INF-236
Análisis y Diseño de Software
Apuntes de clase

Índice general

Índice de figuras

Prefacio

Estos son mis apuntes del ramo INF-236 de la USM. En pocas palabras, es un resumen de todo el curso incluyendo todo el material (de contenido y ejercicios) a los que tuve acceso mientras confeccionaba este documento. No se pretende que estos apuntes reemplacen al material oficial (aunque con gusto agradezco que puedas pensar eso). Si lees esto, prepárate para un compendio de todo lo que vas a enfrentar. Enjoy!

Lectura 2: The State of Practice in Model-Driven Engineering

Los inicios del **MDA (Model-driven architecture)** se ven en el 2001 por parte del OMG (Object Managamente Group) con el propósito de hacer ver a los modelos como una prioridad dentro del desarrollo de software, además de promover que siempre sean lo suficientemente adecuados en términos de automatización para evolucionar sin problemas en el transcurso de vida del proyecto.

De la discusión sobre el MDA, nace el concepto de **Model-driven Engineering (MDE)**. Por definición, **un modelo es una abstracción de un sistema en ejecución**. Durante harto tiempo no hubo una escena clara de lo que debía hacerse (por ejemplo, se hablaba más de los **MDD (Model-Driven Development)**, que se enfoca en la implementación de los modelos en sí). El MDE busca que los modelos utilizados permitan apoyar directamente al desarrollo (algo así como reverse engineering) durante todo el proceso.

Para comprender cuál MDE vale la pena seguir, es necesario conocer **en qué fallaron y en qué fueron exitosos**.

Uno de los factores de éxito de los MDE es **cómo manejan la particularidad**. Empresas reportaron mejores resultados al elegir **lenguajes de dominio especializados (DSLs)** en su rubro antes que alternativas más generales con los lenguajes más robustos o de general-purpose como UML. Es más sencillo implementar un DSL para el entendimiento del modelo, aunque quizás no es tan fácil de implementar al no ser estandarizado.

Otro factor que indica un buen MDE es **la base de donde empieza**. La idea de un MDE no es luchar directamente para representar todo el sistema sino indicar cuáles son los primeros pasos para levantar el desarrollo. También se puede usar de otras formas, como usarlo en conjunto con algún MDD para flexibilizar el entendimiento.

Un detalle que se puede omitir de manera superficial es que **un MDE NO representa necesariamente una generación de código a diferencia de, por ejemplo, un MDD**. Dado que n o necesariamente están relacionados, la creación e implementación de un MDE puede tener costos de baja productividad por naturaleza, aunque por sus beneficios vale la pena incluirlos.

Estos beneficios reales de un MDE son holísticos. Esto significa que la gracia de un MDE radica en su capacidad de descripción y de englobar al dominio como un entre sí. Un MDE puede ser explícito, lo que significa que el equipo de desarrollo no deberá esforzarse en buscar herramientas sino en adaptarse a las que ya tienen disponibles especialmente si se hace en una etapa temprana. Esta imposición, lejos de ser forzada, es bastante orgánica porque da paso a la reutilización de herramientas, como código u otros patrones que se pueden modelar mediante un DSL.

Un MDE funciona para situaciones particulares. Hablando en términos genéricos, un MDE no es una solución factible por los costos que conlleva. Un dominio preciso y bien acotado se acoplará mucho mejor con un MDE porque permite un diseño claro que no se podría representar de otra manera.

Un MDE también se adapta mejor a ciertos miembros de un equipo de desarrollo. Los arquitectos de software pueden encontrar un mejor control al representar y guiar un desarrollo desde un MDE y, con ello, justificar y elegir mejor las decisiones de diseño del proyecto. Por otro lado, hay otros miembros que no simpatizan muy bien con el concepto. Un ejemplo son los “gurús de código”, que son básicamente los que se centran en resolver problemas complicados del área o aquellos que les gusta libremente aprender sobre nuevas tecnologías de manera constante. No es que realmente sea una amenaza para ellos, sino es una manera de limitarlos para seguir normas. Otros miembros pueden ser los administradores de tiempo y no programadores como tal.

Uno de los cuellos de botella de un MDE es que todo el equipo de desarrollo se adapte a este. Esto ocurre porque no todos tienen la capacidad de pensar de manera abstracta al nivel que espera el modelo. Ojo, no es determinante y ocurre generalmente en equipos inmaduros en el uso de MDE.

Una de las características que debe manejar el "gurú del MDE" es la abstracción. Un buen diseñador debe conocer el dominio al dedillo, pues es el MDE se maneja a muy alto nivel.

Hay ciertas directrices que se pueden seguir para garantizar el éxito de un MDE:

- **Mantener los dominios claros y bien acotados**.
- **Poner el MDE como prioridad a seguir en proyectos que sí lo necesitan**.
- **Tener en cuenta las ganancias en otros aspectos (como la productividad o la generación de código)**.
- **La mayoría de proyectos falla al escalar.**
- **No obsesionarse con la generación de código.**

A diferenncia de lo que se imparte en las universidades, el enfoque de un MDE debe estar orientado con un enfoque **Bottom-Up** en vez de **Top-Down**. Esto significa que un buen MDE nacerá desde la concepción de un equipo de desarrollo pequeño y será propagado al resto de la organización, y no al revés. Esto se hace así para facilitar la abstracción y promover las habilidades que comúnmente no se enseñan a la hora de mostrar código.

En síntesis, la única manera de manejar la complejidad y la abstracción de un problema es uniendo tres áreas de estudio: **Modelado de Software, Diseño y estudio de Software y estudio de organizaciones**.