonKADS, that can be adapted [4] for helping to obtain that part of the ontology, but this is not the case for others parts.

World and objects states. Dynamic properties and relations between objects (like position or different states for resources), and their transitions, that are needed to define the dynamics of the domain, and required by the own planning process, have also to be modeled and represented. The *world and objects states ontology* represents the predicates (relations) necessary to define the domain dynamics, and it also allows the definition of deductive axioms in order to derive other states from logic combinations of more simple predicates. Some relations (i.e. predicates) can be automatically obtained, from domain objects slots, but there are other relations that appear to be necessary to represent only when a detailed analysis of actions requirements and effects is done in the validation stage.

Temporal and resources (scheduling) constraints. Apart from the necessary time representation for every dynamic property or relation, there is also information about weather forecasting (temperature, humidity and wind speed and direction) with attached time constraints. In addition, the use of resources is subject to legal temporal regulations, thus one has to consider: periods of availability of resources, legal safety constraints (maximum number of hours

[4]See [9] for a detailed description about adapting CommonKADS to the development of SIADEX

of flight for aricrafts), and schedule of the shifts of workers, by contracting agreement. The *temporal and resources constraints ontology* is devoted to this category of knowledge, and it associates time constraints (time points or intervals) to properties and relations described in the previous parts of the overall ontology.

Actuation guidelines and standard operating protocols. This category includes knowledge about the tasks that have to be accomplished in a fire fighting episode. These tasks are classified as strategy, logistics, attack, deployment and withdrawal procedures. In addition to this tasks knowlege, heuristics about the use of resources and conditions about the use of procedures in specificic situations have also to be represented. This knowledge is modeled and represented in the *task ontology* , that is defined as a class-hierarchy of tasks where the representation of tasks follows a HTN standard: tasks may be primitive or compound, every compound task is associated to a set of methods that represents different ways to accomplish a task, and every method is a collection of (primitive or compund) tasks [14]. Tasks (compound or primitive) are represented with a name, arguments whose types are extracted from the domain objects ontology, and temporal constraints (a duration, and start and end time points). Primitive tasks contains preconditions and effects, represented by standard logical expresions. Every method, represented as a (possible unordered) task list, also contains a precondition that defines their application conditions (useful to encode user heuristics).

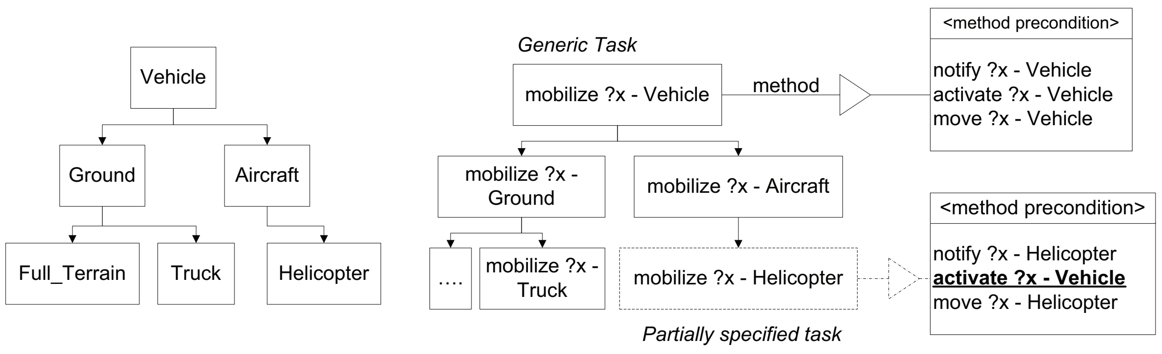


Figure 3: Tasks hierarchy with partially specified tasks

Apart from this basic representation, tasks are organized as a class hierarchy following the idea of an *object oriented foundation class*. This allows a planning expert to carry out knowledge management operations as the definition of generic tasks performed on abstract resources, and the specialization of such generic tasks performed on more concrete resources. The use of this kind of ontology has several advantages since it allows to maintain tasks even if they are partially specified. These kind of tasks are useful to describe cases in which the knowledge elicited (form documents or experts) is not at a sufficient level of detail to be considered fully operational; but it might be introduced to the planner server in a validation process in order to be refined by the interaction with domain and planning experts.

Though part of this modelling language is inspired by the HTN paradigm, it is important to notice that it is not a planner-readable language, and that it supports several modelling operations that are not offered by standard planning languages, but necessary if one want to develop a real-world planning application. These functionalities include: task organization by subject (strategy-definition tasks, logistics operations tasks, deployment tasks, attack tasks, withdrawal tasks, etc.), knowledge inheritance operations (tasks specialization by inheritance of either methods or precondition and effects) and generic tasks management. This splits knowledge engineering operations (domain modelling and validation) from proper planning operations.

3.2 Planner-independent validation

The knowledge stored in BACAREX is not directly recognizable by the planner module. This is not a drawback, but an added value. This allows to define a planner-independent validation process, which starts from a translation of the domain modeling and representation language to a planner-readable language. At present, we have developed some "plugins" that translates the BACAREX knowledge into the task formalism of SHOP [14] and an extension of PDDL [12] able to manage hierarchical tasks. Its main features are the following ones:

- Predicates are esaily translated from the world and objects states ontology, and the classes hierarchy of the domain objects ontology are also translated as a hierarchy of types in the PDDL domain description. Deductive axioms are translated as *derived predicates* in PDDL 2.2.
- Slots in the domain objects ontology that contain numerical values that may change during a planning episode, mainly numerical resources like the level of fuel of a vehicle, are declared like fluents in the PDDL domain, so that the use of arithmetic operations and functions are allowed over them.
- The temporal knowledge related to durations and delays of tasks, like subgoals with deadlines, the maximum makespan allowed for a plan, durations of actions or any other temporal constraints defined between actions, are translated into the formalism described in [4], that extends the expresiveness of the level 3 of PDDL devoted to durative planning actions. Other temporal constraints in the *temporal constraints ontology* (like work shifts or weather forecasting) are translated into *timed initial literals* in PDDL 2.2.
- Primitive tasks are translated as PDDL 2.2 durative actions (with the time formalism extended as described in [4]). Compound tasks, their time constraints, and their associated methods are translated into a hierarchical extension of PDDL (inspired in the SHOP task formalism). In order to avoid a high branching factor in the planner, only the most specific tasks (leafs of the task hierarchy) are translated into the planning domain.

The result of the translation process is a planning domain and/or problem ready to use for a planning engine. This output is used in a validation process based on querys and answers performed against the planning server. These query and answer transactions are defined over a XML-RPC protocol that allows to use any planning engine that supports this protocol. That process allows to detect different kinds of mistakes usually produced in domain modeling and writing: syntactical bugs, semantical inconsistencies, and partially specified tasks.

4 The Planner

The planning module is still under development so we may not show performance or efficiency data here but it is a planner based on the HTN paradigm [11] over the ideas previously developed in our nonhierarchical temporal planner MACHINE [4] and our hybrid hierarchical planer HYBIS [3].

- The initial state is obtained from the world and states ontology by querying the ontology about the main slots of the instances of clases like facilities, human resources, vehicles, water points, etc. The domain is also extracted from the ontology, through the translation process explained above.
- The goal is interactively defined as a situation asessment of the episode made up of instances of the ontology that describe the desires of the technical staff. Then, the goal of a problem is defined in the following terms (by extending the PDDL representation of goals):
  - Geographical deployment of the episode (GIS information): situation of the episode, main fire lines, main focuses, water discharging areas, defense lines (grouped by sectors each of which has a fighting director that is responsible of it), command area and waiting areas.
  - All the items related to fire fighting activity have a slot that defines its intensity, that is, a numeric evaluation of the power of resources that should be assinged to it and that is fixed by hand by the fire fighting director. From the intensities, the planner may pre-select several combinations of resources, that are submitted to the fight director who finally selects one of them. Later, the planner will select a specific set of instances of resources that fits within the selected combination of resources.
  - The fire director also assigns different tasks to be carried out at each item with fire fighting activity (among the available tasks in the ontology) and, since every task is preconditioned with the type of resource that is able to develop it, the planner automatically assigns resources to tasks.

Finally, thanks to the expressive power of using Temporal Constraints Networks as the underlying formalism to represent temporal knowledge, the planner is able to obtain approximate temporal plans, that are plans whose timeline is flexible and may be scheduled in several different ways to adapt to unexpected delays, but this is discussed in the following section (for more details, see [4, 5]).

5 Monitoring and Replanning

Up to now, we have described a batch process that starts with the knowledge stored in the ontology server and ends with the raising of one (or more) approximate temporal plans ready for execution. This section describes the monitoring process and the replanning process (the role of the replanner is also played by the same planning module since it is an incremental planner able to plan an replan over the same episode [10]) whose interaction is sketched in Figure 4 and described below.

The monitoring algorithm [4, 10] is a real time algorithm that follows the execution of the temporal plan at the highest level of detail since temporal plans still represent the time at which every effect of every action is achieved. It checks that everything executes as predicted, otherwise, it may detect and, in some cases repair, some of the problems[5]. The type of problems and their possible solutions are the following ones:

[5]Clearly, the user may also interrupt the execution of the plan at any moment.

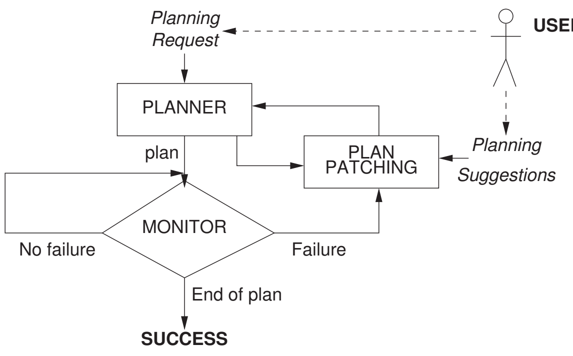


Figure 4: The monitoring user-centered replanning processes

Unexpected delays. These are detected when an action exceeds the time predicted to achieve one of its effects. In this case, it must be taken into account that a temporal plan encodes many temporal constraints between its actions like deadlines, relative constraints and time windows for execution and that most of these temporal constraints come from the cause/effect relationships between actions in the plan. For example, let us consider that action a_1 executes at time t and it produces an effect e at time $t + \delta_e$. Let us suppose again that action a_2 requires the effect e to be true before its execution and that it must be executed before the deadline $t + \delta_{a_2}$. This means that action a_2 must be necesarilly executed in the time window $[t + \delta_e, t + \delta_{a_2}]$ and that any delay in the achievement of the effect e would also delay a_2 and even more, it might put in risk a correct execution of action a_2. Therefore, there may be three different situations.

- Local delays. They are delays that only affect locally to an isolated branch of the temporal plan without affecting the remaining actions. This is the easiest case since it could not affect deadline goals or makespans which depend on more global temporal constraints. In these cases, a new reschedule may be found[6] only for the actions of the affected branch leaving the remaining schedule unaltered.
- Global delays. They are more important since they might affect all of the remaining actions and deadline goals or makespans might also be affected and hence, a whole new reschedule might be needed.
- Infeasible delays. When an action a_j has been delayed beyond its limits, then no reschedule is possible and the possibility of replanning should be considered. In the previous example, if the effect e takes too much time to be achieved (when $\delta_e > \delta_{a_2}$) then a temporal inconsistency would rise and the temporal plan would not be valid.

Execution failures. These situations mean a real fail of execution of an action and they are very serious since, as a consequence of the failure, the cause/effect relationships between actions could have been damaged and the plan could fail to achieve its goal. Therefore, they are situations that necesarilly imply some replanning decisions to be taken and that may be as easy as re-executing the failed action or as complex a re-designing a whole branch of the plan. These execution failures may be due to one of the following reasons:

[6]The rescheduling process is decribed in [8] and it consists of propagating the detected delay among all the actions that have not executed yet and detecting possible inconsistencies in the propagation.

- Missing condition. A condition that was previously achieved by the execution of a previous action has dissapeared. For example, a defense line that was open by a bulldozer to protect an area but that the fire has jumped over it and now the protected area is threatened.
- Missing effect. An effect of an action under execution would never be achieved. For example, when an aircraft breaks down and it is not able to drop a discharge of water that had been previously scheduled.
- Unexpected condition. A condition that was not previously known suddenly raises. For example an unforeseen increase in the speed of the wind that impedes the flight of helicopters.

In any of these cases, the revision and redesign of the failed branch is carried out in a close interaction between the technical staff and the planning module in a "user centered" plan patching episode [10] like that outlined in Figure 4. Its main features are the following ones.

On the one hand, this episode allows the interaction with the user, that is, during this plan patching episode, the user may edit the plan and either delete and suggest conditions or actions to reflect his strategy to resume the execution of the plan or even define a new assesment of the situation by deleting goals and introducing new goals for the planner. This is a very important point since in critic situations, where there may be lives in danger, a completely automated process is not very realistic and the skills of human operators could not be subtitued. After the patching of the failed plan, the own planning algorithm is able to regenerate a new plan adapted to these changes of the user and then subbmitted again to the technical staff for consideration until an appropriate solution is found and scheduled for execution.

On the other hand, this regeneration process is strongly based on local changes made on the failed plan, so that no radical changes are introduced and only the failed branches are redesigned, leaving the remaining of the plan unaltered. This issue is very important since otherwise, a global redesign of the plan could produce dramatic changes on the resources and their tasks and a chaotic migration from the older plan to the new one that would be unrealistic.

6 Final remarks

Specifically in the field of forest fire fighting there are several approaches in the literature. PHOENIX [6] (1989-1993) or CHARADE [1] (1992-1995) are good examples, but they have failed in their application as assistants to real fire fighting scenarios. The reasons for these unsuccessful approaches are: (1) They have neglected the development of appropiated planning knowledge engineering techniques; (2)The lack of deliberative planning techniques able to generate new plans for a situation without the requirement of having a predefined skeleton or general plan previously stored, monitoring of plan execution and replanning techniques to allow flexibility and responsiveness in the execution of the plan; (3) They consider user interaction at edition level, not allowing users and the planner to collaborate in the resolution of the same problem.

The knowledge modeling and validation process here described has to be considered as a step forward in the development of a standard tool, based also on a widely recognized representation language for engineering planning knowledge. In addition, the temporal reasoning framework described is devoted to primitive tasks, more work have to be done in order to fully extend the Temporal Constraint Networks reasoning process to a hierarchical planning algorithm.

References

[1] P. Avesani, A. Perini, and F. Ricci. Interactive case-based planning for forest fire management. *Applied Intelligence*, 13(1):41–57, 2000.

[2] B.Schattenberg and S.Biundo. On the identification and use of hierarchical resources in planning and scheduling. In *Proceedings of the 6th International Conference on AI Planning and Scheduling, Toulouse, France, April 2002, AAAI Press, Menlo Park, California*, 2002.

[3] L. Castillo, J. Fdez-Olivares, and A. González. On the adequacy of hierarchical planning characteristics for real world problem solving. In *Proc. of VI European Conference of Planning*, 2001.

[4] L. Castillo, J. Fdez-Olivares, and A. González. A temporal constraint network based temporal planner. In *Workshop of the UK Planning and Scheduling Special Interest Group, PLANSIG 2002*, pages 99–109, 2002.

[5] L. Castillo, J. Fdez-Olivares, and A. González. Some issues on the representation and explpoitation of imprecise temporal knowledge in an AI planner. In *Knowledge-Based Intelligent Information and Engineering Systems*, Lecture Notes in Artificial Intelligence, LNAI-2774, pages 1321–1328. Springer-Verlag, 2003.

[6] P.R. Cohen, M.L. Greenberg, D.M. Hart, and A.E. Howe. Trial by fire: understanding the design requirements for agents in complex environments. *AI Magazine*, 10(3):32–48, 1989.

[7] M. de la Asunción, L. Castillo, J. Fdez-Olivares, O. García-Pérez, A. González, and F. Palao. SIADEX: Assisted Design of Forest Fire Fighting Plans by Artificial Intelligence Planning techniques. `http://siadex.ugr.es`, 2003.

[8] M. de la Asunción, L. Castillo, J. Fdez-Olivares, O. García-Pérez, A. González, and F. Palao. Handling fuzzy temporal constraints in a planning framework. Submitted to the special issue of Annals of Operations Research on Personnel Scheduling and Planning, 2004.

[9] M. de la Asunción, L. Castillo, J. Fdez-Olivares, O. García-Pérez, A. González, and F. Palao. Siadex: A real-world planning approach to forest fire fighting. In *STAIRS 04*, 2004.

[10] Marc de la Asunción, Luis Castillo, Juan Fernández-Olivares, Oscar García-Pérez, Antonio González, and Francisco Palao. Local (human-centered) replanning in the SIADEX framework. In L. Castillo and M.A. Salido, editors, *Conference of the Spanish Association for Artificial Intelligence, II Workshop on Planning, Scheduling and Temporal Reasoning*, pages 79–88, 2003.

[11] K. Erol, J. Hendler, and D. Nau. UMCP: A sound and complete procedure for hierarchical task-network planning. In *AIPS-94*, 1994.

[12] D. Long and M. Fox. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research*, 20:61–124, 2003.

[13] T. L. McCluskey, D Liu, and R. Simpson. GIPO II: HTN planning in a tool-supported knowledge engineering environment. In *Proceedings of the International Conference on Automated Planning and Scheduling, June, 2003 AAAI press. (ICAPS'03)* , 2003.

[14] D. Nau, T. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman. SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research*, 20:370–404, 2003.

[15] H. Mu noz Avila, D. W. Aha, L. Breslow, and D. Nau. HICAP: An interactive case-based planning architecture and its application to noncombatant evacuation operations. In *Ninth Conference on Innovative Applications of Artificial Intelligence*, pages 879–885. AAAI Press, 1999.

[16] G. Schreiber, H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. Van de Velde, and B. Wielinga. *Knowledge Engineering and Management – The CommonKADS Methodology*. The MIT Press, 1999.

[17] A. Tate, B. Drabble, and R. Kirby. O-PLAN2: An open architecture for command, planning and control. In M. Zweben and M. Fox, editors, *Intelligent scheduling*. Morgan Kaufmann, 1994.

A middle-ware for the automated composition and invocation of semantic web services based on temporal HTN planning techniques

Juan Fdez-Olivares[1], Tomás Garzón[2], Luis Castillo[1], Óscar García-Pérez[2], and Francisco Palao[2]

[1] Dpto. Ciencias de la Computación e I.A., University of Granada, SPAIN
[2] IActive Intelligent Solutions `http://www.iactive.es`

Abstract. This work presents a middle-ware able to translate OWL-S web services descriptions into a temporal HTN domain in order to automatically compose and execute sequences of web service invocations, including parallel branches and complex synchronizations, based on the combination of HTN planning and temporal reasoning techniques.

1 Introduction

Semantic web services (SWS) techniques [11] support the way in which already existing "syntactic" web services (usually described in WSDL [10]) can be extended with a semantic layer in order to be automatically discovered, composed and invoked.The main goal of such representation is to provide a logical framework in order for a software system to be capable of both *interpreting* SWS descriptions and, given a service request, *reasoning* about them in order to automatically compose and execute a sequence of web service invocations that provides the resquested service. The main long-term goal of SWS is getting the Semantic Web in its full potential by semantically annotating both data and web processes, but there is a short-term goal that is concerned with the semantic annotation of business web services deployed on service oriented enterprise architectures. In this case SWS may help to leverage the Business Process Management (BPM)[2] life cycle in which *business processes* (workflow schemes designed to specify the operation of business web services) of a company are at present manually composed and orchestrated in order to be executed by standard commercial BPM engines. The application of SWS techniques may lead to a more agile and flexible BPM life cycle by automating the composition and orchestration of business processes while keeping human intervention to a minimum.

Although there are several standard proposals of SWS languages [1], OWL-S [4] is a language that may serve to this purpose for two reasons: firstly, it allows to represent web services as *processes* with typed input/output parameters, preconditions and effects, as well as their data model on the basis of an OWL ontology. And, second, the core concept of OWL-S is the *Process Model*: an

OWL ontology that allows to describe both the semantics of web services as a compositional hierarchy of atomic (that represent already existing WSDL web services) and composite processes (that represent high-level services), and the operation of every composite process as a *workflow scheme* that specifies both order constraints (by using **sequence, unordered, split, and join** structs) and the control flow logic (by using conditional, **if-then-else**, and iterative, **while and repeat-until** control structs) that sub-processes should follow in order to obtain an executable sequence of web services.

From the point of view of OWL-S, web service composition consists on finding a suitable sequence of atomic processes (that is, web services invocations) that provides a high-level service-request expressed as a composite process, and considering the process model as a *guideline* to be followed by the composition process. Regarding the automated composition of web services as OWL-S processes, although several techniques may be applied to this problem [12], AI Planning and Scheduling (P&S)[7] seems to be the most promising one since during the last 40 years it has dealt with the development of planning systems capable of interpreting a planning domain as a set of actions schemes (i.e. a process model) and reasoning about them in order to compose a suitable plan (i.e. a sequence of actions) such that its execution reaches a given goal (that can be seen as a service request) starting from an initial state. Concretely, HTN planning [5, 3] becomes the most suitable AI P&S technique since it supports the modeling of planning domains in terms of a compositional hierarchy of tasks representing compound and primitive tasks by describing how every compound task may be decomposed into (compound/primitive) sub-tasks and the order that they must follow, by using different methods, and following a reasoning process that is guided by the knowledge encoded in the HTN domain.

Therefore, considering this previous discussion, in this work we present a middle-ware able to both *interpret* OWL-S web services descriptions, by translating them into an HTN domain and problem description, and *carry out a reasoning process* based on HTN planning techniques in order to automatically compose a sequence of executable web services. This sequence is obtained by following the workflow scheme defined in the OWL-S process model and provides a high-level service request introduced as a problem. The cornerstone of this architecture is SIADEX, an own developed HTN planner [3, 6] that receives as input an HTN domain automatically translated from an OWL-S process model and a planning problem representing both, a goal extracted from a high-level service request, and an initial state extracted from the instances of the OWL-S' underlying OWL data model. In the following section we will describe in detail the main features of SIADEX that make it suitable in its application to web service composition as well as its related aspects with the OWL-S Process Model. Then, a mapping algorithm that translates a OWL-S Process Model into a SIADEX domain will be illustrated. Finally, the architecture of the middle-ware will be shown and we will briefly describe a service oriented enterprise application in a simulated scenario where this middle-ware has been tested in order to interpret and execute business processes modeled as OWL-S processes.

2 SIADEX in a nutshell

SIADEX is an AI Planning and Scheduling system that uses as its planning domain and problem description language an HTN extension of the PDDL standard in such a way that primitive tasks are encoded as PDDL 2.2 level 3 durative actions (see [3] for details). In addition, methods used to decompose tasks into sub-tasks include a precondition that must be satisfied by the current world state in order for the decomposition method to be applicable by the planner. The basic planning process of SIADEX is a state-based forward HTN planning algorithm that, starting from the initial state and a goal expressed as a high-level task, iteratively decomposes that top-level task and its sub-tasks by selecting their decomposition methods according to the current state and following the order constraints posed in tasks decomposition schemes as a search-control strategy (See Figure 1).

(a)

```
(:derived (vip_user ?u - User)
  (and (> (salary ?u) 3000)
          (genre ?u F)))
(:task GetPrice
 :parameters (?c - Car ?u - User)
 (:method is_vip_user
  :precondition (vip_user ?u)
  :tasks (
          (getPrice ?c ?p)
          (getDisccount ?c ?d)
          (final_price ?p ?d ?f)
          (:inline () (price ?c ?u ?f))))
 (:method not_vip_user
  :precondition (not (vip_user ?u))
  :tasks (
          (getPrice ?c ?p)
          (final_price ?p 0 ?f)
          (:inline () (price ?c ?u ?f)))))
```

(b)

```
(:derived (vat_applied ?p ?v)
   {import math
    ?v = ?p * 1.16
    return 1}))

(:durative-action final_price
 :parameters(?p ?d ?f - number)
 :duration (= ?duration 3)
 :condition(and (vat_applied ?p ?v)
 :effect(and (assign ?f (- ?v ?d)))))

(:task while-loop
  :parameters (?x - Number)
  (:method base case
   :precondition (and not (> ?x 0))
   :tasks ())
  (:method loop
   :precondition (and (> ?x 0))
   :tasks ((do-something ?x)
           (:inline ()
              (assign ?x (- ?x 1)))
           (while-loop ?x))))
```

Fig. 1. The basics of HTN planning domains in SIADEX' domain language: (a) A derived literal inferring whether a given user is or not a vip user and a compound *task* with two different *methods* of decomposition, describing how to compute the price of a car depending on the profile of a given user. The decomposition method uses an *inline task* to assert in the current state the price of the car for that user . (b) A primitive *action* that computes a final-price with discount, it is preconditioned with a derived literal that infers, by using a Python script, the VAT applied to a price. The task `while-loop` exploits the capability of recursive decompositions in order to describe repetitive tasks.

This process makes possible to know the current state of the world at every step in the planning process and, concretely, when preconditions of both methods and primitive actions are evaluated, what allows to incorporate significant inferencing and reasoning power as well as the ability to call external programs (that in this case might be web services) to infer new knowledge by requesting information to external sources. For this purpose, SIADEX uses two mechanisms: on the one hand, deductive inference tasks of the form `(:inline <p> <c>)` that may be fired, in the context of a decomposition scheme, when the logical expression `<p>`(condition) is satisfied by the current state , providing additional bindings for variables or asserting/retracting literals into the current state, depending on the logical expression described in `<c>`(consequent); on the other hand, abductive inference rules represented as PDDL 2.2 derived literals of the form `(:derived <lit> <expr>)`, that allow to satisfy a literal `<lit>`when it is not present in the current state by evaluating the expression `<expr>`that may be either a logical expression or a Python script that both binds its inputs with variables of that literal and returns information that might be bound to some of the variables of `<literal>.`This one is a crucial capability since, as it will be detailed in the following sections, supports the way in which SIADEX interacts with external web services.

Furthermore, the domain description language of SIADEX and the planning algorithm support to explicitly represent and manage time and concurrency in both compound and primitive tasks, thanks to the handling of metric time over a Simple Temporal Network (See [3] for more details). This temporal representation provides enough expressivity power to represent OWL-S workflow schemes such as sequence, unorder, split and join. Finally, the search control strategy followed by SIADEX allows to represent other patterns like conditional or iterative control constructs, giving support to fully represent an OWL-S process model. This will be seen in next section where the translation process from OWL-S to the domain description language of SIADEX is illustrated.

3 Mapping an OWL-S process model into a SIADEX domain

Mapping Overview.The translation process first maps the OWL data model into the PDDL data model by translating OWL classes, properties and instances into PDDL types, predicates and objects, respectively [3].Then it maps the OWL-S process model into a SIADEX HTN domain that represents the operation of both atomic and composite processes as primitive tasks and task decomposition schemes, respectively. Atomic processes are mapped as PDDL durative actions (see below) and the workflow pattern of every composite process is mapped into a method-based task decomposition scheme that expresses the operational semantics of the control structs found in that composite process. In that section it will be shown how the mapping process exploits (1) the order between sub-tasks in order to represent sequence and unordered control structs, (2) the management of temporal constraints to represent split, join and split-join of processes, and (3) the search control used to decompose tasks in order to represent conditional structs and the possibility of describing recursive decompositions as the basis to represent iterative control structs. However, firstly we will start by illustrating how to map an atomic OWL-S process into primitive actions managed by SIADEX.

Atomic processes as PDDL durative actions.The header of an atomic process (i.e. its name, input and output properties, See Figure 2) is directly

[3] Space limitations prevents detailing this ontology mapping process, although a similar one on a frame-based ontology is described in [6]

mapped into the header of a PDDL durative-action with typed parameters (these types correspond to classes of the OWL data model). This is also the case for preconditions/effects, since there is also a direct correspondence between expressions inside preconditions and effects of any atomic process and preconditions/effects of PDDL actions[4].

This direct translation works for world-altering only atomic processes (i.e. only alter internal states in processes) and that don't need to manage external information. However, atomic processes might be associated to WSDL information-providing web services in such a way that atomic process' outputs might be filled with information sent back by the web service once it has been invoked. This real need reveals as a key issue the management of information gathering at planning (i.e. composition) time since a considerable part of the knowledge needed by SIADEX to reason about methods and primitive actions' preconditions might not be contained either in the initial or the current state, but accessible from external information sources. In this case it becomes neces-

```
<process:AtomicProcess rdf:ID="#GetPrice">
<process:hasInput rdf:resource="#CarModel"/>
<process:hasInput rdf:resource="#User_Car"/>
<process:hasOutput rdf:resource="#Price"/>
<process:hasPrecondition rdf:resource="#AlwaysTrue"/>
<process:hasEffect> Price(User_Car Price) />
</process:AtomicProcess>
```

Fig. 2. An OWL-S atomic process that returns the price of a car

sary to represent in the SIADEX domain both, the atomic process structure and the web service invocation, since it will be needed to obtain information at planning time. This is done by translating the correspondence (defined in the *Service Grounding*) between the atomic process and the WSDL service as an inference rule represented as a *derived literal*. This inference rule has the general form `(:derived <header> <call>)` where `<header>` is a literal automatically generated from the header of the atomic process (that corresponds with the "header" of the WSDL web service) and `<call>` is a Python script (also automatically generated) that invokes the web service by passing the input parameters of the composite process and provides a bind for the output parameters with the information sent-back by the web service. Finally, the literal `<header>` is added to the preconditions of the PDDL action in order to be evaluated when the action is applied in the planning process. The example shows how the correspondence be-

```
(:derived (d_getPrice ?m ?c ?p)
 {i1 = ?m
  i2 = ?c
  <invoke wsdl#getPrice i1 i2 o1>
  ?p = o1})
```

```
(:durative-action ActiongetPrice
  :parameters (?m - Model ?c - Car ?p - Number)
  :precondition (d_getPrice ?m ?c ?p)
  :effect (Price ?c ?p))
```

Fig. 3. The correspondence between the atomic process `GetPrice` and the web service `wsdl#GetPrice` described as a primitive action and a derived literal, respectively

[4] We have used the Protege OWL-S tab plug-in for editing OWL-S process models. This tab allows to represent preconditions an effects in several formats like SWRL or KIF. We have chosen to represent them as strings with the KIF format, a similar representation to the one of PDDL.

tween the atomic process `getPrice(input:Model input:Car output:Price)` and it associated WSDL web service is translated into a derived literal which is added to the preconditions of the action `ActionGetPrice` corresponding tho the atomic process. This representation allows, on the one hand, to bind the variable `?p` (for price) with a value coming from an external source through a web service invocation and, on the other hand, when the action is applied to the current state, to incorporate this binding at planning time when asserting the effect of the action in the current state.

Composite processes and management of time-order constraints.The translation algorithm maps every OWL-S composite process and its sub-processes into a SIADEX task decomposition scheme where sub-tasks (corresponding to sub-processes) are structured in methods depending on the control structure that is modeled in the composite process. This is done in two steps: firstly, process parameters are mapped into task parameters as in atomic processes, process preconditions are added to the preconditions of every method of the corresponding task and, since HTN domain descriptions do not allow to explicitly describe effects in a tasks, process' effects are mapped into an *in-line inference task* of the form `(:inline () <consequent>)` where `()` stands for an empty condition part (representing a condition that is always true) and the consequent contains the logical expression of the effects of that composite process. This allows to assert, at planning time, the effects (if any) of the translated composite process in the current state. The second step considers the control struct of the composite process making a distinction between control structs that define the execution order between processes (sequencing, unordering, splitting or join), and those that define the control flow logic of processes (conditional an iterative ones.)

```
(:task Purchase
  :parameters (?m - CarModel ?user_car - Car)
  (:method
   :precondition ()
   :tasks ((getAvailability ?m)
          [(getPrice ?m ?user_car ?p)
              (getDiscount ?m ?user_car ?d)]
           (bookCar ?m ?user_car)
           (payCar ?user_car))))
```

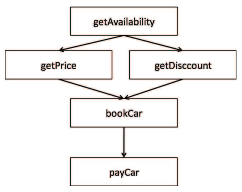


Fig. 4. The decomposition scheme of action `Purchase` and its associated plan implementing a split-joint construct

In the former case the translation process generates one single method that expresses such control structures by describing order constraints between its component sub-tasks. For this purpose SIADEX allows sub-tasks in a method to be either sequenced, and then their signature appears between parentheses (T1,T2) , or splitted, appearing between braces [T1,T2]. Furthermore, an appropriate combination of these syntactics forms may result in split, join or split-join control structs. For example, the decomposition method of task purchase `(?m - Model ?user_car - Car)` in Figure 4 specifies that, in a plan for the composition of web services, in order to purchase a car of a model, before to invoke booking and paying web services, it is firstly necessary getting the availability of cars of that model and, concurrently, obtaining the price and discount of that model.

Current state-based forward planners (HTN and non-HTN, like SHOP2[5] or OWLSXPLan[8]) with application to web service composition lack of the required expressivity for representing web services execution paths as the one shown in the previous example. The reason is that these planners return plans as a totally ordered sequence of actions and, as opposed to them, SIADEX is capable of obtaining plans with true parallel branches of execution due to the handling of metric time over a Simple Temporal Network (STN). At planning time, SIADEX deploys its partially generated plan over a STN that associates a pair of start and end time-points to either every compound or primitive task. All the time points and constraints of the STN are posted and propagated automatically, observing the order constraints defined in the decomposition scheme, every time that a compound or primitive task is added to the plan. Therefore, the control construct initially modeled in OWL-S contains implicit temporal constraints that, when translated, are automatically explicited and managed by SIADEX. This is a clear advantage of SIADEX with respect to other approaches, since despite OWL-S does not support time-related information in processes, the planning process of SIADEX is aware of these temporal constraints between processes, and capable of automatically manage and infer them from qualitative order relations like those above illustrated.

Conditional and iterative control constructs are translated into task decomposition schemes that exploit the main search control technique of SIADEX. Briefly, a composite process *p* that contains a conditional struct ***if*** *c* ***then*** *p1* ***else*** *p2* is translated into a task decomposition scheme `(:task` *p* `(:method :precondition c :tasks (`*p1*`)) (:method :precondition (not c) :tasks (p2)))` describing that if *c* holds in the current state then decompose the task *p1* else decompose the task *p2*. A composite process *p* that contains an iterative struct ***while*** *c p1* is translated into a task decomposition scheme `(:task` *p* `(:method :precondition` *(not c)* `:tasks ()) (:method :precondition c :tasks (`*p1 p*`)))`, describing that the task *p1* should be repeatedly performed (and so recursively decomposed) while *c* holds in the current state.

Finally, it is important to recall that given an OWL-S process model and its associated service grounding the translation process [5] above described allows to automatically generate a planning domain represented as a hierarchical extension of PDDL2.2, capable of representing information providing actions by invoking external web services, which is fully ready to use (without human intervention) by SIADEX in order to solve problems of web services composition. On the basis of this translation process we have also developed an architecture for the dynamical composition and execution of semantic web services described in OWL-S, that is shown in the next section.

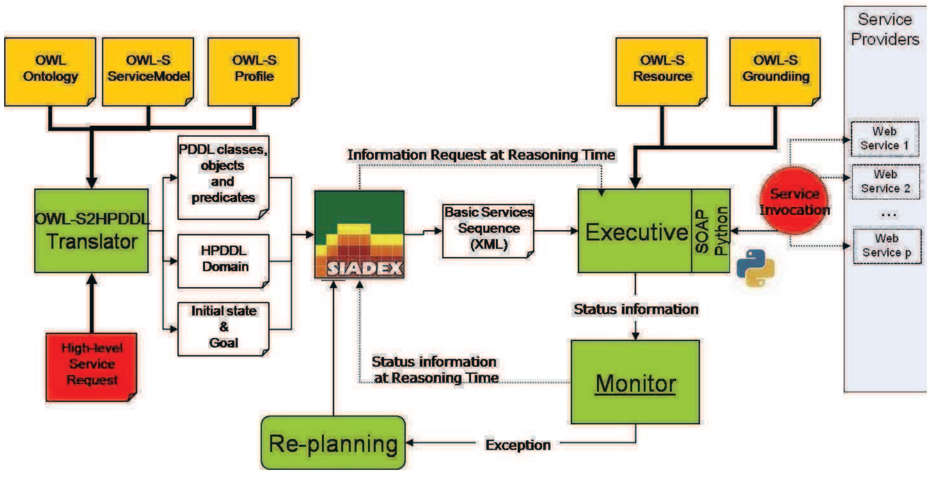


Fig. 5. A middle-ware where an HTN planner (SIADEX) plays the role of a web services composer for the automated composition of OWL-S semantic web services

4 Middle-ware for the composition and execution of web services

Figure 5 shows the architecture of the middle-ware here presented able to both *interpret* OWL-S web services descriptions, by translating them into an HTN domain as explained in the previous section, and *carry out a reasoning process* based on HTN planning techniques in order to automatically compose and execute a sequence of executable web services. This sequence is obtained by following the workflow scheme defined in the OWL-S process model and provides a high-level service request introduced as a problem. The proposed architecture has the following components: a **Translator** that maps an initial service-request (through a java interface) into both, an HTN goal (represented as a high-level task that is an instance of a composite process already modeled in OWL-S), and an initial state which is made from OWL instances of the OWL-S data model (any way, most of the information needed to planning resides in external sources and recall that the planner can access to it by means of web services invocations). The problem together with the translated OWL-S process model are sent to the **Web Services Composer**(SIADEX), the cornerstone of this architecture, in order to start the composition (planning) process. Then the planner makes use of the knowledge encoded in the domain (representing the OWL-S process model) as a guide to find a sequence of temporally annotated primitive actions that represents a suitable composition (with possibly parallel branches) of atomic processes. This sequence is sent to the **Monitor** that is in charge of both scheduling the execution of atomic processes according to their temporal information and sending execution orders to the **Executive**. This module is in charge of executing web services invocations (requested at either planning or

[5] The sources and java .jar files can be downloaded from http://decsai.ugr.es/˜faro/OwlsTranslator

plan execution time) and sending back the information. At planning time, the requested information may result in a fail when the web service requested is not available or the information returned gets a precondition unsatisfied. In that case SIADEX performs a backtracking process that may lead to select a different web service (thus carrying out a form of web service discovery) or even a completely different way to compose the high-level service (if so encoded in the OWL-S process model). At execution time, the execution of a web service might raise an exception the notification of which is sent to the Monitor that raises a **Re-planning process**. This module is in charge of manage the uncertainty when executing web services and at present is in development, but it is being designed in order to fastly, locally repair the composed sequence. Any way, at present and in case of an exception is raised, the Monitor informs to the user that the service requested is unfeasible and a new composition episode is initiated.

This middle-ware has been tested in the framework of a geographically distributed, service oriented enterprise application devoted to car sales. In this simulated scenario the above described middle-ware plays the role of a semantically extended Business Process Engine that bridges the gap between business process modeling and web services execution phases, by automatically interpreting, composing and enacting them. Furthermore, apart from this application we have developed a standalone Protégé plug-in (called ProblemEditor [6]) in order to locally edit, visualize and test HTN planning problems and domains automatically extracted from an OWL-S process model.

5 Related Work

Regarding the application of AI P&S to the composition of OWL-S web services in [5] can be found a translation process from OWL-S to SHOP2 domains that inspired the work here presented. Nevertheless, SHOP2 authors neglect the management of temporal constraints what prevents to translate fully OWL-S process models containing split and join constructs, what limits its real application to web composition problems as the ones here faced by SIADEX. Furthermore, in [3] we show a detailed experimentation proving that SIADEX clearly outperforms SHOP2. OWLSXPlan [8] is a planner that faces the composition of OWL-S *service profiles* with non-HTN planning techniques what makes impossible to interpret full OWL-S process models (indeed it is focused on automatically discovering and composing non-hierarchical OWL-S service profiles already published in the web, a different approach to the one here presented). Authors of OWLSXPlan recognize that, due to the absence of temporal reasoning, control structs like *unordered sequence* are not realizable. Recall that, apart from its time performance, the main advantage of SIADEX is the capability of making explicit the management of implicit temporal constraints found in every OWL-S process model, allowing to represent parallel branches of execution as well as complex synchronization mechanisms. Finally [9] translates OWL-S process

[6] This plug-in and the complete OWL-S model can be downloaded from http://decsai.ugr.es/~faro/OwlsTranslator

models into conditional web services sequences ready to be interpreted by standard workflow engines what allows to manage the uncertainty at execution time by establishing conditional courses of execution. However this approach has a high computational cost that might be reduced with the alternative approach here presented that allows both to incorporate and manage external information at planning time and to fast and dynamically repair a sequence that raises an execution exception.

6 Conclusions

In this work we present three significative advances regarding web services composition and its relation with AI Planning and business process management: first, we have introduced a novel and fully automated translation process from OWL-S process models to a hierarchical extension of the PDDL standard that allows a temporal HTN planner to automatically compose and execute OWL-S web services. Secondly, plans obtained represent sequences of web services invocations including parallel and synchronization mechanisms what makes the middle-ware here presented to be considered as an important step forward in the application of AI Planning techniques to real SWS composition problems. Finally, a full application has been developed where business processes are modeled as OWL-S processes that are used to automatically compose and orchestrate business web services of a simulated virtual enterprise. At present we are working in the management of execution exceptions based on an HTN plan repairing process.

References

1. W3c standard submissions. http://www.w3c.org/.
2. Workflow management coalition. http://www.wfmc.org/.
3. L. Castillo, J. Fdez-Olivares, O. García-Pérez, and F. Palao. Efficiently handling temporal knowledge in an HTN planner. In *Proc. ICAPS*, 2006.
4. Martin D., Burstein M., and al. Describing web services using owl-s and wsdl. http://www.daml.org/services/owl-s/1.0/owl-s-wsdl.html, October 2003.
5. Sirin E., Parsia B., Wu D., Hendler J., and Nau D. Htn planning for web service composition using shop2. *Journal of Web Semantics*, 1(4), 2004.
6. J. Fdez-Olivares, L. Castillo, O. García-Pérez, and F. Palao. Bringing users and planning technology together. Experiences in SIADEX. In *Proc. ICAPS*, 2006.
7. Ghallab M., Nau D., and Traverso P. *Automated Planning: Theory and Practice*. Elsevier, 2004.
8. Klusch M., Gerber A., and Schmidt M. Semantic web service composition planning with owl-sxplan. In *Int. AAAI Fall Symp. on Agents and Semantic Web*, 2005.
9. Traverso P. and Pistore M. Automated composition of semantic web services into executable processes. In *International Semantic Web Conference*, 2004.
10. Graham S., Davis D., and al. *Building Web Services with Java*. 2005.
11. McIlraith S.A., Son T.C., and Zeng H. Semantic web services. *IEEE Intelligent Systems*, 2(16):46–53, 2001.
12. Charif Y and Sabouret N. An overview of semantic web services composition approaches. In *"Proc. Int. Workshop on Context for Web Services"*, 2005.

SIADEX: A Real World Planning Approach for Forest Fire Fighting

Marc de la Asunción, Oscar García-Pérez, Francisco Palao.[*]
{*marc, oscar, palao*}*@decsai.ugr.es*

Abstract. This paper describes the SIADEX project (Assisted Design of Forest Fire Fighting Plans by means of Artificial Intelligence Techniques), which is being developed under a research contract with the Andalusian Regional Government. First we give an overview of SIADEX and previous works on this subject. Then, we describe the development methodology we are following, the SIADEX framework, the knowledge base of SIADEX (named BACAREX) and the planning and replanning model we are using. Finally, conclusions are shown.

1 Introduction.

Forest fires constitute one of the biggest threats for natural spaces preservation, and so imply a big worry for any public Administration. Last years, Andalusian public Administration has carried out a big effort on improving the efficacy of the resources devoted to forest fire fighting, leading to the creation of the Andalusian Forest Fire Plan (INFOCA Plan).

We have to confront with a dynamic domain, where decisions must be updated frequently. Plan design time is crucial and all operations have important temporal constraints. Also, the environment is unpredictable and uncertain.

There are several previous intelligent planning applications to forest fire domain. PHOENIX [5] (1989-1993) and CHARADE [6] (1992-1995) are good examples, but they failed in their application as assistants in real fire extinction episodes. The reasons for these unsuccessful approaches are: (1) The lack of deliberative planning techniques, monitoring of plan execution and replanning techniques; (2) They consider user interaction at edition level, not allowing plans assisted design.

The main goal of our project is to develop a system, based on intelligent planning techniques, for the assisted design of forest fire fighting plans (hereafter, SIADEX [3]). SIADEX will improve the current manual process of fire extinction plans design. Also, SIADEX must not be understood as a substitution of the technical staff, but as a tool that improves the skills of an extinction expert, assisting him in an interactive design process of extinction plans. The development of SIADEX will lead to a tool which can be used in two different ways:

- As a virtual learning environment for forest fire extinction. Thus technical staff can acquire skills and experiences in strategies design for forest fire extinction.

[*]Department of Computer Science and Artificial Intelligence. E.T.S. Ingeniería Informática. University of Granada (Spain).

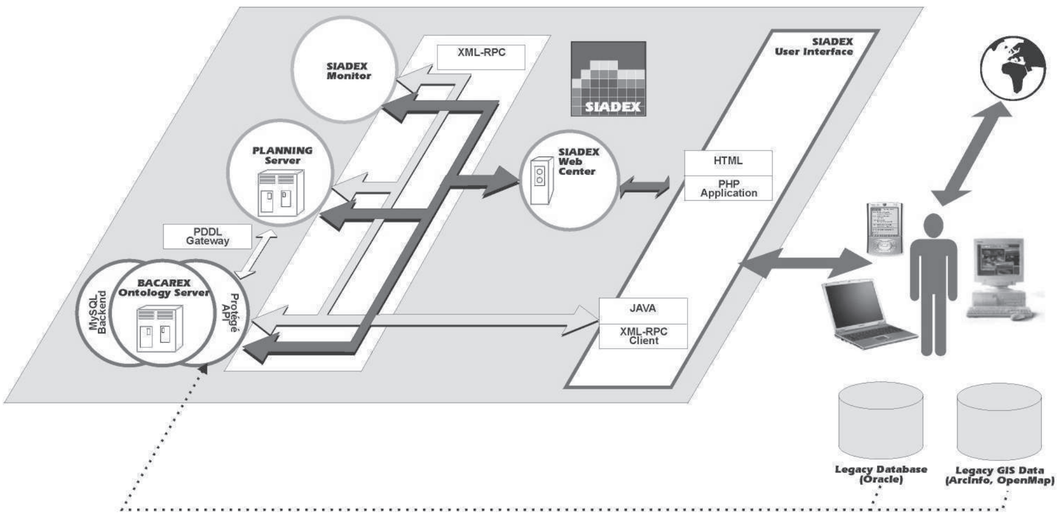


Figure 1: General overview of SIADEX Architecture.

- As a real fire extinction intelligent decision support system (IDSS) based on AI planning techniques. Thus experts can take design decisions of the operation plan from different proposals the system shows him, in function of the knowledge stored in it.

2 The SIADEX Development Methodology.

One of the main issues that we thought carefully before we started the SIADEX project was about the work methodology we had to use to develop our system.

It is well known that there is not an unified and world-wide accepted methodology for developing a planning system. Planning research is a relatively young field. We find a lack of standardized methodologies and tools that may help us to do the task.

So we based our development strategy in a well known and widely used methodology as CommonKADS [1], that is used to develop *knowledge base systems* (KBS). CommonKADS is a compendium of general models, concepts, methods and techniques that can be adapted easily to the process of developing a planning system.

For the project management, we have to develop our system in three phases. In the first phase we have to accomplish the knowledge base (BACAREX) that holds all the information about the domain we are working on (forest fire fighting). In the second phase we will develop the planner itself (SIADEX). And finally, in the last phase, we will develop the interface with other systems including human experts. At the end of each phase we will obtain a different product (see Figure 1) fully functional. We are now at the second stage. One of our main objectives is to maintain each of the modules as independent as we can for the others, so we can reuse the work previously done in other projects with minimum changes. We used, for each module, the life-cycle model proposed in CommonKADS.

3 The SIADEX Framework/Architecture.

Here we present a general overview of SIADEX architecture. We can see the architecture like an interconnected but independent distributed multi-agent system, where each element covers a specific task, that serves as support for the rest of the modules (see Figure 1). The communication between the modules is performed using an API based on the XML-RPC Protocol[1]. We will describe now the main characteristics and functionality of each module:

- Planning Agent: It is the core of the SIADEX architecture, it's a generative planner that obtains extended plans following hierarchical planning techniques. The planning server receives the world description, the initial situation and the goal and gives back a solution.
- Ontology Agent: It serves to the planning agent the world description and the initial situation of objects and resources, and the procedural knowledge needed to build the plan.
- Interface to external databases and applications (World Interface): It gives to the planning agent possible information that is not available in the ontology, like cartographic maps or weather forecasts. It gives the possibility to plug-in other intelligent agents.
- User Interface: The user will have the possibility to communicate with the system through a TCP/IP connection. The overall system is designed to be OS independent, thus the user can use a personal computer, a laptop or even a PDA device.
- Monitor: The plans are then executed under the supervision of the Monitor. The monitor alerts a human operator about the events that are going to happen. The operator can confirm the execution of an operation or its fail. Under faulty conditions it can launch a replanning process to present alternatives for repairing the plan.
- WebCenter: All the system is coordinated by the WebCenter module. It coordinates all of the interactions between SIADEX and the external world: (1) Receives planning requests by the user; (2) Ask the planner for new or repaired plans; (3) Upon notification of the monitoring module, it launches execution orders to the human operator; (4) Gathers information on the execution of the plan by indirect observation of a human operator or by direct gathering and translates them into the Monitor; (5) Raises alerts about possible execution failures upon notification of the Monitor or upon direct user request.

4 BACAREX: The SIADEX Ontology.

BACAREX is the SIADEX subsystem that maintains the ontology that hold all the information that SIADEX needs to prepare plans, for example, information about resources that could be used during the forest fire extinction. The architecture had to be as modular as possible, so it can be reused and flexible. Thus we have to maintain the ontology format and implementation independent. BACAREX architecture has been designed in a client/server fashion (see Figure 1). Its main characteristics are:

- The knowledge must be accessed from any place, any platform, any OS. This was done thanks to the BACAREX/SIADEX distributed architecture. The client application sends and receives the information through an Internet service. The language for writing the client software can be any that supports XML-RPC protocol.

[1]XML-RPC: XML Remote Procedure Call http://www.xmlrpc.com

- PDDL Compliant. It is important to observe that SIADEX receives the world information in PDDL format [10]. We design a PDDL Gateway capable of transforming the ontology representation into PDDL. This guarantees that any planner that is PDDL compliant can be plugged into the system without the necessity of changing the Ontology.
- The ontology needs to be accessible, not only for the researches, but for the final users too. Then a set of tools that provides a friendly user-interface is needed. We developed a web-interface for the final-users. Within this web the users can query, navigate, insert and update the concepts covered by the ontology in a visual fashion.
- Common knowledge. The information is centralized, therefore there will not be data-inconsistence problems.
- Possibility to port the ontology knowledge to a standard Web-Semantic language (e.g. OWL or XML), and merge with other ontologies, using external specialized plug-ins.
- Quick access, because the knowledge is stored in a relational database.

We searched for a language that allows to represent hierarchies of objects, tasks and its associated properties among complex relations between them. We decided to take OWL as our target language. After some experiments we noticed some important conclusions. Because this type of languages evolve and change very quickly: first, the tools developed to give support to this languages usually are under development and are not well tested. This is a big risk for the research of a project. And second, if you write your ontology in one of these languages, you might have the chance that your ontology became technologically obsolete in a short term. So from our point of view is more important to use a widely used and stable tool. We found this tool in Protégé[9]. Protégé is an open-source ontology editor and a knowledge-base editor that has an easy GUI. It has a plugin-based architecture, so you can add complex visualization tools or capabilities for importing/exporting into other ontology representation languages (like OWL). So, at last, our ontology is not written in OWL (but it might be). In fact, for efficiency reasons, it is stored into a relational MySQL database.

We didn't use Protégé itself as a way of access for the final users for a couple of reasons (see Figure 1). First, our experts wanted a more adapted easy-to-use GUI. Second, we need a distributed work architecture, and the capacities of Protégé of working in server-mode are currently under development. We used the API of Protégé to develop a web-access application for the final users.

5 A Planning and Replanning Approach.

In classical planning, the planner assumes to know everything in the world. This type of planners usually doesn't fit well in many real-life problems [4]. In our case, while the plan is executed we have to monitor that the assumed conditions still hold, and that the actions have been carried out correctly. The planning and monitoring processes have to be executed interleavedly, or even concurrently when the plan is so complex that we begin the execution before it has been completely instantiated. The planning system needs to respond in a short period of time to a faulty condition, even repairing the current plan or also launching a parallel new planning process to solve it.

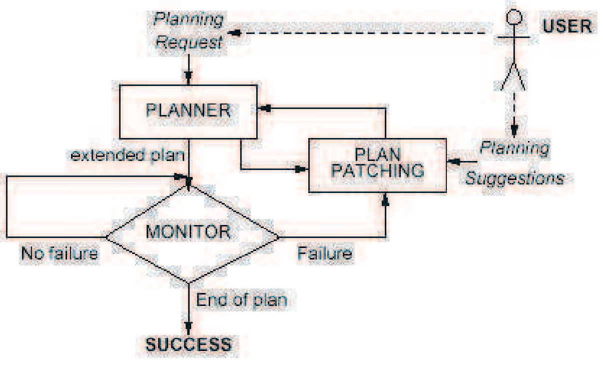


Figure 2: The mixed-initiative planning and replanning loop in SIADEX

The monitoring module of SIADEX is based on a temporal scheduling and rescheduling policy over temporal plans [8] so that actions in a plan are continuously being selected for execution following the best temporal ordering, their execution is monitored and possible faults are detected (Figure 2). When a fault is detected its impact is calculated, and a replanning episode starts in which the user may interact with SIADEX, making some suggestions (delete or add actions by hand, add or delete goals or literals). These suggestions are processed by the planning algorithm and new interactions are requested to the user until a valid plan is obtained, that is scheduled again for its execution and monitoring.

The environment is dangerous, there are risks to lose resources or even human lives. So the experts are rather reluctant for an autonomous process to take the control over all the planning decisions. Also they have expertise knowledge that the planner lacks. We had to elaborate a planner system that can collaborate with the experts, and assist them not as an autonomous closed system but in a *mixed-initiative* [2] way. Thus the experts get involved in the planning process: they share, propose, reject and discuss decisions and objectives with the system in order to achieve a common goal.

Human experts tend to organize his knowledge at different abstraction layers, working better from a strategic, generic, high-level point of view of the problem. The system have to take advantage of that, doing a hierarchical decomposition of the knowledge, and allowing the user to express goals and tasks at different levels. Thus the planner will be based on HTN (Hierarchical Task Network) planning. The language that the system will use for reading domain and problem specifications is PDDL. We'll use the last PDDL version available, that is, PDDL 2.2 [10]. However, at this moment, PDDL partially supports part of the HTN formalism, so it will be necessary to extend it to fully support this new feature.

Another important point is the negotiation process between experts and the planning system. We have to find a way in that human and machine can communicate easily. SIADEX will have a graphical interface (GUI), from where the user can easily interact with the system. This GUI will allow the user to: (1) Allocate or remove all the resources in a map graphically; (2) Control the monitoring process, and explain the causes of possible plan fails; (3) Set the objectives and tasks at different abstraction layers; (4) Dialogue with the system in order to find a consensual plan; (5) Modify or repair the obtained plan; (6) Make query or explanation requests to the system; (7) Load data for other applications, send the commnads, etc.

At last we will present the sequence of stages about the way in which the user and the system will collaborate:

1. Inform the system about the current situation.
2. Discuss the global strategy (divide the fire into sectors, set high-level tasks that have to be accomplished for every sector, etc).
3. The system obtains a set of plans for a selectable time-horizon period. The user can then choose one of then, ask for other alternatives or suggest possible plan modifications. The plan can also be simulated using some tools [7].

The monitor controls the correct execution of plans. In case of fail or due to an expert requirement the process can be restarted.

6 Conclusions and Future Work.

In this paper we have described the SIADEX project. As we have seen, SIADEX is an ongoing system, based on intelligent planning techniques, aimed to assist an expert in the task of designing forest fire extinction plans. Also, it can be used as a virtual learning environment for forest fire extinction. We have described the methodology we are following, the architecture of SIADEX, its knowledge base (BACAREX) and the planning and replanning techniques we are using. At this moment BACAREX has been fully developed and the planner and the graphical user interface are still under development.

References

[1] G. Schreiber, H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. Van de Velde, and B. Wielinga. Knowledge Engineering and Management — ”The CommonKADS Methodology”. The MIT Press, Cambridge, Massachusetts; London, England, 1999.

[2] Karen L. Myers, Peter A. Jarvis, W. Mabry Tyson, Michael J. Wolverton. A Mixed-initiative Framework for Robust Plan Sketching. ICAPS 2003.

[3] M. de la Asunción, L. Castillo, J. Fdez-Olivares, O. García-Pérez, A. González and F. Palao. SIADEX: Assisted Design of Forest Fire Fighting Plans by Artificial Intelligence Planning Techniques. http://siadex.ugr.es, 2003.

[4] Karen L. Myers. CPEF A continuous Planning and Execution Framework. AAAI 1999.

[5] P. R. Cohen, M. L. Greenbreg, D.M. Hart, and A. E. Howe. Trial by fire: Understanding the design requirements for agents in complex environments. AI Magazine, 10:32-48, 1989.

[6] P. Avesani, A. Perini, and F. Ricci. Interactive case-based planning for forest fire management. Applied Intelligence, 13(1):41-57, 2000.

[7] Caballero, D.; Martinez-Millan, J.; Martos, J.; Vignote,S. (1994). CARDIN 3.0, a model for forest fire spread and fire fighting simulation. Proceedings of the 2st International Conference on Forest Fire Research. Coimbra, Portugal, Vol.1: 501.

[8] L. Castillo, J. Fdez. Olivares, and A. Gonzalez. Some issues on the representation and explotation of imprecise temporal knowledge in an AI planner. In Knowledge-Based Intelligent Information and Engineering Systems, Lecture Notes in Artificial Intelligence, LNAI-2774, pages 1321-1328. Springer-Verlag, 2003.

[9] Protégé Project. http://protege.stanford.edu

[10] S. Edelkamp and J. Hoffmann. The Language for the Classical Part of the 4th International Planning Competition, draft for the use of IPC-2004 participants. http://www.informatik.uni-freiburg.de/∼hoffmann/ipc-4/DOCS/pddl2.2.ps.gz

Reducing the impact of AI Planning on end users[*]

Luis Castillo [1] Juan Fdez-Olivares [1] Óscar García-Pérez [2] Francisco Palao [2] Tomás Garzón [2]

[1]Dpto. Ciencias de la Computación e I.A.
University of Granada, SPAIN
{L.Castillo,Faro}@decsai.ugr.es

[2]IActive Intelligent Solutions
{O.Garcia,F.Palao,T.Garzon}@iactive.es
http://www.iactive.es

Abstract

In this paper we analyze the lessons learned during three projects that are being developed between our research group and our spin-off IActive Intelligent Solutions. These projects are based on the deployment of AI planning technology in three different business environments with a different degree of maturity in ICT. In these three cases we will focus on how to extract domain and problem files and how to integrate action plans with existing information services on the end-user side in order to facilitate the integration of P&S technology with legacy software and end-users work environment.

Introduction

Artificial Intelligence Planning research community has always kept an eye out for applying its technology to real world problems and reach an industry level deployment comparable to that of other disciplines like neural networks, genetic algorithms or fuzzy logic, just to mention some of them. Despite being a relative old discipline, at least older than most of those mentioned before, planning technology has some very good examples of applications but it does not seem to be mature enough to lead an enterprise-wide deployment and to be part of the whole enterprise or business jigsaw puzzle.

There may seem to be many reasons behind this lack of success like the need for efficient planning algorithms, the need for enhanced underlying reasoning processes (uncertainty, time, resources, ...), the need to deal with exogenous events or sensing operations among others. These are some of the lines that are being pursued from the own research community to bridge this well known gap. Although all of these problems focus on the inner part of the planning piece of Figure 1, and this is an important effort to drive this technology forward, there are other important questions that should also be addressed with regards to insert the planning piece into the whole puzzle. Both categories of problems are very important, but papers that fall on the former type seem to dominate in mainstream conferences. Here, we intend to share some of the interfacing issues of the second type that we have learned from our past and current experience that started in our research group and is now being undertaken at our spin-off IActive Intelligent Solutions[1].

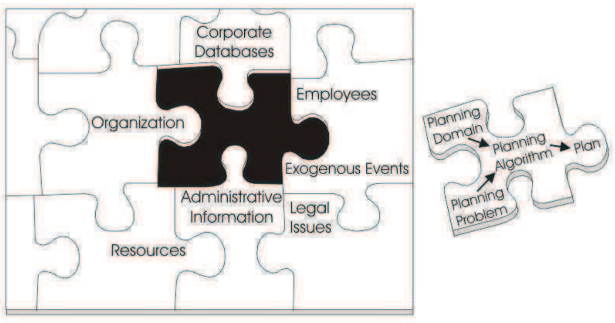


Figure 1: The enterprise-level (business-level) integration puzzle

The main issue: knowledge integration

One of the most distinguishing features of planning technology is that it is a knowledge-intensive task, that is, planning a solution to a problem in a human-centered environment implies taking into account a lot of knowledge about the problem being solved. This knowledge comes from many different and, mostly, heterogeneous sources and it must be "prepared" somehow to the PDDL (Edelkamp & Hoffmann 2004) input gateway (or similar) to our state of the art planning engines in the form of problem and domain files. In addition to this, the solution plan must also be processed to fit into the structure of the whole puzzle. Therefore we must analyze how these inputs and outputs of our planning algorithms impact into the remaining components of the puzzle and try to reduce this impact as a way to ensure (part of) the success of the planning technology.

In our very modest point of view, the integration of AI planning technology into any existing enterprise or business environment must take into account the following points:

- Technology is an enabling factor of change, not the

[*]This work has been partially supported under the research contract NET033957/1 with the Andalusian Regional Ministry of Environment and the CICYT Project TIN2005-08945-C06-02

[1]http://www.iactive.es

change itself. That is, the integration of new technology should not produce a change for the environment to adapt to it, instead, technology should adapt to the existing business environment. This is very important because business stakeholders (decision makers, actors and managers) are not intended to know about planning domain languages, nor logical predicates nor any other formalism planning researchers are used to deal with. So there must be a mapping between the language that business stakeholders are used to deal and the language that our planners handle. This mapping is not only a matter of translation but it may also imply a hard work on knowledge extraction and validation.

- Technology must be as transparent as possible. Business environments are plenty of software for their daily work and we do not foresee the planning engine to be a standalone application by itself. Instead, we do foresee a planning engine as something as a "plug and play module" or a "helper application" for legacy software so end users may have access to planning technology from and to their everyday applications.

The combination of these two points is, therefore, an important issue to get planning technology integrated into the whole business puzzle. They are not exclusive of AI planning technology, but it is also shared with other mature AI technology. However, planning technology strongly depends on the representation of operational knowledge[2] and this knowledge is not easily available in a explicit form in these environments, so its integration appears to be more difficult. In a increasing number of cases, business stakeholders are starting to pay more attention to business process modeling and management so that they use different languages to represent their operative knowledge and to execute and monitor their business processes. In the best situation, extracting the operative knowledege required for the planner becomes easier and less costly. Easier because all the processes are already modeled and described by the own end user so a planning domain expert only needs to query these models and translate the required details into a planning domain description language like PDDL. And less costly because, if this translation process could be done automatically, any maintenance change on the business rules or processes made by the end users, are immediately translated into the planning domain without the intervention of planning domain experts.

In the forthcoming sections we discuss how the consideration of these points have affected to three different projects. In all of them there was a need to integrate AI planning techniques but none of their end users was supposed to know about these techniques nor artificial intelligence in general, so the integration of the planning piece in their corresponding business puzzle was a subtle issue.

[2]Operational knowledge must be understood as the knowledge about how to get things done, as opposite, or at least different, to descriptive knowledge about how things are.

First case of study: crisis management in Andalusia

In (Fdez-Olivares *et al.* 2006) we were (and in fact we still are) engaged in the application of planning techniques to help fire directors to define a forest fire fighting plan in Andalusia (Spain), it fully covers the stages of preparedness and response and it might be generalized to any other crisis situation. End users are fire directors, in charge of making the main decisions and driving all the operations during the episode, and technical and administrative staff in their chain of command, who are responsible of launching execution orders, monitoring the execution of the plan and updating administrative information.

The problem

The description of the initial state of the problem must contain an exhaustive description of all the fire fighting resources in Andalusia. This implies the representation of large amount of information about facilities (32), brigades (341), pumping vehicles (94), spraying helicopters (27), etc. The goal of the problem must be described in terms of the geographical deployment of operational areas and the target operations. The main issue here is that the planner needs this information and this information is being queried and updated daily by human operators as well, who are not intended to know anything about PDDL nor other planning formalisms. Obviously, there must be a common representation, accessible both by human users and by the planner with the following features:

- It must be rich enough to represent all the knowledge required for the planner.
- It must be easily accessible by human operators to query and update daily operations, with no training effort on any additional language.
- It must be easily accessible by the planner to extract the required knowledge into a PDDL problem file.

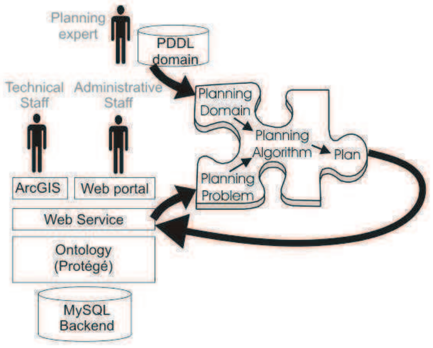
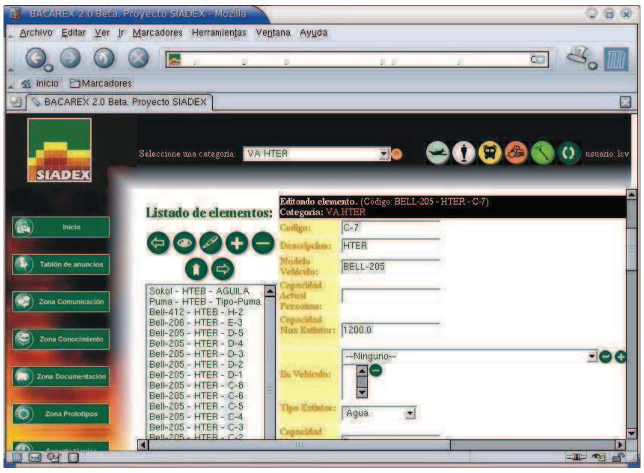


Figure 2: Integration of the first case on crisis situations

The representation chosen was an underlying ontology designed in Protégé[3] with a MySQL back-end to support an efficient and concurrent access to the information. On top of

[3]http://protege.stanford.edu

this ontology we built a web service with a clearly defined API. This web service may be used either from a web portal (so that administrative staff easily query and update the knowledge in the ontology through any web browser, see Figure 3) or from an existing GIS application[4] (so that technical staff may easily introduce the geographical layout and target areas of the goal, see Figure 4)


BACAREX 2.0 Beta. Proyecto SIADEX - Mozilla
Archivo Editar Ver Ir Marcadores Herramientas Ventana Ayuda
Inicio Marcadores
BACAREX 2.0 Beta. Proyecto SIADEX
SIADEX
Seleccione una categoría: VA HTER
usuario: lcv
Listado de elementos:
Editando elemento. (Código: BELL-205 - HTER - C-7)
Categoría: VA HTER
Inicio
Tablón de anuncios
Zona Comunicación
Zona Conocimiento
Zona Documentación
Zona Prototipos

Sokol - HTEB - AGUILA
Puma - HTEB - Tipo-Puma
Bell-412 - HTER - H-2
Bell-206 - HTER - E-3
Bell-205 - HTER - D-5
Bell-205 - HTER - D-4
Bell-205 - HTER - D-3
Bell-205 - HTER - D-2
Bell-205 - HTER - D-1
Bell-205 - HTER - C-8
Bell-205 - HTER - C-6
Bell-205 - HTER - C-5
Bell-205 - HTER - C-4
Bell-205 - HTER - C-3

Codigo:	C-7
Descripcion:	HTER
Modelo Vehiculo:	BELL-205
Capacidad Actual Personas:	
Capacidad Max Extintor:	1200.0
En Vehiculo:	--Ninguno--
Tipo Extintor:	Agua

Figure 3: Access to the knowledge base through a web browser for administrative purposes

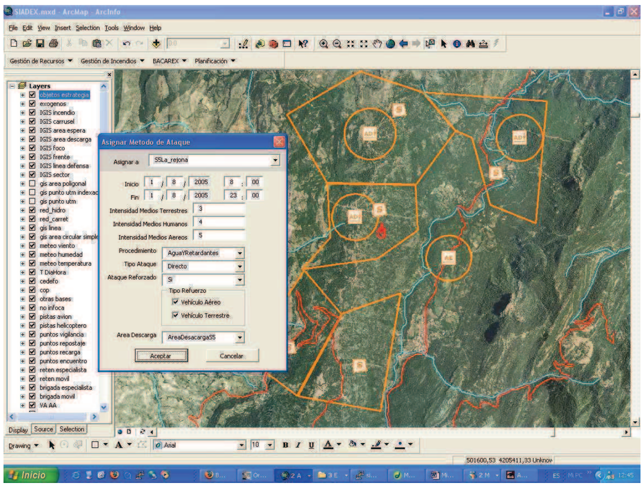


Figure 4: Access to the knowledge base through ArcMap for technical purposes

The domain

The current degree of maturity on ICT of the forest fire fighting service in Andalusia does not allow them to have formalized their fire fighting protocols in any well known process language like OWL-S or XPDL. Instead, they only have training and working documents that explain these protocols. Therefore, we decided to encode the domain by hand for our HTN planner, as an HTN extension of PDDL 2.2 (Castillo *et al.* 2006), and keep it away from administrative and technical staff so that only a planning expert may update and modify it. We were not happy with this decision because neither administrative or technical staff may have access to the fire fighting protocols encoding and thus, their independence (and therefore the transparency of planning technology) was severely reduced. This was mandatory since, currently, administrative and technical staff are not expected to be introduced in process description languages. However, Andalusian administration is introducing these languages gradually at all the levels of the administration and we foresee that, in a near future, forest fighting technical staff will be skilled enough to formalize and represent their fighting protocols in some of these languages and then, the adoption of our planning technology will be much more transparent, as it is shown in the next cases of study.

[4]ArcMap by ESRI http://www.esri.com

The plan

Finally, the planner has been integrated as an additional toolbar of the ArcGIS suite so that it is called just by clicking a button on their everyday desktop. The plan obtained by our planner is also introduced in the whole business puzzle of the technical and administrative staff. Our planner is able to obtain the plan in an enhanced XML format that includes, all the temporal constrains (direct and inferred constrains), binding of variables and annotations gathered during the search process. This allows us to translate this XML plan into other formalisms like a chronogram for the GIS platform, a MS Excel file, a Gantt chart or a proprietary format of the technical staff (Figure 5).

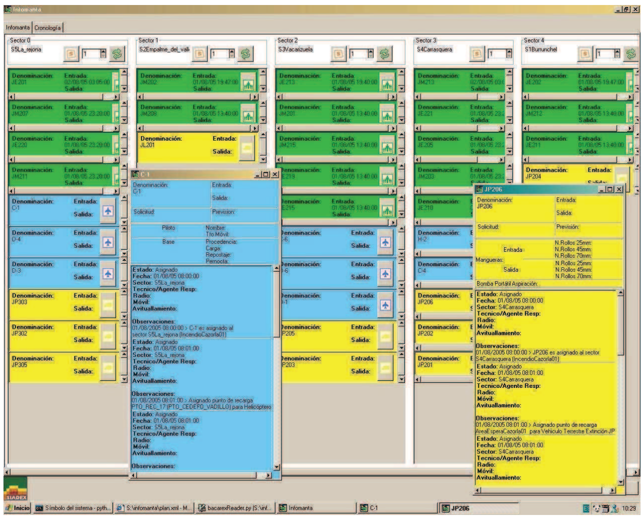


Figure 5: A proprietary format for plan visualization

As may be seen, introducing the plan into the legacy software infrastructure is the easiest task given our enhanced XML representation of the plan.

In summary, planning technology has been introduced silently, integrated with the regular working environment of the administrative and technical staff (web browsing, GIS software, spreadsheets and Gantt editors). The unique drawback is that technical staff cannot modify the domain by themselves but through our intervention. This is a matter of the maturity of the forest fire fighting field and it is expected to change in the near future with the adoption of standard business process modeling languages.

Second case of study: learner centered design

This second case focus on the distance learning field, particularly in what is known as learner centered design. In this case, the introduction of planning technologies allows us to define customized learning paths for a given course. That is, the goal is the arrangement of the resources associated to the course taking into account the goals selected by the instructor and the own needs, features and constraints of every student, so that every student in the same course will have its own learning path to the goals.

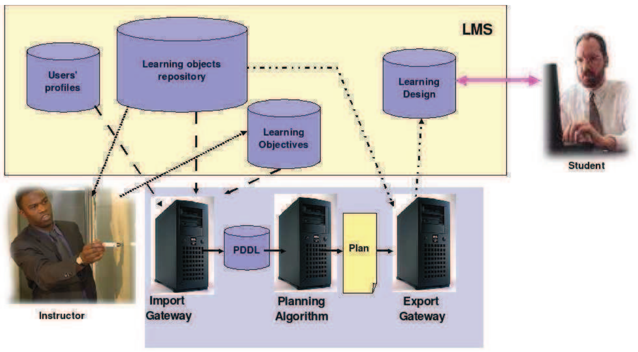


Figure 6: General view of a Learning Management System

A Learning Management System (LMS) is composed of several related databases (Figure 6):

- The learning objects repository contains all the educational resources (documents, videos, photographs, schemes, etc) that could be linked to make up a course. Every learning object is labeled by means of an extensive set of standard metadata (IMS-GLC 2007) so that the instructor may describe the main features of the learning object and its adequacy to different student profiles.
- The user profiles database contains extensive information about each student: personal data, preferences, learning style, academic history, his/her hardware/software platform and others. It follows the IMS-LIP standard (IMS-GLC 2007).
- The learning objectives are specified by the instructor for each course, so all the students of the same course are intended to reach the same goals.
- The learning design database contains a timed sequence of learning objects that each student must follow to reach the course' goals adapted to his/her own features. It follows the IMS-LD specification (IMS-GLC 2007).

The goal of a LMS is to serve as a framework for the definition of a course and for the student to follow that course in a distance learning setting.

The introduction of planning techniques in this environment may be described by the following steps (Figure 7:

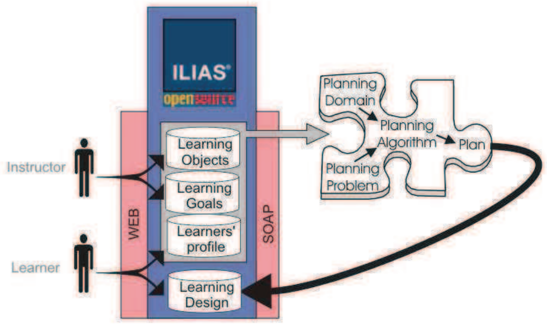


Figure 7: Integration of AI planning into the ILIAS Learning Management System

1. The learning objects repository is labeled using a extensive set of standard metadata, mainly a specific subset of metadata that encode the structure and dependence of the learning objects (for more details see (Castillo *et al.* 2007)).
2. (Dotted lines) The instructor explores the repository and define the learning objectives of a given course.
3. (Dashed lines) Our system explore the different databases of users profiles, learning objects and learning objectives and generate the necessary PDDL 2.2 (Edelkamp & Hoffmann 2004) files for our HTN planner to run. The planner is executed and a customized learning plan is obtained for every student registered at the same course.
4. (Dotted/dashed lines) The learning plan is translated into a form playable or understandable by the LMS, usually under the IMS-LD specification.
5. The plan is executed (or played) by the student to follow the course adapted to its own features and needs.

The problem and the domain

This case of study provides a more formal framework for inserting planning technology since the standards used for labeling metadata and user profiles provide a great amount of knowledge (Figure 8) that can be exploited to extract descriptive and operational knowledge for the planner. In particular in (Castillo *et al.* 2007) we show that this knowledge is rich enough so as to automatically extract the problem and the domain files for a HTN planner from a SOAP interface (W3C 2007) provided by the web services of the ILIAS LMS (ILIAS 2007), a well known platform for distance learning.

These metadata are introduced directly from the LMS (Figure 9) and they all belong to the standards commonly used in distance learning, so there is no additional impact on end users (instructors).

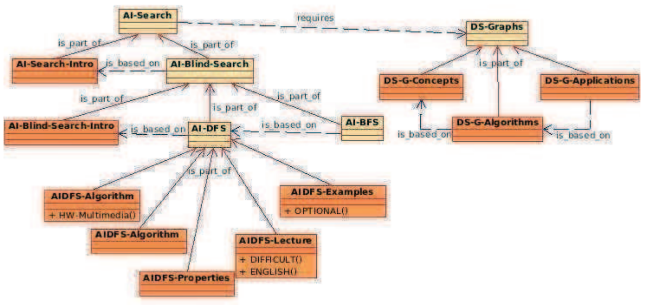
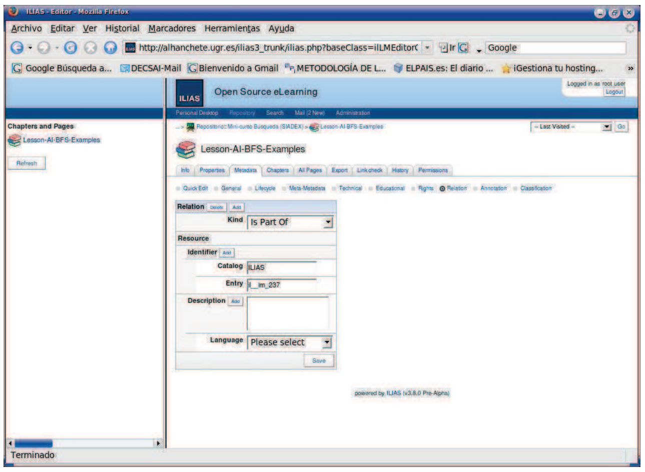


Figure 8: An exhaustive labeling of learning objects (compound objects in light shadow and simple objects in darker shadow) showing ordering, dependence and composition relations. It also shows that every simple object may be labeled with other features like its language, its hardware and software requirements, its degree of difficulty and its optionality amongst other



ILIAS - Editor - Mozilla Firefox
Archivo Editar Ver Historial Marcadores Herramientas Ayuda
http://alhanchete.ugr.es/ilias3_trunk/ilias.php?baseClass=ilLMEditorG
Google
ILIAS Open Source eLearning
Chapters and Pages
Lesson-AI-BFS-Examples
Refresh
Lesson-AI-BFS-Examples
Terminado

Relation
Kind: Is Part Of
Resource
Identifier
Catalog: ILIAS
Entry: il__lm_237
Description
Language: Please select
Save

powered by ILIAS (v3.8.0 Pre-Alpha)

Figure 9: All the metadata and profiles are introduced through the standard web interface of the LMS

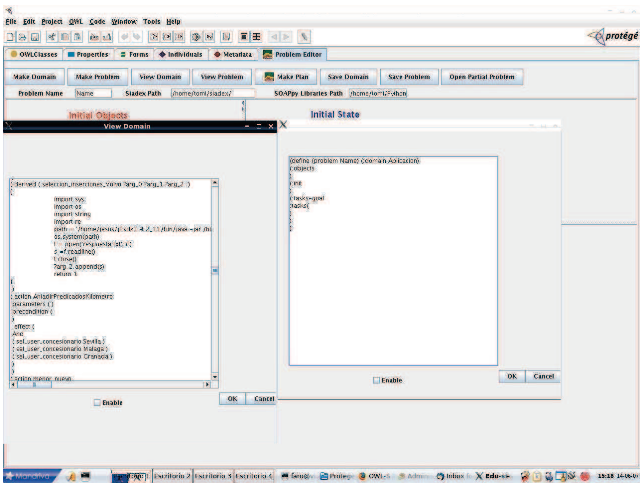
The plan

The HTN domain and problem files, which are automatically extracted from the LMS, are fed into the planner and a plan is obtained for every student registered in the same course. Although each plan might be different, all of them will allow students to reach the same goals, but taking into account the special features of each student. Finally, this plan is inserted back into the LMS to be played in the form of a IMS-LD compliant file.

In summary, the cost of introducing planning techniques in this business environment is dramatically reduced and the technology is fully transparent to end users (instructors). The effort made by the instructor to encode the metadata of the learning objects, something that can be considered usual in any LMS, is enough for obtaining the most subtle part of the planning piece: the problem and the domain files. Later on, the plan is easily inserted in the LMS platform with no additional cost. This means that, if end users are skilled on some high level formalism for their daily work and this formalism is able to encode some descriptive and/or operative knowledge useful for the planner, then we can extract problem and domain knowledge directly from these formalisms without having to depend on others (planning experts) nor having to learn a different formalism. The following case follows this argumentation and introduces a third case of study in which end users are skilled in a process description language.

Third case: semantic web services composition

Semantic web services techniques support the way in which already existing syntactic web services can be extended with a semantic layer in order to be automatically discovered, composed and invoked. The main goal of this third case of study is to provide a logical framework for an HTN planner to be capable of both interpreting SWS descriptions and, given a high level service request, reasoning about them in order to automatically compose and execute a sequence of web services that provides the service requested (see (Fdez-Olivares *et al.* 2007) for more details). There are several standard proposals for SWS but OWL-S (Martin *et al.* 2003) is a very good choice to this purpose for the following reasons. On the one hand, OWL-S process model allows to represent web services as processes with typed input/output parameters, preconditions and effects and a compositional hierarchy of atomic and compound processes. And, on the other hand, it is based on a data model built on top of an OWL ontology consisting of classes, properties and instances. Therefore, our goal, in this case, is translating the OWL-S process and data models into an HTN extension of PDDL 2.2 domain and problem files, call the planner and obtain a plan as a timed sequence of actions that could be used as an executable sequence of web services to give a response to the high level service request.



Problem Name: Name
Siadex Path: /home/tomi/siadex/
SOAPpy Libraries Path: /home/tomi/Python

```
(:derived ( seleccion_inserciones_Volvo ?arg_0 ?arg_1 ?arg_2 )
{
            import sys
            import os
            import string
            import re
            path = '/home/jesus/j2sdk1.4.2_11/bin/java -jar /h
            os.system(path)
            f = open('respuesta.txt','r')
            s =f.readline()
            f.close()
            ?arg_2.append(s)
            return 1
}
 )
(:action AniadirPredicadosKilometro
:parameters ( )
:precondition (
)
 :effect (
 And
 ( sel_user_concesionario Sevilla )
 ( sel_user_concesionario Malaga )
 ( sel_user_concesionario Granada )
)
)
(:action menor_nuevo
```

```
(define (problem Name) (:domain Aplicacion)
(:objects
)
(:init
)
(:tasks-goal
:tasks(
)
)
)
```

Figure 10: Our system has been embedded into an OWL and OWL-S editor environment as Protégé

The problem and the domain

The translation process first maps the OWL data model into the PDDL data model by translating OWL classes, properties and instances into PDDL types, predicates and objects, respectively.Then it maps the OWLS process model into an HTN domain that represents the operation of both atomic and composite processes as primitive tasks and task decomposition schemes, respectively. Atomic processes are mapped as PDDL durative actions and the workflow pattern of every composite process is mapped into a method-based task decomposition scheme that expresses the operational semantics of the control structures found in that composite process.

The plan

The planner makes use of the knowledge encoded in the domain (representing the OWL-S process model) as a guide to find a sequence of temporally annotated primitive actions that represents a suitable composition (with possibly parallel branches) of atomic processes. This sequence is sent to a Monitor module that is in charge of both scheduling the execution of atomic processes according to their temporal information and sending execution orders to an Executive module. This module is in charge of executing web services invocations and sending back the information.

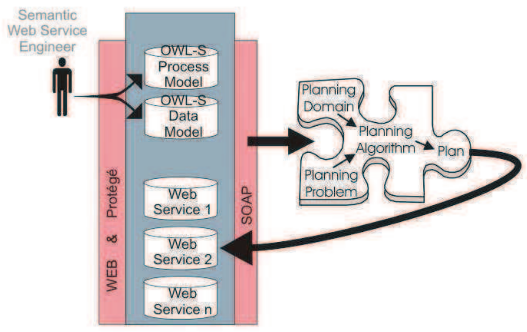


Figure 11: The integration of our HTN planner into a web service composition based business environment

In summary, this last example has also shown how a planning engine may be seamless integrated into environments with a strong underlying formalization of processes. In this case, it is a web service composition based environment and domain experts are supposed to have skills on process design languages like OWL-S (in the case of other languages, the procedure would be similar). The point is that their business models written in OWL-S are rich enough so as to extract valid PDDL domain and problem files, so the introduction of planning technology is fully transparent for these end users and it may be fully embedded into their existing working environments.

The way forward

After these three cases, we have learned two important lessons that might be considered complementary. On the one hand, there is a common issue about the **integration** of AI planning technology into existing business environments, what imply the need to share the information between end users and the planner. AI planning needs much knowledge and most part of it is dynamic, it depends on end-users databases and not only the user but also the planner may modify this knowledge, so there must be a common representation of the knowledge or, at least, a gateway to get the knowledge from and to the available sources. No one will accept to replicate their data or re-type it by hand as the input to the planner. The planner must adapt to the existing structure of the data and get what it needs wherever it is. In most cases, the sources of knowledge are very heterogeneous since end users may have its information distributed on different platforms, operating systems or database systems. This also implies that the target environment must be either based on, or ready to adopt, a service oriented architecture (**SOA**) based on the extensive use of web services, in order to enable this **interoperability** of different platforms and to grant access to the whole set of data available in the enterprise. In the case that the target environment is not adapted to a SOA, the planning module might be a catalyst for the introduction of such technologies since it is the main interested part in having a common access to the whole enterprise data (Fdez-Olivares *et al.* 2006).

On the other hand, and very related to this, it is the question about the level of automation of the workflow management at the target business environment. In the case that this business environment has automated (or is automating) its operational processes within what is known as **Business Process Management** (BPM) (Fischer 2007) that takes into account resources, employees, applications, documents and the own organization, the application of AI planning technology seems to be less costly. It would also be less independent of third party planning experts, since all of the knowledge needed to encode domain and problems for the planner may be extracted from their BPM suites (like in cases 2 and 3 before). This is becoming particularly good since most relevant enterprises are currently engaged in a process of automation of their operational business processes by using standard languages like OWL-S but mostly XPDL and BPEL (Fischer 2007) and there is a clear mapping between these languages and our planning domain description languages like PDDL either for plain or HTN domains (Castillo *et al.* 2007; Fdez-Olivares *et al.* 2007; Sirin *et al.* 2004). Since BPM suites integrate business analysts, technical developers and business managers, they can modify their business rules by themselves and the planning domain will be automatically updated, without the intervention of external planning experts and thus, increasing the transparency of this technology. Even more, these BPM suites are strongly based on the use of an underlying SOA, what provides the best context for a seamless and deep integration of AI planning technology. In the case that the target environment is not aware of this BPM technologies, the implantation of AI planning would be more difficult since business experts will still depend on planning experts to modify planning domains as soon as their business rules change (like the first case in this paper).

In any case it is worth to recall that these issues of transparency and integration of AI planning technology are a key factor to implement and deploy our technology, that is, we must seamlessly integrate with end users data, but also with their business rules, so that we do not induce a change on the enterprise before the adoption of our technology but reduce the impact that this technology might have after adopting our technology.

References

Castillo, L.; Fdez-Olivares, J.; García-Pérez, O.; and Palao, F. 2006. Efficiently handling temporal knowledge in an HTN planner. In *Sixteenth International Conference on Automated Planning and Scheduling, ICAPS*.

Castillo, L.; Reynaga, L.; González-Ferrer, A.; Fdez-Olivares, J.; and García-Pérez, O. 2007. Knowledge engineering and planning for the automated synthesis of customized learning designs. In *Conference of the Spanish Association for Artificial Intelligence*.

Edelkamp, S., and Hoffmann, J. 2004. The language for the 2004 international planning competition. http://ls5-www.cs.uni-dortmund.de/ edelkamp/ipc-4/pddl.html.

Fdez-Olivares, J.; Castillo, L.; García-Pérez, O.; and Palao, F. 2006. Bringing users and planning technology together. experiences in SIADEX. In *Sixteenth International Conference on Automated Planning and Scheduling, ICAPS*.

Fdez-Olivares, J.; Garzón, T.; Castillo, L.; García-Pérez, O.; and Palao, F. 2007. A middleware for the automated composition and invocation of semantic web services based on htn planning techniques. In *Conference of the Spanish Association for Artificial Intelligence*.

Fischer, L. 2007. *2007 BPM & Workflow Handbook*. Workflow Management Coalition.

ILIAS. 2007. ILIAS Learning Management System website. `http://www.ilias.de/ios/index-e.html`.

IMS-GLC. 2007. IMS Global Learning Consortium. `http://www.imsglobal.org/`.

Martin, D.; Burstein, M.; Lassila, O.; Paolucci, M.; Payne, T.; and McIlraith, S. 2003. Describing web services using owl-s and wsdl. http://www.daml.org/services/owl-s/1.0/owl-s-wsdl.html. DAML-S Coalition working document.

Sirin, E.; Parsia, B.; Wu, D.; Hendler, J.; and Nau, D. 2004. Htn planning for web service composition using shop2. *Journal of Web Semantics* 1(4).

W3C. 2007. Soap version 1.2. `http://www.w3.org/TR/soap12-part1`.

Improving Planning Techniques for Web Services

Francisco Carlos Palao Reinés
Dept. of Computer Science and Artificial Intelligence
University of Granada, SPAIN
palao@decsai.ugr.es

Abstract

The new trend on software development is oriented to web services running in collaborative environments. Intelligent planning techniques are very useful to compose complex calls to these web services. However, there are still some issues that need to be improved to use planning and scheduling techniques in dynamic and collaborative contexts like the web service environment. This paper proposes some extensions to our planning system for using it to compose web service calls.

Introduction

This research extends the SIADEX environment, which is a planning system oriented to assist the command technical staff for decision support in forest fire fighting operations. The system is composed of different components communicating each other and working together through the Internet. These components are implemented as web service: they are pieces of software that make themselves available over the Internet and uses standard XML messaging system. The World Wide Web is turning into a new paradigm called the Collaborative Web (Pallot, Prinz, & Schaffers 2005) where not only documents are connected through the network but collaborative services as well. The SIADEX project has some different web services working together for a common goal (Figure 1). One web service to store the knowledge of the problem; another to make the planning process; and a third monitors the plan execution. The planning web service works with the SIADEX planner that has been developed by us and is a forward state-based HTN temporal planner (Castillo *et al.* 2006). Moreover there is a user interface where technical staff can introduce the problem to solve it, see the plan generated by the planner and follow the execution.

To achieve a correct system performance we need to call the different web services in the correct order and time. In order to do that we have developed a central server that synchronizes all the component of the architecture. This central server has been called the InfoCenter. The InfoCenter is based on a publish/subscribe architecture (Carzaniga, Rosenblum, & Wolf 2001) (PSA) that works as follows. Each web service or user interface can publish information in the central server (InfoCenter) and also can subscribe to the information (published by others web services or clients) that they want. For instance, when the technical staff publishes the problem, the InfoCenter send it to the planner web service that is subscribed to it.

At the present, our architecture is very simple because only a web service of each kind is available and no external web services can connect to the InfoCenter to make the system more complete. Therefore the InfoCenter can easily compose calls to the web services to achieve the requests, the execution and the monitoring of the plans.

Now, we want to extend the system not only to assist the technical staff for decision support in forest fire fighting but also for e-business, e-tourism and workflow applications among others. Therefore we need to extend the system with new web services, some of them different than the current ones and another with similar capabilities. The current PSA presents some lacks that impede the extension of the system. For instance, the InfoCenter can not choose what web service is the correct one (or optimal one) when there are more than one web services that offer the same functionality. Furthermore, the PSA is purely reactive because it only make web service calls when receive a publication and it is not able to compose sequences of web services with a longer time horizon.

In order to extend the system architecture we are going to use planning techniques into the InfoCenter, as well some frameworks to use planning techniques for web services composition have been purposed (Madhusudan & Uttamsingh 2006; Mithum, desJardins, & Finin 2003) but there is still a lot of issues to solve to fulfill our InfoCenter requirements. We are thinking of using our own planner, SIADEX, to use it inside the InfoCenter. However, we need to improve it for some reasons. Firstly, the web services environment is a very dynamic context. Therefore, the state of web services (the domain) and the requests of the users (the goal) can change during the execution of calls to web services. And our planner can not check domain and state changes during planning time. Secondly, there could be a large number of web services with similar capabilities and the InfoCenter have to evaluate them to decide which one is the best to achieve its goals. And thirdly, we are thinking in an architecture oriented to the Collaborative Web, so our intelligent InfoCenter need to communicate with others intelligent servers in order to access trough them to resources that are not directly connected to it. So, the planner needs to be extended with distributed planning skills and to be able to understand web services standard languages.

In this work we present our system SIADEX as a framework that will be extended with novel ideas to give solutions to all these problems.

The SIADEX architecture

SIADEX is a system being developed under a research contract with the Andalusian Regional Ministry of environment. Its objective is to assist the command technical staff in the design, dispatching and progress of forest fire fighting plans. It is composed of different, domain independent web services (Figure 1), that offers different services, that are distributed and communicate with each other using XML-RPC standard protocols.

- SIADEX Planner: Is a planning web service that can be called by XML-RPC protocol. SIADEX is a forward state-based HTN temporal planner (Castillo *et al.* 2006). It uses its own hierarchical extension of PDDL 2.2 level 3 language, that makes it very expressive. It also has the capability to include embedded Python scripts in the domain definition, that allows us to implement external calls at planning time.
- BACAREX: Is an ontology web service that stores the knowledge related to the planning domain. In our case its stores information about the forest fire fighting domain in Andalusia (Spain). BACAREX is also capable of generating domain and problem files that are processed by our planning web service.
- Monitor service: This web service splits the plan into several pieces and sends every piece to the person in charge of executing it. These parts of the plan will be presented to the user using any portable electronic device. The monitor controls the plan executions attending the dependences between tasks and its possible delays (Castillo *et al.* 2006).
- User interfaces: We have provided GUI capabilities to the planning system for the expert. The GUI is built on top of the ArcView GIS tool (ESRI). This GUI is totally domain dependent and oriented toward the interaction with the forest fire technical staff. We have also developed a web interface to monitor the execution of the plan with any available web browser.
- InfoCenter: It is the central component of our architecture. All the web services that we saw above are connected to the InfoCenter and collaborate each other by passing messages through it. The InfoCenter has been developed as a publish/subscribe architecture (PSA) in which the others web services can subscribe to the information that they want and publish the information that they have to share with others web services. The PSA works correctly in small environments like this with a few web services but it is purely reactive. And we want to extend this architecture to larger and more dynamic environments where we would need a deliberative server able to compound sequences of calls to web services. To achieve this we are thinking on extending the InfoCenter with planning techniques that we need to develop and are explained below.

Composing Web Services and SIADEX

There is an interchange of information between all the web services described above during a planning episode to assist the technical staff for decision support. In this section we show how this interchange of information is done at the present and how would be done in the future supporting larger environments of web services.

Present operation between web services

The InfoCenter is the Broker Server in our PSA . Each web service or user interface can publish information in the central server (InfoCenter) and also can subscribe to the information (published by others web services or clients) that they want. The basic cycle of the present architecture is:

1. The user interface publishes the goal of the planning problem defined by the technical staff and the InfoCenter sends it to the ontology web service that is subscribed to all new information about the world state and the new goals.
2. The ontology web service BACAREX publishes the domain and the problem translated into PDDL from the ontology knowledge and the InfoCenter sends it to the SIADEX Planner that is subscribed to all the domains and problems in PDDL generated.
3. The SIADEX Planner publishes the plan generated and the broker server sends it to the Monitor because it is subscribed to new plans.
4. The Monitor publishes the actions that have to be executed at each time and the InfoCenter sends it to the technical staff in charge of doing it.
5. Until the plan is completely executed, the technical staff send (public) confirmations about actions completed to the InfoCenter and the Monitor, that is subscribed to new events of the actions, publics new actions if it corresponds. BACAREX is also subscribed to the new events of the actions in order to update the world state in the ontology.

The cycle shown above can be carried out with the present PSA that implements the InfoCenter. However, if we had more web services connected and some of them offer the same or similar functionalities, we would need to make more complex compositions of web services that we can not make now. At the moment, the InfoCenter can not choose what web service is the correct one (or optimal one) when there are more web services that offer the same functionality. Furthermore, the PSA is purely reactive because it only make web service calls when receive a publication and it is not able to compose sequences of web services with a longer time horizon. Note that it is not only a selection problem

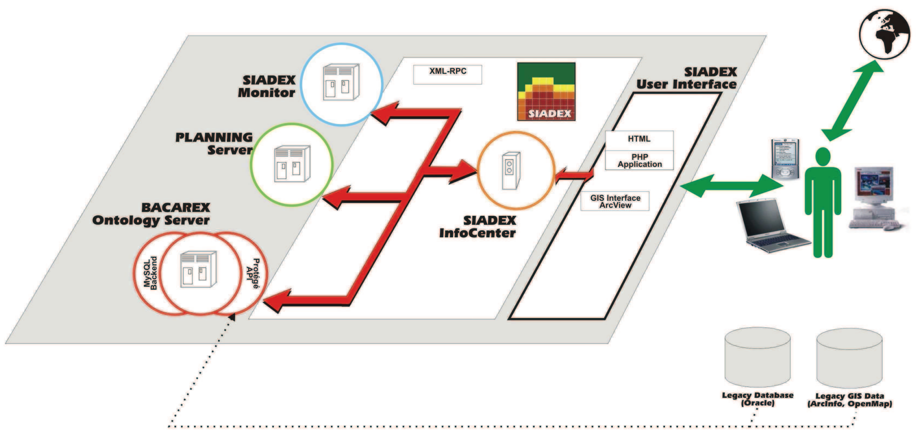


Figure 1: General overview of SIADEX Architecture.

to pick the best web service, we need to make a sequence of calls to web services to know if the goal can be achieved with the available web services. We want to keep the easy connectivity and scalability of our current PSA but in a deliberative way. In order to do that, we will extend the InfoCenter or Broker Server with intelligent planning techniques like we describe below.

Future operation between web services

In the last section we have seen that we need to compose complex calls to web services in dynamic and larger environments. In order to do that we need to improve some features of our own SIADEX planner to use it in the InfoCenter. The features with which we need to extend our planner are shown in this section.

Continuous revision of the state and the goal. A great advantage of our PSA is the high response capabilities to the environment changes. So, it has to be supported by the planner. The set of services available could be constantly changing as online or offline status while we are executing the sequence of calls to web services. In addition the user could change his goals at execution time. Therefore the planning process needs to be continually check for changes in the domain, in the state or in the goals (Giunchiglia & Traverso 1999; Madhusudan & Uttamsingh 2006) during the execution of web services. We can achieve this using the embedded Python scripts in the domain definition to implement external calls at planning time. There are two kinds of calls to extern web services. Firstly, calls to ensure a complete solution by checking that the web service is available. Secondly, calls to ensure a sound solution by checking that the web service behavior is the correct one.

Automated generation of domains and heuristics. As new web services are connected or disconnected to the InfoCenter the domain knowledge changes and the planner need to know it to compose the sequences of calls to web services (plans). We need to make an ontology inside the InfoCenter that stores the web service information (the domain) and define an automatically process that translates the ontology knowledge into the PDDL readable by the planner (Sirin *et al.* 2004). In addition, this web service has to be evaluated to make optimal plans. We need to implement automated heuristics generations (Zimmerman & Kambhampati 2003). To judge the optimality of the plans we need to evaluate the web services by using metrics about the network behavior (response time, transfer rating) and the final user preferences (it could be the price of a product, the duration of a trip, etc).

Distributed planning capabilities. As we have said in the Introduction section, Internet is turning into a new paradigm named the "Collaborative Web" where services collaborate between them. Therefore the InfoCenter has to be able to share with other intelligent servers its plans (or part of them) or to ask others intelligent servers for plans (or part of them). In order to do that we need to consider all the work done in collaborative planning environments (desJardins & Wolverton 1999). Furthermore, the InfoCenter has to be able to understand standard web service description languages such as WSDL and to generate plan outputs as web service flows with standard specifications such as BEPEL4WS or OWL-S.

Concluding remarks

We have described a new approach in the SIADEX planning system architecture in order to prepare it for the new trends on web services environments. The main challenges faced at planning time are in dynamic conditions and the need to collaborate with others web services architectures. We present an extension of our planner to check the completeness and soundless of the solutions in these environments and to be able to communicate with others web services architectures. All the changes proposed are about the central component of the architecture: the InfoCenter. That is the one in charge to compose the sequences of calls to web services with the new planning techniques that will be developed. These sequences of calls to web services have to be formulated in standards specifications such as BPEL4WS or OWL-S.

Acknowledgements

This research was supported by the contract NET033957 with the Andalusian Regional Ministry of Environment for the assisted design of forest fighting plans.

References

Carzaniga, A.; Rosenblum, D.; and Wolf, A. 2001. Design and evaluation of a widearea event notification service. *ACM Transactions on Computer Systems* 19:332–383.

Castillo, L.; Fdez-Olivares, J.; Garcia-Perez, O.; and Palao, F. 2006. Efficiently handling temporal knowledge in an htn planner. In *International Conference on Automated Planning & Scheduling*.

desJardins, M., and Wolverton, M. 1999. Coordinating planning activity and information flow in a distributed planning system. *AI Magazine* 20:45–53.

Giunchiglia, F., and Traverso, P. 1999. Planning as model checking. In *In Proc. 5th European Conference on Planning*.

Madhusudan, T., and Uttamsingh, N. 2006. A declarative approach to composing web services in dynamic environments. In *Decision Support System 41 (2) 325-357*.

Mithum, S.; desJardins, M.; and Finin, T. 2003. A planner for composing services described in daml-s. In *ICAPS2003 Workshop on Planning for Web Services*.

Pallot, M.; Prinz, W.; and Schaffers, H. 2005. Future workplaces, towards the 'collaborative web'. In *1st AMI@WORK Communities Forum Day*.

Sirin, E.; Parsia, B.; Wu, D.; Hendler, J.; and Nau, D. 2004. A declarative approach to composing web services in dynamic environments. In *J. Web Sem 1(4): 377-396*.

Zimmerman, T., and Kambhampati, S. 2003. Learning-assisted automated planning: Looking back, taking stock, going forward. *AAI Magazine* 24:(2) 7396.

Bringing Users and Planning Technology Together. Experiences in SIADEX

Abstract

This work describes the solutions adopted to the problems tackled during the development of SIADEX, an intelligent planning and scheduling application where users play a central role. The system is being developed under a research contract with the Andalusian Regional Ministry of Environment (Spain) and integrates planning and scheduling techniques in a service oriented architecture devoted to support decisions on crisis intervention planning.

Introduction

A largely pursued objective in Intelligent Planning and Scheduling is to make it widely used, bringing its complex techniques nearer to users of the technology, by helping them to solve common, everyday problems. Very recent successful AI Planning systems have been developed (Bresina *et al.* ; Ruml, Do, & Fromherz 2005; Nau *et al.* 2005) in domains where highly skilled staff (in both domain knowledge and computer knowledge) use the applications to solve problems on highly technological domains. As opposite to these successful systems, this work describes the solutions adopted to the problems tackled during the development of SIADEX, an intelligent planning and scheduling application where users (which are non AI experts) play a central role, and that has been integrated into their familiar working environment in order to solve their everyday problems about decision making. The system is being developed under a research contract with the Andalusian Regional Ministry of Environment and integrates planning and scheduling techniques in a service oriented architecture devoted to support decisions on crisis intervention planning.

The current domain application of SIADEX is forest fire fighting, where experts need to obtain valuable tentative fire attack plans, composed of a sequence of timed fighting operations, before making the most suitable decisions when defining the strategy to fight a forest fire. Therefore, in order to cover this need, the system obtain temporal fire attack plans, providing support to monitor their execution, and allowing to revise experts decisions by re-defining the strategy (if needed) by a continuous interaction with the user in every stage of the crisis episode.

In order to understand the problems to be faced while developping such a system, we will start by introducing the decision making process presently followed by technical staff.

Decision making in crisis intervention planning

The management of crisis episodes such as forest fire fighting usually follows a sequence of stages like that shown in Figure 1, in which several stages of situation assessment, intervention planning, plan dispatching and execution and plan execution monitoring are followed up to completely overcome the crisis. Due to the dynamics and the many uncertainties of real world, a revision process of the intervention plans might be carried out, triggering again a new cycle.

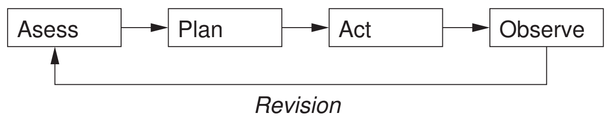


Figure 1: Life cycle of crisis management

This cycle usually relies in a chain of decision making stages from high skilled staff (in charge of assessing the situation, defining the strategic goals of the intervention, and the procedures to achieve these goals) to ground operators (in charge of carrying out operational activities that materialize procedures).

Technical staff related to SIADEX implement this lifecycle in the following steps: first, once a fire arises and its geographical initial point is identified, all the information about the fire local environment is gathered (geographical information such as orientation of the terrain, slope and access routes for fight means, forest fuel, and weather forecasts for the next hours such as temperature, humidity and wind speed and direction). This information, together with the output of a simulation of the fire spread, is used to roughly assess the best strategy to fight the fire in a given time horizon (usually no more than 12 hours).

The strategy consists in defining a fire scenario including both the definition of a group of operational geographical areas (called sectors) where the attack will be focused, and the statement of high-level fighting procedures to be performed at every sector. Experts also assign fighting means (such as terrestrial, aerial and human means) to these procedures, with no more information than the provided by their own, personal expertise. At present, the only guide at their disposal is provided by ”standard operating procedures” (called by technical staff ”fighting protocols”), a specification at different levels of detail about what decisions have to be made and which tasks have to be carried out under these decisions. Protocols are then used to make subsequent decisions about what operational fighting activities (such as mobilization or deployment) should be carried out by assigned fighting means.

Thus, apart from the environmental information, and in order to make the best decisions on the strategy, experts must consider further crucial information (most of which is not yet available) about means such as current geographical positions, availability, fighting capacity (in order to determine the most appropriate ones regarding distance to the fire, experience, etc.), expected times of arrival to the fire, work shifts of the fighting brigades, periods of availability of aerial means, scheduled aerial water discharges, reservoir points to recharge both terrestrial and aerial means, etc. Considering the large amount of fighting means [1] and the short time response required decision making becomes much harder and, in order to make an anticipated analysis of the consequences of their decisions, experts need to have a completely detailed *attack plan* composed of a sequence of timely programmed fighting activities of the assigned means in the strategy.

At present, technical staff only have at their disposal some tools to record and manage the overwhelming amount of information gathered from the environment[2], but no tool is used to allow experts to evaluate their decisions about the best strategy to be defined by offering valuable tentative attack plans. Furthermore, due to the dynamics and uncertainty of the environment where the operations are executed, strategies and subsequent attack plans are revised every certain interval of time according to the evolution of the fire. Therefore, it becomes also necessary to allow experts to revise their decisions once a given plan is accepted, according to the life-cycle of Figure 1.

SIADEX is being developed in order to cover these needs and, therefore, it has been designed as a planning and scheduling system that obtain attack plans, providing support to monitor their execution, and allowing to revise experts decisions by re-defining the strategy (if needed) by a continuous interaction with the user in every stage of the crisis episode. The development of such a system introduces some challenges to current state of the art planning techniques that will be summarized next.

[1]At present, means devoted in Andalusia to forest fire fighting amount about 300 squads (grouping about 3000 people), 200 terrestrial vehicles (devoted either to transport or fire suppression) and 30 aicrafts

[2]Most of them based on ArcView (ESRI), a GIS application intensively used by technical staff

Problems to be faced

With respect to Intelligent Planning and Scheduling techniques, and regarding the above explained decision making life-cycle, temporal reasoning becomes one of the key aspects, as the system has to obtain temporal plans as a sequence of timed fighting activities, taking into account that top-level goals are constrained by deadlines (recall that experts define high-level procedures under a given time horizon). Additionally, in order to be suitable for validation and execution by technical staff, plans must follow standard fighting protocols, that is, the planning process should obtain plans by making decisions as if they were made by human experts. This has led us to develop a domain independent HTN planning system able to accept a hierarchical domain description based on the fighting protocols and perform a reasoning process that decomposes top-level tasks into subtasks, following the guidelines encoded in the domain. The novelty of our HTN planner, with respect to Intelli gen Planning and Scheduling techniques, is two-fold: on the one hand, it accepts HTN domains based on a hierarchical extension of the PDDL standard and, on the other hand it integrates temporal reasoning on tasks at every level of detail in the task hierarchy, using Temporal Constraint Networks as the underlying temporal representation formalism.

Furthermore, scalability and time processing requirements have also to be considered: fire attack plans range from one hundred fighting operations performed by about ten fighting means (in small extinction episodes), to several hundred actions and tens (or few hundred) of means involved. As the system has to provide several tentative plans, the planning process should allow to obtain several plans in a short time in order to be evaluated by experts, that will be able then to define the most suitable strategy to be followed. As will be shown, maintaining the causal structure in the plan allows to obtain an efficient HTN temporal planner that outperforms execution times of SHOP (Nau *et al.* 2001), an HTN planner widely used in several practical applications (Nau *et al.* 2005).

Considering the dynamics of the domain as well as the revision requirements, it is necessary to adopt a solution to the management of uncertainty in the execution of the plan. We have designed a plan execution and monitoring process, integrated with the output of the deterministic planner, in order to provide support to ex cution failures that raise a later plan revision process.

With respect to the integration of AI Planning technology in a real application, the SIADEX planning life-cycle must be seamlessly integrated and coupled with current staff’s workflow and decision making life-cycle, a strong practical requirement that affects decisively the success of the application. This requirement has led us to design an architecture that integrates the planning process within the familiar working environment of technical staff, thus enhancing the role of end-users. Furthermore, although users are not expected to have a background knowledge on AI, they require to affect the knowledge of the planner, and the architecture must offer solutions to this issue and to the following ones: *flexible knowledge management an integration*, since the system must represent and provide access to the large amount of data coming from heterogeneous sources of information and needed to make decisions, *support for ubiquitous access of end-users*, since the system is operated in a hostile and distributed environment and most of the inputs come from (and most of the outputs are directed to) end-users located at different places, and *integration with legacy software*, since the planning process income as well as the outcome must be redirected to legacy software so that end users may painlessly understand, process and deliver activity plans.

The rest of this work is devoted to first describe the architecture of the system, and then to explain the different stages of the planning process life-cycle successfully integrated into experts decision making life-cycle.

The architecture

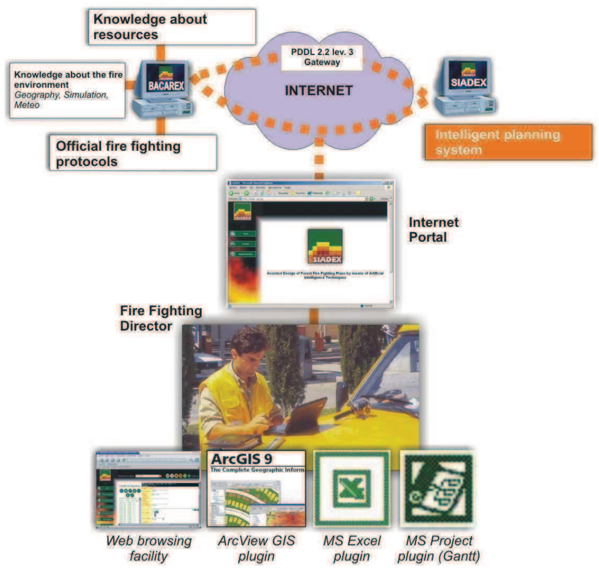

Knowledge about resources
Knowledge about the fire environment
Geography, Simulation, Meteo
BACAREX
Official fire fighting protocols
PDDL 2.2 lev. 3 Gateway
INTERNET
SIADEX
Intelligent planning system
Internet Portal
SIADEX
Assisted Design of Forest Fire Fighting Plans by means of Artificial Intelligence Techniques
Fire Fighting Director
ArcGIS 9
The Complete Geographic Inform
Web browsing facility
ArcView GIS plugin
MS Excel plugin
MS Project plugin (Gantt)

Figure 2: The architecture of SIADEX

SIADEX has been developed as a general, service-oriented architecture (Figure 2) to support decisions on crisis management episodes such as forest fire fighting. Its main components, implemented as web services, are the knowledge base (named BACAREX), that stores in Protege ((National Library of Medicine)) all the knowledge that would be useful for the planning engine, and the planning module (named SIADEX), the core of the architecture in charge of building fire fighting plans.

The planning algorithm and its knowledge representation are built as two independent servers (the ontology server and the planning server respectively) that provide services under a TCP/IP connection (OS independent) and can be accessed from any device with internet connectivity (a desktop computer, a laptop or a PDA). The architecture also includes a plan execution and monitoring web service in charge of tracking all the changes produced by the execution of the actions in a fire fighting plan. All the communication between the modules of the architecture is performed using the XML-RPC Protocol[3].

The system interface is composed of several front-ends, plugged into the legacy software that conforms the familiar working environment of technical staff, devoted to support interaction in every stage of the decision making process. Considering the decision making life-cycle above described, the process followed by experts when making decisions supported by SIADEX includes the following summarized steps:

Problem description In this step the fire fighting scenario is introduced by the technical staff through an user-friendly interface implemented as a software plug-in of the ArcView GIS (ESRI).

Knowledge integration The scenario introduced is stored in BACAREX and then integrated with the remaining of the knowledge needed by the planning process, such as standard protocols, fighting resources and other environmental information. Therefore, knowledge about resources and fire scenario share the same representation, and all this information is visible to other users by means of a web browsing facility.

Requesting a plan Once the scenario is integrated into the knowledge base, and when requested by the user, the planning engine is called and a plan is obtained. The planning engine is not able to read the domain and the problem stored in Protege, therefore a PDDL Gateway has been implemented that translates problem and domain into an HTN extension of PDDL 2.2 level 3 (Edelkamp & Hoffmann 2004).

Displaying the plan The plan obtained is delivered in XML format and may be displayed in a number of "user-friendly" alternatives like Microsoft Excel, in the form of a chronogram, or Microsoft Project in the form of a Gantt chart.

Plan execution and monitoring The plan may be launched for execution, distributed amongst all the technical staff with some responsibility in the fire fighting episode, and concurrently monitored.

Next sections describe in detail every stage of the process so far explained.

Problem description

Taking into account the description of the decision making cycle performed by fighting experts, this step starts after the environmental information has been gathered and the fire simulation is performed. This information is currently recorded and managed using ArcView (ESRI), a GIS application intensively used by technical staff. Hence, trying to maintain the integration with legacy software and aimed to provide a user friendly interface, we have developed an interface implemented as a plug-in of Arcview, devoted to visually introduce the information about the fire scenario.

[3]XML-RPC: XML Remote Procedure Call http://www.xmlrpc.com

Through this interface, and after the simulation, the fire director defines in an effortless way (using basic drawing tools over a geographical map) the operational units of the forest fire: the sectors. These are relevant areas of the environment where the attack will be focused (see Figure 3). Every fire scenario may have several sectors and every sector may be composed into operational targets like fire lines, control lines and spraying areas.

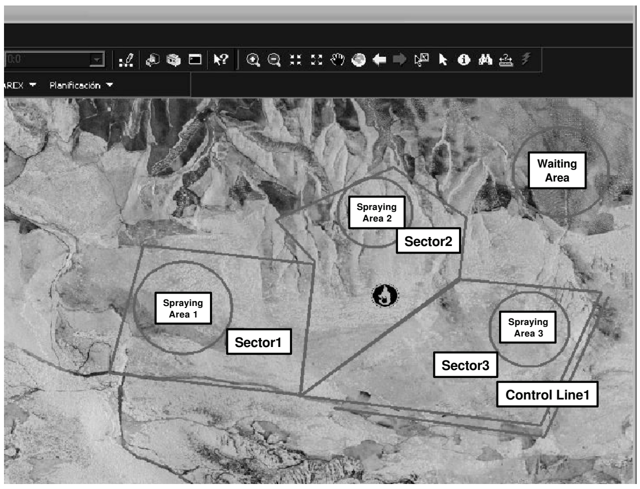


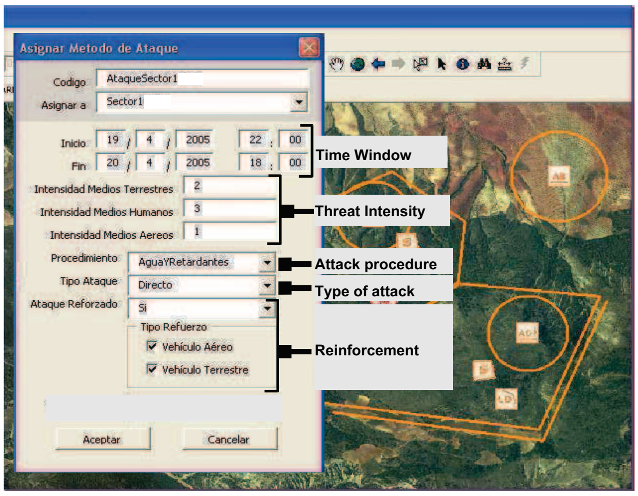
Figure 3: A simulated fire scenario with three sectors. Every sector contains a spraying area, Sector3 contains a control line. A waiting area, close to the fire, is also shown. Sectors, spraying areas and the control line are considered operational targets, the waiting area is devoted to logistics and intendence operations.

Once the sectorization has been defined, the fire director establishes the strategy to be carried out in the scenario. The strategy is defined as a set of top-level tasks to be accomplished at every operational target, a time window of activity associated to every task, and the amount of resources to be assigned to such tasks (see Figure 4). At present, and following the guidelines of technical staff, the amount of resources to be assigned is defined as three integer values: the number of human resources (squads), the number of aerial vehicles and the number of ground vehicles. These values are interpreted as a measure of the fire threat intensity and the planner is in charge of determining the instances of resources to be concretely assigned to every task.

The information introduced in this step is sent to the ontology server that is involved in the following step devoted to integrate the fire scenario with the remaining of the contents stored in the knowledge base.

Integrating knowledge

The ontology server is in charge of performing all the processes related to knowledge integration. It contains an ontology that has been designed with Protégé 2000 [4] that supports the integration and management of several categories



Asignar Metodo de Ataque

Codigo	AtaqueSector1
Asignar a	Sector1
Inicio	19 / 4 / 2005 22 : 00
Fin	20 / 4 / 2005 18 : 00
Intensidad Medios Terrestres	2
Intensidad Medios Humanos	3
Intensidad Medios Aereos	1
Procedimiento	AguaYRetardantes
Tipo Ataque	Directo
Ataque Reforzado	Si

Tipo Refuerzo
- ✓ Vehículo Aéreo
- ✓ Vehículo Terrestre

Figure 4: A description of the strategy to be carried out in Sector1 described as a top-level task including the following information: **perform a direct attack**, on the **operational target** Sector1, using a combination of procedures with water and fire retardants, and reinforced by aerial and terrestrial vehicles; in order to mitigate the **fire with an intensity** of 2 terrestrial vehicles, 3 squads and 1 aircraft; within an **activity window** of 20 hours (start date: 19/04/2005 22:00, end date: 20/04/05 18:00).

of knowledge, including fighting resources, geographical information and fighting protocols. It also provides several knowledge interfaces depending on the requester needs. Details about the knowledge stored and the interfaces provided are explained in the following.

Categories of knowledge

The most important part of the ontology is devoted to fire fighting means which are modeled either as material resources or human resources (Figure 5). Material resources may be facilities like operation bases, airports, etc and they represent static objects since most of their attributes will remain unchanged although they are very important like for example, the geographic position of an airport, the availability of refueling and water recharging facilities, etc. The remaining resources, either vehicles or human workers, are very dynamic since many of their attributes (also called *operational slots*) change during the execution of plans, for example their geographic position, their state of availability, the work they are carrying out, etc.

Temporal knowledge is also represented in the ontology. The main source of such category of knowledge comes, on the one hand, from legal issues (such as the maximum duration of the shifts of the squads or the fire directors, *Legal Shifts* in Figure 5), or periods of availability of contracted for aerial resources (*Contract Availability* in Figure 5). On the other hand, as previously seen the strategy defined in the fire scenario contains top-level tasks to be accomplished in a given operational area, between a *time window* described as start and end dates that constraint all the low level firefighting operations to be carried out as part of the strategy.

[4] with more than 130 classes and more than 2000 instances only for planning objects and without taking into account the representation of activities

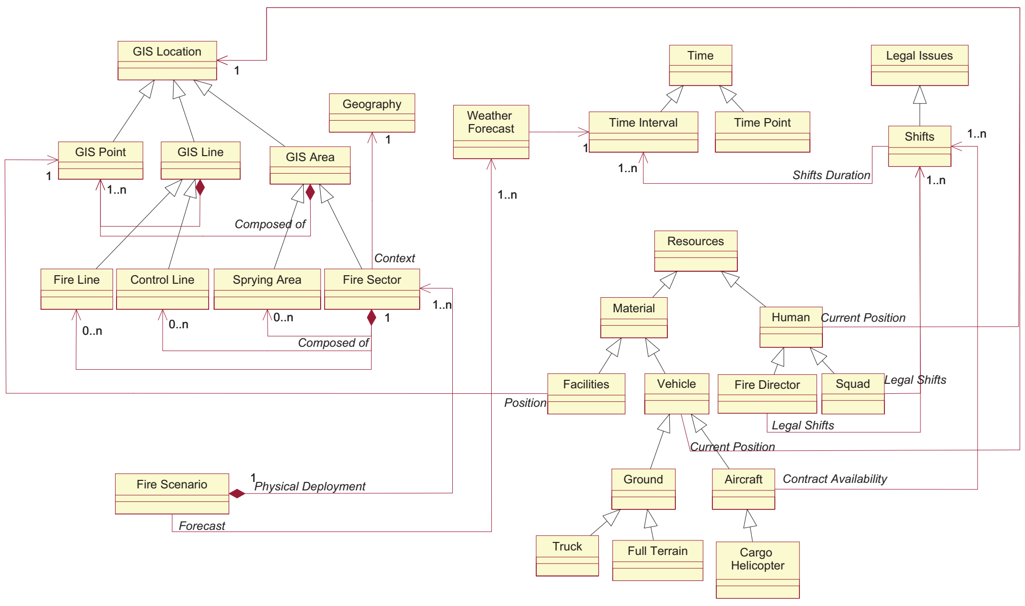


Figure 5: A UML representation of a simplified part of the ontology of SIADEX

Therefore, this temporal information is also represented.

Geographical and environmental information, such as the sectorization of a fire scenario or weather forecasts, is also stored since decisions made by experts following the fighting protocols depend on this crucial information.

Finally, another part of the knowledge base is devoted to represent knowledge about operational tasks and fighting protocols. These standard operating procedures are directly represented as a hierarchy of tasks following an HTN domain description, and details an examples about this representation are shown in a later section.

Knowledge interfaces

The ontology server also provides different services depending on the requester needs:

- From the user point of view it provides a common place to easily query browse and update the information, as it integrates in a single model heterogeneous knowledge coming from different sources such as GIS or external resource management data bases (extracted by plug-ins and mappers). It also provides both offline and on-line access facilities. Offline access is done by the standard Protégé framework, so that the development team may access the knowledge and carry out maintenance and validity checking operations with full operability. On the contrary, online access is done through the Protégé API on which a web access service has been built. This on-line access is devoted to staff users that do not have skills on knowledge representation for planning but may painlessly access the knowledge in a web browsing fashion by means of a hierarchy of objects and activities close to their understanding of the problem.

- From the point of view of the plan execution and monitoring service, the ontology is used to maintain updated the situation of means and resources as fighting operations are executed during the plan progress.

- From the planning server point of view the ontology is used to obtain the problem and the domain. As seen above the domain is directly represented in an HTN formalism. With respect to the problem, it is composed of, on the one hand, the initial state represented in the ontology as both, the values of operational slots of means and resources instances, and the information about the geographical deployment of the fire. On the other hand, the top-level goal, that is obtained by querying the information about the top-level tasks introduced in the strategy.

This last knowledge interface is indeed a process that carries out the translation of the problem defined by the user from the ontology representation to the representation used by the planner. This process is triggered when the user requests the planning services, the next step after storing the information of the scenario.

Requesting planning services

When requested by the user, the planning server gets the domain and problem representation sent by the ontology server and triggers the planning process on these inputs. The planning algorithm implemented in this server is based on an HTN algorithm that integrates temporal reasoning using Simple Temporal Networks as the underlying temporal representation mechanism. This section describes the most relevant details of the planning process and the domain and problems representation used in SIADEX, explaining the reasons that lead us to adopt this solution. Finally, the translation process that extracts the problem from the ontology to the planner is outlined.

HTN and temporal reasoning

As previously noted, in order to be validated by experts, plans obtained must follow standard operating procedures and, therefore, fighting protocols should be encoded in the domain description in order for the planner to make them operational. Standard PDDL actions as well as PDDL 2.2 level 3 durative actions (Edelkamp & Hoffmann 2004) provide enough expressivity to describe operational, lowest-level fighting activities (mainly due to the use of fluents to support resource reasoning, and the use of temporal constraints such as action duration as shown in Figure 6), but durative-actions are not expressive enough to represent high-level processes such as those specified in fighting protocols. Protocols are a process specification at different levels of detail that also includes constraints on the order of tasks to be carried out. This kind of specification is best suited by the HTN (Nau *et al.* 2001; Erol, Hendler, & Nau 1994) formalism, where planning domains are designed in terms of a hierarchy of tasks representing compound and atomic activities (called *compound tasks* and *primitive tasks or operators* respectively). In such hierarchical domains, it is possible to describe how every compound task may be decomposed into different subtasks and the order that subtasks must follow, by using different *methods*.

```
(:durative-action move
  :Parameters (?g - Group ?v - Vehicle ?p1 ?p2 - GIS_Location)
  :duration (= ?duration (/ (distance ?p1 ?p2) (cruise ?v)))
  :condition (and (current_position ?v ?p1)
                  (current_position ?g ?p1))
  :Effect (and (not (current_position ?v ?p1))
               (not (current_position ?g ?p1)
               (current_position ?v ?p2)
               (current_position ?g ?p2)
               (decrease (current_autonomy ?v) ?duration))
```

Figure 6: A PDDL durative action representing the movement of a human group transported by a vehicle. Temporal (duration of the transport) and resource constraints (the autonomy of the vehicle decreases) have proved to be expressive enough in this practical application.

Motivated by the lack of expressivity found in PDDL durative-actions, we have developed an HTN extension of the PDDL standard in such a way that hierarchical domains in SIADEX encode primitive operators as PDDL 2.2 level 3 durative actions (Edelkamp & Hoffmann 2004) . In addition, basic issues of compound tasks included in our representation are inspired by the domain description language of SHOP (Nau *et al.* 2001), where the methods used to decompose tasks into subtasks include a precondition that must be satisfied by the world state in order for the method to be applicable by the planner. Therefore, the basic algorithm of our planner follows the HTN paradigm of SHOP: it is a state-based forward HTN planning algorithm that uses an ordered task decomposition search-control strategy. Thus tasks are decomposed according to the order in which they will be executed and methods are selected considering the current state.

But this is only the basics of SIADEX, and further extensions has been introduced to the basic representation for tasks above described, as well as in the planning algorithm, in order to overcome some recognized weaknesses of state-based forward planners (either HTN-based, like SHOP, or not) regarding practical applications. Usually this category of planners return plans as a totally ordered sequence of actions but, in many real world applications such as forest fire fighting, activities may be executed concurrently, and a total order of activities is not very appropriate for practical reasons. The domain description language of SIADEX and the planning algorithm support to explicitly represent and manage time and concurrency in both compound and primitive tasks (increasing the expressiveness of the planner), thanks to the handling of metric time over a Simple Temporal Network (See (Castillo *et al.* 2005b; 2005a)) for more details.

This is a novel representation that provides a great expressivity power in order to support two fundamental requirements of experts: on the one hand, the representation of deadline top-level goals, as required in the top-level tasks formulation shown in the problem description in Figure 4; on the other hand, it must also be possible to define complex control structures required by the representation of protocols such as sequencing, splitting and synchronization of high-level processes. Both issues are described next.

Deadline goals

As shown in the section devoted to present the problem description process, the earliest stage of decision making requires to pose one or more metric temporal constraints over both the start and the end of a top-level task. Furthermore, the underlying aim of a user while imposing constraints to such top-level task is that its component subtasks also inherit them. SIADEX allows to post deadline constraints, and to inherit them in the case of subtasks of a given task, on either the start, the end (or both) time points of any task either primitive or not. Any task has two special variables associated to it: `?start` and `?end` that represent its start and end time points, and some constraints (basically `<=, =, >=`) may be posted to them. In order to do that, when describing a decomposition method, any subtask may be preceded by a logical expression that defines the desired deadline.

Deadline goals are easily encoded in the STN of a temporal plan as absolute constraints with respect to the absolute start point of the STN. They may also appear in the top-level goal and they are very useful for defining time windows of activity like the one shown in the example of Figure 4. The top-goal formulated following this example is shown in Figure 7.

```
(define (problem FireOne)
(:domain encoded-protocols)
(= :time-start "19/4/2005 17:0:0")
(= :time-unit :minutes)

(:objects
  ...
  (is-part-of Fire Sector1)
  (is-part-of Fire Sector2)
  ... )

(:tasks-goal
:tasks (and (>= ?start 300) (= ?end 1500)) (attack Fire Sector1)
))
```

Figure 7: An example of a problem described following the HTN extension of PDDL where some information about the start time of the fire is included, as well as the top-level goal, that is formulated in terms of an instance of a top-level task `(attack ?f - GIS_Fire ?s - GIS_Sector)` with a deadline. According to the example of Figure 4 the attack to Sector1 starts 5 hours after the beginning of the fire and must end 20 hours later.

```
(:TASK attack
 :PARAMETERS (?f - GIS_fire ?s - GIS_Sector)
 (:METHOD not-reinforced
  :PRECONDITION (not (reinforced-attack ?s))
  :TASKS ((Select_Means ?f ?s)
          (Mobilize_Deploy_All_Groups ?f ?s)))
 (:METHOD reinforced
  :PRECONDITION (reinforced-attack ?s)
  :TASKS ((Select_Means ?f ?s)
          [ (Mobilize_Deploy_All_Groups ?f ?s)
            (Mobilize_Deploy_All_Aerial ?f ?s)
            (Mobilize_Deploy_All_Terrestrial ?f ?s)]))
```

Figure 8: An HTN task implementing a standard operating procedure to attack a sector ?s in a forest fire ?f. The doctrine establishes that, in case of not being a reinforced attack, first select appropriated human means, then deploy and mobilize them. In case of reinforced attack, first select means, then deploy every human, aerial and terrstrial mean selected.

Ordering constraints and synchronization between tasks

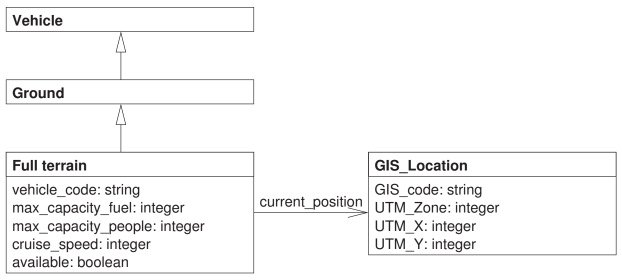
A natural description need for standard operating procedures such as fighting protocols used by technical staff is given by the use of different control structures for sequencing, splitting and synchronizing processes, included in almost all languages devoted to process specification such as workflow languages (OWL-S 2003; BPEL). As shown in Figure 8, domain descriptions in SIADEX also allow to specify such control structures between the subtasks of a method. Tasks might be either sequenced, and then their signature appears between parentheses `(T1,T2)` , or splitted, appearing between braces `[T1,T2]`. For example, the second method of Figure 8 specifies that human, aerial and terrestrial means should be concurrently (if possible) mobilized and deployed after being selected and assigned to a sector. However, these order structures represent qualitative order constrains that are necessary but not sufficient in order to allow practical execution of concurrent tasks since, in practical applications, synchronization between tasks is needed. This can be done in SIADEX due to the quantitative, temporal constraints included in the STN of a temporal plan that are not only extracted from the order relations between task, but also from the causal structure allowing to manage timed causal links (See for more details (Castillo *et al.* 2005b; 2005a)). For example, as subtasks of the top-level task of Figure 8 inherit the temporal constraints posed in the problem (see in Figure 7), the method *reinforced* shown in Figure 8 represents a split-join struct as all the splitted subtasks must end at the same time that the top-level task.

Once the basic issues about the planner and the domain representation has been explained, next we will the describe how the problem, initially stored in the ontology server, is translated into a problem representation understandable by the planner.

Getting the problem

The initial state of the problems managed by SIADEX maintains basically the same representation that in the PDDL standard, and it is expressed as a conjunction of literals that represent the set of facts that are known to be true. It may be obtained by querying the ontology about the main slots of the instances of classes like facilities, human resources, vehicles, water points, etc. This process is crucial when trying to maintain the planning knowledge as transparent as possible to end-users. In order to do that, a PDDL gateway has been designed so that, at the beginning of a planning episode and when requested by the user, this gateway iterates over the whole ontology and translates the content of the main instances into PDDL in a process whose main features are the following ones:

- The classes hierarchy of the ontology is translated as a hierarchy of types in the PDDL domain description. As illustrated above, the parameters of compound and primitive tasks are typed regarding this translated hierarchy of types.
- Predicates are easily translated since every operational slot of the ontology (that is, whose content might be relevant for the planning process) has a special template on how to translate the content of the slot into a PDDL literal. Binary slots of the form `Instance.slot=value` are easily translated into PDDL literals of the form `(slot instance value)`. Non binary predicates or slots that represent references to other instances have a similar translation process, and even some of the slots may produce multiple literals (see Figure 9). .
- Slots that contain numerical values that may change during a planning episode, mainly numerical resources like the current autonomy of a vehicle, are declared like fluents in the PDDL domain, so that the use of arithmetic operations and functions are permitted over them and they allow the planner to reason about the use of resources.
- Slots representing temporal knowledge (like shifts of workers) or environmental information (like weather forecasts of day/night events) are translated into PDDL 2.2 timed initial literals.



```
(current_position vehicle_code GIS_code) and
(coordinates GIS_code UTM_Zone UTM_X UTM_Y)
```

Figure 9: Translation of some references in the ontology

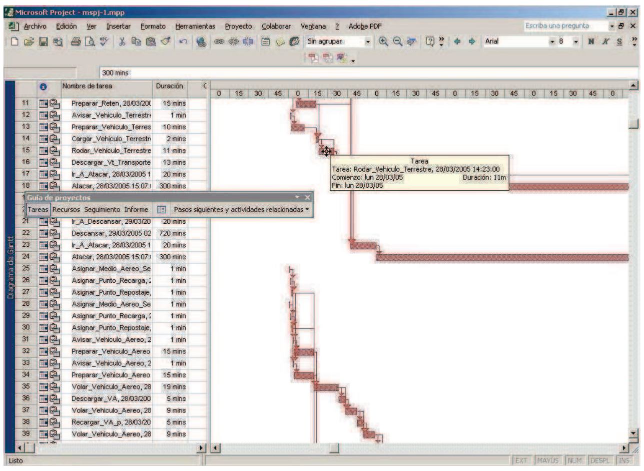
The top-level goal is also extracted from the ontology using the knowledge stored in the initial step of problem description. As shown in a previous example of this section, and considering the goal formulation of Figure 7, the top-level goal is composed of the top-level tasks defined by the user, and the time window defined by the user is translated as temporal constraints on this task. In addition, considering that the geographical deployment of the fire episode is stored as instances into the ontology, and that the slots of the instances are also considered as operational, the information about the sectorization is also translated as part of the initial state.

In addition, as also shown in Figure 7, every top-level task defined as part of the strategy contains information about the numeric evaluation of the power of resources that should be assigned to it and that is fixed by hand by the fire fighting director. From these values, the planner may pre-select several combinations of resources, that are submitted to the fight director who finally selects one of them. Later, the planner, and following standard protocols, will select a specific set of instances of fighting resources that fits within the selected combination.

Finally, thanks to the expressive power of using Temporal Constraint Networks as the underlying formalism to represent temporal knowledge, the planner is able to obtain temporal attack plans, that is plans whose actions are executed in a time line that is flexibly constrained to the time windows established in the definition of the goal. These plans are also suitable for being evaluated by users, as they are generated by following standard procedures encoded in the domain description, that has been proved to be expressive enough to describe complex ordering and synchronization mechanisms between tasks. The plans so obtained are served by the planning server in order to be visualized, analyzed and, if accepted, monitored, as is discussed in the following section (for more details, see (Castillo, Fdez-Olivares, & González 2002; de la Asunción *et al.* 2004)).

Plan visualization and monitoring

Plans obtained may be displayed in a number of "user-friendly" alternatives like Microsoft Excel, in the form of a chronogram, or Microsoft Project in the form of a Gantt chart (see Figure 10). These interfaces are used by expert in order to evaluate the suitability of the decisions made when defining the strategy in the problem description stage. If a plan is evaluated and accepted by the fire director, it is dispatched for execution, and distributed amongst all the technical staff with some responsibility in the fire fighting episode, and concurrently monitored.


Microsoft Project - mspj-1.mpp
Archivo Edición Ver Insertar Formato Herramientas Proyecto Colaborar Ventana ? Adobe PDF
Sin agrupar
Arial
Guía de proyectos
Tareas Recursos Seguimiento Informe
Pasos siguientes y actividades relacionadas
Diagrama de Gantt
Listo

300 mins

	Nombre de tarea	Duración
11	Preparar_Reten, 28/03/20(	15 mins
12	Avisar_Vehiculo_Terrestre	1 min
13	Preparar_Vehiculo_Terres	10 mins
14	Cargar_Vehiculo_Terrestr	2 mins
15	Rodar_Vehiculo_Terrestre	11 mins
16	Descargar_Vt_Transporte	13 mins
17	Ir_A_Atacar, 28/03/2005 1	20 mins
18	Atacar, 28/03/2005 15:07:	300 mins
21	Ir_A_Descansar, 29/03/20	20 mins
22	Descansar, 29/03/2005 02	720 mins
23	Ir_A_Atacar, 28/03/2005 1	20 mins
24	Atacar, 28/03/2005 15:07:	300 mins
25	Asignar_Medio_Aereo_Se	1 min
26	Asignar_Punto_Recarga,	1 min
27	Asignar_Punto_Repostaje,	1 min
28	Asignar_Medio_Aereo_Se	1 min
29	Asignar_Punto_Recarga,	1 min
30	Asignar_Punto_Repostaje,	1 min
31	Avisar_Vehiculo_Aereo, 2	1 min
32	Preparar_Vehiculo_Aereo	15 mins
33	Avisar_Vehiculo_Aereo, 2	1 min
34	Preparar_Vehiculo_Aereo	15 mins
35	Volar_Vehiculo_Aereo, 28	19 mins
36	Descargar_VA, 28/03/200	5 mins
37	Volar_Vehiculo_Aereo, 28	9 mins
38	Recargar_VA_p, 28/03/20	5 mins
39	Volar_Vehiculo_Aereo, 28	9 mins

Tarea
Tarea: Rodar_Vehiculo_Terrestre, 28/03/2005 14:23:00
Comienzo: lun 28/03/05 Duración: 11m
Fin: lun 28/03/05

Figure 10: Gantt chart output

Plan execution and monitoring is also implemented as a web service and incorporates a web interface. Through this interface experts might supervise in real time the execution of the plan, and track all the changes produced by the execution of the actions in the plan, the time they take to execute and it updates the ontology accordingly so that these changes are publicly available for any query at any time.

The monitoring algorithm (de la Asunción *et al.* 2004; 2005) is a real time algorithm that follows the execution of the temporal plan at the highest level of detail since temporal plans, as they are represented on top of a STN, still represent flexibly (by means of an interval of time) the time at which every effect of every action is achieved. As explained above it is possible to check that everything executes as predicted, otherwise, it is possible to detect and, in some cases repair, some of the problems without the need of starting a new planning episode[5]. The type of problems and their possible solutions are the following ones:*local delays* (they only affect to an isolated branch of the plan, and thus, a new local reschedule may be found only for that branch), *global delays* (they might affect all the remaining actions of the plan, and a whole new reschedule might be needed), and *infeasible delays* (no reschedule is possible, and a new planning episode considered). Finally, experts may also detect some perturbation in the execution of a plan and, in this case, a new cycle of decision making supported by SIADEX might be started

A short note on efficiency

Just to give an idea of the performance of SIADEX, we include here a short comparison in the ZENO domain of the International Planning Competition 2002 (Long &

[5]Clearly, the user may also interrupt the execution of the plan at any moment.

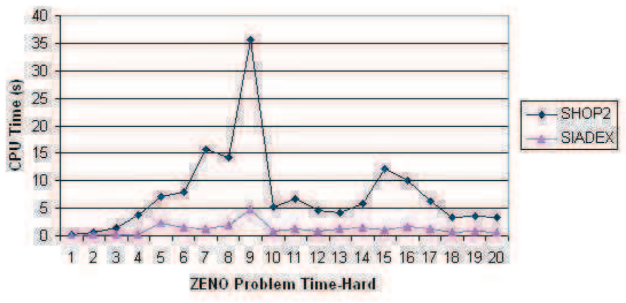


Figure 11: Performance of SHOP2 and SIADEX in zeno-travel time problems. Y-Axis represents computation time in seconds and X-Axis represents the set of test problems.

Fox 2003). Figure 11 shows the CPU times of SIADEX and SHOP2 (Nau *et al.* 2003) running on the same machine and operating system, solving all the hard instances (time+numeric) of this domain, and it may be seen that SIADEX outperforms SHOP2 quite clearly.

Additionally, two other experiments are shown. Figures 12.a) and b) show the performance of SIADEX solving two real problems, named INFOCA-1 and INFOCA-2. The X-Axis shows the number of actions in the resulting temporal plan and the Y-Axis shows CPU time in seconds. In order to give a measure about the kind of real problems we are talking about it must be said that the size of the initial state is about 14000 literals, translated from about more than 130 classes and more than 2000 instances. At present the domain contains about 107 tasks, 121 methods, and 55 durative-actions [6].

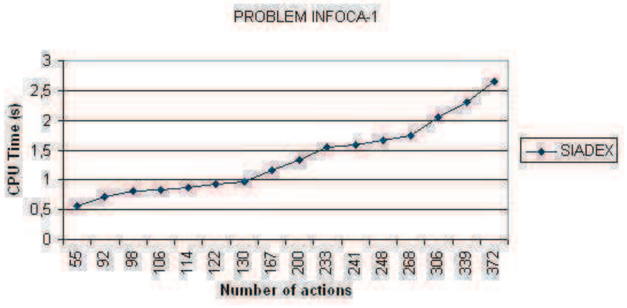
Related work

Regarding general crisis situations SIPE (Bienkowski 1995) has been used for obtaining plans of attack for oil spill threats in the sea. Although its architecture is very similar to SIADEX, its main drawback is that it requires end users to have a deep knowledge of planning techniques either to interact or to provide knowledge for the system.

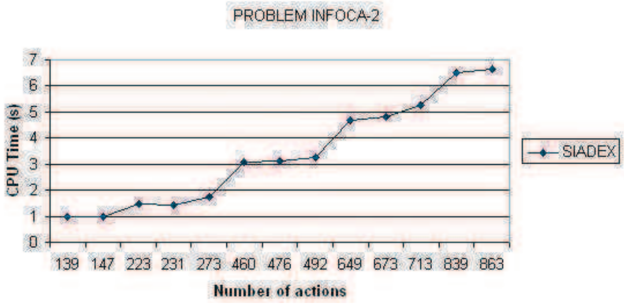
Another system used in civil crises was that described in (Biundo & Schattenberg 2001) where hybrid hierarchical techniques were developed to obtain plans of response for floods in Germany, but as far as authors know, it does not have monitoring or temporal reasoning integrated with HTN, what impedes severely its use in real environments.

Specifically in the field of forest fire fighting there are several approaches in the literature: PHOENIX (Cohen *et al.* 1989) (1989-1993) , CHARADE (Avesani, Perini, & Ricci 2000) (1992-1995), CARICA (Avesani, Perinni, & Ricci 1997) (1995-1997) or the work presented in (Rollon *et al.* 2003). However, they have failed in their application as assistants to real fire fighting scenarios. The reasons for these unsuccessful approaches are: (1) They have neglected the development of appropriated knowledge integration techniques, considering user interaction at edition level, not allowing users to affect the knowledge of the planner; (2)The lack of deliberative planning techniques able to flexibly generate new plans for a situation without the requirement of having a predefined skeleton or general plan previously stored; (3) None of these systems integrates HTN reasoning with temporal reasoning, what provides a great expressivity in order to encode fighting protocols; (4) Absence of plan execution and monitoring techniques to allow flexibility and responsiveness in the execution of the plan.

[6]Please consult http://siadex.ugr.es for more details on this real application



(a)



(b)

Figure 12: Performance of SHOP2 and SIADEX in zeno-travel time problems. Y-Axis represents computation time in seconds and X-Axis represents the set of test problems.

Conclusions and Lessons Learned

In this work we have described the solutions adopted to the problems tackled during the development of SIADEX, an intelligent planning and scheduling application where users play a central role. SIADEX obtains temporal fire attack plans in a very time-efficient planning process, providing support to monitor their execution, and allowing to revise experts decisions by re-defining the strategy (if needed) by a continuous interaction with the user in every stage of the crisis episode.

The main lesson learned from the development of SIADEX is that, in order to be widely used, planning technology has to be developed assigning to users a central role, trying to bring the technology near to the user. This means that in order to provide solutions to the real, common problems of non AI expert users planning and scheduling processes must be part of a wider, overall architecture integrated in legacy software, such as the one presented here, allowing an easy to learn interaction process in every stage of the planning life-cycle. As users in forest fire fighting usually work in different kinds of (even hostile) environments, during the development of the arquitecture of SIADEX we have had to solve two additional problems: how to integrate a large amount of heterogeneous knowledge coming from different sources, and how to provide ubiquitous and concurrent access to users.

But the central role of the users not only affect to outside issues of planning and scheduling. This work also shows that, as plans must be finally evaluated by experts, they must follow standard fighting protocols and, thus, the planning process should obtain plans by making decisions as if they were made by human experts. This has led us to develop new techniques, extendindg current state of the art planning and scheduling techniques. The novelty of the planner we have developed, with respect to Intelligent Planning and Scheduling techniques, is two-fold: on the one hand, it accepts HTN domains based on a hierarchical extension of the PDDL standard and, on the other hand it integrates temporal reasoning on tasks at every level of detail in the task hierarchy, using Temporal Constraint Networks as the underlying temporal representation formalism. This provides support for two fundamental requirements of experts: on the one hand, the representation of deadline top-level goals, as required in the top-level tasks formulation when defining the fire scenario; on the other hand, it must also be possible to define complex control structures required by the representation of protocols such as sequencing, splitting and synchronization of high-level processes.

Furthermore, due to the reponse times demanded by users in their decision making cycle, time processing requirements have been also considered, and we have shown that our planner is extremely efficient regarding the type of planning and scheduling real problems to be solved (hundred of actions managing tens, even hundred of fighting resources).

This project is presently at a 75% of development and there are still pending issues, the most important is, the definition of a plan repairing and replanning process based on mixed initiative techniques. At present we are centering our efforts in this ongoing work.

References

Avesani, P.; Perini, A.; and Ricci, F. 2000. Interactive case-based planning for forest fire management. *Applied Intelligence* 13(1):41–57.

Avesani, P.; Perinni, A.; and Ricci, F. 1997. CBET: a case-based exploration tool. In *Fifth Congress of the Italian Association for A.I.*

Bienkowski, M. 1995. Demonstrating the Operational Feasibility of New Technologies: The ARPI IFDs. *IEEE Expert* 10(1):27–33.

Biundo, S., and Schattenberg, B. 2001. From abstract crisis to concrete relief - a preliminary report on combining state abstraction and htn planning. In *6th European Conference on Planning (ECP-01)*.

BPEL, T. Bpel4ws:business process execution language for web sevices (version 1.1). Technical report.

Bresina, J. L.; Jonsson, A. K.; Morris, P.; and Rajan, K.

Castillo, L.; Fernández-Olivares, J.; García-Pérez, O.; and Palao, F. 2005a. Efficiently handling temporal knowledge in an htn planner. In *Submitted to ICAPS06*.

Castillo, L.; Fernández-Olivares, J.; García-Pérez, O.; and Palao, F. 2005b. Temporal enhancements of an HTN planner. In *CAEPIA 2005*.

Castillo, L.; Fdez-Olivares, J.; and González, A. 2002. A temporal constraint network based temporal planner. In *Workshop of the UK Planning and Scheduling Special Interest Group, PLANSIG 2002*, 99–109.

Cohen, P.; Greenberg, M.; Hart, D.; and Howe, A. 1989. Trial by fire: understanding the design requirements for agents in complex environments. *AI Magazine* 10(3):32–48.

de la Asunción, M.; Castillo, L.; Fdez-Olivares, J.; García-Pérez, O.; González, A.; and Palao, F. 2004. Handling fuzzy temporal constraints in a planning framework. To appear in the special issue of Annals of Operations Research on Personnel Scheduling and Planning.

de la Asunción, M.; Castillo, L.; Fdez-Olivares, J.; García-Pérez, O.; González, A.; and Palao, F. 2005. Siadex: an interactive knowledge-based planner for decision support in forest fire fighting. *AI Communications* 18(4).

Edelkamp, S., and Hoffmann, J. 2004. The language for the 2004 international planning competition. http://ls5-www.cs.uni-dortmund.de/ edelkamp/ipc-4/pddl.html.

Erol, K.; Hendler, J.; and Nau, D. 1994. UMCP: A sound and complete procedure for hierarchical task-network planning. In *AIPS-94*.

ESRI. Arc view gis software http://www.esri.com.

Long, D., and Fox, M. 2003. The 3rd international planning competition: Results and analysis. *Journal of Artificial Intelligence Research* 20:1–59.

National Library of Medicine. http://protege.stanford.edu/.

Nau, D.; Muoz-Avila, H.; Cao, Y.; Lotem, A.; ; and Mitchel, S. 2001. Total-order planning with partially ordered subtask. In *Proceedings of the IJCAI-2001*.

Nau, D.; Au, T.; Ilghami, O.; Kuter, U.; Murdock, J. W.; Wu, D.; and Yaman, F. 2003. SHOP2: An HTN Planning System. *Journal of Artificial Intelligence Research* 20:379–404.

Nau, D.; AU, T.; Ilghami, O.; Kuter, U.; Wu, D.; and Yaman, F. 2005. Applications of shop and shop2. *IEEE Inteligent Systems* 34–41.

OWL-S, T. 2003. Owl-s:semantic markup for web services. technical white paper (owl-s version 1.0. Technical report.

Rollon, E.; Isern, D.; Agostini, A.; and Cortés, U. 2003. Towards the distributed management of emergencies: forest fires case study. In *IJCAI Workshop on Environmental Decision Support Systems*.

Ruml, W.; Do, M. B.; and Fromherz, M. 2005. On-line planning and scheduling for high-speed manufacturing. In *Proceedings of the ICAPS05*, 2–11.

Efficiently handling temporal knowledge in an HTN planner*

Luis Castillo and **Juan Fdez-Olivares** and **Óscar García-Pérez** and **Francisco Palao**
Dpto. Ciencias de la Computación e I.A.
University of Granada, SPAIN
{L.Castillo,Faro,Oscar,Palao}@decsai.ugr.es
http://siadex.ugr.es

Abstract

This paper presents some enhancements in the temporal reasoning of a Hierarchical Task Network (HTN) planner, named SIADEX, that, up to authors knowledge, no other HTN planner has. These new features include a sound partial order metric structure, deadlines, temporal landmarking or synchronization capabilities built on top of a Simple Temporal Network and an efficient constraint propagation engine boosted by exploiting the causal structure of plans.

Introduction

The achievement of an efficient and expressive handling of time is still a pending task for most HTN planners. This issue becomes even harder if the planner follows a state-based forward paradigm like SHOP (Nau *et al.* 2003) or SIADEX (de la Asunción *et al.* 2005) since, despite being a very fast HTN planning paradigm, it does not easily allow to obtain plans with timed concurrent branches. From a practical point of view, real world applications need the plans to have the possibility of executing several activities at the same time. Furthermore, real applications usually require complex synchronization mechanisms between the activities of the plan for a successful execution. And last, but not least, temporal knowledge has to be efficiently handled so as to allow a fast response of the planning system. This paper explains how the HTN planner SIADEX has been extended to cope with all these requirements thanks to the use of Simple Temporal Networks (STN) (Dechter, Meiri, & Pearl 1991). STNs have been widely used as the underlying representation of temporal constraints in planning and scheduling frameworks like in Mapgen (Ai-Chang *et al.* 2004), Mexar (Oddi *et al.* 2002), PASSAT (Myers *et al.* 2002), Ixtet (Laborie & Ghallab 1995) or OPlan (Tate, Drabble, & Kirby 1994); since they provide a very expressive power to represent a variety of temporal constraints and a flexible execution of plans with flexible timelines. All these approaches use temporal constraints of different nature to represent their own temporal knowledge. The main contribution of this paper is based on how temporal constraints are extracted and propagated in a HTN framework:

- Any temporal constraint, either precedence constraints or deadlines, defined on an abstract task, or between several of them, are implicitly inherited by its constituent subtasks.
- Abstract tasks and primitive actions may generate temporal landmarks on their start or end points to achieve complex synchronization schemas between them, either between tasks or between actions and tasks.
- PDDL 2.2 timed initial (Edelkamp & Hoffmann 2004) literals are easily represented in a STN framework and they are used as backtracking points that serve as anchorage points for tasks and actions.
- Despite being an HTN planner, the causal rationale of primitive actions is recorded and used to propagate accurate temporal constraints between them.
- Since HTN planners are used in many real applications with tight response times, the propagation algorithms defined for STN, although polynomial, still produce a considerable overload when handling large temporal plans in the order of hundreds or thousands of actions. Therefore, this paper also proposes a modification of a well known propagation algorithm named PC-2 (Dechter 2003) to boost its performance thanks to the information extracted from the causal structure of the plan, obtaining excellent results.

These contributions have raised in the framework of application of SIADEX devoted to forest fire fighting planning (Fdez-Olivares *et al.* 2006) due to the need to obtain plans with complex temporal requirements like the synchronization of several teams of workers, the existence of temporal windows of activity, precedence constraints due to the use of shared resources or dealing with timed exogenous events. However, they are very common needs for other real problems and therefore of general interest for other application areas.

The paper is structured as follows. Next section outlines SIADEX, an HTN state-based forward planner. Then, we will present the main extensions to SIADEX to cope with the most important representational issues of time in

*This work has been partially supported under the research contract NET033957/1 with the Andalusian Regional Ministry of Environment and the CICYT Project TIC2002-04146-C05-02

```
(:task travel-to
 :parameters (?destination)
 (:method Fly
  :precondition (flight ?destination)
  :tasks ((go-to-an-airport)
          (take-a-flight-to ?destination)))
 (:method Drive
  :precondition (not (flight ?destination))
  :tasks ((take-my-car)
          (drive-to ?destination))))
```

(a)

```
(:durative-action drive-to
 :parameters(?destination)
 :duration (= ?duration
            (/ (distance ?current ?destination)
              (average-speed my-car)))
 :condition(and (current-position ?current)
                (available my-car))
 :effect(and (current-position ?destination)
             (not (current-position ?current))))
```

(b)

Figure 1: The basics of HTN planning domains in SIADEX' domain language: (a) A compound *task* with two different *methods* of decomposition. (b) A primitive *action*.

a HTN framework. Later, we introduce the use of deadlines and their inheritance from tasks to actions and present a very simple improvement in temporal constraints propagation that achieves great experimental results in different domains. Last section relates this approach with the existing literature.

Description of SIADEX

SIADEX is a state-based forward HTN planner with the same foundations than SHOP (Nau *et al.* 2003). Before going into the details, some introductory notions are explained first.

HTN planning foundations

HTN planning domains are designed in terms of a hierarchy of compositional activities. Lowest level activities, named actions or primitive operators, are non-decomposable activities which basically encode changes in the environment of the problem. In SIADEX, these primitive operators are represented as PDDL 2.2 level 3 durative actions (Edelkamp & Hoffmann 2004) (Figure 1.b). PDDL is the standard planning domain description language and it is the basis of most well known planners. On the other hand, high level activities, named tasks, are compound actions that may be decomposed into lower level activities. Depending on the problem at hand, every task may be decomposed following different schemas, or methods, into different sets of sub-activities. These sub-activities may be either tasks, which could be further decomposed, or just actions (Figure 1.a).

Tasks and their, possibly multiple, decompositions encode domain dependent rules for obtaining a plan, that can only be composed of primitive actions. Other HTN features are

```
• Set $\mathcal{A}$, the agenda of remaining tasks to be done, to the set of high level tasks
  specified in the goal.
• Set $\Pi = \emptyset$, the plan.
• Set $\mathcal{S}$, the current state of the problem, to be the set of literals in the initial state.
  1. Repeat while $\mathcal{A} \neq \emptyset$
   (a) **Extract** a task $t$ from $\mathcal{A}$
   (b) if $t$ is a primitive action, then
      i. If $\mathcal{S}$ satisfies $t$ preconditions then
        A. Apply $t$ to the state, $\mathcal{S} = \mathcal{S} + additions(t) - deletions(t)$
        B. Insert $t$ in the plan, $\Pi = \Pi + \{t\}$
        C. **Propagate-Temporal-Constraints**($\Pi$)
      ii. Else **FAIL**
   (c) if $t$ is a compound action, then
      i. If there is no more decomposition methods for $t$ then **FAIL**
      ii. **Choose** one of its decomposition methods of $t$ whose preconditions are true
          in $\mathcal{S}$ and map $t$ into its set of subtasks $\{t_1, t_2, \ldots\}$
      iii. Insert $\{t_1, t_2, \ldots\}$ in $\mathcal{A}$.
  2. **SUCCESS**: the plan is stored in $\Pi$.
```

Figure 2: A rough outline of an HTN planning algorithm showing the point at which temporal constraints are propagated (step 1(b)iC).

the following ones:

- The initial state is a set of literals that describe the facts that are true at the beginning of the problem.
- Unlike non HTN planners, goals are not specified as a well formed formula that must be made true by the planner from the initial state. Instead, goals are described as a partially ordered set of tasks that need to be carried out.
- The main planning algorithm (Figure 2) takes the set of tasks to be achieved, explores the space of possible decompositions replacing a given task by its component activities, until the set of tasks is transformed into a set of primitive actions that make up the plan.

Including inference capabilities in SIADEX

HTN planning approaches have a very rich knowledge representation that may arise in a variety of forms, from methods preconditioning and control of the search (Nau *et al.* 2003), ontology based representation of planning objects (Gil & Blythe 2000; Fdez-Olivares *et al.* 2006) or knowledge-intensive planning procedures (Wilkins & desJardins 2001). In our case we use two augmented inference capabilities that allows the planner to infer new knowledge, either by abduction or deduction, over the current state of the problem at every planning step.

Regarding the use of deductive inference, SIADEX provides inference tasks that may be fired when needed in a task decomposition scheme. The effect of these inference tasks is that they may produce binding of variables and assert/retract new literals into the current state. The form of these deductive inference rules is
`(:inline <precondition> <consequents>)`
where both precondition and consequent are logical expressions, what means that when `<precondition>` is true in

```
(:task travel-to
 :parameters (?destination)
 (:method Fly
  :precondition (flight ?destination)
  :tasks ((go-to-an-airport)
          (:inline (and (current-position ?airport)
                        (has-wifi-hotspot ?airport))
                   (enabled-wifi))
          (take-a-flight-to ?destination)))  ... )
```

Figure 3: Including deductive rules in the expansion of a high level task

the current state, the effects of the `<consequent>` are applied to the current state, allowing to assert or retract new literals. Let us consider the task description shown in Figure 3. We may see that the deductive inference rule allows the planner to include a new literal in the state (enabled-wifi) whenever the departure airport has a wi-fi hot-spot.

This might be seen as a conditional effect of some previous action but there are some differences that have to be clarified. Firstly, the activities in the decomposition which precede the inference rule may not be primitive actions, therefore, it is not always possible to encode them as conditional effects since high level tasks are not allowed to have effects. And secondly, the problem of the context of firing has to be considered, that is, if we encode every possible deducible consequence of a primitive action as sequence of exhaustive conditional effects, then we might be leading to a very well known problem in planning, the ramification problem (McIlraith 2000), and to an overload of the deductive process and unifications with the current state. Therefore, the inclusion of an inference rule into the decomposition of a high level task provides the context in which this inference is necessary and therefore, it will only fire at that moment.

Although the use of deductive inference has also appeared in the literature (Wilkins 1988), the use of abductive inference is becoming more widely used in the form of axioms (Nau *et al.* 2003) or derived literals, in terms of PDDL 2.2 (Edelkamp & Hoffmann 2004). These are abductive rules which appear in the form of a Horn clause and that allow to satisfy a given condition when this condition is not present in the current state, but it might be inferred by a set of inference rules of the form
`(:derived <literal> <logical_expression>)`
meaning that `<literal>` is true in the current state when the literals in `<logical_expression>` are also true in the current state. Both deductive and abductive inference rules are extensively used to incorporate additional knowledge to the planning domain while maintaining actions and tasks representations as simple as possible. This is particularly true in real-world planning problems like forest fire fighting plans design, the main application of SIADEX so far. In these cases, planning knowledge usually comes from very different sources, not always related to causality and therefore not very appropriate to encode in the causal structure of actions preconditions and effects. Another of the main purposes of these rules is to allow a process of temporal landmarking over the temporal representation of SIADEX that enrich the set of temporal constraints that may be encoded in a planning problem.

Temporal enhancements of SIADEX

One of the main drawbacks of state-based forward planners (HTN and non-HTN) is that they usually return plans as a total order sequence of activities, that is, a chain of actions.

$$\Pi = \{a_1, a_2, \ldots, a_n\}$$

If the planner is based on states, then, this sequence of actions also induces a sequence of states.

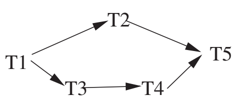
$$INITIAL + s_1 + s_2 + s_3 + \ldots + s_n$$

where s_i is the state that results from the execution of action a_i over the state s_{i-1}. However, in many real world applications, several activities may be carried out in parallel and a total order of activities is not very appropriate for practical reasons. In order to obtain plans with parallel branches, SIADEX uses several techniques: the definition of qualitative partial order relationships in the domain, the inference of new metric temporal constraints from the causal structure of the plan, the definition of deadline goals and complex synchronization schemas. These techniques are based on the inference capabilities explained before and the handling of metric time over a STN. A STN (Dechter, Meiri, & Pearl 1991) is a structure (X, D, C) such that X is the set of temporal points, D is the domain of every variable and C is the set of all the temporal constraints posted. In our case, a plan is deployed over a STN following a simple schema, every action $a_i \in \Pi$ owns two time points $start(a_i)$ and $end(a_i)$. Besides actions, every task t_i that had been expanded in the plan Π generates two time points $start(t_i)$ and $end(t_i)$ which bound the time points of its subactivities. There is an additional time point TR which represents the absolute temporal reference at the beginning of a plan. All the time points share the same domain $[0, \infty)$ and the constraints in C are posted and propagated at planning time in a two-steps refinement process. First, once an abstract task is decomposed into subtasks, low detail qualitative temporal constraints are introduced in the plan to arrange subtasks according to their order relation expressed in the domain. This could be seen as the plan's temporal skeleton. Later, once their sub-actions are being included definitely in the plan, more precise temporal constraints are added to encode the causal relationships between every final action in the plan.

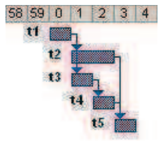
First step: introducing qualitative orderings

One of the main sources of temporal constraints between the actions of the plan comes from the order in which the subtasks of a given task appear in its decomposition. In this way, SIADEX' domains may impose three different types of qualitative ordering constraints in every decomposition.

Sequences They appear between parentheses `(T1,T2)`, and they are a set of subtasks that must execute in the same order than the decomposition, that is, first `T1` and then `T2`. Please note that any possible subtask of `T1` and `T2` inherit these relations too, so that the imposed ordering is maintained even through their decompositions.



(a)



(b)

Figure 4: A graphical representation of the partial order decomposition given by `(T1 [T2 (T3 T4)] T5)` (left hand side) and the plan representation found by SIADEX (right hand side)

Unordered They appear between braces `[T1,T2]`, and they are a set of subtasks that are not ordered in their decomposition, that is, either `T1` or `T2` could execute first.

Permutations They appear between angles `<T1,T2>`, and they are set of subtasks that must execute in any of the total orders given by any of their permutations, that is, first `T1` and then `T2` or vice versa. In these cases, the choice of the best permutation takes part in the search process of the planner. Please note that any possible subtask of `T1` and `T2` also inherit these relations.

For example, a method that decompose the task `t` into `(T1 [T2 (T3 T4)] T5)` represents a partially ordered decomposition depicted in Figure 4.a) and the plan obtained by SIADEX is shown in Figure 4.b).

These qualitative temporal constraints are posted as $[0, \infty)$ between the start and end points of the respective actions in the STN.

Second step: introducing quantitative orderings from causal links

The qualitative orderings seen before allow to encode relatively simple order relations that are the main sources of partial ordering in the final plan. But there is an additional source of ordering information that is very useful for encoding more accurate metric temporal constraints: the causal structure of actions. This section explains how to enhance the representation of states with temporally annotated literals, and how to exploit the existence of causal links between primitive activities to encode sharply defined metric temporal constraints between them, providing a more precise, but complementary, source of temporal constraints than these qualitative orderings.

All the literals l_j^k in the effects of every action a_j have a delay Δt_j^k by which they are achieved after the execution of their corresponding action. This delay Δt_j^k may range from 0 (the effect is achieved at the beginning of a_j) or the duration of a_j (it is achieved at the end). This is a generalization of PDDL 2.2 `at-start` and `at-end` effects

```
(:durative action a
...
:effects (at Δt_j^k (literal))
```

We may have "at start" effects, leaving $\Delta t_j^k = 0$, or "at end" effects, in this case $\Delta t_j^k = duration(a_j)$ or any other intermediate value.

SIADEX uses the information about delayed effects to represent states as a temporally annotated extension of classical states, where every literal is timestamped with the time by which it is achieved with respect to the action that produced it. Therefore, given a total order sequence of actions and its induced sequence of states

$$\Pi = a_1, a_2, \ldots, a_n \rightarrow INITIAL + s_1 + s_2 + s_3 + \ldots + s_n$$

every state s_i is given by $s_i = \{< l_j^k, a_{j,j \leq i}, \Delta t_j^k >\}$, that is, a set of literals l_j^k coming either from the effects of some previous action a_j ($l_j^k \in effects(a_j)$) or from the initial state (in this case the literal would have the form $< l_0^k, 0, 0 >$). This means that literal l_j^k was introduced in the state s_i Δt_j^k time units after the execution of a_j given by the time point $start(a_j)$. Then, a temporally annotated state allows to know which action, if any, produced that literal amongst the preceding actions and at which time they were achieved.

This information is particularly useful during the planning phase for two different reasons.

Firstly, they allow to propagate accurate temporal constraints between actions (Figure 2, step 1(b)iC). For every literal in the preconditions of an action a_i that matches a temporally annotated literal $< l_j^k, a_j, \Delta t_j^k > \in s_{i-1}$ we record a temporal causal link and use this information to post a temporal constraint, given by $[\Delta t_j^k, \infty)$, between the producing action, i.e. $start(a_j)$, and the consumer action, i.e. $start(a_i)$. This means that the consumer action $start(a_i)$ must wait at least Δt_j^k time units after $start(a_j)$, that is, the time needed for a_j to produce the desired effect l_j^k. Therefore, a temporal causal link allows to propagate temporal constrains due to the causal structure of the plan and taking into account the delays of the effects of supporting actions, and adds specific temporal constraints to the qualitative temporal constraints included formerly.

And secondly they allow to unfold the total order sequence of actions into a partially ordered plan where actions only depend of those other actions that produce any of the literals needed to satisfy their preconditions and are independent of the others. To ensure a correct causal structure of the plan within this unfolding operation, a simple protection mechanism, similar to threat removal operations in partial order planning (Weld 1994), adds some additional constraints. When an action a_i is included in the plan, and a_i deletes a literal l_j^k, an empty causal link is added such that orders a_i after any other action $a_k, k < i$ already present in the plan that depends on the same literal (has a non-empty causal link). Resulting temporal plans are not required to have a total order structure, in fact, they only have the minimum required temporal constraints to ensure a correct causal structure, which may be a total order or not depending on the causal structure of the plan.

Temporally extended goals

The process explained so far was mainly devoted to unfold the total order relation in which actions are obtained (see Figure 2) into a partial order plan that only records the causal structure of the plan, as a least commitment unfolding strategy (Figure 4.b). However, some real problems require the posting of more complex temporal constraints between the actions of a plan that not always have a cause-effect relationship. In this section, two additional enhancements are presented devoted, on one hand side, to deadline goals, that is, goals that must be achieved at a certain time, and in the other hand, complex synchronization schemas to allow actions to interact along the time.

Deadline goals

A deadline activity (either a task or an action) is an activity that may have defined one or more metric temporal constraints over its start or its end or both. Furthermore, in the case of tasks, SIADEX also allows to post deadline constraints on the start or the end of an activity (or both). Any sub-activity (either task or action) has two special variables associated to it: `?start` and `?end` that represent its start and end time points, and some constraints (basically `<=, =, >=`) may be posted to them. In order to do that, when any activity appears in the decomposition of a higher level task, it may be preceded by a logical expression that defines the desired deadline, either simple or compound, as it is shown in Figure 5. It shows that task `A` has no deadlines in its decomposition but task `A'` has a deadline that states that its subaction `A2` must start after 3 time units and end before 5 time units from the beginning of the plan. Deadline goals are easily encoded in the STN of a temporal plan as absolute constraints with respect to the reference time point TR, the absolute start point of a STN. They may also appear in the top level goal and they are very useful for defining time windows of activity, that is, sets of activities that must be executed within a given temporal interval, or timed trajectories of goals, i.e., sequences of goals that may be achieved one after the other like a recipe.

Time points of subtasks of any task with deadlines are embraced by the time points of the task, this means that, in practice, subtasks inherit the deadlines. For example, Figure 6 shows how the deadlines defined over task `A2` in Figure 5 constraint the timepoints of its subactions `a21` and `a22`. In this example, this inheritance process also has an useful collateral effect. Since the makespan of task `A2` is restricted to be within $[3, 5]$ time units, this enforces the planner to look only for valid decompositions of `A2` that meet these constraints, or in other words, any possible decomposition of `A2` that takes longer than 2 time units would produce a inconsistency in the STN and, thus, would make the planner to backtrack. Therefore, posting deadlines either on top level tasks or on intermediate tasks, would lead to a simple optimization of the makespan either of the whole plan or of isolated branches of it.

```
(:task A
 :parameters ()
 (:method A
  :precondition ()
  :tasks ((A1)
          (A2))))

(:task A'
 :parameters ()
 (:method A'
  :precondition ()
  :tasks ((A1)
          ((and (>= ?start 3)(<= ?end 5)) (A2)))))
(:task A2
 :parameters ()
 (:method A2
  :precondition ()
  :tasks ((a21)
          (a22))))
```

Figure 5: Task `A` has no deadlines in its decomposition, just that `A2` must appear after `A1`. Task `A'` has deadlines in its decomposition stating that subtask `A2` must appear after `A1` but `A2` must start after 3 time units and end before 5 time units since the beginning of the plan.

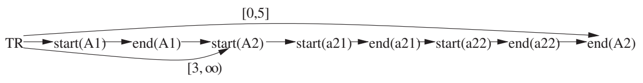


Figure 6: A task with deadlines imposes its constraints to its subtasks.

Temporal landmarking and complex synchronizations

SIADEX is also able to record the start and end of any activity and to recover these records in order to define complex synchronizations schemas between either tasks or actions as relative deadlines with respect to other activities. The first step is the definition, by assertion, of the temporal landmarks that signal the start and the end of either a task (Figure 7.a) or an action (Figure 7.b). These landmarks are treated as PDDL fluents but they are associated to the time points of the temporal constraints network and, therefore, fully operational for posting constraints between them in the underlying STN.

These landmarks are asserted in the current state, and later on, they may be recovered and posted as deadlines to other tasks in order to synchronize two or more activities. For example, Figure 8 shows the constraints needed to specify that action `b` must start exactly at the same time point than task `A2`.

In particular, thanks to the expressive power of temporal constraints networks and to the mechanism explained so far, a planning domain designer may explicitly encode in a problem's domain all of the different orderings included in Allen's algebra (Allen 1983) between two or more tasks, between two or more actions or between tasks and actions. Table 1 shows how these relations may be encoded between task `A2`, composed of actions `a21` and action `a22` with a

```
(:task A2
 :parameters ()
 (:method A2
  :precondition ()
  :tasks ((:inline ()
            (and (assign (start A2) ?start)
            (assign (end A2) ?end)))
        (a21)
        (a22))))
```

(a)

```
(:durative-action b
 :parameters()
 :duration (= ?duration 1)
 :condition()
 :effect(and (assign (start b) ?start)
                    (assign (end b) ?end)))
```

(b)

Figure 7: Generating temporal landmarks both for a task (a) and for an action (b). These landmarks are treated as fluents but they really represent time points of the underlying Simple Temporal Network underlying the plan, enabling the posting of additional constraints over them.

```
(:task A3
 :parameters ()
 (:method A3
  :precondition (...)
  :tasks (((= ?start (start A2)) (b)))))
```

Figure 8: Recovering a temporal landmark in order to define a synchronization scheme in which action `b` starts exactly at the same time that task `A2`

duration of 1 time unit each, and action `b` with a duration of 5 time units. These constraints posted on start and end points of tasks are also inherited by its corresponding subtasks.

Timed initial literals

Timed initial literals, as defined in PDDL 2.2 (Edelkamp & Hoffmann 2004) are also easily supported by SIADEX to represent timed exogenous events, that is, events that are produced (and possibly repeated) along the timeline outside of the control of the planner. In fact, SIADEX uses a generalization of timed initial literals to allow them to appear regularly along the timeline. Timed initial literals in SIADEX appear either like `(between <time1> and <time2> <literal>)` to represent that `<literal>` is true from time point `<time1>` to time point `<time2>`, or

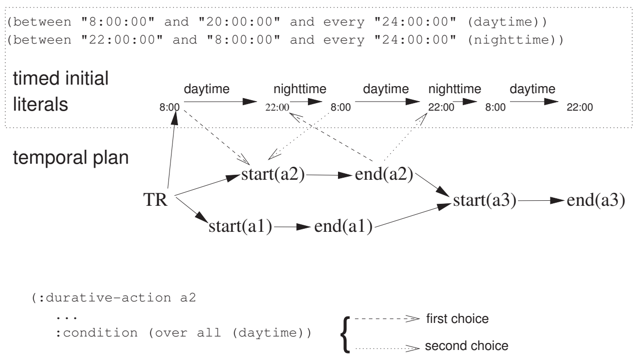
```
(between <time1> and <time2>
and every <shift> <literal>)
```

to represent that `<literal>` appear regularly at intervals given by `<shift>` along an infinite timeline. For example

```
(between "8:00:00" and "20:00:00"
 and every "24:00:00" (daytime))
(between "22:00:00" and "8:00:00"
and every "24:00:00" (nighttime))
```

represents that the literal `(daytime)` is true between a time point fixed at "8:00" and the time point fixed at "20:00" and that this is repeated every 24 hours. The literal `(nighttime)` behaves similarly.

These intervals in which a literal is true are represented as a temporal skeleton underlying the temporal plan as fixed time points with an absolute temporal reference to TR, the time point that fixes the absolute beginning of the STN. Since they may appear several times along the timeline, they also represent a choice point and, therefore, a backtracking point during the satisfaction of the preconditions of an action (either "at-start" or "over-all") (Figure 9).



```
(between "8:00:00" and "20:00:00" and every "24:00:00" (daytime))
(between "22:00:00" and "8:00:00" and every "24:00:00" (nighttime))
```

```
(:durative-action a2
   ...
   :condition (over all (daytime))
```

Figure 9: Timed initial literals deployed over a STN and its relation with a temporal plan. Precondition `(daytime)` of action `a2` may have multiple satisfiers amongst the timed initial literals.

Boosting constraint propagation

The constraint propagation engine used in SIADEX is the algorithm PC-2 (Dechter 2003) that is sketched in Figure 10. This is an incremental propagation method, very useful in planning problems, where constraints are posted increasingly as the problem is being solved. Propagation is needed in SIADEX for two reasons. The main one is to check the consistency of the underlying STN and the other one is to schedule actions along the timeline. Although it is a very efficient algorithm (it is $O(|X|^3)$ where X is the set of time points of the underlying STN) it still requires a high computational effort in large plans in the order of hundreds or thousands of actions. This overload comes from step 4c in Figure 10, where, once a constraint has been modified, all the time points and their connections with the just modified constraint are revised. The fact is that many of these revisions could not be necessary if the involved time points are causally independent and, therefore, their relation would not affect the final solution.

It is well known that the use of additional knowledge may speed up constraint propagation with structural information able to prune unnecessary propagation effort (Yorke-Smith 2005). Furthermore, cause-effect relationships between its actions or the plan rationale (Wilkins 1988), is a source of structural information that might be used to improve the efficiency of the underlying search processes in planning problems (Helmert 2004). Therefore, we propose a simple, but effective, improvement of PC-2 to reduce unnecessary con-

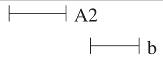
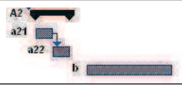
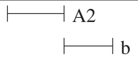
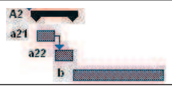
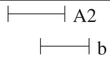
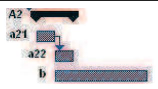
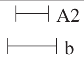
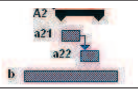
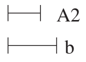
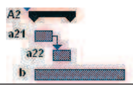
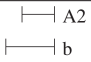
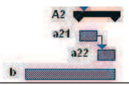
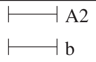
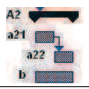
Allen's relation	SIADEX encoding	SIADEX output
 **A2 BEFORE b**	`((> ?start (end A2)) b)`	
 **A2 MEETS b**	`((= ?start (end A2)) b)`	
 **A2 OVERLAPS b**	`(and (> ?start (start A2))` `(< ?start (end A2))` `(> ?end (end A2)) b)`	
 **A2 DURING b**	`(and (< ?start (start A2))` `(> ?end (end A2)) b)`	
 **A2 STARTS b**	`((= ?start (start A2)) b)`	
 **A2 FINISHES b**	`((= ?end (end A2)) b)`	
 **A2 EQUAL b**	`(and (= ?start (start A2))` `(= ?end (end A2)) b)`	

Table 1: Encoding all Allen's relations between task `A2` and action `b`. In all the cases action `b` has a duration of 5 time units except in the `equal` relation, whose duration is 2 time units.

```
• Propagate-Temporal-Constraints-PC2(Π)
  1. Let $\Pi = \{a_1, a_2, \dots, a_n\}$ be a temporal plan and $R = (X, D, C)$ a Simple Temporal Network on top of which $\Pi$ is built, such that $X$ is the set of temporal points (given by start and end points of all of the actions in $\Pi$), $D$ is the domain of every variable, that is, $[0, +\infty)$, and $C$ is the set of all the temporal constraints posted in $\Pi$
  2. Let $(i, j)$ the time points affected by the last posted constraint in $C$.
  3. Let $Q \leftarrow \{(i, k, j), 1 \leq i < j \leq |X|, 1 \leq k \leq |X|, k \neq i, k \neq j\}$
  4. while $Q \neq \emptyset$
  (a) Select and delete a tuple (i,j,k) from $Q$
  (b) $C_{ij} = Revise(i, j, k)$
  (c) if $C_{ij}$ has changed after the call to $Revise(.)$ then foreach $l, 1 \leq l \leq |X|, l \neq i, l \neq j$
      i. $Q \leftarrow Q \cup \{(l, i, j)(l, j, i),\}$
• Revise(i,j,k)
  1. $C_{ij} = C_{ij} \cap (C_{ik} \circ C_{kj})$
```

Figure 10: The incremental algorithm PC2 for constraint propagation in STNs

	P1	P2	P3	P4	P5	P6	P7	P8
Sequential	8	11	12	14	19	22	27	32
Parallel	40	55	60	70	95	110	135	160

Table 2: Experiments "Sequential" and "Parallel" and the sizes of plans, that is, the number of actions obtained for every problem instance out of 8 different problems.

straints propagations by exploiting the causal structure of the plan and to boost its performance.

Algorithm PC-2 propagates constraints between all of the actions in a plan, causally dependent or not. This produces a large amount of information that is mostly unnecessary regarding the final solution of a problem. In a general constraint satisfaction framework, this process is enough to ensure a correct propagation of constraints and to detect inconsistencies. However, taking into account that it is being used in a planning framework, the completeness of the algorithm only depends on a correct constraint propagation through actions that have an explicit temporal relation, that is, they have a cause-effect relationship or that have a deadline or a landmark between them. Thus, a new propagation schema is proposed, named PC2-CL (Figure 11), that extends PC2 with an analysis of the causal structure of the plan to eliminate unnecessary propagations. In order to do that, step 4(c)i only includes in the propagation queue Q those time points belonging to an action that have an explicit temporal constraint either because there is a causal link with the action involved in the new constraint C_{ij} or because there is a deadline or a landmark between them.

```
• Propagate-Temporal-Constraints-PC2-CL($\Pi$)
  1. Let $\Pi$ be a temporal plan and $R = (X, D, C)$ its Simple Temporal Network
  2. Let $(i, j)$ the time points affected by the last posted constraint in $C$.
  3. Let $Q \leftarrow \{(i, k, j), 1 \leq i < j \leq |X|, 1 \leq k \leq |X|, k \neq i, k \neq j\}$
  4. while $Q \neq \emptyset$
   (a) Select and delete a tuple (i,j,k) from $Q$
   (b) $C_{ij} = Revise(i, j, k)$
   (c) if $C_{ij}$ has changed after the call to $Revise(.)$ then
       foreach $l, 1 \leq l \leq |X|, l \neq i, l \neq j$
       i. if $l$ belongs to an action with a causal link towards
          the action that owns either $i$ or $j$ or $l$ is a time point
          with a deadline or landmark associated to $i$ or $j$ then
          $Q \leftarrow Q \cup \{(l, i, j)(l, j, i),\}$
```

Figure 11: The modified version of PC2 that accounts for the existence of causal links

This extension obviously implies less propagations and a lower number of calls to the procedure $Revise(.)$ and, given that the representation of a list of temporal causal links is very simple, it is not expected to produce a computational overload. What is clear is that the whole improvement depends largely on the density of causal links, i.e., the greater the number of causal links the lower the gain. In addition to this, PC2-CL is provably correct since making use of the temporal causal links, it is guaranteed that constraints are propagated only to those actions that are affected by a cause-effect relationship or have an explicit constraint and not to unnecessary independent actions.

Some experiments

In order to empirically check the efficiency of SIADEX with PC2-CL four different experiments are performed. Two of them, named "Sequential" and "Parallel" are extracted from a domain of electronic tourism in which a planner is used to find plans of visit adapted to the preferences of a certain tourist (Fernández, Sebastiá, & Fdez-Olivares 2004). The third one, named "Infoca", is extracted from (Fdez-Olivares *et al.* 2006) and it consists of the application of a planner to the design of real forest fire fighting plans in Andalusia (Spain). The last experiment, named "Zeno", is framed in the Zeno domain of the 2002 international Planning Competition (Fox & Long 2002) and is performed over the hard instances of temporal and numeric problems. The size of the plans obtained for every experiments are shown in Tables 2, 3 and 4.

The experimental results of Figure 12 show much less propagation effort in PC2-CL measured as the number of calls to the $Revise(.)$ procedure. And, additionally, in terms of CPU time (Figure 13), PC2-CL is much faster in all the experiments except some instances of the "Sequential" domain in which, since the time is very low and plans are very small, the time devoted to check causal links might exceed

P1	P2	P3	P4	P5	P6	P7	P8
55	92	98	106	114	122	130	167
P9	P10	P11	P12	P13	P14	P15	P16
200	233	241	248	268	306	339	372

Table 3: Experiment "Infoca" and the sizes of plans, that is, the number of actions obtained for every problem instance out of 16 different problems.

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
73	107	156	171	272	296	297	318	373	245
P11	P12	P13	P14	P15	P16	P17	P18	P19	P20
237	240	262	255	298	290	287	304	277	284

Table 4: Experiment "Zeno" and the sizes of plans, that is, the number of actions obtained for every problem instance out of 20 different problems extracted from the hard instances of temporal and numeric problems in IPC 2002 (Fox & Long 2002).

the benefit of using PC2-CL, achieving thus a worse performance. However, in the remaining cases, the real example of "Infoca" and the hard temporal problems of "Zeno" PC2-CL outperforms PC2.

And just to give a relative idea of the overall performance of SIADEX, Figure 14 shows a comparison of the CPU time needed by SIADEX and SHOP2 (Nau *et al.* 2003), the best known HTN planner to date[1] to solve all the Zeno problems.

Related work

The use of STNs in planning frameworks is not new and has also been widely addressed in the literature. In a hierarchical planning setting, one of the earlier works is OPlan (Tate, Drabble, & Kirby 1994) in which many of the temporal constraints have to be explicitly encoded in the STN. SIADEX also encodes these constraints but only those that depend on high level goals (like deadlines or landmarks) are explicit and the remaining ones are implicitly encoded in the causal structure of the plan without an additional effort of the domain modeler to make them explicit. SHOP (Nau *et al.* 2003) does not use STNs but temporal constraints have also to be explicitly encoded amongst the effects and preconditions of operators in the Multi-Timeline Preprocessing scheme (MTP), requiring an important effort to write temporal domains. However, the use of causal links, either empty or not, to propagate constraints between actions and protect the achievements of literals in the current state seems to be fully equivalent to MTP. The main difference is that SHOP with MTP produces a unique schedule that coincides with the schedule of SIADEX' STN with the earliest execution time for every action, but SIADEX is able to handle

[1]Both planners running on the same machine, a Pentium IV 3 GHz, 1GB Ram. SIADEX is compiled from C++ and SHOP2 is written in Lisp running on Allegro CL. SIADEX was running a translation of the same domain used by SHOP2 in the IPC 2002 (Fox & Long 2002) with an equivalent HTN expressive power and the same constraints.

deadlines and landmarks and to obtain different schedules according to other criteria (i.e., latest execution time). Ixtet (Laborie & Ghallab 1995) also uses STNs with an equivalent expressive power and temporal cause effect relationships are encoded by means of two simple predicates named *events* (that represent an instantaneous change of the world) and *assertions* (that represent the persistence of some attributes along the timeline).

On the other hand, the use of different sources of knowledge to restrict temporal constraint propagation to a subnetwork of the former STN in order to obtain a better performance has also been studied like in Ixtet (Ghallab & Vidal 1995) or in PASSAT (Yorke-Smith 2005). These works and the one presented in this paper may have different performance results on the same tests and a comparative study might be useful, but it is worth noting that the approach in (Yorke-Smith 2005), in which constraints are propagated taking into account the HTN structure of the plan, is particularly relevant since it is an orthogonal approach to this one and it seems that they might be combined to obtain even more efficient results.

Conclusions

In summary, this paper has presented several valuable temporal extensions of an HTN planner that allow to cope with a very rich temporal knowledge representation like temporal causal dependencies, deadlines, temporal landmarks or synchronization schemas and timed initial literals. These capabilities have been found to be of extreme necessity during the application of SIADEX to the research contract NET033957 with the Andalusian Regional Ministry of Environment for the assisted design of forest fighting plans (de la Asunción *et al.* 2005; Fdez-Olivares *et al.* 2006) and, up to author's knowledge, no other HTN planner has these capabilities for handling temporal constraints. In addition to this temporal expressive power, SIADEX also shows an excellent performance compared to other well known HTN planner like SHOP.

References

Ai-Chang, M.; Bresina, J.; Charest, L.; Chase, A.; jung Hsu, J. C.; Jonsson, A.; Kanefsky, B.; Morris, P.; Rajan, K.; Yglesias, J.; Chafin, B. G.; Dias, W. C.; and Maldague, P. F. 2004. Mapgen: Mixed-initiative planning and scheduling for the mars exploration rover mission. *IEEE Intelligent Systems* 8–12.

Allen, J. 1983. Maintaining knowledge about temporal intervals. *Comm. ACM* 26(1):832–843.

de la Asunción, M.; Castillo, L.; Fdez-Olivares, J.; García-Pérez, O.; González, A.; and Palao, F. 2005. Siadex: an interactive artificial intelligence planner for decision support and training in forest fire fighting. *Artificial Intelligence Communications* 18(4).

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49:61–95.

Dechter, R. 2003. *Constraint processing*. Morgan Kaufmann.

Edelkamp, S., and Hoffmann, J. 2004. The language for the 2004 international planning competition. http://ls5-www.cs.uni-dortmund.de/ edelkamp/ipc-4/pddl.html.

Fdez-Olivares, J.; Castillo, L.; García-Pérez, O.; and Palao, F. 2006. Bringing users and planning technology together. experiences in siadex. In *Sixteenth International Conference on Automated Planning and Scheduling, ICAPS*.

Fernández, S.; Sebastiá, L.; and Fdez-Olivares, J. 2004. Planning tourist visits adapted to user preferences. In *Workshop on Planning and Scheduling: Bridging Theory to Practice, European Conference on Artificial Intelligence*.

Fox, M., and Long, D. 2002. Domains of the 3rd. international planning competition. In *Artificial Intelligence Planning Systems (AIPS 02)*.

Ghallab, M., and Vidal, T. 1995. Focusing on the subgraph for managing efficiently numerical temporal constraints. In *FLAIRS-95*.

Gil, Y., and Blythe, J. 2000. PLANET: A shareable and reusable ontology for representing plans. In *AAAI 2000 workshop on representational issues for real-world planning systems*.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In *International Conference on Automated Planning and Scheduling, ICAPS*.

Laborie, P., and Ghallab, M. 1995. Planning with sharable resource constraints. In *IJCAI'95*, 1643–1649.

McIlraith, S. A. 2000. Integrating actions and state constraints: A closed-form solution to the ramification problem (sometimes). *Artificial Intelligence Journal* 87–121.

Myers, K. L.; Tyson, W. M.; Wolverton, M. J.; Jarvis, P. A.; Lee, T. J.; and desJardins, M. 2002. PASSAT: A user-centric planning framework. In *Proc. of the Third Intl. NASA Workshop on Planning and Scheduling for Space*.

Nau, D.; Au, T.; Ilghami, O.; Kuter, U.; Murdock, J. W.; Wu, D.; and Yaman, F. 2003. SHOP2: An HTN Planning System. *Journal of Artificial Intelligence Research* 20:379–404.

Oddi, A.; Cesta, A.; Policella, N.; and Cortellessa, G. 2002. Scheduling downlink operations in mars express. In *Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space*.

Tate, A.; Drabble, B.; and Kirby, R. 1994. O-PLAN2: An open architecture for command, planning and control. In Zweben, M., and Fox, M., eds., *Intelligent scheduling*. Morgan Kaufmann.

Weld, D. 1994. An introduction to least commitment planning. *AI Magazine* 15(4):27–61.

Wilkins, D. E., and desJardins, M. 2001. A call for knowledge-based planning. *AI Magazine* 22(1):99–115.

Wilkins, D. E. 1988. *Practical planning: Extending the classical AI planning paradigm*. Morgan Kaufmann.

Yorke-Smith, N. 2005. Exploiting the structure of hierarchical plans in temporal constraint propagation. In *Proceedings of AAAI'05*.

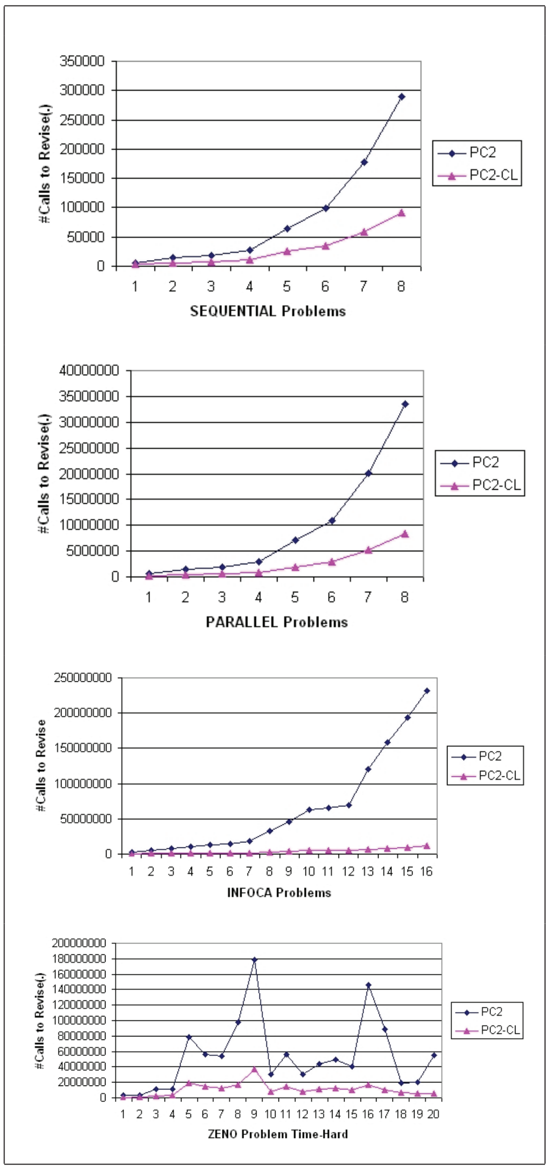


Figure 12: Number of calls to the procedure $Revise(.)$ in PC2 versus PC2-CL for all the instances of every problem.

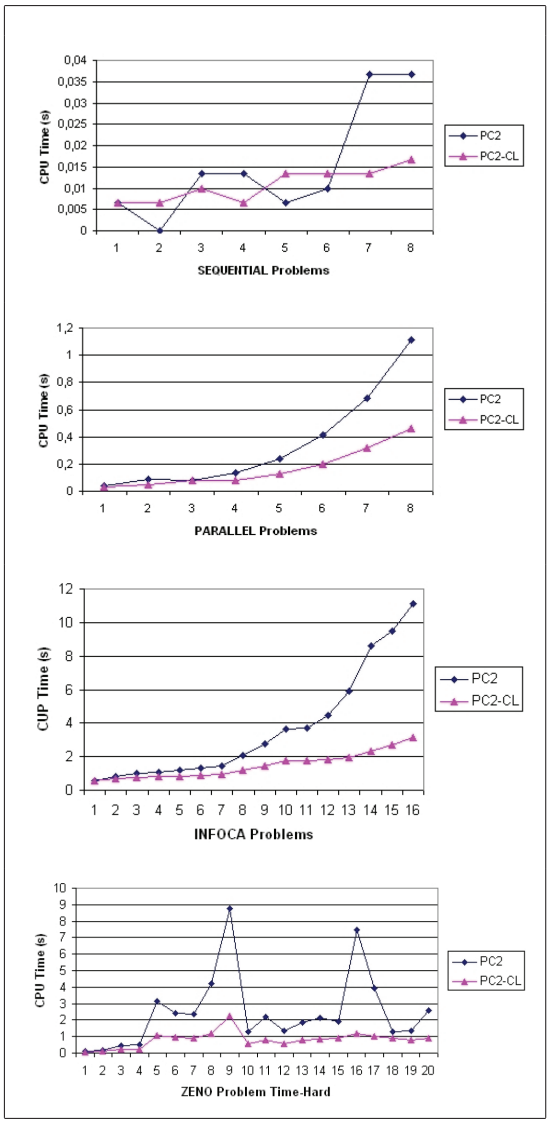


Figure 13: Compared CPU time in PC2 and PC2-CL in all the instances of the problems.

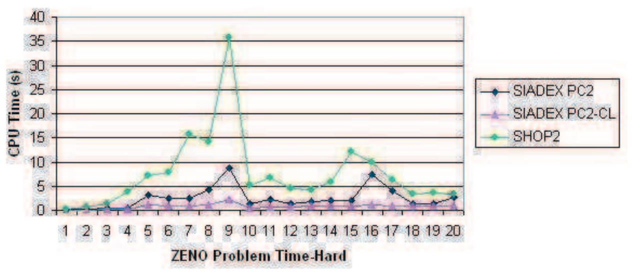


Figure 14: Compared CPU time between SIADEX and SHOP2 in the hard instances of ZENO (temporal+numeric).

SIADEX. An integrated planning framework for crisis action planning[*]

L. Castillo, J. Fdez-Olivares, O. García-Pérez and F. Palao
Dept. of Computers Science and Artificial Intelligence
ETSI Informática, University of Granada
18071, Granada, SPAIN
{L.Castillo,Faro,Oscar,Palao}@decsai.ugr.es
Phone:+34.958.240803, +34.958240805

Abstract

SIADEX is an integrated framework to support decision making during crisis episodes by providing realistic temporally annotated plans of action. The main component of SIADEX is a forward state-based HTN temporal planner.

Introduction

The design of plans of activity for crisis situations is a very sensitive field of application for mature AI planning and scheduling techniques (Allen *et al.* 1995; Myers 1999; Biundo & Schattenberg 2001; Avesani, Perini, & Ricci 2000). However, a successful approach requires a subtle integration of several research and development issues like

- Integration of several technologies. These systems are not usually a monolithic approach, but a composition of technologies that integrate with each other with different functionalities like planning (to determine the appropriate set of activities), scheduling (to handle time and resources), pathfinding (to find optimal movement plans in complex networks), etc.
- Enhancing the role of end-users. End users of these systems are not expected to have a background knowledge on AI, therefore the system must use user-friendly interfaces in order for end-users to establish goals, to understand what the system does and what the system is demanding without having to use a technical language like a planning domain description language or a constraint programming language.
- Flexible knowledge representation. The system must represent a large amount of data coming from heterogenous sources of information like GPS locations of resources, facilities, legal issues that constraint the activities (for example, contracting conditions or durations of shifts), exogenous events (i.e., meteo conditions, day and night periods), and many more. Even more, although this might seem simple it is a very important issue, the system must access to all this information on-line, that is, extracting it from legacy databases and translating them into known planning and scheduling domain description languages.
- Support of distributed and concurrent access of end-users. Usually, these systems are operated in hostile environments like a forest fire, natural disasters scenarios, etc, and most of the inputs come from (and most of the outputs are directed to) end-users located at these places. These systems are too complex as to be installed and run on small devices with limited computation capability like a laptop or a PDA, therefore providing a centralized high-capability computing facility with full connectivity and accessibility to end users is a valuable feature.
- Integration with legacy software. Not only the income but also the outcome must be redirected to legacy software so that end users may painlessly understand, process and deliver activity plans. In this case a user-friendly output to GIS or project management and monitoring is strongly required.
- Quick response. Last, but not least, the system must be very efficient so that it obtains a response in an acceptable time with respect to the own latency time of the crisis situation (that may range from minutes to hours).

The architecture of SIADEX

SIADEX is an open problem solving architecture based on the intensive use of web services to implement most of its capabilities (Figure 1). Its main components are the knowledge base (named BACAREX), that stores in Protege ((National Library of Medicine)) all the knowledge that would be useful for the planning engine, and the planning module (named SIADEX), the core of the architecture in charge of building fire fighting plans.

The planning algorithm and its knowledge representation are built as two independent modules, which are accessible from any device with internet connectivity (a desktop computer, a laptop or a PDA). This allow users to query or modify the state of the world by using almost any existing web browser or by using a well known GIS software (ESRI), recall that during a crisis episode most of the objects and resources are associated have geographical properties (see Figure 2). In the same way, temporal plans designed by SIADEX may be downloaded into some project management software in the form of Gantt charts (see Figure 3) or any other software commonly used by technical staff. The basic process is as follows.

[*]This work is being funded by the Andalusian Regional Ministry of the Environment, under research contract NET033957

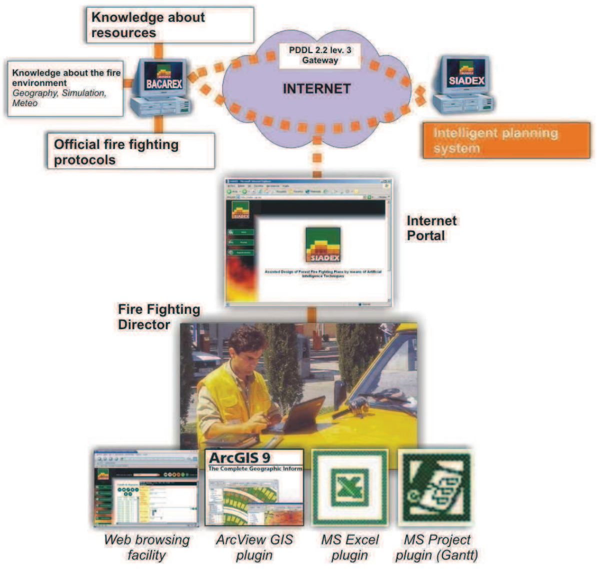

Knowledge about resources
Knowledge about the fire environment
Geography, Simulation, Meteo
BACAREX
PDDL 2.2 lev. 3 Gateway
INTERNET
SIADEX
Intelligent planning system
Official fire fighting protocols
Internet Portal
Fire Fighting Director
ArcGIS 9
Web browsing facility
ArcView GIS plugin
MS Excel plugin
MS Project plugin (Gantt)

Figure 1: Architecture of SIADEX

Describing the problem The fire fighting scenario is introduced by the technical staff consisting of the targets areas, the general attack procedures and an estimation of the number of resources to be used. This may be done by the web browsing utility or, much more easily, by the ArcView GIS software plugin (ESRI).

Storing the scenario The fire fighting scenario is stored in Protege format in BACAREX. Therefore, knowledge about resources and fire scenario share the same representation. All this information is visible to other users by means of the web browsing facility, although only the knowledge about the problem may be accessed (the knowledge about the domain, tasks and actions, is only visibile for the development team).

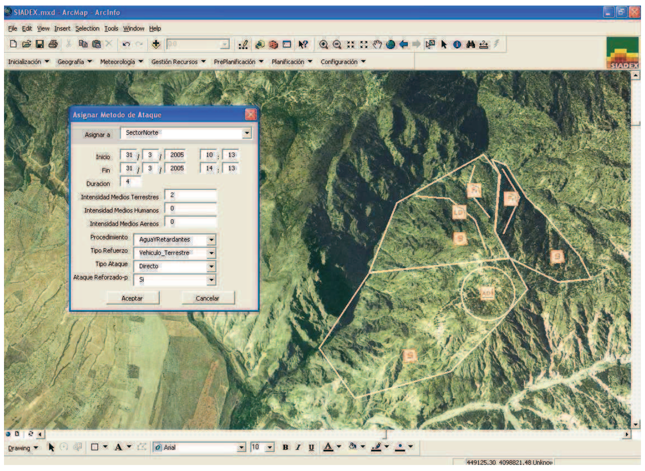
Requesting a plan The planning engine is not able to read the domain and the problem stored in Protege, therefore a PDDL Gateway has been implemented that translate problem and domain into PDDL 2.2 level 3 (Edelkamp & Hoffmann 2004). After that, the planning engine is called and a plan is obtained (or not).

Displaying the plan The plan obtained may be displayed in a number of "user-friendly" alternatives like Microsoft Excel, in the form of a chronogram, or Microsoft Project in the form of a Gantt chart (see Figure 3).

Plan execution and monitoring (in development) The plan may be launched for execution, distributed amognst all the technical staff with some resposibility in the fire fighting episode, and concurrently monitored.

Domain knowledge

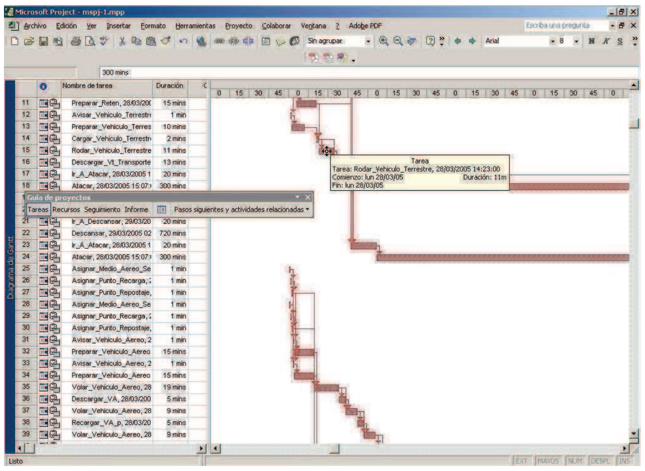
The knowledge about the planning objects (places, facilities, task forces, resources, etc) is stored in an ontology of the problem represented in Protégé, an ontology editor and


SIADEX.mxd - ArcMap - ArcInfo
File Edit View Insert Selection Tools Window Help
Inicialización Geografía Meteorología Gestión Recursos PrePlanificación Planificación Configuración
SIADEX
Aceptar
Cancelar

Asignar Metodo de Ataque

Asignar a: SectorNorte
Inicio: 31 / 3 / 2005 10 : 13
Fin: 31 / 3 / 2005 14 : 13
Duracion: 4
Intensidad Medios Terrestres: 2
Intensidad Medios Humanos: 0
Intensidad Medios Aereos: 0
Procedimiento: AguaYRetardantes
Tipo Refuerzo: Vehiculo_Terrestre
Tipo Ataque: Directo
Ataque Reforzado-p: Si

Figure 2: The ArcView plugin


Microsoft Project - mspj-1.mpp
Archivo Edición Ver Insertar Formato Herramientas Proyecto Colaborar Ventana ? Adobe PDF
Sin agrupar
Arial
300 mins
Guía de proyectos
Tareas Recursos Seguimiento Informe
Pasos siguientes y actividades relacionadas
Diagrama de Gantt
Listo

	Nombre de tarea	Duración
11	Preparar_Reten, 28/03/200	15 mins
12	Avisar_Vehiculo_Terrestre	1 min
13	Preparar_Vehiculo_Terres	10 mins
14	Cargar_Vehiculo_Terrestr	2 mins
15	Rodar_Vehiculo_Terrestre	11 mins
16	Descargar_Vt_Transporte	13 mins
17	Ir_A_Atacar, 28/03/2005 1	20 mins
18	Atacar, 28/03/2005 15:07:	300 mins
21	Ir_A_Descansar, 29/03/20	20 mins
22	Descansar, 29/03/2005 02	720 mins
23	Ir_A_Atacar, 28/03/2005 1	20 mins
24	Atacar, 28/03/2005 15:07:	300 mins
25	Asignar_Medio_Aereo_Se	1 min
26	Asignar_Punto_Recarga, 2	1 min
27	Asignar_Punto_Repostaje,	1 min
28	Asignar_Medio_Aereo_Se	1 min
29	Asignar_Punto_Recarga, 2	1 min
30	Asignar_Punto_Repostaje,	1 min
31	Avisar_Vehiculo_Aereo, 2	1 min
32	Preparar_Vehiculo_Aereo	15 mins
33	Avisar_Vehiculo_Aereo, 2	1 min
34	Preparar_Vehiculo_Aereo	15 mins
35	Volar_Vehiculo_Aereo, 28	19 mins
36	Descargar_VA, 28/03/200	5 mins
37	Volar_Vehiculo_Aereo, 28	9 mins
38	Recargar_VA_p, 28/03/20	5 mins
39	Volar_Vehiculo_Aereo, 28	9 mins

Tarea
Tarea: Rodar_Vehiculo_Terrestre, 28/03/2005 14:23:00
Comienzo: lun 28/03/05 Duración: 11m
Fin: lun 28/03/05

Figure 3: Gantt chart output

knowledge acquisition tool (National Library of Medicine). A web browsing tool has been designed so that end users may easily access to the hierarchy of objects, to query or modify their properties, without having to know anything related to knowledge representation[1]. This hierarchy of objects also supports the definition of the goal scenario (geographical targets, goal tasks, etc) either from a web browser or from a GIS software. Once a plan is requested by the user, the knowledge stored in this knowledge base is then translated into PDDL 2.2 level 3 (Edelkamp & Hoffmann 2004), with support for timed initial literals and derived predicates, following the next outline:

- Classes of the ontology are translated into a hierarchy of PDDL types.
- Instances are translated as typed planning objects (only the slots relevant for the planning process are translated).
- The domain is stored directly in the form of tasks, meth-

[1]The development team may also use the standard Protégé shell to run knowledge consistency checking and validation.

ods and actions compliant with PDDL 2.2 level 3, so it does not need to be translated.

- Other constraints of the problems are also translated accordingly like maximun legal duration of shifts (fluent), day/night events (timed initial literals), activity windows over the scenario (deadline goals), etc.

The planner

The planning module is a forward state-based HTN planning algorithm with the following features:

- Primitive actions are fully compliant with PDDL 2.2 with durative actions and numeric capabilities (Figure 4).
- It makes use of an extension of PDDL to represent timed HTN tasks and methods.
- SIADEX's domains also embed some functionalities to control and prune the search in order to make the planning process more efficient.
- SIADEX also supports the use of external functions calls by embedding Python scripts in the domain definition, to access external sources of information or perform complex computations during the planning process (Figure 5).

```
(:durative-action Refuel_Plane
 :parameters (?a - Aircraft ?p - Refueling_Point)
 :prettyprint "?start > Aircraft ?a starts refueling
               at ?p. Finishing at ?end"
 :duration (= ?duration (refueling_time ?a))
 :condition  (and
  (in_fire ?a ?fire)
  (over all (daylight ?fire))
  (GIS ?p ?gis)
  (current_position ?a ?gis))
 :effect (and
  (state ?a refuelling)
  (at end (assign (current_autonomy ?a)
          (max_autonomy ?a))))
)
```

Figure 4: PDDL 2.2 level 3 primitive actions

Temporal and resource reasoning

One of the most important features of SIADEX is that it allows a powerful handling of temporal knowledge. SIADEX's plans are built on top of a temporal constraint network (Dechter, Meiri, & Pearl 1991) that records temporal and causal dependencies between actions so that, although it is a state based process, plans may have a partial order structure with temporal references either qualitative or numeric. This allows SIADEX to obtain very flexible schedules (N. Policella 2004) that might be redesigned during the execution of the plan to adapt to unforeseen delays without the need to replan. In addition to this, SIADEX also supports the definition of constraints on the makespan of the plan and deadline goals over primitive and compound tasks (in the case of compound tasks, deadline goals are inherited by its component tasks).

```
 (:functions
 (distance ?x1 ?y1 ?x2 ?y2)
{
 import math
 return math.sqrt ( (?x2 - ?x1) *
    (?x2 -?x1) +
    (?y2 - ?y1) *
    (?y2 - ?y1))
}
...
```

Figure 5: Embedded python in the domain

Since SIADEX handles numerical objects like PDDL fluents (that can also be dynamically linked to external Python calls) it achieves a basic handling of numeric resources.

References

Allen, J.; Schubert, L.; Ferguson, G.; Heeman, P.; Hwang, C.; Kato, T.; Light, M.; Martin, N.; Miller, B.; Poesio, M.; and Traum, B. 1995. The TRAINS project: a case study in building a conversational planning agent. *Experimental and Theoretical Artificial Intelligence* 7:7–48.

Avesani, P.; Perini, A.; and Ricci, F. 2000. Interactive case-based planning for forest fire management. *Applied Intelligence* 13(1):41–57.

Biundo, S., and Schattenberg, B. 2001. From abstract crisis to concrete relief - a preliminary report on combining state abstraction and htn planning. In *6th European Conference on Planning (ECP-01)*.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49:61–95.

Edelkamp, S., and Hoffmann, J. 2004. The language for the 2004 international planning competition. http://ls5-www.cs.uni-dortmund.de/ edelkamp/ipc-4/pddl.html.

ESRI. http://www.esri.com.

Myers, K. L. 1999. CPEF: A continuous planning and execution framework. *AI Magazine* 20(4):63–69.

N. Policella, S. Smith, A. C. A. O. 2004. Generating robust schedules through temporal flexibility. In *14th International Conference on Automated Planning and Scheduling, ICAPS*.

National Library of Medicine. http://protege.stanford.edu/.

SIADEX. An integrated planning framework for crisis action planning[*]

L. Castillo, J. Fdez-Olivares, O. García-Pérez and F. Palao
Dept. of Computers Science and Artificial Intelligence
ETSI Informática, University of Granada
18071, Granada, SPAIN
{L.Castillo,Faro,Oscar,Palao}@decsai.ugr.es
Phone:+34.958.240803, +34.958240805

Abstract

SIADEX is an integrated framework to support decision making during crisis episodes by providing realistic temporally annotated plans of action. The main component of SIADEX is a forward state-based HTN temporal planner.

Introduction

The design of plans of activity for crisis situations is a very sensitive field of application for mature AI planning and scheduling techniques (Allen *et al.* 1995; Myers 1999; Biundo & Schattenberg 2001; Avesani, Perini, & Ricci 2000). However, a successful approach requires a subtle integration of several research and development issues like

- Integration of several technologies. These systems are not usually a monolithic approach, but a composition of technologies that integrate with each other with different functionalities like planning (to determine the appropriate set of activities), scheduling (to handle time and resources), pathfinding (to find optimal movement plans in complex networks), etc.
- Enhancing the role of end-users. End users of these systems are not expected to have a background knowledge on AI, therefore the system must use user-friendly interfaces in order for end-users to establish goals, to understand what the system does and what the system is demanding without having to use a technical language like a planning domain description language or a constraint programming language.
- Flexible knowledge representation. The system must represent a large amount of data coming from heterogenous sources of information like GPS locations of resources, facilities, legal issues that constraint the activities (for example, contracting conditions or durations of shifts), exogenous events (i.e., meteo conditions, day and night periods), and many more. Even more, although this might seem simple it is a very important issue, the system must access to all this information on-line, that is, extracting it from legacy databases and translating them into known planning and scheduling domain description languages.
- Support of distributed and concurrent access of end-users. Usually, these systems are operated in hostile environments like a forest fire, natural disasters scenarios, etc, and most of the inputs come from (and most of the outputs are directed to) end-users located at these places. These systems are too complex as to be installed and run on small devices with limited computation capability like a laptop or a PDA, therefore providing a centralized high-capability computing facility with full connectivity and accessibility to end users is a valuable feature.
- Integration with legacy software. Not only the income but also the outcome must be redirected to legacy software so that end users may painlessly understand, process and deliver activity plans. In this case a user-friendly output to GIS or project management and monitoring is strongly required.
- Quick response. Last, but not least, the system must be very efficient so that it obtains a response in an acceptable time with respect to the own latency time of the crisis situation (that may range from minutes to hours).

The architecture of SIADEX

SIADEX is an open problem solving architecture based on the intensive use of web services to implement most of its capabilities (Figure 1). Its main components are the knowledge base (named BACAREX), that stores in Protege ((National Library of Medicine)) all the knowledge that would be useful for the planning engine, and the planning module (named SIADEX), the core of the architecture in charge of building fire fighting plans.

The planning algorithm and its knowledge representation are built as two independent modules, which are accessible from any device with internet connectivity (a desktop computer, a laptop or a PDA). This allow users to query or modify the state of the world by using almost any existing web browser or by using a well known GIS software (ESRI), recall that during a crisis episode most of the objects and resources are associated have geographical properties (see Figure 2). In the same way, temporal plans designed by SIADEX may be downloaded into some project management software in the form of Gantt charts (see Figure 3) or any other software commonly used by technical staff. The basic process is as follows.

[*]This work is being funded by the Andalusian Regional Ministry of the Environment, under research contract NET033957

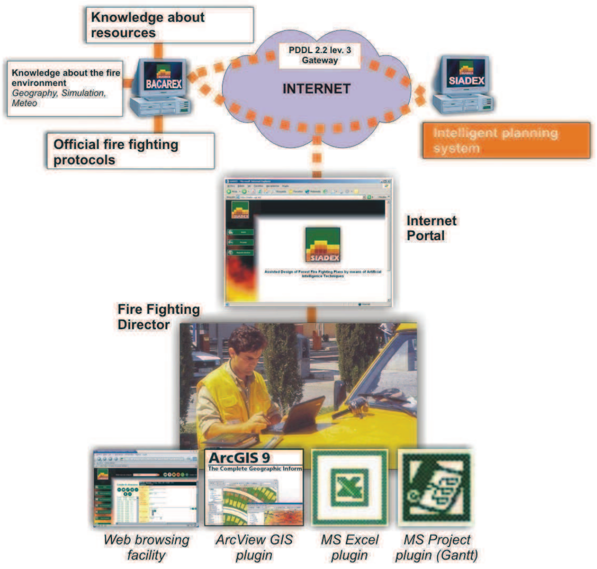

Knowledge about resources
Knowledge about the fire environment
Geography, Simulation, Meteo
BACAREX
PDDL 2.2 lev. 3 Gateway
INTERNET
SIADEX
Intelligent planning system
Official fire fighting protocols
Internet Portal
Fire Fighting Director
ArcGIS 9
Web browsing facility
ArcView GIS plugin
MS Excel plugin
MS Project plugin (Gantt)

Figure 1: Architecture of SIADEX

Describing the problem The fire fighting scenario is introduced by the technical staff consisting of the targets areas, the general attack procedures and an estimation of the number of resources to be used. This may be done by the web browsing utility or, much more easily, by the ArcView GIS software plugin (ESRI).

Storing the scenario The fire fighting scenario is stored in Protege format in BACAREX. Therefore, knowledge about resources and fire scenario share the same representation. All this information is visible to other users by means of the web browsing facility, although only the knowledge about the problem may be accessed (the knowledge about the domain, tasks and actions, is only visibile for the development team).

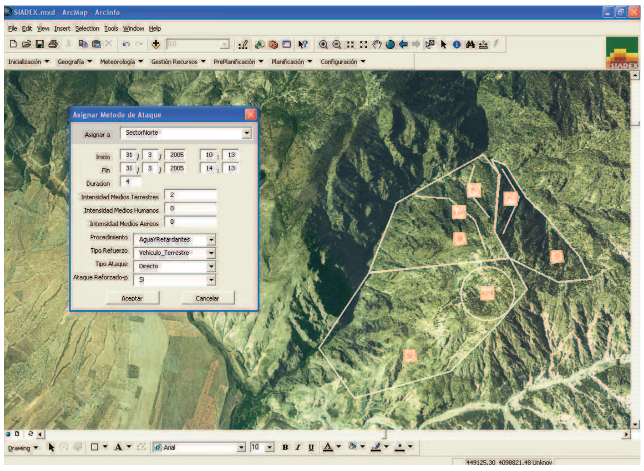
Requesting a plan The planning engine is not able to read the domain and the problem stored in Protege, therefore a PDDL Gateway has been implemented that translate problem and domain into PDDL 2.2 level 3 (Edelkamp & Hoffmann 2004). After that, the planning engine is called and a plan is obtained (or not).

Displaying the plan The plan obtained may be displayed in a number of "user-friendly" alternatives like Microsoft Excel, in the form of a chronogram, or Microsoft Project in the form of a Gantt chart (see Figure 3).

Plan execution and monitoring (in development) The plan may be launched for execution, distributed amognst all the technical staff with some resposibility in the fire fighting episode, and concurrently monitored.

Domain knowledge

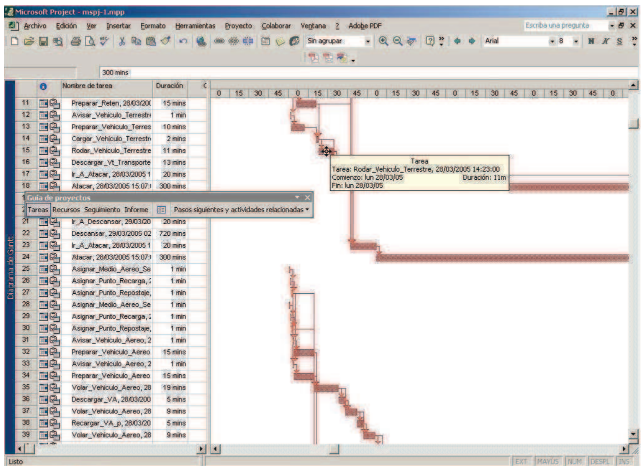
The knowledge about the planning objects (places, facilities, task forces, resources, etc) is stored in an ontology of the problem represented in Protégé, an ontology editor and


SIADEX.mxd - ArcMap - ArcInfo
File Edit View Insert Selection Tools Window Help
Inicialización Geografía Meteorología Gestión Recursos PrePlanificación Planificación Configuración
Aceptar
Cancelar

Asignar Metodo de Ataque

Asignar a	SectorNorte
Inicio	31 / 3 / 2005 10 : 13
Fin	31 / 3 / 2005 14 : 13
Duracion	4
Intensidad Medios Terrestres	2
Intensidad Medios Humanos	0
Intensidad Medios Aereos	0
Procedimiento	AguaYRetardantes
Tipo Refuerzo	Vehiculo_Terrestre
Tipo Ataque	Directo
Ataque Reforzado-p	Si

Figure 2: The ArcView plugin


Microsoft Project - mspj-1.mpp
Archivo Edición Ver Insertar Formato Herramientas Proyecto Colaborar Ventana ? Adobe PDF
Guía de proyectos
Tareas Recursos Seguimiento Informe Pasos siguientes y actividades relacionadas
Diagrama de Gantt
Listo

300 mins

	Nombre de tarea	Duración
11	Preparar_Reten, 28/03/20(	15 mins
12	Avisar_Vehiculo_Terrestre	1 min
13	Preparar_Vehiculo_Terres	10 mins
14	Cargar_Vehiculo_Terrestr	2 mins
15	Rodar_Vehiculo_Terrestre	11 mins
16	Descargar_Vt_Transporte	13 mins
17	Ir_A_Atacar, 28/03/2005 1	20 mins
18	Atacar, 28/03/2005 15:07:	300 mins
21	Ir_A_Descansar, 29/03/20	20 mins
22	Descansar, 29/03/2005 02	720 mins
23	Ir_A_Atacar, 28/03/2005 1	20 mins
24	Atacar, 28/03/2005 15:07:	300 mins
25	Asignar_Medio_Aereo_Se	1 min
26	Asignar_Punto_Recarga, 2	1 min
27	Asignar_Punto_Repostaje,	1 min
28	Asignar_Medio_Aereo_Se	1 min
29	Asignar_Punto_Recarga, 2	1 min
30	Asignar_Punto_Repostaje,	1 min
31	Avisar_Vehiculo_Aereo, 2	1 min
32	Preparar_Vehiculo_Aereo	15 mins
33	Avisar_Vehiculo_Aereo, 2	1 min
34	Preparar_Vehiculo_Aereo	15 mins
35	Volar_Vehiculo_Aereo, 28	19 mins
36	Descargar_VA, 28/03/200	5 mins
37	Volar_Vehiculo_Aereo, 28	9 mins
38	Recargar_VA_p, 28/03/20	5 mins
39	Volar_Vehiculo_Aereo, 28	9 mins

Tarea
Tarea: Rodar_Vehiculo_Terrestre, 28/03/2005 14:23:00
Comienzo: lun 28/03/05 Duración: 11m
Fin: lun 28/03/05

Figure 3: Gantt chart output

knowledge acquisition tool (National Library of Medicine). A web browsing tool has been designed so that end users may easily access to the hierarchy of objects, to query or modify their properties, without having to know anything related to knowledge representation[1]. This hierarchy of objects also supports the definition of the goal scenario (geographical targets, goal tasks, etc) either from a web browser or from a GIS software. Once a plan is requested by the user, the knowledge stored in this knowledge base is then translated into PDDL 2.2 level 3 (Edelkamp & Hoffmann 2004), with support for timed initial literals and derived predicates, following the next outline:

- Classes of the ontology are translated into a hierarchy of PDDL types.
- Instances are translated as typed planning objects (only the slots relevant for the planning process are translated).
- The domain is stored directly in the form of tasks, meth-

[1]The development team may also use the standard Protégé shell to run knowledge consistency checking and validation.

ods and actions compliant with PDDL 2.2 level 3, so it does not need to be translated.

- Other constraints of the problems are also translated accordingly like maximun legal duration of shifts (fluent), day/night events (timed initial literals), activity windows over the scenario (deadline goals), etc.

The planner

The planning module is a forward state-based HTN planning algorithm with the following features:

- Primitive actions are fully compliant with PDDL 2.2 with durative actions and numeric capabilities (Figure 4).
- It makes use of an extension of PDDL to represent timed HTN tasks and methods.
- SIADEX's domains also embed some functionalities to control and prune the search in order to make the planning process more efficient.
- SIADEX also supports the use of external functions calls by embedding Python scripts in the domain definition, to access external sources of information or perform complex computations during the planning process (Figure 5).

```
(:durative-action Refuel_Plane
 :parameters (?a - Aircraft ?p - Refueling_Point)
 :prettyprint "?start > Aircraft ?a starts refueling
               at ?p. Finishing at ?end"
 :duration (= ?duration (refueling_time ?a))
 :condition  (and
  (in_fire ?a ?fire)
  (over all (daylight ?fire))
  (GIS ?p ?gis)
  (current_position ?a ?gis))
 :effect (and
  (state ?a refuelling)
  (at end (assign (current_autonomy ?a)
          (max_autonomy ?a))))
)
```

Figure 4: PDDL 2.2 level 3 primitive actions

Temporal and resource reasoning

One of the most important features of SIADEX is that it allows a powerful handling of temporal knowledge. SIADEX's plans are built on top of a temporal constraint network (Dechter, Meiri, & Pearl 1991) that records temporal and causal dependencies between actions so that, although it is a state based process, plans may have a partial order structure with temporal references either qualitative or numeric. This allows SIADEX to obtain very flexible schedules (N. Policella 2004) that might be redesigned during the execution of the plan to adapt to unforeseen delays without the need to replan. In addition to this, SIADEX also supports the definition of constraints on the makespan of the plan and deadline goals over primitive and compound tasks (in the case of compound tasks, deadline goals are inherited by its component tasks).

Since SIADEX handles numerical objects like PDDL fluents (that can also be dynamically linked to external Python calls) it achieves a basic handling of numeric resources.

```
 (:functions
 (distance ?x1 ?y1 ?x2 ?y2)
{
 import math
 return math.sqrt ( (?x2 - ?x1) *
    (?x2 -?x1) +
    (?y2 - ?y1) *
    (?y2 - ?y1))
}
...
```

Figure 5: Embedded python in the domain

References

Allen, J.; Schubert, L.; Ferguson, G.; Heeman, P.; Hwang, C.; Kato, T.; Light, M.; Martin, N.; Miller, B.; Poesio, M.; and Traum, B. 1995. The TRAINS project: a case study in building a conversational planning agent. *Experimental and Theoretical Artificial Intelligence* 7:7–48.

Avesani, P.; Perini, A.; and Ricci, F. 2000. Interactive case-based planning for forest fire management. *Applied Intelligence* 13(1):41–57.

Biundo, S., and Schattenberg, B. 2001. From abstract crisis to concrete relief - a preliminary report on combining state abstraction and htn planning. In *6th European Conference on Planning (ECP-01)*.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49:61–95.

Edelkamp, S., and Hoffmann, J. 2004. The language for the 2004 international planning competition. http://ls5-www.cs.uni-dortmund.de/ edelkamp/ipc-4/pddl.html.

ESRI. http://www.esri.com.

Myers, K. L. 1999. CPEF: A continuous planning and execution framework. *AI Magazine* 20(4):63–69.

N. Policella, S. Smith, A. C. A. O. 2004. Generating robust schedules through temporal flexibility. In *14th International Conference on Automated Planning and Scheduling, ICAPS.*

National Library of Medicine. http://protege.stanford.edu/.

Local (Human-Centered) Replanning in the SIADEX Framework

Marc de la Asunción, Luis Castillo, Juan Fernández-Olivares, Oscar García-Pérez, Antonio González, Francisco Palao

Dpt. Ciencias de la Computación e I.A.
ETSI Informática
18071 Granada, SPAIN
siadexwww@decsai.ugr.es

Abstract. This paper presents the replanning capabilities in the SIADEX planning framework[6], which are based in the chronological order of generation of actions during the planning stage. This is a local replanning strategy since the replanning of an action only affects to older actions in the search tree and newer actions (actions that were generated after the replanned action) are maintained unaltered. This process is intended to regenerate the plan taking into account the the chronological steps given to obtain the current plan in a mixed initiative approach with the collaboration of a human expert.

1 Introduction

Planning for real world problems is becoming a need to bridge the gap between theory and practice in the AI Planning community. However this task is full of difficulties that arise from many different sources but one of the most important ones is the uncertainty of the knowledge being used to obtain plans. In this sense, the available knowledge is usually faulty so, there may be knowledge that is only partially known (incompleteness), uncertain knowledge about the effects of an action in its environment (nondeterminism) or knowledge that may not be perfectly known, mainly different types of metric knowledge (imprecision).

There are only two approaches to overcome this uncertainty: a preventive one and a palliative one. Preventive approaches for handling uncertain knowledge in AI planning frameworks try to foresee this uncertainty during the planning process and before the execution of the final plan. Therefore one may find different approaches that build alternative, conditional, branches to foresee any possible contingency during the execution of a plan [7, 11, 3], approaches that take into account the probability of every possible outcome of an action and build different strategies to react upon the detection of unexpected effects [2] or approaches that bound the imprecision of part of the knowledge and obtain plans adapted to these boundaries [8, 5]. These approaches might be called “off-line” approaches since they separate the stages of planning and execution as different, sequential phases of the resolution of a problem like a batch process.

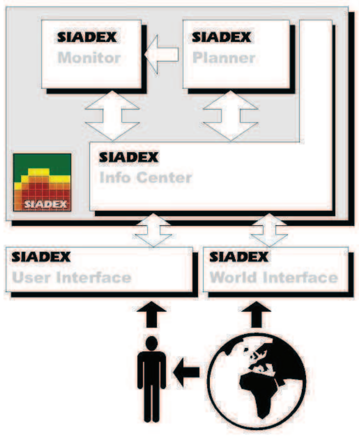
On the other hand, palliative approaches make a simplifying assumption that everything will go as expected so plans are obtained deterministically and, in the case that something could go wrong during the execution, the planner might be invoked again to design a subplan for the new contingency. Therefore, these approaches might be called "online" approaches since they may interleave several planning and execution stages during the resolution of the same problem. Basically one may find approaches for continual planning [9, 10], that interleave the design and execution of pieces of a plan until the complete problem is solved, or replanning approaches [13, 1] that redesign and replace a piece of a plan after a failure has been detected. In these approaches there are three main issues to be solved. The first one is detecting the source of the failure, the second one is delimiting the impact of the failure and the third one is redesigning the part of the plan that has failed.

The framework presented in this paper follows this last replanning approach and gives a solution for the second and third issues but taking into account that the planner is continuously being monitored and validated by a human operator. In this sense, the replanning capabilities of SIADEX must be human centered, that is, they only produce a local redesign of the part of the plan that has failed maintaining the remaining of the plan unaltered. Global replanning is only carried out if the impact of the failure is very deep, and when even the goal of the plan could be put in danger. This replanning strategy is based on the chronological order of generation of actions and it affects either older actions or newer actions that causally depend on the part that has failed. The remaining actions in the plan are maintained unaltered.

2 The SIADEX framework

SIADEX is a planning framework that is being developed under a research contract with the Andalusian Regional Government (Regional Ministry of Environment) [6]. It is intended to assist technical staff in the design of forest fire fighting plans but the ideas presented in this paper are domain independent and they might be applied to any mixed initiative replanning framework. The core of the SIADEX architecture, Figure 1.a), is a generative planner that obtains extended plans following a classical partial order causal link based planning algorithm [12] outlined in Figure 1.b). This algorithm will be explained later. These plans are then executed under the supervision of the monitoring module and a human operator. This is coordinated by the InfoCenter module that interfaces all of the interactions between SIADEX and the outer world:

– Receives planning requests by the user (through the User Interface).
– Ask the planner for new plans.
– Upon notification of the monitoring module, it launches execution orders to the human operator (through the User Interface) or to external agents (through the World Interface)
– Gathers information on the execution of the plan by indirect observation of a human operator (through the User Interface) or by direct gathering (through the World Interface) and translates them into the Monitor.
– Raises alerts about possible execution failures upon notification of the Monitor or upon direct user request.



(a)

```
SIADEX($\Pi$,$\mathcal{O}$, $\mathcal{L}$, $\mathcal{A}$, $\mathcal{D}$)

1. If $\mathcal{A}$ is empty Return $\Pi$,$\mathcal{O}$, $\mathcal{L}$
2. Goal Selection: Extract $(l, a, t)$ from $\mathcal{A}$
3. Action Selection: Select an action $a'$ to solve
   $(l, a, t)$ such that either
   (a) $a' \in \Pi$ (Reuse an action)
   (b) $a' \in \mathcal{D}$ (Insert a new action)
   (c) Return FAIL
4. Update
   (a) $\mathcal{L} = \mathcal{L} \cup (a', l, a, NOW())$
   (b) $\mathcal{O} = \mathcal{O} \cup a' < a$
   (c) $\Pi = \Pi \cup a'$
   (d) $\forall l \in Preconditions(a'), \mathcal{A} = \mathcal{A} \cup$
       $(l, a', \infty)$
5. Solve causal interferences: if an action $a^t$ threat-
   ens a causal link $(a^p, l, a^c, t)$ then either
   (a) $\mathcal{O} = \mathcal{O} \cup a^t < a^p$
   (b) $\mathcal{O} = \mathcal{O} \cup a^c < a^t$
   (c) Return FAIL
6. Recursively call SIADEX($\Pi$,$\mathcal{O}$, $\mathcal{L}$, $\mathcal{A}$, $\mathcal{D}$)
```

(b)

Fig. 1. The architecture of SIADEX and the generation of chronologically extended plans. Π stands for a set of actions, i.e., the plan, $\mathcal{O}$ is an order relation over Π, $\mathcal{L}$ is the plan rationale, $\mathcal{A}$ is the agenda of pending subgoals to be solved, and $\mathcal{D}$ is the set of actions schemas in the problem domain

As may be seen this architecture assumes that the execution of a previously designed plan in a dynamic world might not go as expected so it includes two different levels of response to this dynamicity. The first level is the monitoring module, that allows to trace the execution of the plan, detect possible failures and determine the possible impact of the failure. This is explained in sections 2.1 and 3. After the failure of an action has been detected, a mixed-initiative replanning episode is launched. The interaction is coordinated through InfoCenter and it may be as simple as re-executing the failed action (if this is feasible taking into account the temporal constraints in the problem) or as complex as deleting the remaining plan and redesigning a new one. This is explained in section 5. In any case, the new plan is monitored again in a continuous loop that ends with the successful execution of a plan.

2.1 Generation of chronologically extended plans

These extended plans obtained in SIADEX (Figure 1.b) contain a temporally ordered sequence of actions Π and the plan rationale $\mathcal{L}$. The plan rationale records the cause-effect relationships between actions in the plan as well as the chronological point in the resolution process at which every relationship was included. Thus, every record in $\mathcal{L}$ is a tuple (a^p, l, a^c, t) that means that at time stamp t, action $a^p \in \Pi$ was used to solve the precondition l of action $a^c \in \Pi$. These records are named *causal links*. Taking into account that the planning algorithm follows a backwards search process, the age of an action $a \in \Pi$ in the plan, that is, the chronological point at which the action was included is given by $\tau(a) = \min_{(a,l,a',t) \in \mathcal{L}} t$.

The agenda $\mathcal{A}$ that contains the pending subgoals to be solved is also annotated with time stamps. A goal (l, a, t) in the agenda is a precondition l of an action a that is to be solved. If $t = \infty$ then the goal is being newly generated, otherwise $t \neq \infty$ means that it is a goal being replanned and it was originally solved at time stamp t.

For monitoring purposes only the plan Π, the temporal constraints defined over the actions $\mathcal{O}$ and the plan rationale $\mathcal{L}$ are needed. The time stamp t in every record of the plan rationale is only used for replanning.

3 Monitoring in dynamic environments

The monitoring module of SIADEX is based on a temporal scheduling and rescheduling policy over temporal plans [5] so that actions in a plan are continuously being selected for execution following the best temporal ordering, their execution is monitored and possible faults are detected (Figure 2). When a fault is detected its impact is calculated, and a replanning episode starts in which the user may interact with SIADEX, making some suggestions (delete or add actions by hand, add or delete goals or literals). These suggestions are processed by the planning algorithm and new interactions are requested to the user until a valid plan is obtained, that is scheduled again for its execution and monitoring.

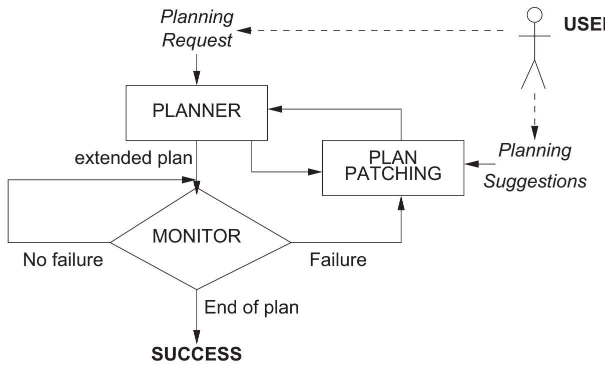


Fig. 2. The mixed-initiative planning and replanning loop in SIADEX

In order to correctly monitor the execution of a plan, SIADEX is based on two main constructs, the *Currently Known Horizon* (CKH) and the *Current Planning Horizon* (CPH). CKH Is the current state of the world since the initial state of the problem along actions have been completely executed. Initially CKH is the initial state of the problem. CPH is a prediction about the next changes to come in the near future. The monitoring procedure is shown in Figure 3.

1. Let us consider
   - $\mathcal{X}$ The set of actions already executed
   - Π The set of actions that are to be executed
   - $\mathcal{Q}$ The queue of actions that are currently under execution
2. Initially $X = \emptyset$, $Q = \emptyset$ and $\Pi =$ the plan, $CKH = Initial - State$, $CPH = \emptyset$
3. Loop
   (a) Extract from Π the next actions to be executed and insert them into Q
   (b) Insert all of their effects in CPH
   (c) If a literal in CPH has been achieved in the real world, move it from CPH to CKH
   (d) If an action has achieved all its effects, move it from $\mathcal{Q}$ to $\mathcal{X}$
   (e) If an action in $\mathcal{Q}$ cannot achieve some of its effects or a literal in CKH has been deleted then return FAILURE
   (f) If $\Pi = \mathcal{Q} = \emptyset$ then return SUCCESS

Fig. 3. The monitoring steps

Initially, CPH is empty and every time a action is launched for its execution, all its effects are included in CPH. Every effect takes a different time to be achieved [4, 5] so the execution is continuously monitored and, once an effect has been achieved in the real world, it is moved from CPH to CKH.

If everything goes well, all of the actions are correctly executed, CKH contains the description of a goal state and CPH is empty. However, a plan may fail at some point in its execution. The source of this failure may be either a missing condition, i.e., a literal that suddenly dissapears from CKH, or a missing effect, that is a literal that cannot be removed from CPH. A missing condition may be due to an external event that deletes a previously achieved effect or an initial condition. A missing effect may be due to the failure of an action (or the agent that executes it) so that part of its effect cannot be achieved.

Both cases produce a missing literal that, taking into account the plan rationale $\mathcal{L}$, might invalidate the causal dependencies of a set of actions. So, after the detection of the failure, the next step is determining its impact in the causal structure of the plan. This is done by function **DeleteLiteral(l)** shown in Figure 4. Additionally, during the monitoring of the plan, new unexpected literals might arise. In these cases function **AddLiteral(l)** is easily used.

Function **DeleteLiteral(l)** carries out three main activities. On the one hand, it start to regenerate the part of the plan rationale that had been altered by the deletion of l. In order to do so, it includes in the agenda $\mathcal{A}$ a new replanning subgoal for every altered causal link. These subgoals are not new subgoals but subgoals to be re-satisfied. Therefore they will have a time stamp $t \neq \infty$. Secondly, every action with a missing precondition due to the deletion of l is labeled as $OPEN$. These actions cannot be executed without re-satisfying the missing precondition. And finally, every action that

DeleteLiteral(l)

```
1. If $\nexists(a, l, a', t) \in \mathcal{L}$ do nothing
2. Otherwise $\forall(a, l, a', t) \in \mathcal{L}$ do
    (a) Remove $(a, l, a', t)$ from $\mathcal{L}$
    (b) Add $(l, a', t)$ to $\mathcal{A}$
    (c) Label $a'$ as $OPEN$
    (d) $\forall(a', l', a'') \in \mathcal{L}$
        i. Label $a''$ as $UNSTABLE$
```

AddLiteral(l)

```
1. $CKH = CKH \cup l$
```

Fig. 4. Two functions that may be used after the detection of a failure.

depends directly or indirectly on an $OPEN$ action is labeled as $UNSTABLE$ meaning that their preconditions are fully satisfied but that there may be some supporting action with a missing precondition.

4 User centered plan patching

This labeling of the action in the plan determines the future impact of the missing condition detected by the monitoring module. The decision to be taken is not easy, and it is left to the judgment of the human operator. The most conservative decision is to re-execute the failed action. The most aggressive decision is to purge the plan, that is, the deletion of all $OPEN$ actions and all $UNSTABLE$ actions and replan them completely. All of these possibilities may be carried out through InfoCenter in a plan patching stage that is driven under the control of a human operator (Figure 2). This transition in the loop of planning and replanning may be carried out in two different ways (depending on the problem domain).

- The execution of the plan is fully stopped. Then CPH and CKH are immediately stopped until the plan patching ends and the (possibly new) plan may continue its execution. This corresponds with a batch replanning system.
- The execution of the plan is not stopped and every action that were not altered by the failure is continued. Then CPH and CKH continue changing while there is any executable action not labeled as $OPEN$ either as $UNSTABLE$. This schema corresponds with an asincronous replanning system.

In any case, these are the suggestions that may be made by the human operator.

DeleteAction(a) This suggestion deletes the desired action a (only if it has not been executed yet), it deletes all of its effects in the plan, and recursively deletes all of the actions that were included explicitly to solve any precondition of a[1]. So this suggestion deletes the causal structure supporting a that is no longer needed, labels as $UNSTABLE$ the actions that causally depend on a, and generates some replanning goals.

[1] These are actions newer than a in the chronological order. Actions older than a are not deleted since they were introduced for other purpose and they were later reused by a during the planning process.

```
DeleteAction(a)

1. Only if $a \in \mathcal{Q} \cup \Pi$
2. Remove $a$ from $\mathcal{Q}$ or $\Pi$
3. $\forall l \in effects(a)$ DeleteLiteral($l$)
4. $\forall (a', l, a, t) \in \mathcal{L}$
   (a) Remove $(a', l, a, t)$ from $\mathcal{L}$
   (b) If $a' \in \mathcal{Q} \cup \Pi$
       i. If $\tau(a') > \tau(a)$ DeleteAction($a'$)
```

AddAction(a) This suggestion creates a set $\mathcal{P}$ of patches of the user that contains actions explicitly added by the user. These actions will be eventually reused later by the planner during the replanning stage.

```
AddAction(a)

1. $\mathcal{P} = \mathcal{P} \cup a$
```

DeleteGoal(g) This suggestion is very strong. It deletes all of the actions that were exclusively added to solve that goal and generate a replanning subgoal and, possibly, many replanning subgoals.

```
DeleteGoal(g)

1. Let $(a, g, END, t)$ the causal link that records the satisfaction of the goal $g$ by
   means of the action $a$.
2. DeleteAction(a)
3. Insert $(g, END, t)$ in $\mathcal{A}$
```

Addgoal(g) This action simply adds a new primary goal to the agenda.

```
AddGoal(g)

1. $\mathcal{A} = \mathcal{A} \cup (g, END, \infty)$
```

5 Local replanning in SIADEX

After the plan patching stage, the plan contains a damaged causal structure, with many open conditions still to be re-solved, some actions are new and others have dissapeared. The replanning episode has to regenerate the missing pieces of the plan to obtain a complete, valid plan. In order to do this, instead of replanning completely the missing actions, a local replanning procedure is carried out taking into account the time stamps in the newly generated replanning subgoals. This is a very close procedure to a human revision of a plan since a global redesign of a plan is only carried out if it is strictly necessary given that the execution of a completely new plan may be very costly in terms of reconfiguration of the agents and actuators in the real world. In this sense, local redesigns of a globally valid plan are more preferred. However, in the case that a local replanning is infeasible, a global replanning must be carried out.

Instead of using a specialized replanning algorithm, the original algorithm in Figure 1.b) was redesigned so that it is able to work in a purely generative episode and also in a replanning episode. The main additions are

– It is able to replan over a plan that is being executed, taking into account the existence of already executed actions and CKH.
– It reuses and extends the remaining plan rationale $\mathcal{L}$.
– It is able to include the actions suggested by the user in $\mathcal{P}$ although, due to the search process these actions might be finally rejected.
– The role of the time stamp in the causal structure is increased to provide a local replanning capability.
– In the case of a backtracking during a local replanning (steps 3e and 5c), all of the actions newer than the current problem being replaned are deleted triggering a global replanning process.

Therefore, during a generative episode $\mathcal{Q} = \mathcal{P} = CKH = \emptyset$ and all of the goals in $\mathcal{A}$ are of the form (l, a, ∞). On the other hand, during a replanning episode $\mathcal{Q}$, $\mathcal{P}$ and CKH may be non empty, and goals in $\mathcal{A}$ may have a time stamp different than ∞

6 Concluding remarks

This paper has shown the replanning capabilities of SIADEX, a system being developed under a research contract with the Andalusian Regional Government to assist technical staff in the design of forest fire fighting plans. Although this a very specific domain, the techniques shown in the paper are domain independent and they might be applied to any domain. These techniques are based on a local replanning capability with the following features

– It is a local replanning procedure since it does not produce a global replanning process if it is not necessary, so only the neighborhood of the failed action is replanned.
– It is a mixed initiative approach in which the user is informed of the existence and the impact of the failure and he/she is able to patch the plan by adding/deleting either actions or goals.
– It might be used to start a planning process either from scratch or from initial suggestions of the user and regenerating a plan over them. This is because this process is inherently the same to replanning so that the interaction with the user may be even closer.

On the opposite hand, it must be said that these techniques are not complete since after a failure in the execution, only the plan rationale is deleted and regenerated. Ordering and binding constraints posted by failed actions are not deleted neither regenerated though they are being studied for a more comprehensive replanning procedure in the SIADEX project.

```
SIADEX-R($\mathit{\Pi}$,$\mathcal{Q}$, $\mathcal{P}$, $\mathcal{O}$, $\mathcal{L}$, $\mathcal{A}$, $\mathcal{D}$, $CKH$)

1. If $\mathcal{A}$ is empty Return $\mathit{\Pi}$,$\mathcal{O}$, $\mathcal{L}$
2. Goal Selection: Extract $(l, a, t)$ from $\mathcal{A}$
3. Action Selection: Select an action $a'$ to solve $(l, a, t)$ such that either is a new action or a
   previously existing action with a time stamp older than the goal[a].
   (a) $a' \in \mathcal{X}$ and $l \in CKH$ Reuse an already executed action whose effects are still in
       $CKH$
   (b) $a' \in \mathcal{P}$ Reuse an action suggested by the user
   (c) $a' \in \mathcal{Q} \cup \mathit{\Pi}$ and $\tau(a') < t$ Reuse an action still to be executed. This corresponds to
       a delay in the execution of $a$ that was originally scheduled to be executed before, but,
       due to the failure it is still possible to execute it later reusing part of the existing plan
       rationale.
   (d) $a' \in \mathcal{D}$
   (e) Return FAIL and
       i. $\forall a \in \mathit{\Pi} \cup \mathcal{Q}$ such that $\tau(a) > t$ DeleteAction(a)
4. Update
   (a) if $t = \infty$ (new goal) $\mathcal{L} = \mathcal{L} \cup (a', l, a, NOW())$
       otherwise (replanning goal)
       i. $\mathcal{L} = \mathcal{L} \cup (a', l, a, t)$
       ii. Increase in 1 time unit the time stamp of every goal $(l, a, t') \in \mathcal{A}$ and every causal
           link $(a_1, l, a_2, t')$ such that $t < t'$
   (b) $\mathcal{O} = \mathcal{O} \cup a' < a$
   (c) $\mathit{\Pi} = \mathit{\Pi} \cup a'$
   (d) $\forall l \in Preconditions(a'), \mathcal{A} = \mathcal{A} \cup (l, a', t+1)$
5. Solve causal interferences[b]: if an action $a^t$ threatens a causal link $(a^p, l, a^c, t)$ then either
   (a) $\mathcal{O} = \mathcal{O} \cup a^t < a^p$
   (b) $\mathcal{O} = \mathcal{O} \cup a^c < a^t$
   (c) Return FAIL and
       i. $\forall a \in \mathit{\Pi} \cup \mathcal{Q}$ such that $\tau(a) > t$ DeleteAction(a)
6. Recursively call SIADEX-R($\mathit{\Pi}$,$\mathcal{Q}$, $\mathcal{P}$, $\mathcal{O}$, $\mathcal{L}$, $\mathcal{A}$, $\mathcal{D}$, $CKH$)
```

[a] For newly generated goals (with time stamp ∞) every action is a possible one. For replanning goals with a time stamp t only actions whose time stamp is lower are considered, that is, actions that were previously generated in the search tree. Newer actions, i.e., actions that were generated later in the search tree are not considered.

[b] This step does not consider time stamps since interferences may appear between actions and causal links of any age.

Fig. 5. The planning and replanning algorithm of SIADEX

References

1. J.A. Ambros-Ingerson and S. Steel. Integrating planning, executiong and monitoring. In *Proceedings of AAAI-88*, pages 83–88, 1988.
2. J. Blythe. Decision-theoretic planning. *AI Magazine*, 20(2):37–54, 1999.
3. L. Castillo, J. Fdez-Olivares, and A. González. A conditional approach for the autonomous design of reactive and robust control programs. In *Proc. of AIPS02 Workshop on Planning for Real World*, pages 59–68, 2002.
4. L. Castillo, J. Fdez-Olivares, and A. González. A temporal constraint network based temporal planner. In *Workshop of the UK Planning and Scheduling Special Interest Group, PLANSIG 2002*, pages 99–109, 2002.
5. L. Castillo, J. Fdez-Olivares, and A. González. Some issues on the representation and explpoitation of imprecise temporal knowledge in an AI planner. In *Knowledge-Based Intelligent Information and Engineering Systems*, Lecture Notes in Artificial Intelligence, LNAI-2774, pages 1321–1328. Springer-Verlag, 2003.
6. M. de la Asunción, L. Castillo, J. Fdez-Olivares, O. García-Pérez, A. González, and F. Palao. SIADEX: Assisted Design of Forest Fire Fighting Plans by Artificial Intelligence Planning techniques. `http://decsai.ugr.es/siadex`, 2003.
7. O. Etzioni, S. Hanks, D. Weld, D. Draper, N. Lesh, and M. Williamson. An approach to planning with incomplete information. In *Proc. Third. Int. Conf. on Principles of KRR-92*, pages 115–125, 1992.
8. P. Morris and N. Muscettola. Execution of temporal plans with uncertainty. In *AAAI 2000*, pages 491–496, 2000.
9. N. Muscettola, P. Pandurang Nayak, B. Pell, and B. C. Williams. Remote agent: to boldly go where no AI systems has gone before. *Artificial Intelligence*, pages 5–48, 1998.
10. K. L. Myers. CPEF: A continuous planning and execution framework. *AI Magazine*, 20(4):63–69, 1999.
11. M. A. Peot and D. E. Smith. Conditional nonlinear planning. In *Proc. First Int. Conf. of AIPS*, pages 189–197, 1992.
12. D. Weld. An introduction to least commitment planning. *AI Magazine*, 15(4):27–61, 1994.
13. D. E. Wilkins. *Practical planning: Extending the classical AI planning paradigm*. Morgan Kaufmann, 1988.

2. Spin-off creadas

2.1 IActive Intelligent Solutions, S.L.

Compañía fundada en 2006 para transferir los avances científicos en Inteligencia Artificial gestados en la Universidad de Granada al mercado. La compañía está especializada en el desarrollo de sistemas inteligentes a medida basados en planificación automática así como en el suministro de esta tecnología a otras compañías para que la integren en sus productos dotándolos de valor añadido. Actualmente trabajan más de 30 personas en la compañía.

2.2 Egestia Sistemas Inteligentes de Gestión OnLine, S.L.

Compañía fundada en 2009 para desarrollar aplicaciones en Internet de gestión empresarial con un modelo de negocio SaaS basadas en tecnologías en Inteligencia Artificial. Actualmente trabajan 9 personas en la compañía.

3. Productos desarrollados

3.1 IActive Knowledge Studio

IActive Knowledge Studio

Herramienta de desarrollo de sistemas basados en planificación automática que permite el modelado de dominios en notación EKMN y la integración del planificador inteligente en sistemas mediante XML.

3.2 IActive Intelligent Decisor

Decisor

Planificador jerárquico automático desarrollado sobre tecnología JAVA.

3.3 Smartourism

smartourism

Solución basada en planificación automática que permite la personalización de visitas turísticas basándose en el tiempo disponible, los recursos económicos y las preferencias de los turistas.

3.4 Doolphy

DOOLPHY®

Aplicación web para la gestión de proyectos a través de Internet que permite la planificación automática de proyectos basada en un planificador automático.

4. Otros méritos

A continuación se enumeran los premios obtenidos, tanto a nivel científico como a nivel empresarial en relación y como resultado del trabajo de la tesis.

- **2004:** Mejor proyecto final de carrera Ingeniería Informática. Un entorno web para la planificación heurística. Universidad de Granada.
- **2005:** Premio al mejor trabajo en Transferencia Tecnológica en Inteligencia Artificial. I Congreso Español en Informática. Jornadas TTIA.
- **2006:** Best Application Paper Award. ICAPS 2006 Conference, Cumbria (United Kingdoms).
- **2008:** Primer premio certamen de Creación de Empresas. Confederación de Empresarios de Andalucía.
- **2009:** Primer premio empresa Tecnológica Salón Mi Empresa. Madrid.
- **2010:** Primer premio Emprendedor XXI Andalucía. La Caixa.
- **2011:** Primer premio Jóvenes Emprendedores Empresa de Base Tecnológica. Bancaja.

III. Bibliografía

1. *Local (human-centered) Replanning in the SIADEX Framework.* **M. de la Asunción, L. Castillo, J. Fdez.-Olivares, O. García-Pérez, A. González, F. Palao.** s.l. : X Conference of the Spanish Association for Artificial Intelligence, Workshop on Planning, Scheduling and Temporal Reasoning., 2003.

2. *SIADEX: an interactive artificial intelligence planner for decision support and training in forest fire fighting.* **F. Palao., M. de la Asunción, L. Castillo, J. Fdez-Olivares, O. García-Pérez, A. González.** s.l. : AIComm special issue on Binding Environmental Sciences and artificial intelligence., 2005.

3. *Knowledge and plan execution management in planning fire fighting operations.* **M. de la Asunción, L. Castillo, J. Fdez.-Olivares, O. García-Pérez, A. González, F. Palao.** s.l. : Workshop on Planning and Scheduling: Bridging Theory to Practice. European Conference on Artificial Intelligence., 2004.

4. *Temporal enhancements of an HTN planner.* **L. Castillo, J. Fdez.-Olivares, O. García-Pérez, F. Palao.** s.l. : Spanish Conference on Artificial Intelligence, CAEPIA., 2005.

5. *Improving Planning Techniques for Web Services.* **Palao-Reinés, F.** s.l. : Doctoral Consortium ICAPS 06., 2006.

6. *SIADEX. An integrated planning framework for crisis action planning.* **L. Castillo, J. Fdez.-Olivares, O. García-Pérez, F. Palao.** s.l. : International Conference on Automated Planning and Scheduling, ICAPS 2005, Software Demonstrations Track, 2005.

7. *EKDL: A graphic notation for modeling hierarchical planning domains.* **Luis Castillo, Juan Fernández, Óscar García, Francisco Palao.** s.l. : Workshop KEPS. ICAPS (International Conference on Automated Planning and Scheduling), 2011.

8. *Smartourism: An Intelligent Systems for personalizing Sightseeing trip.* **Luis Castillo, Juan Fernandez, Óscar García y Francisco Palao.** s.l. : ICAPS (International Conference on Automated Planning and Scheduling)., 2011.

9. **Gartner.** Evacuation Case of Study. [En línea] 2011.

10. **W. Chan Kim, Renée Mauborgne.** *Blue Ocean Strategy.* s.l. : Harvard Business School Press, 2005.

11. *Workflow Management: Models, Methods, and Systems.* **W.M.P. van der Aalst, K.M. van Hee.** s.l. : MIT press, Cambridge, MA, 2002., 2002.

12. *Workflow Management: Modeling Concepts, Architecture and Implementation.* **S. Jablonski, C. Bussler.** s.l. : International Thomson Computer Press, London, UK,, 1996.

13. **Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, Mathias Weske.** Business Process Management: A Survey . s.l. : Lecture Notes in Computer Science. Volume 2678/2003, 1019, 2003.

14. **Inc., TIBCO Software.** Business Process Management (BPM) Resource Center. [En línea] http://www.tibco.com/solutions/bpm/default.jsp.

15. **OMG.** Business Process Model and Notation. *http://www.omg.org/spec/BPMN/.* [En línea] 2011.

16. **(IBM), ILOG.** ILOG Business Rule Management Systems. [En línea] 2010. http://www-01.ibm.com/software/websphere/products/business-rule-management/.

17. **Efraim Turban, Ramesh Sharda, Jay E. Aronson, David King.** *Business Intelligence: A Managerial Approach.* s.l. : Pearson/Prentice Hall. ISBN: 978-0-13-234761-7, 2010.

18. **Daniela Grigor, Fabio Casati, Malu Castellanos, Umeshwar Daya, Mehmet Sayal, Ming-Chien Shan.** *Business Process Intelligence.* s.l. : Computers in Industry. Volume 53, Issue 3, April 2004, Pages 321-343, 2003.

19. *Mastering the Unpredictable: How Adaptive Case Management Will Revolutionize the Way That Knowledge Workers Get Things Done.* **Swenson, Keith D.** s.l. : Meghan-Kiffer Press., 2010.

20. *Smart Process Management: Automated Generation of Adaptive Cases based on Intelligent Planning Technologies. .* **Arturo González Ferrer, Juan Fdez-Olivares, Inmaculada Sánchez-Garzón and Luis Castillo.** s.l. : roc. of BPM Demonstration Track 2010, Hoboken, USA, September 14-16, 2010, Vol. 615 CEUR-WS.org, 2010.

21. **IBM.** How BRMS extends the value of BPM. [En línea] http://www-01.ibm.com/software/websphere/products/business-rule-management/process-improvement/.

22. *A Call for Knowledge-Based Planning.* **David E. Wilkins, Marie desJardins.** s.l. : AI Magazine Volume 22 Number 1. (AAAI), 2001.

23. *Reducing the impact of AI Planning on end users.* **L. Castillo, J. Fdez-Olivares, O. García-Pérez, A. Gonzalez, F. Palao.** s.l. : ICAPS 2007, Workshop on Moving Planning and Scheduling Systems into the Real World (Keynote talk), 2007.

24. **Fikes, R. E., Hart, P. E., Nilsson, N. J.,.** Learning and executing generalized robot. *Artificial Intelligence.* 1972.

25. *From Abstract Crisis to Concrete Relief - A Preliminary Report on Combining State Abstraction and HTN Planning.* **S. Biundo and B. Schattenberg.** s.l. : 6th European Conference on Planning (ECP-01), 2001.

26. *HICAP: An interactive case-based planning architecture and its application to noncombatant evacuation operations.* **H. Muñoz-Avila and D. W. Aha and L. Breslow and D.**

Nau. s.l. : AAAI Press, Ninth Conference on Innovative Applications of Artificial Intelligence, 1999.

27. *Applying an AI plannet to military operations planning.* **Desimone, D. E. Wilkins and R. V.** s.l. : Intelligent Scheduling, 1994.

28. *Interactive Case-Based Planning for Forest Fire Management.* **Ricci, P. Avesani and A. Perini and F.** s.l. : Applied Intelligence, 13-1, Pg: 41-57., 2000.

29. *A Nonlinear Planner for Solving Sequential Control Problems in Manufacturing System.* **L. Castillo, A. González.** s.l. : Progress in Artificial Intelligence, Lecture Notes in Artificial Intelligence, Springer-Verlag., 1998.

30. *Mixed-Initiative Planning in MAPGEN: Capabilities and Shortcommings.* **Rajan, John L. Bresina and Ari K. Jonsson and Paul H. Morris and Kanna.** 2005.

31. *CBET: a case base exploration tool.* **Avesani, P. and Perini, A. and Ricci, F.** s.l. : Lecture Notes in Computer Science. Vol: 1321: Pg: 405., 1997.

32. *Iterative repair planning for spacecraft operations using the ASPEN system.* **G. Rabideau and R. Knight and S. Chien and A. Fukunaga and A. Govindjee.** 199.

33. *PDDL: the planning domain definition.* **M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, D. Wilkins.** s.l. : AIPS-98 Planning Committee., 1998.

34. *Smart Process Management: Automated Generation of Adaptive Cases based on Intelligent Planning Technologies.* **Arturo González Ferrer, Juan Fdez-Olivares, Inmaculada Sánchez-Garzón, Luis Castillo.** s.l. : Proc. of BPM Demonstration Track 2010, Hoboken, USA, September 14-16, 2010, Vol. 615 CEUR-WS.org, 2010.

35. *A Middleware for the automated composition and invocation of semantic web services based on HTN planning techniques.* **Juan Fernandez Olivares, Tomás Garzón, Luis Castillo Vidal, Óscar García Pérez, Francisco Palao.** s.l. : Current topics in Artificial Intelligence (CAEPIA 2007). Springer LNAI 4788., 2007.

36. *HTN planning: Complexity and expressivity.* **K. Erol, J. Hendler, D. Nau.** s.l. : Proceedings of the Twelfth National Conference on Artificial Intelligence, Pg: 1123-1128., 1994.

37. *SIADEX: A Real World Planning Aproach for Forest Fire Fighting.* **Marc de la Asuncion, Óscar García Pérez, Francisco Palao.** s.l. : ISBN:1-58603-451-0. IOS Press., 2004.

38. *Planning in a hierarchy of abstraction spaces.* **Sacerdoti, D. E.** s.l. : Artificial Intelligence, 5, Pg: 115, 135., 1974.

39. *Generating project networks.* **Tate, A.** s.l. : Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Pg: 888-893., 1977.

40. *Domain-independent planning: Representation and plan generation,.* **Wilkins, D.E.** s.l. : Artificial Intelligence,22-, Pg: 269-301., 1984.

41. *UMCP: A sound and complete procedure for hierarchical task-network planning.* **K. Erol, J. Hendler, D. Nau.** s.l. : Proc. 2nd Intl. Conf. on AI Planning Systems, Pg: 249-254., 1994.

42. *PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains.* **M., Long D. and Fox.** s.l. : Journal of Artificial Intelligence Research, 20, 2004.

43. *SHOP2: An HTN Planning System.* **D. Nau and T.C. Au, O. Ilghami.** s.l. : Journal of Artifial Intelligence Research, 20, Pg: 379-404., 2003.

44. **R. Fikes, N. Nilsson.** *STRIPS: A new approach to the application of theorem proving to problem solving.* s.l. : Artificial Intelligence, 2, Pg: 189, 208., 1971.

45. **D. Nau, S.J.J. Smith, K. Erol.** *Control strategies in HTN planning:Theory versus practice.* s.l. : Proceedings of the National Conference on Artificial Intelligence, 15, Pg: 1127-1133., 1998.

46. *ADL: Exploring the middle ground between STRIPS and the situation calculus.* **Pednault, E.** 1989.

47. *Integrating planning and learning: The PRODIGY architecture.* **M. Veloso, J. Carbonell, A. Pérez, D. Borrajo, E. Fink.** s.l. : Journal of Experimental and Theoretical Artificial Intelligence,1-7., 1995.

48. *Practical planning: Extending the classical AI planning paradigm.* **Wilkins, D. E.** s.l. : Morgan Kaufmann, 1988.

49. *An introduction to least commitment planning.* **Weld, D.** s.l. : AI Magazine,15-4, Pg: 27, 61., 1994.

50. *A heuristic estimator for means-ends analysis in planning.* **D., McDermott.** s.l. : AAAI Press, Proceedings of the 3rd International Conference on Artificial Intelligence Planning Systems (AIPS-96), Pg: 142-149., 1996.

51. *Plan Constraints and Preferences in PDDL3.* **A. Gerevini, D. Long.** s.l. : ICAPS Workshop on Soft Constraints and Preferences in Planning, 2006.

52. *A Limited Extension of PDDL for planning with Non-Primitive Actions.* **Biplav, Srivastava.** s.l. : Workshop on the Competition: Impact, Organization, Evaluation, Benchmarks. ICAPS. , 2003.

53. *PDDL+ level 5: An Extension to PDDL2. 1 for Modelling Planning Domains with Continuous Time-dependent Effects.* **M. Fox, D. Long.** s.l. : Technical report, University of Durham., 2001.

54. **Pérez, Óscar Jesús García.** *Tesis Planificación en dominios temporales.* Granada : Universidad de Granada, 2007.

55. *Process Interchange Format and Framework, Version 1.1.* **Lee, J. (ed.), Gruninger, M., Jin, Y., Malone, T., Tate, A., Yost. G.** s.l. : MIT Center for Coordination Science, Working Paper No. 194, 1996.

56. *"Workflow Management Coalition Glossary", A Workflow Management Coalition Specification.* **Coalition, Workflow Management.** s.l. : http://www.wfmc.org, November, 1994.

57. Planning Initiative Shared Planning and Activity Representation. [En línea] DARPA/Air Force Research Laboratory (Rome) Planning Initiative (ARPI). http://www.aiai.ed.ac.uk/project/spar/.

58. *Engineering, GIPO II: HTN Planning in a Tool-supported Knowledge.* **T. L. McCluskey, D. Liu, R. M.Simpson.** s.l. : ICAPS-03 Proceedings., 2003., 2003.

59. *itSIMPLE2.0: An Integrated Tool for Designing Planning Domains.* **Tiago S. Vaquero1, Victor Romero, Fernando M. Sette, Flavio Tonidandel, Jose Reinaldo Silva.** s.l. : ICAPS, 2007.

60. Unified Modeling Language. [En línea] http://www.omg.org/spec/UML/.

61. Object Management Group. [En línea] http://www.omg.org/.

62. *Using the SIPE-2 Planning System.* **E.Wilkins, David.** s.l. : Artificial Intelligence Center. SRI International., 2000.

63. *Approach to the Application of Knowledge Based Planning and Scheduling Techniques to Logistics.* **Tate, A., Drabble. B. and Dalton, J., An Engineers.** s.l. : O-Plan Final Technical Report . December 1995.

64. *Engineering and compiling planning domain models to promote validity and efficiency.* **T. L. McCluskey, J. M. Porteous.** s.l. : Artificial Intelligence, vol. 95, nº. 1, págs. 1–65, 1997.

65. *Using Knowledge Engineering for Planning Techniques to leverage the BPM life-cycle for dynamic and adaptive processes.* **Juan Fdez-Olivares, Arturo Gonzalez-Ferrer, Inmaculada Sanchez-Garzon, Luis Castillo.** Toronto, Canada : ICAPS 2010. Workshop on Knowledge Engineering for Planning and Scheduling (KEPS 2010)., 2010.

66. *T. L. McCluskey, D. Liu, R. M.Simpson.* **Engineering, GIPO II: HTN Planning in a Tool-supported Knowledge.** s.l. : ICAPS-03 Proceedings., 2003.

67. *Mixing expresiveness and efficiency in a manufacturing planner.* **A. González, L. Castillo, J. Fdez-Olivares.** s.l. : Journal of Experimental and Theoretical Artificial Intelligence.13, Pg: 141-162., 2001.

68. *Temporal constraint network.* **R. Dechter, I. Meiri, J. Pearl.** s.l. : Elsevier Science Publishers Ltd. Essex, UK, Artificial Intelligence,49 (1-3), 1991.

69. **D.S Weld, C.R. Anderson, D.E Smith.** *Extending Graphplan to Handle Uncertainty and Sensing Actions.* s.l. : Proceedings of the 15th National Conference on AI, Pg: 897-904., 1998.

70. *PDDL2-1: an extension to PDDL for expressing temporal planning domains.* **M. Fox, D. Long.** s.l. : ., 2001.

71. *Planning in a hierarchy of abstraction spaces.* **Sacerdoti, D. E.** s.l. : Artificial Intelligence, 5, Pg: 115, 135., 1974.

72. **Inc., TIBCO Software.** Business Process Management (BPM) Resource Center. [En línea] http://www.tibco.com/solutions/bpm/default.jsp.

73. *Workflow Management: Modeling Concepts, Architecture and Implementation.* **S. Jablonski, C. Bussle.** s.l. : International Thomson Computer Press, London, UK, 1996.

74. *—.* **S. Jablonski, C. Bussler.** s.l. : International Thomson Computer Press, London, UK, 1996.

75. **Hentenryck, Pascal Van.** *The OPL Optimization Programming Language.* s.l. : The MIT Press. ISBN-10:0-262-72030-2, 1999.

76. **Kumar, Vipin.** *Algorithms for Constraint-Satisfaction Problems: A Survey.* s.l. : AI Magazine, Vol 13, No 1, 1992.

77. **IBM.** Business Rules and Business Process Improvement. [En línea] 2010. http://www-01.ibm.com/software/websphere/products/business-rule-management/process-improvement/.

78. *Bringing users and planning technology together. Experiences in SIADEX.* **L. Castillo, J. Fdez.-Olivares, O. García-Pérez, F. Palao.** s.l. : 16th International Conference on Automated Planning and Scheduling (ICAPS 2006), 2006.