Inteligencia Artificial 47(2010), 54-70
doi: 10.4114/ia.v14i47.1575

INTELIGENCIA ARTIFICIAL

http://erevista.aepia.org/

Sistema basado en reglas para la validación del despliegue de servicios

Laura Díaz-Casillas, Fco. Javier Blanco, Mercedes Garijo

Departamento de Ingeniería de Sistemas Telemáticos
Universidad Politécnica de Madrid
Madrid, España
{ldcasillas, fcojavibr, mga}@dit.upm.es

Resumen El despliegue de servicios es una tarea compleja, al depender de múltiples factores que pueden variar en el tiempo, que se realiza normalmente de forma manual. En este artículo se propone emplear un sistema de validación de planes de despliegue, implementado a través de un sistema basado en reglas, para asegurar que las acciones del despliegue a realizar son adecuadas al estado del entorno sobre el que se desea actuar. Dicho sistema se encuentra integrado dentro de un gestor de cambios de entorno y basado en su modelo de información.

Palabras clave: validación, sistema basado en reglas, despliegue, servicios, OSGi.

1. Introducción

Numerosas aplicaciones empresariales desarrolladas hoy en día se caracterizan por presentar una arquitectura orientada a servicios (SOA, del inglés *Service Oriented Architecture*) [15]. Su uso facilita la interoperabilidad entre sistemas heterogéneos, al definir una manera estándar de anunciar e invocar servicios que actúan de manera independiente. Además, su empleo favorece la fiabilidad, mantenibilidad y escalabilidad de dichas aplicaciones.

Una de las actividades a llevar a cabo durante el desarrollo de este tipo de aplicaciones es el despliegue de los servicios, que consiste en realizar todas las acciones necesarias para poder poner dichos servicios en funcionamiento. Es habitual que los servicios cooperen entre sí, estableciéndose dependencias entre ellos en función de los recursos ofertados y los requisitos demandados, siendo necesario comprobar su disponibilidad. Por otro lado, es necesario tener en cuenta las características de los entornos sobre los que se realiza el despliegue. Dichos entornos son normalmente heterogéneos, estando constituidos por componentes de diversa índole, y distribuidos, es decir, los servicios suelen ejecutarse sobre distintos nodos que conforman una red. Por otro lado, se debe tener en cuenta que durante el desarrollo de las aplicaciones empresariales se suelen utilizar varios entornos de despliegue, para de esta forma, adecuarse a las necesidades concretas de cada fase del proyecto.

Todo ello conduce a que el despliegue sea una actividad compleja que, sin embargo, se realiza normalmente de forma manual, siendo una de las principales fuentes de error durante el desarrollo de los sistemas, lo que implica un aumento en el tiempo requerido para su puesta en funcionamiento y en el coste asociado. Este hecho ha dado lugar a la investigación en este campo, con el objetivo de facilitar la realización de dicha tarea.

ISSN: 1988-3064(on-line)

La inteligencia artificial ofrece diversas técnicas que pueden ayudar a resolver esta situación, como el uso de sistemas de representación del conocimiento, el empleo de métodos de aprendizaje automático o la utilización de sistemas basados en reglas, entre otros [23].

Su aplicación a la ingeniería del software [2] se realiza desde hace muchos años, pues ésta es una actividad intensiva en conocimiento, que requiere tanto el entendimiento del dominio de la aplicación como del software en sí mismo, siendo de gran utilidad el empleo de técnicas de inteligencia artificial para simplificar y facilitar el desarrollo de los sistemas. Aún así, ésta es un área en evolución, pues a pesar de haberse realizado mucha investigación al respecto, todavía es necesario implementar aplicaciones prácticas en las que se pongan de manifiesto los beneficios del empleo de dichas técnicas en el ámbito de la ingeniería del software.

Briand [6] agrupa en cuatro categorías las principales áreas relacionadas con la ingeniera software que pueden beneficiarse de la aplicación de tecnologías enmarcadas dentro del ámbito de la inteligencia artificial: planificación, seguimiento y control de calidad de los proyectos; mejora de los procesos de las organizaciones de software; apoyo en la toma de decisiones; y automatización.

En la actualidad, una de las principales aplicaciones de la inteligencia artificial en este campo se basa en el empleo de sistemas expertos, pues su uso permite automatizar las tareas, con lo que se pretende reducir el número de errores derivados del desarrollo del software. El objetivo es por tanto asegurar dicho proceso, reduciendo los riesgos asociados [27] y garantizando la calidad del producto implementado [22]. Pero la elaboración de sistemas expertos no es una actividad sencilla, pues requiere conocer la información asociada al dominio de la aplicación y emplearla de manera adecuada, lo que implica la colaboración de conocedores del dominio junto con desarrolladores de software, siendo necesario verificar si el producto obtenido es el adecuado [30].

El uso de inteligencia artificial permite la automatización de tareas tan complejas como el despliegue de servicios en entornos extensos empresariales bajo una arquitectura SOA. En ingeniería software, dichas tareas suelen llevar asociado un modelo del dominio que permite su ejecución en base a planes de despliegue para facilitar su correcta organización y temporización y situar sus dependencias de una forma simple y clara. Bajo esta metódica se encuentran algunos sistemas que usan inteligencia artificial para llevar a cabo la validación de dichos planes e incluso del modelo asociado [11] [16] [24].

En este artículo se propone emplear un sistema de validación de planes de despliegue con el objetivo de informar al usuario acerca de la viabilidad y las repercusiones de la ejecución de las actividades a realizar en el futuro proceso de despliegue según el estado actual del entorno sobre el que se va a trabajar. Las restricciones a considerar durante la validación se han definido a través de un sistema de reglas, facilitando su organización y visualización mediante un lenguaje más natural, su adaptación a las necesidades de cada caso y su actualización futura por parte de expertos del dominio.

El sistema se encuentra integrado en un gestor de cambios de entorno, desarrollado dentro del proyecto de investigación ITECBAN (Infraestructura Tecnológica y Metodológica de Soporte para un Core Bancario), incluido en el programa CENIT de I+D, llevado a cabo por el Gobierno español.

El resto del artículo se estructura de la siguiente manera. En la sección 2 se realiza una introducción al contexto en el que ha sido desarrollado el sistema de validación, explicándose cómo se lleva a cabo la gestión del despliegue dentro del proyecto ITECBAN, para a continuación expresar los objetivos a alcanzar con el desarrollo del sistema. En la sección 3 se realiza un breve resumen de las tecnologías empleadas en la solución propuesta, descrita en detalle en la sección 4. En la sección 5 se muestran algunas pruebas de concepto realizadas. Posteriormente, en la sección 6, se presentan algunos trabajos relacionados, y por último, en la sección 7, se exponen las conclusiones y líneas futuras de trabajo.

2. Gestión del despliegue de servicios en el proyecto ITECBAN

El principal objetivo del proyecto ITECBAN es el desarrollo de una plataforma que sirva como base para la creación de sistemas de gestión destinados al sector bancario, eliminándose las limitaciones actuales de los sistemas de información empleados en entornos financieros. Las aplicaciones empresariales empleadas en este tipo de áreas se caracterizan por presentar en la mayor parte de los casos una arquitectura orientada a servicios, lo que las permite adaptarse de manera continua y flexible a los cambios que se producen en su alrededor, en respuesta a la demanda del mercado. Durante el desarrollo de este tipo

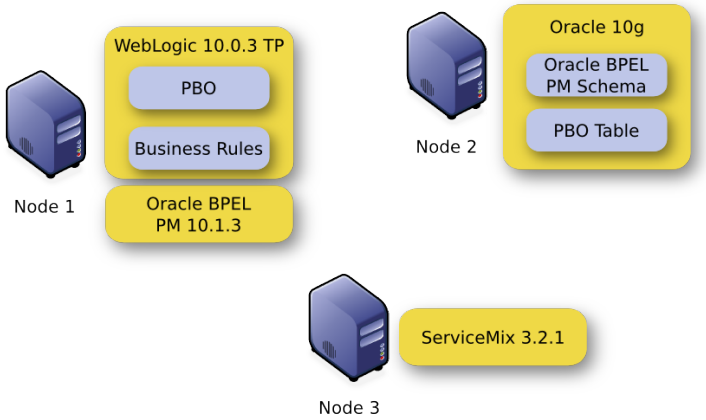

Figura 1: Escenario de despliegue

de aplicaciones es necesario el empleo de varios entornos de despliegue, a través de los cuales se evoluciona hasta alcanzar el entorno de producción. Los motivos que conducen a utilizar distintos entornos son diversos, entre ellos destacan el control de acceso, el cual debe ir aumentando a medida que se avanza en el proceso de desarrollo, y la topología del entorno, la cual va cambiando a lo largo del ciclo de desarrollo, haciéndose más compleja en la fase de producción, cuando se requiere ampliar el número de recursos físicos empleados, siendo necesario utilizar sistemas de respaldo. A esta situación se añade el hecho de que dichos entornos sean heterogéneos, distribuidos y muy dinámicos, con constantes cambios en el estado, disponibilidad y configuración de los servicios que los pueblan. Surge entonces la necesidad de disponer de una solución que facilite el despliegue de los servicios, adecuándolos a las necesidades y características de cada entorno en concreto, lo que ha llevado a establecer, como una de las tareas a realizar dentro del proyecto ITECBAN, la creación de un gestor de cambios de entorno. El objetivo de dicho sistema es la creación de procedimientos y el soporte a las operaciones de despliegue y configuración de unidades software sobre los distintos entornos gestionados por la organización. Este sistema ha sido implementado dentro de la arquitectura de despliegue y configuración de ITECBAN (de ahora en adelante ADCI).

El gestor de entornos emplea un modelo de recursos presentado en [9], basado en CIM (*Common Information Model*) [10] y en el estándar de componentes y despliegue de sistemas distribuidos establecido por OMG (*Object Management Group*) [25]. En concreto, el modelo de información empleado se divide en tres modelos específicos, dedicados a la representación de los entornos de despliegue (*Deployment Target*), las unidades software o servicios a desplegar (*Deployment Unit*) y los planes de despliegue de unidades, configuración y gestión de recursos (*Deployment Plan*).

Un entorno se encuentra formado por diversos nodos distribuidos a lo largo de una red. Cada nodo suele representar una máquina específica, aunque también puede modelar nodos virtualizados o que formen parte de un *cluster* ubicado sobre una máquina, y presenta una serie de recursos con ciertas propiedades que lo caracterizan. Los nodos albergan contenedores, elementos software que contienen a las unidades software. Existen diversos tipos de contenedores según despliegen un tipo u otro de unidades, así por ejemplo existen servidores de aplicaciones o bases de datos. Los contenedores también presentan propiedades que los caracterizan y pueden ser configuradas. Las unidades software representan los servicios y aplicaciones a ser ejecutados. Las unidades suelen exportar recursos, que pueden ser requeridos por otras unidades, lo que se representa mediante dependencias, y establecer una serie de restricciones que deben ser cumplidas por el contenedor sobre el que se realizará el despliegue. Una vez desplegada, cada unidad pasa a modelarse como una unidad en ejecución, que contiene la configuración específica del despliegue realizado, con las asociaciones existentes entre unidades y los valores ya configurados de sus propiedades. En la figura 1 se muestra un escenario de ejemplo, en el que el entorno presenta tres nodos, sobre cada uno de ellos se encuentran desplegados uno o más contenedores, albergando distintos servicios o unidades software.

Los planes de despliegue contienen las actividades a realizar sobre las unidades software en un entorno

concreto. El gestor de cambios de entorno debe ser responsable de que los planes generados sean válidos, evitando por ejemplo que se intenten desplegar unidades sobre nodos sin los recursos requeridos. Para ello, debe analizar las características de cada unidad a desplegar, considerando sus dependencias y restricciones, y el estado del entorno sobre el que se desea llevar a cabo el despliegue. Además, es posible que desde la creación de un plan hasta su ejecución, el estado del entorno varíe, siendo necesario comprobar que las operaciones contenidas en las actividades se pueden cursar, debido a que todos los requisitos y dependencias previos se siguen cumpliendo y que además no entran en conflicto con las unidades desplegadas actualmente en el entorno. Con el objetivo de automatizar dicha tarea, se propone integrar dentro del gestor de cambios de entorno un sistema de validación de planes de despliegue.

El propósito de dicho sistema es asegurar el correcto funcionamiento de los planes antes de llevarlos a cabo. Para ello, debe contrastar la información contenida en el plan con el estado actual del entorno sobre el que se desea realizar el despliegue, verificando que la ejecución de las actividades contenidas en el plan coincide con el esperado inicialmente. El resultado de la validación es mostrado al usuario, en concreto, existen tres tipos posibles de respuestas en las que se indica que: se puede continuar con el plan, se advierte de los posibles problemas que pudieran surgir o de que el plan debe abortarse debido a que existe algún problema grave. El objetivo perseguido es por tanto facilitar al usuario la realización de la acción más adecuada a llevar a cabo antes de ejecutar un plan de despliegue.

3. Sistemas de razonamiento basados en reglas: Drools

Para la realización del sistema de validación de planes se propone aplicar un sistema basado en conocimiento, es decir, un sistema que trata de resolver los problemas mediante razonamiento. Dado que todo el conocimiento necesario para la validación de los planes es bastante complejo, se ha considerado que el uso de reglas o tablas de decisión es adecuado para su representación, es decir, se ha optado por emplear un sistema de razonamiento basado en reglas.

El empleo de un sistema basado en conocimiento permite al usuario experto en el dominio definir las comprobaciones a realizar durante la validación y las acciones más adecuadas a seguir en cada caso. El motor de inferencia será el encargado de contrastar dicha información, almacenada en la base de conocimiento, con la contenida en el plan de despliegue que se desea ejecutar y el estado actual del entorno sobre el cual se va a realizar el proceso. En este caso, el resultado de dicho proceso será procesado por el gestor de cambios de entorno, encargado de mostrar al usuario la información resultante.

Existen varias opciones mediante las cuales es posible implementar un sistema experto, como son Drools, Jess, Mandaraz y OpenRules. Entre ellas, se ha seleccionado Drools Expert [18], uno de los componentes de Drools, plataforma de integración de lógica de negocio de código abierto de la comunidad JBoss. El motor de inferencia de Drools Expert cuenta con diversas características que han sido relevantes a la hora de desarrollar el sistema:

- Implementación optimizada y mejorada del algoritmo Rete secuencial para sistemas orientados a objetos (ReteOO).
- Mantenimiento de verdad con aserciones lógicas.
- Bases de conocimiento dinámicas, lo que permite añadir o eliminar reglas en tiempo de ejecución.
- Agente de conocimiento.
- Herencia de reglas y posible inclusión de metadatos sobre éstas.
- Declaraciones de tipos específicos.
- Motor de reglas con lógica de predicados de primer orden que permite uso de cuantificadores, restricciones y conectores.

Drools posee un lenguaje específico para la definición de reglas y también permite emplear tablas de decisión para su representación. En concreto, Drools soporta el formato de hojas de cálculo (Microsoft Excel, OpenOffice.org Calc y CSV), con sus características habituales de captura y manipulación de

datos. La formación de reglas a través de los datos insertados en hojas de cálculo es muy sencilla: cada una de las filas representa una regla y las columnas que contiene forman dos grupos, en primer lugar, se encuentran las condiciones que deben cumplirse para que se ejecute la regla, y después, las acciones a realizar. Esta forma de representación es apropiada para la generación de reglas de activación con condiciones y acciones.

4. Sistema de validación de despliegue de servicios

Para poder llevar a cabo el despliegue de una unidad software, el usuario debe seleccionar la unidad a desplegar y el entorno sobre el que se realizará la operación, lo que desencadenará la creación de un plan con todas las actividades necesarias para poder realizar dicha tarea correctamente. Cada actividad contendrá una unidad de despliegue y una operación asociada a realizar sobre un contenedor concreto del entorno elegido. Además, es posible que las actividades tengan dependencias con otras actividades, de tal forma que éstas deban ejecutarse primero para poder realizar la actividad dependiente sin contratiempos.

Sin embargo, puede ocurrir que desde la creación de un plan hasta su ejecución, el estado del entorno varíe, siendo necesario llevar a cabo la comprobación de dicho plan para determinar que no ha quedado obsoleto o contiene acciones que perturben la estabilidad o la correcta configuración del resto de servicios y del entorno en sí. Este es el objetivo del sistema desarrollado, el cual, a partir del plan a ejecutar, analizará el estado actual del entorno seleccionado para su despliegue, mostrando como resultado un informe al usuario de las acciones más adecuadas a realizar.

4.1. Operativa de validación

En principio, la validación tendrá lugar inmediatamente antes de lanzar el plan, por lo que será invocada por el gestor de ejecución tras recibir la orden de ejecutar el plan. Éste invocará al gestor de entornos, el cual, además de la información contenida en el plan de despliegue, recogerá el estado actual del entorno concreto sobre el que se desea ejecutar el plan, para pasársela al sistema de validación. Es posible que la validación sea ordenada por el usuario para comprobar únicamente la validez actual del plan, aunque no se vaya a proceder a su ejecución, en este caso, también será el gestor de ejecución el encargado de iniciar el proceso.

Para poder realizar las comprobaciones requeridas, el sistema deberá acceder al repositorio de despliegue, con el objetivo de conocer la información relativa a las unidades software. La comunicación con dicho sistema se realizará mediante servicios web.

En la figura 2 queda representado el intercambio de información entre los sistemas presentes en la arquitectura.

Para realizar la validación, el sistema debe analizar y validar una a una las actividades contenidas en el plan, ordenadas previamente en función de las dependencias existentes entre ellas. De esta forma, si el resultado de la validación de una es negativo, las actividades dependientes serán invalidadas sin necesidad de ser analizadas. La validación a realizar dependerá del tipo de actividad, las condiciones que deben cumplirse para instalar una unidad correctamente no son las mismas que para su desinstalación, por ejemplo. Diversas causas pueden conducir a que se determine que va a haber un error en la ejecución de una actividad, es posible que alguno de los nodos, contenedores, unidades o recursos no se encuentren disponibles o puede suceder que alguna de las operaciones a realizar se haya ejecutado con anterioridad, entre otros motivos. Esto provocará que en algunos casos sea necesario abortar la actividad, ya que su ejecución es inviable o muy probablemente ocasione fallos graves en el entorno, mientras que en otros será suficiente con advertir al usuario de los posibles conflictos que puede originar su ejecución o informarle de que la acción puede ser realizada correctamente. Si la validación de la actividad genera un resultado positivo, como última fase se simula la operación sobre la unidad software considerada en dicha actividad sobre el estado considerado del entorno, de forma que en la validación de las siguientes actividades se tenga en cuenta que las actividades anteriores ya han sido ejecutadas, considerando así las posibles dependencias existentes entre las actividades.

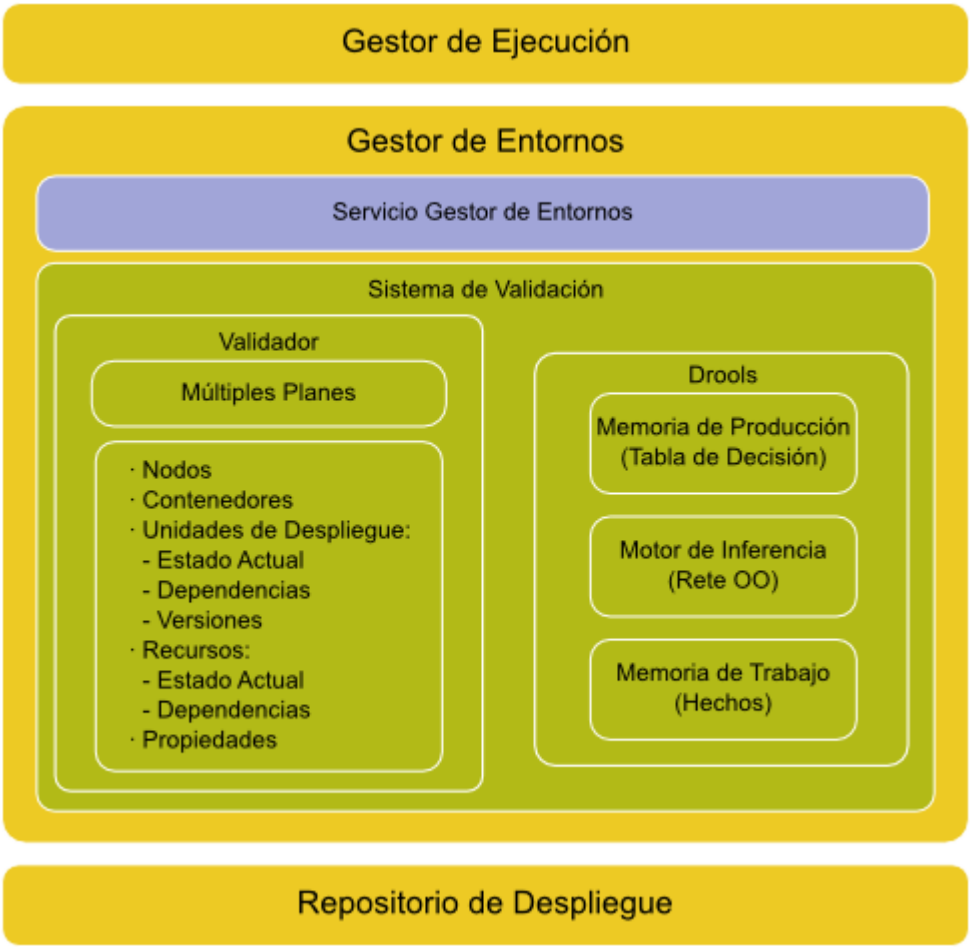

Figura 2: Arquitectura del sistema

4.1.1. Validación de las actividades de un plan

Los planes a validar pueden presentar tres tipos de actividades: despliegue, gestión y configuración de recursos.

- Las actividades de despliegue son responsables de ejecutar una operación sobre una unidad de despliegue. En concreto, estas operaciones pueden ser instalar, arrancar, actualizar, parar o desinstalar dicha unidad. La validación de este tipo de actividades depende de cada operación, aunque en general, después de verificar que el destino es correcto, es decir, que el entorno, el nodo y el contenedor destino se encuentran disponibles, se comprobará si la unidad a desplegar y los recursos y unidades requeridos se encuentran en dicho contenedor y si su estado es el adecuado, además se analizará si existen otras versiones de esa misma unidad.

- Las actividades de gestión de recursos tienen como objetivo añadir o eliminar recursos en un contenedor. Para su validación, tras comprobar que el contenedor sobre el que se desea actuar se encuentra disponible, se analiza si el recurso a añadir no se encuentra actualmente en dicho contenedor o en el caso de que se desee eliminar un recurso, que éste se encuentra presente y no es requerido por ninguna otra unidad desplegada en el entorno.

- Las actividades de configuración de recursos tienen como objetivo la configuración de las propiedades de un contenedor, de los recursos de un contenedor o de los recursos de una unidad desplegada sobre un contenedor. Siendo en este caso necesario comprobar que las propiedades de los elementos sobre

los que se desea actuar no influirán negativamente sobre el resto de componentes desplegados sobre el entorno, en concreto:

- A la hora de configurar las propiedades de un contenedor, tras comprobar que el contenedor se encuentra en el nodo y entorno seleccionados, deben analizarse las propiedades presentes en dicho contenedor y su estado, determinando si se puede realizar o no la actividad.
- En el caso de la configuración de las propiedades de un recurso de un contenedor, después de verificar que el entorno, el contenedor y el recurso objetivo están disponibles, se analiza si dicho recurso está siendo utilizado por alguna unidad. En caso negativo, se comprueba que las propiedades del recurso a configurar se encuentran en un estado distinto al programado en la actividad.
- Si el objetivo de la actividad es configurar las propiedades de un recurso de una unidad desplegada, debe comprobarse que dicha unidad y su recurso se encuentran en el contenedor indicado. Una vez realizada dicha operación, se analiza, al igual que en el caso anterior, que dicho recurso no es requerido por ninguna unidad y que se encuentra en un estado distinto al que se desea alcanzar.

4.1.2. Validación de múltiples planes

Es posible que un usuario desee validar un plan con una fecha planificada de ejecución posterior a la fecha actual. En esta situación, puede ocurrir que desde la fecha actual en la que se está validando el plan, y por lo tanto en la que se recoge el estado del entorno, hasta su ejecución, otros planes sean ejecutados, modificando el estado del entorno. Por ello, se ha decidido contemplar este hecho, y antes de validar un plan, se comprueba su fecha de ejecución para ver si existen otros planes a desplegar sobre el mismo entorno con una fecha planificada de ejecución anterior. En el caso de existir, dichos planes serán validados previamente en orden cronológico para, si se obtiene un resultado positivo, simular la ejecución de sus actividades sobre la foto actual del estado del entorno que se ha recibido. Si por el contrario el resultado de la validación es negativo, se analizarán las dependencias de dicho plan con el resto de planes pendientes de validar. De manera que, si un plan depende de otro que ha obtenido un resultado negativo de la validación, se abortará directamente, sin necesidad de validar una a una las actividades contenidas en él. En concreto, las dependencias consideradas entre planes son:

- Si el plan contiene una actividad de despliegue, se abortarán los planes que presenten:
  - Actividades de despliegue que operen sobre la misma unidad o sobre unidades que dependan de los recursos exportados por la unidad afectada (si las operaciones asociadas son de instalación o arranque).
  - Actividades de gestión de recursos que eliminen recursos de contenedor requeridos por la unidad afectada.
  - Actividades de configuración de recursos que operen sobre recursos exportados por la unidad afectada o requeridos por ella.
- Si el plan invalidado presenta una actividad de gestión de recursos de contenedor, se abortarán los planes que contengan:
  - Actividades de despliegue que operen sobre unidades que dependan del recurso afectado.
  - Actividades de gestión de recursos que actúen sobre dicho recurso.
  - Actividades de configuración de recursos que operen sobre el recurso implicado.
- Si el plan a abortar contiene una actividad de configuración de recursos:
  - Si la actividad es de configuración de propiedades de contenedor, se abortarán los planes que contengan actividades asociadas a dicho contenedor.

- Si la actividad es de configuración de propiedades de recursos de contenedor, se anularán los planes que presenten actividades de despliegue en las que se opere sobre unidades que dependan de los recursos configurados en la actividad abortada y de gestión y configuración de recursos que operen sobre el recurso afectado.
- Si en la actividad se configuran propiedades de recursos de unidades desplegadas, se anularán los planes que contengan actividades de despliegue que actúen sobre dicha unidad o sobre unidades que dependan de los recursos exportados por la unidad afectada, además de aquellos que presenten actividades de configuración de dicho recurso.

4.2. Datos iniciales

El sistema experto parte de la información incluida en el plan de despliegue que se desea validar, es decir, el entorno sobre el cual se llevará a cabo y el conjunto de actividades a realizar. En algunos casos será necesario acceder al gestor de ejecución, para obtener los planes agendados con una fecha de ejecución anterior a la del plan seleccionado, y al repositorio de despliegue, para conocer la información asociada a las unidades de despliegue, pudiendo analizarse entonces las dependencias con otras unidades y las restricciones que deben cumplir los contenedores sobre los que puede desplegarse.

Estos requisitos se tienen en cuenta en la elaboración del plan, pero es necesario comprobar que el estado del entorno no ha variado desde su creación hasta que realmente se lleven a cabo las operaciones incluidas en el plan, siendo por tanto necesario analizar el estado del entorno y comprobar que las condiciones siguen siendo válidas o en caso contrario, recomendar al usuario las acciones más adecuadas a realizar.

4.3. Representación del conocimiento de validación

El conocimiento de validación ha sido expresado mediante reglas que definen las acciones a realizar ante determinadas situaciones a través de una tabla de decisión. Esta permite mostrar claramente las condiciones que determinan el comportamiento a seguir y las operaciones a llevar a cabo en cada caso.

Las columnas de la tabla indican las condiciones bajo las que se cumplen las reglas y las acciones a efectuar en el caso de que dichas condiciones se cumplan. Estas acciones consistirán en ir realizando de forma secuencial las comprobaciones adecuadas para detectar posibles conflictos entre el estado actual del entorno y las actividades contenidas en el plan. Si aparece algún inconveniente que impida la ejecución de alguna actividad, finalizará la comprobación asociada, informando al usuario de la causa del error.

Cada una de las filas de la tabla se corresponde con una regla, éstas pueden clasificarse en varios grupos:

1. Inicialización, inician las comprobaciones sobre el entorno de despliegue.

2. Elementos del entorno, verifican que los nodos y contenedores sobre los que se desea ejecutar una actividad se encuentran disponibles en el entorno.

3. Unidades desplegadas, comprueban si la unidad sobre la que se desea realizar una operación existe actualmente en el entorno. En caso afirmativo y si la operación lo requiere, se analizará su estado para determinar si es posible realizar dicha operación.

4. Unidades necesarias, confirman que se cumplen las dependencias impuestas por la unidad, es decir, que las unidades requeridas se encuentran en el entorno y en el estado adecuado. Por ejemplo, si se desea arrancar una unidad que depende de otra, es necesario verificar que esta otra se encuentra arrancada actualmente.

5. Unidades sub/sobreversionadas, comprueban si existen otras unidades con distinta versión en el entorno de despliegue.

6. Recursos necesarios, comprueban si los recursos sobre los que se desea operar se encuentran o no en el entorno, dependiendo del tipo de actividad a realizar.

7. Recursos requeridos, verifican que las operaciones a realizar sobre los recursos no afectan al funcionamiento de las unidades desplegadas en el entorno.
8. Propiedades, analizan si el estado de las propiedades es adecuado a las operaciones a realizar sobre ellas.
9. Finalización, fin del proceso de validación e inicio del proceso de simulación de la actividad en el entorno.
10. Simulación, determinan la acción a realizar sobre el estado del entorno recibido en función de la actividad validada correctamente.

El orden de ejecución de dichas reglas se corresponde con el orden en el que han sido expuestas, de forma que si por ejemplo el contenedor sobre el que se va a desplegar una unidad no se encuentra en el sistema, no se realizarían más comprobaciones, originándose directamente el informe de respuesta correspondiente.

4.4. Resultados

El resultado de la validación de cada actividad se encuentra compuesto por un mensaje que indica el tipo de respuesta obtenido y un texto informativo, en el cual se explica el motivo de dicha respuesta. En concreto, las posibles respuestas son:

- NONE, indica que el resultado de la validación ha sido positivo y por tanto puede continuarse con la ejecución del plan.
- NEXT, en este caso, el resultado de la actividad a realizar ya se encuentra implementado en el entorno, no siendo necesario realizar dicha actividad, por lo que puede obviarse y pasarse a ejecutar la siguiente actividad.
- WARNING, indica que existe algún conflicto y pueden producirse problemas si se continúa con la ejecución de la actividad.
- ABORT, implica que existe o va a dar lugar a algún problema en el entorno y por tanto, no debe continuarse con su realización.

Así por ejemplo, si en una de las actividades del plan se desea parar una unidad de despliegue que no se encuentra actualmente en el entorno, se informará mediante un mensaje de tipo ABORT y en el texto informativo se indicará que la unidad no está presente en el entorno, mostrándose también los datos asociados a dicha unidad.

En el caso concreto en que la respuesta obtenida sea de tipo ABORT, se abortarán automáticamente todas las actividades que dependan de ella, ya que se entiende que estas actividades no pueden cursarse.

A la hora de representar la información ante el usuario, se presentará un mensaje informativo por cada actividad contenida en el plan, el cual indicará si el resultado de la validación ha sido correcto (OK) o incorrecto (ABORT) y varios mensajes, si existe más de una advertencia a considerar antes de ejecutar el plan (WARNING).

4.5. Implementación

El sistema se presenta como un módulo OSGi. OSGi [26] es una plataforma de servicios basada en Java que implementa un modelo de componentes dinámicos, en el que cada componente o módulo se encuentra formado con un conjunto de clases, ficheros de configuración y recursos, que esconde los detalles de su implementación y define una serie de dependencias con otros componentes. La comunicación entre dichos componentes se establece a través de servicios; un registro de servicios permite conocer los componentes disponibles en cada momento. Esto permite al sistema de validación desarrollado actuar como un componente dinámico que puede integrarse en la ADCI, pudiendo ser utilizado por cualquier otro módulo de la arquitectura, simplemente pasándole los argumentos que le son necesarios para realizar la validación. El servicio devolverá al módulo en cuestión los resultados de la validación según el formalismo

expresado anteriormente. El uso de una arquitectura OSGi configura al sistema validador como un servicio que puede registrarse o desregistrase dinámicamente, habilitando el acceso, incluso en paralelo a través de varias instancias cuando sea requerido. Para su desarrollo, se ha hecho uso de Spring DM (*Dynamic Modules*) para plataformas de servicio OSGi [28].

Para la implementación del sistema basado en reglas, fue necesario actualizar el *bundle* proporcionado por Drools a la versión 5.0, ya que éste implementaba la versión 4.7.1. Además, se añadieron librerías específicas para permitir el uso de una tabla de decisión en formato de hoja de cálculo para la gestión de las reglas. En concreto, dichas librerías son:

```
com.springsource.com.thoughtworks.xstream-1.3.0.jar
com.springsource.javax.xml.stream-1.0.1.jar
com.springsource.org.mvel-1.3.5.jar
com.springsource.org.xmlpull-1.1.3.4-O.jar
com.springsource.org.xmlpull-1.1.4.jar
org.eclipse.jdt.core_3.4.4.v_894_R34x.jar
com.springsource.org.drools-5.0.0.jar
jxl-2.4.2.jar
```

Por otro lado, se ha desarrollado una interfaz web para facilitar la interacción entre el usuario y el sistema. Dicha interfaz se encuentra implementada como un paquete de estructura war, que utiliza Spring DM para enlazar con el servicio del sistema de validación a través del gestor de entornos y Spring MVC (*Model View Controller*) para la gestión y representación de los datos.

En concreto, la interfaz muestra al usuario la lista de planes disponibles, permitiendo seleccionar uno de ellos para, tras mostrar su contenido, llevar a cabo su validación y representar los resultados obtenidos. En el caso de que la validación de un plan dependa de otros planes, se mostrará también el resultado de validar dichos planes. Además, existe la posibilidad de validar el entorno asociado al plan, es decir, todos los planes que van a ser ejecutados sobre dicho entorno. En este caso, se presentará como consecuencia una tabla que indicará el resultado de la validación de cada uno de estos planes en orden cronológico, teniendo en cuenta las dependencias existentes entre ellos.

La interfaz web se ha implementado sobre un *servlet*, conectando a un controlador de formulario, mediante el cual se recogen los datos introducidos por el usuario y se gestiona la información representada y la navegación entre las distintas vistas, desarrolladas en base a páginas JSP (*JavaServer Pages*). Dicho controlador es el encargado de ejecutar una instancia del servicio de ejecución, registrado en la ADCI, que representa una interfaz genérica para enlazar con otros servicios registrados en la ADCI, y en este caso, permite enlazar con el sistema de validación de planes.

En la figura 3 se muestra cómo es la interacción entre dichos elementos y el sistema de validación de planes.

5. Pruebas de concepto

El sistema se ha sometido a una serie de pruebas para comprobar su funcionamiento. En primer lugar, se realizaron pruebas unitarias, para ello se crearon diversos planes de despliegue mediante los cuales se cubrían diversas posibilidades. Posteriormente, el sistema se integró dentro del gestor de cambios de entorno, dentro de la ADCI, realizándose pruebas de integración que han permitido analizar el comportamiento del sistema en conjunto.

A continuación se describen varias pruebas de concepto que muestran el funcionamiento del sistema ante diferentes situaciones.

5.1. Validación de múltiples planes

En esta prueba de concepto, tras seleccionar un plan, el usuario visualiza las actividades que lo contienen y a continuación, el sistema lo valida. Pero antes de realizarse dicho proceso, el sistema detecta que existen otros planes almacenados asociados al mismo entorno y con una fecha de ejecución anterior a la del plan a validar. Por ello, a la hora de visualizar el resultado, se observa en la figura 4 cómo, además

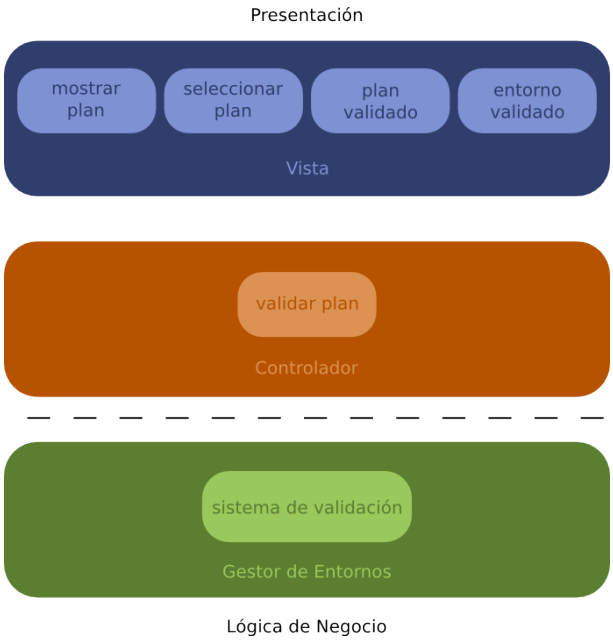

Figura 3: Interfaz web

del resultado de la validación de cada actividad del plan, aparece el resultado de la validación de dichos planes.

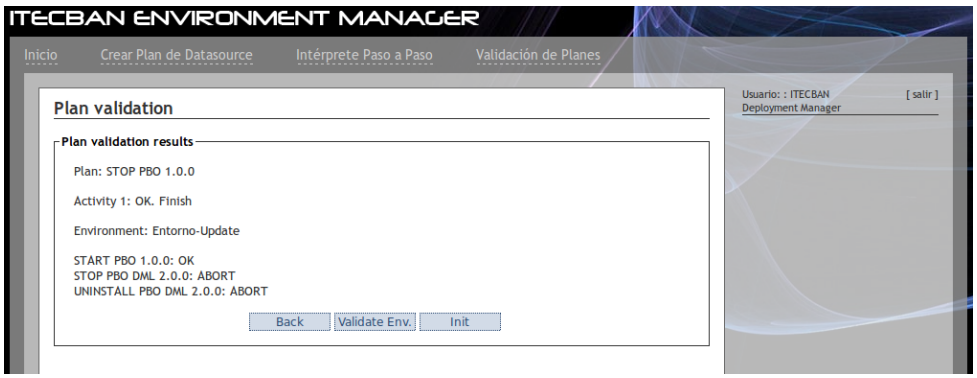

Figura 4: Resultado de la validación de un plan posterior a otros planes

Aunque algunos de los planes previo han tenido una validación negativa, no influyen sobre el plan escogido inicialmente, y en este caso, el resultado de la validación de la única actividad del plan es correcto. Si el usuario decide validar el entorno, podrá observar con más detalle el resultado de la validación de cada plan. En la figura 5 se muestra el resultado obtenido.

En concreto, se observa cómo el primer plan se valida adecuadamente, mientras que el segundo tiene como resultado ABORT, al intentar parar una unidad de la que depende la unidad que ha sido arrancada en el plan anterior. El tercer plan actúa sobre la misma unidad que el segundo, presentando una dependencia con él, por lo que no se valida y directamente se obtiene un resultado negativo.

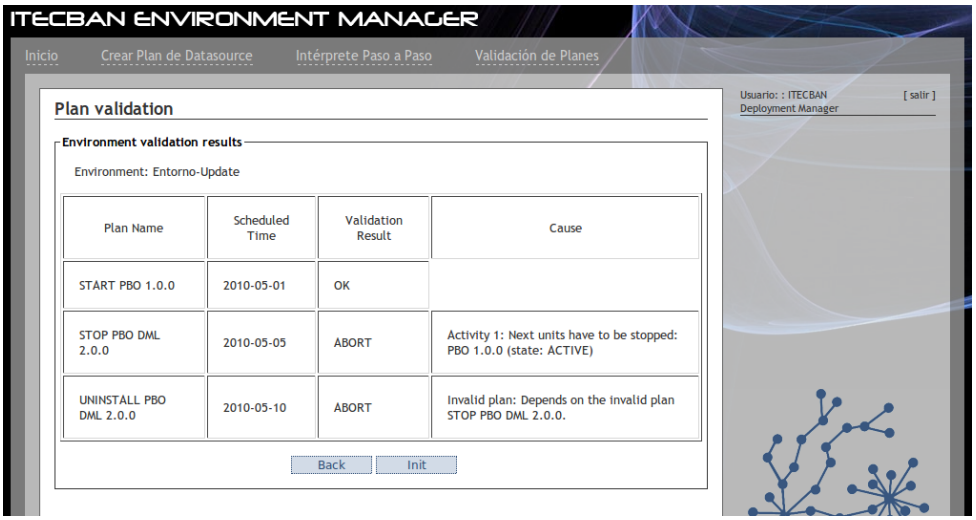

Figura 5: Resultado de la validación de varios planes pertenecientes a un mismo entorno

5.2. Soporte a actividades de configuración y gestión de recursos

Esta prueba de concepto tiene como objetivo mostrar el soporte a dichas actividades. Para ello se valida un plan con varios tipos de actividades enmarcadas dentro de dichas categorías. En concreto, el plan presenta las siguientes actividades:

- Eliminar el recurso `jdbc/Oracle` del contenedor `indra2-oraclebbdd` en el nodo `indra2/138.4.11.137` situado en el entorno `Entorno-Update`
- Añadir el recurso `jdbc/Oracle` del contenedor `indra2-oraclebbdd` en el nodo `indra2/138.4.11.137` situado en el entorno `Entorno-Update`
- Configurar la propiedad `property.user` con el valor `itecbanPBO` del recurso `jdbc/Oracle` del contenedor `indra2-oraclebbdd` en el nodo `indra2/138.4.11.137` situado en el entorno `Entorno-Update`
- Configurar la propiedad `property.user` con el valor `itecbanPBO` del recurso `jdbc/OOOOO` del contenedor `indra2-oraclebbdd` en el nodo `indra2/138.4.11.137` situado en el entorno `Entorno-Update`
- Configurar la propiedad `property.user` con el valor `itecban` del recurso `jdbc/Oracle` del contenedor `indra2-oraclebbdd` en el nodo `indra2/138.4.11.137` situado en el entorno `Entorno-Update`
- Configurar la propiedad `db.name` con el valor `itecban` del recurso pbo-ddl de la unidad pblo-ddl del contenedor `indra2-oraclebbdd` en el nodo `indra2/138.4.11.137` situado en el entorno `Entorno-Update`

El resultado de dicha validación puede observarse en la figura 6, en la que se ve cómo las actividades han sido validadas en el orden determinado por sus dependencias. En primer lugar se valida la primera actividad, que da como resultado NEXT, al no encontrarse el recurso a eliminar en el nodo indicado. A continuación, se valida la sexta actividad, la cual presenta como resultado WARNING, al intentar modificar un recurso que está siendo utilizado por dos unidades. El resultado de la segunda actividad es correcto, pues el recurso puede incluirse sin problemas, y a continuación se observa cómo se validan las actividades de configuración asociadas a dicho recurso, siendo los resultados NEXT, ABORT y OK, pues se intenta configurar un valor de propiedad que ya se encuentra presente, actuar sobre un recurso que no existe, y modificar un valor de una propiedad existente, respectivamente.

Figura 6: Resultado de la validación de un plan con actividades de gestión y configuración de recursos

6. Trabajos relacionados

El despliegue de software es un proceso complejo, pues requiere la coordinación de múltiples elementos de diversa naturaleza sobre un entorno inestable, en el que se debe tener en cuenta que cualquiera de ellos puede fallar en un momento determinado. Los elementos involucrados en el despliegue han ido evolucionando a lo largo de los años, lo que ha complicado aún más el proceso y ha originado el desarrollo de múltiples soluciones a través de las cuales simplificar uno o varios de los aspectos relacionados.

Una de las principales iniciativas es el desarrollo de lenguajes y modelos a través de los cuales definir los conceptos y actividades implicadas, con el objetivo de proporcionar una infraestructura genérica para el despliegue de las aplicaciones. En [14] se muestra una comparación entre OSD y Software MIF, dos propuestas desarrolladas por Microsoft y DMFT, respectivamente. Aunque supusieron un gran avance en el momento de su aparición, no daban soporte al proceso completo de despliegue, por lo que a raíz de ellas surgieron otras, entre las que destaca Software Dock [13]. Esta ofrece una infraestructura que contempla todas las actividades asociadas al despliegue de unidades software, es decir, instalación, actualización, adaptación, reconfiguración y eliminación. Para ello emplea un entorno distribuido basado en el uso de agentes.

Además de caracterizar los elementos partícipes del proceso, otro aspecto clave en este ámbito es la definición de las relaciones entre ellos, para lo cual cabe destacar el uso de grafos para describir las interacciones y los datos intercambiados. En [7] se presenta un ejemplo de su uso aplicado a sistemas *grid*, los cuales se caracterizan por presentar una combinación de amplios conjuntos de datos y servicios distribuidos, que normalmente requieren un procesamiento intensivo. El empleo de grafos contribuye a la creación de algoritmos de planificación complejos [5], lo que aplicado a la validación de planes de despliegue, facilita la gestión de dependencias.

En [4] se proponen una serie de buenas prácticas, actividades y retos relacionados con la actividad de despliegue en el entorno de las telecomunicaciones, con el fin de agilizar el proceso de despliegue de este tipo de aplicaciones. Entre ellas, destacan el disponer de herramientas automáticas que ayuden a los usuarios a verificar que la configuración del entorno sobre el que se va a realizar el despliegue es adecuada a las acciones que se desean llevar a cabo, además de mostrar información acerca de las actividades que se van a desarrollar durante el proceso, facilitando la toma de decisiones en el caso de que exista más de una opción posible e informar acerca de los resultados obtenidos.

En muchas ocasiones es posible diseñar la estrategia a llevar a cabo de diversas maneras, siendo necesario analizar cuál se adapta mejor a las necesidades de cada caso. El empleo de técnicas de inteligencia artificial facilita la gestión de las tareas a realizar, al permitir la automatización de muchas de ellas, mediante el uso de planificadores y sistemas basados en reglas. Así en [20] se define un modelo a través del cual caracterizar las aplicaciones y simplificar el proceso de despliegue mediante el uso de planificadores genéricos. Mientras que en [1] se emplea un sistema basado en reglas que planifica y valida las acciones a realizar durante el despliegue. Y Wilkins y desJardins proponen en [31] el desarrollo de mecanismos

basados en el uso de modelos de conocimiento con el objetivo de implementar sistemas de planificación destinados a entornos complejos.

La elaboración de un plan implica que las actividades contenidas en él sean adecuadas tanto a los elementos que lo integran como al entorno en el cual se va a llevar a cabo, pero el estado de este puede variar a lo largo del tiempo, lo que conduce a que los sistemas de planificación deban poder ser validados durante su ejecución. Esto ha dado lugar al desarrollo de iniciativas destinadas a verificar si la solución tomada es correcta y satisface los requisitos impuestos durante el diseño de la aplicación. En [3] se define una metodología a través de la cual validar la composición de los servicios de sistemas que presentan una arquitectura orientada a servicios. Mientras que Tibermacine et al. proponen en [29] establecer una serie de restricciones a tener en cuenta durante el diseño de la arquitectura de la aplicación para asegurar que la repuesta del sistema es la adecuada durante su despliegue. Yoo et al. sugieren en [32] el uso de redes de Petri para, a través de una representación matemática, modelar los elementos del sistema y analizar la integridad e interoperabilidad de su conjunto. Mecanismo empleado también en [21] para gestionar la incertidumbre del proceso de desarrollo software. Por otro lado, Dubus y Merle plantean en [17] el uso de modelos para validar el despliegue, asegurando que las actividades que lo constituyen y sus opuestas son simétricas y que las políticas aplicadas durante el proceso son correctas. Y Grundy et al. emplean en [12] agentes a través de los cuales validar la funcionalidad del sistema, en concreto, utilizan varios agentes destinados a comprobar distintos aspectos, como pueden ser los tiempos de respuesta, el uso de los recursos o los mecanismos de seguridad de los servicios, entre otros. En cualquier caso, el dinamismo característico de estos entornos hace necesario el estudio de la reconfiguración de los recursos empleados [8] para obtener una disposición óptima [19].

El sistema propuesto se caracteriza por aplicar algunos de los beneficios que ofrece la aplicación de inteligencia artificial en ingeniería del software, en concreto, en la validación del despliegue de servicios. Así emplea modelos para caracterizar los elementos involucrados en el proceso y grafos para especificar las relaciones entre ellos. Además, utiliza un sistema de reglas para determinar que las acciones a realizar cumplen con las restricciones impuestas, adaptándose al estado del entorno en el instante antes de llevarse a cabo el proceso de despliegue, con el objetivo de asegurar la correcta ejecución del proceso. Como resultado, el sistema informa de manera detallada de las actividades a realizar y de los riesgos implicados asociados a cada uno de ellos a la hora de realizar el despliegue.

7. Conclusiones

El despliegue es una fase compleja del desarrollo de aplicaciones empresariales, pues requiere considerar tanto las necesidades y requisitos de las unidades a desplegar como las características dinámicas del entorno sobre el que se va a realizar dicho proceso. Dentro del proyecto ITECBAN, se ha desarrollado una arquitectura de despliegue y configuración para ejecutar procedimientos que pretenden facilitar dicha tarea. Esta arquitectura se encuentra basada en un modelo de información que define tres tipos de recursos principales: entornos, unidades y planes de despliegue.

Los sistemas expertos pueden emplearse para complementar los procedimientos realizados sobre la arquitectura, como son la validación de planes de despliegue y de configuración o la búsqueda de una configuración óptima para llevar a cabo el despliegue. Este tipo de actuaciones son decisiones que actualmente son tomadas directamente por la persona responsable de llevar a buen término la ejecución del plan. Si bien es cierto que tiene suficiente experiencia como para resolver los problemas que se presenten durante la ejecución del plan, este tipo de acciones toman en cuenta múltiples variables y aspectos, requiriendo mucho tiempo y siendo propensas a errores, presentando un alto coste asociado, por lo que la automatización de este tipo de operaciones es un tema de candente investigación. El conocimiento que el experto utiliza para resolver los problemas no es fácil de introducir en el algoritmo determinístico con el que se definen los planes. Los sistemas basados en conocimiento ofrecen más facilidades que los sistemas software convencionales para recoger, interpretar y usar este conocimiento y así servir de ayuda al experto humano o incluso, resolverle automáticamente los problemas planteados, si el experto da su consentimiento.

En este caso se ha decidido utilizar un sistema experto que partiendo de la información contenida en el plan a desplegar y el estado del entorno, indique al usuario si el proceso de despliegue se va a poder realizar correctamente o no, informando de la causa del error. Además, se ha considerado la posibilidad de que el

usuario desee validar planes con una fecha de ejecución posterior a la actual, siendo necesario considerar la existencia de otros planes que presenten una fecha de ejecución anterior, lo que repercutirá directamente sobre el estado del entorno actual. El sistema ha sido implementado con la versión 5.0 de Drools, las reglas han sido representadas mediante una tabla de decisión, facilitando su entendimiento y por tanto, su posible actualización futura. El sistema se presenta como un paquete de OSGi, permitiendo su integración con la arquitectura de despliegue y configuración de ITECBAN. Además, se ha desarrollado una interfaz web, que permite la interacción con el usuario y facilita el acceso a dicha información.

Como trabajo futuro, se propone:

- Modificar automáticamente las actividades contenidas en el plan de despliegue en función de los resultados obtenidos tras la validación.
- Complementar el sistema basado en conocimiento de forma que, además de verificar si un plan en correcto o no, determine si el proceso a realizar es el óptimo.

Agradecimientos

Nos gustaría mostrar nuestro agradecimiento al Gobierno de España, por la financiación del proyecto ITECBAN (MITYC CDTI-CENIT 2005) a través del Ministerio de Industria. Además, también nos gustaría agradecer a José Luis Ruiz, Félix Cuadrado, Rodrigo García y Marco Antonio Prego su apoyo durante el desarrollo del sistema.

Referencias

[1] Naveed Arshad, Dennis Heimbigner, and Alexander L. Wolf. Deployment and dynamic reconfiguration planning for distributed software systems. *Software Quality Control*, 15(3):265–281, 2007. doi: 10.1007/s11219-007-9019-2.

[2] D. Barstow. Artificial intelligence and software engineering. In *ICSE '87: Proceedings of the 9th international conference on Software Engineering*, pages 200–211, Los Alamitos, CA, USA, 1987. IEEE Computer Society Press.

[3] Domenico Bianculli and Carlo Ghezzi. Towards a methodology for lifelong validation of service compositions. In *SDSOA '08: Proceedings of the 2nd international workshop on Systems development in SOA environments*, pages 7–12, New York, NY, USA, 2008. ACM. doi: 10.1145/1370916.1370919.

[4] Lance Bloom and Nancy Clark. IT-management software deployment: field findings and design guidelines. In *CHiMiT '08: Proceedings of the 2nd ACM Symposium on Computer Human Interaction for Management of Information Technology*, pages 1–2, New York, NY, USA, 2008. ACM. doi: 10.1145/1477973.1477985.

[5] Avrim L. Blum and Merrick L. Furst. Fast planning through planning graph analysis. *ARTIFICIAL INTELLIGENCE*, 90(1):1636–1642, 1995.

[6] Lionel C. Briand. On the many ways software engineering can benefit from knowledge engineering. In *SEKE '02: Proceedings of the 14th international conference on Software engineering and knowledge engineering*, pages 3–6, New York, NY, USA, 2002. ACM. doi: 10.1145/568760.568762.

[7] Mario Cannataro and Domenico Talia. The knowledge grid. volume 46, pages 89–93, New York, NY, USA, 2003. ACM. doi: 10.1145/602421.602425.

[8] Jian Cao, Jie Wang, Shensheng Zhang, and Minglu Li. A dynamically reconfigurable system based on workflow and service agents. *Engineering Applications of Artificial Intelligence*, 17(7):771 – 782, 2004. doi: 10.1016/j.engappai.2004.08.030.

[9] Félix Cuadrado, Juan Carlos Dueñas, Rodrigo García, and José Luis Ruiz. A model for enabling context-adapted deployment and configuration operations for the banking environment. *Networking and Services, International conference on*, 0:13–18, 2009. doi: 10.1109/ICNS.2009.85.

[10] DTMF. Common Information Model (CIM) 2.1, 2005. `http:\www.dmtf.org/standards/cim`.

[11] Maria Fox, Richard Howey, and Derek Long. Validating plans in the context of processes and exogenous events. In *AAAI'05: Proceedings of the 20th national conference on Artificial intelligence*, pages 1151–1156. AAAI Press, 2005.

[12] John Grundy, Guoliang Ding, and John Hosking. Deployed software component testing using dynamic validation agents. *Journal of Systems and Software*, 74(1):5–14, 2005. doi: 10.1016/j.jss.2003.05.001.

[13] Richard S. Hall, Dennis Heimbigner, and Alexander L. Wolf. A cooperative approach to support software deployment using the software dock. In *ICSE '99: Proceedings of the 21st international conference on Software engineering*, pages 174–183, New York, NY, USA, 1999. ACM. doi: 10.1145/302405.302463.

[14] Richard S. Hall, Dennis M. Heimbigner, and Alexander L. Wolf. Evaluating software deployment languages and schema. In *ICSM '98: Proceedings of the International Conference on Software Maintenance*, page 177, Washington, DC, USA, 1998. IEEE Computer Society.

[15] Sayed Hashimi. Service-Oriented Architecture Explained. O'Reilly Media, Inc., 2003.

[16] Richard Howey, Derek Long, and Maria Fox. Val: Automatic plan validation, continuous effects and mixed initiative planning using pddl. *Tools with Artificial Intelligence, IEEE International Conference on*, 0:294–301, 2004. doi: 10.1109/ICTAI.2004.120.

[17] J. Dubus and P. Merle. Towards model-driven validation of autonomic software systems in open distributed environments. In *ECOOP Workshop on Model-Driven Adaptation*, 2007.

[18] JBoss Community. Drools Expert, 2010. `http://www.jboss.org/drools/drools-expert.html`.

[19] Tatiana Kichkaylo and Vijay Karamcheti. Optimal resource-aware deployment planning for component-based distributed applications. In *High performance Distributed Computing, 2004. Proceedings. 13th IEEE International Symposium on*, pages 150 – 159, 4-6 2004. doi: 10.1109/HPDC.2004.25.

[20] Sébastien Lacour, Christian Pérez, and Thierry Priol. Generic application description model: Toward automatic deployment of applications on computational grids. In *GRID '05: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, pages 284–287, Washington, DC, USA, 2005. IEEE Computer Society. doi: 10.1109/GRID.2005.1542755.

[21] Xiao Ming Li, Ying Luo Wang, Lin Yan Sun, and Ling Li. Modeling uncertainties involved with software development with a stochastic petri net. *Expert Systems*, 23:302–312, 2006. doi: 10.1111/j.1468-0394.2006.00411.x.

[22] Martín Agüero and Franco Madou and Gabriela Esperón and Daniela López De Luise. Artificial intelligence for software quality improvement. *World Academy of Science, Engineering and Technology*, 63:63–41, 2010.

[23] Farid Meziane, Sunil Vadera, Farid Meziane, and Sunil Vadera. *Artificial Intelligence Applications for Improved Software Engineering Development: New Prospects*. Information Science Reference - Imprint of: IGI Publishing, Hershey, PA, 2009.

[24] Alice M. Mulvehill, Brian Krisler, and Renu Bostwick. Deriving reliable models revisions from executed plan data analysis. In *ICCRTS'09: 14th International Command and Control Research and Technology Symposium*, 2009.

[25] OMG. Deployment and configuration of component based distributed applications version 4.0. OMG documents formal/06-04-02, 2006.

[26] OSGi Alliance. OSGi - The Dynamic Module System for Java, 2009. `http://www.osgi.org`.

[27] Farah Naaz Raza. Artificial intelligence techniques in software engineering (AITSE). In *International MultiConference of Engineers and Computer Scientists (IMECS 2009)*, volume 1, 2009.

[28] SpringSource Community. Spring Dynamic Modules for OSGi Service Platforms, 2009. `http://www.springsource.org/osgi`.

[29] Chouki Tibermacine, Didier Hoareau, and Reda Kadri. Enforcing architecture and deployment constraints of distributed component-based software. In *FASE'07: Proceedings of the 10th international conference on Fundamental approaches to software engineering*, pages 140–154, Berlin, Heidelberg, 2007. Springer-Verlag.

[30] Setsuo Tsuruta, Takashi Onoyama, and Yoshio Taniguchi. Knowledge-based validation method for validating intelligent systems. In *Proceedings of the Fifteenth International Florida Artificial Intelligence Research Society Conference*, pages 226–230. AAAI Press, 2002.

[31] David E. Wilkins and Marie desJardins. A call for knowledge-based planning. *AI MAGAZINE*, 22:99–115, 2000.

[32] Taejong Yoo, Buhwan Jeong, and Hyunbo Cho. A petri nets based functional validation for services composition. *Expert Systems with Applications*, 37(5):3768–3776, 2010. doi: 10.1016/j.eswa.2009.11.046.