# Modelamiento y Validación con VDM++ de un Sistema de Control Automatizado de Velocidad en Vehículos usando Bluetooth Low Energy Beacons

Alvin Chunga Mamani, Estudiante[1]
[1]Universidad Nacional de San Agustín, Perú, alvin@episunsa.edu.pe.

***Abstract*– Con el crecimiento del parque automotor en la ciudad de Arequipa, aumenta el número de accidentes de tránsito por el exceso de velocidad en los vehículos, al analizar esta situación de plantea el desarrollo de un sistema de control de velocidad automatizado en vehículos usando dispositivos Bluetooth Low Energy Beacons para controlar la velocidad limite en zonas críticas de manera segura sin dañar los vehículos usando rompemuelles. Dado que es un sistema de control crítico se utiliza especificación formal para su desarrollo y verificación utilizando VDM.***

***Keywords*—Bluetooth low energy, Bluetooth beacons, VDM, beacons, car velocity control.***

## I. INTRODUCCIÓN

En tiempos recientes en la creación de Sistemas de software complejos se tiende a cometer errores en la especificación de requerimientos antes de haber comenzado a desarrollarlo, en dichos sistemas de se requiere de una precisión alta y los errores hacen que se produzca una demora en la producción de dicho sistema además de generar inmensos gastos al tener requerimientos ambiguos. Esto se produce a causa de usar lenguaje natural el cual es ambiguo y se presta a diferentes interpretaciones [1].

El uso de Especificaciones Formales para la realización de un sistema provee una serie de ventajas al usar un lenguaje estructurado no ambiguo usando notación matemática para la especificación de requerimientos en la etapa inicial del software, para poder detectar y corregir errores en los requerimientos antes de empezar a desarrollar el software.

Sistemas Complejos que requieren un nivel de precisión critico en donde errores en el software producen un inminente fallo del sistema, se requiere un análisis detallado para producir un software que cumpla con los requisitos establecidos y lograr la correcta funcionalidad del Sistema sin fallos. Estos sistemas son críticos en el que un error en la elaboración del software conlleva a una inminente catástrofe.

Se planea implementar un Sistema Automatizado para controlar la Velocidad de los Vehículos mediante el uso de dispositivos Bluetooth-Beacons, para ello se utilizará la herramienta VDM++ para modelar el sistema usando métodos formales propios de la herramienta y validarlos usando la misma herramienta, creando test propios que sirvan para detectar errores en los modelos y corregirlos.

## II. TRABAJOS RELACIONADOS

Sune Wolf [2] presenta un modelo ejecutable del protocolo de comunicación entre un sistema de auto-defensa utilizado a bordo de aviones de combate usando Vienna Development Method (VDM) y validado usando distintos escenarios para cubrir diferentes casos que puedan ocurrir en el sistema. Una alternativa de análisis de los diferentes escenarios en vez de hacerlos manualmente, que consume mucho más tiempo y conlleva muchos errores.

Shota, Michitoshi, Erjing y Masaru [12,14] presentan un sistema para gestionar la asistencia de estudiantes utilizando dispositivos BLE (Bluetooth Low Energy) el cual resuelve el problema de las SmartCard que los estudiantes usaban para registrar asistencia y salirse de clases, además del tiempo que empleaban los estudiantes en escanear su tarjeta por un solo dispositivo terminal. Se realizó una aplicación para dispositivos Android que recibe la señal del Beacon en el aula periódicamente, el cual garantiza la asistencia a clases de los estudiantes.

Kumaresan y Gokulnath [13] plantean un sistema de monitoreo y seguimiento de vehículos utilizando IBeacon para guardar la información del vehículo y notificar a dispositivos Smartphone la ubicación del vehículo que pasa por determinada calle. Con la autorización del usuario se accederá a la información guardada en el IBeacon y se calculará la posición y velocidad en base a las señales emitidas por el IBeacon y los dispositivos Smartphone.

Gaurav y Varun [15] desarrollaron una herramienta para la vigilancia en tiempo real de una institución, registrar

asistencias y, enviar enlaces a páginas web y notificaciones para comunicarse con los estudiantes.

En los trabajos vistos se ha realizado sistemas de monitoreo y control usando la tecnología de dispositivos BLE.

## III. ESPECIFICACION FORMAL Y VDM++

### A. *Definición*

Una especificación formal usa notación matemática para describir de manera precisa las propiedades que un sistema de información debe tener, sin preocuparse por la forma de obtener dichas propiedades. Describe lo que el sistema debe hacer sin decir cómo se va a hacer. Esta abstracción hace que las especificaciones formales sean útiles en el proceso de desarrollar un sistema, porque permiten responder preguntas acerca de lo que el sistema hace con confianza, sin la necesidad de tratar con una gran cantidad de información no relevante que se encuentra en el código de programa del sistema en un lenguaje de programación cualquiera, o especular sobre el significado de frases en un impreciso Pseudocódigo [3].

### B. *VDM++*

Vienna Development Method(VDM) es una colección de técnicas para especificación formal y desarrollo de software. VDM se origina en los laboratorios del IBM en Viena en 1970. El primer lenguaje soportado por el método de VDM se llamó VDL por sus siglas en inglés (Vienna Definition Language), el cual evolucionó en el lenguaje especificado VDM-SL el cual fue ISO estandarizado. A través de los años, las extensiones fueron definidas al modelo orientado a objetos [4].

### C. *Class Definition*

En VDM++ un modelo consiste en un conjunto de clases especificadas. Existen clases pasivas y activas, las activas son aquellas que tienen su propio hilo de control y no necesitan de eventos externos para trabajar, mientras que las pasivas siempre son manipuladas desde el hilo de control de una clase activa. El termino objeto es usado para denotar una instancia de una clase. Se pueden crear múltiples instancias de una clase usando el operador `new` que retorna una referencia al objeto.

Una especificación de clase tiene los siguientes componentes como se observa en la Fig. 1 [9]:

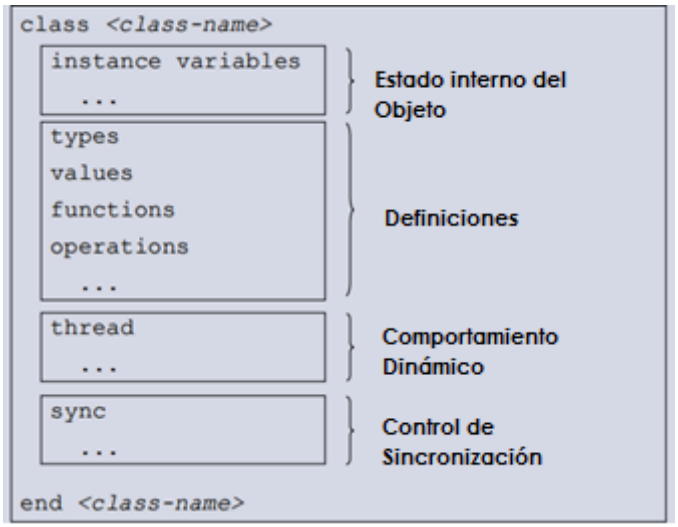


**Fig. 1** Esquema de la Clase.

a) Encabezado de Clase: El encabezado de clase es el nombre de la clase e información de herencia. Una y múltiples herencias son soportadas
b) Variables de Instancia: Las variables de instancia son el estado del objeto que consiste en un conjunto de variables de tipo que pueden ser simples como *bool* o *nat*, o tipos complejos como *sets*, *sequences*, *maps*, *tuples*, *records* y *objetos.*
c) Operaciones: Las operaciones son los métodos de clase que pueden modificar el estado, pueden ser definidos implícitamente, usando pre y post condiciones, o explícitamente, usando declaraciones imperativas y opcionalmente pre-post condiciones.
d) Funciones: Las funciones son similares a las operaciones, pero el cuerpo de estas es una expresión en lugar de una declaración imperativa.
e) Sincronización: La sincronización en VDM++ es asíncrona.
f) Hilos: Un hilo es una secuencia de sentencias que se ejecutan hasta la terminación en cuyo punto el hilo muere. Es posible especificar subprocesos que nunca terminan.

### D. *Expresiones*

Las expresiones se usan para describir cálculos que no producen efectos secundarios; Esto significa que nunca pueden afectar el valor de una variable de instancia (a menos que contenga una llamada a la operación). VDM ++ tiene 25 categorías diferentes de expresiones.

Una de las principales categorías utilizadas para definir precondiciones, condiciones post e invariantes son expresiones cuantificadoras. La cuantificación de expresiones es un tipo de expresión lógica. Se utilizan con frecuencia

cuando es necesario una afirmación sobre una colección de valores. Hay dos tipos de cuantificadores de expresión: cuantificador universal (`forall`) y cuantificador existencial (`exist`) [10].

### *E. Herramienta: VDM++ ToolBox*

VDMTools es un conjunto de herramientas que le permiten desarrollar y analizar modelos precisos de sistemas informáticos. Cuando se usan en las primeras etapas del desarrollo del sistema, estos modelos pueden servir como especificaciones del sistema, o como una ayuda para comprobar la consistencia y completitud de los requerimientos de los usuarios [7].

VDM++ToolBox provee soporte para validación de la especificación a través de la ejecución de pruebas utilizando un intérprete. El intérprete permite ejecutar partes de la especificación utilizando valores seleccionados por el desarrollador (casos de prueba) [4].

## IV. BLUETOOTH LOW ENERGY

### *A. Definición.*

Bluetooth Low Energy es similar al clásico Bluetooth, ya que comparten el mismo espectro digital y la técnica de modulación, así como la tasa de bits, la única diferencia es el espaciamiento de la capa física que es de 2 megaHertz en Bluetooth clásico mientras que en Bluetooth Low Energy es de un megaHertz. Como su nombre indica, la energía baja de Bluetooth usada para el consumo bajo de la energía y los dispositivos Bluetooth Low Energy son mucho más baratas que los dispositivos clásicos de Bluetooth [11].

Un Beacon en tecnología inalámbrica es el concepto de la difusión de pequeñas piezas de información como se muestra en Fig. 2. La información puede ser cualquier cosa, desde datos ambientales (temperatura, presión de aire, humedad, etc.) hasta datos de micro-localización (seguimiento de activos, minoristas, etc.) o datos de orientación (aceleración, rotación, etc.).


**Fig.2** Dispositivo Beacon

Los datos transmitidos son típicamente estáticos, pero también pueden ser dinámicos y cambiar con el tiempo. Con el uso de la energía baja de Bluetooth, los beacons se pueden diseñar para funcionar por años con una sola batería de tamaño de la moneda [8].

### *B. Características.*

En más detalles, Ibeacon son transmisores de baja complejidad que promueven una carga útil BLE particular con información de identificación:

A. UUID de proximidad (identificador universalmente único): valor de 128 bits que identifica de manera única a uno o más Ibeacon. Este identificador es obligatorio.
B. Major data: un número sin signo de 16 bits utilizado para segregar Ibeacon que tiene el mismo UUID de inmediatez. Este valor es voluntario.
C. Minor data: un número sin signo de 16 bits utilizado para separar Ibeacon que tiene el mismo UUID de inmediatez y el valor principal. Este valor también es voluntario.

Ibeacon que presentan el mismo UUID forman una región de faro. Entonces, cuando un dispositivo móvil con la aplicación activada BLE entre en la región ibeacon, recibirá una notificación apropiada. Estas aplicaciones también pueden controlar la distancia relativa al ibeacon, usando el valor de RSSI [13].

## V. CASO DE APLICACIÓN

### *A. Planteamiento del problema*

La ciudad de Arequipa es la segunda con un parque automotor más grande en Perú, lo que también implica un aumento en los accidentes de tránsito, según el INEI en el año 2014 Arequipa fue el departamento que registro la tasa de accidentes de tránsito más elevada con 6,518 accidentes por cada 100mil habitantes, Según el estudio del INEI, el 42.9% de accidentes de tránsito fue ocasionado por el exceso de velocidad [16].

Actualmente no se cuenta con un sistema de control de velocidad que restrinja rígidamente la velocidad de los vehículos en determinadas zonas como colegios, se ha tratado de solucionar esta problemática construyendo rompemuelles en para obligar a los vehículos a reducir la velocidad en determinadas zonas transitadas en las que podría ocurrir accidentes por el exceso de velocidad, como es el caso de los colegios.

Tal como se muestra en Fig. 3 la creación de rompemuelles es realizada en zonas cercanas a los colegios para obligar a los vehículos que transiten por la zona se vean obligados a reducir la velocidad al entrar a una zona muy transitada por escolares, para que no puedan ocurrir accidentes por un exceso de velocidad.

**Fig. 3** Rompemuelle tradicional.

Construir rompemuelles es una solución que se ha implementado en la mayoría de lugares en los que se necesita obligar a los conductores a reducir la velocidad de su vehículo, se cumple con el objetivo, pero al costo de que los vehículos sufren daños por causas de este por lo que también podría decirse que son un "rompeautos".

### *B. Propuesta*

Se ha considerado utilizar dispositivos Bluetooth Low Energy Beacons para el Sistema Propuesto, colocándolos en zonas estratégicas como se observa en Fig. 4, a fin de hacer que un automóvil que entre en el rango del dispositivo baje su velocidad periódicamente hasta la máxima indicada en el dispositivo Beacon.

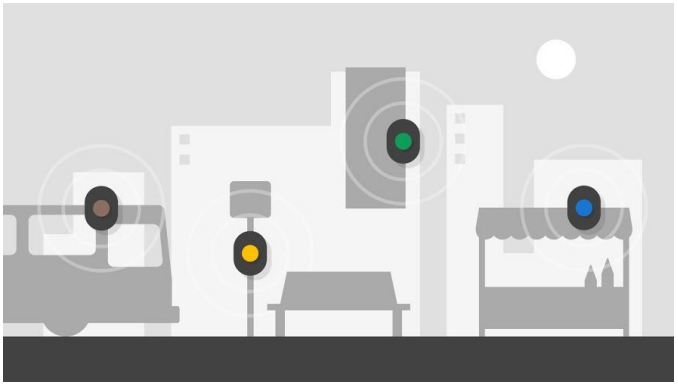

**Fig. 4** Ubicación de Dispositivos Beacon.

Los requerimientos identificados son:

**R1** Un sistema para el control de velocidad en Carros usando Beacons

**R2** El Beacon debe guardar un código que lo identifique para ser validad y la velocidad máxima que se le asigne.

**R3** Un carro puede estar en el rango de varios Beacons a la vez.

**R4** La velocidad del carro debe reducirse periódicamente si excede a la velocidad que marque algún Beacon en el rango de alcance.

### *C. Diagrama de Clases*

En Fig. 5 se muestra el diagrama de clases generado con la especificación realizada en las clases Beacon y Car, y la validación de los modelos usando la clase Test.

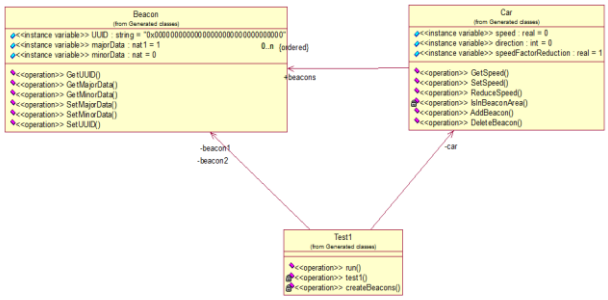


**Fig. 5** Diagrama de clases

*D. Especificación en VDM++*

Se utilizó la herramienta VDM++Toolbox.

En las Fig. 6 y 7 se presenta la especificación formal en VDM++.

```
4: -- Generated at 29-Sep-16 by the UML-VDM++ Link
5: --
6: class Beacon
7: types
8:  public string = seq of char;
9: instance variables
10:  public UUID : string:="0x00000000000000000000000000000000";
11:  public majorData : nat1:=1;
12:  public minorData : nat:=0;
13:  inv
14:   majorData>=1 and len UUID=34 and majorData<(2**16);
15: operations
16:  public
17:  GetUUID:() ==> string
18:  GetUUID() == return UUID;
19:  public
20:  SetUUID:string==>()
21:  SetUUID(uuid)==
22:   UUID:=uuid;
23:
24:  public
25:  GetMajorData:() ==> nat1
26:  GetMajorData() == return majorData;
27:  public
28:  SetMajorData:nat1==>()
29:  SetMajorData(major)==
30:   majorData:=major;
31:
32:  public
33:  GetMinorData:() ==> nat
34:  GetMinorData() == return minorData;
35:  public
36:  SetMinorData:nat==>()
37:  SetMinorData(minor)==
38:   minorData:=minor;
39:
40: end Beacon
```

**Fig. 6** Especificación de la clase Beacon.

En la Fig. 6 se modela el Beacon, especificando los componentes propios del dispositivo de manera formal.

En las líneas 10 a la 12 se especifica las propiedades que tiene un Beacon como son el UUID (Identificador propio de cada Beacon), majorData (majorData del Beacon) y minorData (minorData del beacon), de los cuales se utilizara el majorData para registrar el máximo de velocidad marcada por el Beacon.

Las operaciones (línea 15 en adelante) son de acceso(get) y modificación(set) de cada componente del Beacon para poder ser manipulado.

```
4: -- Generated at 29-Sep-16 by the UML-VDM++ Link
5: --
6: class Car
7:
8: instance variables
9:  public speed : real:=0;
10:  public direction : int:=0;
11:  public beacons : seq of Beacon:=[];
12:  public speedFactorReduction : real:=1.0;
13:  inv
14:   speed >= 0;
15:  inv
16:   speed < 300;
17:  inv
18:   direction > -2;
19:  inv
20:   direction < 2;
21:
22:
23: operations
24:  public
25:  GetSpeed:()==>(real)
26:  GetSpeed()==
27:  return speed;
28:  public
29:  SetSpeed:real ==>()
30:  SetSpeed(sp)==
31:  speed:=sp;
32:
33:  IsInBeaconArea :() ==> bool
34:  IsInBeaconArea()==
35:   return len beacons<>0;
36:  public ReduceSpeed : ()==> ()
37:  ReduceSpeed()==
38:   (
39:   dcl flag:bool;
40:   flag:=false;
41:   for p in beacons do
42:    if speed > p.GetMajorData()
43:    then flag:=true;
44:   if flag
45:   then speed:=speed-speedFactorReduction;
46:   )
47:  pre IsInBeaconArea();
48:  public DeleteBeacon:(Beacon)==>()
49:  DeleteBeacon(beacon)==
50:  (dcl nk : seq of Beacon := [];
51:   for elem in beacons do
52:    if elem.GetUUID() <> beacon.GetUUID()
53:    then nk:=nk^[elem];
54:   beacons:=nk;
55:  )
56:  pre IsInBeaconArea();
57:  public AddBeacon:(Beacon)==>()
58:  AddBeacon(beacon)==
59:  beacons:=(beacons^[beacon]);
60:
61: end Car
62:
```

**Fig. 7** Especificación de la clase Car

En la Fig. 7 se muestra el vehículo modelado en el lenguaje formal VDM simulando el comportamiento real de un vehículo. En las líneas 9 y 10 se crean variables para representar la velocidad y dirección del vehículo, en la línea 11 una secuencia de Beacons los cuales representan una lista, en la cual se añade un Beacon si el vehículo entro en su rango, y al salir del rango se elimina de la lista, en base a esta lista se escoge el Beacon que tenga la menor velocidad máxima en su majorData y será la velocidad que se le ponga como limite al vehículo mientras esté en su rango.

En la línea 13 a la 19 se definen las invariantes, que son las condiciones sobre las que trabaja el sistema, la primera invariante (línea 14) representa la velocidad del vehículo que siempre es mayor igual que 0, en la línea 16 la velocidad máxima a la que puede llegar un vehículo sin restricciones de velocidad, en la línea 18 y 20 se representa la dirección del vehículo que pueden ser 3 valores: -1 en sentido contrario, 0 sin velocidad y 1 si se encuentra en movimiento hacia adelante.

**R1** Está definido en la clase Car Fig.7.

**R2** Está definido en la clase Beacon Fig. 6 que considera 3 atributos, "*UUID"* representa el código de identificación propio del Beacon que consta de 16bytes línea 10, "*majorData*" representa el límite de velocidad que se le asigna al Beacon línea 11.

**R3** Está definido en la clase Car Fig 7., el atributo beacons que es una secuencia de Beacons línea 11, la adición de Beacons se da en la función "*addBeacon*" en las líneas 56 a la 59, la eliminación de Beacons se da en la función "*DeleteBeacon*" en las líneas 48 a la 55.

**R4** Está definido en la clase Car Fig 7. en la función "*ReduceSpeed*" que se encarga de la comprobación de velocidad del vehículo y los Beacons que estén en su rango reduciendo la velocidad periódicamente como se ve en las líneas 36 a la 46

*E. Validación*

```
>> rtinfo vdm.tc
 100%    1 Car`GetSpeed
 100%    2 Car`SetSpeed
 100%    2 Car`AddBeacon
 100%    2 Car`ReduceSpeed
 100%    1 Car`DeleteBeacon
 100%    3 Car`IsInBeaconArea
 100%    1 Test1`run
 100%    1 Test1`test1
 100%    1 Test1`createBeacons
 100%    4 Beacon`GetUUID
 100%    2 Beacon`SetUUID
 100%    3 Beacon`GetMajorData
 100%    1 Beacon`GetMinorData
 100%    2 Beacon`SetMajorData
 100%    3 Beacon`SetMinorData

Total Coverage: 100%
```

**Fig. 8** Validación del modelo.

En la Fig. 8 se muestra la validación del sistema realizado en VDM++ comprobando la funcionalidad y cobertura de todos los métodos implementados en las clases Car y Beacon en un ambiente simple de prueba en el que se simula la actividad que tendrá el sistema, modelado en la clase Test1 Fig. 9.

Test1.vpp

```
1: class Test1
2:
3: instance variables
4:  beacon1 : Beacon;
5:  beacon2 : Beacon;
6:  car : Car;
7:
8: operations
9:
10:  public run:()==>()
11:  run()==test1();
12:
13:  test1:() ==> ()
14:  test1() ==
15:  (
16:   createBeacons();
17:   car:= new Car();
18:   car.SetSpeed(80);
19:   car.SetSpeed(car.GetSpeed());
20:   car.AddBeacon(beacon1);
21:   car.ReduceSpeed();
22:   car.AddBeacon(beacon2);
23:   car.ReduceSpeed();
24:   car.DeleteBeacon(beacon1);
25:
26:
27:  );
28:  createBeacons:()==>()
29:  createBeacons()==
30:  (
31:   beacon1:=new Beacon();
32:   beacon2:=new Beacon();
33:   beacon1.SetUUID("0xEBEFD08370A247C89837E7B5634DF524");
34:   beacon2.SetUUID("0xEBEFD08370A247C89837E7B5634DF525");
35:   beacon1.SetMajorData(70);
36:   beacon2.SetMajorData(50);
37:   beacon1.SetMinorData(0);
38:   beacon2.SetMinorData(0);
39:   beacon1.SetMinorData(beacon1.GetMinorData());
40:  );
41: end Test1
```

**Fig. 9** Clase Test1.

En la Fig. 9, en la línea 16 se inicializan 2 beacons invocando a una función "*createBeacons"*, en esta función, se crean 2 beacons (línea 31 y 32), se les asigna sus UUID (línea 33 y 34) que es su identificador único, y en las líneas 35 y 36 se les asigna la velocidad máxima que tendrán almacenadas, uno con 70 y otro con 50. En la línea 17 se crea un nuevo vehículo, se le asigna una velocidad en la línea 18. En la línea 20 se añade un Beacon al vehículo, simulando que este entro en el radio de ese Beacon, entonces el Beacon es añadido a la lista de beacons del vehículo. En la línea 21 se invoca a la función "*ReduceSpeed"* la cual evalúa la lista de beacons en los que se encuentra el vehículo y analiza si la velocidad actual del vehículo es menor que la velocidad máxima indicada por los beacons, de no serlo procederá a disminuir la velocidad en un factor preestablecido, de lo contrario el vehículo permanecerá con la velocidad que tenga. En la línea 24 se simula que el vehículo ha salido del rango de un Beacon con lo que se invoca al método "*DeleteBeacon"* el cual remueve el Beacon de la lista del vehículo porque este ya no lo afecta.

## VI. RESULTADOS

La ciudad de Arequipa presenta un crecimiento del parque automotor cada año y con este también los accidentes que se producen por el exceso de velocidad, el sistema planteado en este trabajo muestra el uso de tecnología de bajo costo como son los Beacons en un sistema de control de velocidad de vehículos, siendo el reemplazo a los rompemuelles que dañan los vehículos y no garantizan el control de velocidad en puntos críticos como son los centros educativos.

Se ha modelado y realizado la verificación del sistema propuesto en este trabajo utilizando la herramienta VDM++ Tools, evaluando el correcto comportamiento y funcionamiento realizado en un ambiente simulado

## REFERENCES


[1] Sten Agerholm and Peter Gorm Larsen, Modeling and Validating SAFER in VDM-SL, In Fourth NASA Langley Formal Methods Workshop. NASA, November 1997

[2] Wolff, S. (2015). Using Executable VDM++ Models in an Industrial Application-Self-defense System for Fighter Aircraft. Technical Report Electronics and Computer Engineering.

[3] J. Michael Spivey ,The Z Notation: A reference manua , vol. 2, 1992.

[4] Cliff B. Jones, Systematic Sodtware Development Using VDM, Second Edition, June 8 1995

[5] Elizabeth Vidal-Duarte. "Aplicación y validación de especificaciones formales ligeras en el modelo conceptual: reduciendo la ambigüedad e incrementando la conformidad entre los requerimientos y el código", 2012.

[6] Shawn A. Bohner, Denis Gra˘canin1, Michael G. Hinchey and Mohamed Eltoweissy, "Model-based evolution of collaborative agent-based system". Journal of the Brazilian Computer Society, 2007, vol.13, n.4, pp.17-38. ISSN 0104-6500

[7] VDMTools User Manual (VDM++) ver1.1

[8] JoakimLindh. Bluetooth® Low Energy Beacons Application Report SWRA475A–January 2015–Revised October 2016

[9] Jozef Hooman and Marcel Verhoef. Format Semantics of a VDM Extension for Distributed Embedded Systems. Concurrency, Compositionality, and Correctness: Essays in Honor of Willem-Paul de Roever, pp 145.

[10] Elizabeth Vidal Duarte. "Refining light-weight formal specifications validations using black box testing and code coverage analysis: an electrocardiograph application". 14th LACCEI International Multi-Conference for Engineering, Education, and Technology: "Engineering Innovations for Global Sustainability".

[11] M. Siekkinen, M. Hiienkari, J. K. Nurminen and J. Nieminen, "How low energy is bluetooth low energy? Comparative measurements with ZigBee/802.15.4," Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE, Paris, 2012, pp. 232-237

[12] Shota Noguchim, Michitoshi Niiborim, Erjing Zhou and Masaru Kamada." Student attendance management system with bluetooth low energy beacon and android devices". 18th International Conference on Network-Based Information Systems 2015.

[13] G. Kumaresan and J. Gokulnath. "Beacon based vehicle tracking and vehicle monitoring system". nternational Journal of Advanced Research

in Computer and Communication Engineering Vol. 5, Issue 3, March 2016
[14] Cüneyt BAYILMIŞ and Mehmet ÖZDEMİR. "A student attendance system based on beacon and smartphones equipped with bluetooth low energy technology. BİLİŞİM TEKNOLOJİLERİ DERGİSİ, CİLT: 9, SAYI: 3, EYLÜL 2016.
[15] Gaurav Saraswat and Varun Garg. "Beacon controller campus surveillance". 2016 Intl. Conference on Advances in Computing, Communications and Informatics (ICACCI), Sept. 21-24, 2016, Jaipur, India
[16] Arequipa presenta la mayor tasa de accidentes de tránsito, Diario Gestión, http://gestion.pe/economia/arequipa-presenta-mayor-tasa-accidentes-transito-2154183