Component Based Software Development

Diego Cardozo

github.com/diegocard

Agenda

1. Motivation
2. ¿What is CBSD?
3. Components vs Objects
4. Well known applications
5. ¿How do CMSs fit in?
6. Applying thseconcepts
    - WebMatrix
    - OrchardCMS

Warning: this talk is about Software Engineering

Motivation (1)

If GM had kept up with technology like the computer industry has, we would all be driving $25.00 cars that got 1,000 miles to the gallon.

Motivation (2)

But you would...

- Crash at least 2 times a day
- Have to buy a new car every time a new traffic signal is installed
- Have to learn to drive again every time you buy a new model

¿Conclusion?

- Other industries follow different approaches
- They get very good results in areas were we don't.
  - Car industry: agile process, complex product that is built rapidly.
  - Construction industry: much higher cost associated to change, but they anticipate it better.

¿What do other industries have in common?

They work using components

- Higher re-use rate
- Simplifies testing
- Simplifies maintenance
- Better overall quality

Components developed by third parties

- Shorter development cycle
- Better ROI

¿What is a SW component?

"A software component is a unit of composition with contractually specified interface and explicit context dependencies only.
A software component can be deployed independently and is subject to composition by third parts."

Szyperski, 2002

7 Criteria

1. Can be used by other SW components
2. Can be used by others without the intervention of the authoring developer (Like CMS).
3. Includes specifications for all of its dependencies.
4. Includes documentation of all the offered functionalities.
5. Its behavior can be understood from its specifications.
6. Can be coupled to other components.
7. Can be incorporated to a SW system rapidly and smoothly.

Disadvantages

- Clairvoyance: design a component without knowledge of its final user.
- Customization: it is hard to customize a component without access to its implementation.
- Lack of support: once a component is created and sealed, ¿who maintains it and how?

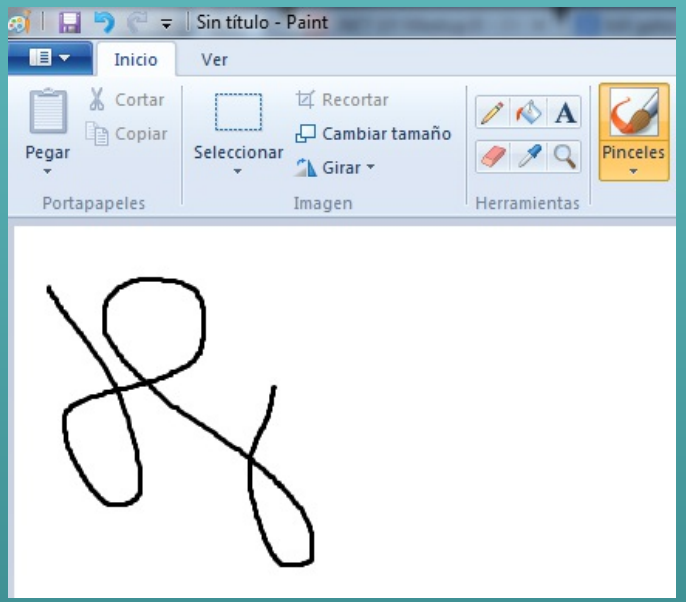
Objects vs Components

	Object	Component
Polymorphism	Yes	No
Instantiation	Late	As late as possible
Encapsulation	Sort of	Real and enforced
Inheritance	Si	Interface inheritance and binary reuse

Well known applications

- Low level: COM (Component Object Model)
  - Process communication in different languages
  - .NET precursor
- CORBA
  - Standard created by OMG
  - Widely used in the Java world
- Other applications
  - Programming for graphic design

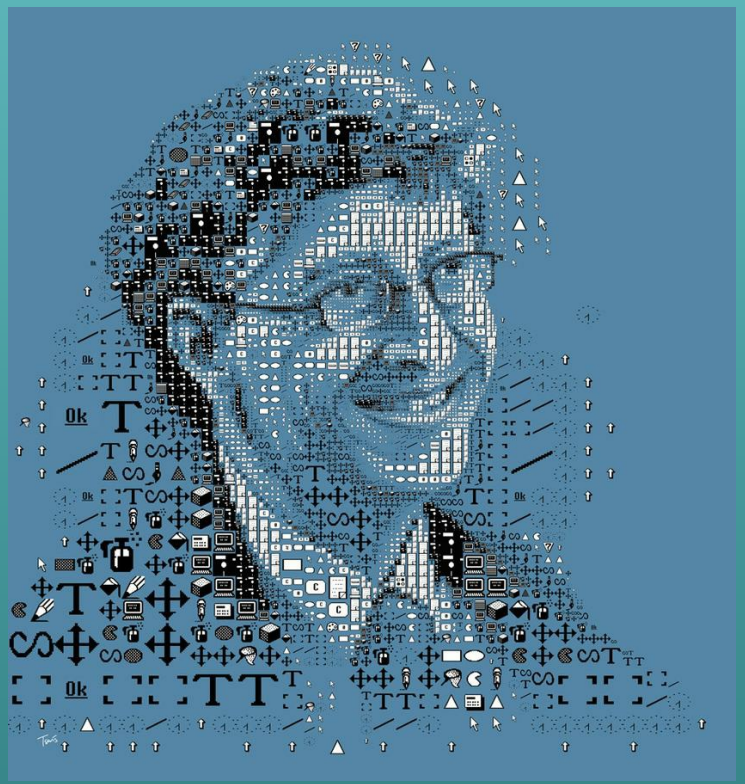
Painter model (1)

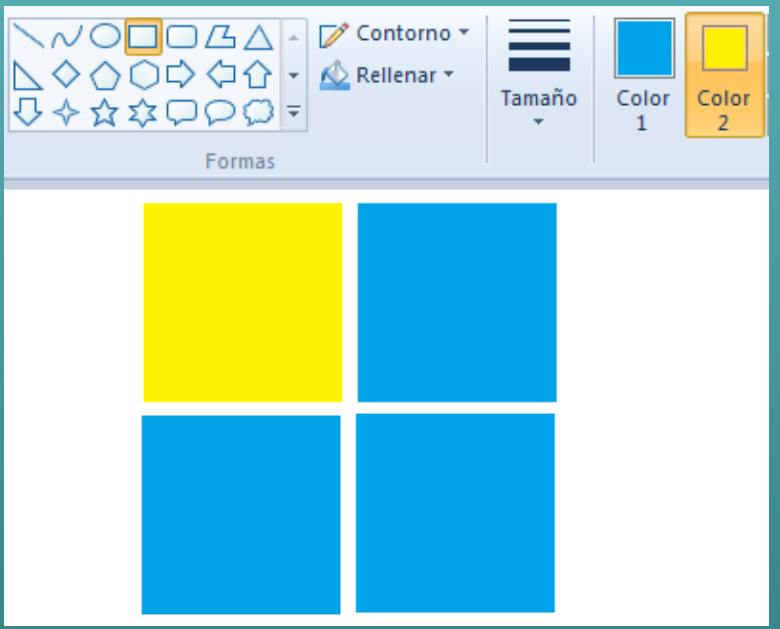


Painter model (2)

- There are 2 techniques to create images
- One of them is the painter model
- You take an element (brush, pencil) and use it to create images from scratch.
- Free strokes are used.
- Natural approach.
- I believe it is the perfect analogy for how we develop software.
  1. We begin with a blank page.
  2. Imagine the product as if it were finished.
  3. Implement the solution line by line.

Composition model (1)





Composition model (2)

- It is about creating images from existing elements.
- Example: mosaic
- Example: draw a chess board.
- I believe this is the approach we must follow when creating software.
    1. Take known and well defined elements.
    2. Put them together in a smart way.
    3. Create only the missing components.
- There is an enormous amount of existing components in the world of software.

Painter vs composition

- One method can be faster than the other, depending on the type of image involved.
- Also consider which one achieves a better result.
- Nevertheless, I believe that most problems in the software world adjust better to the composition model.

Enough philosophy...

Concrete example:

- Create a website for the Uruguayan .NET community
- Include a forum

¿Why use a CMS?

- They are fundamentally based around CBSD.
- Both structure and content are treated as components
  - Pages, images, posts, widgets, modules, etc.
- Huge community and component catalog available.

Microsoft WebMatrix (1)

- Free development environment by Microsoft
- Lightweight: 40MB out of the box compared to several GB from Visual Studio
- Designed for the cloud
- Handles many languages and frameworks
  - ASP.NET
  - PHP
  - Node.js
- Integrates with con GIT and TFS

Microsoft WebMatrix (2)

- Visual Studio feels like a laboratory for creating software
- WebMatrix is ideal for DSBC
  - Provides components:
    - Framework gallery
    - Incorporates NuGet package manager
    - Useful extensions
  - Isolates framework from components

OrchardCMS (1)

- Completely Open Source
- Rapid growth
- MVC Architecture
- This means that all components follow a MVC architecture.
  - Creating new components is simple
  - Components are easy tounderstand and extend

Orchard CMS (2)

Components

- Content
- Module
  - Features
- Themes
- Templates
  - Shapes
- Widgets
- Users, roles, permissions

Resources

- Slides and code:
  - github.com/diegocard/CBSD-presentation
- CBSD in MSDN (spanish):
  - http://msdn.microsoft.com/es-es/library/bb972268.aspx#ref07back
- Great related presentation (spanish):
  - http://www.slideshare.net/ulicruz/desarrollo-de-software-basado-en-componentes
- Paper
  - Component-Based Software Engineering – New Paradigm of Software Development (Crnkovic, Larsson)

The end

"It has been a long time in coming, but the Industrial Revolution of software is finally upon us. Specialization of resources, standards for interchangeable parts, and streamlined assembly tools have been used in other industries for hundreds of years to speed the development of highly complex products.
Despite their ubiquity, application of these concepts to the modern software industry is just beginning.."

Bill Gates, 1997