Lenguaje relacional para la verificación de software

Relational language software for verifying

Germán Huanca
Instituto de Investigaciones en Informática
Carrera de Informática
Facultad de Ciencias Puras y Naturales
Universidad Mayor de San Andrés
La Paz - Bolivia
Autor de correspondencia: germanh_4@hotmail.com

Resumen

A medida que aumenta la complejidad de los sistemas de software, surgen nuevos aspectos desarrollo de aplicaciones, que hasta ese momento no se habían tenido en cuenta, al menos de una forma explícita. De este modo, el proceso de desarrollo se ha ido convirtiendo gradualmente en una labor de ingeniería, en la que nuevas tareas, como la elaboración de especificaciones, el diseño del sistema, la construcción de prototipos, la integración y pruebas, la gestión de la configuración y otras muchas han ido cobrando importancia frente a las labores de programación que eran las que primaban en un inicio. La Ingeniería de Software ha ido respondiendo a esta situación con el desarrollo de nuevos modelos, notaciones, técnicas y métodos. El presente trabajo propone una alternativa para la verificación formal de sistemas permitiendo demostrar con certeza que el sistema desarrollado realiza las funciones requeridas, es decir, cumple con ciertas especificaciones.

Palabras clave: especificación, ingeniería, software, sistemas, verificación.

Abstract

With increasing complexity of new software systems application development aspects that until then had not been taken into account, at least one explicitly arise. Thus, the development process has been gradually becoming a work of engineering, where new tasks such as the development of specifications, system design, prototyping, integration and testing, management configuration and many others have been gain in importance with the work program which was initially prevailed. Software Engineering has responded to this situation with the development of new models, notations, techniques and methods. This paper proposes an alternative form of systems for allowing verification prove with certainty that the developed system performs the required functions, i.e. meets certain specifications.

Keywords*: specification, engineering, software, systems, verification.*

Introducción

La verificación formal de sistemas de software es una disciplina relativamente antigua en las ciencias de la computación; su antecedente más destacado se remonta a los trabajos de Floyd, Hoare y Dijkstra sobre verificación de programas. El enfoque general de los métodos formales es abstraer los sistemas en modelos matemáticos, establecer la especificación del sistema deseado en términos de propiedades matemáticas que se deben cumplir y luego demostrar formalmente que los modelos producidos verifican dichas propiedades. Es claro que los factores antes mencionados afectan la inserción actual de los métodos formales en la industria del desarrollo de software si se tiene en cuenta que los objetivos primarios de la Ingeniería de Software son la mejora de la calidad de los productos de software y el incremento de la productividad de los ingenieros de software. Para que la aplicación de métodos formales en la industria se generalice, es necesario desarrollar herramientas que permitan la verificación automática de las propiedades de un sistema, o la traducción automática de un diseño dado de una forma a otra más adecuada para su verificación formal.

Métodos

Se inició el tratamiento del proyecto, con la revisión de herramientas que fueron proporcionados en el área de los métodos formales para el desarrollo y verificación de sistemas de software, notando que, aunque se han realizado importantes avances, aun no se ha producido una transferencia efectiva de dichos resultados a la industria. Por otro lado muchos métodos formales han sido desarrollados como una alternativa a las prácticas de Ingeniería de Software actuales.

A partir de la revisión que se hizo se pudo evidenciar varias herramientas formales que apoyan al desarrollo de sistemas de software, tales como *Uni ForM Workbench, Auto Focus*y *Concurrency Factory*. A partir de estas experiencias se empezó a plantear aspectos, que nos permitan fundamentar las especificaciones formales necesarias para la aplicación de estos métodos y sean escritas en un lenguaje accesible a diseñadores y programadores. Es por ello se considera que la utilización de un lenguaje gráfico facilitará la integración de los métodos formales a las prácticas industriales.

En una segunda etapa en el desarrollo del proyecto, se centró en determinar un punto de análisis y verificación de los sistemas de software, notando que ese punto viene a ser la arquitectura y los componentes de un sistema de software. En vista que los sistemas de software han tenido arquitectura desde que el primer programa fue dividido en módulos, y los programadores han sido responsables de establecer las interacciones entre dichos módulos y lograr así determinadas propiedades globales para sus sistemas.

De este modo, la descripción de los aspectos arquitectónicos del software ha estado tradicionalmente limitada al uso de ciertas expresiones, tales como arquitectura cliente/servidor, arquitectura en capas, etc., por lo general acompañadas con diagramas informales. Estas descripciones carecen de un significado preciso, lo que limita de manera drástica su utilidad, impidiendo, por ejemplo, cualquier análisis de las propiedades de los sistemas así descritos.

Por lo tanto para el cumplimiento de nuestro objetivo, en primer lugar, consideramos a la arquitectura de los sistemas de software para elevar el nivel de abstracción, facilitando la comprensión de

los sistemas de software complejos. En segundo lugar, a través de esa abstracción se aumentan las posibilidades de reutilizar tanto la arquitectura como los componentes que aparecen en ella. Por último, la notación utilizada tiene una base formal adecuada, es posible analizar la arquitectura del sistema, determinando cuáles son sus propiedades aún antes de construirlo.

En este sentido, el dominio identificado a través de la arquitectura de un sistema software, asociamos al modelo *Kripke*(modelo que tiene como base y fundamento en la lógica hibrida, denominativo que se da a la fusión entre lógica modal y proposicional) permitiéndonos describir por medio de un conjunto de fórmulas bien formadas las propiedades que se desea que el sistema cumpla.

Finalmente, a través del algebra relacional (que es el fondo de este trabajo de investigación) al aplicar en el denominado programa de la lógica algebraica, permite traducir los problemas lógicos en cuestiones algebraicas, para luego usar la 'maquinaria' del algebra universal para resolver el problema algebraicamente y trasladar la solución otra vez a la lógica original, de esta forma la verificación del sistema software se hace más efectiva. Complementariamente al trabajo, otro elemento que surge con el modelo planteado para la verificación de software, es que esta se plasma en un documento del sistema software que se trata, el cual de acuerdo a las normas estándares ISO/IEC, la documentación que surge en cualquier proceso de desarrollo de software, es un factor preponderante para que este cumpla con los requerimientos o especificaciones del mismo.

Resultados

El método de verificación de sistemas que se propone en el trabajo muestra algunas características interesantes: una es la utilización de una notación a la vez estándar y flexible para la especificación del diseño del sistema, esta característica hace que nuestro método sea de fácil aprendizaje por parte de los programadores y diseñadores del sector industrial, permitiendo al mismo tiempo su inclusión en distintas etapas del ciclo de vida del software como parte de metodologías de desarrollo actualmente en uso.

Discusión

- La experiencia de la comunidad de métodos formales indica que para que estos métodos sean aplicados en el desarrollo industrial de software se debe proveer de herramientas automáticas que soporten el método. En nuestro caso el uso del sistema de manipulación de relaciones como el *RELVIEW*.
- Por otro lado la limitación del método es la de referirse solamente a propiedades estáticas del diseño de sistema de software, no brinda la posibilidad de expresar y verificar propiedades de los componentes dinámicos de un sistema tal como se requeriría para la completa descripción de una arquitectura de software.

Conclusiones

El método de verificación formal de propiedades de diseños de sistemas de software propuesto en este trabajo es fácil de utilizar, automatizable y pasible de ser incorporado a distintas metodologías existentes, formales o informales, de desarrollo de software; para esta afirmación se consideró algunos sistemas de

información que como ejemplo se utilizan académicamente en las materias de sistemas de la carrera de Informática.

Aunque limitado a verificar las propiedades estructurales estáticas del diseño su mayor bondad respecto de los algoritmos de verificación de modelos implementados en lógicas y la posibilidad de ser ampliado a propiedades dinámicas, hacen de este método una interesante herramienta que debería ser testeada en un ambiente de desarrollo de software, y sus posibles intensiones analizadas en subsecuentes trabajos de investigación.

Un aspecto que no hemos desarrollado en este trabajo ha sido el filtrado que realiza el método propuesto. Utilizando un conjunto de fórmulas modales es posible obtener información relevante del sistema modelado, filtrando la información que no se requiere para el análisis que se está efectuando. Es natural pensar que existe una forma equivalente de filtrado utilizando el cálculo relacional, aunque no hemos abordado esa tarea aun. Este mecanismo resulta interesante ya que nos permite obtener distintas vistas de la estructura del sistema y posiblemente sea de utilidad en la etapa de diseño y análisis del sistema

Referencias

Loeckx, J. & Sieber, K. (1984). "*The Foundations of Program Verification*", Wile Teubner series in Computer Science.

Manna, Z. & Pnueli, A. (1995) "*Temporal Verification of Reactive Systems: Safety*", Springer-Verlag.

Medel, R. H. (2000). "*Utilizando RELVIEW para la verificación de especificaciones en Lógica Modal*", en *Anales del WAIT '99-Workshop* Argentino de Informática Teórica, págs. 93-106, Buenos Aires.

Medel, R. H. y Baum, G. A. (1999) "*Aplicación de Algebras Forken la verificación automática de sistemas especificados en Lógica Modal*", en *Anales del CACIC'99*

Orlowska, E. (1989). "*Relational Formalisation of Non-classical Logics*", en págs. 91-106.

Parnas, D. L., (1972). "*On the Criteria to beused in Decomposing Systems in Modules*", en *Communications of the ACM*.

Presentado: La Paz, 9 de octubre de 2015

Aceptado: La Paz, 27 de noviembre de 2015