“Aproximación a los Métodos Formales: una experiencia con un método de desarrollo liviano.”

Fernando Aguirre, Edgardo Belloni

Dirección de Investigación y Desarrollo - Departamento de Ingeniería y Cs. de la Producción - Universidad Gastón Dachary

Campus Urbano. Av. Lopez y Planes y Av. Jauretche. Posadas. Misiones

Tel. (0376) 447-6999

faguirre@ugd.edu.ar, ebelloni@ugd.edu.ar

RESUMEN

El presente trabajo tiene como objetivo determinar cuáles son las bondades de la implementación de técnicas formales para el desarrollo de software, enfocándose en los métodos denominados semi formales, ya que éstos suponen una aplicación más al "alcance" de la industria, debido a su menor complejidad y costo. Para ello dos equipos desarrollarán un mismo componente de software. Uno de los cuales utilizará una técnica de desarrollo semi formal y se compararán los resultados, basados en la medición de las métricas previamente definidas.

Se utilizará la herramienta de Microsoft Spec#, y el lenguaje de programación C# de la Suite Visual Studio .Net.

CONTEXTO

El presente proyecto se desarrolla en el ámbito de la Dirección de Investigación y Desarrollo de la Universidad Gastón Dachary, con el aporte del Departamento de Ingeniería y Cs. de la Producción.

El proyecto fue presentado y aprobado en la convocatoria de Proyectos de Investigación del año 2.012, que fuera realizada por la mencionada Dirección.

El proyecto además se encuentra en consonancia con el objetivo de la tesis en realización de uno de los investigadores, para la obtención del título de la Maestría en Ingeniería de Software, dictada en la Universidad de La Plata, cuyo título es "Un aporte empírico al desarrollo formal liviano en base a Spec#"

INTRODUCCIÓN

¿Cómo aseguramos que nuestros programas sean correctos y libres de errores?

A diferencia de otras disciplinas formales, la Ingeniería de Software no ha

desarrollado técnicas precisas para el modelado de los objetos a construir, antes que estos fueren efectivamente desarrollados. No existen actualmente, o al menos no ampliamente difundidos, técnicas tales como los modelos matemáticos, que permiten predecir el resultado de construcciones del mundo real con exactitud.

Ahora entonces, veamos con que mecanismos cuenta la Ingeniería del Software para evitar construcciones sin errores:

Lamentablemente todas las aproximaciones existentes, distan bastante de ser soluciones exactas. “No existe la bala de plata” [F. Brooks 87] reza Brooks, dado que el modo con que cuentan los informáticos para respaldar la confianza de los programas es la verificación empírica [JuristoMoreno 06]. Con lo cual, el principal problema consiste en la imposibilidad de garantizar la no existencia errores en una aplicación, dado que depende primero de la porción de programa que se esté probando y del grado de dependencia que tenga con otras porciones de programa (acoplamiento), y segundo, del grado cobertura de casos del universo de datos de pruebas suministrado.

Por lo tanto, usualmente se realiza la detección y corrección de errores en simultáneo con la construcción de la aplicación en sí. Tal actividad requiere un esfuerzo igual o superior al esfuerzo de la programación propiamente dicha.

Muchos errores están relacionados con la complejidad inherente del software, tales como errores en la definición del dominio, en la especificación de los flujos de trabajos, etc., para lo cual se requiere soluciones orientadas a atacar las dificultades esenciales del software.

Ahora bien, los demás tipos de errores, que son los que están más cercanos a las denominadas complejidades no esenciales, mayormente inherentes al trabajo de programación en sí, son los más atacados por las técnicas de validación y verificación de software.

Pretender generar una técnica de validación y verificación del software con una efectividad cercana al 100% hoy en día, constituye una empresa con un éxito altamente improbable, sin embargo, las líneas de investigación actuales consisten en generar aportes para la reducción del esfuerzo en las pruebas y lograr la conversión paulatina de las técnicas manuales y artesanales de validación a procesos cada vez más automáticos.

Verificación formal de software.

Los métodos formales en para verificación de software constituyen el intento de la Ingeniería de Software de parecerse a las demás ingenierías, proveyendo herramientas matemáticas para la detección y corrección de errores.

Constituye la principal referencia dentro de las técnicas de verificación estáticas. Implican una especificación formal del sistema y un análisis matemático detallado de la especificación y pueden desarrollar argumentos formales para que un programa se ajuste a su especificación formal. [HuthRyan 04]

Spec#, un aporte al desarrollo de software de alta calidad.

El sistema de programación Spec# provee un conjunto de herramientas para la verificación formal del software. Su principal ventaja es

que permite al programador abstraerse de las complejas especificaciones matemáticas, y por ello se considera como un sistema formal "liviano".

Se fundamenta conceptualmente en la lógica de Hoare [Hoare 69], y es una extensión del lenguaje de programación C#, creado por Microsoft e incluido por primera vez en la Suite de Desarrollo Visual Studio 2005.

Spec# se encuentra en plena etapa de investigación, con algunos resultados prometedores. Consiste en estructuras de especificación tipo pre y pos condición, tipos "no nulos" y algunas otras funcionalidades para abstracciones de datos de alto nivel. [Microsoft 08]

LÍNEAS DE INVESTIGACIÓN Y DESARROLLO

Esta línea de investigación se encuadra dentro de los estudios relacionados a la aplicación de los métodos formales en la industria de software.

Se centra en la aplicación de un método formal *semi liviano* para la construcción de un componente de software y cuáles son sus efectos en el desarrollo de software. Para ello se trabajará con distintos grupos de desarrolladores que implementarán las especificaciones del componente, por un lado utilizando una metodología tradicional y por otro lado, incluyendo técnicas de verificación semi formal.

Existen otros trabajos relacionados, como [R. Khun 98], el cual aporta un análisis del costo de implementación de los métodos formales, haciendo referencia a que los métodos formales pueden ser aplicados productivamente mediante técnicas formales livianas, analizando especificaciones parciales de software o inclusive en los requerimientos, de forma temprana.

La línea principal de trabajo se centra en la aplicación real de técnicas formales livianas, utilizando el paradigma Orientado a Aspectos, mediante la herramienta de Microsoft Spec#.

RESULTADOS OBTENIDOS/ESPERADOS

Se espera del proyecto: Aportar mayor evidencia empírica en la utilización de los métodos formales de verificación y validación de software con el objetivo de ratificar las bondades que brindan, aún teniendo en cuenta las limitaciones asociadas a que Spec# todavía se encuentra en etapas tempranas de evolución.

A la vez, se espera también aportar mecanismos y/o procedimientos, que contribuyan a facilitar y/optimizar la implementación de dicha herramienta colaborando en la difusión y masificación del uso de la misma.

FORMACIÓN DE RECURSOS HUMANOS

La estructura del proyecto está principalmente conformada por dos docentes de la UGD, uno de los cuales se encuentra desarrollando la tesis de la Maestría en Ingeniería de Software de la Universidad Nacional de La Plata, que desarrolla su trabajo en el ámbito del proyecto, contribuyendo de esta manera a su formación académica.

El equipo del proyecto se complementa con la incorporación de Auxiliares de Investigación.

Se incorporarán al proyecto 1 (un) Auxiliar de Investigación de 1ª, quien preferentemente se encontrará realizando su tesis para la obtención de título de grado y 4 (cuatro) Auxiliares de Investigación de 2ª.

Se pretende que los demás auxiliares generen sus tesis de grado en el ámbito del proyecto.

REFERENCIAS

[F. Brooks 87] F. Brooks. *No Silver Bullet, Essence and Accidents of Software Engineering*. Computer Magazine of University of North Carolina at Chapel Hill. Abril 1987.

[JuristoMoreno 06] N. Juristo, A. Moreno, A. Vegas. *Técnicas de Evaluación de Software*. Universidad Politécnica de Madrid. Octubre 2006.

[HuthRyan 04] M. Huth and M. Ryan. *Logic in Computer Science.* Modeling and Reasoning about Systems. Cambridge University Press. 2004.

[Hoare 69] Hoare, C. A. R. *An axiomatic basis for computer programming.* Communications of the ACM, 12(10):576?585, October 1969.

[Microsoft 08] Microsoft Research. *Program Verification Using the Spec# Programming System.* ETAPS Tutorial. Marzo 2008.

[Wooodcock 99]. Woodcock, P. Gorm Larsen, J. Bicarregui, J. Fitzgerald., *Formal Methods: Practice and Experiencie.* ACM Computer Survay 41(4). 2009.

[R. Kuhn 03] R. Kuhn, D. Craigen, M. Saaltink. *Practical Application of Formal Methods in Modeling and Simulation*. 2003

[JacksonWing96] D. Jackson and J. Wing, *Lightweight Formal Methods* IEEE Computer, Abril 1996, pp. 21-22

[M. Barnet 11]M. Barnett, M. Fahndrich, K. Rustan, M. Leino, P. Muller, W. Schukte, H. Venter, *Specification and Verification: The Spec# Experience*. 2011

[R. Khun 98] R. Kuhn, R. Chandramouli, R. Butler. *Cost Effective Use of Formal Methods in Verification and Validation*. 1998