Un enfoque para la generación de Plan Corpus con un Algoritmo de Planning

Juan Francisco Silva Logroño, Luis Berdún, Marcelo Armentano[1], Analia Amandi[1]

Instituto de Investigación ISISTAN, Fac. de Cs. Exactas, UNCPBA
Campus Universitario, Paraje Arroyo Seco, Tandil, 7000, Argentina
[1]también CONICET, Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina
{juanf.silval}@gmail.com;{lberdun, marmenta, amandi}@exa.unicen.edu.ar

Abstract. Obtener un plan corpus a partir del cual aprender una biblioteca de planes que sirva de base de conocimiento para un sistema de reconocimiento de planes es una tarea difícil. En este trabajo se presenta un enfoque basado en planning para la generación automática de secuencias de acciones que llevan al cumplimiento de diferentes objetivos. El método propuesto permite adicionalmente la especificación de preferencias que se utilizan a la hora de elegir qué acción seguir en un punto determinado del plan de manera tal de poder modelar fácilmente diferentes perfiles de usuarios que utilicen el sistema.

Keywords: Reconocimiento de planes, Planning, modelado de usuarios

1 Introducción

El reconocimiento de planes tiene como finalidad reconocer el objetivo de un sujeto basando en las acciones que éste ejecuta en un ambiente determinado. Dicho objetivo del usuario tiene asociado uno o más planes que pueden predecir el comportamiento subsecuente del sujeto. Todo sistema de reconocimiento de planes necesita una biblioteca de planes como entrada. Las bibliotecas de planes son una base de conocimiento que codifica de alguna manera las creencias de un agente con respecto a cómo el usuario puede alcanzar un objetivo particular dentro del dominio de la aplicación. En la literatura se han propuesto varias formas de representar bibliotecas de planes así como también varios métodos para inferir la intención del usuario utilizando este conocimiento.

Las bibliotecas de planes han sido tradicionalmente codificadas manualmente por un experto del dominio de la aplicación [1, 2, 3, 4, 5, 6]. En estos casos, el experto del dominio es el encargado de analizar todos los posibles objetivos que deben ser modelados y todas las secuencias alternativas de tareas o acciones que el usuario debe ejecutar para alcanzar dichos objetivos, junto a las restricciones de ejecución existentes entre ellas.

El éxito o fracaso de un sistema de reconocimiento de planes depende altamente de la correctitud y completitud de la biblioteca de planes que utiliza. Sin embargo, construir una biblioteca de planes de manera manual es una tarea tediosa y propensa a errores.

Partiendo de la hipótesis que los usuarios generalmente muestran patrones recurrentes de comportamiento cuando interactúan con una aplicación de software, una solución alternativa a la construcción manual de bibliotecas de planes es hacer uso de ejemplos de

secuencias ejecución de tareas para alcanzar un objetivo determinado para obtener de manera automática las bibliotecas de planes.

Es por lo expuesto anteriormente que en los últimos años varios grupos de investigación han prestado especial atención a la representación de bibliotecas de planes mediante modelos que capturan regularidades en el comportamiento del usuario a partir de trazas de ejecución obtenidas de la interacción del usuario con la aplicación. Parte de estos esfuerzos se ha centrado en aprender los parámetros de un modelo cuya estructura predefinida permanece inalterable [7, 8, 9, 10, 11, 12]. Por otro lado, se ha estudiado la tarea de aprender bibliotecas de planes a partir del historial de interacción de un usuario con la aplicación [13, 14, 15, 16, 17, 18].

Aprender bibliotecas de planes a partir de secuencias de entrenamiento tiene la principal ventaja que no es necesario contar con ninguna información adicional del dominio a modelar más que el conjunto de tareas y el conjunto de objetivos posibles.

En este trabajo se propone la utilización de técnicas de planning [20] como motores de generación de bibliotecas de planes. El trabajo presenta una técnica que, partiendo de los operadores del dominio y los objetivos deseados, permite obtener un conjunto posible de planes que conformara la biblioteca de planes para el dominio. Asimismo, se muestra cómo es posible que dicho conjunto sea generado teniendo en cuenta las preferencias de los usuarios.

El trabajo se organiza de la siguiente forma: en la sección 2 se presenta nuestra solución propuesta, mientras que en la sección 3 se detallan los pasos de la técnica y los detalles del algoritmo. En la sección 4 se muestran algunos resultados experimentales. La sección 5 presenta algunos de los trabajos relacionados. Por último, la sección 6 presenta las conclusiones del trabajo.

2 Creación automática del plan corpus

Dados los inconvenientes del proceso de generar un plan corpus manualmente, en este trabajo proponemos la utilización de un algoritmo de planning de orden parcial[1] para generar automáticamente un plan corpus. La técnica propuesta permite obtener un plan corpus etiquetado por objetivos y una estructura jerárquica para el dominio deseado, de forma precisa y de bajo costo.

La premisa bajo la cual desarrollamos el trabajo consiste en que el usuario establezca de manera simple cuál es el dominio del cual desea obtener el plan corpus, cuáles son los operadores que aplican al mismo y cuáles son los distintos objetivos a modelar. Asimismo se busca permitir ingresar a través de preferencias y restricciones, cómo se desea caracterizar al plan, por ejemplo modelar distintos perfiles de usuario que podrían utilizar una herramienta, tales como un usuario experto, o un usuario inexperto. De todo lo anterior se deduce que el diseñador no necesita conocer todas las posibles interacciones complejas de los operadores para lograr alcanzar un objetivo, logrando con la técnica propuesta obtener el plan corpus por descripción y no por enumeración exhaustiva.

[1] Los algoritmos de planning de orden parcial se diferencian de los de orden total en que sólo establecen las restricciones de orden mínimas entre las acciones necesarias para alcanzar el objetivo, dando lugar a diferentes órdenes totales.

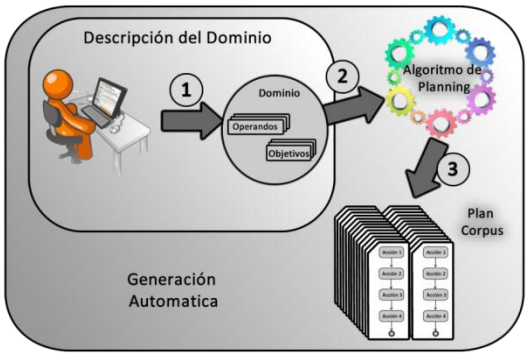

Fig. 1. Marco de trabajo propuesto.-

La Figura 1 presenta el marco del trabajo propuesto: el usuario especifica los operadores del dominio y los distintos objetivos y, a partir de ellos, el algoritmo de planning permite obtener los planes que conformarán el plan corpus para el dominio planteado.

Normalmente un algoritmo de planning de propósito general sólo permite acceder a una única solución que resuelve el problema en cuestión. Para nuestro fin es necesario obtener un determinado número de planes que describan caminos alternativos para alcanzar un objetivo, ya que estos formarán el plan corpus a partir del cual se entrenará el sistema de reconocimiento de planes. El motivo por el cual se utilizó un algoritmo de orden parcial es que al obtener una solución se obtiene un plan el cual sólo posee asociado las restricciones mínimas de orden entre los operadores. Esto nos permite extraer de un plan de orden parcial varios planes de orden total y por lo tanto varias secuencias alternativas de acciones que llevan a cumplir el objetivo. La Figura 2 nos muestra un ejemplo en el cual un plan de orden parcial de 4 acciones puede ser visto como distintos planes de orden total.

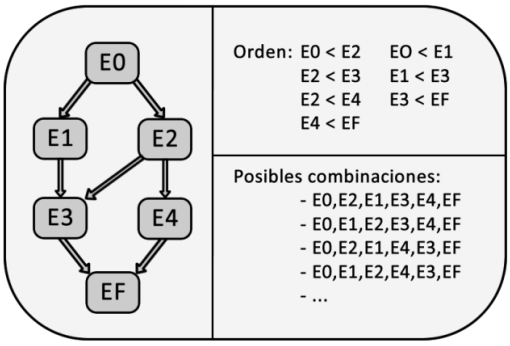

Fig. 2. Ejemplo de Plan de orden parcial y los distintos planes de orden total que representa.-

3 Generación del Plan Corpus como un problema de planning

A continuación se detallan los pasos que permiten ver el problema de obtener un plan corpus para un determinado dominio como un problema de planning.

3.1 Definición de las acciones

En el proceso de generar el plan corpus uno de los primeros pasos a realizar es definir el conjunto de operadores con los que se va a trabajar. Por ejemplo, si se modela un cliente de web mail algunas de las posibles operaciones serían "Escribir Mail", "Escribir Subject", "Buscar Destinatario", "Escribir cuerpo", etc.

Una vez identificados los operadores es necesario definir las acciones que los representan en el contexto de un problema de planning. Para ello el usuario debe extraer de los operadores identificados su "significado". La técnica en este punto es bastante simple, por cada operador se debe identificar qué efectos tiene su ejecución y bajo qué condiciones es aplicable; lo que necesita son las precondiciones de la acción, mientras que lo que produce son los efectos de la misma.

A fin de ilustrar el comportamiento se utiliza notación basada en Prolog, la cual representa el problema de manera simple y natural. La notación utilizada es la siguiente *action(NOMBRE(PARAMETROS), LISTAPRE, LISTAPOST)*. Donde *NOMBRE* representa al nombre de la acción *PARAMETROS* son parámetros asociados a la acción, *LISTAPRE* es la lista de Precondiciones de la acción y por último, *LISTAPOST* es la lista de efectos de la acción. Siguiendo el ejemplo del cliente de webmail, nos podríamos encontrar con las siguientes acciones:

```
action (escribirMail(aMail), [], [cuerpoMail(aMail)])
action (escribirSubject(aMail), [], [subjectMail(aMail)])
action (escribirDestinatario(aMail), [], [destinatarioMail(aMail)])
action (buscarDestinatario(aMail), [], [destinatarioMail(aMail)])
action (enviarMail(aMail),
        [cuerpoMail(aMail), subjectMail(aMail), destinatarioMail(aMail)],
        [enviadoMail(aMail), not(cuerpoMail(aMail)), not(subjectMail(aMail)),
        not(destinatarioMail(aMail))] ).
action (enviarMailCorto(aMail),
        [cuerpsubjectMail(aMail), destinatarioMail(aMail)],
        [enviadoMail(aMail), not(subjectMail(aMail)), not(destinatarioMail(aMail))])
```

En el ejemplo anterior pueden apreciarse distintos tipos de acciones. Por ejemplo, la acción *escribirMail*, no posee precondiciones, con esto se establece que en cualquier punto el usuario podría elegir escribir el mail. Lo que produce dicha acción es el cuerpo del mail escrito, representado mediante el hecho *cuerpoMail(aMail)*. Algo similar ocurre con *escribirSubject* y *escribirDestinatario*. Sin embargo la acción *enviarMail*, solicita que esté escrito el cuerpo, el destinatario y el subject, el efecto es el mail enviado y que no existen más las partes del mail. Por otro lado la acción *enviarMailCorto* es similar a la anterior, sólo que no pide que se haya escrito el cuerpo del mail.

Este ejemplo nos permite apreciar cómo es posible definir desde el conocimiento del dominio de las operaciones las acciones para el algoritmo de planning.

3.2 Definición de los objetivos

Un algoritmo de planning busca mediante un plan de acciones satisfacer un conjunto de objetivos. Estos objetivos poseen una correlación directa con los objetivos que se desean modelar en el plan corpus. Siguiendo con el ejemplo anterior, si se desea modelar que se desea enviar un mail, la especificación sería *enviadoMail*(xx). Esto nos permitiría obtener una secuencia de acciones que lo satisfaga, por ejemplo *escribirMail, escribirSubject, escribirDestinatario, enviarMail.*

3.3 Definicion del punto de partida

Un problema de planning consta de una 3-upla definida por el estado inicial, el estado final deseado y el conjunto de acciones que se pueden utilizar. El punto que nos queda por definir es el estado inicial, es decir desde donde se parte en la búsqueda de una solución. Para la definición de el estado inicial no es necesario el agregado de más elementos, ya que este estará conformado por el estado al iniciar la búsqueda. Adicionalmente, el estado inicial tambien podría ser vacío, si es que se está partiendo desde "cero". En nuestro ejemplo, podriamos arrancar con el destinatario ya redactado (si se modela una respuesta), con el cuerpo completo(si es una respuesta automatica), etc.

3.4 Creación automática del Plan Corpus

Plan Corpus es un término utilizado para referirse a los datos de entrenamiento consistentes de una lista de **objetivos** y las **acciones** realizadas por los usuarios para alcanzarlos. En nuestro ejemplo del cliente de mail, el objetivo es modelar el envío de un mail y algunas de las acciones podrían ser las mencionadas en las secciones previas. De la correctitud y completitud de este conjunto de datos dependerá la calidad de la biblioteca de planes que se genere.

Para la creación automática del Plan Corpus será utilizado el algoritmo de planning Ag-Ucpop [19]. Un algoritmo de planning nos permite obtener un conjunto de acciones, que al ser ejecutadas en orden permiten alcanzar un conjunto de objetivos dados [10]. En nuestro caso particular el objetivo será alguno de los identificados en el proceso resultante de la sección 3.2, las acciones que podrá disponer el planner a fin de alcanzar el objetivo estarán compuestas de aquellas resultantes del proceso especificado en la sección 3.1, el plan resultante será una secuencia de acciones que permite obtener el objetivo planteado.

3.5 Selección del algoritmo de planning

A la hora de seleccionar un algoritmo de planning se enumeraron cuáles eran las características que éste debería poseer a fin de poder generar el plan corpus. A partir de un análisis previo surgieron las siguientes características:

a) **Generación de N resultados posibles**: Uno de los puntos clave con los que se desea contar es la posibilidad de generar N soluciones a un problema, es decir N planes que permitan satisfacer un objetivo.
b) **Naturaleza recursiva e incremental**: Un aspecto clave en la utilización del algoritmo es que como estado inicial sea posible especificar tanto un plan

inicial vacío o un plan mucho más complejo, y por ende modelar las secuencias que partan del mismo y que satisfagan el objetivo.

c) **Planning de orden Parcial**: Un plan de orden parcial sólo provee las restricciones mínimas de orden, es decir que es posible obtener varios planes de orden total que satisfagan las restricciones de orden impuestas por el plan inicial. Con esto, al alcanzar un plan, se pueden obtener muchas secuencias de acciones válidas.

d) **Especificación de preferencias**: Con las preferencias se puede modelar modos de actuar particulares que posee el usuario que se desea modelar. En este punto es necesario aclarar que con esto se modela el usuario que se está representando mediante las secuencias de acciones, ya que un usuario experto, por ejemplo, tendrá ciertas preferencias por el uso de algunas acciones que un novato no. Con este conocimiento simple del dominio se pueden obtener secuencias de acciones acorde a los usuarios que las generarían. El uso de conocimiento del dominio permitiría obtener secuencias que se adecuarían a la realidad del sistema, y no solo a las posibles combinaciones aleatorias del mismo.

e) **Especificación de restricciones**: En este caso el algoritmo debiera permitir restricciones sobre las secuencias de acciones que se obtendrán del mismo. De esta forma es posible especificar, por ejemplo que no se puede elegir la acción *buscarDestinatario*, o algo mucho más complejo como que no es válido una determinada combinación de acciones: realizar el *attach* de archivos adjuntos más de 3 veces, etc. Las restricciones son otra forma simple de imponer reglas del dominio al plan corpus que se desea obtener.

La elección del algoritmo Ag-Ucpop se debe a la simplicidad que presenta para obtener todos aquellos planes necesarios para alcanzar un objetivo en particular. El algoritmo Ag-Ucpop es una extensión del algoritmo de orden parcial UCPOP [20] con el agregado de preferencias y restricciones. La forma de utilización del algoritmo es la siguiente: mediante la definición de un estado final (un objetivo a alcanzar), se obtienen un plan que permite alcanzar dicho estado partiendo de un estado inicial definido.

Para poder adecuarnos a los requerimientos se extendió el algoritmo de planning a fin de poder buscar un conjunto de soluciones posibles, y no sólo la primera. También se hizo una extensión a fin de poder obtener de una solución de orden parcial, todos los distintos planes de orden total.

4 Caso de estudio

A continuación se detalla cómo se aplicó la técnica planteada para obtener el plan corpus correspondiente al uso de una herramienta CASE (Computer Aided Software Engineering) y la detección de patrones de diseño.

Un patrón de diseño describe una estructura recurrente de componentes que se comunican para resolver un problema general de diseño en un contexto particular. Nomina, abstrae e identifica los aspectos claves de una estructura de diseño común, lo que los hace útiles para crear un diseño orientado a objetos reutilizable. Identifica las clases e instancias participantes, sus roles y colaboraciones y la distribución de responsabilidades [21].

4.1 Definicion de las acciones acorde a la herramienta CASE

Para definir las acciones, se procedió primero a identificar las operaciones que se realizaban dentro de la herramienta. Para esto se restringió el uso a la creación de un diagrama de clases, esto teniendo en cuenta que lo que se deseaba modelar era la deteccion de Patrones de diseño. Los operadores identificados fueron los presentados en la Tabla 1.

Tabla 1. Operaciones consideradas junto a una breve descripción de las mismas.-

Operación	*Descripción*
AgrInterfaz	Agrega una Interfaz al diagrama
AgrClassAbs	Agrega una Clase abstracta al diagrama
AgrClass	Agrega una clase concreta al diagrama
DefAgregacion	Establece una relación de agregación entre dos elementos del diagrama
DefComposicion	Establece una relación de composición entre dos elementos del diagrama
DefGeneralizacion	Establece una relación de generalización entre dos clases del diagrama
DefAsociacion	Establece una relación de asociación entre dos elementos del diagrama
DefDependencia	Establece una relación de dependencia entre dos clases del diagrama
DefImplementacion	Establece una relación de Implementación entre duna clase y una interfaz

En la Figura 3 se puede apreciar como quedan definidas las distintas acciones para cada una de las operaciones identificadas. En el ejemplo se puede apreciar como la acción "Agregar Interfaz" (agrInterfaz), no posee precondiciones y como efecto agrega una Interfaz *I*. En el caso de definir una composición se generaron dos acciones, una para la definición entre una clase y una interfaz, y la otra entre dos clases (en esta también se solicitan que sean distintas), el efecto en ambos casos es el mismo: la definición de una relación de composición entre los dos elementos, definido como *composicion(A,B)*. La acción "Definir Generalización" (defGeneralizacion) es la que se encarga de establecer una relación que comunmente conocemos como herencia, es por esto que la misma solicita que existan las dos clases, y que la clase que "hereda" se encuentre disponible, es decir que no herede de otra clase, esto representado con *libre(A)*; los efectos de la acción son que la clase A hereda de la clase B y que no se encuentra mas disponible.

```
action( AgrInterfaz( I ),  [], [interfaz( I )] ).
action( AgrClaseAbs ( C ), [], [clase( C ), tipo(C, abstracta), libre( C)] ).
action( AgrClase( C ),  [], [clase( C ), tipo(C, concreta), libre( C )] ).
action( defComposicionC(A, B ),  [clase( A ), clase(B), distintas(A,B)], [composicion(A,B)] ).
action( defComposicionI(A, B ),  [clase( A ), interfaz(B)], [composicion(A,B)] ).
action( defGeneralizacion(A, B ), [clase( A ), libre( A ), clase( B ), distintas(A,B)], [generalizacion(A,B), not (libre( A) )] ).
action( defAssociacion(A, B ), [clase( A ), clase( B )], [associacion(A,B)] ).
action( defDependencia(A, B ), [clase( A ), clase( B ), distintas(A,B)], [dependencia(A,B)] ).
action( defImplementacion(A, B ), [clase( A ), interfaz( B )], [implementacion(A,B)] ).
action( defAgregacion(A, B ), [clase( A ), clase( B )], [agregacion(A,B)] ).
```

Fig. 3. Definicion de acciones de dominio.-

4.2 Definicion del estado final

Definidas las acciones que compondrán el problema de planning resta definir el estado final. En nuestro caso de estudio el estado final lo determinará un patrón de diseño dado. De esta forma por cada patrón de diseño tendremos un estado final, y con éste un conjunto de planes que permita alcanzarlo. En la Figura 4 puede observarse un ejemplo de uno de los posibles estados finales.

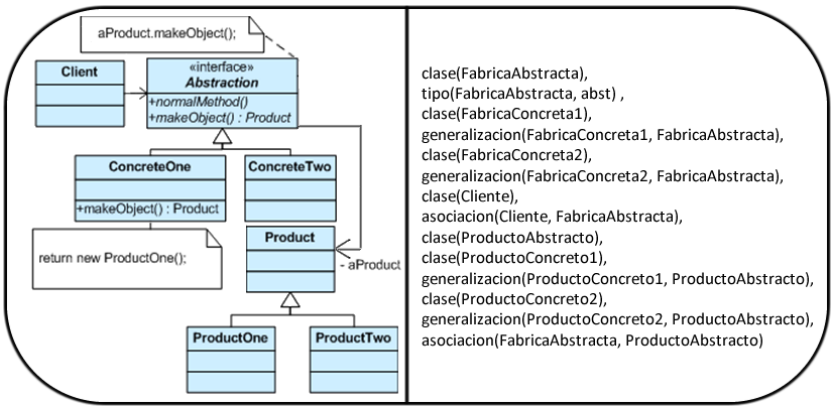

Fig. 4. Diagrama de clases correspondiente a una posible instanciación del patrón Factory Method [21], junto a la especificación de este como estado final.

4.3 Definición del estado inicial

En este caso particular se trabajó con un estado inicial vacío. De esta forma se modela que el usuario comienza a crear el diagrama. Sin embargo, como se mencionó anteriormente, el trabajo propuesto permite partir de cualquier estado inicial. La especificación del mismo sólo incorporaría una descripción del estado de manera similar a lo realizado para el objetivo del plan.

4.4 Generación del plan corpus

Una vez definida la 3-upla que conforma el problema de planning, se procede a la ejecución del algoritmo a fin de obtener el plan corpus deseado para el estado final planteado. Como se mencionó, el resultado que se obtiene de la ejecución del algoritmo consta de un conjunto de acciones que se relacionan para alcanzar el objetivo dado. En la Figura 5 se muestra un ejemplo de un posible plan resultante, en este caso el ejemplo utilizado corresponde al patrón Composite [21]. Como se puede observar, existen acciones que se encuentran en un mismo nivel, es decir que en este punto es posible obtener varios planes de orden total que cumplirán con todas las restricciones de orden.

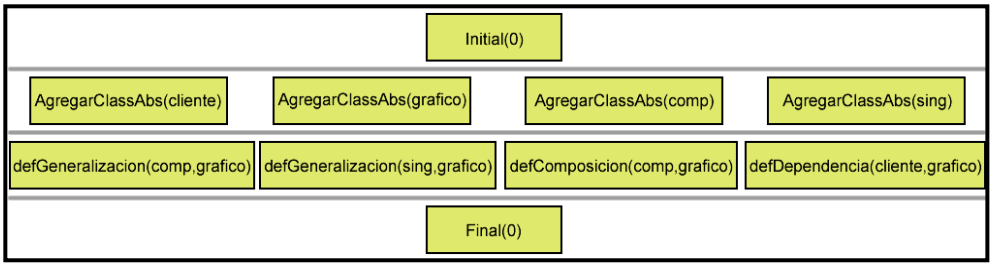

Fig. 5. Captura de un plan resultante del algoritmo.

Del plan de orden parcial presentado en la Figura 5, es posible extraer más de 576 planes que surgen de la combinación simple de orden por cada uno de los niveles. El ejemplo anterior es sólo una solución para el patrón dado, recordemos que el algoritmo se adaptó para obtener más de una solución.

Para el armado final de un conjunto de secuencias que permitan alcanzar un patrón dado, se extraen las 4 primeras soluciones que otorga el algoritmo y por cada una de ellas se extraen los diferentes planes de orden total. Dicho mecanismo se repite para cada uno de los patrones, logrando obtener de manera automática un plan corpus correcto y completo, lo cual nos asegura la calidad del modelo a crear.

Tabla 2. Datos sobre el plan corpus generado tomando 4 planes.-

Patrones de Diseño	*Acciones Promedio*	*Numero de clases de Acciones*	*Acciones completas distintas Promedio*	*Planes Generados*
Abstract Factory	24	4	23	913
Adapter	7	4	6	643
Bridge	16	5	16	829
Command	16	7	16	882
Composite	9	4	8	688
Decorator	13	4	12	846
Factory Method	13	3	13	751
Iterator	20	5	19	875
Mediator	11	2	11	900
Observer	16	7	16	881
Prototype	10	4	9	800

La Tabla 2 presenta un resumen respecto de diferentes aspectos estadísticos de los planes obtenidos. La cantidad de acciones de un plan depende de la complejidad del patrón. La última columna muestra la cantidad de planes diferentes que se obtienen a partir de las primeras 4 soluciones alcanzadas. Cada uno de ellos representa una forma distinta de alcanzar el patrón de diseño objetivo.

Para el dominio presentado algunos posibles conjuntos de preferencias son

```
Preference(addClase(C), 10, 100) :- tipo(C, concreta).
Preference(addClase(C), 5, 100) :- tipo(C, abstracta).
```

En el caso anterior se muestra un ejemplo en donde se indica que existe una preferencia por agregar clases concretas antes que clases abstractas.

```
Preference(defComposicion(A,B), 10, 100): clase(A), clase(B).
Preference(defComposicion(A,B), 1, 100): clase(A), interfaz(B).
Preference(defAsociacion(A,B), 5, 100).
```

En el ejemplo se puede apreciar cómo se establece una preferencia de peso 10 para la definición de una composición entre dos elementos si son clases. En el caso que B sea una interfaz, el peso de la preferencia es 1. Asimismo, definir una asociación tiene un peso de 5. El planner ante esta disyuntiva, primero intentará definir una composición entre dos clases. En caso de fallar tratará de definir una asociación y por último probará definir una composición entre una clase y una interfaz. Una descripción completa del formato de las preferencias puede encontrarse en [19].

Es importante destacar que los resultados obtenidos por esta técnica fueron utilizados en [27] para el entrenamiento de un reconocedor de planes, demostrando que el copus obtenido es adecuado para el reconocimiento de planes y para la tarea de reconocimiento de acciones de un usuario.

5 Trabajos Relacionados

Obtener secuencias de entrenamiento para un sistema de reconocimiento de planes es una tarea difícil. Existen algunas colecciones de este tipo de secuencias tales como el corpus de más de 168.000 comandos de Unix creado por Davison y Hirsh [14] por medio de la observación de 77 usuarios por un periodo de 2-6 meses. Sin embargo, este tipo de colecciones que no tienen pre-identificado a qué objetivo del usuario corresponde cada secuencia de observaciones no son de utilidad directa para el entrenamiento de un sistema de reconocimiento de planes, ya que sería necesario primero aplicar alguna técnica de agrupamiento que permita identificar los diferentes objetivos que representan las secuencias observadas.

Existen muy pocos datasets de secuencias de entrenamientos etiquetadas por objetivo. Albretch et al. [22] por ejemplo, construyeron un plan corpus a partir de los logs de un juego multiusuario orientado a objetivos. Este log incluye la secuencia de las ubicaciones del usuario dentro del ambiente del juego y el comando ejecutado en cada ubicación. Por otro lado, existen las colecciones de comandos Unix de Lesh [23] y de commandos Linux de Blaylock y Allen [24]. Ambos corpus fueron creados utilizando experimentos más controlados que el de Albretch et al., en los cuales se les dio a un conjunto de sujetos diferentes narrativas de objetivos a cumplir mediante la ejecución de comandos. La secuencia de comandos ejecutada era registrada por el sistema y al finalizar cada sesión se le preguntó a los sujetos si habían completado satisfactoriamente el objetivo dado. En base a los logs creados a partir de estas sesiones se creó un plan corpus más confiable dado que el objetivo del usuario era conocido de antemano.

En cuanto a la creación automática de plan corpus etiquetado por objetivos, los autores sólo tienen conocimiento del enfoque propuesto por Blaylock y Allen [25]. Estos autores

modificaron SHOP2 [26], un planner jerárquico de transición de red (Hierarchical Transition Network, HTN) completo y sólido, con el objetivo de generar aleatoriamente un plan para un determinado objetivo y estado inicial de manera no determinística. Una de las diferencias de esta propuesta con la nuestra consiste en que el planner jerárquico utilizado provee una solución de orden total. Esto implica que la solución obtenida por cada corrida del algoritmo es única, y se debe luego re evaluar el planner para obtener más soluciones. En nuestro caso, la solución de orden parcial obtenida puede implicar más de una solución de orden total, permitiendo obtener un número mayor de planes. Otra de las diferencias que se destaca es la naturaleza recursiva del planner elegido en nuestro caso, que permite partir de un estado inicial vacío o con elementos disponibles, logrando una mejor adaptación al problema de dominio. La posibilidad de especificar preferencias es otra diferencia importante ya que se permite al experto modelar no solo el problema sino usuarios con diferentes características.

6 Conclusiones

En el presente trabajo, se propuso una técnica que permite obtener un plan corpus de una manera computacional a partir de una especificación del dominio, utilizando para esto un algoritmo de planning inteligente de orden parcial. A lo largo del trabajo se puede apreciar cómo especificando el conocimiento del dominio es posible obtener un plan corpus de manera automática. La técnica planteada permite obtener resultados significativos minimizando el error humano que proviene de la gestión manual del plan corpus, e incluso desafectando el proceso de observar diferentes usuarios.

Como trabajo futuro queda pendiente la generación de una base de planes para diferentes dominios y entrenar con éstas diferentes reconocedores de planes para poder analizar los resultados. En este sentido se hicieron algunos experimentos con el plan corpus generado a partir del ejemplo presentado en este artículo, obteniendo resultados prometedores [27].

7 Bibliografía

1. Kautz, H. A formal theory of plan recognition. Ph. D thesis, Department of Computer Science, University of Rochester. 1987.
2. Kautz, H. A formal theory of plan recognition and its implementation. En: Allen, J. F., Kautz, H. A., Pelavin, R., and Tenenberg, J., editors, Reasoning About Plans, Pág. 69-125. Morgan Kaufmann Publishers, San Mateo (CA), USA. 1991.
3. Charniak E., Goldman R. A probabilistic model of plan recognition. En: National conference on artificial intelligence. AAAI-91. 1991.
4. Horvitz, E., Breese, J., Heckerman, D., Hovel, D., and Rommelse, K. The Lumière project: Bayesian user modeling for inferring the goals and needs of software users. In Cooper, G. F. and Moral, S., editors, Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, Pág. 256-265, San Mateo. Morgan Kaufmann. 1998.
5. Lesh N, Rich CSC. Using plan recognition in human-computer collaboration. En: Seventh international conference of user modeling, Banff, Canada. 1999.
6. Goldman, R., Geib, C., and Miller, C. Learning hierarchical task models by defining and redefining examples. En: Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99), Pág. 245-254, San Francisco, CA. Morgan Kaufmann. 1999.

7. Oliver, N., Horvitz, E., and Garg, A. Layered representations for human activity recognition. En: Proceedings of the 4th IEEE International Conference on Multimodal Interfaces (ICMI 2002), Pág. 3-8. IEEE Computer Society. 2002.
8. Bui, H. H. A general model for online probabilistic plan recognition. En: Gottlob, G. and Walsh, T., editors, IJCAI 03, Proceedings of the 8th International Joint Conference on Artificial Intelligence, Pág. 1309- 1318, Acapulco, Mexico. Morgan Kaufmann. 2003.
9. Liao, L., Patterson, D. J., Fox, D., and Kautz, H. A. Learning and inferring transportation routines. Artificial Intelligence, 171 (5-6) pag. 311-331. 2007.
10. Philipose, M., Fishkin, K. P., Perkowitz, M., Patterson, D. J., Fox, D., Kautz, H., and Hahnel, D. Inferring activities from interactions with objects. Pervasive Computing Magazine, 3(4) pages 10-17. 2004.
11. Nguyen, N. T., Phung, D. Q., Venkatesh, S., and Bui, H. H. Learning and detecting activities from movement trajectories using the hierarchical hidden Markov model. En: IEEE Computer Vision and Pattern Recognition or CVPR, Pág. 955-960. IEEE Computer Society. 2005.
12. Duong, T. V., Phung, D. Q., Bui, H. H., and Venkatesh, S. Human behavior recognition with generic exponential family duration modeling in the hidden semi-markov model. En: International Conference on Pattern Recognition, 3 Pág. 202-207. 2006.
13. Bauer, M.: Acquisition of abstract plan descriptions for plan recognition. En: Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98), Madison, WI. Pág. 936-941. 1998.
14. Davison, B.D., Hirsh, H.: Predicting sequences of user actions. En: Notes of the AAAI/ICML 1998 Workshop on Predicting the Future: AI Approaches to Time Series Analysis, Madison, Wisconsin. 1998.
15. Bauer, M. From interaction data to plan libraries: A clustering approach. In IJCAI '99: Proceedings of the 6th International Joint Conference on Artificial Intelligence, Pág. 962-967, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. 1999.
16. Gorniak P. and Poole, D. Building a stochastic dynamic model of application. In Boutilier, C. and Goldszmidt, M., editors, 6th Conference on Uncertainty in Artificial Intelligence (UAI-2000), pages 230-237, Stanford University, Stanford, California, USA. Morgan Kaufmann. 2000.
17. Garland A. and Lesh N. Learning hierarchical task models by demonstration. En: Technical report, Mitsubishi Electric Research Laboratories. 2002.
18. Martín G. Marchetta, Raymundo Forradellas: Supporting Interleaved Plans in Learning Hierarchical Plan Libraries for Plan Recognition. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial 10(32) Pág. 47-56. 2006
19. Luis Berdun, A. Amandi. Planning para agentes inteligentes. En: Proceedings del 7º Simposio Argentino de Inteligencia Artificial (ASAI 2005), Pág. 12-23, Rosario, Argentina. 2005.
20. Weld, D.S. An Introduction to Least Commitment Planning. En: AI Magazine, 15(4) Pág. 27-61. 1994.
21. E. Gamma, R. Helm, R. Johnson and J. Vlissides, Design patterns, elements of reusable object-oriented software, Addison-Wesley. 1994.
22. Albrecht, D.W., Zukerman, I., Nicholson, A.E.: Bayesian models forkey hole plan recognition in an adventure game. User Modeling and User-Adapted Interaction 8 Pág. 5–4. 1998.
23. Lesh, N.: Scalable and Adaptive Goal Recognition. PhD thesis, University of Washington (1998)
24. Nate Blaylock and James Allen. Corpus-based, statistical goal recognition. Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-2003), pages 1303-1308, Acapulco, Mexico, August Pág. 9-15. 2003.
25. Nate Blaylock and James Allen. Generating artificial corpora for plan recognition. In Liliana Ardissono, Paul Brna, and Antonija Mitrovic, editors, User Modeling 2005, number 3538 in Lecture Notes in Artificial Intelligence, Pág. 179-188. Springer, Edinburgh, Julypag. 24-29. 2005.
26. Nau, D., Au, T.C., Ilghami, O., Kuter, U., Murdock, J.W., Wu, D., Yaman, F.: SHOP2: An HTN planning system. Journal of Artificial Intelligence Research 20, Pág. 379–404. 2003.
27. Juan Francisco Silva Logroño, Luis Berdun, Marcelo Armentano y A. Amandi. Análisis de secuencias discretas para la detección de Patrones de Diseño de Software. En: Proceedings del 11º Simposio Argentino de Inteligencia Artificial (ASAI 2010), Buenos Aires, Argentina. 2010.