



# Loosening Coupling



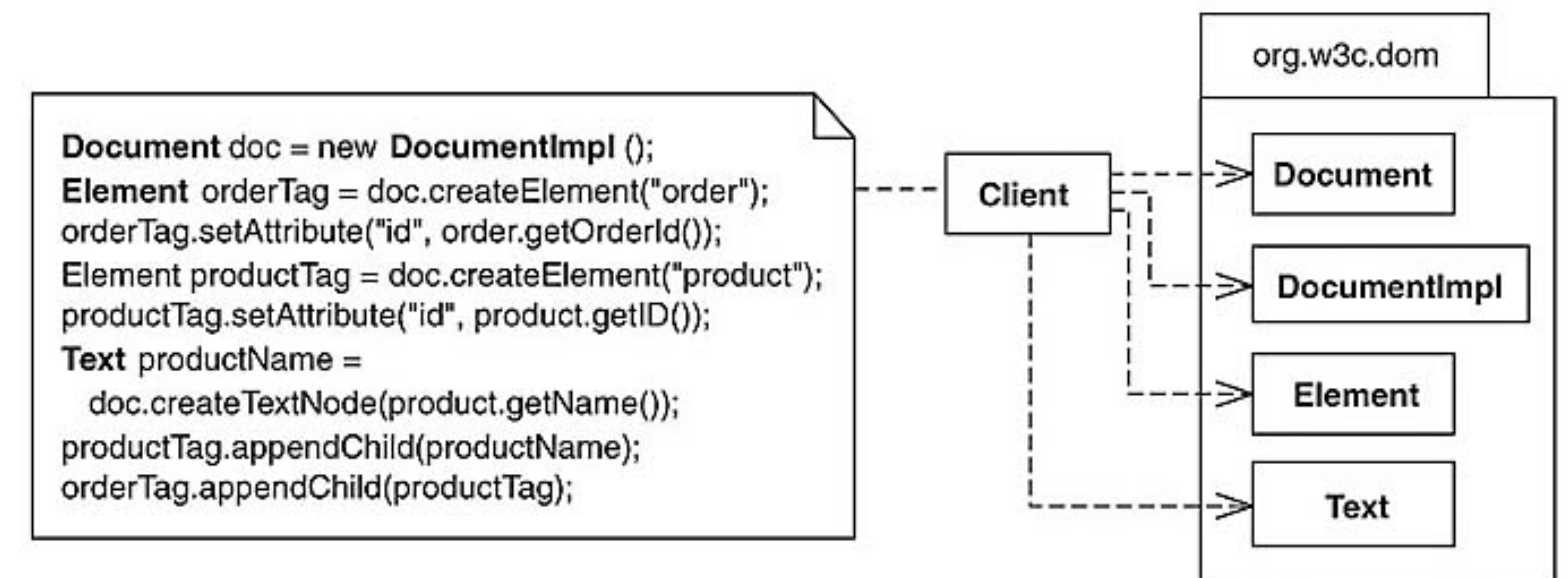
Dario Campagna

Head of Research and Development

# Composite construction

Building a Composite is repetitive,  
complicated, or error-prone

- You can forget to add a new node to a parent
- You can add a new node to the wrong parent
- Same batch of steps over and over again
- Client code coupled to Composite

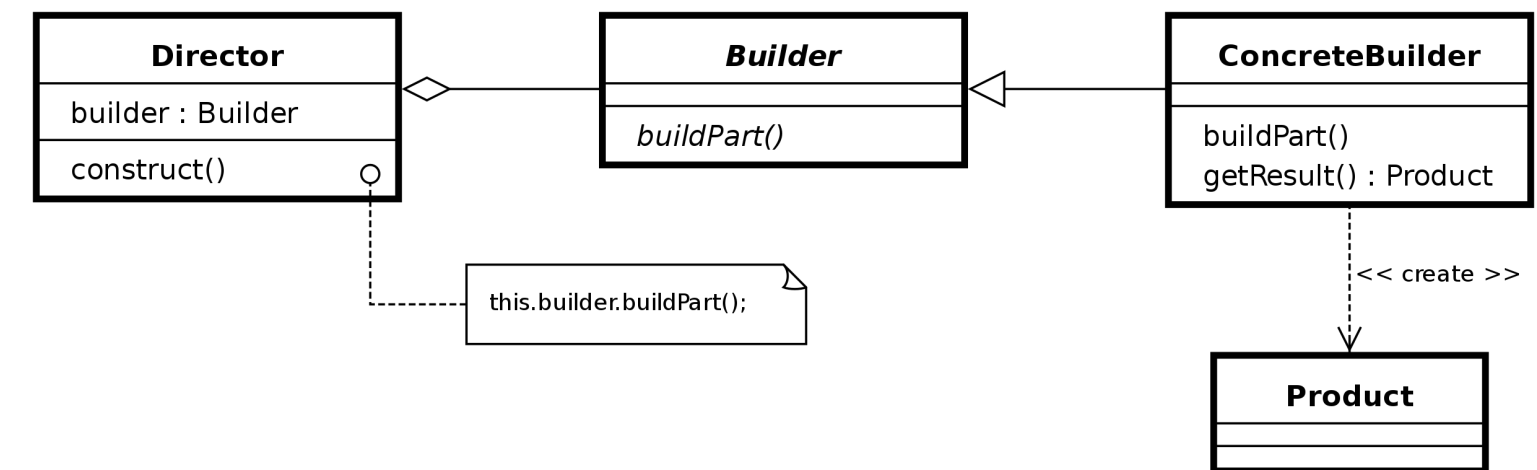


# Builder

Separate the construction of a complex object from its representation so that the same construction process can create different representations.

## Motivation

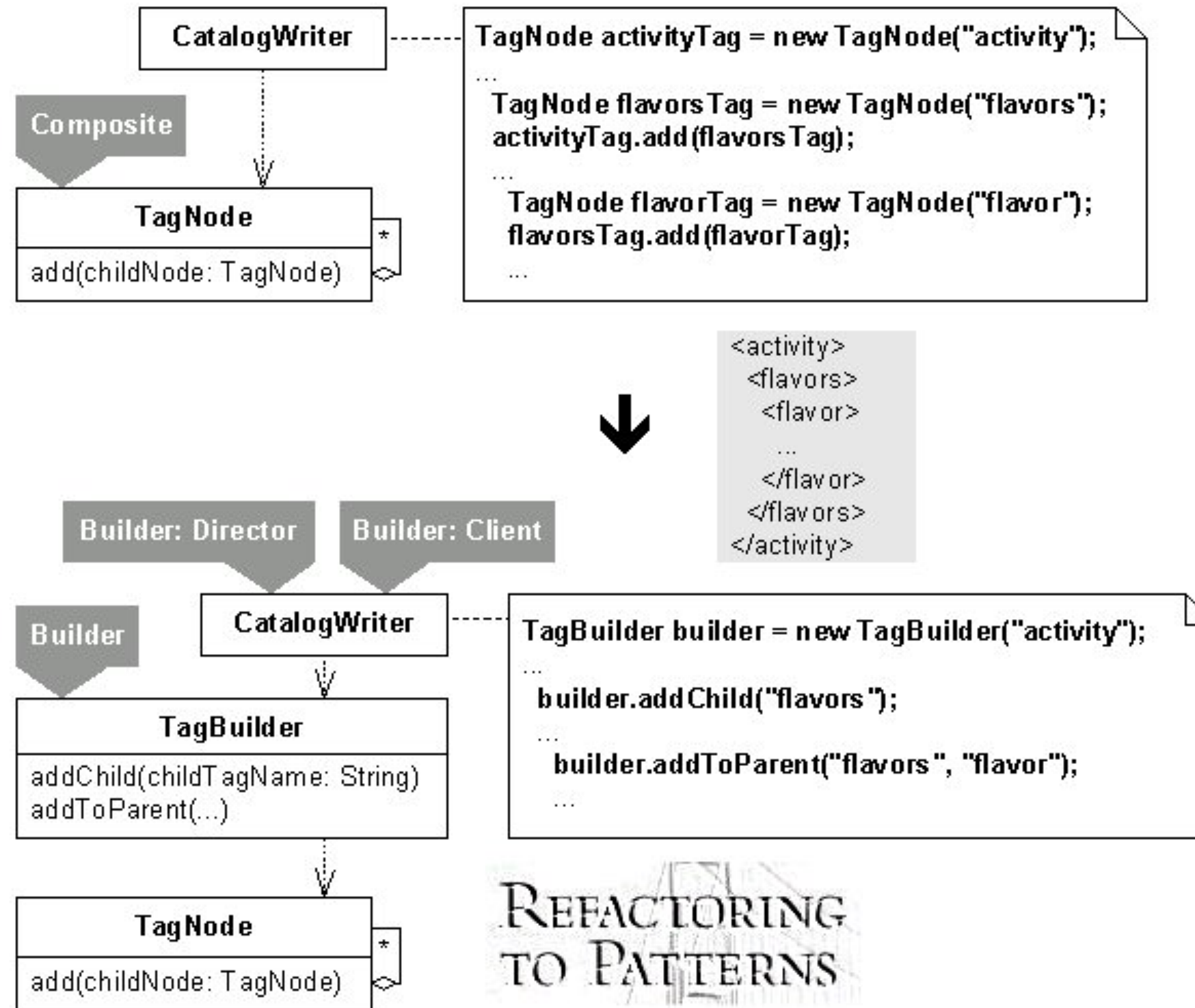
- A reader for the RTF format needs to convert RTF to many text formats
- Objects that requires laborious, step-by-step initialization of many fields and nested objects



## Applicability

- Construction process must allow different representations for the constructed object
- Objects with “telescoping constructors”

# Encapsulate Composite with Builder



# Encapsulate Composite with Builder

Benefits	Liabilities
Simplifies a client's code for constructing a Composite.	May not have the most intention-revealing interface.
Reduces the repetitive and error-prone nature of Composite creation.	
Creates a loose coupling between client and Composite.	
Allows for different representations of the encapsulated Composite or complex object.	



# Encapsulate Composite with Builder – Mechanics

1. Create a **builder**, make it possible for it to produce a one-node Composite.  
✓ Compile and test
2. Make the builder capable of building children.  
✓ Compile and test
3. Make the builder capable of settings attributes and values (if any).  
✓ Compile and test
4. Reflect on how simple your builder is for clients to use, and then make it simpler.  
✓ Compile and test
5. Refactor your Composite-construction code to use the new builder.  
✓ Compile and test



# Encapsulate Composite with Builder – Example

Let's apply this refactoring to the Composite TagNode in the *composite* branch of the following repository.

<https://github.com/dario-campagna/encapsulate-composite-with-builder>

- Example from Refactoring to Patterns
- Continuation of “Replace Implicit Tree with Composite”

