The Computer Revolution Hasn't Happened Yet, OOPSLA 1997 (Alan C. Kay)

Tapicer, Krasny, Grünberg, Rozansky

Sobre Alan Kay

- Involucrado en el diseño inicial de Internet (ARPANET).

Sobre Alan Kay

- Involucrado en el diseño inicial de Internet (ARPANET).
- En 1970 entra a Xerox PARC donde se involucra en el diseño de Smalltalk.

Sobre Alan Kay

- Involucrado en el diseño inicial de Internet (ARPANET).
- En 1970 entra a Xerox PARC donde se involucra en el diseño de Smalltalk.

Hablar un poco sobre esto porque fue muy importante:

In 1970, Kay joined Xerox Corporation's Palo Alto Research Center, PARC. In the 1970s he was one of the key members there to develop prototypes of networked workstations using the programming language Smalltalk. These inventions were later commercialized by Apple Computer in their Lisa and Macintosh computers.

Kay is one of the fathers of the idea of object-oriented programming, which he named, along with some colleagues at PARC and predecessors at the Norwegian Computing Center. He conceived the Dynabook concept which defined the conceptual basics for laptop and tablet computers and E-books, and is the architect of the modern overlapping windowing graphical user interface (GUI). Because the Dynabook was conceived as an educational platform, Kay is considered to be one of the first researchers into mobile learning, and indeed, many features of the Dynabook concept have been adopted in the design of the One Laptop Per Child educational platform, with which Kay is actively involved.

Sobre Alan Kay

- Involucrado en el diseño inicial de Internet (ARPANET).
- En 1970 entra a Xerox PARC donde se involucra en el diseño de Smalltalk.
- Considerado uno de los padres de la POO (y acuñador del término).

Sobre Alan Kay

- Involucrado en el diseño inicial de Internet (ARPANET).
- En 1970 entra a Xerox PARC donde se involucra en el diseño de Smalltalk.
- Considerado uno de los padres de la POO (y acuñador del término).
- Turing Award en 2003 por sus contribuciones a la POO.

Sobre Alan Kay

- Involucrado en el diseño inicial de Internet (ARPANET).
- En 1970 entra a Xerox PARC donde se involucra en el diseño de Smalltalk.
- Considerado uno de los padres de la POO (y acuñador del término).
- Turing Award en 2003 por sus contribuciones a la POO.
- OOPSLA: Object-Oriented Programming, Systems, Languages & Applications.

Sobre Alan Kay

- Involucrado en el diseño inicial de Internet (ARPANET).
- En 1970 entra a Xerox PARC donde se involucra en el diseño de Smalltalk.
- Considerado uno de los padres de la POO (y acuñador del término).
- Turing Award en 2003 por sus contribuciones a la POO.
- OOPSLA: Object-Oriented Programming, Systems, Languages & Applications.
- En OOPSLA 1997 (aniversario 25 años de la creación de Smalltalk) da la conferencia "The Computer Revolution Hasn't Happened Yet" de la que vamos a hablar.

Creatividad

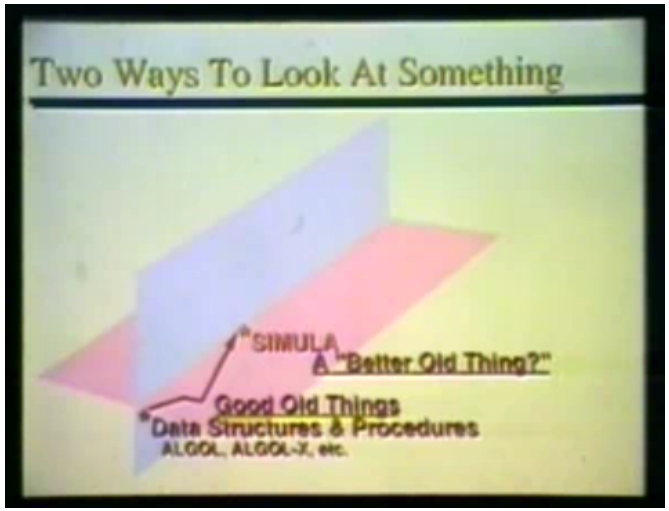
- Aprender es un acto de creatividad (The Act of Creation de Arthur Koestler).

- Aprender es un acto de creatividad (The Act of Creation de Arthur Koestler).

One of the great books he wrote was about what might creativity be. -Learning.- He realized that learning, of course, is an act of creation itself, because something happens in you that wasn't there before.

Creatividad

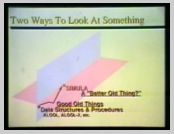
- Aprender es un acto de creatividad (The Act of Creation de Arthur Koestler).
- Salir del paradigma, ir del contexto rosa al celeste.



- Optimización vs Invención.

Creatividad

- Aprender es un acto de creatividad (The Act of Creation de Arthur Koestler).
- Salir del paradigma, ir del contexto rosa al celeste.



- Optimización vs Invención.

He used a metaphor of thoughts as ants crawling on a plane. In this case it's a pink plane, and there's a lot of things you can do on a pink plane. You can have goals. You can choose directions. You can move along. But you're basically in the pink context. It means that progress, in a fixed context, is almost always a form of optimization, because if you're actually coming up with something new, it wouldn't have been part of the rules or the context for what the pink plane is all about. Creative acts, generally, are ones that don't stay in the same context that they're in. He says, every once in a while, even though you have been taught carefully by parents and by school for many years, you have a blue idea.

Escalabilidad

- Analogía software (SOs) con cuchas de perro.

Cualquiera puede hacer una cucha de perro, pero si quiere hacer algo 100 veces más grande el mecanismo no es el mismo. Si igualmente lo hace termina con algo malo y eso le pasó a muchos SOs modernos (hace referencia a Windows que por el 97 estaba sacando la segunda versión de Windows 95 basada todavía en DOS).

Escalabilidad

- Analogía software (SOs) con cuchas de perro.
- Solución: arquitectura, saber combinar los materiales para darle al todo mayor integridad que las partes.

Solución: arquitectura, saber combinar materiales elementales para lograr la sinergia necesaria. El objetivo de POO es ayudar a organizar la arquitectura.

Escalabilidad

- Analogía software (SOs) con cuchas de perro.
- Solución: arquitectura, saber combinar los materiales para darle al todo mayor integridad que las partes.
- Una Catedral es más grande que el panteón y sin embargo usa menos materiales y posee arquitectura muy buena.

- Analogía software (SOs) con cuchas de perro.
- Solución: arquitectura, saber combinar los materiales para darle al todo mayor integridad que las partes.
- Una Catedral es más grande que el panteón y sin embargo usa menos materiales y posee arquitectura muy buena.

A non-obvious, a non-linear interaction between simple materials to give you non-obvious sinergies, and a fast multiplication of materials. It's quite remarkable to people when I tell them that the amount of material in [unintelligible] cathedral, which is an enormous, physical structure, is less than the amount of material that was put into the Parthenon. The reason is that it's almost all air, and almost all glass. Everything is cunningly organized in a beautiful structure to make the whole [hole/whole] have much more integrity than any of its parts. That's the other way you can go, and part of the message of OOP was, that, as complexity starts becoming more and more important, architecture's always going to dominate material, and in fact, the sad fact, I think, about OOP, is [that] people didn't get interested in architecture because of the beauty of it. They're only starting to get interested in architecture now, when the Internet is forcing everybody to do it. That's pretty pathetic.

Escalabilidad

- Analogía software (SOs) con cuchas de perro.
- Solución: arquitectura, saber combinar los materiales para darle al todo mayor integridad que las partes.
- Una Catedral es más grande que el panteón y sin embargo usa menos materiales y posee arquitectura muy buena.
- Procesos biológicos vs. Software ("la revolución de la computación no llegó aún").

Los procesos biológico escalan manejando información equivalente a varios terabytes, el software debería ser así ("la revolución de la computación no llegó aún"). Las cuchas de perros no escalan, los relojes (o las cosas mecánicas) no escalan, pero las células escalan en factores de millones de veces y siguen funcionando igual, eso todavía no lo puede reproducir la computación ("la revolución no llegó aún").

Escalabilidad

- Analogía software (SOs) con cuchas de perro.
- Solución: arquitectura, saber combinar los materiales para darle al todo mayor integridad que las partes.
- Una Catedral es más grande que el panteón y sin embargo usa menos materiales y posee arquitectura muy buena.
- Procesos biológicos vs. Software ("la revolución de la computación no llegó aún").
- Problema actual: lenguajes OO mezclados con procedurales.

- Analogía software (SOs) con cuchas de perro.
- Solución: arquitectura, saber combinar los materiales para darle al todo mayor integridad que las partes.
- Una Catedral es más grande que el panteón y sin embargo usa menos materiales y posee arquitectura muy buena.
- Procesos biológicos vs. Software ("la revolución de la computación no llegó aún").
- Problema actual: lenguajes OO mezclados con procedurales.

El problema es que en la actualidad lo que se encapsula como "objeto" reutilizable el la computadora, y no particionar el espacio de diseño. Ese es el problema principal de los lenguajes procedurales, y también en lenguajes modernos orientados a objetos como C++ y Java, que creen ayudar al programador pareciéndose a los viejos lenguajes procedurales, y eso le hace más difícil al programador ver las capacidades del nuevo paradigma.

Escalabilidad

- Analogía software (SOs) con cuchas de perro.
- Solución: arquitectura, saber combinar los materiales para darle al todo mayor integridad que las partes.
- Una Catedral es más grande que el panteón y sin embargo usa menos materiales y posee arquitectura muy buena.
- Procesos biológicos vs. Software ("la revolución de la computación no llegó aún").
- Problema actual: lenguajes OO mezclados con procedurales.
- Internet: arquitectura que evolucionó/escaló sin downtimes.

- Analogía software (SOs) con cuchas de perro.
- Solución: arquitectura, saber combinar los materiales para darle al todo mayor integridad que las partes.
- Una Catedral es más grande que el panteón y sin embargo usa menos materiales y posee arquitectura muy buena.
- Procesos biológicos vs. Software ("la revolución de la computación no llegó aún").
- Problema actual: lenguajes OO mezclados con procedurales.
- Internet: arquitectura que evolucionó/escaló sin downtimes.

Ejemplo de bueno diseño: estuvo involucrado en el diseño de ARPANET (lo que hoy es Internet) y nada del código original hoy en día se mantiene, el sistema creció muchísimo (en cientos de millones) y nunca dejó de funcionar para hacer los cambios.

Lenguajes meta-reflectivos

- Fuerza Aérea, 1961: problema con cintas de datos, una de las primeras ideas de abstracción en Software.

- Fuerza Aérea, 1961: problema con cintas de datos, una de las primeras ideas de abstracción en Software.

Idea de abstracción en software muy antigua: Fuerza Aérea, 1961, se transportaban muchas cintas con muchos datos diversos y en muchos formatos diferentes. Se implementó un mecanismo en el que cada cinta se dividía en 3 partes fijas: 3era parte: los datos en sí de un tipo; 2da parte: los procedimientos para manejar los datos de ese tipo/formato; 1era parte: punteros a los procedimientos para acceso directo (con los primeros tipos para operaciones estandarizadas). Para usar la cinta sólo hay que leer la parte de punteros y después hacer saltos indirectos.

Lenguajes meta-reflectivos

- Fuerza Aérea, 1961: problema con cintas de datos, una de las primeras ideas de abstracción en Software.
- Internet, HTML, formato con problemas

Lenguajes meta-reflectivos

- Fuerza Aérea, 1961: problema con cintas de datos, una de las primeras ideas de abstracción en Software.
- Internet, HTML, formato con problemas

HTML tiene el problema que tenían en la fuerza aérea: supone que el browser sabe cómo mostrarlo y no es autocontenido, por eso existe la guerra de los browsers. El sistema para mostrar HTML debería ser muy simple y autocontenido para evitar las incompatibilidades que hoy existen.

Lenguajes meta-reflectivos

- Fuerza Aérea, 1961: problema con cintas de datos, una de las primeras ideas de abstracción en Software.
- Internet, HTML, formato con problemas
- Los lenguajes deben ser capaces de autodefinirse, aspecto meta-reflectivo.

- Fuerza Aérea, 1961: problema con cintas de datos, una de las primeras ideas de abstracción en Software.
- Internet, HTML, formato con problemas
- Los lenguajes deben ser capaces de autodefinirse, aspecto meta-reflectivo.

Aspecto meta-reflectivo: en 1962 un libro sobre LISP describe en media página de código LISP un intérprete para LISP. Los lenguajes tienen que ser suficientemente expresivos para autodefinirse. Java no tiene ese capacidad, es demasiado complejo.

Lenguajes meta-reflectivos

- Fuerza Aérea, 1961: problema con cintas de datos, una de las primeras ideas de abstracción en Software.
- Internet, HTML, formato con problemas
- Los lenguajes deben ser capaces de autodefinirse, aspecto meta-reflectivo.
- Los lenguajes se deben poder construir sobre sí mismos y encapsularse a sí mismos, Squeak.

- Fuerza Aérea, 1961: problema con cintas de datos, una de las primeras ideas de abstracción en Software.
- Internet, HTML, formato con problemas
- Los lenguajes deben ser capaces de autodefinirse, aspecto meta-reflectivo.
- Los lenguajes se deben poder construir sobre sí mismos y encapsularse a sí mismos, Squeak.

Las implementaciones de los lenguajes pueden tener problemas de eficiencia que son encapsulados en el lenguaje (estructuras, etc.) para ser mejorados en futuras implementaciones. Eso se puede aplicar en la construcción del lenguaje mismo: “Cuanto más el lenguaje pueda ver sus propias estructuras, menos atado va a estar a la implementación”.

Acerca del crecimiento de Internet: va a aparecer la necesidad de interacción e intercambio de mensajes entre objetos, donde los sistemas puedan descubrir la “capacidad” de otros objetos de otros sistemas dinámicamente, interactuando entre sí (1997 predice que en 10 años van a aparecer cosas así, hoy en día hay algo parecido, “Web Services Discovery”, y otros).

Conclusiones

- Creatividad: “La mejor forma de predecir el futuro es inventarlo.” (Thinking outside the box).

Conclusiones

- Creatividad: “La mejor forma de predecir el futuro es inventarlo.” (Thinking outside the box).
- Escalabilidad: sobre la cucha de perro “Put more garbage on it, [...] say, Yes, we were really trying to do pyramids, not gothic cathedrals. That, in fact accounts for much of the structure of modern operating systems today.”

Conclusiones

- Creatividad: “La mejor forma de predecir el futuro es inventarlo.” (Thinking outside the box).
- Escalabilidad: sobre la cucha de perro “Put more garbage on it, [...] say, Yes, we were really trying to do pyramids, not gothic cathedrals. That, in fact accounts for much of the structure of modern operating systems today.”
- Lenguajes:
  - “No sé quién inventó el agua pero no fueron los peces”.
  - “Yo inventé el término *orientado a objetos* y puedo asegurarles que no tenía a algo como C++ en mente.”.

Conclusiones

- Creatividad: "La mejor forma de predecir el futuro es inventarlo." (Thinking outside the box).
- Escalabilidad: sobre la cucha de perro "Put more garbage on it, [...] say, Yes, we were really trying to do pyramids, not gothic cathedrals. That, in fact accounts for much of the structure of modern operating systems today."
- Lenguajes:
  - "No sé quién inventó el agua pero no fueron los peces".
  - "Yo inventé el término *orientado a objetos* y puedo asegurarles que no tenía a algo como C++ en mente.".

"No sé quién inventó el agua pero no fueron los peces", peces = nosotros, agua = nuestra estructura de creencias, nuestro contexto. Quiere decir que la forma en que pensamos viene impuesta por el contexto, es difícil salir del mundo de la programación procedural y pensar OO.
Tres etapas ante la aparación de una nueva idea según Schopenhauer: 1. se dice que son locuras, 2. se dice que siempre fue obvia, 3. los que la tuvieron dicen ser quiénes lo inventaron. Self-referencia por la POO, pero además sobre la resistencia a POO que había al principio.
Cuando Smalltalk estuvo desarrollándose en PARC durante 10 años cambiaba muy seguido y cada cosa en el lenguaje se construía sobre lo anterior (el lenguaje se construía sobre sí mismo).

La gente hace una religión de la POO cuando no saben realmente usarlo como deberían.

Conclusiones

- Creatividad: “La mejor forma de predecir el futuro es inventarlo.” (Thinking outside the box).
- Escalabilidad: sobre la cucha de perro “Put more garbage on it, [...] say, Yes, we were really trying to do pyramids, not gothic cathedrals. That, in fact accounts for much of the structure of modern operating systems today.”
- Lenguajes:
  - “No sé quién inventó el agua pero no fueron los peces”.
  - “Yo inventé el término *orientado a objetos* y puedo asegurarles que no tenía a algo como C++ en mente.”.
- Sobre su “pelea” con Dijkstra: “I don’t know how many of you have ever met Dijkstra, but you probably know that arrogance in computer science is measured in nano-Dijkstras.”

Conclusiones

- Creatividad: "La mejor forma de predecir el futuro es inventarlo." (Thinking outside the box).
- Escalabilidad: sobre la cucha de perro "Put more garbage on it, [...] say, Yes, we were really trying to do pyramids, not gothic cathedrals. That, in fact accounts for much of the structure of modern operating systems today."
- Lenguajes:
  - "No sé quién inventó el agua pero no fueron los peces".
  - "Yo inventé el término *orientado a objetos* y puedo asegurarles que no tenía a algo como C++ en mente.".
- Sobre su "pelea" con Dijkstra: "I don't know how many of you have ever met Dijkstra, but you probably know that arrogance in computer science is measured in nano-Dijkstras."

Oposición a Dijkstra porque trata a la computación como una ciencia pura y para Kay es más práctica.
Paper Dijkstra: On the fact that the Atlantic has two sides, sobre en el enfoque más matemático que se le da a cs. de la comp. en Europa que en USA. Paper refutación de Kay: On the fact that most of the software in the world is written on one side of the Atlantic (no se consigue, quizá no sea real), dice que cs. de la comp. no se puede adaptar a la matemática clásica. Dice que cs. de la comp. es una especie de matemática práctica, combinación de estructuras enormes y convencerse de haber cubierto todo los casos como en matemática.