Un Método para la Especificación de Requisitos

Pedro Rossel Cid
Departamento de Ingeniería Informática y Cs. de la Computación
Universidad de Concepción
Concepción, Chile
e-mail : prossel@inf.udec.cl

Resumen: En este paper se discutirá la importancia de la etapa de especificación de requisitos en el ciclo de vida del software. Además se mostrarán las dos grandes formas de realizar especificaciones: la formal y la informal, desde un punto de vista de sus características más relevantes y de costo. Con ello se concluye la necesidad de poseer métodos de especificación que sean fáciles de aprender y trabajar, pero que tengan un soporte formal que los haga poderoso. De esta manera se pretende exponer la idea de generar un nuevo método para especificar requisitos que cumpla con las características anteriormente dichas.

1. Antecedentes

Desde la llamada "crisis del software", por allá finalizando los años '60, se han creado varios modelos de desarrollo de software, tales como el **modelo cascada** y el **modelo espiral**, entre otros. Estos modelos tienen dentro de sus etapas la *especificación de requisitos*, primera etapa dentro del ciclo de vida del software. Según [Berzins91] el costo de corregir un error de requerimiento una vez que ha sido entregado el software, es cien veces mayor que si el mismo error es corregido en la etapa de especificación de requerimientos.

Debería ser natural suponer que esta etapa se realizará con el mayor cuidado posible, y utilizando los métodos que nos aseguren una buena realización de la misma.

Obviando toda la etapa de entrevistas con los usuarios, y pensando que se tienen los requisitos del software a construir, surge la inquietud de plasmar estos requisitos de alguna manera tal que la representación de éstos sea *clara*, *completa*, *no ambigua* y *precisa* [Fraser91].

Según [Fraser91] las consecuencias de tener requerimientos inadecuados incluye:

- rechazo al sistema
- instalación de un sistema peligroso
- excesiva mantención del sistema
- falla del sistema
- pérdida de credibilidad hacia el diseñador

2. Estado del Arte

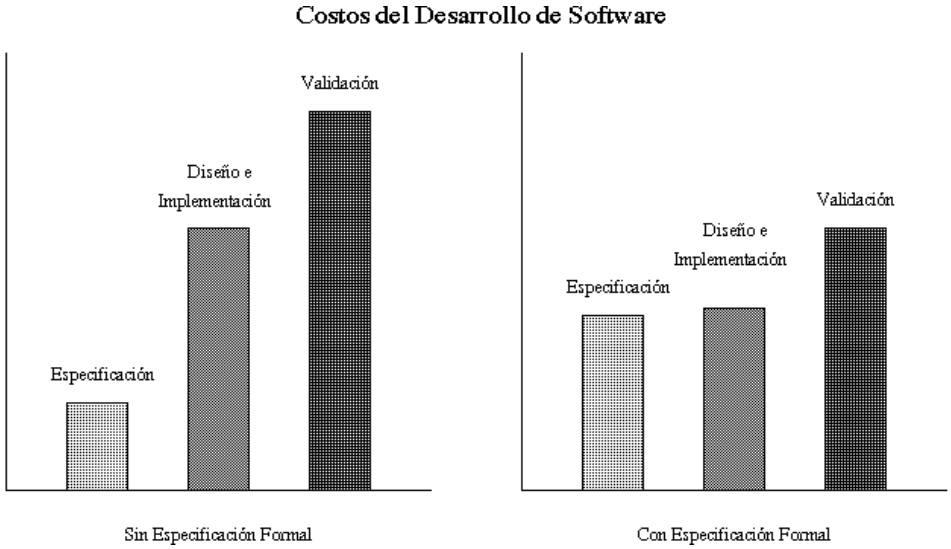
Existen variadas formas de especificar requisitos. Una división amplia las cataloga en **técnicas de especificación formal e informal**. Las formales están compuestas normalmente por una sintaxis, una semántica y un conjunto de relaciones, y los informales no.

Dentro de los formales los más conocidos son **VDM** y **Z**. Dentro de los informales **Análisis Estructurado**, **Warnier** y **Análisis Orientado a Objetos**.

Obviamente estas dos categorizaciones tiene ventajas y desventajas, las que desde cierto punto de vistas son complementarias. Las especificaciones informales son más convenientes para la obtención de requerimientos, son más fáciles de aprender y comunicar; en cambio los lenguajes formales proveen mayor exactitud, claridad y precisión, y es más conveniente para análisis y verificación.

Se debe recordar que los lenguajes informales usan una combinación de gráfica y gramática textual semiformal, para describir y especificar requerimientos; en cambio los formales tienen una base matemática, usualmente lógica formal. Por estos motivos los primeros son más fáciles de usar que los segundos, pero así mismo son más débiles en cumplir con las expectativas de *claridad*, *completitud*, *no ambigüedad* y *precisión*.

Desde un punto de vista de costos, [Sommerville92] proporciona los siguientes gráficos:



Claramente se ve que la especificación informal es menos costosa que la formal, pero esto lleva asociado un mayor costo en las etapas posteriores del ciclo de vida del software. De manera inversa, las especificaciones formales son más costosas que las informales, pero el costo de las etapas posteriores es menor que en el primer caso. Luego una aspiración válida es mantener los costos lo más bajo posible, pero conservando ciertas características que nos aseguren un buen producto final.

Existe un trabajo realizado por [Fraser91] el que permite desarrollar especificaciones desde el **Análisis Estructurado** (Yourdon) a **VDM** y viceversa. De manera similar, [Liu93] expone su método, similar al anterior, pero basándose en el análisis estructurado de De Marco. Estos son aportes significativos, pero no son más que una forma de ir de un lado a otro (aun

cuando tenga base formal). Luego tal vez sería bueno desarrollar un lenguaje sustentable en sí mismo, con componentes formales e informales.

3. Propuesta

La generación de una solución apropiada al problema planteado, se encuentra inserta en la preparación de una tesis de grado.

La idea básica para poder resolver el problema consiste en recopilar las características o aspectos relevantes de algunos lenguajes formales como por ejemplo VDM, Z, LARCH, que permitan asegurar *claridad*, *completitud*, *no ambigüedad* y *precisión*. No será necesario evaluar una gran cantidad de lenguajes formales, sino que sólo algunos, que se elegirán como los más representativos. Con estas características se generará un lenguaje formal para la especificación de requisitos, el que además tendrá una representación gráfica.

No se debe olvidar que los requisitos se deben poder entregar a los usuarios, para validarlos o para posteriores revisiones, y esta actividad involucra a gente que no necesariamente tiene amplios conocimientos en el área matemática, base de las especificaciones formales. Luego es mucho más fácil tener metodologías gráficas, pero con un soporte formal, que permitan una buena interacción usuario-diseñador y que asegure las características enunciadas en el párrafo anterior. De esta forma el método servirá también para clarificar y obtener requerimientos.

Se ve la necesidad de tener un forma de representar los requisitos, que sea canónica (estándar) intermedia entre lo formal y lo informal, esto debido a la aparente poca preocupación por parte de la comunidad a generar métodos fáciles de usar para las especificaciones. Tal vez se ha dado más importancia a las etapas posteriores del ciclo de vida del software, aun cuando la etapa de especificación de requisitos es tanto o más importante que éstas.

4. Características Deseables de un Lenguaje Formal

Los lenguajes formales poseen características como no *ambiguos* y *precisos*, pero éstas son demasiado generales como para saber qué camino seguir para alcanzarlas.

Según [Tse91], son tres las características que todo lenguaje formal debiera proveer:

I. Abstracción del mundo real

Un lenguaje de especificación de requerimientos es un medio mediante el cual los usuarios pueden realizar un modelo del mundo real, y pueden especificar sus problemas y requerimientos. Es el vínculo entre desarrolladores y usuarios. Así, debe asegurar que todos los involucrados en el proceso de especificación puedan entenderse.

a) Conocimiento por parte del usuario del lenguaje de especificación: Los usuarios deben conocer el lenguaje de especificación, o por lo menos tener familiaridad con él. Es por esto que un nuevo lenguaje de especificación de requerimientos debe usar elementos conocidos, para disminuir el trabajo de aprendizaje.

b)Estilo del lenguaje: El estilo del lenguaje puede ser un de los tres siguientes:

1. Lenguaje textual
2. Lenguaje gráfico
3. Lenguaje híbrido (textual y gráfico)

Unos se pueden adaptar mejor a ciertos dominios de aplicación, y eso es algo que se debe evaluar.

c) Complejidad: Se debe proveer de un medio para asegurar la claridad conceptual de los problemas a especificar. Esto se puede lograr separando las características lógicas de las físicas y proveyendo varios niveles de abstracción.

d) Modificabilidad: La especificación de requerimientos debe ser estructurada de tal forma que sea fácilmente modificada para las nuevas necesidades de los usuarios, cambios en la tecnología y/o cambios en el ambiente.

II. Manipulación de representaciones

a) Herramientas para la Manipulación: Se hace necesario proveer dos cosas: *formalismo* y *rigurosidad*. La primera para eliminar ambigüedades durante la construcción e implementación del sistema, que además aseguren una única notación, y la segunda para probar la correctitud de la implementación.

b) Transformación: Se debe lograr un formalismo que sea transparente para los usuarios finales, de tal manera que tanto usuarios como analistas entiendan lo mismo. Además se debe considerar la posibilidad que ciertos lenguajes de especificaciones pueden ser mejores que otros dado cierto dominio de la aplicación.

c) Independencia del Diseño y la Implementación: El soporte del lenguaje debe ser orientado al modelo y a lo no procedural. En otras palabras se debe contestar el "qué hacer" y no el "cómo hacerlo".

III. Construcción de un sistema real

Los requerimientos deben ser posibles de llevarse a un diseño y a una implementación. Esto asegura que las operaciones en el sistema final (construido) sean atribuibles a los requerimientos realizados en la especificación original.

De esta forma, teniendo estas características, debiera resultar más fácil completar con éxito la propuesta expuesta en este trabajo.

5. Conclusiones

El resultado que se espera obtener es un método formal apoyado con representación gráfica, que permita de una manera más simple especificar requisitos, con la consecuente disminución de costos que podría involucrar una especificación formal. Este método será una

representación formal reducida, en primera instancia, no por lo cual será menos válida, sino que tomará algunas características deseables de las especificaciones formales. Esta representación podrá ser ampliada en futuras investigaciones.

Por otro lado para llegar a buen término se hace necesario restringir el dominio de aplicación para el cual el método a desarrollar servirá. Por ejemplo, el método podría no considerar problemas de **tiempo real**, que son un poco más difíciles de especificar. Por último, para saber si el método generado es mejor que otros, se deberán realizar pruebas con casos reales previamente documentados.

6. Referencias y Bibliografía

[Berzins91] : Software Engineering with Abstractions
Berzins and Luqi
Addison-Wesley Publishing Company
1991

[Fraser91] : Informal and Formal Requirements Specification Languages: Bridging the Gap
Martin D. Fraser, Kuldeep Kumar, Vijay K. Vaishnavi
IEEE Transactions on Software Engineering
VOL. 17, NO. 5, MAY 1991

[Liu93] : A Formal Requirements Specification Method Based on Data Flow Analysis
Shaoying Liu
The Journal of Systems and Software
VOL 21 NO 2, MAY 01 1993

[Pressman93] : Ingeniería de Software. Un Enfoque Práctico
Roger S. Pressman
McGraw-Hill
Tercera edición, 1993

[Sommerville92] : Software Engineering
Ian Somerville
Addison-Wesley Publishing Company
Cuarta edición, 1992

[Serrano95] : Ambiente de Requisitos
Maria Amélia B. Serrano, Arndt von Staa.
PUC-Rio InfMCCC 36/95. Reporte Técnico
1995

[Tsai92] : A Hybrid Knowledge Representation as a Basis of Requirement Specification and Specification Analysis
Jeffrey J. P. Tsai, Thomas Weigert, Hung-Chin Jang
IEEE Transactions on Software Engineering
VOL. 18, NO. 12, DECEMBER 1992

[Tse91] : An Examination of Requirements Specification Languages
T. H. Tse, L. Pong
The Computer Journal
VOL. 43, NO. 2, 1991