

Aplicación de Métodos Formales para la Coordinación en Tiempo Real de Ambulancias en Situaciones de Emergencia

pchoquea@ulasalle.edu.pe

Rodrigo Cesar Ancori Tacca
Ingeniería de Software
Universidad La Salle
Arequipa, Perú
rancorit@ulasalle.edu.pe

Elías Efraín Manchego Navarro
Ingeniería de Software
Universidad La Salle
Arequipa, Perú
emanchego@ulasalle.edu.pe

Evelyn Milagros Chipana Ramos
Ingeniería de Software
Universidad La Salle
Arequipa, Perú
echipana@ulasalle.edu.pe

Rodrigo Alonso Ruelas Lope
Ingeniería de Software
Universidad La Salle
Arequipa, Perú
rruelasl@ulasalle.edu.pe

Paul Choque Adco
Ingeniería de Software
Universidad La Salle
Arequipa, Perú

Keywords—Component; formatting; style; styling; insert (key words)

Abstract— En situaciones de emergencia, la coordinación eficiente y en tiempo real de ambulancias es crucial para salvar vidas. Los sistemas tradicionales suelen fallar debido a la naturaleza impredecible de emergencias como accidentes de tráfico, desastres naturales o eventos masivos. Este trabajo propone un sistema basado en métodos formales para la asignación y coordinación de ambulancias, asegurando una verificación y validación rigurosas. El sistema propuesto optimiza el despacho de ambulancias, minimiza los errores y maximiza la eficiencia operativa. Los

métodos formales permiten un modelado matemático, garantizando la fiabilidad del sistema en condiciones críticas. El modelo desarrollado mejora la eficiencia y seguridad en la toma de decisiones para el despliegue en tiempo real de ambulancias, apoyado en ajustes dinámicos y capacidades predictivas.

Palabras clave: Coordinación en tiempo real, métodos formales, despacho de ambulancias, gestión de emergencias, validación de sistemas.

I. INTRODUCCIÓN

La coordinación en tiempo real de ambulancias durante emergencias es fundamental para minimizar los tiempos de respuesta y salvar vidas. Los enfoques tradicionales, como la planificación manual de rutas y la gestión de recursos, a menudo fallan debido a la imprevisibilidad inherente de las emergencias. Accidentes de tráfico, desastres naturales y eventos masivos pueden causar demoras, ineficiencias y asignaciones subóptimas de recursos, lo que impacta negativamente en la efectividad de la respuesta.

Este trabajo aborda la necesidad de un sistema robusto y automatizado que emplee métodos formales (herramientas matemáticas y computacionales) para asegurar una correcta asignación y coordinación en tiempo real de ambulancias. Estos métodos permiten la verificación y validación rigurosa del sistema, garantizando que las soluciones propuestas sean seguras y eficientes, minimizando así los errores en situaciones críticas.

II. OBJETIVO

El objetivo principal de esta investigación es desarrollar un modelo formal que optimice la coordinación en tiempo real de ambulancias, reduciendo los tiempos de respuesta y mejorando la utilización de recursos en escenarios de emergencia. Los objetivos específicos son los siguientes:

- Desarrollar un modelo formal para la coordinación en tiempo real de ambulancias, optimizando los tiempos de respuesta y el uso de recursos disponibles.
- Implementar un proceso de validación y verificación riguroso, asegurando que el sistema cumpla con los requisitos de los usuarios (validación) y que se haya desarrollado correctamente (verificación).
- Reducir el tiempo de respuesta en emergencias mejorando la asignación y el despacho de ambulancias mediante la optimización basada en modelos matemáticos y pruebas formales.

III. REQUERIMIENTOS FUNCIONALES

Para que el sistema propuesto funcione de manera eficiente y cumpla con los objetivos establecidos, se definen los siguientes requerimientos funcionales:

- Coordinación en tiempo real: El sistema debe poder monitorear en tiempo real la ubicación de las ambulancias, la disponibilidad de hospitales y las condiciones del tráfico, tomando decisiones de asignación automáticamente.
- Adaptación dinámica a las condiciones del entorno: El sistema debe ser capaz de ajustar en tiempo real las rutas de las ambulancias y la asignación de recursos según cambios en las condiciones del tráfico, disponibilidad de hospitales y nuevos eventos de emergencia.
- Predicción y prevención proactiva de la demanda: El sistema debe integrar capacidades de análisis predictivo para anticipar la demanda de ambulancias en función de datos históricos y patrones de emergencia. Esto permitirá pre-posicionar ambulancias en áreas con mayor probabilidad de incidentes, optimizando la respuesta antes de que ocurra una emergencia.

IV. ANTECEDENTES

La necesidad de un sistema optimizado para la respuesta en emergencias ha sido ampliamente reconocida, y varios trabajos previos proporcionan valiosas aportaciones en esta área:

- K. Holtermann, Desarrollo de sistemas de servicios de emergencias médicas: experiencia de los Estados Unidos de América para países en desarrollo. En este documento se analiza la experiencia de Estados Unidos en la creación de sistemas de servicios de emergencias médicas (SSEM), abordando la importancia de la planificación y organización para

- B. M. Nuñez Rojas, Implementación de un Sistema de Gestión de Emergencias en Centrales de Emergencias (etapa de prueba). Este trabajo de investigación se centra en la organización y los mecanismos necesarios para una respuesta coordinada ante emergencias, incluyendo la capacitación del personal y la optimización de recursos.
- F. Vítolo, Evacuación de hospitales: Principios básicos. Se establece un protocolo que detalla las acciones y responsabilidades del personal hospitalario en situaciones de emergencia, enfatizando la importancia de la coordinación y la evacuación.
- Clasificación y estándares mínimos para los equipos médicos de emergencia, Organización Panamericana de la Salud. Un análisis sobre la estructura de gestión de incidentes y la necesidad de contar con un centro de operaciones de emergencia (COE) para garantizar la efectividad en la respuesta a crisis.
- S. Lorenzo Martinez, J. J. Mira Solves y O. Moracho del Rio, La gestión por procesos en instituciones sanitarias. Este documento discute cómo la gestión por procesos puede mejorar la atención en emergencias, destacando la participación de profesionales no gestores en la identificación y resolución de problemas.
- R. Vásquez-Alva, C. Luna-Muñoz y C. M. Ramos-Garay, "El triage hospitalario en los servicios de emergencia". Un estudio que examina la saturación en los servicios de salud y su impacto en la atención de emergencias, especialmente en áreas urbanas con creciente demanda.
- R. Llewelyn-Davies y H. M. C. Macaulay, Planificación y administración de hospitales. Este trabajo ofrece directrices generales para la planificación y administración de hospitales, abordando los obstáculos y errores comunes en el establecimiento de servicios de salud.

V. ESPECIFICACIONES FORMALES Y VDM++

1. VDM++

Vienna Development Method (VDM) es uno de los métodos formales orientados a modelos más antiguos para el desarrollo de sistemas y software basados en computadora. Consiste en un grupo de lenguajes matemáticamente bien fundamentados y herramientas para expresar y analizar modelos de sistemas durante las primeras etapas de diseño, antes de que se hagan costosos compromisos de implementación. La construcción y el análisis del modelo ayudan a identificar áreas de incompletitud o ambigüedad en las especificaciones informales del sistema, y proporcionan cierto nivel de confianza en que una implementación válida tendrá propiedades clave, especialmente las de seguridad o protección.

2. DEFINICIONES DE CLASES

Los modelos en VDM++ consisten en un conjunto de clases. Una clase representa una colección de objetos que comparten elementos comunes como atributos u operaciones. La estructura de la descripción de una clase se muestra en la figura 1. La clase se representa con la palabra reservada `class`, seguida por el nombre de la clase. La descripción consta de varios bloques, precedidos por la palabra reservada que indica el tipo de elemento descrito en dicho bloque. En la figura 1 se muestra la estructura de una clase en VDM++, que incluye: variables de instancia, representando el estado interno del objeto; tipos, valores, funciones y operaciones, que definen los tipos de datos, constantes y acciones de la clase; un bloque de `thread`, que controla el comportamiento dinámico; y un bloque de sincronización (`sync`), encargado de gestionar la concurrencia entre hilos. Esta estructura permite modelar el comportamiento estático y dinámico de los sistemas en VDM++.

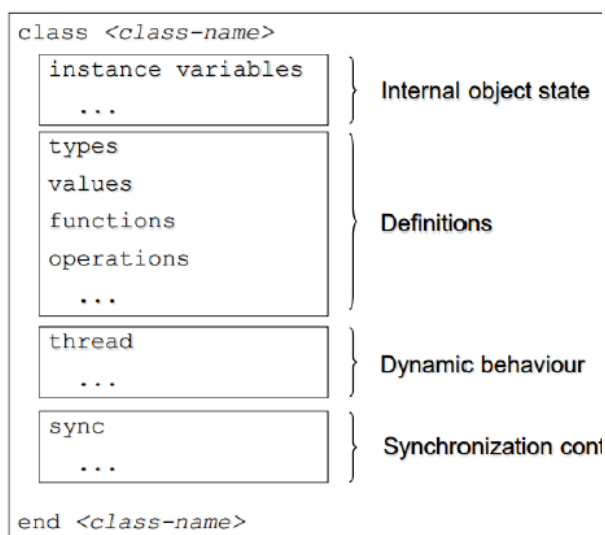


Figura 1. Estructura de Clase en VDM++

VI. PROPUESTA

El objetivo es que el sistema coordinará el despliegue de ambulancias en tiempo real hacia emergencias, asegurando una rápida respuesta. Al igual que los sensores y alarmas en el sistema de detección de gases, nuestro sistema deberá contar con mecanismos de monitoreo y respuesta automática para asignar la ambulancia más cercana y notificar sobre situaciones de riesgo.

1. Requerimientos

Para la determinación de los requerimientos en el sistema de coordinación en tiempo real de ambulancias en situaciones de emergencia, se debe de tener en cuenta los siguientes:

- R1: El sistema debe gestionar la asignación de ambulancias para responder a emergencias de forma rápida y eficiente.
- R2: El sistema debe ser capaz de calcular la distancia más corta entre la ubicación de la ambulancia y la emergencia.
- R3: Cada ambulancia debe estar equipada con un sistema de localización para reportar su ubicación en tiempo real.
- R4: El sistema debe monitorear el estado de las ambulancias (disponible, en camino, ocupada).
- R5: En caso de una emergencia, el sistema debe enviar un mensaje a la ambulancia más cercana y cambiar su estado a "en ruta".
- R6: El sistema debe gestionar las situaciones donde más de una ambulancia es requerida simultáneamente.
- R7: La ambulancia deberá estar disponible nuevamente una vez que la emergencia haya sido atendida y el sistema debe reflejar este cambio de estado.

2. Sistema de control

El Sistema de Coordinación de Ambulancias actúa como el controlador central del proyecto, gestionando la asignación eficiente de ambulancias a emergencias en tiempo real. Similar al sistema de detección de gases peligrosos, este sistema monitorea las ubicaciones y el estado de las ambulancias (disponible, en camino, ocupada) y asigna la ambulancia más cercana a una emergencia según su gravedad y proximidad. Además, el sistema maneja diferentes tipos de emergencias (accidentes, incendios, problemas médicos) y prioriza las más críticas. Una vez que una emergencia ha sido atendida, la ambulancia se libera y queda disponible para otra asignación. Este sistema asegura una respuesta rápida y optimizada para cada situación de emergencia, mejorando la eficiencia del manejo de recursos en situaciones críticas.

VII. ESPECIFICACIONES FORMALES

Las especificaciones formales en VDM++ permiten modelar el comportamiento de las entidades clave de nuestro sistema (ambulancias, hospitales, ciudad, emergencias y la coordinación). Utilizamos tipos básicos como **NAT** para los identificadores y coordenadas, y se definen estados de las ambulancias (disponible, en camino, ocupada). Cada clase tiene sus variables de instancia y métodos correspondientes que controlan el comportamiento del sistema, como el cálculo de la distancia entre una ambulancia y una emergencia, o la asignación automática de la ambulancia más cercana.

```

1: class Ambulance
2: types
3: State = <AVAILABLE> | <EN_ROUTE> | <OCCUPIED>;
4:
5: instance variables
6: id: nat; -- ID único de la ambulancia
7: location: nat; -- Coordenada actual de la ambulancia
8: state: State; -- Estado actual de la ambulancia
9:
10: operations
11: -- Constructor de la clase Ambulance
12: public Ambulance: nat * nat ==> Ambulance
13: Ambulance(ambId, initLocation) ==
14: (
15:   id := ambId;
16:   location := initLocation;
17:   state := <AVAILABLE>
18: );
19:
20: -- Asignar la ambulancia a una emergencia
21: public AssignToEmergency: nat ==> ()
22: AssignToEmergency(emergencyLocation) ==
23: (
24:   location := emergencyLocation;
25:   state := <EN_ROUTE>
26: )
27: pre state = <AVAILABLE>;
28:
29: -- Marcar la ambulancia como disponible después de la emergencia
30: public MarkAvailable: () ==> ()
31: MarkAvailable() ==
32: (
33:   state := <AVAILABLE>
34: )
35: pre state = <OCCUPIED>;
36:
37: -- Llegada a la emergencia
38: public ArriveAtEmergency: () ==> ()
39: ArriveAtEmergency() ==
40: (
41:   state := <OCCUPIED>
42: )
43: pre state = <EN_ROUTE>;
44: end Ambulance

```

Figura 2. La clase Ambulancia modela una ambulancia con un identificador único, una ubicación (coordenadas x, y), y un estado que puede ser "Disponible", "En camino", u "Ocupada". Define operaciones para cambiar su ubicación, asignarla a una emergencia (cambiando su estado a "En camino") y liberarla una vez terminada la emergencia, devolviéndole al estado "Disponible". Además, permite consultar su estado actual para gestionar su disponibilidad dentro del sistema de coordinación de emergencias.

```

1: class City
2: instance variables
3: ambulances: map nat to Ambulance := {}; -- Mapa de ambulancias en la ciudad
4: hospitals: map nat to Hospital := {}; -- Mapa de hospitales en la ciudad
5: emergencies: set of Emergency := {}; -- Emergencias en la ciudad
6:
7: operations
8: -- Añadir una ambulancia a la ciudad
9: public AddAmbulance: Ambulance ==> ()
10: AddAmbulance(amb) ==
11:   ambulances := ambulances union {amb.id -> amb};
12:
13: -- Añadir un hospital a la ciudad
14: public AddHospital: Hospital ==> ()
15: AddHospital(hosp) ==
16:   hospitals := hospitals union {hosp.id -> hosp};
17:
18: -- Añadir una emergencia a la ciudad
19: public AddEmergency: Emergency ==> ()
20: AddEmergency(emerg) ==
21:   emergencies := emergencies union {emerg};
22:
23: -- Asignar la ambulancia más cercana a una emergencia
24: public AssignAmbulanceToEmergency: Emergency ==> nat
25: AssignAmbulanceToEmergency(emerg) ==
26: (
27:   dol nearestAmbulance: nat := 0;
28:   dol minDistance: nat := 10000; -- Valor inicial alto para comparación
29:   for all ambId in set dom ambulances do
30:     let amb = ambulances[ambId] in
31:     (
32:       if amb.state = <AVAILABLE> and abs(amb.location - emerg.location) <
minDistance then
33:         (
34:           nearestAmbulance := amb.id;
35:           minDistance := abs(amb.location - emerg.location)
36:         )
37:       );
38:     ambulances[nearestAmbulance].AssignToEmergency(emerg.location);
39:   return nearestAmbulance
40: )
41: pre emergencies <> {} and nearestAmbulance <> 0;
42: end City

```

Figura 3. La clase Ciudad gestiona una ciudad con ambulancias, hospitales y emergencias. Permite agregar ambulancias, hospitales y emergencias, almacenándose en mapas o conjuntos. Su operación principal asigna la ambulancia más cercana a una emergencia, calculando la distancia mínima

entre la ambulancia y la ubicación de la emergencia, y luego la asigna a la emergencia. También maneja la disponibilidad de ambulancias para asegurar que solo las disponibles sean asignadas a emergencias activas.

```

1: class CoordinationSystem
2: instance variables
3: city: City; -- La ciudad donde se gestionan las ambulancias y emergencias
4:
5: operations
6: -- Constructor de la clase CoordinationSystem
7: public CoordinationSystem: City ==> CoordinationSystem
8: CoordinationSystem(c) ==
9: (
10:   city := c
11: );
12:
13: -- Manejar una nueva emergencia: Añadir emergencia y asignar ambulancia
14: public HandleEmergency: nat ==> ()
15: HandleEmergency(emergencyLocation) ==
16: (
17:   -- Crear una nueva emergencia
18:   dol emerg : Emergency;
19:   emerg := new Emergency(emergencyLocation);
20:   city.AddEmergency(emerg);
21:
22:   -- Declarar e inicializar la ambulancia asignada
23:   let assignedAmbulance : nat = city.AssignAmbulanceToEmergency(emerg) in
24:   (
25:     IO.println("Ambulancia " ^ nat2str(assignedAmbulance) ^ " asignada a la
emergencia en la ubicación " ^ nat2str(emergencyLocation) ^ ".")
26:   )
27: )
28: pre city <> nil; -- Verificar que la ciudad esté correctamente inicializada
29: end CoordinationSystem

```

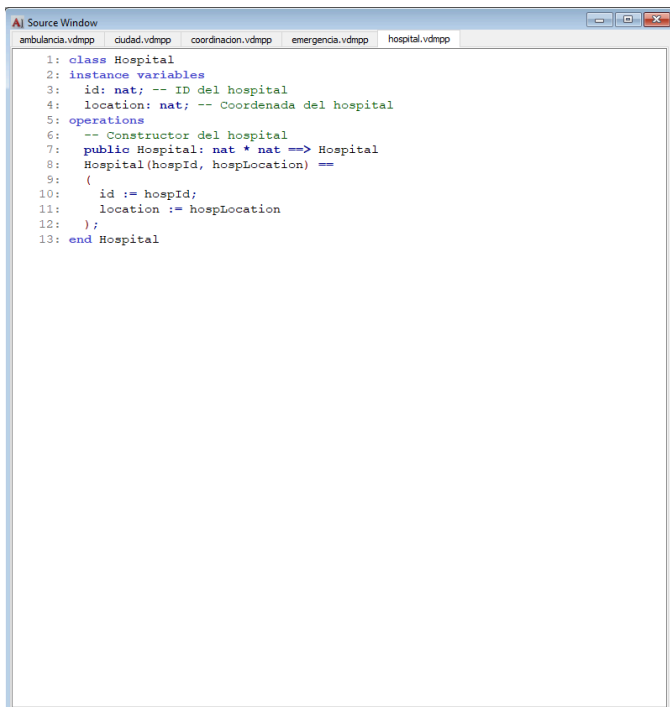
Figura 4. La clase Coordinación gestiona la coordinación de emergencias y ambulancias dentro de una ciudad. A través de su constructor, se inicia con una ciudad, y su operación principal, Handle Emergency, añade una nueva emergencia y asigna la ambulancia más cercana para atenderla. La emergencia es creada y agregada al sistema de la ciudad, y luego la ambulancia más próxima es asignada, notificando el ID de la ambulancia asignada y la ubicación de la emergencia.

```

1: class Emergency
2: instance variables
3: location: nat; -- Coordenada de la emergencia
4:
5: operations
6: -- Constructor de la emergencia
7: public Emergency: nat ==> Emergency
8: Emergency(emergencyLocation) ==
9: (
10:   location := emergencyLocation
11: );
12: end Emergency

```

Figura 5. La clase Emergencia representa una emergencia con una ubicación específica. Tiene una única variable de instancia, location, que almacena la coordenada de la emergencia. Su constructor permite inicializar una emergencia con una ubicación dada, asignando dicha coordenada a la variable location. Esta clase facilita el manejo de emergencias dentro del sistema de coordinación de ambulancias.



```

1: class Hospital
2: instance variables
3: id: nat; -- ID del hospital
4: location: nat; -- Coordenada del hospital
5: operations
6: -- Constructor del hospital
7: public Hospital: nat * nat ==> Hospital
8: Hospital(hospId, hospLocation) ==
9: (
10: id := hospId;
11: location := hospLocation
12: );
13: end Hospital

```

Figura 6. La clase Hospital modela un hospital con un identificador único (id) y una ubicación (location). Su constructor inicializa el hospital con un ID y una coordenada específica. Esto permite que los hospitales se gestionen dentro del sistema de la ciudad, facilitando su identificación y uso en operaciones relacionadas con la coordinación de ambulancias y emergencias.

REFERENCES

- K. Holtermann, *Desarrollo de sistemas de servicios de emergencias médicas: experiencia de los Estados Unidos de América para países en desarrollo*, Washington, D.C.: OPS, 2003. ISBN: 92 75 32461. [En línea]. Disponible en: <https://www.binasss.sa.cr/opac-ms/media/digitales/Desarrollo%20de%20sistemas%20de%20servicios%20de%20emergencias%20m%C3%A9dicas.%20Experiencia%20de%20Estados%20Unidos%20de%20Am%C3%A9rica%20para%20opa%C3%ADses%20en%20desarrollo.pdf>
- B. M. Nuñez Rojas, *Implementación de un Sistema de Gestión de Emergencias en Centrales de Emergencias (etapa de prueba)*, Tesis de Maestría, Programa Académico de Maestría en Gestión Pública, Escuela de Posgrado, Universidad César Vallejo, Perú, 2022. [En línea]. Disponible en: https://repositorio.ucv.edu.pe/bitstream/handle/20.500.12692/106210/Nuñez_RBM-SD.pdf?isAllowed=y&sequence=1
- F. Vítolo, *Evacuación de hospitales: Principios básicos*, Biblioteca Virtual NOBLE, marzo 2019. [En línea]. Disponible en: <http://asegurados.descargas.nobleseguros.com/download/pods/March2019/xgH80pqsFPuu2QfxmtbP.pdf> [Accedido: 12-sep-2024].
- Organización Panamericana de la Salud, *Clasificación y estándares mínimos para los equipos médicos de emergencia*, 2023. ISBN: 978-92-75-32782-1 (PDF), 978-92-75-12782-7 (versión impresa). [En línea]. Disponible en: <https://bvs.minsa.gob.pe/local/MINSA/6905.pdf> [Accedido: 12-sep-2024].
- S. Lorenzo Martinez, J. J. Mira Solves y O. Moracho del Rio, *La gestión por procesos en instituciones sanitarias*, Módulo 8, Máster en Dirección Médica y Gestión Clínica, Fundación Hospital Alcorcón, Universidad Miguel Hernández, 2018. [En línea]. Disponible en:

https://calite.umh.es/data/docs/110/Gestion_procesos.pdf [Accedido: 12-sep-2024].

R. Vázquez-Alva, C. Luna-Muñoz y C. M. Ramos-Garay, "El triage hospitalario en los servicios de emergencia", *Rev. Fac. Med. Hum.*, vol. 19, no. 1, pp. 90-100, enero 2019. DOI: 10.25176/RFMH.v19.n1.1797. [En línea]. Disponible en: <https://inicib.urp.edu.pe/cgi/viewcontent.cgi?article=1092&context=rjfmh> [Accedido: 12-sep-2024].

R. Llewelyn-Davies y H. M. C. Macaulay, *Planificación y administración de hospitales*, Publicación Científica No. 191, Organización Panamericana de la Salud, Washington, D.C., E.U.A., 2019. [En línea]. Disponible en: <https://iris.paho.org/bitstream/handle/10665.2/1239/40223.pdf> [Accedido: 12-sep-2024].