RACCIS 4(2), pp. 52-56 (2014)

Revista Antioqueña de las
Ciencias Computacionales y la Ingeniería de Software

ISSN: 2248-7441

www.fundacioniai.org/raccis
raccis@fundacioniai.org

The importance of training in formal methods in Software Engineering

La importancia de formar en métodos formales en Ingeniería de Software

John Polansky[1], Merck Sinclair[2]
Symantec U.S. *jpoly(AT)ussymantec.com*[1], *msinr(AT)ussymantec.com*[2]

INFORMACIÓN DEL ARTÍCULO

Tipo
Reflexión

Historia
Recibido: 30-08-2014
Correcciones: 19-11-2014
Aceptado: 04-12-2014

Keywords
Curricula, curriculum development, training methods, software quality

Palabras clave
Planes de estudios, desarrollo curricular, métodos de formación, calidad del software.

ABSTRACT
The paradigm of formal methods provides systematic techniques and rigorous to software develop and, due the crescent complexity and quality requirements of current products, is necessary introduce them in curriculum of software engineer. In this article is analyzed the importance of train in formal methods and described specific techniques to achieved it efficiently. This techniques are the result of an experimental process in the class room of more than fifteen years in undergraduate and graduate programs, the same as company training. Also are presented a proposal a curriculum to systematic introduction of this paradigm and description of a program in training methods that has been success to industry. Results shows that students gain confidence in formal methods just when found out of the benefits of this in the context of software engineer.

RESUMEN
El paradigma de los métodos formales proporciona técnicas sistemáticas y rigurosas para el desarrollo de software y, debido a la creciente complejidad y a los requisitos de calidad de los productos actuales, es necesario que se introduzcan en los planes de estudios de Ingeniería de Software. En este artículo se analiza la importancia de formar en métodos formales y se describen técnicas específicas para lograrlo eficientemente. Estas técnicas son el resultado de un proceso experimental en el aula de más de quince años en programas de pregrado y posgrado, lo mismo que en capacitaciones empresariales. También se presenta una propuesta de plan de estudios para la introducción sistemática de este paradigma y la descripción de un programa de método formativo que ha sido exitoso para la industria. Los resultados demuestran que los estudiantes ganan confianza en los métodos formales sólo cuando se enteran de los beneficios de éstos en el contexto de la Ingeniería de Software.

1. Introducción

A pesar que desde hace más de cincuenta años se vienen realizando esfuerzos para desarrollar teorías, lenguajes, métodos y herramientas de soportes, la práctica de la Ingeniería de Software todavía no ha podido encontrar las instrucciones y los métodos eficaces para salir de la crisis referida en los años 60. El paradigma de los métodos formales se propuso y desarrolló con el objetivo de hacerle frente a este problema mediante técnicas matemáticas como el razonamiento lógico, la especificación formal, el refinamiento, y la verificación. En teoría, actualmente se sabe cómo utilizar notaciones formales para escribir especificaciones, cómo usar cálculos refinados para transformar gradualmente una especificación en una implementación correcta, y cómo aplicar las lógicas de Hoare o de Dijkstra para probar programas, con el mismo grado de rigor que se aplican los teoremas matemáticos. Sin embargo, para los profesionales de las Ciencias Computacionales no es tan simple aplicarlas en los actuales proyectos software. El problema es la complejidad de éstos y la ineficiencia de los métodos formales para tratar con sistemas a gran escala y los frecuentes cambios en los requisitos y en el diseño.

Esto no quiere decir que los métodos formales sean inútiles. De hecho, son más necesarios hoy que nunca antes en la historia de la Ingeniería de Software, porque sus productos son un componente tecnológico fundamental en más y más sistemas para una sociedad software-dependiente [1]. Lo que se debe comprender es que actualmente su papel es diferente al de otras técnicas, debido a que se orientan principalmente a la formación, y su capacidad sólo se puede aplicar a los proyectos de Ingeniería de Software a través de lo que aprenden, dominan y aplican los desarrolladores. La manera de utilizar los métodos formales en la práctica es a través de los métodos de ingeniería formal [2]. Por ejemplo, el método SOFL [3] proporciona un enfoque de tres pasos para construir especificaciones formales para ayudar: 1) al análisis de requisitos y al diseño del sistema, 2) a la inspección basada en la especificación, y 3) a las pruebas

para detectar errores en las especificaciones y los programas. El desarrollo de proyectos software es una actividad humana, cuyo producto se debe entregar en el tiempo acordado y dentro el presupuesto especificado, y que frecuentemente se enfrenta a la inestabilidad de los equipos de desarrollo. En tal situación, la aplicación de métodos formales rara vez es completamente práctica. Pero se podría mejorar la calidad del software si se enseñara a esos equipos a pensar de manera disciplinada y rigurosa a través de una formación lógica en métodos formales.

Con el fin de alentar a más desarrolladores de software a aprender métodos formales, primero hay que motivarlos mediante la demostración de los claros beneficios de aplicarlos en la mejora de las prácticas actuales de la Ingeniería de Software. De hecho, este es un gran reto y es aún más difícil cuando en todo el mundo más y más jóvenes se muestran desinteresados en las matemáticas. Sin embargo, esta parece ser la única forma en que posiblemente se pueda avanzar en la educación en esta área. En este trabajo se describen varias técnicas para formar en métodos formales. La idea fundamental es ponerla en el contexto de la Ingeniería de Software, pero, como señala Parnas [4], los métodos formales no se deben limitar a una sola ingeniería, deben vincularse e integrarse a la ingeniería en general.

2. Técnicas para enseñar métodos formales

A continuación se describen algunas técnicas específicas para la enseñanza de los métodos formales, que han sido probadas en cursos de VDM [5], SOFL [1], y en el refinamiento del cálculo de Morgan [6] en varias universidades y empresas.

2.1 Utilización de ejemplos

Aprender métodos formales es similar a aprender otras teorías o técnicas, por lo que a los estudiantes les gusta comenzar con ejemplos sencillos. Estos ejemplos deben proceder de la vida diaria y deben ser capaces de vincular el problema y la práctica a una solución potencial con estos métodos. Esta forma de enseñanza motiva a los estudiantes y desarrolla sus intereses en esta materia. Por ejemplo, al explicar el problema de la ambigüedad en las especificaciones informales y el hecho de que puede ser resuelto con formalización, se puede utilizar una operación para buscar un número entero en una lista de enteros; y después de explicar la imprecisión de las declaraciones de requisitos informales se presenta una especificación formal, que es a la vez precisa y concisa. Este ejemplo les ayuda a los estudiantes a entender el poder potencial de la formalización.

2.2 Introducción gradual de conceptos importantes

Los conceptos fundamentales son la clave para entender el espíritu de los métodos formales, y son muy eficaces para ayudarles a los estudiantes a entender el principio esencial de éstos cuando se aplican esfuerzos suficientes para enseñarlos. Por ejemplo, cuando se introducen especificaciones formales el objetivo es ilustrar los principios de las pre y las post condiciones. Una manera eficaz de enseñar el concepto es comparándolo con el algoritmo correspondiente, y dejar que los estudiantes entienden la diferencia real y la relación entre una especificación y un algoritmo. La comparación puede hacerse sobre la base de un simple cálculo científico, como en la operación para obtener la raíz cuadrada de un entero. La pre condición de esta operación es $x \geq 0$, y la pos condición es $y^2 = x$, donde x es la entrada y y la salida. Pero el algoritmo correspondiente sería algo como $y = math.sqrt(x)$. Este ejemplo da lugar a un problema en el que la salida y producida por el algoritmo puede no satisfacer la post condición de la operación, porque se obtiene sólo una aproximación a la raíz cuadrada real de algunos números enteros positivos. En esta circunstancia, es útil demostrarles a los estudiantes la importancia de notar esta falta de coherencia entre la especificación y la implementación. Este también es un buen ejemplo para mostrar la necesidad de utilizar o construir teorías adecuadas en el dominio de la aplicación.

Por otra parte, una operación específica utilizando pre y post condiciones define una forma de transformar un estado inicial a un estado final. Con el fin de permitirles a los estudiantes comprender esta idea esencial se deben hacer grandes esfuerzos en la explicación de los conceptos fundamentales, como estado, tipo, y declaración de variables. La explicación se puede dar desde diferentes ángulos, por ejemplo, desde el punto de vista de las matemáticas y del software. Después que los estudiantes comprendan los conceptos básicos se les puede mostrar, con ejemplos sencillos, cómo un algoritmo transforma paso a paso un estado inicial a un estado final a través de sus declaraciones, y cómo esa transformación se puede abstraer en pre y post condiciones. Esta forma de enseñanza les ayuda a los estudiantes a construir una asociación entre los programas y las especificaciones, lo que allana el camino para la enseñanza de técnicas de verificación basada en la especificación, como la verificación formal, la revisión, las pruebas, o sus combinaciones.

2.3 Profundizar en los conceptos básicos

Escribir especificaciones formales suficientemente correctas requiere que el desarrollador tenga un buen conocimiento de las características de los distintos tipos de datos y altas habilidades en la aplicación de operadores bien definidos sobre los tipos de datos, como booleanos, conjuntos, secuencias, y tipos de mapas. Por lo tanto, los estudiantes deben profundizar mediante ejercicios masivos en los conceptos básicos. Una forma eficaz de incorporar ejercicios (en lo posible problemas) en un plan de estudios es dejar que los resuelvan inmediatamente después que se introduce un tipo de datos. Por ejemplo, luego de hacer la introducción de los tipos de conjuntos, deben aprender el significado de los operadores, como unión, intersección, cardinalidad, pertenencia, sub-conjunto, sub-conjunto propio, y así sucesivamente, mientras los aplican a valores de conjuntos específicos. Si el tiempo lo permite, es útil realizar un debate público sobre los resultados que obtuvieron. Una discusión de este tipo puede ayudarles a los estudiantes a ser capaces de averiguar la razón de sus errores y a otros a averiguar la forma correcta de pensar.

Este proceso es similar a la formación básica en deportes. Para ser un excelente jugador de fútbol, por ejemplo, hay que correr rápido y tener un cuerpo fuerte. Para desarrollar estas cualidades el deportista debe pasar mucho tiempo y hacer grandes esfuerzos en su formación básica. Cualquiera que haga caso omiso de esto no podrá desempeñarse satisfactoriamente en un partido. El mismo principio se aplica en la enseñanza y el estudio de las técnicas de refinamiento y de verificación formal. Para dominar el cálculo de refinamiento los estudiantes deben resolver muchos ejercicios (problemas) pequeños aplicando todas las leyes de refinamiento. Para ser hábiles para la verificación formal deben hacer lo mismo en la comprensión del significado de cada axioma y regla de inferencia en las lógicas correspondientes y sus aplicaciones en programas pequeños. El punto importante aquí es lograr que entiendan el principio y las habilidades subyacentes de las técnicas que posiblemente aplican, incluso de manera informal, en la práctica.

2.4 Desarrollar habilidades en abstracción

La utilización eficaz de un método formal requiere que el desarrollador tenga altos conocimientos y habilidades en abstracción, especialmente en el contexto de desarrollo de software. Por tanto, cómo ayudarles a los estudiantes a fortalecer estas habilidades y capacidades se convierte en un tema importante en la formación en métodos formales. Si bien ha sido reconocido ampliamente como algo difícil de enseñar, existen programas y profesores que han ganado suficiente conocimiento y comprensión a través de sus experiencias a lo largo del tiempo. Teniendo en cuenta el hecho de que las operaciones básicas necesarias en un sistema software usualmente incluyen buscar, clasificar, fusionar, adicionar, eliminar, actualizar, calcular, y sus combinaciones, la enseñanza debe hacer énfasis en cómo expresar todas estas funciones utilizando apropiadamente los tipos de datos y sus operadores relacionados. Cada una de ellas podrá formar un patrón de especificación que permanecerá en la mente de los estudiantes y estará disponible para aplicarlo en el desarrollo de software real. Por ejemplo, ¿cuáles son los posibles patrones de especificación para una función que comprueba que una colección de enteros está vacía? Para responder a esta pregunta primero hay que definir formalmente una colección de enteros como un conjunto y una secuencia respectivamente, es decir, *int set: set of int* e *int seq: seq of int*. Posteriormente se discuten los patrones de especificación más utilizados para cada una de esas abstracciones de datos. Para el conjunto de los enteros se pueden utilizar los siguientes patrones para expresar el hecho de que el conjunto está vacío: *int set = fg* y *card(int set) = 0*. Por supuesto, se pueden tener más patrones para expresar el mismo significado, pero serían mucho más complejos y no sería fácil leerlos. Le corresponde al profesor decidir si es necesario discutir un patrón complicado en el tiempo de enseñanza. En el caso de una secuencia de enteros se podrían utilizar los siguientes patrones para expresar el hecho de que está vacía: *int set = []* y *len(int set) = 0*.

Después que cada patrón básico de especificación sea dominado por los estudiantes, entonces se puede ir más allá para explicar cómo aplicarlos en una situación más complicada, como en una operación para buscar un número entero en una colección de enteros. Para explicar cómo se especifica una operación de este tipo se toma el mismo enfoque que el de la enseñanza de los patrones básicos, definiendo en primer lugar la colección de enteros como un conjunto de números enteros y una secuencia de números enteros respectivamente, y entonces explicar cómo especificar la operación para combinar los patrones básicos para cada una de las abstracciones de datos.

2.5 Prácticas a través de pequeños proyectos

Mientras que la formación básica es importante en la enseñanza y el estudio de los métodos formales, no se debe olvidar nunca darles a los estudiantes oportunidades para vincularlos a la Ingeniería de Software. En otras palabras, necesitan que se les enseñe cómo les ayudarán en la práctica del desarrollo de software; de lo contrario, quizás con algunas excepciones, estarán propensos a perder la motivación por su aprendizaje o aplicación. Una de las maneras más efectiva para lograrlo es que los estudiantes lleven a cabo pequeños proyectos. Por ejemplo, después de la introducción a VDM y de ejercicios masivos con los conceptos básicos, se les puede pedir que realicen uno o dos proyectos pequeños. Un podría ser la construcción de una especificación formal de un sistema para una biblioteca, y otra posibilidad es dejar que completen una especificación formal para un software ATM. A través de este tipo de proyectos los estudiantes realmente pueden sentir cómo construir especificaciones formales y organizar proyectos de desarrollo de software reales. Por supuesto, esta práctica también puede darles la oportunidad de encontrar las debilidades del lenguaje de especificación que están utilizando. Por ejemplo, a falta de un mecanismo intuitivo para ordenar un sistema completo de manera estructurada se podría utilizar el lenguaje de especificación formal, mediante el uso de datos intuitivos y diagramas de flujo y descomposiciones de procesos formalizados. De hecho, muchas notaciones formales existentes se centran sólo en un aspecto del problema en Ingeniería de Software e ignoran los otros. Si un método o técnica se limita a ayudar a resolver un problema a la vez que crea más problemas en el contexto de la Ingeniería de Software, es poco probable que sea popular entre los practicantes y menos que la apliquen en proyectos reales. En este sentido, métodos como SOFL han demostrado ser la excepción, ya que proporciona un proceso sistemático y riguroso para la integración de técnicas formales en las prácticas de desarrollo de software existentes, y no crea más problemas.

2.6 Utilizar métodos de ingeniería formal

El objetivo final de la enseñanza de los métodos formales es crear la posibilidad de que los estudiantes los apliquen en la práctica. Los métodos de ingeniería formal indican cómo se pueden aplicar los métodos formales en proyectos reales. Uno de los aspectos más importantes de estos métodos es el énfasis en la combinación de diagramas, notación formal, y lenguaje natural de una manera coherente y sistemática para escribir

especificaciones [2]. El propósito de esto es ayudarles a los desarrolladores a entender fácilmente las especificaciones que están escribiendo y las escritas por otros. La visualización es intuitiva y adecuada para describir la idea general y la arquitectura del sistema; la notación formal tiene fuerza para lograr la precisión de las declaraciones en las especificaciones; y el lenguaje natural se puede utilizar para proporcionar una interpretación amigable de las expresiones formales. En general, los métodos de ingeniería formal difieren de los métodos formales en que éstos intenta responder a la pregunta ¿qué se debe hacer y por qué? en el desarrollo de software, mientras que los primeros intenta responder a ¿qué se puede hacer y cómo? Para lograrlo, se centran en técnicas y métodos para integrar los métodos formales en todo el proceso de desarrollo de software, de manera que la fuerza de los mismos se pueda utilizar en la práctica y evitar su debilidad de ser complejos. Los métodos de ingeniería formal muestran cómo los sistemas software, incluyendo todos los documentos de nivel, son creados en realidad y expresados formalmente, no sólo una simple mezcla de notaciones formales con imágenes. Como una introducción detallada a estos métodos está más allá del alcance de este artículo, se refiere al lector al libro SOFL [2] para una descripción más completa.

De hecho, el mismo principio de los métodos de ingeniería formal también se puede aplicar de manera efectiva a la enseñanza de los métodos formales, puesto que la enseñanza es en realidad una especie de proyecto software cuyo producto es formar a los estudiantes. Por ejemplo, al explicar una expresión matemática como *Z = (X U Y)*, se puede utilizar una representación gráfica como los diagramas de Venn para ilustrar la operación de unión, y al mismo tiempo realizar una descripción en lenguaje natural para explicar el significado de la misma. Al introducir una operación en VDM el proceso se puede describir como en el lenguaje SOFL para mostrar la entrada, la salida, y las variables externas, pero los detalles de la función de la operación se definen mediante pre y post-condiciones. Con explicaciones informales en lenguaje natural el significado de toda la especificación de la operación puede ser fácilmente comprendida por los estudiantes.

2.7 Herramientas de soporte

Es muy común que en los cursos de programación se experimente con herramientas en la enseñanza de lenguajes, y se ha demostrado que son eficaces para ayudarles a los estudiantes a que escriban, ejecuten, y prueben sus programas. Muchos de los profesores de métodos formales aplican esta estrategia de enseñanza en sus cursos. Sin embargo, la experiencia en enseñar lenguajes como VDM o Z centrados en técnicas de especificación formal, sugieren que el uso de herramientas no es necesariamente efectiva; tal vez menos eficaz que no usar ninguna. Hay dos razones para ello: 1) el aprendizaje de los métodos formales requiere que los estudiantes aprendan tanto la sintaxis como la semántica del lenguaje de especificación relacionado. La manera más efectiva para que lo recuerden es escribir especificaciones formales a mano, a medida que aprenden el inglés como lengua extranjera. Es factible, porque los ejercicios asignados en las clases son de pequeña escala. También es eficaz para memorizar la sintaxis y para profundizar en la comprensión de las técnicas de abstracción, porque los estudiantes no tendrían oportunidad de *copiar y pegar* sin pensar por sí mismos, como se hace a menudo en un computador. 2) El propósito de escribir una especificación no es para que la ejecute directamente el computador, sino para que las personas la lean y la comprendan. Por lo tanto, escribir a mano con buen estilo las especificaciones formales es mucho más útil para el aprendizaje que mediante el uso de una herramienta para mejorar automáticamente el estilo y el formato. En el caso de la programación, sin una herramienta como un compilador, no se podría ejecutar el programa; pero al escribir una especificación no hay necesidad de ejecutarla, así que una herramienta de soporte no creará ninguna ventaja significativa. En cambio, para algunos estudiantes que no quieren aprender métodos formales la herramienta creará oportunidades para que *copien y peguen* sin pensar.

Esto no significa que las herramientas de apoyo no sean necesarias para usar los métodos formales en la práctica. Por el contrario, son cruciales para mejorar la productividad y reducir las posibilidades de errores. Por esta razón es conveniente dejar que los estudiantes utilicen una herramienta de apoyo, como IFAD VDMTools [7] o el editor SOFL GUI, cuando llevan a cabo un proyecto pequeño, pero sólo luego de un aprendizaje sistemático de técnicas de especificación formal en sus cursos. De esta manera también se les muestra que las herramientas ofrecen una alta automatización tanto para la escritura de las especificaciones como para su análisis. Así es posible sopesar lo difícil del aprendizaje de los métodos formales a mano. Esto es similar a la situación en la que una persona se siente feliz cuando tiene la oportunidad de comer deliciosa comida después de un largo tiempo de hambre.

2.8 Limitaciones de tiempo

Generalmente, los conceptos y expresiones matemáticas requieren que los estudiantes tomen tiempo para digerirlos, por lo que la enseñanza debe tomar un ritmo lento con muchos ejemplos. Sin embargo, un curso es como un proyecto software porque también tiene limitaciones de tiempo. A menudo, los profesores se enfrentan con el dilema de querer enseñar más contenidos importantes en el estudio de los métodos formales, pero no cuentan con el suficiente tiempo para lograrlo. Para hacer frente a este problema la experiencia sugiere que los cursos no deben ser demasiado ambiciosos, sino que se deben focalizar. Por ejemplo, se puede enseñar la especificación formal, el refinamiento, y la verificación formal en tres cursos diferentes; y sería efectivo enfocar la enseñanza en cada uno de ellos en las partes más fundamentales, pero importantes, y darles a los estudiantes el tiempo suficiente para que puedan aplicar las técnicas aprendidas. Se podrían enseñar las técnicas para redactar especificaciones formales, utilizando pre y post-condiciones, mediante un enfoque

entrelazado, es decir, enseñar conceptos y luego pedirles a los estudiantes que los practiquen. Después pedirles que lleven a cabo un pequeño proyecto en el que deban aplicar todo el conocimiento aprendido. De esta manera se les proporcionan muchas oportunidades para aprender cómo aplicar los resultados teóricos de manera efectiva en la práctica.

3. Conclusiones

La formación en métodos formal es importante pero no necesariamente significa que sea popular entre los estudiantes. De acuerdo con diversas experiencias en la enseñanza de lenguajes de especificación, las personas con cierta experiencia en ellos por lo general encuentran a los métodos formales, especialmente a las técnicas de especificación formal, fáciles de aprender y de usar, pero no es lo mismo pata los estudiantes sin experiencia laboral. Las razones importantes son que los estudiantes no suelen entender profundamente la importancia del papel de los métodos formales en el aseguramiento de la calidad del software y sus contenidos son bastante complejos y detallados [8]. Dado que la educación en estos métodos es necesaria, una posible solución a este problema es organizarlos como cursos obligatorios en lugar de asignaturas optativas. De este modo, cada estudiante estaría obligado a aprender métodos formales. Además, mediante la aplicación de métodos eficaces de enseñanza, como los mencionados anteriormente, y de requisitos apropiados para estudiantes de diferentes niveles, sería posible dejar que cada vez más aprendan de este tema. Sin embargo, aunque esta posibilidad se haga realidad en la actualidad o en el futuro no garantiza que los métodos formales sean atractivos para los estudiantes. La experiencia sugiere que para hacerlos llamativos se debe lograr un buen equilibrio entre simplicidad, visualización, y precisión, a la vez que demostrar sus beneficios para garantizar la calidad del software y para reducir el costo de los proyectos. También es conveniente proporcionarles diversión, como la infografía o la animación, pero por desgracia, pocos se han cumplido estos criterios y es difícil imaginar que cualquier método de enseñanza mejorará significativamente esta situación.

Dado que el desarrollo de software necesita razonamiento lógico es posible pensar que no importa si los métodos formales son atractivos o no, la educación en ellos debe continuar en las universidades y es de esperar que en la industria también. Sólo una adecuada educación puede lograr que la aplicación de los métodos formales, directa o indirectamente, sea posible en la Ingeniería de Software. Pero para lograrlo los estudiantes, los profesionales, la industria, y la universidad deben comprometerse con una formación continua en estos temas, porque sólo la práctica permanente hará que se comprendan y apliquen en la medida que se necesitan.

La educación es el camino necesario y más eficaz para transferir los métodos formales a la industria del software. El más importante factor de influencia para este éxito es que los planes de estudios los pongan en el contexto de la Ingeniería de Software. En este trabajo se han descrito varias técnicas para enseñar métodos formales a estudiantes con y sin experiencia, cada uno de las cuales ha sido probada en la práctica. No importa si los métodos formales se puedan utilizar como una técnica efectiva de Ingeniería de Software, pero sin duda una adecuada educación beneficiará su práctica. La única manera de transferir eficazmente los métodos formales a la industria es educación, educación, y educación.

Referencias

[1] Serna, M.E. (2010). Ontological approach to maintenance of software. Revista Facultad de Ingeniería Universidad de Antioquia 55, pp. 184-19.

[2] Liu, S. (2004). Formal engineering for industrial software development using the SOFL method. Springer-Verlag.

[3] Liu, S. et al (1998). SOFL: A formal engineering methodology for industrial applications. IEEE Transactions on Software Engineering 24(1), pp. 24-45.

[4] Parnas, D. (1990). Education for computing professionals. Computer 23(1), pp.17-22.

[5] Jones, C. (1990). Systematic software development using VDM. Prentice-Hall.

[6] Morgan, C. (1994). Programming from specifications. Prentice-Hall.

[7] The VDM-SL Tool Group (1994). User Manual for the IFAD VDM-SL Toolbox. Technical Report for IFAD-VDM-4. The Institute of Applied Computer Science.

[8] Serna, M.E. (2010). Formal Methods and Software Engineering. Revista Virtual Universidad Católica del Norte 30, pp. 158-184.