



ESCUELA TÉCNICA Y SUPERIOR DE INGENIERÍA
INFORMÁTICA

Ingeniería Informática
Curso Académico 2012/2013

Proyecto de Fin de Carrera

***mHealth Demonstrator
Análisis, Diseño e Implementación del
módulo de Asignación de Descansos
y del Servicio de Rutas.***

Autor:
Víctor Arribas Raigadas

Tutor:
Holget Billhard, Marin Lujak

RESUMEN

Este trabajo fin de carrera se enmarca dentro del proyecto **Agreement Technologies**, en el que colabora el grupo de Inteligencia Artificial de la Universidad Rey Juan Carlos junto con otras instituciones. El proyecto se engloba dentro del dominio de las emergencias médicas, aportando soluciones de gestión y decisión en ámbitos donde el servicio puede ser mejorado o simplificado.

En el ámbito de los equipos de urgencia móviles, la disponibilidad de los recursos es un factor clave. La planificación, gestión y movilización de los recursos es una tarea muy compleja, debido especialmente a la indeterminación inherente al problema.

Este proyecto tiene como objetivo principal aliviar uno de los aspectos de la gestión de los recursos humanos: los descansos. Los **descansos** son los periodos de tiempo que debe disponer el personal médico para poder aliviar su estrés laboral o, en la mayoría de los casos, poder comer. Especialmente cuando la jornada laboral es de veinticuatro horas.

Para afrontar este problema de asignación de descansos dinámica se propone una solución que permita la optimización de la planificación a lo largo de todo el espacio temporal teniendo en cuenta la información obtenida en tiempo real. A lo largo del documento se presentará el análisis diseño e implementación del **sistema de coordinación de descansos** encargado de la planificación de los mismos.

En este proyecto también se encontrará la realización de un **servicio de rutas**, en el cual se aplicarán las competencias adquiridas de comunicación de red y jerarquía de memoria. Su realización no fue una elección realizada a priori, sin embargo, los buenos resultados de la misma han provocado su introducción en este documento. Con respecto a la implementación anterior, la diseñada en este documento es hasta **veinte veces** más rápida.

ESTRUCTURA DEL DOCUMENTO

El documento está dividido en cinco capítulos. El primer capítulo está destinado a la introducción al contexto de la aplicación. El segundo capítulo describe el ámbito de la aplicación y se determinan los objetivos.

En el tercer capítulo se desarrolla el módulo de asignación de descansos. Lo que sería el mainframe del proyecto. En el cuarto capítulo, como aditivo independiente, se desarrolla el servicio de rutas.

El quinto y último capítulo se ha reservado para las conclusiones y trabajo de futuro.

AGRADECIMIENTOS

A Holger y Marin por su inestimable apoyo durante la realización de este proyecto, a Rubén, quien abrió la ventana a esta posibilidad, a David y Diego, compañeros partícipes en el desarrollo. Y a mi familia, por estar ahí siempre.

Esta obra está sujeta a la licencia Reconocimiento-NoComercial 3.0 España de Creative Commons. Para ver una copia de esta licencia, visite <http://creativecommons.org/licenses/by-nc/3.0/es/> o envíe una carta a Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

CAPÍTULO 1.....	1
Introducción	1
Introducción al proyecto AT.....	1
Dominio de las Emergencias Médicas.....	2
Asistencia para emergencias médicas en el SUMMA112	2
Demostrador mHealth	4
Servicios de Localización y Navegación.....	7
 CAPÍTULO 2.....	 9
Asignación de descansos.....	9
Descripción del problema	9
Motivación	12
Objetivos	13
Metodología utilizada	14
 Servicio de Rutas	 15
Descripción del problema	15
Motivación	17
Objetivos	17
 CAPÍTULO 3.....	 19
Análisis del módulo de asignación de descansos.....	19
Capacidades	19
Restricciones	19
Métrica de control y calidad: Satisfacción	20
Factores dinámicos	22
 Diseño de las bases del problema	 23
Satisfacción	23
Política de restricciones: Triangulación de Delaunay.....	24
 Algoritmos de asignación de descansos	 25
Algoritmo inicial	29
Algoritmos diseñados.....	31
Consideraciones y compromisos.....	37
Autoajuste: afinamiento de los parámetros algorítmicos en tiempo real	39

Implementación.....	41
Diseño monolítico	41
Diseño a dos niveles.....	42
Diseño basado en repositorio	45
Alarmas: un gestor de tareas en tiempo real.....	46
Triangulación de Delaunay: envoltura de la implementación existente.....	47
Ficheros de configuración	48
Integración en el mHealth Demonstrator	49
Experimentos.....	50
Minutos descansados.....	50
Tiempos de asistencia a paciente	53
CAPÍTULO 4.....	56
Análisis del servicio de rutas	56
Alternativas	56
Diseño del servicio de rutas	57
Sistema de caché.....	57
Definición de los proveedores	62
Introducción del modelo de comunicación.....	68
Modelo de despliegue.....	71
Speedup, una medida de rendimiento	74
CAPÍTULO 5.....	77
Conclusiones.....	77
Líneas futuras	79
BIBLIOGRAFÍA	81
APÉNDICE A	83
APÉNDICE B	85
APÉNDICE C.....	88
ANEXO 1	90

Capítulo 1

Introducción

En este capítulo, se explican una serie de conceptos necesarios a modo de introducción para la comprensión del trabajo realizado en este proyecto. Muchos de esos conceptos se irán desarrollando con más detalle a lo largo de la memoria.

Introducción al proyecto AT

El trabajo realizado en este proyecto fin de carrera forma parte del proyecto de investigación Agreement Technologies, financiado por el Ministerio de Ciencia e Innovación, coordinado por el Instituto de Investigación en Inteligencia Artificial (IIIA-CSIC) y cuyos miembros colaboradores son la Universidad Politécnica de Valencia y la Universidad Rey Juan Carlos.

El principal objetivo del proyecto AT es el desarrollo de modelos, métodos y algoritmos para la construcción de un nuevo paradigma de comunicación en sistemas distribuidos. Este nuevo paradigma se estructura a través del concepto del acuerdo entre agentes software, el cual permite que los agentes, una vez aceptado el acuerdo, se requieran mutuamente para usar servicios.

Las líneas de investigación del proyecto AT se han estructurado en una arquitectura de torre de 5 niveles.



Arquitectura por niveles del proyecto AT

- Nivel 1: Semántica y gestión de recursos. Solucionar incompatibilidades semánticas y establecer ontologías para comprender las normas o acuerdos.
- Nivel 2: Definición de las normas que determinan las restricciones que todo acuerdo debe satisfacer. En este nivel se debe estudiar la capacidad del software para adaptarse a la normativa.
- Nivel 3: Organizar las estructuras sociales de los agentes: roles y relaciones entre ellos.
- Nivel 4: Este nivel es para el estudio de métodos de argumentación y negociación. Esto se traduce en alcanzar acuerdos que respeten las restricciones que las normas y organizaciones imponen a los agentes.
- Nivel 5: La capa de confianza que se encargará de construir las relaciones entre agentes a partir de la reputación que adquieran estos. Muy importante para un sistema abierto como éste.

Los avances logrados en el aspecto teórico del proyecto AT son probados a través de tres demostradores software: eProcurement (comercio electrónico), mHealth (servicios sanitarios móviles) y mWater (gestión de recursos hídricos).

El segundo de ellos, el demostrador mHealth [RS10, LBCH10], forma parte del desarrollo de este proyecto fin de carrera.

Dominio de las Emergencias Médicas

En los centros médicos de urgencia una asistencia comienza cuando alguien llama al centro con el fin de pedir ayuda. En este momento el centro tiene que determinar que el incidente es una emergencia médica, y si ese es el caso, entonces se evaluará el estado del paciente.

De acuerdo con la evaluación, el centro asigna recursos para ayudar al paciente. Cuando la ambulancia llega a la ubicación del paciente, el personal médico realiza los primeros auxilios. Si el equipo de la ambulancia no es capaz de proporcionar un tratamiento adecuado in situ, a continuación, el paciente se transporta al hospital.

Diferentes centros de emergencia tienen distintas maneras para manejar sus incidencias. Nos centraremos en el SUMMA112, que maneja las emergencias médicas en la Comunidad Autónoma de Madrid en España.

Asistencia para emergencias médicas en el SUMMA112

El SUMMA112 fue creado a partir de la unión de dos servicios sanitarios preexistentes: el SERCAM y el 061.

Su misión es proporcionar asistencia sanitaria en urgencias, emergencias, catástrofes y situaciones especiales en la Comunidad de Madrid. Además, es responsable de gestionar la coordinación funcional entre diferentes niveles de asistencia.

Actualmente, los servicios proporcionados por el SUMMA112 incluyen:

- Recepción y gestión de llamadas médicas
- Regulación Médica de solicitudes de asistencia
- Gestión de camas en los hospitales
- Coordinación y asistencia de urgencias médicas in situ
- Coordinación con otros centros de emergencia
- Transferencia de pacientes con ambulancias convencionales Inter-hospitalaria
- Transferencia con soporte vital avanzado, incluida la transferencia neonatal Inter-hospitalaria
- Coordinación y gestión del transporte sin ayuda urgente
- Gestión del transporte no urgente, programada o no
- Asistencia catástrofe
- Asistencia en situaciones especiales de cobertura preventiva
- Apoyo médico de salvamento en condiciones especiales
- Coordinación del grupo médico de los planes de emergencia de protección civil
- La asistencia médica en catástrofes internacionales
- Formación y enseñanza en materia de emergencias

Las llamadas entrantes son generadas por incidencias. Algunas emergencias médicas pueden causar múltiples llamadas entrantes, que se asocian con la incidencia correspondiente. Las incidencias se clasifican según cuatro niveles:

- Nivel 0 - Vital
- Nivel 1 - Lo antes posible
- Nivel 2 - Siempre que sea posible
- Nivel 3 - No es urgente

Estos niveles de prioridad se pueden cambiar de forma dinámica a lo largo del proceso de la entrevista. Por ejemplo, un operador de llamada puede establecer el nivel de prioridad a 0, pero un médico, después de hacer una serie de preguntas, podrá decidir el nivel de prioridad 1.

Este organismo ha proporcionado los conocimientos y la información que se necesita sobre el dominio de las emergencias médicas, conocimiento muy importante para lograr un demostrador lo más real posible.

Demostrador mHealth

El mHealth Demonstrator es un software de validación de algoritmos. Proporciona estadísticas, métricas, y un histórico de acciones para reproducir los resultados.

Trata con la prestación de servicios sanitarios de alta calidad para apoyar la asistencia sanitaria regular en caso de urgencias. En particular, intenta proporcionar valor añadido tanto a pacientes como a los profesionales médicos. Dentro del ámbito de aplicación del proyecto AT, tiene dos propósitos fundamentales:

- La utilización y evaluación de resultados de investigación –algoritmos, técnicas, métodos y modelos– producidos en el proyecto AT en una situación real.
- Construir prototipos de aplicaciones que aborden problemas reales de tal forma que tengan un gran potencial de evolucionar eventualmente a una aplicación comercial.

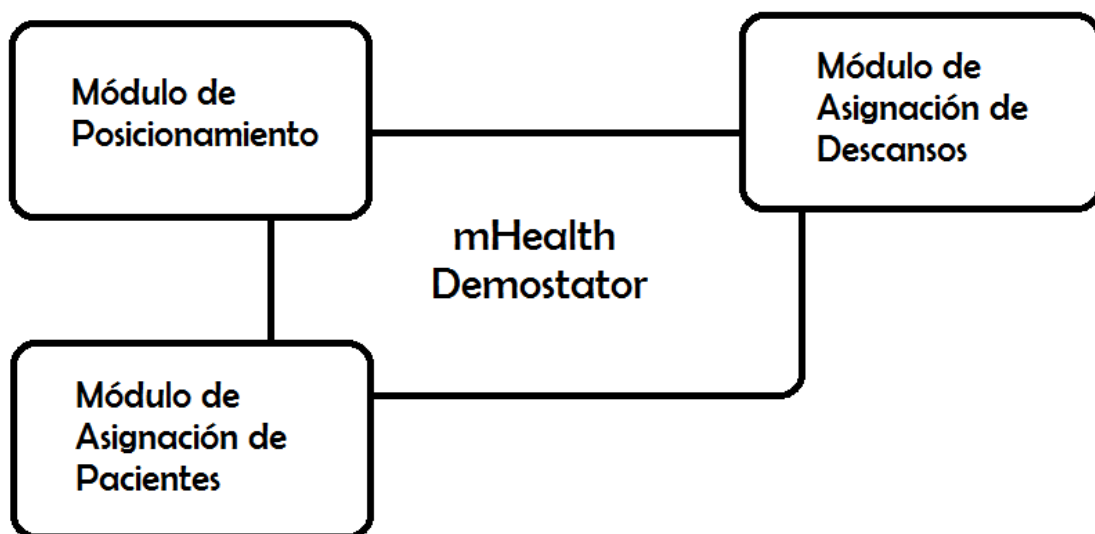
Desde el punto de vista de los profesionales médicos, podrían ser liberados de ciertas tareas rutinarias de decisión y negociación, que actualmente son desempeñadas por humanos pero podrían ser delegadas a agentes artificiales.

El Demostrador mHealth ha sido diseñado usando como referencia el modelo implementado en la Comunidad de Madrid a través del centro coordinador SUMMA112.

Su uso principal es el modelado de los escenarios de uso en los que el SUMMA112 tiene deficiencias para la comparación de este modelo con modelos optimizados propuestos por el equipo del AT.

Está formado por módulos que, proponiendo un mecanismo de coordinación, pretenden resolver un factor clave en el dominio de las emergencias médicas.

En la siguiente ilustración podemos ver la composición del mHealth Demonstrator.



Módulo de posicionamiento

El módulo de posicionamiento ofrece una solución concreta en el ámbito de la coordinación de las Ambulancias.

Actualmente, el servicio del SUMMA112 mantiene las UVI's móviles en sus respectivos hospitales, y sólo son desplegadas a puntos críticos cuando está planificado un gran evento social que aglutine a un cuantioso número de personas.

Los centros urbanos cambian, las ciudades crecen, se construyen nuevas carreteras, y los hospitales son edificados donde mejor puedan acomodarse las instalaciones.

Una de las deficiencias del servicio actual es la cobertura de emergencia. El sistema de coordinación embebido en este módulo pretende cubrir esta deficiencia.

El coordinador utiliza un mapa de voronoi pesado como núcleo del sistema de posicionamiento.

Un mapa de voronoi es una división euclídea del espacio en el que las líneas divisorias de cada área de voronoi se encuentran a una distancia equidistante de los puntos de interés [Aur91].

Uno de los campos en los que se usan mapas de voronoi es en robótica móvil, donde los puntos de interés son cuerpos con volumen, y las líneas, equidistantes a todos estos cuerpos u obstáculos, son el camino más seguro para navegar entre ellos.

En el escenario actual se emplea un mapa de voronoi pesado donde las posiciones de las ambulancias como puntos de interés.

Bajo el mapa de voronoi, hay un espacio que define las probabilidades de ocurrencia de las incidencias, y se utiliza el algoritmo de Lloyd para equilibrar la probabilidad entre todas las áreas.

Módulo de asignación de pacientes

El módulo de asignación de pacientes ofrece una solución concreta en el ámbito de la gestión de recursos de urgencia móviles.

Ante la aparición de una incidencia, los operarios del SUMMA112 determinan el nivel de urgencia de la misma. En función de este nivel y de la accesibilidad de lugar donde se ha producido, ésta puede ser resuelta mediante el envío de uno de los siguientes vehículos de urgencia:

- *UVI móvil:* es un vehículo de asistencia ocupada por cuatro personas: un médico, un DUE y dos técnicos en emergencias médicas. Dado que las unidades de UVI tienen que estar disponibles 24 horas, los diferentes equipos se rotan los turnos. Las UVI móviles son apropiados para prestar asistencia (in situ) y el transporte de los pacientes a los hospitales. De acuerdo con las normas de la Comunidad de Madrid, las UVI móviles deben contener todo el equipo necesario para la ALS (Advanced Life Support).

- *Vehículos de Intervención Rápida (VIR)*: son unidades equipadas con la tecnología y los instrumentos médicos similares a UVI móvil. El propósito de RIV es proporcionar ayuda en caso de emergencia (in situ), que llega al paciente lo más rápido posible. Se determina posteriormente si es necesario transportar el paciente al hospital y en el que este tipo de vehículo de transporte debe ser hecho. Estos vehículos están especialmente preparados para las dificultades orográficas y climatológicas.
- *Unidades de Asistencia Hogar (HAU)*: asisten a las solicitudes de asistencia de emergencia en casa. SUMMA112 tiene Medical HAUs y Enfermería HAUs.
- *Los helicópteros*: uno con base en Las Rozas y el otro en Lozoyuela. Estas unidades tienen los mismos equipos que las UVI móviles. Se utilizan para atender a las llamadas de los lugares con difícil acceso, o aquellas solicitudes de lejos desde los helicópteros mejorar significativamente el tiempo de respuesta.

La asignación de cada uno de estos vehículos de urgencia se evalúa de forma individual. El sistema de coordinación embebido en este módulo pretende cubrir esta deficiencia tratando y optimizando el problema a nivel global mediante mecanismos de subasta.

Módulo de asignación de descansos

El módulo de asignación de descansos ofrece una solución concreta en el ámbito de la gestión de los turnos en la correcta asignación de los tiempos de trabajo-descanso del personal médico de urgencia.

Es el objeto principal de estudio y desarrollo de este proyecto, el cual será desarrollado en el Capítulo 3.

Servicios de Localización y Navegación

Los servicios de localización y navegación son utilizados diariamente por miles de personas y entidades. Proporcionan conocimiento inmediato acerca de la posición, velocidad, tiempo estimado, itinerario hasta el destino, etc.

Estos servicios son empleados por el SUMMA112 para determinar la localización de los vehículos de urgencia, la localización de un herido en la montaña u otra ubicación de difícil acceso y la ruta óptima para asistir a un paciente entre otras.

Los mecanismos de coordinación desarrollados en el mHealth también emplean estos sistemas de geolocalización y navegación para tomar de manera más informada sus decisiones.

Servicio de Navegación: Rutas

El servicio de navegación es un recurso empleado por el mHealth Demonstrator para ofrecer una representación gráfica en tiempo real del comportamiento de los agentes móviles, es decir, las Ambulancias, en el entorno de trabajo.

También es empleado por módulos como el de asignación de pacientes para aportar una estimación lo más precisa posible en términos de tiempo de asistencia.

El servicio sobre el que se apoya el equipo AT es el servicio de rutas para desarrolladores ofrecido por Google Maps.

El empleo en la mayor parte de los casos de estimaciones reales de distancia y tiempo tanto por ese módulo como por otros ha aumentado los requisitos de uso de éste servicio. En consecuencia, es necesario plantear otras soluciones para afrontar los límites del servicio actual.

Esta alternativa será desarrollada en el Capítulo 4.

Asignación de descansos

Descripción del problema

En el contexto de las emergencias médicas, la correcta gestión de los tiempos de trabajo-descanso siempre ha supuesto un “quebradero de cabeza” para el personal administrativo de los centros médicos.

Se trata de un entorno hostil en el que no se puede prever ni el número, ni la localización, ni el tipo de urgencia médica que se puede producir. En estos casos, siempre se trata con planes de prevención y contingencia, pero normalmente estos planes están inducidos por la existencia de un evento social con gran aglomeración de gente.

Tradicional e idealmente, el personal sanitario, humano, tiene una gran devoción a su trabajo y anteponen la salud de los pacientes a su propio bienestar.

El objetivo principal de este proyecto es recompensar la labor de estos trabajadores, diseñando un mecanismo de coordinación que tenga en cuenta el factor humano del personal sanitario, teniendo en cuenta su estado anímico, su estrés, sus preferencias, etc.

El escenario seleccionado para el desarrollo de los algoritmos de toma de decisiones es el escenario propuesto para el desarrollo de los métodos englobados en el mHealth y modelados a través del mHealth Demonstrator. Dicho escenario está centrado en el Servicio de Transporte de Emergencias Médicas, y todas sus consideraciones que afecten de forma colateral o directa a la asignación de los descansos deben ser descritas para su estudio.

Los datos requeridos han sido proporcionados por el SUMMA112. A continuación, se desarrolla el desglose de los conceptos principales.

Objetos de control

El coordinador de descansos actuará sobre las UVI móviles. Para el centro de gestión del SUMMA112, estos vehículos pueden asumir uno de los siguientes estados:

- Activo
 - En espera para una asignación.
 - Transmitido, período comprendido entre la cesión y comienzo de la misión.
 - Movilización, moviéndose a la ubicación del paciente.
 - Intervención "in situ" la asistencia.
 - Traslado, del paciente a un hospital.
- Inactivo
 - Sin personal.
 - En reparación.
 - Bajo limpieza.
 - Otro.

Medidas de calidad

El objetivo general del Servicio de Transporte de Emergencias Médicas del SUMMA112, así como de cualquier otro servicio de emergencia encargado de gestionar el transporte de emergencias médicas, es asegurar que las personas con enfermedades repentinas en cualquier lugar posible dentro del área de influencia de un centro de coordinación recibirán ayuda rápida y sean transferidos (si es necesario) a un centro de emergencia en un hospital en el que pueda recibir el tratamiento para su enfermedad. Desde un punto de vista global, los parámetros que miden la calidad de un servicio de transporte de emergencia en particular (y, por tanto, implícitamente, de una manera particular para coordinar dichos servicios) son los siguientes:

Coste: El objetivo es reducir el coste en términos de número de vehículos y en términos de distancias efectuadas por los vehículos.

Tiempo de Asistencia: El tiempo medio que tarda un vehículo para llegar al lugar donde se encuentran los pacientes con el fin de proporcionar la asistencia médica.

Tiempo de transferencia: El tiempo medio que tarda un vehículo para llevar al paciente a un hospital.

Mortalidad: El porcentaje de casos en los que el paciente muere mientras su caso es "Activo".

Asignación adecuada Hospital: la proporción de pacientes que han sido trasladados a un hospital adecuado, considerando como "adecuación hospitalaria" aquella que puede tratar eficazmente la enfermedad del paciente (ya que tiene capacidad suficiente y porque proporciona los tratamientos obligatorios).

En general, la mortalidad y el tiempo de asistencia se consideran los temas más importantes. Un servicio se considera más eficaz si se puede reducir el coste y tiempo de transferencia y

augmentar la asignación del hospital adecuado, al mismo tiempo que la reducción del mantenimiento o la reducción de los mismos niveles de mortalidad y la hora de llegada.

Servicios mínimos

El sistema de coordinación de los descansos debe asegurar el cumplimiento de unos servicios mínimos y debe ser diseñado con la capacidad de incorporar estos servicios mínimos.

Los servicios mínimos detectados son:

1. El número mínimo de Ambulancias desplegadas proporcionando servicio médico.
2. El tiempo de respuesta ante la aparición de un paciente.

Turnos

Actualmente, el SUMMA112, cuenta con tres tipos de jornadas laborales.

Está establecido un periodo de descanso de sesenta minutos por cada ocho horas de trabajo.

Este intervalo de tiempo es aprovechado por los trabajadores para comer.

Con estas premisas, las jornadas de trabajo quedan representadas tal que:

- La jornada de 24h, se divide en tres tramos de 8 horas y en cada uno de ellos debe producirse un descanso de sesenta minutos.
- La jornada de doce horas se divide en dos tramos de seis horas con dos descansos de sesenta minutos.
- La jornada de ocho horas tiene un único descanso de sesenta minutos.

Estos periodos de descanso son aprovechados por el personal sanitario para comer.

El mecanismo actual de asignación de descansos deja mucho que desear, por ello se ha pedido la inclusión de un módulo que aporte este aspecto a la solución final.

Interrupciones

Mientras una ambulancia está en periodo de descanso, esta puede ser llamada por la aparición de una alerta.

En el modelo actual, tras ocuparse de la alerta los sanitarios terminan su periodo de descanso. Sin embargo, no hay ningún impedimento para que vuelvan a cubrir una incidencia. Dicha alerta genera incertidumbre e insatisfacción en el personal sanitario. Y no ofrece ninguna garantía acerca del cumplimiento de la planificación existente.

La mayoría de los conductores y del equipo de atención de las ambulancias evitan comer en los restaurantes y áreas de descanso adheridas para evitar tener que dejar la comida en caso de recibir un aviso.

Preferencias

Las preferencias del personal sanitario es un factor a tener en cuenta para una correcta planificación orientada a dicho personal. Están constituidas por una lista de horas preferidas con prioridad descendente; la primera hora de la lista es la más preferida y su predilección se reduce al descender en el ranking.

No tener en cuenta estas preferencias es no tener en cuenta el factor humano.

Motivación

Actualmente, el servicio del SUMMA112 gestiona manualmente la planificación de los descansos. Son los propios equipos de cada ambulancia quienes eligen los periodos en los que realizarán sus descansos.

La elección individual puede provocar carencias del servicio en determinadas áreas de influencia, y una mejor gestión reduciría los tiempos de asistencia y la tasa de mortalidad.

Esta gestión se puede apoyar o realizar en su totalidad mediante sistemas automáticos.

Se puede encontrar una extensa literatura [Mus06, BQB10] en la que se proponen soluciones estáticas para el problema. Estas soluciones requieren a priori toda la información para realizar la planificación.

El dinamismo implícito del problema es un factor determinante en la calidad final de la solución. Nosotros proponemos una solución dinámica que lidie con este factor, una aproximación que, en la medida de nuestro conocimiento, creemos que no ha sido explorada en este ámbito aunque sí se pueden encontrar referencias en otros campos [MRM10].

Aunque aportar esta aproximación al área de conocimiento pueda ser considerado de gran valor, personalmente opino que mejorar un servicio tan importante como este, que yo mismo o cualquier otra persona acabará utilizando, sin empeorar la situación laboral de los trabajadores es motivo más que suficiente para realizar este proyecto. Y en definitiva, soñar con la ilusión de estar salvando vidas.

Objetivos

Los principales objetivos de este sistema de coordinación son:

- Crear un sistema de gestión de los descansos eficaz.
Este objetivo no debería ser muy complejo, debido a que se está comparando con un sistema sin coordinación. Sin embargo, la eficacia también mide los requisitos en tiempo y cómputo frente a la mejora obtenida.
- Realizar una asignación de los descansos justa.
La asignación justa de los descansos debe ser el punto de partida para el coordinador de descansos. Un sistema justo garantiza una mayor calidad de los resultados.
- Mejorar el sistema actual de gestión de los descansos.
La creación de este coordinador debe aportar un valor añadido para ser considerado útil.
- Integrar mecanismos que aseguren el cumplimiento de los servicios mínimos.
En el dominio de las emergencias médicas, la reducción de un servicio por debajo de los mínimos estipulados puede suponer la pérdida de vidas. Integrar estos mecanismos no es un requisito opcional.
- Integrar mecanismos adaptativos y/o con aprendizaje.
El ámbito en el que se va a trabajar es bastante complejo y dinámico. Un sistema de aprendizaje por realimentación o refuerzo debería maximizar los resultados sin requerir un esfuerzo analítico adicional.
- Ofrecer una estimación del cumplimiento de la planificación.
El coordinador de descansos no es útil si la planificación que realiza es imposible. Poder contar con una estimación en la acometida de esta planificación reduciría los esfuerzos del personal administrativo en las tareas de validación y monitorización de esta planificación.
- Ofrecer medidas de calidad y de rendimiento.
Las métricas de calidad y rendimiento están intrínsecamente relacionadas con la validez y validación de los algoritmos subyacentes al coordinador.

Metodología utilizada

No se ha utilizado formalmente ninguna metodología concreta, sino que se ha desarrollado un plan de actuación basado en hitos.

Los hitos más relevantes se nombran a continuación:

Modelado del problema

Tras conocer los requerimientos, el primer hito es el modelado del problema.

Este modelo integra el modelo de datos, el plan de actuación y los requisitos cubiertos.

Algoritmo de asignación

El primer hito real es la construcción del núcleo del programa. Comienza con el diseño del algoritmo de asignación de descansos.

Módulo de asignación

Con el algoritmo de asignación diseñado, el siguiente paso es la creación de un módulo auto contenido que permita la ejecución de experimentos para su comparación.

Integración con el sistema

Con el módulo funcionando por sí solo, el siguiente paso es la integración en el Demostrador mHealth.

Experimentación

Una vez validada la congruencia del sistema, el siguiente paso es la simulación del modelo. Es decir, realizar diversos modelos de experimentación que permitan la comparación, tanto cuantitativa como cualitativa de los diversos algoritmos modelados/diseñados.

Estos han sido los hitos planteados inicialmente y existirán facetas del problema que no estén directamente representadas con estos hitos. Cada uno de estos hitos aglutinan hitos de menor categoría.

Servicio de Rutas

Descripción del problema

La ejecución de los experimentos era un proceso muy costoso. Teniendo en cuenta que el objetivo del mHealth Demonstrator era la evaluación de los mecanismos de coordinación integrados, los tiempos de simulación de los experimentos han sido un factor muy importante.

El primer cuello de botella se debía al módulo de posicionamiento. Este tenía unos requerimientos computacionales bastante elevados, pero finalmente fue resuelto mediante el paralelismo de datos y el uso de un servidor de alta potencia¹.

El segundo cuello de botella estaba en el cálculo de las rutas.

El coordinador de asignación empleaba rutas reales como parte de su toma de decisiones. Utilizar tiempos de ruta reales en lugar de distancias aproximadas, como la distancia en línea recta, era un salto cualitativo a nivel de información por la topología de las carreteras. La demanda de rutas por parte de este coordinador, y en un futuro por el de asignación de descansos, era demasiado elevada para ser administrada por el actual proveedor de rutas.

La primera implementación del proveedor de rutas estaba basada en navegador web y extracción de datos mediante llamadas entre Java y JavaScript y tenía el problema de requerir dependencias gráficas. La segunda implementación era la migración de este sistema de cliente emulado al servicio web RESTful proporcionado por Google. Este paso permitió la eliminación de las librerías gráficas, diferentes en cada arquitectura de despliegue.

Ambos recurren a GoogleMaps [GM] como proveedor de rutas.

Este proveedor impone restricciones de uso:

- Número limitado de peticiones diarias.
- Número limitado de peticiones por segundo.

Sobrepasar cualquiera de los dos límites supone la denegación del servicio.

Para mitigar estos límites, se incluyó a un nuevo proveedor de rutas: MapQuest [MQ] (cuyos límites son más laxos).

Aunque la implementación proporcionada utilizaba dos proveedores de rutas, los límites se alcanzaban demasiado pronto cuando era necesario ejecutar baterías de experimentos.

El problema añadido desconcertante, a parte del dispendio en tiempo, era la cancelación de la simulación en caso de un fallo por ambos proveedores. En casos extremos, una simulación de veinticuatro, o incluso hasta setenta y dos horas en caso de ejecutarse en un equipo obsoleto, podría detenerse al noventa y dos por ciento por una denegación continuada del servicio de rutas; ya fuese por haber sobrepasado los límites, o por una caída de la red de la Universidad.

Ante la incomodidad de este comportamiento, se integró un sistema incremental de espera infinita. Sin embargo, este sistema era altamente falible. Unos límites muy bajos suponían un

¹ Servidor Intel de 6 núcleos y 24 hilos de ejecución.

bucle infinito, mientras que unos límites muy altos demoraban la simulación durante días, aunque al menos no se cancelaba.

Este parche era afuncional por el modelo de rutas actual. Al ser un recurso interno de cada simulación, cada instancia de ejecución del demostrador era independiente cumpliendo las restricciones de espera individualmente, pero incumpléndolas globalmente.

Este suceso era común en el Servidor Ortega, donde varios miembros del equipo de investigación lanzaban los experimentos que requerían para su evaluación.

Por último, se comprobó que la mayor causa de cancelación de experimentos, no era la denegación del servicio, sino la existencia de zonas muertas.

Las zonas muertas son áreas geográficas que, por falta de actualización o por estar en obras, no existe una ruta entre una dirección situada dentro de esta área y otra dirección situada fuera. Esto se debe a que se está tratando con un grafo no conexo y no existe ninguna solución inmediata al no conocer la cuantía de estas zonas muertas, de tal manera que, si en algún momento aparecieran nuevas o las existentes serían resueltas al actualizarse los mapas del proveedor.

Por este motivo, un gran número de experimentos importantes no pudieron ser evaluados.

Motivación

Estos hechos, unidos a la imposibilidad de solventarlos sobre la solución actual, motivaron la migración hacia un cambio de paradigma: un servicio.

Si este servicio permitía la inclusión de proveedores que no tuviesen esas zonas muertas, todos los experimentos podrían concluirse y se habría descubierto una implementación de libre distribución para un servicio off-line de rutas basado en OpenStreetMap² [TS].

Además, tras varias pruebas, se detectó que la limitación del servicio por parte de los proveedores estaba a nivel de IP. Por lo que un servicio de rutas que emplease una granja de IPs era la presunta solución a todos los problemas.

Objetivos

Cuatro objetivos han encaminado la consecución del nuevo servicio de rutas:

- Externalización y centralización del servicio.
La existencia de un único servicio es prioritaria. Permite un mejor control del cumplimiento de los límites, el uso de cachés de rutas y su persistencia tras la finalización de la simulación, así como de la gestión a largo plazo del mismo.
- Despliegue en múltiples nodos.
Sin este objetivo, todo lo relativo a este servicio se queda en agua de borrajas. Contar con múltiples nodos permite aumentar los límites globales del proveedor y la replicación del servicio en caso de caída.
- Agregación de nuevos servicios de rutas.
Actualmente, el modelo SAAS está muy extendido y ha entrado en el ámbito del cálculo de rutas. Por ello, en un futuro será fácil encontrar otros proveedores que no tengan zonas muertas, por lo que un diseño que permita su inclusión en el futuro alargará la vida del software.
- Integración de un sistema de cachés
La repetitividad de los experimentos y la inmutabilidad de ciertas variables, como la posición de los hospitales, hacen necesario y conveniente el uso de caché de rutas. Ya se había introducido una caché FIFO, pero en su forma actual no daba buenos resultados.

² Esta implementación, Traveling Salesman, fue desechada posteriormente y no se encontrarán más referencias a la misma.

Capítulo 3

Análisis del módulo de asignación de descansos

En este apartado se realiza un análisis de las consideraciones, condiciones y elecciones de diseño sobre el módulo de asignación de descansos. En adelante, referido como MLB en representación de su denominación interna como Módulo de Lunch Breaks.

Capacidades

Al módulo de asignación de descansos se le han permitido las capacidades de:

- Cambiar el estado de las Ambulancias y del entorno: la introducción de los descansos supone un nuevo estado para la ambulancia, además de modelar nuevos elementos en el entorno como restaurantes y áreas de descanso.
- Afectar al comportamiento de otros módulos: el rol del descanso para comer supone la alteración del comportamiento de las ambulancias. Esta alteración está permitida y se resolverá en el demostrador o en el módulo si aporta un valor añadido.

Restricciones

Uno de los módulos existentes en el mHealth Demonstrator es el módulo de posicionamiento, internamente denominado como Voronoi Diagrams. Este módulo es el responsable del posicionamiento de las Ambulancias a nivel geográfico, integra algoritmos de análisis y decisión, y por lo tanto, es una competencia que no puede atribuirse total o parcialmente en el diseño del nuevo módulo.

El algoritmo principal de LunchBreaks se basará en el comportamiento de VoronoiDiagrams para asegurar la restricción de servicios mínimos (2).

Responsabilidades

El objetivo principal del mHealth Demonstrator es la gestión de los recursos para mejorar la experiencia de atención de los pacientes. El módulo de LunckBreaks debe tener en cuenta el

carácter principal del programa, la atención a pacientes, y por tanto, debe integrar este aspecto en su diseño.

Anteriormente, se ha introducido el concepto de Turnos en relación a la determinación de las jornadas laborales y el establecimiento de los descansos. Este concepto no forma parte del ecosistema del mHealth, y queda como responsabilidad de este módulo su incorporación y tratamiento.

Posteriormente, esta competencia colateral adquirida será delegada sobre un módulo diseñado para tal fin: el módulo ShiftManagement.

Una de las responsabilidades inherentes por formar parte del mHealth Demonstrator es la determinación de una medida de calidad del modelo. Será necesario diseñar métricas de calidad y métodos que aseguren unos valores mínimos.

Métrica de control y calidad: Satisfacción

El primer hito a realizar es la determinación de las métricas de calidad de la solución. Estas métricas servirán tanto para evaluar la calidad de los algoritmos como para que éstos puedan tomar decisiones informadas.

Estas medidas han sido modeladas como funciones de satisfacción. Una función de satisfacción es una función perteneciente al dominio $[0,1]$ que informa cualitativamente de la calidad del modelo medido. Se ha denominado satisfacción por estar fuertemente ligado al estado anímico de los agentes, es decir, del personal sanitario.

Se ha establecido el valor superior del rango, es decir, 1, como el valor que indica la máxima satisfacción posible.

Estas funciones tienen conocimiento histórico, por lo que para función de satisfacción habrá que determinar dos funciones:

- la función de evaluación, encargada de evaluar el estado actual,
- la función de combinación, encargada de fusionar el valor generado por la función de evaluación con el valor histórico de la función.

Se han detectado tres ámbitos de interés para los que se modelarán funciones de satisfacción: satisfacción por el cumplimiento de las preferencias de descanso, satisfacción por el número de interrupciones y satisfacción por el número de pacientes atendidos.

Las dos primeras miden el estado de ánimo de los agentes. La tercera mide el esfuerzo de los agentes, y ha sido incluida como un factor de recompensa a los mismos.

Preferencias

Las preferencias son una lista ordenada de horas a las que los agentes prefieren descansar. La función de satisfacción modelada tiene en cuenta este orden en su evaluación.

- Función de evaluación: $1 - \frac{n^{\circ}_{orden}}{n^{\circ}_{preferencias}}$
- Función de combinación: $W \times S_{nueva} + (1 - W) \times S_{historica}$

La función de evaluación anterior no tiene en cuenta las interrupciones. De este modo, tendíamos el mismo valor de satisfacción para dos situaciones completamente distintas: haber descansado durante sesenta minutos y haber descansado solo diez.

La inclusión de las interrupciones queda modelada con la siguiente función de evaluación:

$$1 - \left(\frac{n^{\circ}_{orden}}{n^{\circ}_{preferencias}} \times \frac{tiempodescansado}{tiempoasignado} \right)$$

Interrupciones

Las interrupciones son una de las causas que han motivado este proyecto. Inicialmente, se planteó $\frac{1}{n^{\circ}_{interrupciones}+1}$ como función de evaluación, sin embargo, se optó por una función más flexible que admitía un comportamiento dual: comportarse como función exponencial o como función lineal saturada.

- Función de evaluación:
 $base^n, \quad base \in [0,1], \quad n \in (0, \infty) \equiv \text{número de interrupciones}$
- Función de combinación: $W \times S_{nueva} + (1 - W) \times S_{historica}$

Trabajo

Este ámbito puede definirse con dos funciones de satisfacción en función del enfoque de apreciación.

Desde el punto de vista del personal sanitario, la función de evaluación establecida es una función gaussiana. El punto de máximo valor estaría determinado por el número óptimo de pacientes que debería atender. Ni demasiados, disminuyendo la satisfacción de los agentes por el exceso de estrés, ni muy pocos, permitiendo la relajación de los agentes y un aumento de las posibilidades de cometer errores.

Esta función de satisfacción no ha sido utilizada para modelar el problema al no poder ser aprovechada por el algoritmo.

Toda función de satisfacción que no pueda ser alterada por los algoritmos de asignación solo introduce ruido en el modelo.

Desde el punto de vista de gestión, se puede incluir esta faceta del problema como factor de recompensa. Un agente con un alto número de pacientes atendidos debería disfrutar mejor de su descanso para disminuir el estrés.

- Función de evaluación: $\frac{n^{\circ} \text{pacientes atendidos}}{n^{\circ} \text{total de pacientes}}$
- Función de combinación: $W \times S_{nueva} + (1 - W) \times S_{historica}$

La intrusión de este modelo de satisfacción orientado a recompensa será objeto de estudio y permitiría la alteración del comportamiento del algoritmo, y por tanto, del módulo, de forma semi-transparente.

Para su utilización dentro del algoritmo de asignación de descansos, se ha simplificado a un único valor numérico definido por la siguiente función.

$$S_{total} = W_p \times S_{preferencias} + W_i \times S_{interrupciones} + W_t \times S_{trabajo}$$

Cumpléndose que

$$\sum W_i = 1$$

Factores dinámicos

El entorno de las emergencias médicas es un entorno dinámico. Según el modelo FIPA, todo agente debe ser reactivo.

Las decisiones tomadas por el MLB deben tener en cuenta la mutabilidad del entorno y satisfacer los cambios producidos en tiempo real. Además, debe ser compatible con el funcionamiento del mHealth Demonstrator.

En consecuencia, el MLB será diseñado como un módulo que trabajará en tiempo real.

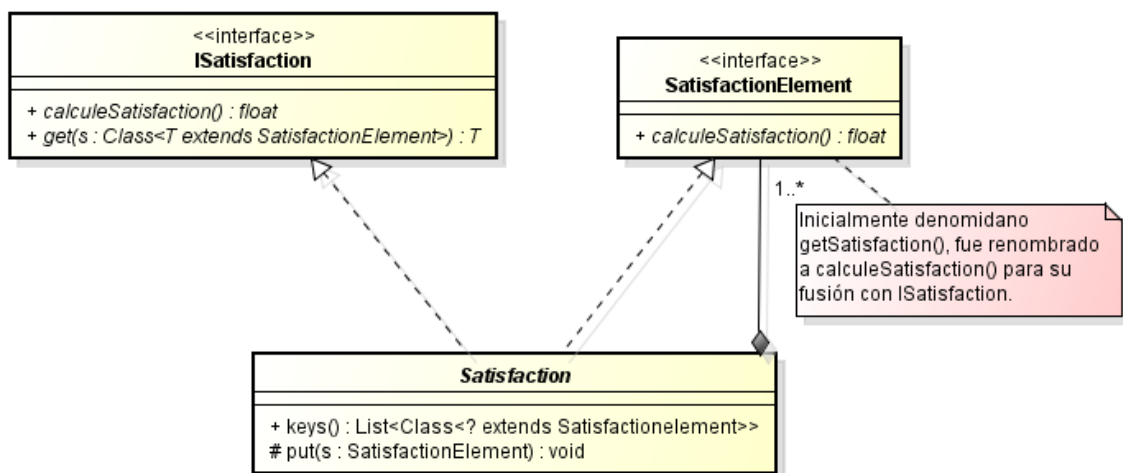
Diseño de las bases del problema

Este apartado engloba las decisiones de diseño así como la explicación de los componentes más relevantes.

Satisfacción

En apartados anteriores han sido definidas tres funciones de satisfacción, sin embargo el problema puede requerir la inclusión de más funciones, centradas en otros aspectos del problema. Por ejemplo, la satisfacción por trabajo fue introducida tiempo después.

Esta conciencia sobre extensibilidad y los requisitos de combinación y comparación de las funciones motivó la realización de una estructura que cumpliera dichos criterios.



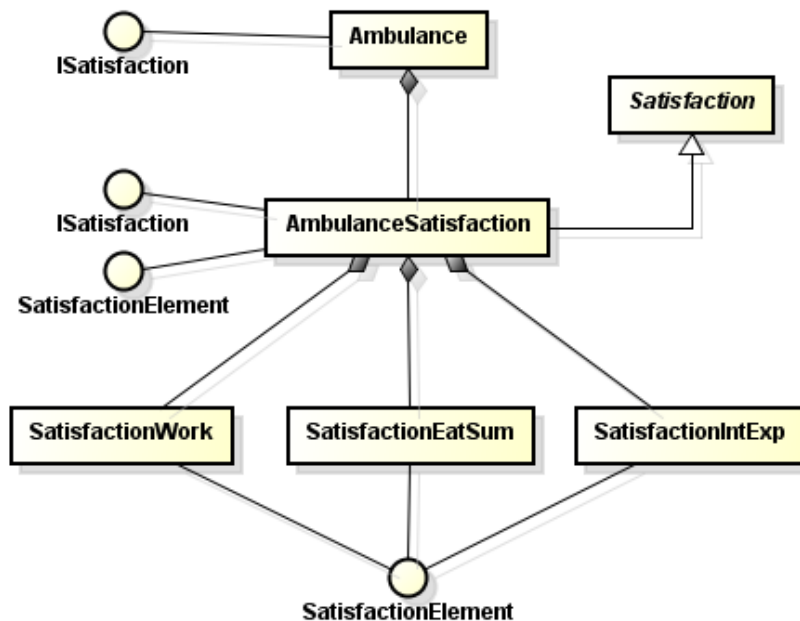
En el diagrama anterior podemos ver la definición de esta estructura, acorde con el patrón de diseño *composite*. Internamente funciona con el patrón *singleton*, ya que cada función de satisfacción es única. Así, es posible encadenar jerárquicamente las distintas funciones de combinación y recuperarlas individualmente para su consulta, actualización o modificación.

El único requisito que deben cumplir las funciones de satisfacción para utilizar este diseño es que deben ser resumibles a un único valor decimal.

La interpretación del valor resumido, correrá a cargo de la clase contenedora, mientras que la actualización de la misma no requerirá la modificación ni adición de métodos en esta.

Creemos que este diseño cumple bastante bien el principio abierto-cerrado, OCP (Open Closed Principle).

A continuación, se presenta la jerarquía de satisfacción para el problema que nos ocupa. En este caso concreto, la entidad *Ambulance* forma parte integral de la jerarquía de satisfacción para poder realizar ordenaciones y actualizaciones de la satisfacción sin afectar al funcionamiento de la misma.



Política de restricciones: Triangulación de Delaunay

Para la definición de las relaciones y restricciones geográficas existen diferentes propuestas como la teselación del área de trabajo, el uso de grupos jerárquicos o la formalización del problema como un CSP [LBS07, JCS]. Estos métodos requerían un conocimiento explícito y complejo de cómo debían agruparse y clasificarse las ambulancias.

En los primeros compases del proyecto este salto de complejidad era demasiado grande como para poder realizarlo.

Por suerte, desde el equipo se planteó que una forma de establecer estas restricciones era realizar una triangulación de Delaunay tomando las posiciones geográficas de las ambulancias como los puntos que conforman la triangulación.

Una triangulación de Delaunay es una red de triángulos que cumple la condición de Delaunay.

Esta condición dice que la circunferencia circunscrita de cada triángulo de la red no debe contener ningún vértice de otro triángulo. Además, asegura que los ángulos del interior de los triángulos son lo más grandes posible, es decir, maximiza la extensión del ángulo más pequeño de la red.

La triangulación crea un grafo completamente conexo, donde los arcos, o aristas de los triángulos de Delaunay, proporcionan el conocimiento de vecindad.

Además, la triangulación de Delaunay y el mapa de voronoi son duales, por lo que su uso creará una coherencia implícita entre ambos módulos y se obtendrá un mejor funcionamiento en su conjunción.

Algoritmos de asignación de descansos

En este apartado se concentrarán meses de análisis, desarrollo y debate exponiendo de forma breve los motivos que han guiado la forma de los algoritmos. Aunque se han estudiado una suma considerable de algoritmos, en este documento sólo serán explicados los implementados y los más relevantes.

Como nota, se puede encontrar el pseudocódigo del primer algoritmo completo de asignación de descansos en el Apéndice A.

En entornos dinámicos complejos, donde es complicado desarrollar un modelo exacto del problema, formalizar una solución deliberativa requiere más esfuerzo que una reactiva.

En consecuencia, con el mismo esfuerzo, un algoritmo reactivo que tome las decisiones *on-line* tendrá más posibilidades de lograr una solución mejor que un algoritmo estático. Además, todo algoritmo reactivo puede incorporar la característica deliberativa si esta es computacionalmente asumible en términos temporales.

Por ejemplo, para tener en cuenta las interrupciones, si no contamos con un modelo de pacientes preciso, se producirá el principio de dualidad probabilística.

Principio de dualidad probabilística

Sea la función probabilística F , existen dos comportamientos esperados en función del conocimiento acerca de la población:

- i. Para una población de tamaño definido, la desviación de la probabilidad en tiempo t será contrarrestada por una desviación similar en sentido contrario en tiempo $t+k$.
- ii. Para una población cualitativamente infinita, una desviación de la probabilidad no compensada supone que la cantidad de individuos categorizados con tal observación, así como el tamaño de la población, es como mínimo el número de individuos ya observados más la parte proporcional de la diferencia entre la categoría sobre-observada y la sub-observada más la compensación de la desviación.

Para una población de tamaño indeterminado, cualquiera de los dos comportamientos es factible.

La triangulación de Delaunay ha sido elegida base por antonomasia para los algoritmos de decisión. Sin embargo tiene una característica no muy deseada en el uso que se le va a aplicar.

El módulo de posicionamiento recoloca las ambulancias de modo que la aparición de un nuevo paciente es, en la medida de lo posible, equiprobable. Esta propiedad refuerza la característica de igualdad entre todos los agentes, de modo que no importa en términos temporales ni espaciales la posición de un agente.

La igualdad entre agentes es el principio para la creación de un sistema justo.

Sin embargo, la triangulación añade una característica distintiva a los agentes, ya que la cantidad de aristas, y por tanto de vecinos, varía según si el agente se encuentra en el centro o en los bordes del área de trabajo.

La diferencia en el número de arcos de cada nodo y la topología concreta de un grafo de Delaunay pueden generar mínimos locales. Es decir, ante un algoritmo de optimización que únicamente maximice el número de asignaciones sin tener en cuenta ningún otro factor, existirán nodos del grafo de Delaunay que siempre pertenecerán a la solución y otros que nunca formen parte de la misma, aunque la topología cambie con unas restricciones de movimiento razonables.

Como conclusión, cualquier algoritmo de optimización basado en la triangulación Delaunay que maximice la satisfacción será intrínsecamente injusto y será factible que nunca se planificará el descanso de una ambulancia porque en términos globales, su insatisfacción hasta el grado de hastío seguirá siendo la mejor solución al implicar un valor acumulado o total de satisfacción mayor.

Este problema puede reducirse si en lugar de maximizar la satisfacción, se maximiza la ganancia, concepto utilizado en uno de los algoritmos.

A medida que se avanza en el proyecto, se puede ver cómo cada vez se producen más concesiones. Aunque creemos que estas concesiones son necesarias para conseguir desarrollar más allá de los conceptos teóricos unos algoritmos que sirvan al menos como línea de base.

Se considera línea de base al mínimo requerimiento algorítmico que suponga un incremento sustancial en términos de calidad al comparar la solución propuesta con la existente.

Como demostración no formal de que estas concesiones son necesarias, vamos a partir del problema acotado y relajar las concesiones tomadas hasta alcanzar un entorno que al menos parezca NP-Hard.

Supongamos que tenemos que planificar un único periodo de descanso, por ejemplo de ocho horas. En este periodo el descanso estipulado es de sesenta minutos, los cuales se tomarán de una sola vez. La repartición de los descansos se realizará de hora en hora.

En este momento, el tamaño del problema es el número de combinaciones de ambulancias tomadas de ocho en ocho.

Ahora supongamos que el tiempo de descanso se puede fraccionar, por motivos de simplificación, en dos periodos. La base combinatoria pasa de ocho a dieciséis.

Si añadimos la posibilidad de veinte minutos, solo esta es de ocho sobre tres. Podemos añadir más fracciones, como diez, quince, y que estas se puedan combinar hasta sumar sesenta.

Ahora supongamos que no hay una restricción de horas de asignación. Estas pueden tomarse en cualquier minuto. En cálculos fáciles, cada cinco minutos. Con el planteamiento original, ocho horas se pueden dividir en dieciocho mil periodos. Si se prohíben los solapes el número de combinaciones final es mucho más reducido, pero la complejidad del problema ha escalado en proporción inversa.

De momento solo se ha hablado de un periodo, el intervalo de planificación puede tener varios. Este incremento del problema es el menos representativo, ya que solo aumenta el problema en términos multiplicativos.

Ahora que se puede suponer que el número de posibilidades del problema es suficientemente grande es momento de introducir los factores dinámicos, que implicaría entre otros la introducción de periodos en los que la ambulancia está ocupada (por ejemplo atendiendo a un paciente), o la variación de la posición de las ambulancias a lo largo del tiempo.

Y por supuesto, el grado de realismo que debe integrar esta aproximación para que la calidad de la solución obtenida sea transferible al modelo real, como por ejemplo, incluir en el modelo los restaurantes, las áreas de descanso, las zonas de avituallamiento, los menús ofrecidos y preferidos, etc.

Asumiendo el turno de veinticuatro horas, este está dividido en breaks de ocho horas (por ejemplo: mañana, tarde, noche). Cada break debe contar con sesenta minutos de descanso.

Si estos sesenta minutos solo pueden ser asignados cada hora, el número de posibilidades es de ocho. Si la restricción es a nivel de minuto, es decir, un descanso puede comenzar a la 1:43 del mediodía, el número de posibilidades crece hasta unas 420.

Sin embargo, si estos descansos pueden tomarse en periodos de tiempo inferior (por ejemplo 15, 20, 30) el número de combinaciones crece factorialmente.

Estos valores son para una única ambulancia, cada nueva ambulancia en el problema supone un incremento exponencial.

Y todavía no se ha tenido en cuenta el factor dinámico del problema, que implicaría la introducción de periodos en los que la ambulancia está ocupada

con una incidencia, o la variación de la posición de las ambulancias a lo largo del tiempo, ni el grado de realismo que debe integrar esta aproximación para que la calidad de la solución obtenida sea transferible al modelo real, como por ejemplo, incluir en el modelo los restaurantes, las áreas de descanso, las zonas de avituallamiento, los menús ofrecidos y preferidos, etc.

Solo tratando de resolver el problema tal y como se contempla en esta explicación pueden entenderse las decisiones tomadas.

Algoritmo inicial

En este apartado, se realizará la definición formal del algoritmo de asignación de descansos. Está formulado como un problema de maximización y tiene en cuenta la satisfacción de las preferencias y las restricciones geográficas a través del grafo de vecindad basado en la triangulación de Delaunay.

Formalización del algoritmo

Algoritmo de asignación de horas para comer

Sea $A = [a_1, a_2, \dots, a_n] \equiv P$, el conjunto de ambulancias ordenadas por Satisfacción.

Sea H el conjunto de horas asignables para comer.

Conociendo $NG_A(C_1, \dots, C_n)$.

El algoritmo cumplirá que:

$\forall i \in [1, 2, \dots, n] \exists h \in H$ tal que $|C_i|_h < k$, $|A|_h < K$ y $S(a_i, h)$ es máxima

donde $k = n^\circ$ máximo de ambulancias vecinas comiendo a la vez,

y $K = n^\circ$ máximo de ambulancias comiendo a la vez en toda la región.

Conceptos

Dados un conjunto de puntos $P = [p_1, p_2, \dots, p_n]$ en 2D, puede construirse un $DT_P(t_1, t_2, \dots, t_n)$ tal que $\forall i \in [1, 2, \dots, n]$, $t_i = [p_a, p_b, p_c]$ es un triángulo con $a \neq b \neq c$ que cumple las propiedades de la Triangulación de Delaunay.

Sea $DP_P (<p_j, p_i>, \dots)$, Delaunay Pairing, una representación binaria de las relaciones del DT_P tal que la tupla $<p_i, p_j> \in DP_P$ si y solo si $\exists t \in DT_P$ tal que $p_i, p_j \in t$, con $i \neq j$

Sea $NG_P (C_1, C_2, \dots, C_n)$, NeighborhoodGraph, una representación de la vecindad del conjunto $P = [p_1, p_2, \dots, p_n]$, donde $C_i \subseteq P$ tal que $\forall p \in P, p \in C_i$ si y solo si $<p_i, p_j> \in DP_P$

Definiciones auxiliares

$|C|_h \equiv |C|$, $x \in C$ si y solo si $x \in C$ y $h = h(x)$, sabiendo que $|C|$ es el número de elementos del conjunto C .

$h(x)$ es una función que devuelve la hora que le ha sido asignada para comer a la ambulancia x .

$S(a, h)$, $a \in A$, $h \in H$, es una función que calcula la satisfacción de la ambulancia a por comer en la hora h .

Problemas

El algoritmo anterior tenía dos problemas:

1. Contenía mínimos locales: la maximización de la satisfacción global jugaba en detrimento de la satisfacción de individuos concretos. Estos individuos podían pasar N pasos de asignación con una satisfacción por debajo de los mínimos de calidad.
2. La implementación realizada era un método de búsqueda en profundidad, es decir, muy lento.

Para eliminar el problema de los mínimos locales y elaborar un algoritmo justo a corto plazo, se incluyó una restricción al problema:

La asignación de los periodos de descansos se haría en orden de satisfacción, de modo que, un agente con satisfacción mínima optaría inamoviblemente a su mejor opción.

Esta restricción supone una mejora extremadamente alta en orden de complejidad, pasando de un orden de complejidad factorial a uno cuadrático en el peor de los casos.

La alternativa es estudiar y conocer las localizaciones favorables y desfavorables y rotar las posiciones de las ambulancias si la injusticia del modelo solo dependiera de eso.

Algoritmos diseñados

Algoritmos con restricciones geográficas

Los siguientes algoritmos realizados toman como base una restricción geográfica. Esta restricción impone medidas para asegurar una distribución uniforme de las asignaciones.

Asignación basada en satisfacción

Este es el algoritmo principal para la asignación de descansos y la base para sus variantes. Realiza una planificación completa para todas las ambulancias y para todas las horas de la franja actual (por ejemplo: desayuno, comida, cena).

A continuación, se describen los pasos realizados por el algoritmo:

1. Se toman todas las ambulancias activas, es decir, que estén trabajando, del sistema.
2. Se clasifican en tres grupos:
 - a. *Atendiendo a pacientes*: las ambulancias pertenecientes a este grupo son aquellas que están atendiendo a una incidencia en el momento de la asignación. Por ejemplo, las ambulancias que estén realizando una asistencia in situ o transportando al paciente hacia un hospital forman parte de este grupo.
 - b. *Descansando*: las ambulancias pertenecientes a este grupo son aquellas que anteriormente fueron planificadas para realizar su descanso y éste aún no ha terminado.
 - c. *En espera*: las ambulancias pertenecientes a este grupo son aquellas que están desplegadas para la cobertura del servicio, pero que no están realizando ninguna tarea. Puede considerarse este estado como de alerta o en espera de recibir una incidencia.
Estas ambulancias son las que, por no estar ocupadas con ninguna labor, serán sensibles a la asignación de descansos.
3. Se comprueban los límites de asignación. Si estos límites han sido alcanzados, el algoritmo termina aquí.
4. Inicio del algoritmo de asignación:
 - a. Se crea un grafo de vecindad que incluye a las ambulancias de los grupos 'descansando' y 'en espera'
 - b. Para cada ambulancia del grupo 'en espera' ordenada por satisfacción
 - i. Para cada preferencia de la lista de preferencias de esta ambulancia
 - ii. Si esta hora cumple los criterios de ocupación:
 1. No existe ninguna ambulancia vecina que haya sido asignada (ahora o en el pasado) en esta franja horaria.
 2. No se han alcanzado los límites de asignación para esta hora.
Entonces se le asigna dicha hora.

Como se puede ver, las asignaciones resultantes de este algoritmo son una planificación de todo el periodo para cada una de las ambulancias. Sin embargo, como el grafo es dinámico, no se pueden garantizar que las asignaciones futuras cumplan las restricciones de vecindad y ocupación.

Por ello, se tiene en cuenta únicamente las asignaciones para el periodo de tiempo más inmediato, dejando las asignaciones para el resto de periodos como una estimación del algoritmo de planificación.

Esta estimación es útil para determinar si en las condiciones actuales es posible realizar una asignación completa cumpliendo con los criterios establecidos y permite tomar medidas extraordinarias en caso de que no se garantice el periodo de descanso para los trabajadores si la situación actual lo permite.

Algoritmo con reasignación

En escenarios donde existen interrupciones, el algoritmo anterior, al no contemplarlas, no es suficiente porque sub-planifica.

Cada vez que se produce una interrupción, la ambulancia implicada debe dejar su descanso no completando el mismo. En consecuencia, en el próximo paso de planificación volverá a necesitar una asignación para completar su tiempo de descanso.

Recordemos que el algoritmo anterior realiza la ocupación de una hora si ésta es la mejor preferencia que cumple las restricciones. Supongamos, como caso degenerado, el siguiente escenario:

El personal sanitario de todas y cada una de las ambulancias ha estimado que la hora a la que más le gustaría comer es la última, después la penúltima, después la antepenúltima, y así sucesivamente.

De este modo, el algoritmo de asignación anterior proporcionaría una máxima ocupación para la última hora, reduciéndose la ocupación de cada franja temporal al reducirse la demanda, hasta que, por ejemplo, la primera hora el turno quede desierta al no ser una hora preferida por nadie.

Si en los primeros compases del turno se produce una interrupción, ésta deberá volver a descansar en un periodo de tiempo posterior. Como ya se había garantizado la máxima satisfacción teniendo en cuenta las preferencias, una de estas interrupciones se producirá cuando todas las horas futuras estén en su máxima ocupación. En este momento y para cada una de las interrupciones siguientes, una de las ambulancias se podrá quedar sin descansar.

Dependiendo del valor de satisfacción entre la que ha sido interrumpida y otra que haya sido planificada en el futuro, una u otra dejará de ser asignada, perdiendo la primera el periodo de tiempo que aún le resta o inhabilitando a la segunda a realizar su descanso.

Esta reducción de la eficacia del algoritmo nunca se produce cuando el escenario es exactamente el opuesto: todas las ambulancias prefieren las primeras horas del turno.

Para paliar esta situación se propone el modelo de reasignación.

El modelo de reasignación realiza lo que su propio nombre indica, reasignar. Realiza todos los pasos descritos en el algoritmo anterior, pero además recuerda cuáles de las asignaciones son las planificadas sobre horas futuras.

Para garantizar una satisfacción más elevada se tienen en cuenta las preferencias de cada una de estas ambulancias, y solo serán asignadas aquellas cuya pérdida por la nueva asignación sea menor. Esta pérdida se puede valorar cuantitativamente como la distancia en la lista de preferencias entre la hora planificada y la hora inmediata.

Al trabajar con una lista ordenada de preferencias, se cumple la siguiente propiedad matemática:

Sea $E(a, h1, h2) = ranking(a, h1) - ranking(a, h2)$ una función que calcule la distancia entre la hora $h1$ y la hora $h2$ en la lista de preferencias de la ambulancia a . Sea $h1$ la hora inmediata a la que se quiere realizar la nueva asignación y $h2$ la hora previamente asignada. Se puede determinar que:

Si $E(a, h1, h2) = 0$, entonces se cumple que $h1 = h2$

Si $E(a, h1, h2) < 0$, entonces $h1$, que es la hora actual, es una hora que está más arriba en la lista de preferencias de a . Es decir, es una hora más preferida para la ambulancia pero por las restricciones del algoritmo no ha podido ser elegida.

Si $E(a, h1, h2) > 0$, entonces $h1$ es una hora menos preferida que $h2$, y el resultado de esta función es la pérdida que asumirá la ambulancia ante la nueva asignación.

La primera conclusión que se puede extraer es que, las reasignaciones para las que el valor de la función sean menor que cero, no son asignaciones válidas con las restricciones actuales, por lo que pueden desecharse.

La segunda conclusión es que si se tiene únicamente en cuenta en rango positivo de la función, incluido el cero, y estas reasignaciones son ordenadas de menor a mayor, las que ocupan los primeros puestos de la lista son las asignaciones realizadas por el anterior algoritmo, y las últimas son las nuevas.

Si además, se ordena por satisfacción aquellas reasignaciones en las que $E() = 0$, el orden de éstas será el mismo que el orden de las asignaciones del algoritmo anterior.

Existen dos opciones para ejecutar este algoritmo:

- A. Realizar una ejecución en dos etapas
 - a. Algoritmo basado en satisfacción
 - b. Algoritmo con reasignación tomando únicamente las reasignaciones con $E() > 0$.
- B. Realizar una ejecución de una etapa, sustituyendo las asignaciones generadas por el algoritmo de asignación original por la reasignaciones con $E() = 0$.

Como podemos observar, la estructura actual de las preferencias ha permitido optimizar la velocidad de ejecución del algoritmo con reasignación.

Algoritmo de ocupación masificada

En escenarios con un elevado número de pacientes, la probabilidad de interrupción es más alta.

Cuando las interrupciones son tan elevadas solo hay dos interpretaciones posibles: i) la demanda actual por los pacientes es tan alta que los servicios de urgencia deben cumplir con su labor y atenderlos a todos, y para reducir esta carga es necesario aumentar los recursos hasta una situación 'normal', o ii) los recursos son los que son, y el sistema de coordinación debe bregar con la situación.

Asumiendo el caso (ii), que es el que implica a este proyecto, se incluye esta variante del algoritmo original de asignación.

En esta variante, la única modificación es la relajación de la restricción de vecindad, permitiendo que haya dos ambulancias cercanas que estén descansando a la vez.

En los experimentos realizados, la inclusión de este algoritmo a los ya existentes permite que todos los trabajadores consigan descansar el tiempo estipulado. Sin embargo, es un descanso muy fraccionado porque se eleva el número de interrupciones.

Sobre este algoritmo hay dos variantes, en la primera variante el orden de asignación es de menor a mayor preferencia, en la segunda variante el orden de asignación es definido por la ganancia de satisfacción. Es decir, se realizan las asignaciones cuya evaluación otorgue una satisfacción más alta.

Este algoritmo sólo se ejecutará como un paso extra de asignación cuando la situación lo requiera.

Si este algoritmo se ejecuta solo, se pueden crear áreas de influencia sin cobertura. Además de no ser un algoritmo justo. Como vimos anteriormente en la introducción a la triangulación de Delaunay, un algoritmo que solo maximice la satisfacción obtenida no es justo al existir agentes que nunca descansarán o recibirán la menor de las retribuciones.

Algoritmo de emparejamiento

Este algoritmo puede utilizar tanto distancias euclídeas como geodésicas. Sin embargo, la parametrización que requiere, así como su baja flexibilidad, le convierten en un candidato perfecto para no realizar su implementación.

El algoritmo de emparejamiento surge como alternativa a la triangulación de Delaunay para implementar las restricciones geográficas.

En términos genéricos, el algoritmo realiza un emparejamiento de las ambulancias dos a dos. Teniendo en cuenta como criterio de emparejamiento unos mínimos de distancia entre ellos, para cada pareja, se elige cuál de ellos puede descansar y cuál debe mantenerse alerta para cubrir su área de influencia y la de su compañero. Si la distancia entre una pareja es muy alta, uno nunca podrá cubrir a tiempo una incidencia producida en el área de influencia del otro.

Estos mínimos no garantizan la existencia de una pareja, por lo que habría ambulancias que nunca podrían descansar.

Sin embargo, un uso muy reducido de este algoritmo sí puede ser válido. Como por ejemplo, en sustitución del algoritmo de *Asignación masificado*.

Algoritmo de reasignación individual

Cuando una ambulancia interrumpe su periodo de descanso para atender a un paciente, es posible que este pueda ser atendido 'in situ' y dado de alta. Si este proceso ha requerido poco tiempo y es inferior al tiempo de descanso que le quedaba por consumir, la planificación realizada sigue siendo válida.

Este algoritmo reevalúa la situación actual, y si la reasignación del descanso cumple los criterios definidos, entonces se valida la asignación. Esta validación es requerida ya que la nueva posición de la ambulancia ha podido cambiar la topología.

Este algoritmo modela la situación actual de los descansos en el SUMMA112, de manera que no solo se retoma el descanso tras cada incidencia, sino que además cumple con las restricciones impuestas.

Algoritmos sin restricciones geográficas

Los algoritmos sin restricciones geográficas han sido añadidos para i) la realización de comparativas con los algoritmos con restricciones geográficas, pudiendo realizar una mejor evaluación de los mismos, y ii) como algoritmos auxiliares y de apoyo al sistema de coordinación.

Algoritmo básico

Este algoritmo realiza una asignación FIFO para la hora establecida y permite la simulación de una asignación aleatoria si se baraja el orden de las ambulancias a asignar.

Asignación en orden de satisfacción

Este algoritmo realiza la asignación empleando la satisfacción actual de los agentes.

Se ordenan las ambulancias por satisfacción, de menor a mayor y, para la hora estipulada, se realiza el número de asignaciones definido.

Es la variante sin restricciones geográficas de *Asignación basada en satisfacción*.

Asignación en orden de preferencias

Este algoritmo utiliza para la asignación el mecanismo de ordenación a partir de las preferencias para las reasignaciones del algoritmo de *Asignación con reasignación*.

Asignación por ganancia de satisfacción

Este algoritmo realiza los siguientes pasos:

1. Calcula el valor de ganancia de cada una de las ambulancias del siguiente modo:
 - a. Toma el valor de la satisfacción actual
 - b. Calcula el valor de la satisfacción final si se produce una asignación de esta ambulancia para la hora H por una duración D.
 - c. La ganancia es la diferencia entre b y a.
2. Se ordenan las ambulancias de mayor a menor valor de satisfacción
3. Las N primeras ambulancias de la lista serán asignadas para el momento H, siendo N el número de asignaciones disponibles y H la hora de la asignación.

El uso de la ganancia propone un modelo justo de asignación al emplearse funciones de satisfacción acotadas. De este modo, una ambulancia con un valor de satisfacción 1 nunca podrá elegir la primera, mitigando el problema de la distinción de clases cuando el escenario o el modelo realizado del mismo no contenga la igualdad de los agentes como parte de su definición.

Encadenamiento algorítmico

El coordinador de descansos puede calibrarse para utilizar uno u otro método de decisión. Estos métodos pueden utilizar varios algoritmos de asignación de forma encadenada.

Actualmente, los dos métodos más empleados son:

- a. Asignación basada en satisfacción + Algoritmo con reasignación + Algoritmo de ocupación masificada.
- b. Algoritmo con reasignación.

Consideraciones y compromisos

Descanso mínimo

Los descansos de corta duración son contraproducentes, ya que crean inseguridad en el personal médico al no poder realizar tareas que requieran tiempo. Como realizar una comida de menú o dejar la ambulancia para algo más que “estirar las piernas”.

El descanso mínimo supone un compromiso con la asignación de un descanso y una mejora para el personal médico. Está planteado para dos posibles escenarios:

- i. Cuando una ambulancia ha visto su descanso fraccionado debido a las interrupciones, es posible que el tiempo restante hasta la consecución de los sesenta minutos sea despreciable. Como ante una planificación perfecta esto no debería ocurrir, como compensación y cumpliendo el criterio de descanso mínimo, podrá disfrutar de un descanso de dicha duración.
Por ejemplo, si habían cubierto cincuenta de los sesenta minutos de descanso y el descanso mínimo está estipulado en veinte, al producirse el nuevo descanso habrán disfrutado de diez minutos extra.
- ii. Ante una incidencia, si la ambulancia más cercana está comiendo, se pueden tomar dos decisiones:
 - a. Se la interrumpe porque es la que cubrirá con mayor premura la incidencia.
 - b. Se busca otra ambulancia disponible que pueda atender la incidencia en un tiempo razonable de tiempo.

Con la introducción del descanso mínimo, se pueden integrar ambas decisiones, de modo que si la ambulancia lleva descansando menos del periodo mínimo, ésta solo será interrumpida si su implicación supone salvar la vida del paciente.

Si ya ha pasado el periodo crítico de descanso, será alertada de la incidencia evitando el desequilibrio de la cobertura y el gasto adicional de combustible.

Acumulación del tiempo de descanso

La acumulación del tiempo de descanso puede realizarse tanto a nivel de break como a nivel de turno. Así, si para un periodo de tiempo no ha podido realizarse el descanso o se ha visto reducido por alguna incidencia, éste podrá ser tomado en un periodo futuro.

Ante una política de no horas extra, sería una buena solución de equilibrio. Esta noción de acumulación cubre las dos caras de la moneda: si no se ha descansado el tiempo estipulado, este se añadirá para el siguiente periodo, y si por una asignación de descanso mínimo se ha sobrepasado el tiempo estipulado, este se verá reducido en el próximo periodo.

La introducción de uno u otro, ambos o ninguno, sería una elección administrativa.

Restricciones adicionales: horas permitidas

Como es sabido, ciertos períodos del día son más propensos a la aparición de incidencias. Esto puede incluir eventos extraordinarios como cabalgatas. La restricción de las horas de descanso a unas horas concretas es un aditivo que cubre precisamente esta necesidad.

Esta restricción debe planearse con cautela ya que pueden no quedar suficientes periodos de descanso para que sean repartidos según los criterios establecidos entre las ambulancias.

Autoajuste: afinamiento de los parámetros algorítmicos en tiempo real

Ante situaciones críticas, el coordinador de descansos se puede configurar para que evite en la medida de lo posible que una ambulancia nunca sea asignada. Como vimos anteriormente, esta imposición implicaba la relajación de las restricciones geográficas y la reducción de la satisfacción al reasignar, así como un aumento drástico de las interrupciones.

En consecuencia, este valor de ajuste, establecido desde la configuración como un valor estático, debe gestionarse con cuidado y el módulo de asignación tiene la capacidad de modificarlo dinámicamente.

Esta modificación es necesaria debido a dos factores:

- La existencia de turnos no homogéneos, como el de veinticuatro y doce horas, en los que los breaks son de ocho y seis horas respectivamente. En este caso, no se puede aplicar el mismo valor de ajuste.
- Las diferencias en utilización de los servicios de urgencia, por ejemplo entre el día y la noche. Por la noche habrá menos incidencias y sería más productivo un valor de ajuste más permisivo.

En la implementación actual, este valor de autoajuste se actualiza a posteriori en base a los resultados obtenidos. De modo que, si al finalizar un break existen ambulancias que no han podido descansar, el valor aumenta y viceversa.

El valor debe ser sensible a los cambios y aumentar su valor rápidamente. Por otra parte, esta disminución no puede ser rápida, ya que el objetivo es llevar a cero el número de no asignaciones, y esto puede producir un comportamiento sinusoidal.

La función de ajuste escogida que modela este comportamiento es la siguiente:

Sea N el número de ambulancias no asignadas, M el número de horas por cada break y D la capacidad en porcentaje de la triangulación de Delaunay. Entonces:

$$Fine\ Adjust(t) = \begin{cases} Fine\ Adjust(t - 1) + \text{ceil}\left(\frac{N}{M * D}\right) & \text{if } N > 0 \\ Fine\ Adjust(t - 1) - 0.5 & \text{if } N = 0 \end{cases}$$

$$Adjust\ Value(t) = \text{ceil}(Fine\ Adjust(t)), \in \mathbb{Z}$$

Siendo $FileAdjust$ la función asimétrica de ajuste y $AdjustValue$ la función responsable de actualizar el valor de ajuste, que es un número entero.

La nueva implementación, todavía no testada, lleva un paso más allá este ajuste dinámico. En lugar de actuar a posteriori en base a las interrupciones producidas, actúa a priori haciendo una predicción de estas interrupciones.

El coordinador de posicionamiento cuenta con un modelo de pacientes, el cual ofrece una predicción aproximada de dónde y en qué cuantía surgirán pacientes. Se puede utilizar este modelo para pronosticar la probabilidad de interrupción.

La utilización de este modelo se puede aplicar a tres niveles, cada uno de ellos más dependiente de la calidad del mismo:

- I. La probabilidad de interrupción es un único valor para todo el break o turno, se obtiene como una simple fracción entre el número de incidencias partido tiempo.
- II. La probabilidad de interrupción es diferente para cada hora de asignación, modelando horas críticas y horas de bajas incidencias como la noche.
- III. La probabilidad de interrupción es única para cada ambulancia y hora, quedando reflejada tanto la probabilidad de aparición de pacientes en un intervalo breve de tiempo como en un espacio acotado. El modelo de predicción requerido para el correcto funcionamiento debe ser muy preciso.

Partiendo del conocimiento de estas probabilidades y de la disposición de las ambulancias, se puede aplicar la teoría básica de coloreado de mapas [JT11].

El coloreado de mapas es un problema de optimización en el cual dos regiones fronterizas no pueden compartir el mismo color, y el objetivo es determinar el número mínimo de colores necesarios para colorear el mapa y el color de cada región.

Este número mínimo de colores corresponde en el problema que nos atiende al número mínimo de horas necesarias. Siendo el grafo a colorear propia la triangulación de Delaunay.

Con este modelo predictivo se puede estimar el número de ambulancias que no serán asignadas. Por ejemplo, si disponemos de ocho horas, pero el número de colores mínimo es diez, significa que la cantidad de ambulancias sin asignar será la cardinalidad de los conjuntos correspondientes a los dos últimos colores.

Este valor es el que emplearemos en la función de ajuste anteriormente descrita.

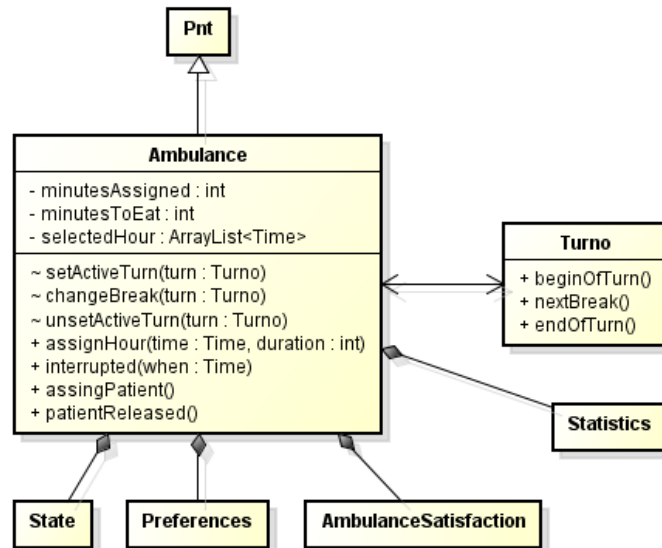
En este contexto, no es necesario calcular exactamente el número mínimo de colores, ya que un valor más alto puede considerarse como una medida de seguridad, pero sí es recomendable estar lo más próximo a él.

Recordar que la distribución de las ambulancias varía a lo largo del tiempo, y si se modela de manera más precisa también su número. Además se están aplicando modificadores probabilísticos al coloreado.

Implementación

Diseño monolítico

Desde un primer momento, y pensando en una rápida implementación y evolución del modelo, se ha planteado una arquitectura monolítica. Todos los elementos necesarios para el funcionamiento de los algoritmos de coordinación están contenidos en la entidad Ambulancia. En el siguiente diagrama puede observarse esta construcción todo en uno.

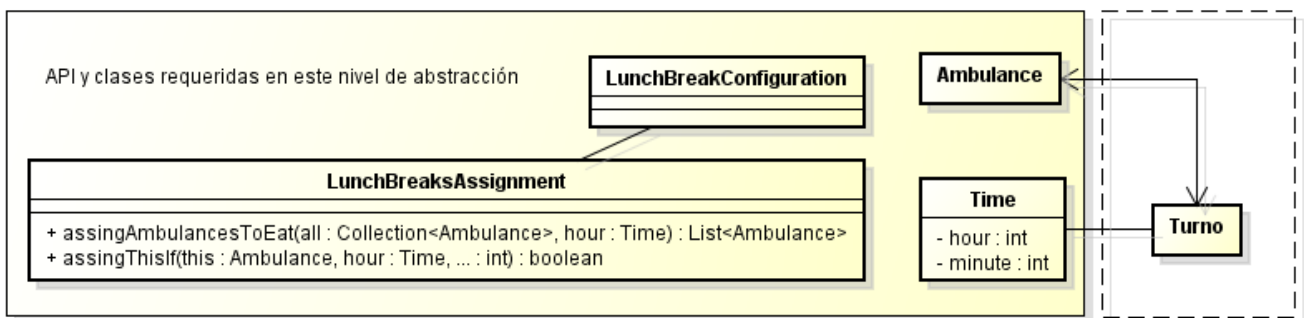


Diseño a dos niveles

Haciendo una correspondencia directa con los hitos establecidos, se ha planteado un diseño a dos niveles.

El primer nivel es el núcleo del sistema coordinador de descansos, incluye los algoritmos de decisión, los parámetros de configuración y las abstracciones que modelan el problema.

El segundo nivel define al sistema como un módulo autónomo, aportando la capacidad de auto-ejecución y auto-ajuste.



API de nivel 1

En esta ilustración podemos ver la API propuesta por el nivel 1 del sistema de coordinación. Consta únicamente de dos métodos:

- assignAmbulancesToEat: tras este método se encuentra toda la maquinaria de decisión del coordinador de descansos. Para todas las ambulancias del problema, se realiza una planificación completa.
- assignThisIf: tras este método se encuentra el *algoritmo de reasignación individual*.

Los parámetros del algoritmo pueden encontrarse en el Apéndice B.

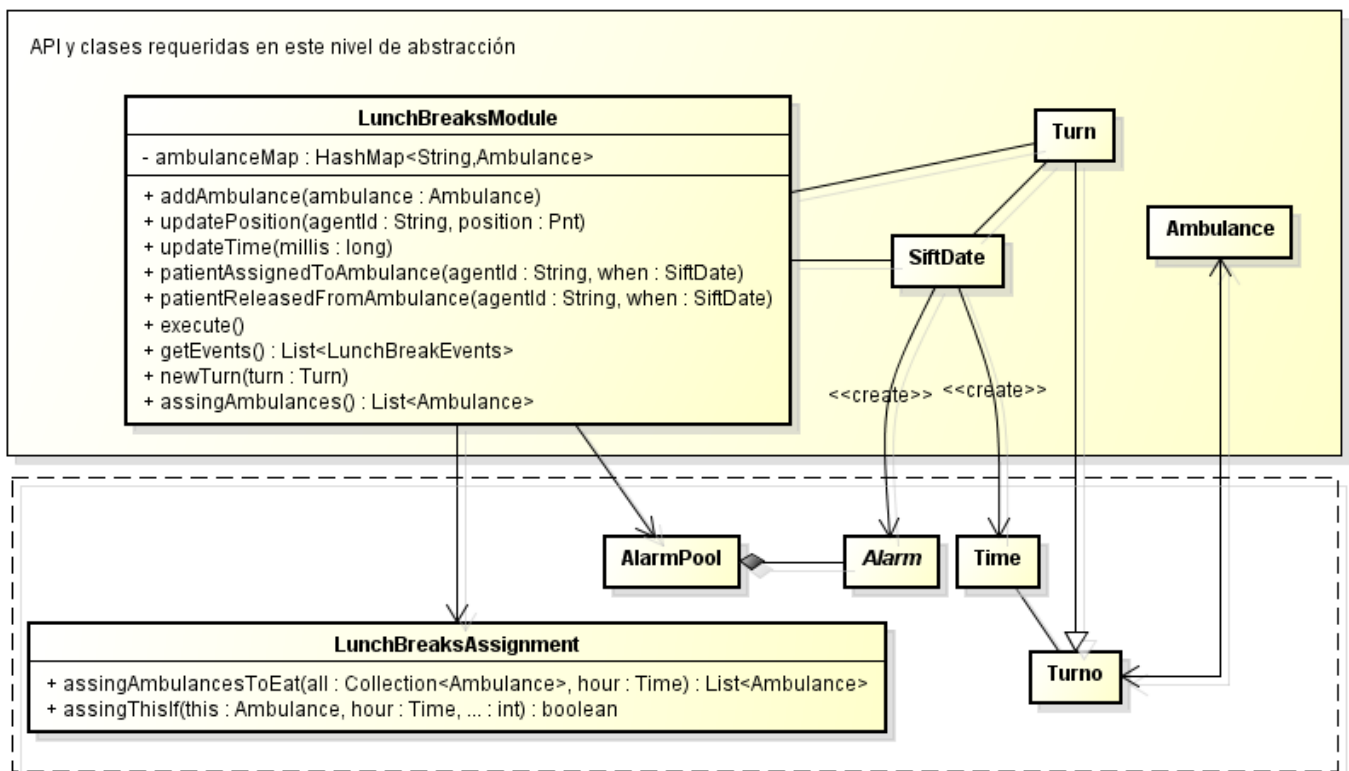
El modelado del entorno corre a cargo de la entidad que use este nivel de abstracción. Para ello, debe definir:

1. Las ambulancias existentes, incluyendo el estado, las preferencias, y todos los elementos que conforman esta clase.
2. Los turnos, los cuales aportan la división horaria, las horas de asignación permitidas y una agrupación lógica de las ambulancias.
3. La relación entre las ambulancias y los turnos a lo largo del tiempo.

Como el mecanismo está diseñado para aportar únicamente las asignaciones para una hora concreta, la planificación explícita de todo el periodo de trabajo requiere la reevaluación por parte del coordinador. Para esta reevaluación se requiere:

1. Establecimiento de las horas a las que se reevaluará el algoritmo.
2. Si el periodo de planificación es mayor de un turno, actualización del vínculo entre las ambulancias y el turno actual.

Como se puede observar, el uso del coordinador no requiere la actualización del estado de la ambulancia. Esto se debe a que las decisiones tomadas por el coordinador son aplicadas *in situ*.



API de nivel 2

Este nivel de abstracción incorpora las competencias colaterales de la gestión de los turnos. Además, debido a que está orientado a ser utilizado por el mHealth Demonstrator, su arquitectura debe ser compatible con la del demostrador. Esta compatibilidad se ha visto realizada con la utilización de un sistema auto-ejecución autónomo basado en eventos y tareas programadas.

La gestión de los turnos requiere el conocimiento de horas reales, de modo que se pueda planificar un turno para las ocho de la mañana del lunes, otro del martes, otro solapado a partir de las doce, etc. Además de poder requerir la planificación de más de un día.

Por ello, es intrínsecamente necesario la utilización de una medida temporal real: la fecha, o *datetime* en términos de programación.

Este *datetime* ha sido plasmado en la clase *SiftDate*, que es una clase *Date* normal de Java pero que gestiona el uso de métodos obsoletos y carece de segundos y milisegundos para una compatibilidad directa con la clase *Time*.

Tras este sistema de tiempo real, se encuentra un sistema de tareas basado en alarmas.

El nivel de abstracción 2 proporciona todos los mecanismos necesarios para una utilización desvinculada de la lógica de negocio. Requiere para su utilización:

1. La inclusión de las ambulancias en el sistema.
2. La inclusión de los turnos en el sistema. La gestión y activación del turno es competencia del módulo.
3. La ejecución del módulo, que comprende tres métodos:
 - a. *updateTime*: sincroniza el sistema de tareas interno.
 - b. *execute*: realiza todas las tareas que el mecanismo de alarmas ha marcado para su ejecución.
 - c. *getEvents*: como es un módulo cerrado y la entidad ambulancia es un recurso interno del mismo, el conocimiento de la toma de decisiones está plasmado en un modelo basado en eventos.

Diseño basado en repositorio

Desde esta arquitectura de implementación, que se puede encontrar en el espacio de nombres *lunchbreaks*, se ha realizado una migración parcial hacia un sistema menos acoplado basado en repositorio. Esta nueva implementación puede encontrarse en el espacio de nombres *lunchbreaks2* y consta de las siguientes características:

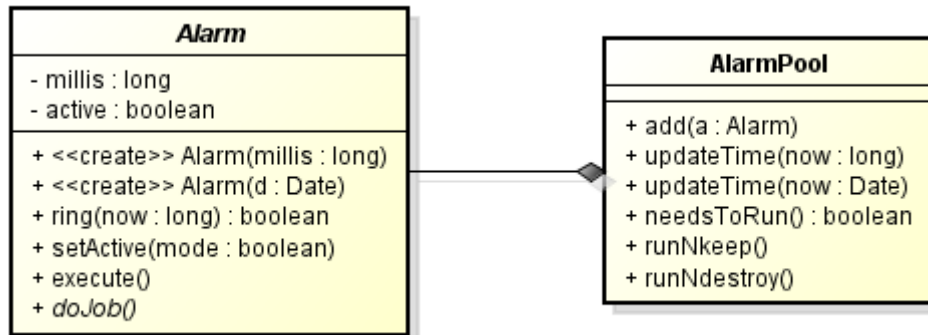
- a. Reducción del acoplamiento entre la clase Turno y Ambulancia, al establecer el vínculo que las une como una referencia externa, o dicho en lenguaje de base de datos, *foreign key*.
- b. Agrupación de los algoritmos por contexto y fraccionamiento de los mismos para que solo realicen una acción. Esta readaptación visual de los algoritmos es peligrosa por dos motivos:
 - a. El algoritmo monolítico y el fraccionado deben ser idénticos para que su reemplazo no suponga ningún cambio en el comportamiento del demostrador y, por tanto, sigan teniendo validez todas las pruebas y experimentos realizados con anterioridad.
 - b. Cada agrupación algorítmica, especialmente en el caso de *DelaunayAssignment*, proporciona un contexto, y por tanto, si se sale del contexto y se vuelve a entrar en él, hay que proporcionar mecanismos para persistirlo o al menos, dejar bien definido el comportamiento. Por ejemplo, si estando en el mismo o en otro contexto cumple la idempotencia o no, siendo esta respuesta el estándar para el desarrollo de cualquier método posterior.
- c. Externalización de los datos y recuperación de los mismos a través de conectores, afrontando ya un diseño orientado a una base de datos real.
- d. Extracción de las competencias adquiridas de gestión de turnos a un módulo externo, *ShiftManagement*. Este módulo se acopla al demostrador como todos los demás y mediante los canales de comunicación proporcionados por el mismo se mantiene la consistencia del modelo.

Esta nueva arquitectura se encuentra en una versión preliminar, aunque ya es funcional. Y ha permitido un mejor conocimiento acerca del nivel de dependencia conceptual y acoplamiento a nivel de código como punto previo al diseño del prototipo final orientado a su implantación.

Alarmas: un gestor de tareas en tiempo real

El sistema de alarmas es una arquitectura de planificación y ejecución de tareas y es el eje central de los módulos de asignación y de gestión de turnos. Alcanzar esta solución y formalizar su diseño no fue un proceso trivial.

El sistema de alarmas se compone de la entidad alarma y la entidad responsable de su ejecución.



La clase *Alarma* proporciona el contexto necesario para la planificación de tareas, y permite formalización de cualquier tarea a través de su método *doJob*.

La clase *AlarmPool* es el contenedor de alarmas y encargado de la evaluación y ejecución de las mismas. Este contenedor permite dos comportamientos: i) *runNkeep*, en el que tras la evaluación y decisión de ejecución de una alarma ésta sigue en la cola de tareas y siempre será ejecutada en cada nuevo paso de ejecución, ii) *runNdestroy*, en el que cada alarma es retirada de la cola de tareas si su evaluación es positiva y va a ser ejecutada.

En el modo *runNdestroy* se permite la recursividad a través de la reinserción de nuevas alarmas en caliente³, de modo que una tarea puede ser la precursora de sus sucesoras.

Se puede emular cualquier escenario de recursividad: lineal, exponencial, factorial, y es extremadamente simple realizar una recursión infinita. De modo que, hay que saber que el criterio de parada por antonomasia de este sistema de tareas es la actualización, también el caliente, de la fecha actual a un tiempo pasado en el que no existía ninguna tarea.

Si se ha acordado que la tarea semilla está planificada en el step cero, el criterio de parada será -1.

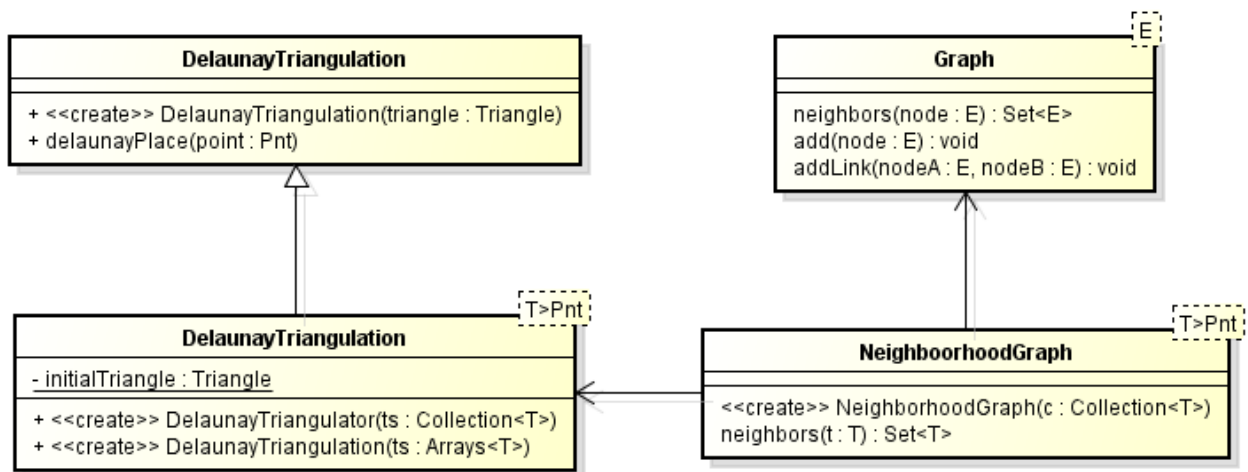
³ La reinserción en caliente es la planificación de una nueva tarea dentro de la definición de otra. De modo que su inclusión en la cola de tareas se producirá mientras esta se está evaluando. Por tanto, deben proporcionarse mecanismos de control que aseguren el mismo comportamiento del encolamiento de una tarea tanto en frío como en caliente.

Triangulación de Delaunay: envoltura de la implementación existente

En el código realizado para el módulo de posicionamiento ya existe una implementación para realizar una triangulación de Delaunay. Esta emplea un triángulo inicial y todos los puntos deben estar contenidos en él. Por tanto, el grafo solución de la triangulación incluirá tres puntos más.

En el contexto de vecindad este comportamiento no era deseable. En consecuencia, se creó una clase por encima de ésta que se encargase de los pormenores, ofreciese una mejor experiencia, incluyendo excepciones concretas, y permitiese el uso de cualquier clase elemento.

Este último punto es de vital importancia, al haber escogido un modelo centralizado de los datos, evitando la traducción al concepto Punto⁴ y viceversa⁵.



El uso de este envoltorio se debe a la restricción de no poder alterar la implementación existente. Su inclusión ha permitido una implementación casi trivial del grafo de vecindad y la abstracción de la clase Triangle.

⁴ La clase Punto, o Pnt, contiene la información de localización de la ambulancia en un formato de coordenadas UTM, al ser un espacio de coordenadas euclídeo.

⁵ Ya había sido alertado de los problemas que este mapeo abajo-arriba había ocasionado. Y es el motivo por el que dentro de la clase punto, sirviendo de precedente, puede encontrarse un String para su posterior identificación.

Ficheros de configuración

Los ficheros de configuración permiten realizar diferentes experimentos sin necesidad de modificar el código. Existen tres ficheros de configuración, encargados de la configuración de los parámetros del coordinador de descansos, de la definición de los turnos que formarán parte del experimento y de especificación de las preferencias de cada ambulancia.

La estructura, carga y almacenamiento de estos ficheros se ha realizado con *Xstream* [xSt].

En el Apéndice C puede encontrarse el fichero de configuración responsable de modelar los turnos.

Integración en el mHealth Demonstrator

Modificaciones requeridas

El diseño del segundo nivel del algoritmo está orientado a su integración casi inmediata con el demostrador. Las modificaciones requeridas no han ido más allá de la introducción de interrupciones en el coordinador de descansos.

Coherencia de los estados

El demostrador tiene su propia representación de las ambulancias y de su estado. Por tanto, es necesario establecer los mecanismos de coherencia para sincronizar el estado de las ambulancias entre ambos lados.

Cada representación tiene sus propios estados específicos, por lo que la sincronización no es directa.

En la siguiente tabla hay un mapeo de los estados de ambos lados y su correspondencia.

Paso de mensajes

El paso de mensajes es el canal de comunicación establecido en la arquitectura mHealth para la comunicación entre los módulos con el demostrador u otros módulos.

En las siguientes tablas se describen los mensajes enviados y recibidos.

Mensajes Recibidos	Origen	Función
OccupationMensajeChanged	Agente Ambulancia	Este mensaje alerta del cambio del estado de ocupación de una ambulancia. Determina si a una ambulancia se le ha asignado un paciente o ha terminado con él.
WorkStatusChangedMessage	Shiftmanagement	Este mensaje alerta de un cambio en el estado de un turno sobre una ambulancia. Puede ser: inicio del turno, fin del turno o cambio de break.

Mensajes Enviados	Destino	Función
LunchBreakAssignmentMessage	Agente Ambulancia	Este mensaje notifica al agente que modela el comportamiento de la ambulancia la asignación de un descanso para que actualice su estado interno.

Experimentos

En este apartado se incluye un breve resumen de los resultados de los experimentos realizados.

Minutos descansados

En la siguiente tabla se recoge una batería de experimentos realizados sobre el día crítico cambiando algunos de los parámetros de configuración.

Experimento	Duración	Nº de pacientes atendidos	Minutos asignados	% de minutos	Nº de ambulancias no asignadas	Satisfacción por descanso	Satisfacción por trabajo	Satisfacción por interrupciones
Exp1 A	24h	223	5199	99.5977	0	0.7362	0.9655	0.919
Exp1 C	24h	223	4980	95	0	0.59	0.96	0.93
Exp2 A	24h	223	5239	100.36	0	0.76	0.96	0.92
Exp2 A+	24h	228	5156	98.7	0	0.76	0.96	0.929
Exp2 B	24h	223	5061	96.9	2	0.65	0.96	0.91
Exp2 C	24h	223	4938	94.5	0	0.61	0.96	0.93
Exp2 C+	24h	225	5062	97	1	0.59	0.96	0.93
Exp4 A	24h	223	5023	96	3	0.70	0.96	0.91
Exp4 C	24h	223	4516	86	8	0.64	0.96	0.94
Exp5 A	24h	223	5217	99.9	1	0.75	0.96	0.92
Exp5 C	24h	223	4516	86.5	8	0.64	0.96	0.94

Leyenda

+	Todos los módulos activados		Solo LunchBreaks activado
---	-----------------------------	--	---------------------------

A	Preferencias en orden creciente	B	Preferencias gaussianas	C	Preferencias en orden descendente
---	---------------------------------	---	-------------------------	---	-----------------------------------

Exp1	
LBA__MAX__GLOBAL__EATING	8
LBA__MIN__GLOBAL__REMAIN	10
LBA__SKIP__DT__LIMITATION	True
LBA__ADJUST__UNASSIGNED__REASSIGNATION	4
LBM__DINAMIC__ADJUST	False

Exp2	
LBA__MAX__GLOBAL__EATING	10
LBA__MIN__GLOBAL__REMAIN	10
LBA__SKIP__DT__LIMITATION	True
LBA__ADJUST__UNASSIGNED__REASSIGNATION	4
LBM__DINAMIC__ADJUST	True

Exp4	
LBA__MAX__GLOBAL__EATING	10
LBA__MIN__GLOBAL__REMAIN	14
LBA__SKIP__DT__LIMITATION	false
LBM__DINAMIC__ADJUST	false

Exp5	
LBA__MAX__GLOBAL__EATING	10
LBA__MIN__GLOBAL__REMAIN	14
LBA__SKIP__DT__LIMITATION	True
LBA__ADJUST__UNASSIGNED__REASSIGNATION	2
LBM__DINAMIC__ADJUST	False

Como se puede observar, el experimento 1 y el 2 son los que mejores resultados han obtenido en términos de cantidad de ambulancias asignadas, así como de tiempo descansado.

Los parámetros de *Exp1* son los mejores para una situación con 29 ambulancias.

El experimento uno es el que tiene los parámetros mejor ajustados. Es virtualmente igual que el dos, salvo por el ajuste dinámico. Como el turno está compuesto por tres breaks, el ajuste dinámico reduce el valor para el último periodo provocando el fallo de asignación. Este fallo es necesario, ya que es la única manera de maximizar la satisfacción.

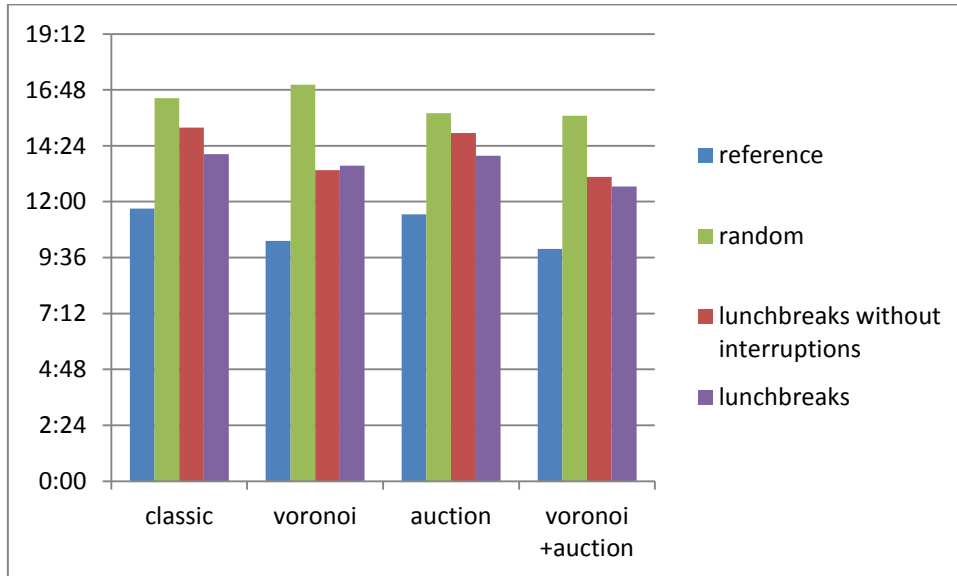
El experimento cuatro es el que peores resultados obtiene de todos. El motivo es doble: el no ajuste dinámico y la no rotura de la triangulación⁶.

Los parámetros presentados son sólo algunos de los disponibles. Hay muchos más y se ha requerido bastante tiempo en calibrarlos. La lista completa de los parámetros y su descripción puede encontrarse en el Apéndice B.

⁶ Se llama rotura de la triangulación cuando se realizan asignaciones que no cumplen con las restricciones definidas por la triangulación. El responsable de este comportamiento es el algoritmo de masificación de la ocupación.

Tiempos de asistencia a paciente

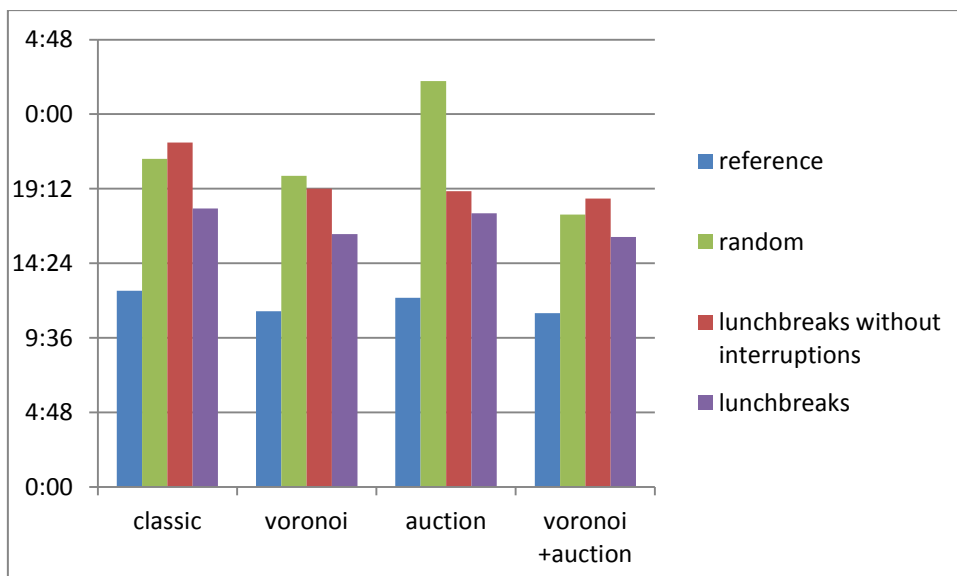
Los experimentos tomados son las configuraciones de veintinueve y veintiún ambulancias para el día crítico.



29 ambulancias

Los resultados entran dentro de lo esperado. La mejor opción implica configurar al coordinador de descansos con interrupciones y formado por los tres algoritmos de asignación principales: asignación, reasignación y masificación de ocupación.

Sin embargo, en el caso de voronoi es un poco peor. Esto se debe a que la existencia o no de interrupciones altera la topología de voronoi, siendo ésta mejor o peor para la disposición de los pacientes del experimento ejecutado.



21 ambulancias

Como es lógico, al haber menos recursos, la diferencia en tiempo es bastante mayor y esta se nota más en el caso de una asignación aleatoria al incidir de manera más crítica.

El resultado anómalo se puede encontrar en el experimento con auctions⁷, aunque cabe la duda si en los otros casos la distribución aleatoria está siendo perjudicial o no.

Como el algoritmo aleatorio tiene interrupciones, éste se comporta mejor en algunos casos. La falta de recursos hace necesaria la fracción de los descansos en periodos más pequeños para mejorar los tiempos de respuesta.

⁷ Auction es el sistema de coordinación de asignación de pacientes basado en subasta.

Análisis del servicio de rutas

Alternativas

Una de las alternativas, implantada y desechada, fue la inclusión en el Servidor Ortega⁸ de un mecanismo de balanceo de carga basado en múltiples IPs. Por desgracia, la cantidad de IPs disponibles era muy reducida y la adquisición de nuevas IPs provocó daños colaterales.

La segunda alternativa, fue el uso de ordenadores ya existentes como granja de IPs. Esta alternativa proporcionaba dos métodos de actuación. La diferencia entre ambos era qué servicio debía alojarse en estas máquinas.

Opción A

La primera opción consistía en la utilización de un proxy web [CTSN02]. Cada máquina de la granja ejecutaría un proxy y el servicio de rutas sólo debería determinar por qué proxy enviar cada petición. Esta vía quedó desechada por un motivo fundamental: los ordenadores empleados serían los disponibles en las aulas de Linux.

A parte de la complejidad y agujero de seguridad provocados por la ejecución de un proxy en el espacio de usuario, estaba el dilema moral de emplear esta granja de proxies con motivos no académicos o maléficos como un ataque de denegación de servicios.

Personalmente, la utilización de este sistema para otros usos, es algo que todo programador acabaría realizando aunque solo fuera para la validación y demostración de las capacidades de su criatura. Así que, finalmente se concluyó que la creación de un servicio genérico de enmascaramiento de IP, y cualquier alternativa de aspecto similar, quedaría completamente prohibida.

Opción B

La segunda opción fue la creación de un programa monofuncional no universal.

En este caso, se planteó la creación de un programa Java, que pudiera reutilizar el servicio actual como su núcleo y cuya comunicación se realizase mediante RMI, en lugar de RESTful.

Dicha opción fue la escogida y será descrita en próximas páginas.

⁸ El servidor Ortega es la máquina destinada al HPC y chapurreo en general del grupo de investigación. En ella se realizan la mayoría de las ejecuciones y simulaciones, así como el despliegue de los servicios, como servidores web, que estas aplicaciones requieren.

Diseño del servicio de rutas

Aunque en conjunto el servicio actual de rutas ha sido completamente modificado, el núcleo del programa, que incluye la decodificación de las rutas de los servicios de Google y MapQuest no se han visto alterados y el mérito se debe exclusivamente a Holger Billhardt.

A continuación, se hace una disección de cada uno de los componentes que forman el servicio de rutas.

Sistema de caché

La inclusión de un sistema de caché completo era un punto determinante en la mejora de prestación de servicio.

Los objetivos principales son:

- Incluir políticas de reemplazamiento
Hasta ahora inexistentes. Era preciso incluir una política mejor que FIFO.
- Modelo de imagen único
Era indispensable diseñar una caché que pudiera contener tanto las rutas de GoogleMaps como de MapQuest, así como de los sucesores.
Esta tarea implica una imagen única de las entradas de rutas, que supone la implicación de los proveedores de rutas y la existencia de un identificador universal.
- Persistencia de datos
Era habitual la reevaluación de los modelos de experimentación. Además, muchos modelos compartían ciertas características, como el modelo de pacientes.
Poder almacenar la caché de un experimento y cargarlo para su reevaluación supondría idealmente una tasa de éxito de la caché del 100%

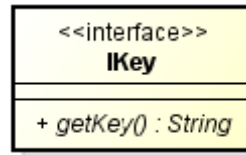
Diseño de una estructura genérica de caché

Una caché no necesita conocer la estructura de los datos que almacena, por tanto su diseño es factiblemente genérico.

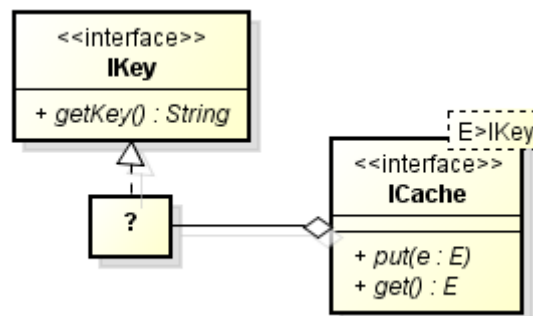
La única invariante en todo sistema de caché es la estructura auxiliar que almacena la etiqueta de la entrada. En cachés con políticas de emplazamiento y reemplazamiento no directas esta estructura auxiliar contendrá los metadatos que necesiten.

Definición de la etiqueta

Para la especificación de la etiqueta se ha escogido un String como contenedor por su flexibilidad y por motivos de depuración. Su longitud indefinida es uno de los principales motivos por los que ha sido elegida en la especificación genérica.



Identificando la funcionalidad básica de una caché, ya tendríamos la estructura básica, o contrato, para especificar cualquier tipo de caché. Y esta sería su representación.



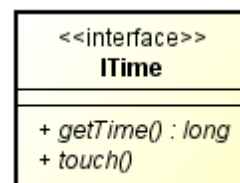
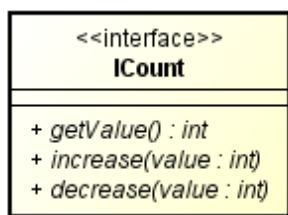
Políticas de reemplazamiento

Con la base definida, es momento de definir las políticas de emplazamiento y reemplazamiento. Nos concentraremos en las políticas de reemplazamiento.

Se han modelado las tres políticas de reemplazamiento más básicas:

- FIFO (First In First Out)
- LU (LessUsed)
- LRU (LeastRecentUsed)

Tanto LU como LRU necesitan metadatos para los algoritmos de reemplazo. La forma de estos metadatos ha sido recogida en los siguientes contratos.



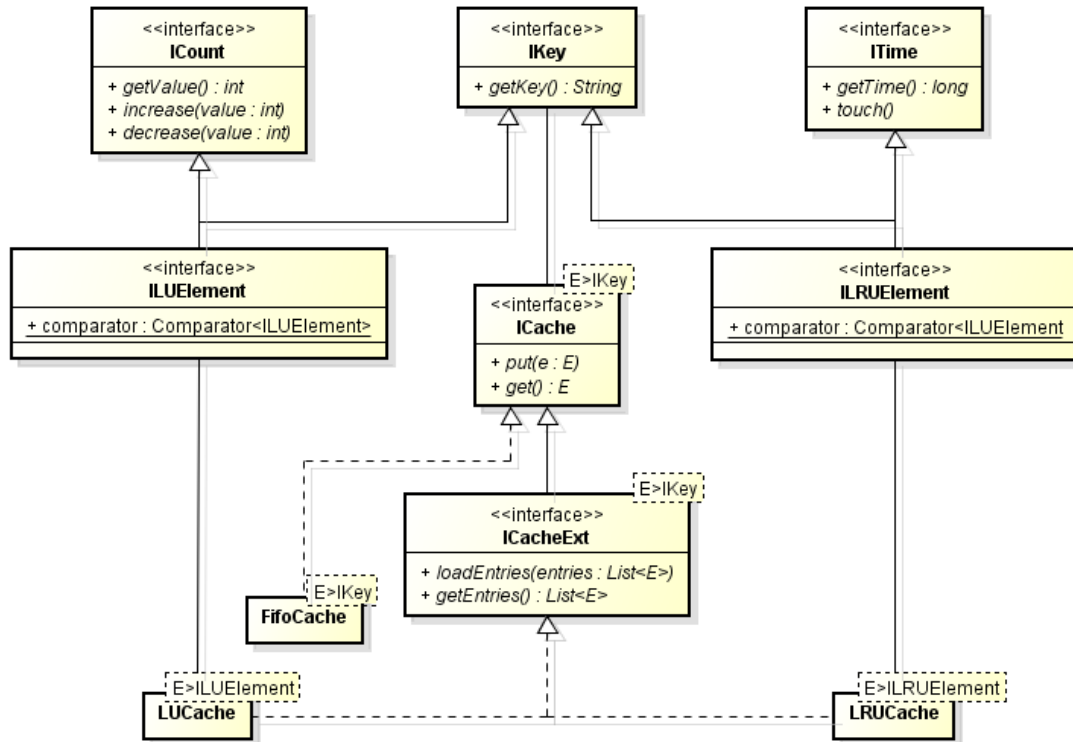
ICount: es la especificación escogida para LU. Esta permite funciones de aumento y disminución de peso asimétricas.

ITime: es la especificación escogida para LRU en su forma más simple.

Arquitectura de la solución

Con los dos puntos anteriores definidos, el resto de la arquitectura puede entenderse como dependiente de la implementación.

En la siguiente ilustración se puede observar la arquitectura.



En esta arquitectura se puede ver el uso de tres contratos más

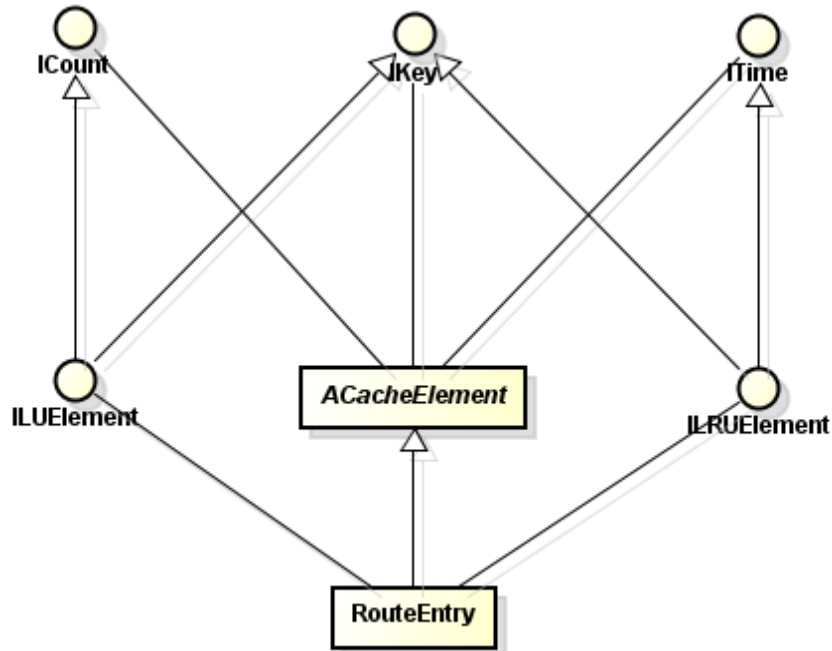
- *ICacheExt*
Este contrato especifica la política de carga y guardado de las entradas de caché para su persistencia.
- *LUElement* y *LRUElement*
Estos dos contratos son requeridos en Java para fusionar los metadatos de las entradas. Además, cumplen una función adicional, la de proporcionar un mecanismo universal para comparar los metadatos al implementar los algoritmos de emplazamiento y reemplazamiento.

Integración con el sistema de rutas

Con el sistema genérico de cachés definido, ya solo queda trasladar esta funcionalidad al servicio de rutas.

La elección de la política de reemplazamiento necesita ser sujeto de estudio, por ello se escogió un sistema que pudiera alternar las políticas aprovechando los datos ya calculados.

Esta es la visión conceptual de este diseño:



ACacheElement define todas las estructuras auxiliares para los metadatos. De este modo, la clase *RouteEntry*, solo requerirá una definición de la etiqueta.

El uso de las cachés LU y LRU se ha hecho explícitamente por motivos de claridad, aunque podría haberse definido implícitamente a través de *ACacheElement*.

Definición de los proveedores

Análisis

Respetando el diseño original se mantuvo la definición a dos niveles de las rutas.

Route: Una estructura que refleja las características de una ruta.

CacheEntry: Un contenedor empleado para almacenar la información auxiliar, como los metadatos de la caché, la información en crudo, analizada y decodificada por etapas más avanzadas en la cadena de responsabilidad, e información temporal, aprovechando el uso de este contenedor para reducir el paso de parámetros.

De modo que, los proveedores trabajaban con entradas de caché, no con la ruta directamente.

Los pasos para obtener una ruta son:

1. Especificar las coordenadas de latitud y longitud de origen y destino.
2. Crear una URL formateada con los parámetros requeridos sobre el servicio web.
3. Realizar una petición web (HTTP/GET).
4. Recibir y almacenar la respuesta.
5. Decodificar la respuesta.
6. Rellenar la estructura Ruta.

Sin embargo, la decodificación de la respuesta, especialmente en GoogleMaps, no es directa.

Los pasos adicionales para este proveedor son:

4. Recepción de la respuesta en formato JSON.
5. Conversión del texto JSON a estructura de Objetos.
6. Recuperación del tiempo y distancia totales de la ruta, ubicados en el sumario.
7. Recuperación de un string con la información codificada de la ruta.
8. Extracción de los datos en crudo mediante un algoritmo de decodificación del string.
9. Recuperación de los datos de la ruta.
10. Rellenar la estructura de la ruta.

La información de estas rutas puede ser utilizada con dos fines:

- Con fines imperativos, en los que a un agente se le ordena desplazarse a un destino concreto y es necesario entregarle el plan de acción.
- Con fines evaluativos, en los que se utiliza la información de distancia y tiempo para la toma de decisiones.

El coste extra de la decodificación de la ruta y su inutilidad en la mayoría de los casos motivó un diseño en dos pasos. En el primer paso, solo se extrae los valores identificativos, útiles para los algoritmos de decisión. En el segundo paso, se decodifica la ruta si estos algoritmos han decidido que es la opción óptima.

Diseño

Tras varios diseños, el ganador fue el que incorporó esta dualidad de los datos en su arquitectura.

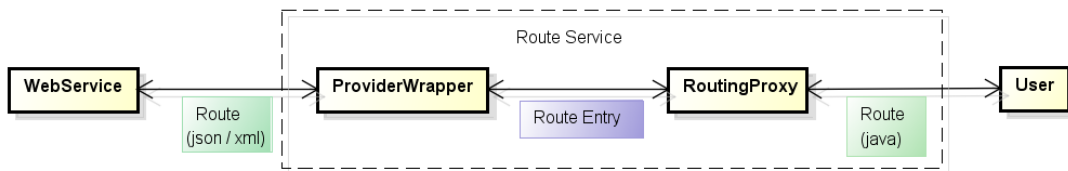
Hay dos agentes implicados para completar el servicio de rutas: el proveedor y el proxy.

El proveedor solicita rutas al servicio web pertinente y lo encapsula en una entrada de caché. Esta entrada es entregada al proxy.

El proxy almacena las entradas en su caché, las cuales guarda celosamente y solo permite el conocimiento de la ruta, quedando de este modo el contenedor de ruta, entrada de caché o *RouteEntry* como un detalle de implementación interno al servicio de rutas.

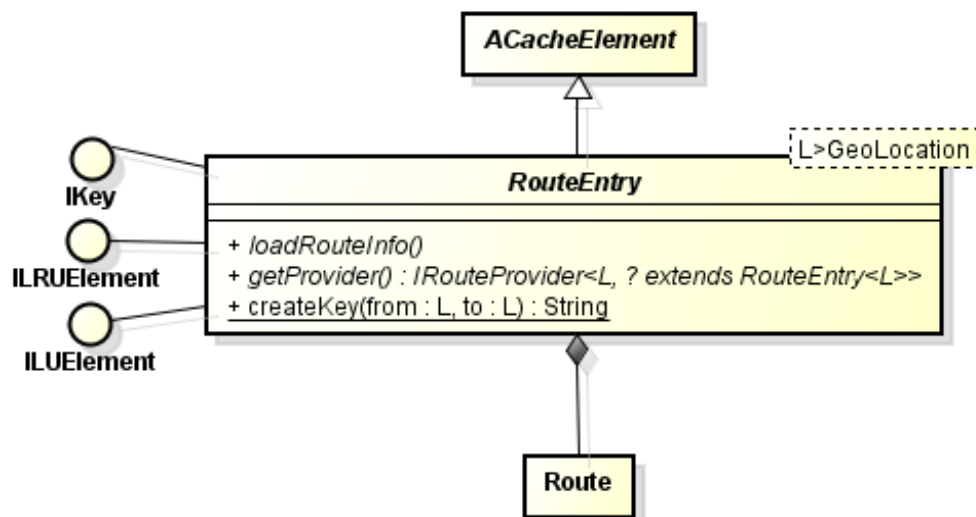
Además, el proxy puede integrar tantos proveedores como desee, proporcionando un punto de entrada único y una caché unificada.

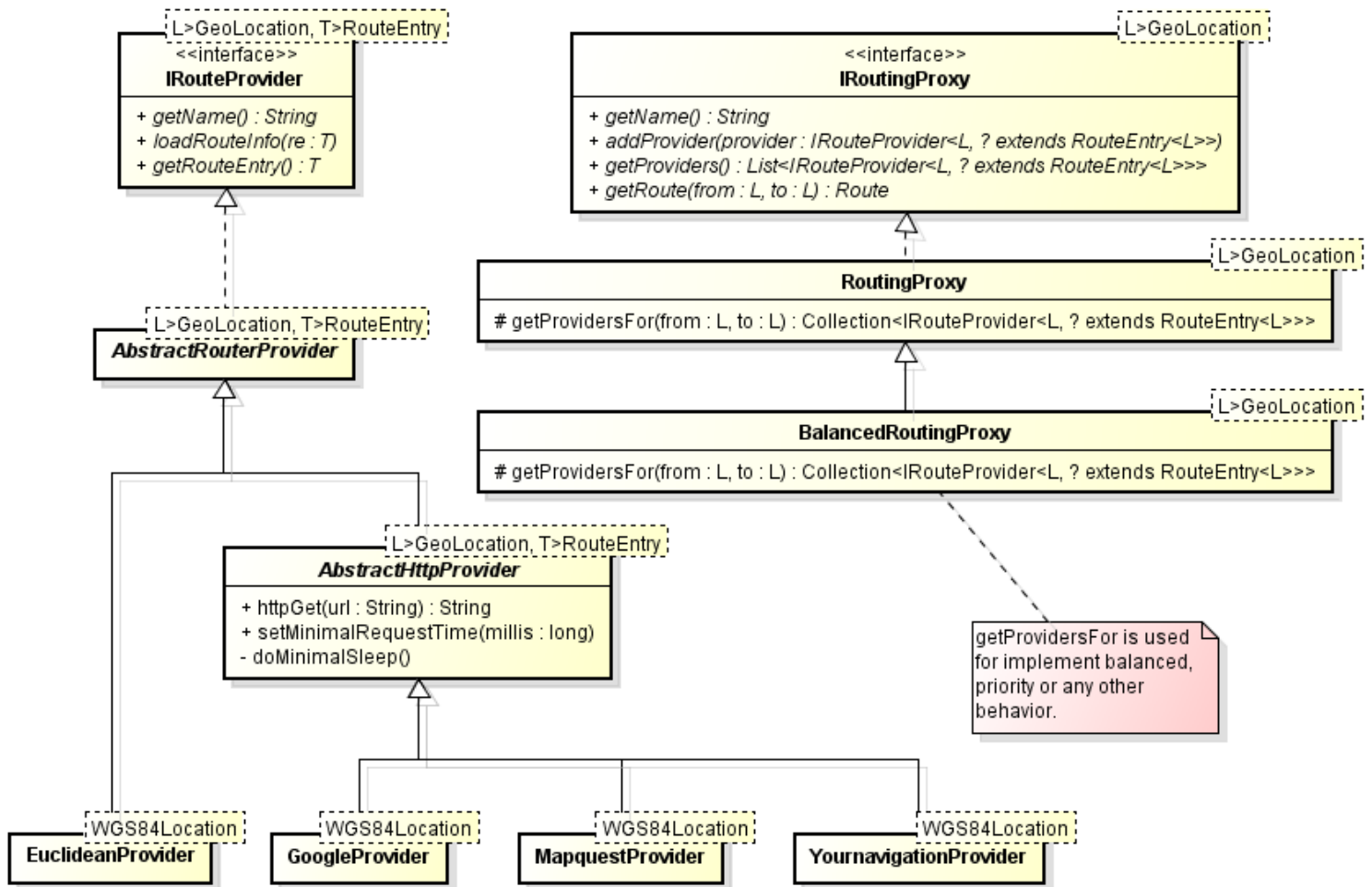
Dicho en palabras sin términos clave es bastante complejo entenderlo. Veamos una ilustración al respecto.



Esta regresión a los orígenes presenta una API más amigable al usuario.

Durante la realización del proyecto, el diseño ha sido mejorado y ampliado. En la siguiente ilustración podemos ver la arquitectura actual.





Vista de la arquitectura

Vista del modelo de datos

Inclusión de dos proveedores nuevos

En la ilustración anterior ya se han introducido los dos proveedores nuevos que serán descritos en esta sección. Estos dos proveedores son *YourNavigation* y *EuclideanProvider*

YourNavigation

YourNavigation es un servicio web orientado al cálculo de rutas. Ofrece tanto front-end web como servicio directo. Su especificación puede encontrarse su página web [YN] y en el código fuente adjunto a esta memoria.

EuclideanProvider

Este proveedor no es un proveedor propiamente dicho. Su inclusión fue necesaria por problemas de zonas muertas.

Existe una región al sur de Madrid en la que ningún proveedor proporciona un servicio fiable. Se trata de un nuevo barrio situado en Parla, al otro lado de la carretera A-42. Esta sección de terreno aún es considerada en obras, por lo que hay carreteras incluidas oficialmente y otras que no lo están. Curiosamente se ha dado de alta una calle inconexa con el resto de la red de carreteras existentes, por lo que cada vez que los algoritmos de enrutado eligen esta calle como punto de partida o llegada, no pueden encontrar ninguna ruta al trabajar con un grafo no conexo.

La API de Google es muy amigable y ante esta situación avisa con un mensaje de error especial, ZERO_RESULTS. Para *MapQuest*, este error es como cualquier otro, indicando el mismo código ya se produzca un error de red, esté la URL mal formateada o cualquier otra cosa. *YourNavigation* sí devuelve una ruta ante esta situación, aunque difiere sustancialmente con la ruta real si esta existiese. Lamentablemente, la resolución de esta ruta aproximada requiere un dispendio de veinte segundos para los algoritmos de *YourNavigation*.

En pos de esta situación y ante la posible existencia de otras zonas muertas, se incluyó un sistema de rutas que nunca podría fallar: la línea recta entre dos puntos.

Esta línea recta, o distancia euclídea, es la ruta que calcula este proveedor y ha sido incluido como salvaguarda ante el peor de los escenarios.

Para entender la importancia de este proveedor de seguridad es necesario explicar que, hasta la inclusión de *YourNavigation* y *EuclideanProvider*, toda experimentación en la que esta zona muerta participase de forma activa no podía continuar y se cancelaba. Esta situación afectó a seis experimentos importantes.

Introducción de jerarquías

Existe una jerarquía simple entre los proveedores de rutas y los proxis. Sin embargo, esta jerarquía no es suficiente para una solución general y extendida.

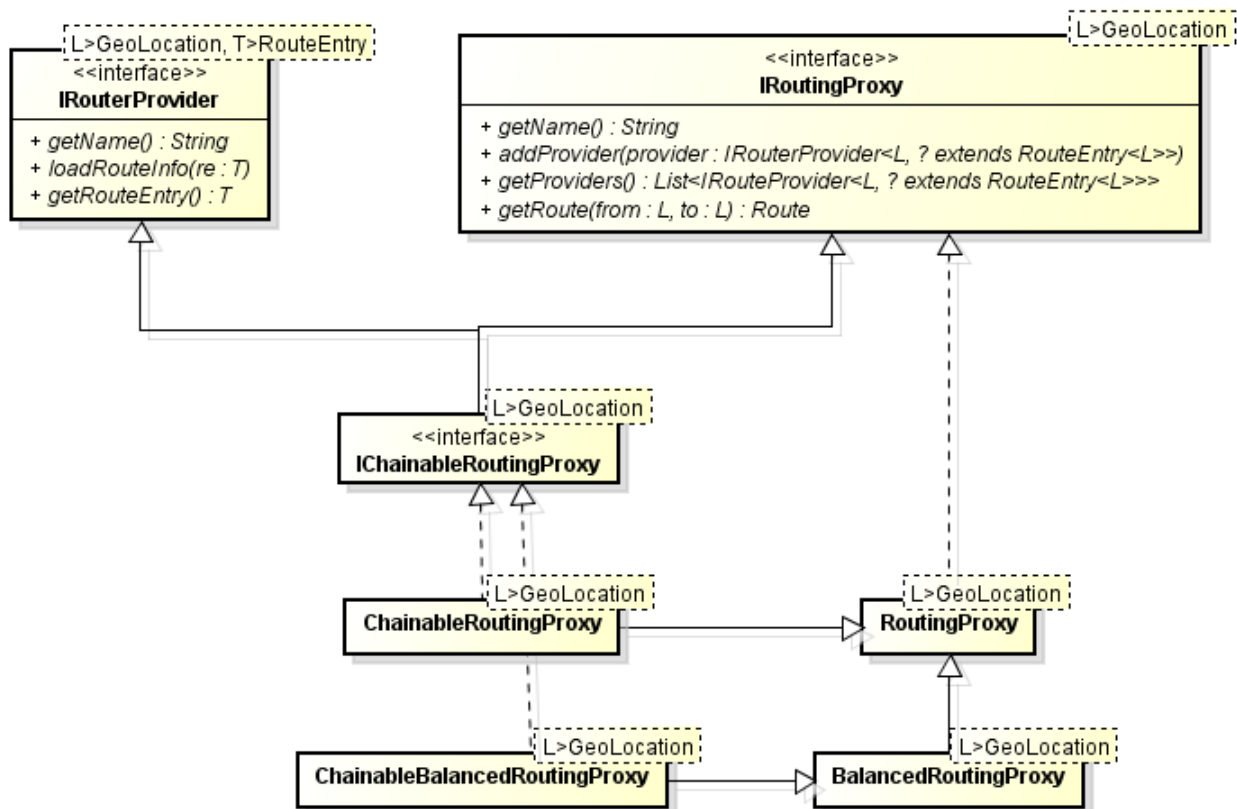
La motivación para la definición de un sistema jerárquico viene dada por las siguientes premisas:

La introducción de una jerarquía compleja permite modelar mayor número de escenarios y de un modo más preciso.

La existencia de nodos heterogéneos otorga flexibilidad y la inclusión de comportamientos a la jerarquía.

Con esta estructura se pueden incluir modelos de predicción, de balanceo de carga, interceptores con caché estática, interceptores de depuración, etc.

Un diseño más complejo requeriría la redefinición de las interfaces actuales y la definición de nuevas interfaces. En consecuencia, en lugar de diseñar una jerarquía genérica y un servicio de rutas orientadas a esta, se ha diseñado una jerarquía orientada al servicio de rutas. Es más práctico, más simple, más directo y así de sencilla es su definición.



La introducción de comportamientos ha sido vital en modelo de despliegue final y en la calidad de servicio ofrecido.

Actualmente, existen dos comportamientos antagonistas orientados a una característica concreta del servicio.

A. Disponibilidad del servicio: el servicio de rutas se considera que está disponible si cumple que:

- a. El servicio responde las peticiones
- b. La respuesta de las peticiones son rutas válidas.

Un nodo diseñado con esta característica empleará a todos los nodos del nivel inferior de la jerarquía para satisfacer esta definición, es decir, el nodo solo fallará la resolución de una ruta si todos sus hijos fallan.

B. Latencia del servicio: el servicio proporcionado debe ser rápido para que compense su uso frente a la solución anterior. Este hecho implica dos restricciones:

- a. El tiempo de respuesta no puede ser mayor de 3 segundos.
- b. Los límites de bloqueo deben ser muy superiores.

Un nodo diseñado con esta característica debe implementar funciones de predicción para aumentar el número de hits de caché e implementa funciones de balanceo para reducir la penalización. En este contexto, con los proveedores web como GoogleMaps, el modo burst sobre ellos, es decir, múltiples peticiones seguidas, sufre mayor penalización por el tiempo obligatorio de espera entre peticiones.

A continuación se describen los nodos definidos.

PriorityNode

Este comportamiento define un orden de prioridad sobre los nodos del nivel inferior. Asegura que, si los nodos están ordenados por calidad, éste la maximizará siempre que sea posible. Responde a un nodo de tipo A.

LRUNode

Este comportamiento asegura la máxima espera de tiempo ante la reutilización de un nodo hijo. Según el número de hijos que emplee para la resolución de una ruta puede ser considerado de tipo A o B. Si emplea todos, será de tipo A y necesitará una condición de parada.

Si no emplea todos, satisfecerá una de las dos restricciones de tipo B. Sin embargo, sin la inclusión de mecanismos de optimización de caché, una petición producirá un fallo de caché cuando existe una copia válida y alcanzable dentro de la jerarquía.

BalancedNode

Este comportamiento asegura un comportamiento eficiente de cada nodo a costa de denegar el servicio. Está diseñado para jerarquías grandes.

Se emplea una función de balanceo como política de predicción y un número limitado de nodos para evitar casos límite, saturación del servicio y baja latencia.

Si la función de balanceo es repetible, es decir, una misma ruta corresponde a un mismo nodo, se optimiza el uso de caché. En términos profanos, para toda entrada de caché idéntica a otra, existe un único camino en la jerarquía que incluye todas las copias.

Responde a un nodo de tipo B.

Introducción del modelo de comunicación

En esta sección nos ocuparemos del diseño de la arquitectura de comunicación. Recordemos que en apartados anteriores ya se definieron las restricciones e ideología a seguir, así como puntos de apoyo en los que asentar la arquitectura de comunicación.

Introducción

Todo canal RMI de comunicación requiere dos componentes:

1. La entidad de descubrimiento y entrega de servicios, en manos del RMI Registry.
2. El componente que proporcionará los servicios.

El RMI Registry es una entidad todo en uno que ofrece el servicio de nombres, y el handshake para el establecimiento del canal de comunicación con un servicio específico. Trabaja sobre TCP con una variante privativa del protocolo de comunicación IIOP.

En la arquitectura RMI se establece un canal exclusivo de comunicación entre cada cliente y servicio. Este canal requiere el uso de puertos TCP en ambos extremos de la comunicación, por tanto, la configuración de seguridad no puede ser muy alta. Es decir, el protocolo no es válido para entornos basados en proxis o con cortafuegos extremadamente restrictivos.

Requisitos de la infraestructura

Con estos principios básicos sobre RMI podemos determinar si la arquitectura a desarrollar podrá ser desplegada en el entorno de desarrollo. Para ello, debe tenerse en cuenta las cuatro entidades implicadas en la arquitectura:

Dispositivo cliente: no está tras un proxy y está permitida la comunicación con el exterior, es decir, ser el cliente o iniciador del handshake en TCP.

Ordenadores de las aulas de Linux: tienen acceso al entorno de usuario y carecen de cortafuegos. En conclusión, pueden alojar servicios.

Servidor Ortega: servidor administrado por el autor de este documento. Requiere la apertura del puerto asociado al RMI Registry, además, de acceso al exterior, es decir, permitir conexiones ya establecidas, para los canales dedicados de comunicación. En términos profanos, actual como servidor para el RMI Registry y como cliente en el resto de comunicaciones.

Infraestructura de red: aunque contiene reglas y la capacidad de bloquear tramas y protocolos concretos, el protocolo RMI está permitido.

Como conclusión al estudio realizado mínimamente reflejado en este apartado, una arquitectura de comunicación sobre RMI es viable.

Diseño

Tras muchas consideraciones, se optó por un diseño mínimamente invasivo sobre la arquitectura existente. Como RMI requiere sus propias directivas en los mecanismos de comunicación, es indispensable el establecimiento de tres entidades:

El contrato del servicio: la definición explícita de los métodos que ofrecerá el servicio. Definidos a través de una Interfaz Java con la directiva Remote.

Para cumplir el criterio de no invasión debe definirse un Interfaz nuevo que sea equivalente al contrato definido en la arquitectura.

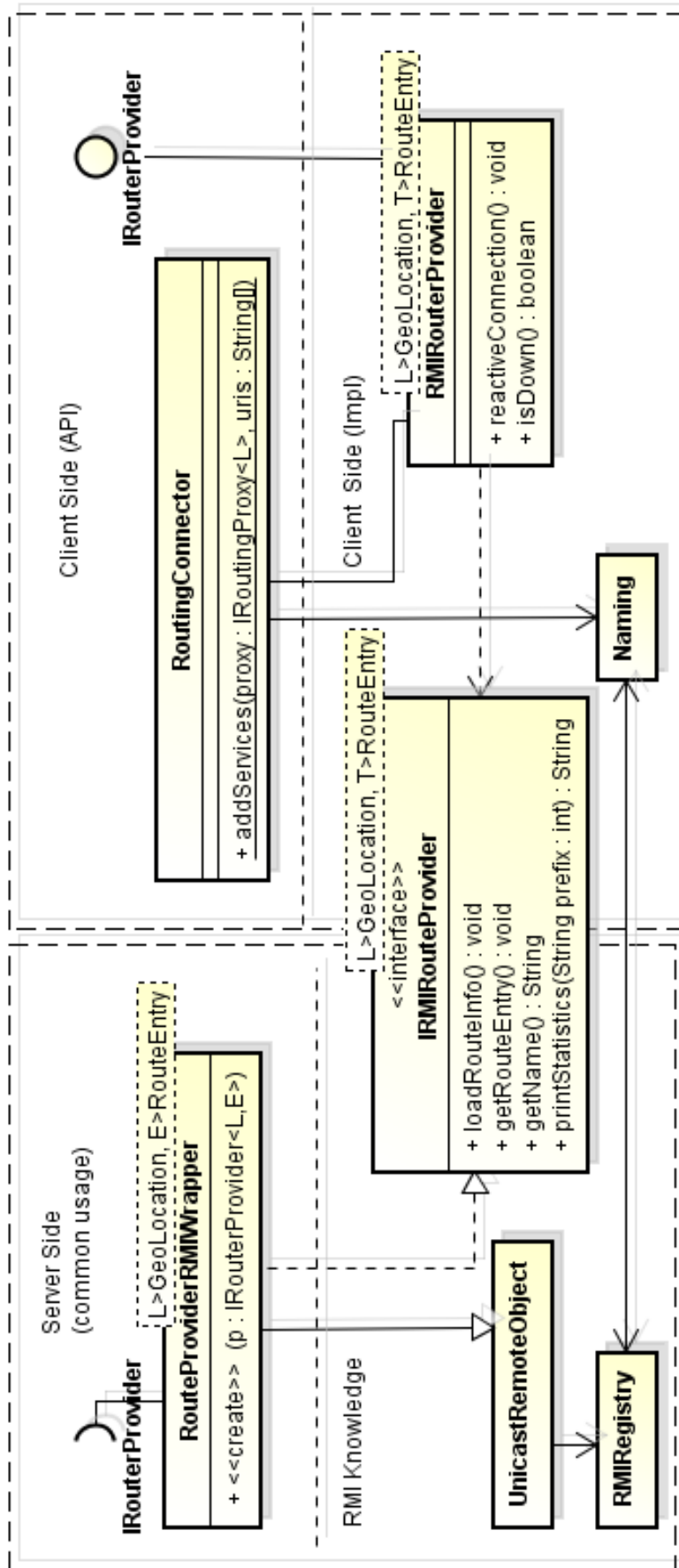
El conector del lado servidor: entidad encargada de proporcionar el servicio. Debe incluir el contrato anterior, por lo que también debe ser una entidad nueva que implemente los requisitos RMI necesarios e imbuya a un proveedor no RMI de forma transparente

El conector del lado cliente: obligatoriamente será una entidad nueva al requerir el conocimiento sobre el canal de comunicación. Debe comportarse hacia el exterior como un proveedor más sin implicar el tratamiento explícito de la comunicación RMI, lo que incluye el tratamiento de las excepciones.

La prioridad en este diseño era proporcionar una API sencilla para el lado cliente. Objetivo cumplido al requerir únicamente la uri del servicio.

Esta dependencia exclusiva de la uri permite la incorporación similar de otros esquemas de comunicación como RESTful, WSDL y cualquier otro mecanismo que pueda ser definido exclusivamente mediante una cadena de texto.

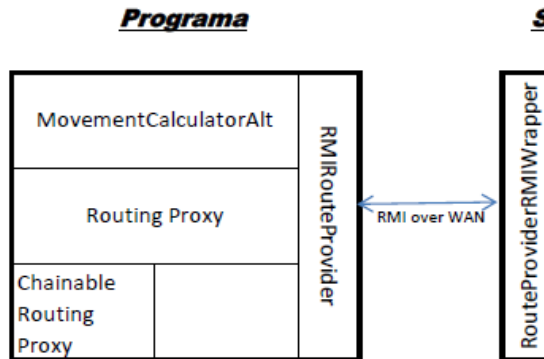
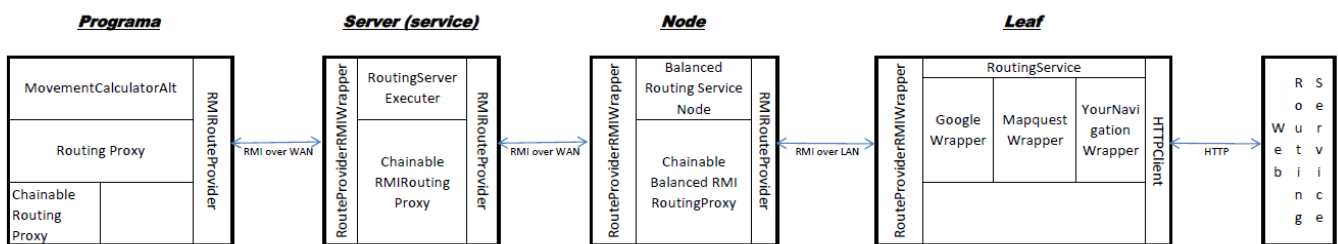
Para finalizar con premura este apartado, se ofrece el esquema de diseño de la arquitectura.



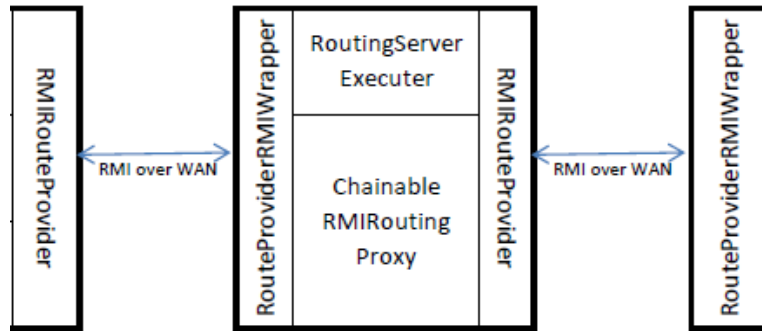
Modelo de despliegue

Para finalizar este bloque, se incluye el modelo de despliegue de la arquitectura. Tanto el conceptual, útil para una comprensión mayor de la arquitectura interna, acoplamiento y directrices de ensamblado, como el modelo de despliegue físico, que aporta una visión global de los nodos de cómputo, viendo cómo y en qué punto de la jerarquía se incluyen dichos nodos.

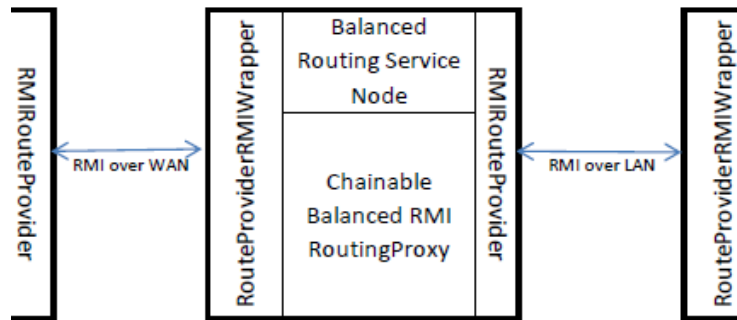
Modelo conceptual



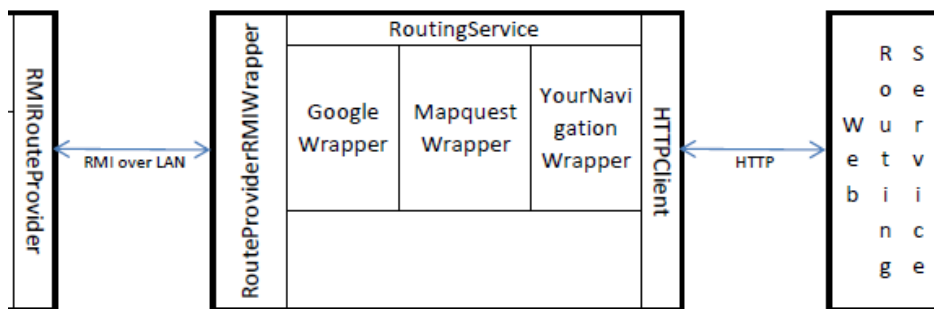
Server (service)



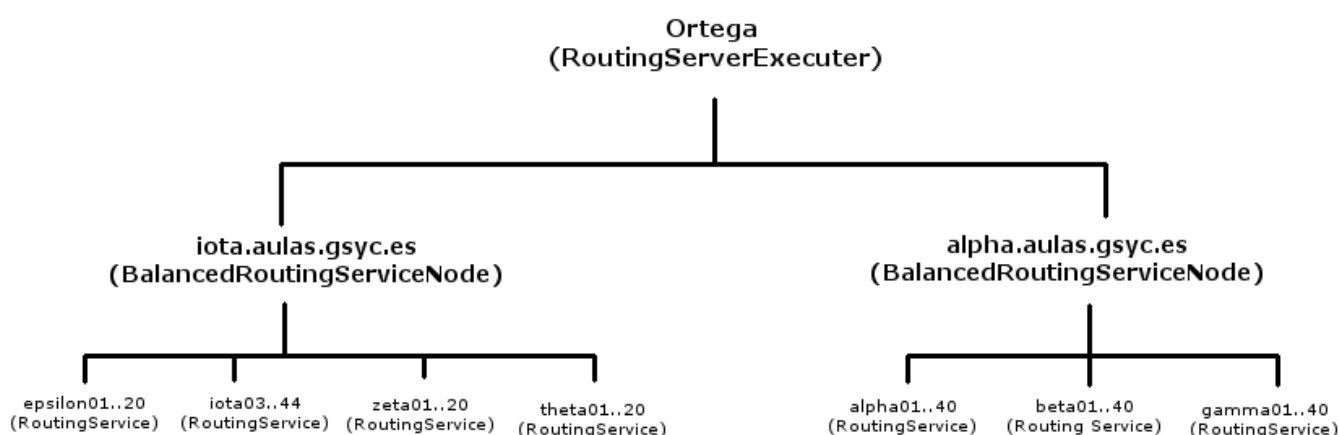
Node



Leaf



Modelo físico



En el modelo físico se puede apreciar la existencia de un joiner⁹ de servicios como front-end para la granja de proveedores. Este nodo intermedio es necesario por tres motivos.

El primer motivo es por simplicidad de uso y mantenimiento del nodo servidor, al requerir menor número de conexiones explícitas.

El segundo motivo es el de abstracción, al situarse el sistema de balanceo de carga en el nivel inferior.

El tercer motivo y más importante es por las limitaciones de la infraestructura de red.

Aunque tradicionalmente el canal de comunicación entre los Campus de Fuenlabrada y Móstoles era de alta calidad, especialmente entre los ordenadores de Linux de ambos Campus, los cambios administrativos de los últimos años han afectado notablemente a estos equipos, tanto a nivel de red como a nivel de equipos.

Actualmente, solo un diez por ciento de los ordenadores del Campus de Móstoles están disponibles para que los alumnos los utilicen y el acceso local es extremadamente más lento que el remoto.

Por ello, a día de hoy, un simple ping desde Móstoles a Fuenlabrada tarda entre uno y dos milisegundos cuando se resuelve la dirección DNS y en torno a uno cuando se emplea la dirección IP directamente.

Sabiendo que en caso de fallo habrá que pagar las latencias de cada llamada adicional, es lógico que estas llamadas se ubiquen entre nodos con mínima latencia. En este caso, dentro de la propia LAN.

Por ello, el número de niveles de la jerarquía y su ubicación física depende de la calidad de la red y del comportamiento de los nodos.

En conclusión, la arquitectura física final, está diseñada para ofrecer la mayor disponibilidad del servicio con el menor coste en latencia.

⁹ El joiner, tal y como está definido en la arquitectura, unifica el mismo servicio proporcionado en diversos huéspedes e integra los mecanismos de balanceo de carga. Se corresponde a `balancedRoutingServiceNode`.

Speedup, una medida de rendimiento

Como conclusión de la eficacia del nuevo servicio de rutas, se incluye la noción de speedup y la comparativa entre ambos.

El speedup es una medida cuantitativa de la diferencia de rendimiento entre dos soluciones. Está asociado a la Ley de Amdahl, que establece que

“la mejora obtenida en el rendimiento de un sistema debido a la alteración de uno de sus componentes está limitada por la fracción de tiempo que se utiliza dicho componente”.

Teniendo en cuenta los aspectos relativos al cómputo requerido, a la latencia de red, al coste del protocolo, al tamaño de la comunicación, al número de peticiones, el número de aciertos en caché, etc, podría definirse una fórmula general para determinar tanto el tiempo teórico empleado como el ancho de banda utilizado, acorde a la formulación original de la ley de Amdahl. Sin embargo, la fórmula resumida del speedup toma únicamente los tiempos de ejecución.

En consecuencia, todos estos factores se pueden simplificar a dos factores:

- a) Los tiempos de espera obligatorios.
- b) La sobrecarga de la jerarquía RMI.

Los tiempos de espera para el antiguo servicio son:

1. Entre 2 y 3 segundos de espera entre peticiones.
2. 24 horas de espera cada 50.000 peticiones.

La sobrecarga del protocolo RMI, incluyendo la jerarquía multinivel y ponderando las latencias cuando la red está en buenas o malas condiciones es de 920 milisegundos. Esta simplificación no aporta un valor de speedup real, pero sí orientativo.

El resto de términos no contemplados son un factor aditivo de la misma magnitud tanto en el numerador como en el denominador. Por tanto, y teniendo en cuenta las propiedades de la división, el speedup real será menor. Como apunte final, la certeza del speedup calculado depende de la diferencia de órdenes de magnitud entre los tiempos comunes y no comunes, y se puede determinar el desvío a partir del ratio de ambos factores.

$$t_{antiguo}(n) = n \times 2.5 \text{ segundos} + (n \bmod (50000 + 1)) \times 24 \text{ horas}$$

Donde n es el número de peticiones. El primer término es el tiempo de espera por cada petición y el segundo término el tiempo de espera por alcanzar el límite del servicio.

$$t_{nuevo}(n, m) = n \times \frac{1^*}{m} \times 2.5 \text{ seg} + n \times 0.92 \text{ seg} + (n \bmod (m \times 50000 + 1)) \times 24 \text{ horas}$$

Donde n es el número de peticiones y m el número de proveedores. El primer término es el tiempo de espera por cada petición, que solo deberá pagarse cuando la política de balanceo de carga elija un proveedor recientemente utilizado. El segundo término es la penalización por el protocolo de red y el tercer término el tiempo de espera por alcanzar el límite del servicio.

En la siguiente tabla se puede ver el speedup obtenido para cada uno de los escenarios presentados.

Parámetros	Tiempo requerido					Speedup [min,max]
	Sistema antiguo	Sistema nuevo (m=1)	Sistema nuevo (m=10)	Sistema nuevo (m=50)	Sistema nuevo (m=100)	
10	25''	34''	11.7''	9.7''	9.45''	0.73-2.64
500	20'	28'	9'	8'	7'	0.73-2.64
5.000	3.4h	4.75h	1.6h	1.34h	1.31h	0.73-2.64
100.000	93h	119h	32h	27h	26h	0.78-3.56
200.000	210h	262h	65h	54h	52h	0.8-4

Para completar esta tabla, se asume la no existencia de caché.

Todas las variantes del nuevo sistema tienen los mismos niveles de jerarquía, incluido el de un solo proveedor. Dicho sistema contará con: el cliente, el servicio en ortega, el nodo de balanceo en iota y el nodo proveedor en esa misma máquina.

El valor de speedup está resumido a dos valores, el mínimo y el máximo. EL mínimo siempre corresponde a la configuración con m=1 y el máximo a la configuración con m=100.

La tabla anterior muestra unos tiempos bastante elevados al no contar con la caché. Sin embargo, el uso de una caché persistente así como de un servicio de rutas cuya vida va más allá de cada experimentación, aumenta la tasa de aciertos de manera sustancial. En la siguiente tabla se puede ver la tasa de aciertos a lo largo del tiempo. Esta ha sido medida en diferentes puntos de la jerarquía.

Nivel de la jerarquía medido	Número total de accesos	Número de aciertos	Tasa de aciertos
Proxy en el cliente	5446	1473	28%
Servicio de Ortega, estadísticas de dos simulaciones	9021	2443	27%
Servicio de Ortega, tras lanzar y cancelar 10 experimentos al 10%	5745	3626	63%
Servicio de Ortega, estadísticas de un mes	220.454	213.896	97%

Hay que recordar que las estadísticas disponibles solo tienen en cuenta el nivel de caché interno, no los anidados. De modo que, para la segunda entrada de la tabla, el número de peticiones totales será de doce mil aproximadamente, correspondientes a los aciertos en caché del cliente.

Si unificamos las conclusiones extraídas de las tablas, podemos estimar el speedup entre la solución anterior previo a cualquier modificación anterior, que solo utilizaba Google Maps y la solución propuesta en este proyecto. Para este ejemplo maximizaremos las diferencias de forma artificial y supondremos que la caché está llena, y por tanto su tasa de acierto será del 97%, y que el servicio está desplegado sobre cien nodos.

$$S(n = 100.000) = \frac{26h \times (1 - 0.97) + 0.97 \times (n \times 920 \text{ milis})}{93h} = 46$$

Para un entorno más estándar, el speedup se reduce a unos valores dentro de lo normal. En este caso, un servicio con una caché con un 6% de tasa de acierto, diez proveedores y pagando el coste del protocolo RMI de un solo nivel.

$$S(n = 5.000) = \frac{1.6h \times (1 - 0.06) + 0.06 \times (n \times 230 \text{ milis})}{3.4h} = 2.23$$

Con estas comparativas damos por cerrado el capítulo.

Conclusiones

En este apartado se reflejan las conclusiones del proyecto. Éstas serán presentadas en orden inverso, primero el servicio de rutas y después el coordinador de descansos.

Servicio de Rutas

El servicio de rutas diseñado cumple con todos los objetivos propuestos y muchos más:

- ✓ Es un servicio único que no requiere ser detenido por ninguna causa: permite la introducción de nuevos proveedores, la eliminación de los actuales, el vaciado de la caché y el guardado y carga de la misma. Su única limitación es estar publicado como servicio Java a través de RMI.
- ✓ La jerarquía multinivel y los comportamientos de sus nodos permiten un balanceo de carga que nos ha permitido olvidarnos de los límites impuestos por los proveedores de rutas, los errores por caídas de red y zonas muertas.
- ✓ La validez de la caché está demostrada con una tasa de acierto del 97%¹⁰
- ✓ La extensibilidad está probada al haber añadido un proveedor más basado en web y otro que no cumple este canon.
- ✓ Ante la caída de un servicio remoto, éste se desactiva de modo que no afecta a la latencia del servicio una vez detectado.
- ✓ Proporciona una terminal de gestión y estadísticas de su funcionamiento.
- ✓ El servicio está disponible las 24h del día en el Servidor Ortega.
- ✓ El mantenimiento del servicio (levantarlo si se cae, adición de nodos y backups) está completamente automatizado mediante scripts y entradas cron [G10, C12].
- ✓ Se ha desplegado con éxito en el 90% de los ordenadores de Fuenlabrada (más de cien) y en el 20% de los ordenadores de Móstoles (menos de cuarenta de los ciento sesenta, al estar las aulas sin mantenimiento), aglutinando en el mejor de los escenarios un total de 200 nodos de cómputo.

¹⁰ Tasa de acierto obtenida para una configuración con 100.000 entradas de caché y 250.000 peticiones de rutas.

- ✓ Ha supuesto una mejora de hasta tres veces respecto al servicio anterior atendiendo únicamente al coste temporal y de hasta veinticinco veces mejor si se tiene en cuenta el sistema de caché

Poco más se puede decir de este servicio. Tiene las limitaciones de RMI y salvo por ello, el servicio se puede considerar terminado en este contexto. Por tanto, se puede estar satisfecho con la realización y funcionamiento del mismo.

Coordinador de descansos

La principal contribución de este proyecto es el desarrollo del sistema coordinador de descansos. Su definición ha permitido comprender mejor el dominio de los descansos y de los turnos, así como el grado de dependencia con estos últimos.

Si el escenario a modelar utiliza turnos para la rotación de los recursos humanos, es imposible crear un sistema de coordinación de descansos que no dependa de los turnos. Como mínimo es necesario saber cuándo acaba el turno para no planificar más allá de este. Este ejemplo es especialmente importante tanto cuando el coordinador puede realizar asignaciones en cualquier momento como cuando los turnos pueden acabar en cualquier minuto o hay turnos solapados y, aunque se restrinja el problema para que se produzca una sincronización implícita entre ambos, los turnos seguirán siendo necesarios para la realización de predicciones.

Otro de los puntos determinantes es la necesidad de un sistema dinámico. La función de autoajuste ha dado muy buenos resultados, especialmente en experimentos de más de tres días.

Cuando los parámetros de configuración habían sido mal definidos o había mezclas de turnos (de doce y veinticuatro horas), el dinamismo del valor de reasignación aportaba un valor añadido.

Con respecto a los algoritmos de asignación, se ha visto que ante escenarios bien planeados, es decir, una relación oferta-demanda (ambulancias-pacientes) equilibrada, un modelo que no permita interrupciones no supone una gran pérdida en los tiempos de atención cuando se emplea el módulo de posicionamiento basado en voronoi. Esto quiere decir que, antes de planear una mejora de este coordinador, quizá sea más adecuado concentrar estos esfuerzos sobre otros ámbitos del problema más relevantes, en los que una mejora menor aportará más beneficio.

Por último, la implementación realizada debe tomarse en su conjunto como un prototipo. Aunque ciertas áreas hayan sido bien diseñadas y puedan ser reutilizadas tal cual son, este no debería implantarse en un entorno real en su forma actual.

Líneas futuras

Aunque el servicio de rutas se considera terminado, su utilización puede trascender más allá del ámbito de este proyecto. Así pues, la reevaluación del modelo de datos y de las interfaces sería el primer objetivo hacia una implementación más genérica.

En términos de funcionalidad, dos capítulos restantes podrían añadirse. El primero es la sustitución de la capa de comunicación RMI por conectores basados en servicios web u otro tipo de protocolos que mejorasen la eficiencia y garantizaran el canal de comunicación. El segundo es la implementación de un proveedor off-line, para no tener que requerir explotar tanto la arquitectura.

Con respecto al coordinador de descanso, las líneas futuras son demasiado numerosas y divergentes como para mentarlas todas. Se pueden plantear todas las alternativas casi como si se realizase *ex novo*. Sin embargo, solo se mencionarán las que tengan relevancia con el proyecto actual.

- Queda, como prueba de concepto, el diseño del coordinador como un problema de satisfacción de restricciones. Probando la funcionalidad, tiempo de decisión y flexibilidad del mismo.
- Algunos de los algoritmos descritos no han podido ser probados, la evaluación de estos algoritmos sería un paso a realizar para aumentar el contexto sobre este ámbito.
- La aplicación no cuenta con ninguna interfaz gráfica de gestión. La realización de la misma para el manejo de la configuración y de los turnos aportarían un gran valor en términos de usuario final.
- La inclusión de mecanismos de autoevaluación de los resultados agilizaría al estudio de los modelos diseñados.

Bibliografía

Contexto

- [BCD+10] Holger Billhardt, Roberto Centeno, José J. Durán, Moser Fagundes, Diego Fradejas, Marin Lujak and Rubén Torres. "D8.3.1 mHealth Requirements Analysis and Design." Agreement Technologies, Deliverable D8.3.1, month 48 (WP8.3) CSD2007-0022, INGENIO 2010
- [LBCH10] Marin Lujak, Holger Billhardt, Helena Cebrian, Alvaro Martinez Higes. "D8.3.3 mHealth Medical Emergency Transportation Prototype Review." Deliverable D8.3.3, month 54 (WP8.3) CSD2007-0022, INGENIO 2010
- [RS10] Eva Rodríguez Yagüe, Miguel Isidoro Sánchez Gómez. "mHealth: Demostrador de emergencias médicas." Universidad Rey Juan Carlos, 2010

Teórico

- [BQB10] Brucker, Peter, Rong Qu, and Edmund Burke. "Personnel scheduling: Models and complexity." *European Journal of Operational Research* 210.3 (2011): 467-473.
- [Mus06] Musliu, Nysret. "Heuristic methods for automatic rotating workforce scheduling." *International Journal of Computational Intelligence Research* 2.4 (2006): 309-326.
- [Aur91] Aurenhammer, Franz. "Voronoi diagrams—a survey of a fundamental geometric data structure." *ACM Computing Surveys (CSUR)* 23.3 (1991): 345-405.
- [JT11] Jensen, Tommy R., and Bjarne Toft. *Graph coloring problems*. Wiley-Interscience, 2011.
- [LBS07] López, Montserrat Abril, F. Barber Sanchís, and MA Salido Gregorio. "Particionamiento y resolución distribuida multivariable de problemas de satisfacción de restricciones." (2007).

Práctico

- [JCS] JaCoP, Java constraint solver: <http://www.osolpro.com/jacop/index.php>
- [xSt] XStream, simple library to serialize objects to XML and back again: <http://xstream.codehaus.org/>

- [GM] Google Maps directions API:
<https://developers.google.com/maps/documentation/directions>
- [MQ] MapQuest directions API: www.mapquestapi.com/directions
www.mapquest.com/routeplanner
- [YN] YourNavigation: <http://wiki.openstreetmap.org/wiki/YOURS>
- [TS] OSM Java Routing implementation:
http://wiki.openstreetmap.org/wiki/Traveling_salesman
- [ExJ] Java Excel API: <http://jexcelapi.sourceforge.net/>
- [G10] Garrels, Machtelt. Bash Guide for Beginners. Fultus Books, 2010.
- [C12] Cooper, Mendel. "Advanced Bash-Scripting Guide." [Online document], Available HTTP: <http://www.tldp.org/LDP/abs/html/index.html> (2012).
- [CTSN02] Chiueh, Tzi-cker, Harish Sankaran, and Anindya Neogi. "Spout: a transparent proxy for safe execution of Java applets." Selected Areas in Communications, IEEE Journal on 20.7 (2002): 1426-1433.

Apéndice A

1. Dividir el mapa en (n_divisions) regiones
2. **PARA** cada franja horaria (desayuno, comida, cena ...) **HACER**
 3. Hacer una lista con las ambulancias cuya distancia al borde (frontera) sea menor que su radio de influencia, guardando también a la región que pertenecen.
 4. **PARA** cada región **HACER**
 5. Se ordenan las ambulancias de esta región de menor a mayor satisfacción
 6. La variable número de ambulancias comiendo a la vez se pone a 0
 7. **PARA** cada ambulancia de esa región (en orden) **HACER**
 8. **SI** está en la lista anterior (su radio de influencia traspasa la frontera entre 2 regiones):
 - a. Buscamos en las regiones contiguas, las ambulancias que están en la lista Y están dentro del radio de esta ambulancia.
 - b. **SI** hay al menos una ambulancia que cumple ambas cosas:
 - i. **PARA** cada hora en la lista de preferencias de esta ambulancia (por orden de preferencia) **HACER**
 1. **SI** no se ha llegado al máximo de ambulancias comiendo en la región a esa hora Y alguna de las ambulancias dentro de su radio de influencia no tiene asignada esa hora para comer (es decir, puede haber 3 ambulancias dentro de un radio de influencia en una frontera, y esto asegura que al menos una estará siempre disponible) , se le adjudica ese horario, se actualiza la satisfacción de esa ambulancia y se aumenta la variable de las ambulancias que están comiendo a la vez en esa hora (Pasamos a la siguiente ambulancia en el paso 7)
 2. **EN OTRO CASO** pasamos a la siguiente hora en las preferencias* (volvemos al paso i.)
 - c. **SI NO** hay ninguna ambulancia que cumple ambas cosas:
 - i. Saltar al paso 10
9. **SI NO** está en la lista anterior:
10. **PARA** cada hora en la lista de preferencias de esta ambulancia (por orden de preferencia) **HACER**
 11. **SI** se ha llegado al máximo de ambulancias comiendo en la región a esa hora, pasamos a la siguiente hora en las preferencias** (volvemos al paso 10)
 12. **SI NO** se ha llegado al máximo de ambulancias comiendo en la región a esa hora, se le adjudica ese horario, se actualiza la satisfacción de esa ambulancia y se aumenta la variable de las ambulancias que están

comiendo a la vez en esa hora (Pasamos a la siguiente ambulancia en el paso 7)

** Se debe asegurar que no hay más ambulancias solapadas (1 por región) que horas para comer en esa franja horaria.*

***Suponemos que (nº de horas totales para cada comida) * (nº máximo de ambulancias comiendo a la vez por cada hora) >= (nº de ambulancias en una región).*

Apéndice B

```
"[ES] Máximo número de ambulancias comiendo a la vez"
"[EN] Maximun number of ambulance while eating"
public static int LBA__MAX_GLOBAL_EATING = 12;

"[ES] Mínimo número de ambulancias que deben permanecer activas"
+NewLine+
"[EN] Minimal number of ambulance that must keep active"
public static int LBA__MIN_GLOBAL_REMAIN = 20;

"[ES] Si está activo, se olvida la limitación de la Triangulación de
Delaunay y se asignan ambulancias aunque haya una vecina comiendo"
"[EN] When active, Delaunay Triangulation limitation is skipped and
could be an assignment even if one neighbor is eating"
public static boolean LBA__SKIP_DT_LIMITATION = true;

"[ES] Ajuste de reasignación. Su valor debe ser aproximadamente el
número de interrupciones esperadas por Break."
"[EN] Adjustment reassignment. His value must be at least the number
of expected interruptions"
public static int LBA__ADJUST_UNASSIGNED_REASSIGNATION = 2;

"[ES] Si se activa, el ajuste de reasignación será dinámico."
"[EN] When active, unassigned Ajustement reassignment change
dynamically"
public static boolean LBM__DINAMIC_ADJUST = true;

"[ES] Para realizar dinámicamente el ajuste de reasignación. Su valor
debe ser aproximadamente la capacidad de llenado de la Triangulación
de Delaunay (por ejemplo: 14-30%)"
"[EN] For algorithm to change dynamically the Adjustment Reassignment.
Its value should be approximately the filling capacity of the Delaunay
triangulation (eg 20-30%)"
public static float LBM__DINAMIC_UNASSIGNED_ADJUST = 24/100f;

"[ES] Tiempo máximo en minutos que puede descansar seguidamente una
ambulancia"
"[EN] Maximum time in minutes that can rest then an ambulance"
public static int LBA__BREAK_MAX_MINUTES_LIMIT = 60;

"[ES] Tiempo mínimo asignable para descansar. Si a una ambulancia le
quedan 5min y este límite es 20, se asignarán 20 minutos (ganancia de
15min)"
"[EN] Minimum assignable to rest. If an ambulance has remaining 5
minutes and this limit is 20, will be allocated 20 minutes (15 min
gain)"
public static int LBA__BREAK_MIN_MINUTES_LIMIT = 20;
```

```

/////SATISFACTION

//Seat
"[ES] Peso asociado a la satisfacción por preferencias/descanso"
"[EN] Weight associated with Preferences/Breaks satisfaction function"
public static float Satisfaction__Combined_Weat = 0.85f;

"[ES] Valor inicial del componente Seat"
"[EN] Initial value of the component Seat"
public static float Satisfaction__Eat_InitialValue = 0.7f;

"[ES] Contribución del valor histórico en la actualización"
"[EN] Contribution from the historical value in the update"
public static float Satisfaction__Eat_Whistoric = 0.5f;

//Sint
"[ES] Peso asociado a la satisfacción por interrupciones"
"[EN] Weight associated with Interruptions satisfaction function"
public static float Satisfaction__Combined_Wint = 0.1f;

"[ES] Valor inicial del componente Sint"
"[EN] Initial value of the component Sint"
public static float Satisfaction__Int_InitialValue = 1f;

"[ES] Contribución del valor histórico en la actualización"
"[EN] Contribution from the historical value in the update"
public static float Satisfaction__Int_Whistoric = 0.25f;

"[ES] Base exponencial. Donde Base^N --> 0, siendo N el número de
interrupciones esperadas/máximas"
"[EN] Exponential basis. Where Base^N --> 0, where N is the number of
expected/maximum breaks"
public static float Satisfaction__Int_Base = 0.79f;

//Swork
"[ES] Peso asociado a la satisfacción por trabajo"
"[EN] Weight associated with Work satisfaction function"
public static float Satisfaction__Combined_Wwork = 0.05f;

"[ES] Valor inicial del componente Swork"
"[EN] Initial value of the component Swork"
public static float Satisfaction__Work_InitialValue = 1f;

"[ES] Contribución del valor histórico en la actualización"
"[EN] Contribution from the historical value in the update"
public static float Satisfaction__Work_Whistoric = 0f;

```

```
/////Satisfaction threshold for retry

"[ES] Si la satisfacción total de la ambulancia es menor que este
límite, intentará asignarla"
"[EN] If the total satisfaction of the ambulance is less than this
limit, try to assign"
public static float LBM__SATISFACTION_THRESHOLD__GLOBAL_TO_TRY = 0.6f;

"[ES] Mínimo valor de la función de satisfacción 'Seat' para que se
realize la asignación"
"[EN] Minimum value of the function of satisfaction 'Seat' for the
allocation is performed"
public static float LBM__SATISFACTION_THRESHOLD__EAT_TO_ASSING = 0.8f;

"[ES] Minutos asignados en caso de reasignación individual"
"[EN] Minutes assigned if an individual reallocation"
public static int LBM__SATISFACTION_THRESHOLD__MINUTES_TO_ASSIGN = 23;
```

Apéndice C

```
<object-stream>
  <shiftmanagement.turns.TurnConfigLoader>
    <beginOfSimulation>1970-01-01 00:00:00.00 CET</beginOfSimulation>
    <turns>

      <shiftmanagement.turns.TurnConfig>
        <type>T24H</type>
        <beginDate>1970-01-01 00:00:00.00 CET</beginDate>

        <workers_uids>
          <string>amb_1</string>
            <string>amb_2</string>
            <string>amb_3</string>
            <string>amb_4</string>
            <string>amb_5</string>
            <string>amb_6</string>
            <string>amb_7</string>
            <string>amb_8</string>
            <string>amb_9</string>
            <string>amb_10</string>
          </workers_uids>
        </shiftmanagement.turns.TurnConfig>

      <shiftmanagement.turns.TurnConfig>
        <type>T12H</type>
        <beginDate>1970-01-01 06:00:00.00 CET</beginDate>

        <workers_uids>
          <string>amb_11</string>
          <string>amb_12</string>
          <string>amb_13</string>
          <string>amb_14</string>
          <string>amb_15</string>
        </workers_uids>
      </shiftmanagement.turns.TurnConfig>

      <shiftmanagement.turns.TurnConfig>
        <type>T24H</type>
        <beginDate>1970-01-02 00:00:00.00 CET</beginDate>

        <workers_uids>
          <string>amb_20</string>
          <string>amb_21</string>
          <string>amb_22</string>
          <string>amb_23</string>
          <string>amb_24</string>
          <string>amb_25</string>
          <string>amb_26</string>
        </workers_uids>
      </shiftmanagement.turns.TurnConfig>
    </turns>
  </shiftmanagement.turns.TurnConfigLoader>
</object-stream>
```

```
        <string>amb_27</string>
        <string>amb_28</string>
        <string>amb_29</string>
    </workers_uids>
</shiftmanagement.turns.TurnConfig>

</turns>
</shiftmanagement.turns.TurnConfigLoader>
</object-stream>
```

Anexo 1

Con este anexo se adjunta uno de los ficheros generados por la ejecución de una experimentación. Corresponde al coordinador de descansos y proporciona el conocimiento de decisiones tomadas así como medidas estadísticas.

La versión ofrecida en este documento se ha visto reducida a la información más relevante para conseguir una legibilidad asequible. Se trata de la simulación del día crítico, en el que se producen más de doscientas llamadas en 24h. Para modelarlo, se ha escogido un solo turno de 24h, el cual está dividido en tres periodos de ocho horas a los que les corresponde un descanso. Para las preferencias se ha asumido una distribución gaussiana determinando que la hora central es la primera elección.

En la siguiente tabla se especifican los parámetros del coordinador de descansos.

VARIABLE	VALUE
LBA_BREAK_MAX_MINUTES_LIMIT	60
LBA_BREAK_MIN_MINUTES_LIMIT	20
LBA_MAX_GLOBAL_EATING	10
LBA_MIN_GLOBAL_REMAIN	10
LBA_ADJUST_UNASSIGNED_REASSIGNATION	4
LBA_SKIP_DT_LIMITATION	true
Satisfaction_Combined_Weat	0.7
Satisfaction_Combined_Wint	0.2
Satisfaction_Combined_Wwork	0.05
Satisfaction_Eat_InitialValue	0.7
Satisfaction_Eat_Whistoric	0.5
Satisfaction_Int_InitialValue	1.0
Satisfaction_Int_Whistoric	0.25
Satisfaction_Work_InitialValue	1.0
Satisfaction_Work_Whistoric	0.0
LBM_DINAMIC_ADJUST	true
LBM_DINAMIC_UNASSIGNED_ADJUST	0.3
LBM_SATISFACTION_THRESHOLD_EAT_TO_ASSING	0.7
LBM_SATISFACTION_THRESHOLD_EAT_TO_ASSING	0.7
LBM_SATISFACTION_THRESHOLD_GLOBAL_TO_TRY	0.7
LBM_SATISFACTION_THRESHOLD_MINUTES_TO_ASSIGN	20

Turn T8H (lunchBreaks.realtiming.Turn@2d7cec96), started at Sun Jan 01 00:00:00 CET 2012

Amb ID	Satisfaction before	Satisfaction after	Break 0	minutes eaten	patients assisted	SatisfactionEatSum	SatisfactionIntExp	SatisfactionWorkDep
amb_1	0.74	0.80125	(03:00)	(60)	0	0.7875	1.0	1.0
amb_10	0.74	0.8422973	(04:00)	(60)	2	0.85	1.0	0.9459459
amb_11	0.74	0.7998986	(03:00)	(60)	1	0.7875	1.0	0.972973
amb_12	0.74	0.79854727	(03:00)	(60)	2	0.7875	1.0	0.9459459
amb_13	0.74	0.7779848	(02:00)(04:00)	(15)(45)	2	0.803125	0.84250003	0.9459459
amb_14	0.74	0.80125	(03:00)	(60)	0	0.7875	1.0	1.0
amb_15	0.74	0.80125	(03:00)	(60)	0	0.7875	1.0	1.0
amb_16	0.74	0.6248987	(01:00)	(60)	1	0.5375	1.0	0.972973
amb_17	0.74	0.7575	(05:00)	(60)	0	0.725	1.0	1.0
amb_18	0.74	0.6248987	(01:00)	(60)	1	0.5375	1.0	0.972973
amb_19	0.74	0.71239865	(02:00)	(60)	1	0.6625	1.0	0.972973
amb_2	0.74	0.7575	(05:00)	(60)	0	0.725	1.0	1.0
amb_20	0.74	0.6248987	(01:00)	(60)	1	0.5375	1.0	0.972973
amb_21	0.74	0.5333446	(00:00)	(60)	4	0.4125	1.0	0.8918919
amb_22	0.74	0.71239865	(02:00)	(60)	1	0.6625	1.0	0.972973
amb_23	0.74	0.6286263	(05:00)(05:56)(07:00)	(25)(12)(23)	4	0.62916666	0.71807504	0.8918919
amb_24	0.74	0.6029848	(00:00)(05:00)	(33)(27)	2	0.55312496	0.84250003	0.9459459
amb_25	0.74	0.7971959	(03:00)	(60)	3	0.7875	1.0	0.9189189
amb_26	0.74	0.53739864	(00:00)	(60)	1	0.4125	1.0	0.972973
amb_27	0.74	0.7282945	(01:00)(04:00)	(23)(37)	1	0.73020834	0.84250003	0.972973
amb_28	0.74	0.71375	(02:00)	(60)	0	0.6625	1.0	1.0
amb_29	0.74	0.5360473	(00:00)	(60)	2	0.4125	1.0	0.9459459
amb_3	0.74	0.7547973	(05:00)	(60)	2	0.725	1.0	0.9459459
amb_4	0.74	0.7998986	(03:00)	(60)	1	0.7875	1.0	0.972973
amb_5	0.74	0.7998986	(03:00)	(60)	1	0.7875	1.0	0.972973
amb_6	0.74	0.8033986	(02:00)(04:00)	(4)(56)	1	0.8375	0.84250003	0.972973
amb_7	0.74	0.845	(04:00)	(60)	0	0.85	1.0	1.0
amb_8	0.74	0.845	(04:00)	(60)	0	0.85	1.0	1.0
amb_9	0.74	0.70225847	(03:00)(06:00)	(31)(29)	3	0.696875	0.84250003	0.9189189

Turn Stats

Minutes eaten	% minutes	n° not eaten	% not eaten	n° assisted patients	SatisfactionWorkDep	SatisfactionEatSum	SatisfactionIntExp
1740	100.0	0	0.0	37	0.9655172	0.6935345	0.96312326

Turn T8H (lunchBreaks.realtiming.Turn@7f724a9d), started at Sun Jan 01 08:00:00 CET 2012

Amb ID	Satisfaction before	Satisfaction after	Break 0	minutes eaten	patients assisted	SatisfactionEatSum	SatisfactionIntExp	SatisfactionWorkDep
amb_1	0.80125	0.7088415	(08:00)(10:00)(13:00)	(5)(2)(53)	7	0.740625	0.71807504	0.9357798
amb_10	0.845	0.64094687	(09:00)(11:00)(15:00)	(8)(16)(36)	4	0.6416667	0.71807504	0.96330273
amb_11	0.80125	0.65504014	(09:00)	(60)	4	0.58125	1.0	0.96330273
amb_12	0.80125	0.7425401	(10:00)	(60)	4	0.70625	1.0	0.96330273
amb_13	0.7806875	0.5174989	(14:00)(15:00)	(12)(12)	4	0.4765625	0.67870003	0.96330273
amb_14	0.80125	0.744375	(10:00)	(60)	0	0.70625	1.0	1.0
amb_15	0.80125	0.74299884	(10:00)	(60)	3	0.70625	1.0	0.9724771
amb_16	0.62625	0.72999847	(11:00)(15:00)	(58)(20)	4	0.73333335	0.84250003	0.96330273
amb_17	0.7575	0.63362384	(09:00)	(60)	3	0.55	1.0	0.9724771
amb_18	0.62625	0.74345756	(11:00)	(60)	2	0.70625	1.0	0.98165137
amb_19	0.71375	0.6782971	(11:00)(12:00)(12:34)(13:00)(13:21)	(10)(15)(0)(0)(35)	8	0.7479167	0.54212564	0.9266055
amb_2	0.7575	0.54223883	(14:00)(15:00)	(20)(26)	3	0.5	0.71807504	0.9724771
amb_20	0.62625	0.74345756	(11:00)	(60)	2	0.70625	1.0	0.98165137
amb_21	0.53875	0.700625	(11:00)	(60)	0	0.64375	1.0	1.0
amb_22	0.71375	0.74703974	(10:00)(11:00)	(5)(55)	5	0.7583333	0.84250003	0.95412844
amb_23	0.63403165	0.45244235	!!!		8	0.31458333	0.92951876	0.9266055
amb_24	0.6056875	0.6036339	(13:00)	(46)	4	0.5640625	0.803125	0.96330273
amb_25	0.80125	0.7331856	(10:00)(13:21)	(29)(31)	5	0.73854166	0.84250003	0.95412844
amb_26	0.53875	0.6044885	(11:00)(13:00)(14:00)	(30)(13)(20)	4	0.58958334	0.71807504	0.96330273
amb_27	0.72964585	0.6713218	(14:00)	(60)	3	0.6151042	0.960625	0.9724771
amb_28	0.71375	0.8304989	(12:00)	(60)	3	0.83125	1.0	0.9724771
amb_29	0.53875	0.65549886	(13:00)	(60)	3	0.58125	1.0	0.9724771
amb_3	0.7575	0.63303655	(09:00)(12:00)(12:27)	(10)(10)(22)	7	0.6604167	0.6197793	0.9357798
amb_4	0.80125	0.8314163	(11:00)	(60)	1	0.83125	1.0	0.9908257
amb_5	0.80125	0.82916516	(10:00)(12:00)	(6)(54)	4	0.875	0.84250003	0.96330273
amb_6	0.80474997	0.6885818	(08:00)(12:00)	(32)(28)	4	0.6854167	0.803125	0.96330273
amb_7	0.845	0.58987385	(08:00)	(60)	3	0.4875	1.0	0.9724771
amb_8	0.845	0.5907913	(08:00)	(60)	1	0.4875	1.0	0.9908257
amb_9	0.7063125	0.48327896	!!!		6	0.3484375	0.960625	0.94495416

Turn Stats

Minutes eaten	% minutes	n° not eaten	% not eaten	n° assisted patients	SatisfactionWorkDep	SatisfactionEatSum	SatisfactionIntExp
1559	89.5977	2	6.8965516	109	0.9655172	0.63843393	0.88068706

Turn T8H (lunchBreaks.realtiming.Turn@688a548b), started at Sun Jan 01 16:00:00 CET 2012

Amb ID	Satisfaction before	Satisfaction after	Break 0	minutes eaten	patients assisted	SatisfactionEatSum	SatisfactionIntExp	SatisfactionWorkDep
amb_1	0.7120525	0.62442446	(17:00)	(60)	3	0.5578125	0.92951876	0.96103895
amb_10	0.6427817	0.67858773	(18:00)	(60)	1	0.6333333	0.92951876	0.987013
amb_11	0.656875	0.69788176	(20:00)(21:01)	(11)(49)	4	0.68854165	0.84250003	0.9480519
amb_12	0.744375	0.6767224	(21:00)(21:40)(23:00)	(23)(25)(20)	5	0.6947917	0.71807504	0.9350649
amb_13	0.5193338	0.77197134	(18:00)(20:43)	(15)(60)	3	0.81640625	0.762175	0.96103895
amb_14	0.744375	0.6284375	(17:00)	(60)	0	0.540625	1.0	1.0
amb_15	0.744375	0.8014894	(19:00)	(60)	3	0.790625	1.0	0.96103895
amb_16	0.73183334	0.742493	(20:00)	(42)	2	0.7166667	0.960625	0.97402596
amb_17	0.635	0.7474513	(19:00)	(60)	2	0.7125	1.0	0.97402596
amb_18	0.744375	0.5389895	(16:00)	(60)	3	0.415625	1.0	0.96103895
amb_19	0.68196684	0.6181791	(17:00)	(60)	3	0.56145835	0.8855314	0.96103895
amb_2	0.543615	0.7659284	(19:00)(20:24)(21:13)	(58)(5)(20)	2	0.8395833	0.6475938	0.97402596
amb_20	0.744375	0.6271388	(17:00)	(60)	2	0.540625	1.0	0.97402596
amb_21	0.700625	0.6940625	(18:00)	(60)	0	0.634375	1.0	1.0
amb_22	0.7493333	0.5480449	(16:00)	(60)	5	0.44166666	0.960625	0.9350649
amb_23	0.4561121	0.48079467	(19:00)(20:00)(20:44)(21:15)(21:44)	(1)(7)(14)(2)(22)	7	0.4895833	0.46315897	0.9090909
amb_24	0.60546875	0.77803296	(19:00)(19:46)	(13)(60)	1	0.8143229	0.7932813	0.987013
amb_25	0.7354792	0.5411178	(16:00)	(60)	5	0.43177083	0.960625	0.9350649
amb_26	0.60632336	0.7318967	(19:00)	(57)	2	0.7104167	0.92951876	0.97402596
amb_27	0.6726979	0.6947168	(18:00)(20:29)	(41)(20)	5	0.6877605	0.83265626	0.9350649
amb_28	0.831875	0.5833888	(16:00)	(60)	2	0.478125	1.0	0.97402596
amb_29	0.656875	0.6708888	(18:00)	(60)	2	0.603125	1.0	0.97402596
amb_3	0.6362475	0.74004096	(19:00)(19:53)	(1)(60)	3	0.77500004	0.74744487	0.96103895
amb_4	0.831875	0.5846875	(16:00)	(60)	0	0.478125	1.0	1.0
amb_5	0.83100003	0.76517695	(18:00)	(60)	3	0.75	0.960625	0.96103895
amb_6	0.6904167	0.6100034	(17:00)	(60)	2	0.53020835	0.9507812	0.97402596
amb_7	0.59125	0.7262256	(19:00)	(60)	1	0.68125	1.0	0.987013
amb_8	0.59125	0.7262256	(19:00)	(60)	1	0.68125	1.0	0.987013
amb_9	0.48603123	0.49857092	(22:00)	(56)	5	0.40755206	0.83265626	0.9350649

Turn Stats

Minutes eaten	% minutes	n° not eaten	% not eaten	n° assisted patients
1762	101.264366	0	0.0	77

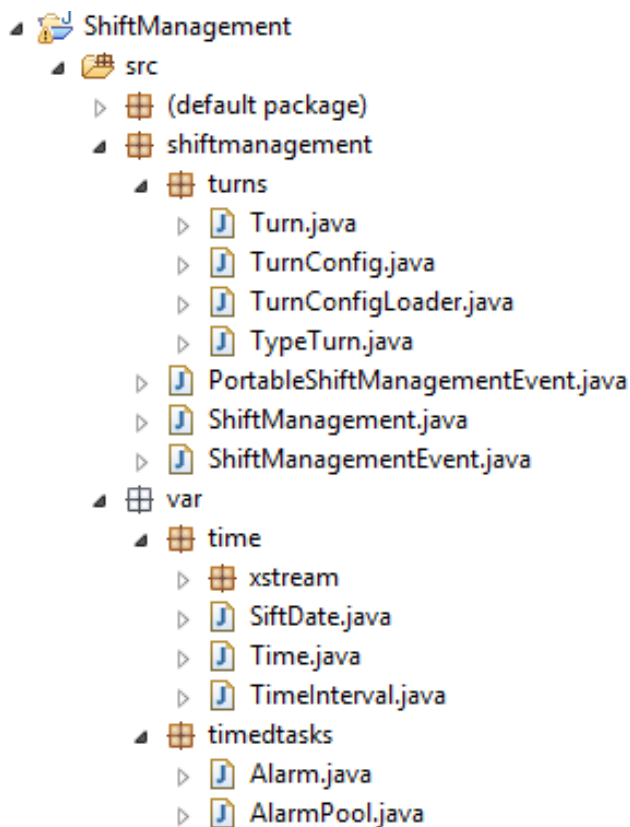
Anexo 2

Junto a esta memoria se entrega un CD con el siguiente contenido:

- Versión electrónica de la memoria
- Código fuente del programa
- Scripts de despliegue del servicio de rutas
- Plataforma de desarrollo eclipse

A continuación se incluye una representación visual en la que se ven los paquetes del programa realizado así como una tabla con el número de líneas de código realizadas.

Proyecto	Número de líneas de código
LunchBreaks	9800
GeoLocatingServicesAlt	6150
ShiftManagement	1100
mHealth Demonstrator	2300
<i>Total propias</i>	<i>19.000</i>
<i>Total ecosistema (realizado por otras personas)</i>	<i>60.000</i>



- ▲ LunchBreaks
 - ▲ src
 - ▲ es.urjc.ia.at.delaunaylunchbreaks
 - ▷ DelaunayTriangulation.java
 - ▷ NeighborhoodGraph.java
 - ▲ lunchBreaks
 - ▷ colors
 - ▷ debug
 - ▲ generic
 - ▷ alarms
 - ▷ satisfaction
 - ▷ CountTable.java
 - ▲ output
 - ▷ generic
 - ▷ log
 - ▷ xls
 - ▷ StatsWriter.java
 - ▲ principal
 - ▲ basics
 - ▷ HourTable.java
 - ▷ Interval.java
 - ▷ Range.java
 - ▷ Time.java
 - ▷ preferences
 - ▷ satisfaction
 - ▲ sorted
 - ▷ ExternalSortedList.java
 - ▷ IntFloatSorting.java
 - ▷ SortedAmbulances.java
 - ▷ SortedFromNumber.java
 - ▷ SortedRankingAmbulances.java
 - ▷ stats
 - ▷ Ambulance.java
 - ▷ AmbulanceState.java
 - ▷ EatState.java
 - ▷ IAgentID.java
 - ▷ InterruptedState.java
 - ▷ Turno.java
 - ▷ TurnoConfig.java
 - ▷ TypeTurn.java
 - ▲ realtiming
 - ▷ Range.java
 - ▷ SiftDate.java
 - ▷ Turn.java
 - ▷ TurnConfig.java
 - ▷ xml
 - ▷ znewimpl
 - ▷ LunchBreakConfiguration.java
 - ▷ LunchBreakEvent.java
 - ▷ LunchBreaksAssignment.java
 - ▷ LunchBreaksModule.java
 - ▷ lunchbreaks2

- ▲ GeolocationServicesAlt
 - ▲ src
 - ▲ es.urjc.ia
 - ▲ geo
 - ▲ location
 - ▷ converter
 - ▷ datum
 - ▷ DistanceCalculator.java
 - ▲ routing
 - ▲ entry
 - ▷ json
 - ▷ RouteEntry.java
 - ▷ exception
 - ▲ provider
 - ▷ offline.euclidean
 - ▲ online.http
 - ▷ googlemap
 - ▷ mapquest
 - ▷ youmavigation
 - ▷ AbstractHttpProvider.java
 - ▷ ProviderConnectionException.java
 - ▷ SWait.java
 - ▷ AbstractProvider.java
 - ▷ IRouteProvider.java
 - ▷ ProviderStats.java
 - ▲ proxy
 - ▷ chainable
 - ▷ BalancedRoutingProxy.java
 - ▷ IRoutingProxy.java
 - ▷ RoutingProxy.java
 - ▲ rmi
 - ▷ clientside
 - ▷ serverside
 - ▷ BalancedRoutingServiceNode.java
 - ▷ IRMIRouteProvider.java
 - ▷ RMIRegistry.java
 - ▷ RoutingServerExecuter.java
 - ▷ PrintableStatistics.java
 - ▷ Route.java
 - ▷ tests
 - ▲ var.cache
 - ▷ fifo
 - ▷ lru
 - ▷ lu
 - ▷ ACacheElement.java
 - ▷ ICache.java
 - ▷ ICacheExt.java
 - ▷ ICount.java
 - ▷ IKey.java
 - ▷ ITime.java

