NIVEL DE DESARROLLO DE LA VERIFICACIÓN FORMAL

Efilia Ariza Vargas

Elizabeth Ariza Vargas

FACULTAD DE INGENIERÍAS

Ingeniería de Sistemas

Director
Prof. Edgar Serna M.

INSTITUTO TECNOLÓGICO METROPOLITANO

Febrero 2016

RESUMEN

Los métodos formales se utilizan mayoritariamente para especificar y validar productos de software crítico, y poco a poco comienzan a penetrar la Ingeniería del Software tradicional para el desarrollo de aplicaciones de todo tipo. Uno de sus inconvenientes es la falta de conocimiento matemático que tienen los profesionales para aplicarlos en sus procesos. En este proyecto se busca determinar el nivel de desarrollo de la especificación formal hasta el momento, con el objetivo de difundir los resultados a la comunidad académica relacionada. Se hará una búsqueda amplia en la literatura para determinar dicho nivel y luego difundir los resultados para que sean tenidos en cuenta como elemento necesario en los planes de estudio de la Ingeniería del Software.

Palabras clave: Métodos formales, Ingeniería del Software, Ingeniería de Requisitos, Verificación y Validación, formalización.

TABLA DE CONTENIDO

1. INTRODUCCIÓN

Desde hace décadas, los métodos formales han interesado en los ambientes académicos y de investigación. Estos métodos, de base matemática para la especificación y el posterior desarrollo de sistemas software, proporcionan un mejor control de la complejidad de los sistemas actuales, ofreciendo facilidades para la abstracción, y hacen menos ambigua y más uniforme la especificación de requisitos y a menor costo (Serna y Serna, 2014). Sin embargo, su aplicación en la industria no ha tenido el nivel de aceptación y utilización que esperaban los pioneros del siglo pasado. Aunque se aplican en proyectos piloto y se conocen algunos casos aislados de grandes esfuerzos, en los entornos industriales su aplicación es hasta el momento limitada. Pero, a pesar de esta aparente falta de interés por aplicar los métodos formales industrialmente, definitivamente hay una necesidad de ellos para llevar a cabo la especificación formal (Almeida et al., 2011), especialmente en sistemas críticos. El problema de la transferencia de tecnología, que a menudo se presenta como una de las principales razones de la tensión entre la investigación y la industria, no es un problema exclusivo de los métodos formales. Las nuevas tecnologías, desarrolladas en otros campos de las Ciencias Computacionales, también tienden a ser adoptadas muy lentamente en la industria, pero en el caso de los métodos formales, la situación parece ser aún peor (Serna, 2010).

El problema central de los métodos formales es que deben ser capaces de garantizar, siguiendo un enfoque rigurosamente matemático, el comportamiento de un sistema dado y en el corazón de ellos se encuentra la noción de la especificación. Una especificación es un modelo de un sistema que contiene una descripción de su comportamiento deseado, es decir, *qué* es lo que se va a implementar, y no *cómo* se va a hacer. Dicha especificación puede ser totalmente abstracta, en cuyo caso el modelo *es* la descripción del comportamiento, o puede ser más operacional, y entonces la descripción, de alguna manera, está contenida o implícita en el modelo.

Por otro lado, el creciente uso de software en las actividades de esta sociedad ha exigido normas y métodos estrictos de desarrollo. Debido a esto, en los últimos años y a través

de la investigación y la innovación, se ha propuesto una serie de técnicas y herramientas orientadas en gran medida a mejorar los procesos de especificación, de diseño y de prueba de este producto. Todo el proceso, junto con las métricas tradicionales de calidad (Metaute y Serna, 2015), hace que el desarrollo de software evolucione de una condición de arte negro a una disciplina de ingeniería respetable y en rápido progreso, pero que no madurará sino hasta dentro de varios años (Serna, 2013). Tradicionalmente, la calidad ha sido definida como la conformidad con la especificación (cuando no se cumple, entonces surgen los fallos). Llegar a esto en el caso del software a menudo es muy complicado, porque de esta definición se desprende que un buen producto requiere no solamente que esta norma se cumpla, sino que es necesario contar con una buena especificación. Esto significa que todos los requisitos tienen que ser especificados plenamente y sin ambigüedades, porque, estrictamente hablando, los fallos no pueden ser reconocidos por funciones que no han sido especificadas previamente.

En este sentido, el principal problema con el software es la falta de capacidad para prever y definir la cantidad y complejidad de los requisitos. Esto incluye las complicaciones que provienen de la combinación de los modos operativos y las condiciones de entrada y de los entornos. Además, debido a que cada vez los sistemas se hacen más grandes y más complejos, se incrementan los problemas en la Ingeniería de Requisitos y a menudo la entrega del producto se retrasa, porque la especificación de requisitos es la base para las demás fases del ciclo de vida del producto. Aquí es donde cobra importancia la especificación formal, porque es una alternativa para unificar los criterios, comprensiones y puntos de vista del equipo de trabajo alrededor de los requisitos del sistema. Pero, si se aplica la formalización en el proceso, se podría garantizar de mejor forma que el producto cumple lo especificado, y por tanto, también podría mejorar su calidad.

Es así, que los métodos modernos (formales) implican un enfoque más matemático para el desarrollo de la Ingeniería del Software y, aunque las pruebas matemáticas de funcionamiento y de validación de la semántica del código no pueden ser perfectas, al menos ofrecen un método más formal de garantía de calidad que el de los métodos anteriores. Para lograrlo, las áreas clave a tener en cuenta son: 1) la especificación

formal, es decir, el uso de métodos basados matemáticamente e incluso automatizados para traducir los requisitos en el diseño; 2) las metodologías de diseño, o librerías de paquetes software reutilizables, con lo que las pruebas de funcionamiento y de validación de código pueden ser en gran medida automatizadas; y 3) las pruebas, en las que las especificaciones de prueba pueden ser generadas automáticamente por las herramientas de diseño.

En este artículo se presentan los resultados de una investigación cuyo objetivo es determinar el nivel actual del desarrollo de la especificación formal. El contenido se estructura de la siguiente forma: en la primera parte se detallan los trabajos relacionados, en la segunda se describe la metodología aplicada, en la tercera se presentan los resultados, los cuales se analizan y discuten en la cuarta sección. Finalmente, en la quinta parte se concluye el trabajo realizado.

2. MARCO TEÓRICO

Desde el surgimiento de las Ciencias Computacionales los investigadores han considerado a la Especificación Formal como una de sus áreas de interés. Finalizando los años cuarenta, Turing (Randell, 1973) observó que el razonamiento acerca de programas secuenciales era más sencillo cuando se hacían anotaciones acerca de las propiedades de su estado en puntos específicos del mismo. En los años sesenta, Floyd (1967), Hoare (1969) y Naur (1969) propusieron técnicas axiomáticas para demostrar la consistencia entre los programas secuenciales y esas propiedades, a las que llamaron especificaciones, y Dijkstra (1975) demostró cómo utilizar constructivamente el cálculo para derivar programas no-determinísticos que las cumplieran.

Por su parte, Parnas (1972) y Liskov (1975) propusieron técnicas específicas para expresar formalmente las propiedades de los programas, particularmente para datos estructurados, y Pnueli (1977) lo hizo para programas concurrentes. Estos aportes conformaron el punto de partida para la Especificación Formal como una nueva área de investigación (Parnas, 1979; Gerhart y Wile, 1979; Abrial, 1980; Heninger, 1980), y desde entonces se ha incrementado continuamente el interés que despertó, lo mismo que los múltiples usos en la Ingeniería del Software (Wing, 1990; *Craigen, Gerhart y Ralston, 1993; Hinchey, y Bowen, 1995; Clarke y Wing, 1996; Wing, Woodcock y Davies, 1999; Cuellar, 2000)*. Hasta el momento el objetivo principal de esta comunidad de investigadores se focaliza en la especificación escrita durante el diseño del modelo funcional preliminar (Wing, 1990), por lo que lo que su investigación se centra principalmente en evaluar y comparar las herramientas relacionadas con estas especificación y en analizar sus fortalezas y debilidades.

El término Especificación Formal se puede referir a diversos aspectos en el ciclo de vida del software, y se utiliza indistintamente para un producto como para su correspondiente proceso, por lo que está sobrecargado en la literatura. Generalmente, se acepta que una especificación formal es la expresión, en algún lenguaje formal y con cierto nivel de abstracción, de una serie de características que el sistema debe satisfacer (Serna y Morales, 2014). Esta definición se utiliza dependiendo de lo que se comprenda

como sistema, del tipo de características a satisfacer, del nivel de abstracción aplicado y del lenguaje formal utilizado (Serna, 2010). El sistema puede ser un modelo descriptivo del dominio de las necesidades, del software y su entorno, del software como tal, de la interfaz de usuario, de la arquitectura del software o de algún proceso a seguir, entre otros. Las características se pueden referir a los objetivos de alto nivel, a los requisitos no funcionales, a la hipótesis del dominio, o a los protocolos de interacción entre esos componentes. Pero para asegurar que una solución resuelve correctamente un problema más allá de las diferentes concepciones de especificación, existe una idea común relacionada con el dominio del mismo: establecer el primer estado en el que éste es correcto (*Akhtar, Guyadec y Oquendo, 2012*). Sin embargo, esta dicotomía es simplista porque generalmente una solución se puede representar como un conjunto de sub-problemas, los cuales se especifican y resuelven a la vez (*Swartout y Balzer, 1982)*; por lo tanto, una especificación debe cumplir con alguna característica de alto nivel y también satisfacer algunas de bajo nivel.

La formalización es un proceso necesario para diseñar, validar, documentar, comunicar, hacer re-ingeniería y reutilizar soluciones informáticas (*Bacelar et al., 2011*). Además, permite obtener especificaciones con un mayor nivel de calidad y proporciona las bases para su automatización. Otras aplicaciones detalladas en los trabajos las describen para plantear preguntas y para detectar problemas serios en la formulación informal, y la semántica del formalismo se utiliza para establecer normas precisas de interpretación que permitan superar varios de los problemas del lenguaje natural.

3. METODOLOGÍA

El procedimiento para esta investigación se basó en una metodología para realizar revisiones fiables de la literatura (Serna y Serna, 2014a), y se orientó a identificar estudios y opiniones que pudieran ser candidatos a incluir en la muestra final. Las fuentes fueron las bases de datos ACM digital library, IEEE Xplore digital library, Sciendirect, Springer y Web of Science. Los parámetros de búsqueda incluyeron las palabras clave: *formal specification, formal methods*, *Requirments Engineering* y *requirements elicitation*. Las combinaciones debían aparecer en el título, el resumen o en el contenido del documento. Para lograr mayor cubrimiento, no se determinó una línea de tiempo específica y no se excluyó ninguna fuente inicial. Además, en los estudios primarios seleccionados se validó la solidez de la metodología aplicada y del análisis a los resultados presentados. Las opiniones y/o análisis debían ser presentadas por instituciones o investigadores que estuvieran relacionados con el área temática y que su trayectoria fortaleciera el postulado.

En la muestra final se incluyeron directamente: 1) los artículos de revistas, por haber pasado por procesos de selección y evaluación estructurados, 2) las presentaciones en eventos, cuyos autores y/o empresas fueran idóneos en el área, y 3) los reportes técnicos, cuyos autores estuvieran relacionados con el trabajo en especificación formal y/o tuvieran experiencia en los campos relacionados. Al final, la muestra inicial quedó conformada por 185 trabajos. Como criterio de calidad, debían hacer un aporte relevante a la temática de investigación. Para cumplirlo, se aplicaron los siguientes procesos: 1) identificar los estudios relevantes, 2) excluir estudios con base en el título, 3) excluir estudios a base en los resúmenes, y 4) analizar los estudios y seleccionar los más relevantes para la temática en función del texto completo. Los criterios de inclusión y exclusión más importantes fueron: formalidad y pertinencia de la publicación, autoridad del autor, calidad y aportes del contenido, fuentes de datos, sustentación de la tesis, calidad de la investigación y coherencia de los resultados. Luego de este proceso se conformó la muestra final para la investigación, cuyo análisis y resultados se presentan a continuación.

CRONOGRAMA DE ACTIVIDADES

Actividades	Meses					
	1	2	3	4	5	6
1. Estructurar el protocolo de búsqueda	X					
2. Definir criterios para evaluar el rigor de los resultados publicados	X					
3. Consultar y analizar el nivel de desarrollo y de madurez	X	X	X			
4. Estructurar y presentar los resultados			X	X	X	X

4. RESULTADOS Y DISCUSIÓN

Desde el surgimiento de las Ciencias Computacionales, los investigadores han considerado a la especificación formal como una de sus áreas de interés. Finalizando los años cuarenta, Turing (Randell, 1973) observó que el razonamiento acerca de un programa secuencial era más sencillo cuando, en puntos específicos del mismo, se hacían anotaciones acerca de las propiedades de su estado. En los años sesenta, Floyd (1967), Hoare (1969) y Naur (1969) propusieron técnicas axiomáticas para demostrar la consistencia entre los programas secuenciales y esas propiedades, a las que llamaron *especificaciones*, y Dijkstra (1975) demostró cómo utilizar constructivamente el cálculo para derivar programas no-determinísticos que las cumplieran. Parnas (1972) y Liskov y Zilles (1975) propusieron técnicas específicas para expresar formalmente las propiedades de los programas, particularmente para datos estructurados, y Pnueli (1977) lo hizo para programas concurrentes. Estos aportes conformaron el punto de partida para la *especificación formal* como área de investigación y desarrollo (Parnas, 1977; Abrial, 1980; Heninger, 1980). Desde entonces se ha incrementado continuamente el interés que despertó, lo mismo que los usos en la Ingeniería del Software (Wing, 1990; Hinchey y Bowen, 1995; Clarke y Wing, 1996), aunque su desarrollo y aplicación han caminado por senderos diferentes.

De acuerdo con Ryan (2010), la especificación formal encaja en la etapa de diseño del sistema y su objetivo es estructurar una definición formal de los requisitos del mismo, eliminando la ambigüedad y la incertidumbre del documento de diseño e implementación. De esta manera se les facilita a los desarrolladores predecir el comportamiento y las propiedades del sistema, antes que se desarrolle (Bjørner, 2006). Los resultados de esta especificación son equivalentes al efecto que tuvieron las matemáticas y la geometría en la arquitectura de las civilizaciones tempranas, eliminando las conjeturas y el ensayo-error. Cuando se configura una especificación del sistema, solamente se modela el comportamiento del mismo. Esto significa que, en gran medida, la especificación puede ignorar los detalles que hacen parte de las decisiones de implementación, tales como el lenguaje de programación y los requisitos de recursos

del sistema. Esto permite libertad para centrarse exclusivamente en el comportamiento del sistema, lo que resulta en un nivel adicional de abstracción por encima del código del programa. De esta manera, el lugar de la especificación formal es entre el diseño informal del sistema y la implementación del código (Ryan, 2010).

Para Serna y Serna (2014), una especificación formal es una especificación expresada en un lenguaje cuyo vocabulario, sintaxis y semántica se definen formalmente, y tiene una base lógica matemática usualmente formal. Es un área de investigación activa en la Ingeniería del Software y se aplica en diversas configuraciones y técnicas. Aunque su uso industrial todavía es limitado, aunque actualmente la comunidad científica tiene una comprensión diferente acerca de su utilidad y necesidad. Hasta el momento, el trabajo de los investigadores se focaliza en la especificación escrita durante el diseño del modelo funcional preliminar, por lo que se centran principalmente en evaluar las herramientas relacionadas.

La creciente importancia de los métodos de especificación formal se refleja en el número de publicaciones que se ocupan de ellos y en las investigaciones que defienden su uso en los procesos de desarrollo de software (Bollin, 2008). De acuerdo con este autor, entre las justificaciones sugeridas se incluyen: que el uso de la especificación formal mejora la visión y comprensión de los requisitos, ayuda a clarificar los requisitos al revelar o evitar contradicciones y ambigüedades en la especificación, permite una verificación rigurosa de las especificaciones y aplicaciones, facilita la transición de la especificación y el diseño a la implementación y se espera que ayuden a mejorar la calidad y fiabilidad del software, reduciendo así los costos del ciclo de vida.

Por su parte, van der Poll y Kotzé (2002) señalan que los métodos de especificación formal no han sido ampliamente aceptados en la industria del software, y que una de las principales razones es que esta especificación se ha ocupado sobre todo de los lenguajes, no del proceso de elicitación. Sin embargo, se ha propuesto una serie de estrategias para su incorporación masiva en el proceso de desarrollo de software. Con la proliferación de estas estrategias se ha llegado a una etapa en su desarrollo donde surge la necesidad de organizarlas y clasificarlas. Este es un paso necesario para la comprensión de su posible utilidad y para identificar similitudes y diferencias entre ellas, y para evaluar su aplicabilidad en diferentes contextos académicos e industriales.

Sin embargo, para la mayoría de profesionales no son fáciles de utilizar en el desarrollo de proyectos reales, lo que todavía resulta ser un desafío para el trabajo con los métodos formales que la academia debe comenzar a enfrentar.

Las especificaciones formales no son documentos que se escriban una vez y, generalmente, no se logran justo al comienzo del proceso de desarrollo (Bollin, 2011). Se necesita tiempo para crear una primera versión de utilidad para, después de mucho esfuerzo y revisiones, poder desarrollar una especificación que esté cerca de los conceptos que tienen en sus mentes los actores del proceso. Como argumenta Bollin (2013), la facilidad de comprensión es una cuestión clave (entre otros atributos de calidad), cuando se trabaja en y con una especificación formal. Para van der Poll y Kotzé (2002) una especificación, construida de acuerdo con los criterios establecidos para el diseño, podría ser más comprensible para los usuarios y más fácil de mejorar, lo que facilitaría el mantenimiento a largo plazo. El hecho es que, cuando se fomenta la integración entre los métodos formales y las comunidades de Ingeniería del Software, parece ser necesario dedicar mayor atención a este atributo de calidad. El supuesto es que, al aumentar la comprensión, también es muy probable que se aumente la aceptabilidad. Esto, a su vez, también facilitaría el uso y la enseñanza de los métodos formales (Garlan, 1994).

A través de los años muchos investigadores de los métodos formales han estado de acuerdo en que la especificación formal no se utiliza tanto como debería (Meyer, 1985; Neumann, 1989; Wing, 1990). En parte, porque la mayoría tiene formación en diseño de lenguajes y en construcción de sistemas operativos, o han incursionado vagamente en el diseño y producción de lenguajes de programación (Holt y Cordy, 1988). Pero, como investigadores, siempre han estado atraídos por los métodos formales. Sin embargo, y por razones pragmáticas, no han sido capaces de aplicarlos a la Ingeniería del Software, concretamente a la especificación formal. Penny, Holt y Godfrey (2005) presentan sus puntos de vista acerca de por qué pasa esto y concluyen que se necesita mayor énfasis en el tema de los métodos formales en los planes de estudio de las Ciencias Computacionales.

En la literatura, el término métodos formales se refiere a diversas técnicas matemáticas utilizadas en la Ingeniería del Software para la especificación y el desarrollo formal.

Consisten de lenguajes de especificación o notaciones formales, y emplean una colección de herramientas para apoyar la comprobación de la sintaxis de la especificación y la prueba de sus propiedades (Baber, 2011). Utilizan ampliamente la abstracción, lo que les permite responder preguntas acerca de qué hace el sistema, independientemente de la aplicación. Por otra parte, la naturaleza no-ambigua de la notación matemática evita el problema de la especulación sobre el significado de las frases en lenguaje natural y lo impreciso de la redacción (Knuth, 2011). El lenguaje natural es inherentemente ambiguo, mientras que los métodos formales emplean notaciones precisas de acuerdo con las normas de inferencia (O'Regan, 2006).

Para O'Regan (2012), la especificación formal se convierte en el punto clave de referencia para las diferentes partes implicadas en el desarrollo del sistema. Se puede utilizar como punto de referencia en la documentación de los requisitos, la ejecución del programa y el plan de pruebas. La especificación formal es un medio valioso para promover un entendimiento común para todos los interesados en el sistema. El término métodos formales se utiliza para describir un lenguaje de especificación y un método formal, para el diseño e implementación de sistemas informáticos.

Por su parte, Spivey (2002) sostiene que la especificación formal tienen tres virtudes: es concisa, es precisa y no tiene ambigüedades. Es concisa porque es capaz de expresar en poco espacio hechos complejos sobre los Sistemas de Información. La experiencia práctica demuestra que una especificación matemática de un sistema, es mucho más corta que una especificación informal equivalente. Es precisa porque permite que los requisitos se documenten con claridad. La función deseada de un sistema se describe de manera que no limite indebidamente las estructuras de datos utilizadas para representar la información en el sistema, o los algoritmos utilizados para calcular con ella. No tiene ambigüedades porque cuando se realiza en un lenguaje estandarizado, evita las diversas interpretaciones y tiene un significado único.

ANÁLISIS Y DISCUSIÓN

Para los autores analizados, si el objetivo es producir software de buena calidad, existe una motivación muy fuerte para utilizar las mejores prácticas en la Ingeniería del Software. Los defectos en el software pueden causar, en el mejor de los casos,

problemas menores, pero también generar contratiempos importantes en los sistemas, incluyendo pérdida de vidas o catástrofes económicas. En consecuencia, las empresas deben emplear las mejores prácticas para desarrollar software de alta calidad, y la especificación formal es una tecnología de vanguardia que puede ser de beneficio para lograrlo, debido a que se podría reducir la aparición de defectos en los productos.

Además, una de las ideas clave de la especificación formal, detectada en esta investigación, es abstraerse de lo particular de la implementación y escribir los detalles del sistema de manera compacta. Sin embargo, esta densidad de expresión de los pensamientos, combinada con la complejidad y el tamaño de los modismos matemáticos, también se convierte en una dificultad para alcanzar la comprensión de las especificaciones en las fases posteriores del ciclo de vida. Como sugieren algunos autores (Diller, 1999), la complejidad de tamaño (que está estrechamente relacionada con el problema de la complejidad inherente) puede abordarse mediante técnicas de descomposición y estructuración, provistas por los lenguajes de especificación. Pero, para la comprensión de la enorme cantidad de texto (y un gran número de dependencias entre los diferentes términos en la especificación informal) se requiere la ayuda de algunas herramientas (Bollin, 2008).

De acuerdo con los resultados obtenidos, aparte del nivel estructural, los lenguajes de especificación no ofrecen mucho apoyo. En algunos de los trabajos analizados se encuentran ejemplos o pautas que, directa o indirectamente, sugieren cómo escribir las descripciones de los requisitos en una especificación formal. Sin embargo, no es posible conocer todavía si las diferentes variantes para hacerlo, solamente son una cuestión de estilo o tienen verdadero impacto en la aparición o no de los errores, en el tiempo de comprensión o en la facilidad de comprensión de los requisitos. Debido a esto, muchos autores se orientan a intentar clarificar la cuestión de qué es una especificación comprensible y qué la hace diferente de una buena especificación (formal o informal). Pero, en términos generales, los aportes no logran convencer completamente y muchos se limitan a tratar de presentar directrices para apoyar la comprensión de los términos usados en la especificación. Esto no puede ser suficiente para garantizar una buena especificación formal, porque debe ser sintáctica y semánticamente correcta, de tal manera que permita hacerle seguimiento a todos los conceptos, sin perder la conexión

con el modelo mental del sistema especificado. Este proceso de mapeo no debe ser percibido como agotador y debe ser completado dentro de un tiempo razonable. Pero esta cuestión no se analiza en los trabajos de la muestra final de esta investigación, y no se perciben aportes que extiendan la percepción común de lo que es una buena especificación formal, es decir: completa, consistente y adecuada. Sin embargo, hay que tener en cuenta que la facilidad de comprensión no es solamente algo que se debe tener en cuanta porque sí, sino que es un requisito esencial para decidir acerca de la corrección semántica de la misma especificación.

Por eso es que para lograr los objetivos de esta investigación, se debe suponer que las especificaciones formales descritas en los trabajos utilizan los tipos correctos y, por tanto, son por lo menos sintáctica y semánticamente correctas (en el sentido de que reflejan el sistema a fondo y en el nivel más adecuado de abstracción). Entonces, para determinar el nivel actual del desarrollo de la especificación formal, solamente queda el aspecto sustancial de analizar si el proceso en sí es de calidad. Pero esto tampoco es fácil de deducir en la revisión, porque generalmente los autores confunden formalidad con precisión, y aunque lo primero implica lo segundo la relación contraria no es cierta. Una especificación es formal si se expresa en un lenguaje que posee tres componentes: 1) reglas para determinar gramaticalmente la formación correcta de oraciones (sintaxis), 2) reglas para interpretar las frases de manera precisa y significativa dentro del dominio considerado (semántica), y 3) reglas para inferir información útil a partir de la especificación (teoría de la prueba). Este último componente proporciona las bases para el análisis automatizado de la especificación.

¿Entonces, cómo determinar el nivel actual del desarrollo de la especificación formal? Para encontrar una respuesta, en esta investigación se formularon interrogantes que debían responder los resultados de la revisión de la literatura:

1. ¿Qué es una buena especificación? De acuerdo con los autores analizados, escribir una especificación correcta es muy difícil (probablemente tanto como escribir un programa correcto). En términos generales, se descubre que para ellos una especificación debe ser: 1) adecuada: expresar adecuadamente el problema en cuestión; 2) coherente internamente: tener una interpretación semántica significativa que haga verdaderas las propiedades de todos los especificados en su

conjunto; 3) no-ambigua: no puede tener múltiples interpretaciones de lo que es cierto; 4) completa con respecto al más alto nivel descriptivo: la colección de propiedades especificadas debe ser suficiente para establecer el más alto de ellos, y a la vez debe ser satisfecha por los niveles inferiores; y 5) mínima: no indicar propiedades irrelevantes para el problema o que solamente sean relevantes para una solución del mismo.

2. ¿Qué es una especificación formal? Puede referirse a cosas diferentes en el ciclo de vida del software, por lo que los aportes que la intentan definir están sobrecargados. Una fuente adicional de confusión es que una sola palabra se utiliza indistintamente para un producto y para el proceso correspondiente. En términos generales, los autores afirman que una especificación formal es la expresión, en un lenguaje formal y con cierto nivel de abstracción, de una colección de propiedades que debe satisfacer un sistema. Pero esta definición, resumen de propósito general, cubre diferentes nociones que dependen de lo que debe comprenderse como *sistema*, qué tipo de propiedades son de interés, qué nivel de abstracción se considera y qué tipo de lenguaje formal se utiliza.
3. ¿Por qué especificar formalmente? Del análisis a las lecturas se concluye que especificar un problema es esencial para el diseño, la validación, la documentación, la comunicación, la reingeniería y la reutilización de las soluciones. Pero en muy pocos trabajos se puede afirmar que la formalidad ayuda verdaderamente a obtener especificaciones de alta calidad en dichos procesos, y que proporciona la base para su automatización. Algunos investigadores han experimentado la formalización en sí misma, pero en los resultados quedan planteadas muchas preguntas y se detectan problemas graves, por lo que sus formulaciones parecen informales. Además, aunque la semántica del formalismo que utilizan les proporciona reglas precisas de interpretación, generalmente caen en los muchos problemas de lenguaje natural.
4. ¿Para quién se especifica formalmente? Uno de los problemas comúnmente descrito en los trabajos analizados, en relación con las especificaciones formales, es que pueden involucrar diferentes clases de actores: clientes, especialistas en el dominio, usuarios, arquitectos, programadores y herramientas. Una solución propuesta por los autores es manejar este tipo de enfrentamientos con especificaciones

multilingües, pero esto eleva la complejidad de los problemas. Por eso es que otros autores afirman que un lenguaje de programación debe ser un lenguaje para el programador, no para la máquina. Aunque este principio todavía no es ampliamente aceptado para los lenguajes de especificación, porque muchos parecen estar todavía diseñados para los programadores o para las herramientas, en lugar de para los especificadores.

5. ¿Cuándo especificar? Para los autores analizados hay varias etapas en el ciclo de vida del software de las que pueden hacer parte las especificaciones formales: al modelar el dominio, al elaborar los objetivos, al elicitar requisitos, al diseñar el prototipo funcional, al diseñar la arquitectura del software, o al modificar o realizar reingeniería al software. Pero, y de acuerdo con los mismos resultados que publican, hasta el momento la especificación formal se utiliza principalmente durante el diseño del modelo funcional preliminar.

Con base en las respuestas halladas y para determinar el nivel del desarrollo de la especificación formal, se consideró la siguiente escala de valoración: 1) *alto*: cuando el ítem bajo análisis es ampliamente utilizado, aceptado y difundido por los investigadores y los resultados publicados son altamente satisfactorios; 2) *medio*: cuando se utiliza, acepta y difunde por comunidades restringidas de investigadores y académicos y los resultados obtenidos son satisfactorios; y 3) *bajo*: cuando solamente algunos investigadores lo utilizan, aplican y difunden, pero los resultados no son satisfactorios. En tal sentido, y de acuerdo con los resultados del análisis, se puede afirmar que el nivel de desarrollo actual de la especificación formal está en un nivel *medio*. Aunque este nivel está cerca del sentido común, hay algunos principios y hechos importantes que a menudo se pasan por alto en la comunidad de la formalización, y que todavía no han permitido un progreso mayor en su nivel de desarrollo:

- Las especificaciones no son formales *per se*. Para definir las propiedades de estado de un sistema, precisa y formalmente, primero hay que averiguar qué son estas propiedades. Lo que tradicionalmente se formula en un lenguaje que todas las partes puedan hablar y entender, es decir, el lenguaje natural.
- Para la Ingeniería del Software tradicional, las especificaciones formales no tienen sentido sin una definición precisa e informal de cómo interpretarlas en el dominio

del problema. Esto se debe a que se describen en lenguaje natural, con términos y predicados que pueden tener significados diferentes. De esta manera, la especificación solamente tiene sentido si el significado de cada término/predicado se indica de forma precisa. Además, para evitar la regresión infinita, no necesariamente estructurada. Un principio obvio que a menudo descuidan los investigadores.

- La especificación formal no es un simple proceso de traducir lo informal en formal. Esta es una creencia que comparten muchas de las empresas de la industria del software e investigadores de los métodos formales. Tomar el lenguaje natural y traducirlo al lenguaje matemático es una actividad de altas exigencias para el equipo de Ingeniería de Requisitos. El problema es que la academia no los capacita para hacerlo, y la mayoría lleva a cabo estos procesos de forma empírica, con las consecuencias que se presentan en los resultados de esta investigación.
- Las especificaciones formales son difíciles de desarrollar y evaluar. Esto se deriva de la diversidad y la sutileza de los errores que se puede cometer y de la multiplicidad de opciones de modelado que se puede tener. Como consecuencia, las especificaciones raramente son correctas. Sin embargo, en la literatura se señala con frecuencia que incluso las especificaciones equivocadas pueden ayudar a descubrir problemas en formulaciones originales.
- La razón de por qué seleccionar un modelado específico en una especificación, es importante para su explicación y evolución (Souquières y Levy, 1993). Por desgracia, este razonamiento raramente lo documentan los investigadores analizados.
- Los subproductos de un proceso de especificación formal son a menudo más importantes que la propia especificación. Pero, generalmente, incluyen una especificación más informal obtenida mediante la retroalimentación de las expresiones formales, la estructuración y el análisis.
- Para que sea útil, un sistema formal debe tener un dominio de aplicabilidad limitado. Por eso es que para la expresión natural y el análisis eficiente, los tipos específicos de sistemas requieren tipos específicos de técnicas. Pero, desconociendo esta realidad, en la literatura todavía se intenta encontrar una técnica de especificación universal.

Por otro lado, y aunque de acuerdo con los autores analizados la investigación sobre los métodos formales es permanente en todo el mundo, surgen diferencias culturales. Por ejemplo, la opinión general en los Estados Unidos es que la verificación formal es un mecanismo para demostrar la equivalencia entre una especificación formal y un programa. Pero, dado que los programas reales, implementados en equipos reales utilizando compiladores reales, tienen numerosas limitaciones, las pruebas necesariamente son difíciles y la verificación formal ha tenido poco impacto en la industria. Por otra parte, la visión europea es que la verificación formal es un mecanismo para mostrar la equivalencia entre una especificación formal y un diseño. Pero, mientras que desde el diseño de software se puede eliminar algo de la implementación real, la verificación formal es más fácil, aunque todavía se tiene el problema de mostrar que el diseño es equivalente con la aplicación. Debido a esto, la verificación formal es más prevalente en las actividades de desarrollo europeas que en las americanas. Estas diferencias culturales y operativas alrededor de los métodos formales, han puesto en evidencia algunas limitaciones de la especificación formal, que tampoco ayudan mucho para su desarrollo:

- Los procesos formales en el software no pueden garantizar que sea perfecto. Como se ha demostrado en este análisis, las técnicas formales se basan en la abstracción de un programa en una especificación abstracta muy cercana a la realidad. Sin embargo, esta especificación rara vez es exacta, por lo que el programa resultante solamente es una aproximación a lo especificado. Además, no se puede olvidar que las mismas pruebas matemáticas pueden tener errores, por lo que, sin duda, las pruebas formales ayudan, pero no son una garantía de perfección.
- Mediante las pruebas, los métodos formales buscan que los programas sean correctos. Pero, a los usuarios y clientes les interesa más que un simple funcionamiento correcto. Porque el costo, el tiempo de desarrollo, el rendimiento, la seguridad y la fiabilidad también son propiedades que hacen parte de una especificación completa. Algo que pocos investigadores en la especificación formal profundizan.
- Actualmente, y debido al estado del desarrollo de los métodos formales, solamente los sistemas altamente críticos se benefician de ellos. Aunque la realidad es que

prácticamente cualquier sistema se beneficiaría con el uso de las técnicas formales. La cuestión es que la misma comunidad, y las diferencias culturales, llevan a que esto no sea una realidad a corto plazo.

5. CONCLUSIONES

Somos una Sociedad software-dependiente porque este producto invade cada vez más los diferentes aspectos de la vida cotidiana. Por eso se requiere que se incremente su calidad y fiabilidad. Las especificaciones formales ofrecen un amplio espectro de posibles caminos hacia ese objetivo, por lo tanto, reciben cada vez más atención en el mundo académico, científico e industrial. Aun así, todavía hay un largo camino por recorrer antes que puedan ser utilizadas para cumplir la meta de superar la crisis del software. Entre los muchos retos planteados, los factores críticos de éxito serán la prestación de asistencia constructiva en el desarrollo y el análisis y la evolución de la especificación; la integración vertical y horizontal de las especificaciones formales dentro del ciclo de vida del software; las abstracciones de alto nivel para la especificación y el análisis de requisitos; la disponibilidad de técnicas formales para los aspectos no-funcionales; e interfaces livianas para la especificación y el análisis de paradigma múltiples.

Actualmente, los métodos formales son una realidad y la industria y la comunidad continúan desarrollando técnicas de especificación formal que, en el futuro, podrán funcionar adecuadamente y ofrecer las ventajas que se espera de ellas, incluso para aplicaciones poco críticas. Desde las técnicas formales han surgido buenas prácticas para la especificación y el desarrollo del software que no se pueden abandonar, el objetivo debe ser mejorarlas. Pero el trabajo futuro deberá reunir los principios de la automatización total de las pruebas y determinar las formas en que la formalización se puede utilizar con mayor efectividad, y buscar que los procesos formativos las incluyan en los contenidos curriculares. Los desarrolladores y la industria deben comenzar a tratar a los métodos formales, y por ende a la especificación formal, como una inversión y trabajar para garantizar que el software alcance la calidad que el usuario espera por su valor.

El desarrollo actual de la especificación formal es medio y, aunque el optimismo latente es que a corto plazo no logrará superarlo, existe una comunidad absorta en buscar alternativas para avanzar a un mayor ritmo. Por el momento es necesario adoptar las

técnicas que existen y perfeccionarlas mediante aplicaciones reales. Pero, lo más importante, es urgente que las Ciencias Computacionales incluyan a los métodos formales en los procesos formativos, porque la comunidad no está logrando el relevo generacional que necesita para cumplir la meta de mejorar la calidad del software.

REFERENCIAS

Abrial, J. (1980). The Specification Language Z. Syntax and Semantics. USA: Oxford University Press.

Akhtar, N.; Guyadec, Y. & Oquendo, F. (2012). Formal requirement and architecture specifications of a multi-agent robotic system. Journal of Computing 4(4), pp. 1-6.

Almeida, J. et al. (2011). An overview of formal methods tools and techniques. In Almeida, J. et al. (Eds.), Rigorous Software Development - An introduction to program verification (pp. 15-44). London: Springer.

Baber, R. (2011). The Language of Mathematics - Utilizing Math in Practice. USA: Wiley.

Bacelar, J. et al. (2011). An overview of formal methods tools and techniques. In Bacelar, J. et al. (Eds.), Rigorous Software Development - An Introduction to Program Verification. USA: Springer.

Barden, R., Stepney, S. y Cooper, D. (1995). Z in Practice. USA: Prentice-Hall.

Bjørner, D. (2006). Software Engineering 1: Abstraction and Modelling. New York: Springer.

Bollin, A. (2008). Concept Location in Formal Specifications. Journal of Software Maintenance and Evolution: Research and Practice 20(2), pp.77-105.

Bollin, A. (2011). Is there evolution before birth? Deterioration effects of formal Z specifications. Proceedings 13th international conference on Formal methods and software engineering (pp. 66-81). Berlin, Germany.

Bollin, A. (2013). Do You Speak Z? Formal Methods under the Perspective of a Cross-Cultural Adaptation Problem. Proceedings 1st FME Workshop on Formal Methods in Software Engineering (pp. 8-14). San Francisco, USA.

Clarke, E. & Wing, J. (1996). Formal Methods: State of the Art and Future Directions. ACM Computing Surveys 28(4), pp. 626-643.

Craigen, D.; Gerhart, S. & Ralston, T. (1993). An International Survey of Industrial Applications of Formal Methods. NISTGCR 93/626, US Dept. Commerce, Computer Systems Lab.3.

Cuellar, J. (2000). Special Issue on Formal Methods in Industry. USA: Elsevier.

Dijkstra, E.W. (1975). Guarded commands, non-determinacy and the formal derivation of programs". Communications of the ACM 18(8), pp. 453-457.

Diller, A. (1999). Z - An Introduction to Formal Methods. USA: John Wiley.

Finney, K., Rennolls, K. y Fedorec, A. (1998). Measuring the comprehensibility of Z specifications. Journal of Systems and Software 42(1):pp.3-15.

Floyd, R. (1967). Assigning Meanings to Programs. Mathematical Aspects of Computer Science 19, pp. 19-32.

Garlan, D. (1994). Integrating Formal Methods into a Professional Master of Software Engineering Program. Proceedings of the Eighth Z User Meeting (pp. 71-85). Cambridge, UK.

Gerhart, S. & Wile, D. (1979). Preliminary report of the delta experiment: Specification and verification of a multiple-user file updating module. Proceedings Specification of Reliable Software Conference, pp. 198-211. Palo Alto, USA.

Gravell, M. (1991). What is a Good Formal Specification? Proceedings of the Fifth Annual Z User Meeting on Z User Workshop (pp. 137-150). London, UK.

Heninger, K. (1980). Specifying Software Requirements for Complex Systems: New Techniques and their Application. IEEE Transactions on Software Engineering 6(1), pp. 2-13.

Hinchey, M. & Bowen, J. (1995). Applications of Formal Methods. USA: Prentice Hall.

Hoare, C. (1969). An Axiomatic Basis for Computer Programming. Communications of the ACM 12(10), pp. 576-583.

Holt, R. y Cordy, J. (1988). The Turing programming language. Communications of the ACM 31(12), pp. 1410-1423.

Jackson, D. (2006). Software Abstractions - Logic, Language, and Analysis. Cambridge: The MIT Press.

Jones, C., Jackson, D. y Wing, J. (1996). Formal Methods Light. Computer 29(4), pp. 20-22.

Knuth, D. (2011). The Art of Computer Programming. New York: Addison-Wesley.

Larsen, P. et al. (2010). The overture initiative integrating tools for VDM. Software Engineering Notes 35(1), pp. 1-6.

Larsen, P., Fitzgerald, J. y Riddle, S. (2006). Learning by doing: Practical courses in lightweight formal methods using VDM++. Technical Report Cs-tr-992. University of Newcastle.

Liskov, B. & Zilles, S. (1975). Specification Techniques for Data Abstractions. IEEE Transactions on Software Engineering 1(1), pp. 7-18.

Meataute, P. y Serna, A.A. (2015). Métricas del software - Herramientas de apoyo cuantificables para la toma de decisiones. En Serna, M.E. (Ed,), Avances en Ingeniería (pp. 237-245). Medellín: Editorial Instituto Antioqueño de Investigación.

Meyer, B. (1985). On Formalism in Specification. IEEE Software 2(1), pp. 6-26.

Naur, P. (1969). Proofs of algorithms by General Snapshots. BIT 6, pp. 310-316.

Neumann, P. (1989). Flaws in specifications and what to do about them. Proceedings Fifth International Workshop on Software Specification and Design (pp. xi-xv). Pittsburgh, USA:

O'Regan, G. (2006). Mathematical Approaches to Software Quality. London: Springer.

O'Regan, G. (2012). A brief history of computing. London: Springer.

Parnas, D. (1972). A Technique for Software Module Specification with Examples. Communications of the ACM 15(5), pp. 330-336.

Parnas, D. (1977). The Use of Precise Specifications in the Development of Software. Proceedings IFIP Congress, pp. 849-867. Toronto, Canada.

Penny, D., Holt, R. y Godfrey, M. (2005). Formal Specification in Metamorphic Programming. Lecture Notes in Computer Science 551, pp. 11-30

Pnueli, A. (1977). The Temporal Logics of Programs. Proceedings 18th IEEE Symposium on Foundations of Computer Science, pp. 46-57. Rhode Island, USA.

Randell, B. (1973). The Origin of Digital Computers. Berlin: Springer-Verlag.

Ryan, A. (2010). Formal specification of moving block railway interlocking using CASL. BSc Dissertation. Swansea University.

Serna, M.E. & Morales, V.D. (2014). State of the art in the research of formal verification. Revista Ingeniería Investigación y Tecnología XV(4), pp. 615-623.

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

Serna, M.E. (2013). Manifiesto por la Profesionalización del Desarrollo de Software. Medellín: Editorial Instituto Antioqueño de Investigación.

Serna, M.E. y Serna A.A. (2014). Formal specification in context: Current and future. Ingeniare. Revista chilena de ingeniería 22(2), pp. 243-256.

Serna, M.E. y Serna, A.A. (2014a). Methodology for perform reliable literature reviews. Revista Información, cultura y sociedad. In press.

Souquières, J. y Levy, N. (1993). Description of specification developments. Proceedings First IEEE Symposium on Requirements Engineering (pp. 216-223). San Diego, USA.

Spivey, J. (1989). The Z Notation. London: Prentice Hall.

Spivey, J. (2002). An introduction to Z and formal specifications. Software Engineering Journal 4(1), pp. 40-50.

Swartout, W. & Balzer, R. (1982). On the Inevitable Intertwining of Specification and Implementation. Communications of the ACM 25(7), pp. 438-440.

van der Poll, J. y Kotzé, P. (2002). What Design Heuristics May Enhance the Utility of a Formal Specification? Proceedings 2002 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on Enablement through Technology (pp. 179-194).

Vinter, R., Loomes, M. y Kornbrot, D. (1998). Applying Software Metrics to Formal Specifications: A Cognitive Approach. Proceedings 5th International Symposium on Software Metrics (pp. 216-223). Bethesda, USA.

Wing, J. (1990). A specifier's introduction to formal methods. IEEE Computer 23(9), pp. 8-23.

Wing, J.; Woodcock, J. & Davies, J. (1999). Memories of World Conference on Formal Methods in the Development of Computing Systems. Berlin: Springer-Verlag.

APÉNDICE (Artículo científico)

Nivel actual del desarrollo de la especificación formal

Edgar Serna M., Efilia Ariza V., Elizabeth Ariza V.

Resumen
El reto de superar la crisis del software se estableció hace más de cinco décadas y, aunque se han logrado importantes avances, todavía no se logra superar. Los métodos formales son una alternativa seria en la que actualmente se pone mucha confianza, pero su desarrollo y progreso todavía no tienen el ímpetu suficiente para masificarlos como herramienta en la Ingeniería de Software actual. Como componente de esta tecnología, la especificación formal es una técnica en la que se invierten esfuerzos y se presentan resultados en la literatura. En este trabajo se presentan los resultados de una investigación cuyo objetivo es determinar el nivel de desarrollo de la especificación formal. De acuerdo con el análisis a la revisión de la literatura, se encontró que no se ha superado el nivel medio de este desarrollo. En el resto del artículo se describe el trabajo realizado y se analizan los resultados para llegar a esta conclusión.

Palabras clave: Métodos formales, calidad del software, Ingeniería de Requisitos, ciclo de vida, lenguajes de especificación, crisis del software.

Abstract
The challenge to overcome the software crisis more than five decades ago and, although has been significant progress, still is not overcome. Formal methods are a serious alternative in which currently it puts a lot of trust, but its development and progress not yet have sufficient momentum to disseminate as a tool in software engineering today. As a component of this technology, the formal specification is a technique in which efforts are invested and results are reported in the literature. This paper presents the results of a research, which aims is to determine the level of development of formal specification. According to the analysis of the literature review, was found that has not been exceeded the average level of this development. In the rest of the article is described the work done and the results are analyzed to arrive at this conclusion.

Keywords: Formal methods, software quality, Requirements Engineering, life cycle, specification languages, software crisis.

INTRODUCCIÓN

Desde hace décadas, los métodos formales han interesado en los ambientes académicos y de investigación. Estos métodos, de base matemática para la especificación y el posterior desarrollo de sistemas software, proporcionan un mejor control de la complejidad de los sistemas actuales, ofreciendo facilidades para la abstracción, y hacen menos ambigua y más uniforme la especificación de requisitos y a menor costo (Serna y Serna, 2014). Sin embargo, su aplicación en la industria no ha tenido el nivel de aceptación y utilización que esperaban los pioneros del siglo pasado. Aunque se aplican en proyectos piloto y se conocen algunos casos aislados de grandes esfuerzos, en los entornos industriales su aplicación es hasta el momento limitada. Pero, a pesar de esta aparente falta de interés por aplicar los métodos formales industrialmente, definitivamente hay una necesidad de ellos para llevar a cabo la especificación formal (Almeida et al., 2011), especialmente en sistemas críticos. El problema de la transferencia de tecnología, que a menudo se presenta como una de las principales razones de la tensión entre la investigación y la industria, no es un problema exclusivo de los métodos formales. Las nuevas tecnologías, desarrolladas en otros campos de las Ciencias

Computacionales, también tienden a ser adoptadas muy lentamente en la industria, pero en el caso de los métodos formales, la situación parece ser aún peor (Serna, 2010).

El problema central de los métodos formales es que deben ser capaces de garantizar, siguiendo un enfoque rigurosamente matemático, el comportamiento de un sistema dado y en el corazón de ellos se encuentra la noción de la especificación. Una especificación es un modelo de un sistema que contiene una descripción de su comportamiento deseado, es decir, *qué* es lo que se va a implementar, y no *cómo* se va a hacer. Dicha especificación puede ser totalmente abstracta, en cuyo caso el modelo *es* la descripción del comportamiento, o puede ser más operacional, y entonces la descripción, de alguna manera, está contenida o implícita en el modelo.

Por otro lado, el creciente uso de software en las actividades de esta sociedad ha exigido normas y métodos estrictos de desarrollo. Debido a esto, en los últimos años y a través de la investigación y la innovación, se ha propuesto una serie de técnicas y herramientas orientadas en gran medida a mejorar los procesos de especificación, de diseño y de prueba de este producto. Todo el proceso, junto con las métricas tradicionales de calidad (Metaute y Serna, 2015), hace que el desarrollo de software evolucione de una condición de arte negro a una disciplina de ingeniería respetable y en rápido progreso, pero que no madurará sino hasta dentro de varios años (Serna, 2013). Tradicionalmente, la calidad ha sido definida como la conformidad con la especificación (cuando no se cumple, entonces surgen los fallos). Llegar a esto en el caso del software a menudo es muy complicado, porque de esta definición se desprende que un buen producto requiere no solamente que esta norma se cumpla, sino que es necesario contar con una buena especificación. Esto significa que todos los requisitos tienen que ser especificados plenamente y sin ambigüedades, porque, estrictamente hablando, los fallos no pueden ser reconocidos por funciones que no han sido especificadas previamente.

En este sentido, el principal problema con el software es la falta de capacidad para prever y definir la cantidad y complejidad de los requisitos. Esto incluye las complicaciones que provienen de la combinación de los modos operativos y las condiciones de entrada y de los entornos. Además, debido a que cada vez los sistemas se hacen más grandes y más complejos, se incrementan los problemas en la Ingeniería de Requisitos y a menudo la entrega del producto se retrasa, porque la especificación de requisitos es la base para las demás fases del ciclo de vida del producto. Aquí es donde cobra importancia la especificación formal, porque es una alternativa para unificar los criterios, comprensiones y puntos de vista del equipo de trabajo alrededor de los requisitos del sistema. Pero, si se aplica la formalización en el proceso, se podría garantizar de mejor forma que el producto cumple lo especificado, y por tanto, también podría mejorar su calidad.

Es así, que los métodos modernos (formales) implican un enfoque más matemático para el desarrollo de la Ingeniería del Software y, aunque las pruebas matemáticas de funcionamiento y de validación de la semántica del código no pueden ser perfectas, al menos ofrecen un método más formal de garantía de calidad que el de los métodos anteriores. Para lograrlo, las áreas clave a tener en cuenta son: 1) la especificación formal, es decir, el uso de métodos basados matemáticamente e incluso automatizados para traducir los requisitos en el diseño; 2) las metodologías de diseño, o librerías de paquetes software reutilizables, con lo que las pruebas de funcionamiento y de validación de código pueden ser en gran medida automatizadas; y 3) las pruebas, en las que las especificaciones de prueba pueden ser generadas automáticamente por las herramientas de diseño.

En este artículo se presentan los resultados de una investigación cuyo objetivo es determinar el nivel actual del desarrollo de la especificación formal. El contenido se estructura de la siguiente forma: en la primera parte se detallan los trabajos relacionados, en la segunda se describe la metodología aplicada, en la tercera se presentan los resultados, los cuales se analizan y discuten en la cuarta sección. Finalmente, en la quinta parte se concluye el trabajo realizado.

TRABAJOS RELACIONADOS

Desde la perspectiva de una Ingeniería de Requisitos y cuando se trata de la especificación formal, lograr la ausencia de defectos es relevante. Pero, visto desde la perspectiva de la Ingeniería del Software o del mantenimiento, cuentan otros atributos. Por eso es que también son importantes factores como legibilidad, comprensibilidad, seguridad y fiabilidad. Por desgracia, son escasos los trabajos en los que se difunden investigaciones relacionadas con el nivel de desarrollo de la especificación formal, y a menudo no están realmente basados en investigaciones empíricas.

Gravell (1991) propuso una serie de principios para construir una especificación escrita en Z (Spivey, 1989) derivados de observaciones personales. Se centró principalmente en la hacer especificaciones más legibles, y se le ocurrió una serie de fragmentos de especificación preferibles y una gran cantidad de ejemplos contrarios. Es comprensible que no demostrara el desarrollo de la especificación formal, porque apenas se estaba difundiendo como principio para el desarrollo de software. Posteriormente, Finney, Rennolls y Fedorce (1998) abordaron nuevamente esta cuestión, y en un estudio empírico observaron la influencia de los comentarios, las menciones y la estructura de las especificaciones en Z, y encontraron incidencias de la influencia de los comentarios sobre las valoraciones obtenidas por los equipos de prueba. Su conclusión fue que sería deseable tener disponibles algunos índices de comprensión del desarrollo de la especificación formal, para estructurar un modelo de Ingeniería de Requisitos para satisfacerlos.

Vinter, Loomes y Kornbrot (1998) demostraron que incluso las especificaciones formales contienen errores (ya sea de los requisitos o porque las construcciones matemáticas que se utilizan mal o de manera incomprensible). Encontraron que el número de defectos en las especificaciones formales está fuertemente correlacionado con el número de implicaciones utilizadas en los predicados de una especificación Z. Además, hallaron una correlación con la complejidad percibida de la misma especificación. Con su trabajo, demostraron que en ese momento se había avanzado poco en el desarrollo de la especificación formal.

Motivados por la estrategia establecida por Barden, Stepney y Cooper (1995), van der Poll y Kotzé (2002) examinan las especificaciones Z y algunas heurísticas de diseño de Ingeniería del Software, y propusieron diez heurísticas de diseño, a nivel de esquema, que aplicaron para escribir la especificación formal. Para ese momento demostraron que el desarrollo de la misma era lento, pero que la comunidad trabajaba decididamente en mejorarlo. Aparte de estas cuestiones con las especificaciones, los llamados métodos formales livianos también ganaron terreno (Jones, Jackson y Wing, 1996; Larsen, Fitzgerald y Riddle, 2006), y con ellos se desarrollaron entornos y lenguajes de apoyo a la transformación automatizada y al análisis gráfico de modelos (Jackson, 2006; Larsen et al., 2010). Sin embargo, a pesar que se pueden generar automáticamente grandes porciones de texto de especificación, la legibilidad sigue siendo un problema, y esto es importante cuando se necesita mantener la especificación actualizada durante el ciclo de vida del desarrollo, o cuando tiene que ser corregida debido a

problemas identificados durante el modelamiento. Con este trabajo se pudo demostrar que el desarrollo de la especificación formal iba por buen camino.

METODOLOGÍA

El procedimiento para esta investigación se basó en una metodología para realizar revisiones fiables de la literatura (Serna y Serna, 2014a), y se orientó a identificar estudios y opiniones que pudieran ser candidatos a incluir en la muestra final. Las fuentes fueron las bases de datos ACM digital library, IEEE Xplore digital library, Sciendirect, Springer y Web of Science. Los parámetros de búsqueda incluyeron las palabras clave: *formal specification, formal methods*, *Requirments Engineering* y *requirements elicitation*. Las combinaciones debían aparecer en el título, el resumen o en el contenido del documento. Para lograr mayor cubrimiento, no se determinó una línea de tiempo específica y no se excluyó ninguna fuente inicial. Además, en los estudios primarios seleccionados se validó la solidez de la metodología aplicada y del análisis a los resultados presentados. Las opiniones y/o análisis debían ser presentadas por instituciones o investigadores que estuvieran relacionados con el área temática y que su trayectoria fortaleciera el postulado.

En la muestra final se incluyeron directamente: 1) los artículos de revistas, por haber pasado por procesos de selección y evaluación estructurados, 2) las presentaciones en eventos, cuyos autores y/o empresas fueran idóneos en el área, y 3) los reportes técnicos, cuyos autores estuvieran relacionados con el trabajo en especificación formal y/o tuvieran experiencia en los campos relacionados. Al final, la muestra inicial quedó conformada por 185 trabajos. Como criterio de calidad, debían hacer un aporte relevante a la temática de investigación. Para cumplirlo, se aplicaron los siguientes procesos: 1) identificar los estudios relevantes, 2) excluir estudios con base en el título, 3) excluir estudios a base en los resúmenes, y 4) analizar los estudios y seleccionar los más relevantes para la temática en función del texto completo. Los criterios de inclusión y exclusión más importantes fueron: formalidad y pertinencia de la publicación, autoridad del autor, calidad y aportes del contenido, fuentes de datos, sustentación de la tesis, calidad de la investigación y coherencia de los resultados. Luego de este proceso se conformó la muestra final para la investigación, cuyo análisis y resultados se presentan a continuación.

RESULTADOS

Desde el surgimiento de las Ciencias Computacionales, los investigadores han considerado a la especificación formal como una de sus áreas de interés. Finalizando los años cuarenta, Turing (Randell, 1973) observó que el razonamiento acerca de un programa secuencial era más sencillo cuando, en puntos específicos del mismo, se hacían anotaciones acerca de las propiedades de su estado. En los años sesenta, Floyd (1967), Hoare (1969) y Naur (1969) propusieron técnicas axiomáticas para demostrar la consistencia entre los programas secuenciales y esas propiedades, a las que llamaron *especificaciones*, y Dijkstra (1975) demostró cómo utilizar constructivamente el cálculo para derivar programas no-determinísticos que las cumplieran. Parnas (1972) y Liskov y Zilles (1975) propusieron técnicas específicas para expresar formalmente las propiedades de los programas, particularmente para datos estructurados, y Pnueli (1977) lo hizo para programas concurrentes. Estos aportes conformaron el punto de partida para la *especificación formal* como área de investigación y desarrollo (Parnas, 1977; Abrial, 1980; Heninger, 1980). Desde entonces se ha incrementado continuamente el interés que despertó, lo mismo que los usos en la Ingeniería del Software (Wing, 1990; Hinchey y Bowen, 1995; Clarke y Wing, 1996), aunque su desarrollo y aplicación han caminado por senderos diferentes.

De acuerdo con Ryan (2010), la especificación formal encaja en la etapa de diseño del sistema y su objetivo es estructurar una definición formal de los requisitos del mismo, eliminando la ambigüedad y la incertidumbre del documento de diseño e implementación. De esta manera se les facilita a los desarrolladores predecir el comportamiento y las propiedades del sistema, antes que se desarrolle (Bjørner, 2006). Los resultados de esta especificación son equivalentes al efecto que tuvieron las matemáticas y la geometría en la arquitectura de las civilizaciones tempranas, eliminando las conjeturas y el ensayo-error. Cuando se configura una especificación del sistema, solamente se modela el comportamiento del mismo. Esto significa que, en gran medida, la especificación puede ignorar los detalles que hacen parte de las decisiones de implementación, tales como el lenguaje de programación y los requisitos de recursos del sistema. Esto permite libertad para centrarse exclusivamente en el comportamiento del sistema, lo que resulta en un nivel adicional de abstracción por encima del código del programa. De esta manera, el lugar de la especificación formal es entre el diseño informal del sistema y la implementación del código (Ryan, 2010).

Para Serna y Serna (2014), una especificación formal es una especificación expresada en un lenguaje cuyo vocabulario, sintaxis y semántica se definen formalmente, y tiene una base lógica matemática usualmente formal. Es un área de investigación activa en la Ingeniería del Software y se aplica en diversas configuraciones y técnicas. Aunque su uso industrial todavía es limitado, aunque actualmente la comunidad científica tiene una comprensión diferente acerca de su utilidad y necesidad. Hasta el momento, el trabajo de los investigadores se focaliza en la especificación escrita durante el diseño del modelo funcional preliminar, por lo que se centran principalmente en evaluar las herramientas relacionadas.

La creciente importancia de los métodos de especificación formal se refleja en el número de publicaciones que se ocupan de ellos y en las investigaciones que defienden su uso en los procesos de desarrollo de software (Bollin, 2008). De acuerdo con este autor, entre las justificaciones sugeridas se incluyen: que el uso de la especificación formal mejora la visión y comprensión de los requisitos, ayuda a clarificar los requisitos al revelar o evitar contradicciones y ambigüedades en la especificación, permite una verificación rigurosa de las especificaciones y aplicaciones, facilita la transición de la especificación y el diseño a la implementación y se espera que ayuden a mejorar la calidad y fiabilidad del software, reduciendo así los costos del ciclo de vida.

Por su parte, van der Poll y Kotzé (2002) señalan que los métodos de especificación formal no han sido ampliamente aceptados en la industria del software, y que una de las principales razones es que esta especificación se ha ocupado sobre todo de los lenguajes, no del proceso de elicitación. Sin embargo, se ha propuesto una serie de estrategias para su incorporación masiva en el proceso de desarrollo de software. Con la proliferación de estas estrategias se ha llegado a una etapa en su desarrollo donde surge la necesidad de organizarlas y clasificarlas. Este es un paso necesario para la comprensión de su posible utilidad y para identificar similitudes y diferencias entre ellas, y para evaluar su aplicabilidad en diferentes contextos académicos e industriales. Sin embargo, para la mayoría de profesionales no son fáciles de utilizar en el desarrollo de proyectos reales, lo que todavía resulta ser un desafío para el trabajo con los métodos formales que la academia debe comenzar a enfrentar.

Las especificaciones formales no son documentos que se escriban una vez y, generalmente, no se logran justo al comienzo del proceso de desarrollo (Bollin, 2011). Se necesita tiempo para crear una primera versión de utilidad para, después de mucho esfuerzo y revisiones, poder desarrollar una especificación que esté cerca de los conceptos que tienen en sus mentes los

actores del proceso. Como argumenta Bollin (2013), la facilidad de comprensión es una cuestión clave (entre otros atributos de calidad), cuando se trabaja en y con una especificación formal. Para van der Poll y Kotzé (2002) una especificación, construida de acuerdo con los criterios establecidos para el diseño, podría ser más comprensible para los usuarios y más fácil de mejorar, lo que facilitaría el mantenimiento a largo plazo. El hecho es que, cuando se fomenta la integración entre los métodos formales y las comunidades de Ingeniería del Software, parece ser necesario dedicar mayor atención a este atributo de calidad. El supuesto es que, al aumentar la comprensión, también es muy probable que se aumente la aceptabilidad. Esto, a su vez, también facilitaría el uso y la enseñanza de los métodos formales (Garlan, 1994).

A través de los años muchos investigadores de los métodos formales han estado de acuerdo en que la especificación formal no se utiliza tanto como debería (Meyer, 1985; Neumann, 1989; Wing, 1990). En parte, porque la mayoría tiene formación en diseño de lenguajes y en construcción de sistemas operativos, o han incursionado vagamente en el diseño y producción de lenguajes de programación (Holt y Cordy, 1988). Pero, como investigadores, siempre han estado atraídos por los métodos formales. Sin embargo, y por razones pragmáticas, no han sido capaces de aplicarlos a la Ingeniería del Software, concretamente a la especificación formal. Penny, Holt y Godfrey (2005) presentan sus puntos de vista acerca de por qué pasa esto y concluyen que se necesita mayor énfasis en el tema de los métodos formales en los planes de estudio de las Ciencias Computacionales.

En la literatura, el término métodos formales se refiere a diversas técnicas matemáticas utilizadas en la Ingeniería del Software para la especificación y el desarrollo formal. Consisten de lenguajes de especificación o notaciones formales, y emplean una colección de herramientas para apoyar la comprobación de la sintaxis de la especificación y la prueba de sus propiedades (Baber, 2011). Utilizan ampliamente la abstracción, lo que les permite responder preguntas acerca de qué hace el sistema, independientemente de la aplicación. Por otra parte, la naturaleza no-ambigua de la notación matemática evita el problema de la especulación sobre el significado de las frases en lenguaje natural y lo impreciso de la redacción (Knuth, 2011). El lenguaje natural es inherentemente ambiguo, mientras que los métodos formales emplean notaciones precisas de acuerdo con las normas de inferencia (O'Regan, 2006).

Para O'Regan (2012), la especificación formal se convierte en el punto clave de referencia para las diferentes partes implicadas en el desarrollo del sistema. Se puede utilizar como punto de referencia en la documentación de los requisitos, la ejecución del programa y el plan de pruebas. La especificación formal es un medio valioso para promover un entendimiento común para todos los interesados en el sistema. El término métodos formales se utiliza para describir un lenguaje de especificación y un método formal, para el diseño e implementación de sistemas informáticos.

Por su parte, Spivey (2002) sostiene que la especificación formal tienen tres virtudes: es concisa, es precisa y no tiene ambigüedades. Es concisa porque es capaz de expresar en poco espacio hechos complejos sobre los Sistemas de Información. La experiencia práctica demuestra que una especificación matemática de un sistema, es mucho más corta que una especificación informal equivalente. Es precisa porque permite que los requisitos se documenten con claridad. La función deseada de un sistema se describe de manera que no limite indebidamente las estructuras de datos utilizadas para representar la información en el sistema, o los algoritmos utilizados para calcular con ella. No tiene ambigüedades porque cuando se realiza en un lenguaje estandarizado, evita las diversas interpretaciones y tiene un significado único.

ANÁLISIS Y DISCUSIÓN

Para los autores analizados, si el objetivo es producir software de buena calidad, existe una motivación muy fuerte para utilizar las mejores prácticas en la Ingeniería del Software. Los defectos en el software pueden causar, en el mejor de los casos, problemas menores, pero también generar contratiempos importantes en los sistemas, incluyendo pérdida de vidas o catástrofes económicas. En consecuencia, las empresas deben emplear las mejores prácticas para desarrollar software de alta calidad, y la especificación formal es una tecnología de vanguardia que puede ser de beneficio para lograrlo, debido a que se podría reducir la aparición de defectos en los productos.

Además, una de las ideas clave de la especificación formal, detectada en esta investigación, es abstraerse de lo particular de la implementación y escribir los detalles del sistema de manera compacta. Sin embargo, esta densidad de expresión de los pensamientos, combinada con la complejidad y el tamaño de los modismos matemáticos, también se convierte en una dificultad para alcanzar la comprensión de las especificaciones en las fases posteriores del ciclo de vida. Como sugieren algunos autores (Diller, 1999), la complejidad de tamaño (que está estrechamente relacionada con el problema de la complejidad inherente) puede abordarse mediante técnicas de descomposición y estructuración, provistas por los lenguajes de especificación. Pero, para la comprensión de la enorme cantidad de texto (y un gran número de dependencias entre los diferentes términos en la especificación informal) se requiere la ayuda de algunas herramientas (Bollin, 2008).

De acuerdo con los resultados obtenidos, aparte del nivel estructural, los lenguajes de especificación no ofrecen mucho apoyo. En algunos de los trabajos analizados se encuentran ejemplos o pautas que, directa o indirectamente, sugieren cómo escribir las descripciones de los requisitos en una especificación formal. Sin embargo, no es posible conocer todavía si las diferentes variantes para hacerlo, solamente son una cuestión de estilo o tienen verdadero impacto en la aparición o no de los errores, en el tiempo de comprensión o en la facilidad de comprensión de los requisitos. Debido a esto, muchos autores se orientan a intentar clarificar la cuestión de qué es una especificación comprensible y qué la hace diferente de una buena especificación (formal o informal). Pero, en términos generales, los aportes no logran convencer completamente y muchos se limitan a tratar de presentar directrices para apoyar la comprensión de los términos usados en la especificación. Esto no puede ser suficiente para garantizar una buena especificación formal, porque debe ser sintáctica y semánticamente correcta, de tal manera que permita hacerle seguimiento a todos los conceptos, sin perder la conexión con el modelo mental del sistema especificado. Este proceso de mapeo no debe ser percibido como agotador y debe ser completado dentro de un tiempo razonable. Pero esta cuestión no se analiza en los trabajos de la muestra final de esta investigación, y no se perciben aportes que extiendan la percepción común de lo que es una buena especificación formal, es decir: completa, consistente y adecuada. Sin embargo, hay que tener en cuenta que la facilidad de comprensión no es solamente algo que se debe tener en cuanta porque sí, sino que es un requisito esencial para decidir acerca de la corrección semántica de la misma especificación.

Por eso es que para lograr los objetivos de esta investigación, se debe suponer que las especificaciones formales descritas en los trabajos utilizan los tipos correctos y, por tanto, son por lo menos sintáctica y semánticamente correctas (en el sentido de que reflejan el sistema a fondo y en el nivel más adecuado de abstracción). Entonces, para determinar el nivel actual del desarrollo de la especificación formal, solamente queda el aspecto sustancial de analizar si el proceso en sí es de calidad. Pero esto tampoco es fácil de deducir en la revisión, porque generalmente los autores confunden formalidad con precisión, y aunque lo primero implica lo

segundo la relación contraria no es cierta. Una especificación es formal si se expresa en un lenguaje que posee tres componentes: 1) reglas para determinar gramaticalmente la formación correcta de oraciones (sintaxis), 2) reglas para interpretar las frases de manera precisa y significativa dentro del dominio considerado (semántica), y 3) reglas para inferir información útil a partir de la especificación (teoría de la prueba). Este último componente proporciona las bases para el análisis automatizado de la especificación.

¿Entonces, cómo determinar el nivel actual del desarrollo de la especificación formal? Para encontrar una respuesta, en esta investigación se formularon interrogantes que debían responder los resultados de la revisión de la literatura:

1. ¿Qué es una buena especificación? De acuerdo con los autores analizados, escribir una especificación correcta es muy difícil (probablemente tanto como escribir un programa correcto). En términos generales, se descubre que para ellos una especificación debe ser: 1) adecuada: expresar adecuadamente el problema en cuestión; 2) coherente internamente: tener una interpretación semántica significativa que haga verdaderas las propiedades de todos los especificados en su conjunto; 3) no-ambigua: no puede tener múltiples interpretaciones de lo que es cierto; 4) completa con respecto al más alto nivel descriptivo: la colección de propiedades especificadas debe ser suficiente para establecer el más alto de ellos, y a la vez debe ser satisfecha por los niveles inferiores; y 5) mínima: no indicar propiedades irrelevantes para el problema o que solamente sean relevantes para una solución del mismo.

2. ¿Qué es una especificación formal? Puede referirse a cosas diferentes en el ciclo de vida del software, por lo que los aportes que la intentan definir están sobrecargados. Una fuente adicional de confusión es que una sola palabra se utiliza indistintamente para un producto y para el proceso correspondiente. En términos generales, los autores afirman que una especificación formal es la expresión, en un lenguaje formal y con cierto nivel de abstracción, de una colección de propiedades que debe satisfacer un sistema. Pero esta definición, resumen de propósito general, cubre diferentes nociones que dependen de lo que debe comprenderse como *sistema*, qué tipo de propiedades son de interés, qué nivel de abstracción se considera y qué tipo de lenguaje formal se utiliza.

3. ¿Por qué especificar formalmente? Del análisis a las lecturas se concluye que especificar un problema es esencial para el diseño, la validación, la documentación, la comunicación, la reingeniería y la reutilización de las soluciones. Pero en muy pocos trabajos se puede afirmar que la formalidad ayuda verdaderamente a obtener especificaciones de alta calidad en dichos procesos, y que proporciona la base para su automatización. Algunos investigadores han experimentado la formalización en sí misma, pero en los resultados quedan planteadas muchas preguntas y se detectan problemas graves, por lo que sus formulaciones parecen informales. Además, aunque la semántica del formalismo que utilizan les proporciona reglas precisas de interpretación, generalmente caen en los muchos problemas de lenguaje natural.

4. ¿Para quién se especifica formalmente? Uno de los problemas comúnmente descrito en los trabajos analizados, en relación con las especificaciones formales, es que pueden involucrar diferentes clases de actores: clientes, especialistas en el dominio, usuarios, arquitectos, programadores y herramientas. Una solución propuesta por los autores es manejar este tipo de enfrentamientos con especificaciones multilingües, pero esto eleva la complejidad de los problemas. Por eso es que otros autores afirman que un lenguaje de programación debe ser un lenguaje para el programador, no para la máquina. Aunque este principio todavía no es

ampliamente aceptado para los lenguajes de especificación, porque muchos parecen estar todavía diseñados para los programadores o para las herramientas, en lugar de para los especificadores.

5. ¿Cuándo especificar? Para los autores analizados hay varias etapas en el ciclo de vida del software de las que pueden hacer parte las especificaciones formales: al modelar el dominio, al elaborar los objetivos, al elicitar requisitos, al diseñar el prototipo funcional, al diseñar la arquitectura del software, o al modificar o realizar reingeniería al software. Pero, y de acuerdo con los mismos resultados que publican, hasta el momento la especificación formal se utiliza principalmente durante el diseño del modelo funcional preliminar.

Con base en las respuestas halladas y para determinar el nivel del desarrollo de la especificación formal, se consideró la siguiente escala de valoración: 1) *alto*: cuando el ítem bajo análisis es ampliamente utilizado, aceptado y difundido por los investigadores y los resultados publicados son altamente satisfactorios; 2) *medio*: cuando se utiliza, acepta y difunde por comunidades restringidas de investigadores y académicos y los resultados obtenidos son satisfactorios; y 3) *bajo*: cuando solamente algunos investigadores lo utilizan, aplican y difunden, pero los resultados no son satisfactorios. En tal sentido, y de acuerdo con los resultados del análisis, se puede afirmar que el nivel de desarrollo actual de la especificación formal está en un nivel *medio*. Aunque este nivel está cerca del sentido común, hay algunos principios y hechos importantes que a menudo se pasan por alto en la comunidad de la formalización, y que todavía no han permitido un progreso mayor en su nivel de desarrollo:

- Las especificaciones no son formales *per se*. Para definir las propiedades de estado de un sistema, precisa y formalmente, primero hay que averiguar qué son estas propiedades. Lo que tradicionalmente se formula en un lenguaje que todas las partes puedan hablar y entender, es decir, el lenguaje natural.

- Para la Ingeniería del Software tradicional, las especificaciones formales no tienen sentido sin una definición precisa e informal de cómo interpretarlas en el dominio del problema. Esto se debe a que se describen en lenguaje natural, con términos y predicados que pueden tener significados diferentes. De esta manera, la especificación solamente tiene sentido si el significado de cada término/predicado se indica de forma precisa. Además, para evitar la regresión infinita, no necesariamente estructurada. Un principio obvio que a menudo descuidan los investigadores.

- La especificación formal no es un simple proceso de traducir lo informal en formal. Esta es una creencia que comparten muchas de las empresas de la industria del software e investigadores de los métodos formales. Tomar el lenguaje natural y traducirlo al lenguaje matemático es una actividad de altas exigencias para el equipo de Ingeniería de Requisitos. El problema es que la academia no los capacita para hacerlo, y la mayoría lleva a cabo estos procesos de forma empírica, con las consecuencias que se presentan en los resultados de esta investigación.

- Las especificaciones formales son difíciles de desarrollar y evaluar. Esto se deriva de la diversidad y la sutileza de los errores que se puede cometer y de la multiplicidad de opciones de modelado que se puede tener. Como consecuencia, las especificaciones raramente son correctas. Sin embargo, en la literatura se señala con frecuencia que incluso las especificaciones equivocadas pueden ayudar a descubrir problemas en formulaciones originales.

- La razón de por qué seleccionar un modelado específico en una especificación, es importante para su explicación y evolución (Souquières y Levy, 1993). Por desgracia, este razonamiento raramente lo documentan los investigadores analizados.

- Los subproductos de un proceso de especificación formal son a menudo más importantes que la propia especificación. Pero, generalmente, incluyen una especificación más informal obtenida mediante la retroalimentación de las expresiones formales, la estructuración y el análisis.

- Para que sea útil, un sistema formal debe tener un dominio de aplicabilidad limitado. Por eso es que para la expresión natural y el análisis eficiente, los tipos específicos de sistemas requieren tipos específicos de técnicas. Pero, desconociendo esta realidad, en la literatura todavía se intenta encontrar una técnica de especificación universal.

Por otro lado, y aunque de acuerdo con los autores analizados la investigación sobre los métodos formales es permanente en todo el mundo, surgen diferencias culturales. Por ejemplo, la opinión general en los Estados Unidos es que la verificación formal es un mecanismo para demostrar la equivalencia entre una especificación formal y un programa. Pero, dado que los programas reales, implementados en equipos reales utilizando compiladores reales, tienen numerosas limitaciones, las pruebas necesariamente son difíciles y la verificación formal ha tenido poco impacto en la industria. Por otra parte, la visión europea es que la verificación formal es un mecanismo para mostrar la equivalencia entre una especificación formal y un diseño. Pero, mientras que desde el diseño de software se puede eliminar algo de la implementación real, la verificación formal es más fácil, aunque todavía se tiene el problema de mostrar que el diseño es equivalente con la aplicación. Debido a esto, la verificación formal es más prevalente en las actividades de desarrollo europeas que en las americanas. Estas diferencias culturales y operativas alrededor de los métodos formales, han puesto en evidencia algunas limitaciones de la especificación formal, que tampoco ayudan mucho para su desarrollo:

- Los procesos formales en el software no pueden garantizar que sea perfecto. Como se ha demostrado en este análisis, las técnicas formales se basan en la abstracción de un programa en una especificación abstracta muy cercana a la realidad. Sin embargo, esta especificación rara vez es exacta, por lo que el programa resultante solamente es una aproximación a lo especificado. Además, no se puede olvidar que las mismas pruebas matemáticas pueden tener errores, por lo que, sin duda, las pruebas formales ayudan, pero no son una garantía de perfección.

- Mediante las pruebas, los métodos formales buscan que los programas sean correctos. Pero, a los usuarios y clientes les interesa más que un simple funcionamiento correcto. Porque el costo, el tiempo de desarrollo, el rendimiento, la seguridad y la fiabilidad también son propiedades que hacen parte de una especificación completa. Algo que pocos investigadores en la especificación formal profundizan.

- Actualmente, y debido al estado del desarrollo de los métodos formales, solamente los sistemas altamente críticos se benefician de ellos. Aunque la realidad es que prácticamente cualquier sistema se beneficiaría con el uso de las técnicas formales. La cuestión es que la misma comunidad, y las diferencias culturales, llevan a que esto no sea una realidad a corto plazo.

CONCLUSIONES

Somos una Sociedad software-dependiente porque este producto invade cada vez más los diferentes aspectos de la vida cotidiana. Por eso se requiere que se incremente su calidad y fiabilidad. Las especificaciones formales ofrecen un amplio espectro de posibles caminos hacia ese objetivo, por lo tanto, reciben cada vez más atención en el mundo académico, científico e industrial. Aun así, todavía hay un largo camino por recorrer antes que puedan ser utilizadas para cumplir la meta de superar la crisis del software. Entre los muchos retos planteados, los factores críticos de éxito serán la prestación de asistencia constructiva en el desarrollo y el análisis y la evolución de la especificación; la integración vertical y horizontal de las especificaciones formales dentro del ciclo de vida del software; las abstracciones de alto nivel para la especificación y el análisis de requisitos; la disponibilidad de técnicas formales para los aspectos no-funcionales; e interfaces livianas para la especificación y el análisis de paradigma múltiples.

Actualmente, los métodos formales son una realidad y la industria y la comunidad continúan desarrollando técnicas de especificación formal que, en el futuro, podrán funcionar adecuadamente y ofrecer las ventajas que se espera de ellas, incluso para aplicaciones poco críticas. Desde las técnicas formales han surgido buenas prácticas para la especificación y el desarrollo del software que no se pueden abandonar, el objetivo debe ser mejorarlas. Pero el trabajo futuro deberá reunir los principios de la automatización total de las pruebas y determinar las formas en que la formalización se puede utilizar con mayor efectividad, y buscar que los procesos formativos las incluyan en los contenidos curriculares. Los desarrolladores y la industria deben comenzar a tratar a los métodos formales, y por ende a la especificación formal, como una inversión y trabajar para garantizar que el software alcance la calidad que el usuario espera por su valor.

El desarrollo actual de la especificación formal es medio y, aunque el optimismo latente es que a corto plazo no logrará superarlo, existe una comunidad absorta en buscar alternativas para avanzar a un mayor ritmo. Por el momento es necesario adoptar las técnicas que existen y perfeccionarlas mediante aplicaciones reales. Pero, lo más importante, es urgente que las Ciencias Computacionales incluyan a los métodos formales en los procesos formativos, porque la comunidad no está logrando el relevo generacional que necesita para cumplir la meta de mejorar la calidad del software.

REFERENCIAS

Abrial, J. (1980). The Specification Language Z. Syntax and Semantics. USA: Oxford University Press.

Almeida, J. et al. (2011). An overview of formal methods tools and techniques. In Almeida, J. et al. (Eds.), Rigorous Software Development - An introduction to program verification (pp. 15-44). London: Springer.

Baber, R. (2011). The Language of Mathematics - Utilizing Math in Practice. USA: Wiley.

Barden, R., Stepney, S. y Cooper, D. (1995). Z in Practice. USA: Prentice-Hall.

Bjørner, D. (2006). Software Engineering 1: Abstraction and Modelling. New York: Springer.

Bollin, A. (2008). Concept Location in Formal Specifications. Journal of Software Maintenance and Evolution: Research and Practice 20(2), pp.77-105.

Bollin, A. (2011). Is there evolution before birth? Deterioration effects of formal Z specifications. Proceedings 13th international conference on Formal methods and software engineering (pp. 66-81). Berlin, Germany.

Bollin, A. (2013). Do You Speak Z? Formal Methods under the Perspective of a Cross-Cultural Adaptation Problem. Proceedings 1st FME Workshop on Formal Methods in Software Engineering (pp. 8-14). San Francisco, USA.

Clarke, E. y Wing, J. (1996). Formal Methods: State of the Art and Future Directions. ACM Computing Surveys 28(4), pp. 626-643.

Dijkstra, E.W. (1975). Guarded commands, nondeterminacy and the formal derivation of programs. Communications of the ACM 18(8), pp. 453-457.

Diller, A. (1999). Z - An Introduction to Formal Methods. USA: John Wiley.

Finney, K., Rennolls, K. y Fedorec, A. (1998). Measuring the comprehensibility of Z specifications. Journal of Systems and Software 42(1):pp.3-15.

Floyd, R. (1967). Assigning Meanings to Programs. Mathematical Aspects of Computer Science 19, pp. 19-32.

Garlan, D. (1994). Integrating Formal Methods into a Professional Master of Software Engineering Program. Proceedings of the Eighth Z User Meeting (pp. 71-85). Cambridge, UK.

Gravell, M. (1991). What is a Good Formal Specification? Proceedings of the Fifth Annual Z User Meeting on Z User Workshop (pp. 137-150). London, UK.

Heninger, K. (1980). Specifying Software Requirements for Complex Systems: New Techniques and their Application. IEEE Transactions on Software Engineering 6(1), pp. 2-13.

Hinchey, M. y Bowen, J. (1995). Applications of Formal Methods. USA: Prentice-Hall.

Hoare, C. (1969). An Axiomatic Basis for Computer Programming. Communications of the ACM 12(10), pp. 576-583.

Holt, R. y Cordy, J. (1988). The Turing programming language. Communications of the ACM 31(12), pp. 1410-1423.

Jackson, D. (2006). Software Abstractions - Logic, Language, and Analysis. Cambridge: The MIT Press.

Jones, C., Jackson, D. y Wing, J. (1996). Formal Methods Light. Computer 29(4), pp. 20-22.

Knuth, D. (2011). The Art of Computer Programming. New York: Addison-Wesley.

Larsen, P. et al. (2010). The overture initiative integrating tools for VDM. Software Engineering Notes 35(1), pp. 1-6.

Larsen, P., Fitzgerald, J. y Riddle, S. (2006). Learning by doing: Practical courses in lightweight formal methods using VDM++. Technical Report Cs-tr-992. University of Newcastle.

Liskov, B. y Zilles, S. (1975). Specification Techniques for Data Abstraction. IEEE Transactions on Software Engineering 1(1), pp. 7-18.

Meataute, P. y Serna, A.A. (2015). Métricas del software - Herramientas de apoyo cuantificables para la toma de decisiones. En Serna, M.E. (Ed,), Avances en Ingeniería (pp. 237-245). Medellín: Editorial Instituto Antioqueño de Investigación.

Meyer, B. (1985). On Formalism in Specification. IEEE Software 2(1), pp. 6-26.

Naur, P. (1969). Proofs of algorithms by General Snapshots. BIT 6, pp. 310-316.

Neumann, P. (1989). Flaws in specifications and what to do about them. Proceedings Fifth International Workshop on Software Specification and Design (pp. xi-xv). Pittsburgh, USA:

O'Regan, G. (2006). Mathematical Approaches to Software Quality. London: Springer.

O'Regan, G. (2012). A brief history of computing. London: Springer.

Parnas, D. (1972). A Technique for Software Module Specification with Examples. Communications of the ACM 15(5), pp. 330-336.

Parnas, D. (1977). The Use of Precise Specifications in the Development of Software. Proceedings IFIP Congress (pp. 849-867). Toronto, Canada.

Penny, D., Holt, R. y Godfrey, M. (2005). Formal Specification in Metamorphic Programming. Lecture Notes in Computer Science 551, pp. 11-30

Pnueli, A. (1977). The Temporal Logics of Programs. Proceedings 18th IEEE Symposium on Foundations of Computer Science (pp. 46-57). Rhode Island, USA.

Randell, B. (1973). The Origin of Digital Computers. Berlin: Springer.

Ryan, A. (2010). Formal specification of moving block railway interlocking using CASL. BSc Dissertation. Swansea University.

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

Serna, M.E. (2013). Manifiesto por la Profesionalización del Desarrollo de Software. Medellín: Editorial Instituto Antioqueño de Investigación.

Serna, M.E. y Serna A.A. (2014). Formal specification in context: Current and future. Ingeniare. Revista chilena de ingeniería 22(2), pp. 243-256.

Serna, M.E. y Serna, A.A. (2014a). Methodology for perform reliable literature reviews. Revista Información, cultura y sociedad. In press.

Souquières, J. y Levy, N. (1993). Description of specification developments. Proceedings First IEEE Symposium on Requirements Engineering (pp. 216-223). San Diego, USA.

Spivey, J. (1989). The Z Notation. London: Prentice Hall.

Spivey, J. (2002). An introduction to Z and formal specifications. Software Engineering Journal 4(1), pp. 40-50.

van der Poll, J. y Kotzé, P. (2002). What Design Heuristics May Enhance the Utility of a Formal Specification? Proceedings 2002 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on Enablement through Technology (pp. 179-194).

Vinter, R., Loomes, M. y Kornbrot, D. (1998). Applying Software Metrics to Formal Specifications: A Cognitive Approach. Proceedings 5th International Symposium on Software Metrics (pp. 216-223). Bethesda, USA.

Wing, J. (1990). A specifier's introduction to formal methods. IEEE Computer 23(9), pp. 8-23.

EFILIA ARIZA
Efilia Ariza Vargas
Estudiante

Elizabeth Ariza
Elizabeth Ariza Vargas
Estudiante

Prof. Edgar Serna M.
Asesor

FECHA ENTREGA: _____

_____
Comité trabajos de grado de la Facultad

ACEPTADO____ RECHAZADO____ ACEPTADO CON MODIFICACIONES_____

ACTA NO._____

FECHA ENTREGA: _____

_____
Consejo de Facultad

ACTA NO._____

FECHA ENTREGA: _____