Portafolio Semana 3

Autor: Vicente Fuenzalida Marín
Fecha de Entrega: 25 de agosto de 2017
Tema: Uso del lenguaje de modelado de sistemas UML

Uso de UML en proyectos de software

Artefacto

El artefacto que escogí esta semana es una entrada de Quora donde se pregunta si las compañías exitosas utilizan regularmente UML.
En particular extraigo las dos primeras respuestas, de Cris Fuhrman y Charles Rivet. Se adjunta una copia del artefacto en formato PDF en el repositorio de Git.
Las respuestas que se brindan intentan resolver si un desarrollador necesitará o no saber/utilizar UML en su trabajo. Esto variará según el tipo de trabajo (o cargo a desempeñar) que tendrá la persona dentro de la empresa.

Como primer diferenciador, Fuhrman señala que UML es visto como un requisito en cargos de tipo *senior* más que desarrollador, debido a que permite una mayor abstracción y comprensión del diseño, dejando de lado los 'detalles tediosos' del código.
Además, él señala que el fenómeno de la *comoditización* del software impulsa a las empresas a buscar programadores que 'completen el trabajo' más que garantizar un software bien diseñado, mantenible o escalable, lo que tiene por consecuencia que se bajen los estándares requeridos para un ingeniero.
Por último, menciona que debido a las exigencias del mercado, al tiempo que toma y a la falta de herramientas útiles, un programador se ve inclinado a evadir el uso de UML.

El otro usuario que comenta, Rivet, da crédito a UML2 diciéndo que se ha vuelto una herramienta más *poderosa* y formal, pero a la vez mucho más compleja.
Rivet hace hincapié en que el uso de UML dependerá de la complejidad y el tamaño de la organización que está desarrollando software, comentando que para empresas más pequeñas que adoptan metodologías ágiles resulta más cómodo adoptar solo algunas partes de UML que respondan mejor a sus necesidades.
Por el contrario, afirma que para el desarrollo de sistemas complejos y de gran tamaño (software + hardware), el uso de UML básicamente 'se paga solo' y aumenta el nivel de abstracción drásticamente. Menciona que él, como desarrollador, ha visto casos en que el modelo mismo (apoyado por UML) es más valioso que el código generado, pero que sólo se utilizó UML debido a que los retornos del proyecto lo admitían (se deduce que el costo de software para modelar en UML es bastante alto).
Finalmente plantea lo siguiente:

> "...if your code and your UML model differ, it is useless and it is usually costly to maintain. Is UML relevant? Yes, but not for everyone and not for every project."

Reflexión

Basado en los comentarios de estos dos desarrolladores de software, lo primero que concluyo es que el uso de UML depende fuertemente del contexto de desarrollo en que este se enmarque, esto es:

- Tamaño del proyecto
- Complejidad del proyecto
- Metodologías de desarrollo utilizadas (ágil, incremental, cascada, ...)
- Estructura y tamaño de la empresa
- Plazos de entrega (cortos, flexibles, largos)
- Costos y presupuesto para el proyecto (para poder costear herramientas modeladoras en UML)

Así, no resulta nada sencillo definir si conviene o no utilizar UML en el desarrollo de software, añadiéndo que los desarrolladores deben dominar las normas para definir diseños y saber qué diseños son los más adecuados para cada caso, ya que como se conversó en clases, los diagramas UML son utilizados por distintas entidades y por lo mismo se descomponen en *4+1* vistas arquitectónicas (lógica, de desarrollo, de proceso, física y de casos de uso).

En mi opinión particular, y basándome en el artefacto que escogí, me parece que los diagramas UML si pueden aportar información valiosa y representaciones más abstraídas para el uso de los distintos participantes en los proyectos de desarrollo de software, aunque también creo que su uso es absolutamente dependiente del tipo de proyecto y su envergadura.

Por lo anterior, utilizar UML me parece pertinente sólo cuando la complejidad supera a lo que un equipo pequeño puede manejar (framework Scrum por ejemplo) o cuando el proyecto de software es transversal a varios equipos de diseño/desarrollo que requieren comunicarse y cuyos artefactos de software son dependientes entre si, ya que agiliza la comprensión de los componentes y aporta un importante nivel de abstracción frente a complejidades innecesarias (a muchos desarrolladores no les importará cómo se realiza un cierto proceso o qué algoritmos utiliza, sino que simplemente el input/output del componente en cuestión).

A modo de cierre, quisiera señalar que si bien no creo que UML sea la mejor alternativa siempre, si permite comprender y explicar mejor el funcionamiento de un software, y como en la mayoría de los casos un proyecto está inmerso en un entorno comercial o al menos involucra a otras áreas/departamentos (financiero, social, etc.), siempre será una alternativa para que estos grupos externos puedan visualizar y comprender qué es lo que hay detras del producto desarrollado (con distintos niveles de profunidad).