La tecnología orientada a objetos y la ingeniería de software ante
la complejidad inherente al software

Como sugiere Brooks. “la complejidad del software es una propiedad esencial, no accidental”[1]. La complejidad de los sistemas informáticos hace a veces necesario el desarrollo de proyectos software de decenas de miles de líneas de código. Esto no puede ser abordado directamente, empezando a programar sin más. Es necesario analizar qué es lo que tenemos que hacer, cómo lo vamos a hacer, cómo se van a coordinar todas las personas que van a intervenir en el proyecto y cómo vamos a controlar el desarrollo del mismo de forma que al final obtengamos los resultados esperados. Las metodologías convencionales de Ingeniería de Software tienen mecanismos robustos para hacer un análisis de necesidades y diseño de los sistemas, poco han evolucionado con la tecnología en lo relacionado con el diseño computacional. Este trabajo propone la inclusión de la tecnología orientada a objetos, en todas las etapas del ciclo de desarrollo del sistema, para disminuir la complejidad. Al llegar a la implementación, los resultados obtenidos se transcriben al lenguaje de programación elegido, cambiando la sintaxis en que se expresa el modelo, mas no la semántica.

Introducción.

La situación actual en los sistemas informáticos se caracteriza por una rápida evolución de los componentes del hardware, que incrementan continuamente su potencial e incluso disminuyen sus precios, junto con una fuerte tendencia a la estandarización (computadoras personales, estaciones de trabajo con sistema operativo UNIX, sistemas distribuidos funcionando sobre plataformas heterogéneas, etc.). Hoy en día existe una gran diversidad de marcas y modelos con atributos y precios similares. En este escenario, el potencial de las grandes computadoras de las décadas pasadas está hoy disponible en una minicomputadora e incluso en una computadora personal. El software es el mecanismo que nos permite utilizar y explotar este potencial. Esto hace que, a la hora de plantearnos la adquisición de un sistema informático completo, ya sea para administrar una empresa, para controlar un proceso industrial, o para uso doméstico, el software es lo que marca la diferencia . El desarrollo de software no es una tarea fácil, su complejidad inherente se deriva de cuatro elementos: la complejidad del dominio del problema, la dificultad de administrar el proceso de desarrollo, la flexibilidad que se puede alcanzar a través del software y los problemas que plantea la caracterización del comportamiento de sistemas discretos [2].

El desarrollo de software es una actividad muy reciente (apenas tiene 50 años), comparada con otras actividades de ingeniería (vgr. la ingeniería civil o incluso la ingeniería eléctrica). Es aún más reciente la Ingeniería de Software, disciplina que se encarga de establecer un método en el desarrollo de sistemas. Existen métodos de desarrollo de software como el clásico, espiral, cascada, etc., sin embargo, en los últimos años la tecnología orientada a objetos se ha desarrollado en diferentes segmentos de la ciencia de la computación como un medio para manejar la complejidad inherente a los sistemas de muy diversos tipos. La pregunta es , ¿Cómo puede ayudar la tecnología orientada a objetos a disminuir la complejidad inherente al software?.

Complejidad del dominio del problema

Cuando los problemas del mundo real se desean resolver con modelos de sistemas computacionales, trae consigo una cantidad indefinida de requisitos que compiten entre sí y algunas veces se contradicen. Dar funcionalidad a un sistema es difícil e incluso comprender los requerimientos como: facilidad de uso, rendimiento, costo, capacidad de supervivencia, fiabilidad, son parte de la complejidad externa que infiere determinantemente en la complejidad interna del sistema.

Bajo este contexto nace la importancia de la relación entre desarrolladores y usuarios del sistema. Habitualmente los usuarios suelen tener dificultades en expresar sus necesidades e ideas. Esto se da en oca-

siones por falta de conocimiento del ámbito de cada uno de los grupos. Los usuarios y los desarrolladores tienen perspectivas diferentes sobre la naturaleza del problema. Contar con herramientas que permitan plasmar estos requisitos de forma clara para ambos grupos es uno de los retos de la Ingeniería de Software.

Dificultad de administrar el proceso de desarrollo

Es difícil contar con un sistema donde su característica sea la simplicidad en el desarrollo del software. La mayoría de los grandes sistemas contienen un alto número de código que impide dar un mantenimiento óptimo a los programas . Cuando los equipos de desarrollo son grandes y heterogéneos la administración de las actividades se hace complicada debido a que se desarrollan por muchos equipos de trabajo.

Hoy en día, tanto el hardware como el software, deben ser capaces de contar con una arquitectura abierta, reusabilidad, facilidad de reconfiguración, aplicación de estandares y diseño modular. Estas características permiten escribir menos códigos y poder reutilizar el software, en un software diferente al que fue diseñado originalmente. La adición debe ser transparente para el sistema receptor.

La flexibilidad que se puede alcanzar a través del software.

El software ofrece la flexibilidad al desarrollador para expresar y representar procesos triviales y complejos del conocimiento humano en un sistema computacional. Esta propiedad no debe producir cambios en los procedimientos y prácticas de las entidades. El sistema debe ser suficientemente flexible para incorporarse a las actividades de las entidades y hacerlas más eficientes, sin causar gastos que eleven el precio de los sistemas.

Por ejemplo, una organización puede tener una diversidad de sistemas de comunicación en su estructura organizacional, el sistema computacional de apoyo a la administración debe ser lo suficientemente flexible para trabajar y/o compatibilizar con cualquier sistema de comunicaciones (transmisión de datos) existente. Por lo tanto, el software no debe forzar al cliente a comprar sistemas de comunicaciones adicionales para poder operar sin problemas.

Los problemas de caracterizar el comportamiento de sistemas discretos

En una aplicación de gran tamaño puede haber cientos o hasta miles de variables, así como más de un flujo de control. El conjunto de todas estas variables, sus valores actuales, y la administración del sistema son procesos que constituyen el estado actual de la aplicación. Al ejecutarse el software en computadoras digitales, se tiene un sistema con estados discretos. En contraste, a los sistemas analógicos (vgr. el movimiento de una pelota lanzada al aire) son sistemas continuos. Parnas sugiere que “cuando se afirma que un sistema se describe con una función continua, quiere decir que no puede contener sorpresas ocultas. Pequeños cambios en las entradas siempre producirán cambios consecuentemente pequeños en las salidas”[3]. Por el contrario, los sistemas discretos por su propia naturaleza tienen un número finito de estados posibles. Todos los eventos externos a un sistema de software tienen la posibilidad de llevar a ese sistema a un nuevo estado, y más aún, la transición de estado a estado no siempre es determinista. En las peores circunstancias, un evento externo puede corromper el estado del sistema, porque sus diseñadores olvidaron tener en cuenta ciertas interacciones entre eventos.

La ingeniería de software y la tecnología orientada a objetos

Dijkstra sugiere, “La técnica de dominar la complejidad se conoce desde tiempos remotos: divide et impera (divide y vencerás)”[4]. Cuando se diseña un sistema complejo de software, es esencial descomponerlo en partes más y más pequeñas, cada una de las cuales se puede refinar de forma independiente. Para entender un nivel dado de un sistema, basta con comprender unas pocas partes a la vez, la descomposición del todo en sus partes permite disminuir la complejidad inherente al software.

El enfoque de orientación por objetos (O.O.) es una tecnología que también cubre el ciclo de vida del software y que permite tener un acercamiento al mundo que se modela y cómo funciona este mundo. En algunas metodologías de la Ingeniería de software (IS) se habla del análisis, diseño e implementación como tres procesos independientes cuya mezcla tiene como resultado final una aplicación que satisface una necesidad. El problema de complejidad inherente al software

y el trato independiente de las fases que componen la metodología de desarrollo de software disminuyen la eficiencia del sistema. El lenguaje que se maneja, los alcances y el resultado final de cada una de las fases del ciclo de vida puede afectar el objetivo final, si se tratan en forma independiente.

El enfoque O.O. puede modelar el mundo que se desea modelar en términos de los objetos que se posee. Cada uno de ellos tiene sus propias características que lo identifican y un comportamiento específico. Con la base en las características y comportamiento del objeto se puede obtener una mejor calidad a lo largo del ciclo de vida de una aplicación, facilitando además el mantenimiento y la creación de nuevas versiones que actualicen el programa.

La tecnología orientada a objeto enriquece a la ingeniería del software

Al disminuir las barreras entre las etapas de análisis y desarrollo, se garantiza que se está hablando de las mismas cosas y en los mismos términos desde el comienzo del análisis hasta el final de la etapa de implementación. Esto evita inconsistencias y permite verificar que las cosas están claramente definidas y cumplen con todos los requerimientos, incluso antes de escribir una línea de código del programa.

Las características de los objetos, como encapsulamiento, herencia, reutilización permiten crear un software mucho más robusto. El hecho de modelar el mundo y no únicamente de datos necesarios para determinada aplicación, permiten crear diversas aplicaciones sobre la misma información sin repetir los procesos de análisis de los mismos. Esto ofrece la posibilidad de dedicarse a cumplir con los requerimientos de la aplicación basándose en las facilidades que ofrecen los objetos del mundo ya modelado.

Beneficios aplicando la tecnología orientada a objetos al ciclo de vida del software.

Se pueden enumerar varios beneficios de la tecnología orientada a objetos, como metodología en el ciclo de vida del desarrollo del software:

*Reutilización del Software***:** Permite describir clases y objetos que podrán ser usados en otras aplicaciones.

*Estabilidad***:** El diseñador piensa en términos de comportamiento de objetos, no en detalles de bajo nivel.

Diseño rápido y de alta calidad: Puesto que se concentra en satisfacer los requerimientos y no en detalles técnicos.

Integridad: Facilidad de programación al usar efectivamente toda la información de la fase de diseño, poniéndola en términos de un lenguaje específico.

Facilidad de mantenimiento: Dado que al tener el modelo del mundo, es fácil realizar mantenimiento en términos de objetos, atributos y métodos de los mismos.

Independencia: Con la tecnología O.O. en el análisis y diseño del software, permite que las plataformas, el software y el hardware sean independientes.

Evolución de la tecnología orientada a objetos (TOO)

El desarrollo de la TOO, es la evolución más importante de los años noventa en la práctica de la ingeniería de software. UML (Unified Modeling Languaje) se ha convertido recientemente en el estándar de la industria para el diseño de software orientado a objetos. La notación UML se deriva y unifica las tres metodologías de análisis y diseño OO más aceptadas:

Metodología de *Grady Booch* para la descripción de conjuntos de objetos y sus relaciones.

Técnica de modelado orientada a objetos de *James Rumbaugh* (OMT: **O**bject-**M**odeling **T**echnique).

Aproximación de *Ivar Jacobson* (OOSE: **O**bject-**O**riented **S**oftware **E**ngineering) mediante la metodología de casos de uso (*use case*).

En 1997 UML fue aprobada por la OMG[1] convirtiéndose en la notación estándar para el análisis y diseño orientado a objetos. UML es el primer método en publicar un meta modelo en su propia notación, incluyendo la notación para la mayoría de la información de requisitos, análisis y diseño. Se trata de un meta modelo autoreferencial.

1 El OMG (Object Management Group) se formó en 1989 con el propósito de crear una arquitectura estándar para objetos, busca el desarrollo de especificaciones para la industria del *software* que sean técnicamente excelentes.

Conclusiones

La tecnología orientada a objetos ofrece múltiples ventajas al desarrollo del software en general, ventajas que podemos aplicar al ciclo de vida del software. Una correcta jerarquía de clases permite obtener software más fácilmente ampliable y reutilizable, además, una jerarquía de clases coherente puede suponer una gran claridad desde etapas tan tempranas como el análisis y el diseño. El uso de esta tecnología permite definir objetos lógicos , interfaces entre aplicaciones y funciones actualizadas o nuevas o incluso definidas por el usuario, sin necesidad de recompilar la totalidad del sistema.

La integración de entornos permite la construcción de un entorno mayor a partir de otros más pequeños que ofrecen su funcionalidad a través de interfaces de comunicación. Además la encapsulación nos permite exportar de alguna forma, de una clase a otra aplicación sin interdependencias, mientras que con el paradigma clásico, esto requeriría de una gran labor de reingeniería.

Referencias

1. Brooks, F. April 1987. No Silver Bullet: Essence and Accidents of Software Engineering. IEEE Computer vol. 20 (4), p. 12.
2. Booch, Grady, Análisis y Diseño Orientado a Objetos. Addison Wesley Longman 2da. Ed. Massachusetts, E.U.A (1998)
3. Parnas, D. July 1985. Software Aspects of Strategic Defense Systems. Victoria, Canada: University of Victoria, report DCS-47-IR.
4. Dijkstra, E. 1979. Programming Considered as a Human activity. Classics in Software Engineering. New York, NY.

Links

Http://www.well.com/
Http://www.projtech.com/info/smmethod.html
Http://www-gris.ait.uvigo.es/~avilas/UML/
http://www.manizales.unal.edu.co/

Olivia Allende Hernández
Profesora Investigadora de la
Universidad Tecnológica de la Mixteca