Esenciales y accidentales

Esenciales hace referencia a lo que hace que el desarrollo de software sea intrínsecamente difícil, siendo esto la unión de distintos conceptos. Es muy precisa y altamente detallada, lo que implica que es igual frente a distintas representaciones. Lo difícil es su especififcación, diseño y testeo de la idea que se tiene.

Cuatro características del software moderno:

- Complejidad: Gran tamaño. No se repiten conceptos por lo cual para su descripción se debe ser muy extenso. La complejidad se modela y los elementos esenciales deben ser modelados. La complejidad lleva a falta de comunicación y aumenta de manera no lineal a medida que aumenta el tamaño.

  Many of the classical problems of developing software products derived from this essential complexity and its nonlinear increased with size. From the complexity comes the difficulty of communication among team members, which leads to product flaws, cost overruns, schedule delays. From the complexity comes the difficulty of enumerating, much less understanding, all the possible states of the program, and from that comes the unreliability. From the complexity of the functions comes the difficulty of invoking those functions, which makes programs hard to use. From complexity of structure comes the difficulty of extending programs to new functions without creating side effects. From the complexity of structure comes the unvisualized state that that constitute security trapdoors.

  Not only technical problems but management problems as well come from the complexity. This complexity makes overview hard, thus impeding conceptual integrity. It makes it hard to find and control all the loose ends. It creates the tremendous learning and understanding burden that makes personnel turnover a disaster.

- Conformidad: Conformarse a complicaciones arbitrarias creadas por personas.
- Cambiable/Abierto al cambio: Su funcionalidad es modificable al ser un resultado del pensamiento. Se requieren nuevas funciones durante el uso, se descubre mejoras.
- Invisibilidad: Invisible y no visualizable. No es graficable en el espacio. Tenemos varios diagramas que muestran el mismo de distintos ángulos pero nunca uno solo muestra todo.

Soluciones a accidentales

- Lenguajes de alto nivel: Permite desligarse de la complejidad del hardware y su funcionamiento, abstrayéndose solamente a nivel lógico.
- Optimización de tiempos durante la programación: En vez de tener tiempos muertos, todo se realiza de manera que el tren de pensamiento se mantenga y no se frene.
- Ambientes de programación unificados: IDE. La unificación de librerías y herramientas de programación en un solo programa permitió optimizar tiempos durante la programación.

Esperanzas de la plata

ADA: Filosofía de modularización más allá de la herramienta en sí.
POO: Acercarse a representar el diseño sin meterse en información sintáctica que no aporta valor. Remueve accidentes de expresión del diseño, no soluciona complejidad.
IA: Se confunde el concepto con el uso de la tecnología para resolver un problema que solo se solucionaba utilizando la inteligencia humana.
Sistemas expertos: Motores de inferencia. Utilizando y explicando su razonamiento al usuario final. El verdadero poder viene de mejores bases de datos que demuestren la realidad, no de mejores mecanismos de inferencia. El problema de estos motores es que se necesita de expertos capaces de desarmar su conocimiento para explicarlo por partes y expresarlo para la inferencia. Su mayor contribución es achicar el puente entre un experto programador y uno novato.
Programación automática: Inferencia mejorada a partir de tener técnicas disponibles. Se debería describir el método de solución, no tanto el problema.
Programación gráfica: Programar con cajitas. Imposible porque falta espacio para sistemas complejos.
Verificación del programa:
Ambiente y herramientas:
Estaciones de trabajo: Más hardware.

Buenos ataques en la esencia conceptual

Comprar en vez de hacer
Refinamiento de requerimientos y prototipado rápido: Se acerca en fases tempranas a algo que el cliente pueda ver y que permita darse cuenta de lo que realmente va a querer en el sistema, porque inicialmente el cliente nunca sabe lo que quiere. Hacer real el concepto, para probar consistencia y usabilidad.
Construcción no, crecimiento de software de manera incremental.
Mejores diseñadores: La gente es el centro del desarrollo de software.