No silver bullet - Essence and Accident in Software Engineering

Frederick P. Brooks, Jr.

Presentación de Emanuel Delgadillo, Igal Frid, Maximiliano Giusto, Santiago Pivetta y Sonia Stedile

7 de Noviembre de 2011

Sobre el autor

- Ingeniero de software y científico de la computación. Doctor en matemática aplicada
- Conocido por dirigir el desarrollo del sistema operativo OS/360 y después escribir honestamente sobre el proceso en su famoso libro *The Mythical Man-Month*: "Es una experiencia humillante el cometer un error de coste multimillonario, pero es también muy memorable."
- Ganador del premio Turing Award en 1999
- Observación famosa: "*Añadir personal a un proyecto retrasado lo retrasará aún más*"

Dificultades Esenciales (inherentes al software en sí mismo)

Complejidad La construcción del software es una de las tareas más complejas ya que no hay dos partes iguales. La complejidad es esencial. Remarca las diferencias con otras ciencias que para hacer un modelo simplifican fenómenos que no son esenciales.

Conformidad El software tiene complejidad arbitraria, impuesta por sistemas e instituciones humanas. Es el que debe ajustarse a otras interfaces, ya sea porque fue el último en aparecer o porque se considera como el más adaptable.

Maleabilidad El software siempre es objeto de cambio. El usuario no sabe lo que quiere. A medida que evoluciona el software se producen cambios inevitablemente.

No visibilidad Dado que el software es invisible, es muy complejo hacer un modelo. Para esquematizar el software se requiere hacer varios diagramas de distintas vistas, lo que hace complejo imaginarlo como una única cosa.

Dificultades Accidentales

Propias del problema de expresar el software en algún lenguaje de programación, usando alguna tecnología, teniendo en cuenta las limitaciones de tiempos del proyecto.

- Brooks reconoce que es el campo en el cual se ha avanzado más.
- Dentro de los avances, hay 3 que él considera como los más importantes.
  - High-level languages
  - Time-sharing
  - Unified programming enviroment

Falsas esperanzas...

- Dado el fuerte avance en los desarrollos técnicos en los últimos tiempos, Brooks sostiene que si bien no hay esperanzas de hallar lo que sería la "bala de plata" para terminar con el problema del desarrollo del software, si existen algunos avances prometedores pero que no dejan de ser incrementales.
- Menciona unas cuantas oportunidades que pueden aparecer pero termina concluyendo que no es muy probable que puedan solucionar el problema.
- Lenguajes de alto nivel, Programación orientada a objetos, Inteligencia artificial, Sistemas expertos, Programación automática, Programación gráfica, Verificación de programas, Entornos y herramientas, Estaciones de trabajo.

Enfoques prometedores

Brooks sostiene que hay varios enfoques que son prometedores para atacar la esencia del problema del software:

1. Comprar *vs* construir: Sostiene que es más barato comprar (si se puede) que construir el software y que se ahorra muchos problemas.
2. Utilización de prototipos: Como el cliente no sabe lo que quiere, hay que hacer prototipos y luego mejorarlos.
3. Desarrollo incremental (*Grow, not build, software*): Agregar más funcionalidades a medida que son ejecutadas, usadas y testeadas.
4. Grandes diseñadores: El diseño es un arte, existen grandes diseñadores, lo que hay que hacer es reconocerlos y fomentar su carrera.

Conclusiones

- Al leer el paper no se nota que tiene 20 años, todo lo que plantea está muy vigente.
- Los problemas planteados aún se encuentran abiertos, no hubo soluciones.
- En las metodologías ágiles se usan los prototipos que plantea.
- También es evidente que se tomó en cuenta la propuesta de desarrollo incremental.
- Una crítica que se le puede hacer es que sobreestima la dificultad de construir software en relación con otras actividades.