

Desarrollo de Software Basado en Componentes

Diego Cardozo



github.com/diegocard

Agenda

1. Motivación
2. ¿Qué es el DSBC?
3. Componentes vs Objetos
4. Aplicaciones conocidas
5. ¿Como encajan los CMS?
6. Bajar a tierra conceptos
 - WebMatrix
 - OrchardCMS

Advertencia: Esta es una charla de Ingeniería de Software

Motivación (1)

Si General Motors evolucionara como la industria de la computación, hoy tendríamos autos que saldrían \$100 consumiendo 1 litro de combustible cada 1 millón de kilómetros.

Motivación (2)

Pero...

- Chocarías al menos 2 veces al día
- Comprar un auto nuevo cada vez que se cambia una señal de tránsito
- Cada vez que comprás un modelo nuevo, tenés que aprender a manejar nuevamente.

¿Conclusión?

- Otras industrias utilizan enfoques distintos
- Obtienen muy buenos resultados en áreas donde nosotros tenemos problemas.
 - Automotriz: procesos ágiles, producto complejo que se construye de forma rápida.
 - Construcción: tienen costos asociados al cambio mucho mayores que nosotros, pero se arreglan mejor.

¿Qué es lo que otras industrias tienen en común?

Trabajan utilizando componentes

- Mayor nivel de re-utilización
- Simplifica las pruebas
- Simplifica el mantenimiento
- Mayor calidad

Componentes desarrollados por terceros

- Ciclos de desarrollo mas cortos
- Mejor ROI

¿Que es un componente en SW?

"Un componente de software es una unidad de composicion con interfaces especificadas contractualmente y únicamente depnedencias contextuales explícitas.

Un componente de software puede ser desplegado de forma independiente y está sujeto a composición por otras partes."

Szyperski, 2002

7 Criterios

1. Puede ser usado por otros elementos de SW
2. Puede ser usado por los clientes sin la necesidad de intervención del desarrollador (CMS).
3. Incluye las especificaciones de todas las dependencias.
4. Incluye documentación de las funcionalidades que ofrece.
5. Se puede entender su funcionamiento en base a las especificaciones.
6. Se puede acoplar a otros componentes.
7. Puede ser incorporado a un sistema de manera suave y rápida.

Desventajas

- Clarividencia: diseñar un componente sin conocer quien lo utilizará.
- Particularización: es difícil "customizar" un componente sin acceder a su implementación.
- Falta de soporte: una vez que un componente es creado y sellado, ¿como y quien lo mantiene?

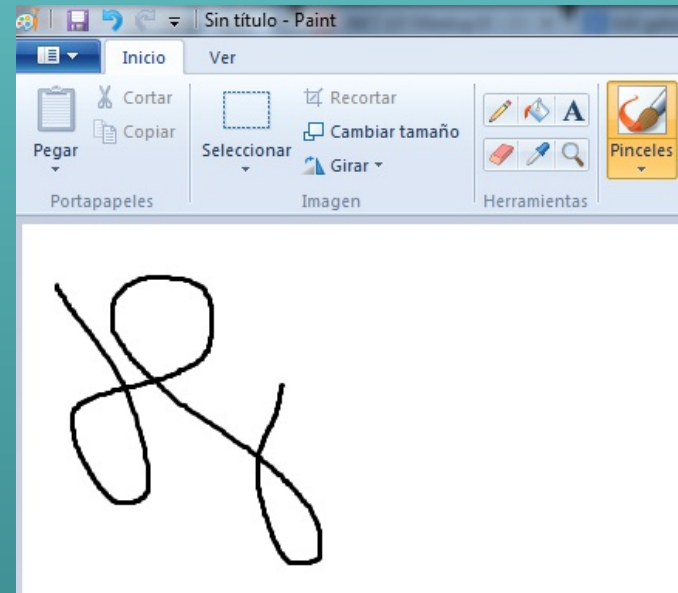
Componentes vs Objetos

	Objeto	Componente
Polimorfismo	Si	No
Instanciación	Algo tardía	Lo mas tarde posible
Encapsulación	Algo	Real y forzada
Herencia	Si	Por interfaz y reuso de binarios

Aplicaciones conocidas

- Bajo nivel: COM (Component Object Model)
 - Comunicación de procesos en distintos lenguajes
 - Precursores de .NET
- CORBA
 - Estándar creado por el OMG
 - Usado dentro del mundo Java
- Otras aplicaciones
 - Programación para diseño gráfico

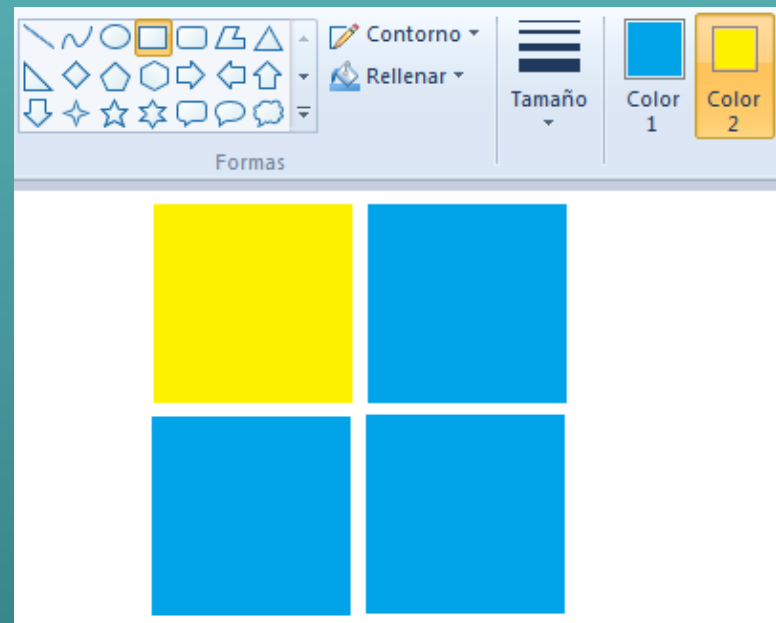
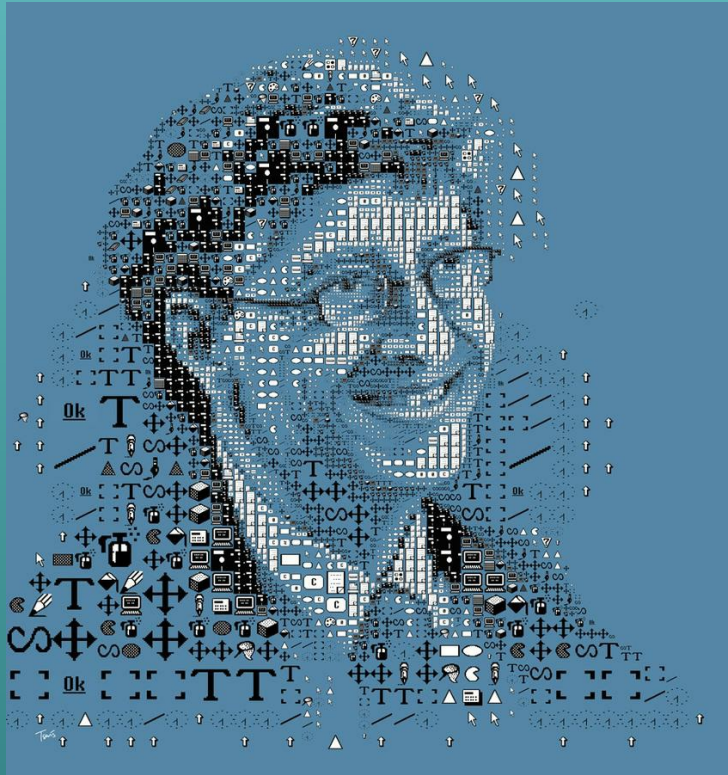
Modelo del pintor (1)



Modelo del pintor (2)

- Hay 2 técnicas utilizadas para crear imágenes
- Una de ellas es el método del pintor
- Se trata de tomar un elemento (pincel, lápiz) y utilizarlo para crear imágenes desde 0.
- Se utilizan trazos libres.
- Es el enfoque más natural para el ser humano.
- Yo creo que es la mejor analogía a como creamos software usualmente.
 1. Comenzamos con una hoja en blanco.
 2. Imaginamos el producto terminado.
 3. Implementamos la solución línea por línea.

Modelo de composición (1)



Modelo de composición (2)

- Se trata de crear imágenes a partir de elementos existentes.
- Por ejemplo: mosaico
- Ejemplo: dibujar un tablero de ajedrez.
- Yo creo que es el enfoque que debemos seguir al crear software.
 1. Tomar elementos conocidos y definidos.
 2. Unirlos de una forma inteligente.
 3. Crear solamente los componentes que faltan.
- En el mundo del software hay una enorme cantidad de componentes existentes.

Pintor vs composición

- Según el tipo de imagen, un método puede ser mas rápido que el otro.
- También se debe analizar cual consigue mejores resultados.
- Sin embargo, yo creo que en el mundo del software la mayoría de los problemas se asemejan más al modelo de composición.

Basta de filosofía...

Ejemplo concreto:

- Crear un sitio web para la comunidad de .NET Uruguay
- Incluir un foro

¿Por qué con un CMS?

- Se basan en la idea de DSBC.
- Tanto la estructura como el contenido son componentes
 - Páginas, imágenes, posts, widgets, módulos, etc.
- Cuentan con una comunidad y un catálogo de componentes existentes enorme.

Microsoft WebMatrix (1)

- Ambiente de desarrollo gratuito de Microsoft
- Liviano: 40MB recién instalado comparado con varios gigas de Visual Studio
- Pensado para la nube
- Soporta varios lenguajes y plataformas
 - ASP.NET
 - PHP
 - Node.js
- Integración con GIT y TFS

Microsoft WebMatrix (2)

- Visual Studio se siente como un laboratorio para crear
- WebMatrix es ideal para DSBC
 - Proporciona componentes:
 - Galería de frameworks
 - Incorpora NuGet package manager
 - Extensiones útiles
 - Aisla al framework de los componentes

OrchardCMS (1)

- Completamente Open Source
- Creciendo rápidamente
- Arquitectura MVC
- Eso significa que todos los componentes siguen una arquitectura MVC.
 - Crear nuevos componentes es sencillo
 - Los componentes existentes son fáciles de comprender y extender

Orchard CMS (2)

Componentes

- Contenido
- Módulo
 - Features
- Temas
- Templates
 - Formas (shapes)
- Widgets
- Usuarios, roles, permisos

Recursos

- Charla y código:
 - github.com/diegocard/CBSD-presentation
- DSBC en MSDN:
 - <http://msdn.microsoft.com/es-es/library/bb972268.aspx#ref07back>
- Excelente presentación relacionada:
 - <http://www.slideshare.net/ulicruz/desarrollo-de-software-basado-en-componentes>
- Paper
 - Component-Based Software Engineering – New Paradigm of Software Development (Crnkovic, Larsson)

Fin

"La Revolución Industrial del software está finalmente ante nosotros. La especialización de recursos, estándares para partes intercambiables, y herramientas de ensamblaje de última generación han sido usadas en otras industrias por cientos de años para acelerar el desarrollo de productos altamente complejos.

A pesar de su ubicuidad, la aplicación de estos conceptos a la industria moderna del software solamente ha empezando."

Bill Gates, 1997