Magritte

[René Magritte, 1966] Decalcomania

www.lukas-renggli.ch
Software Composition Group
University of Bern

Lukas Renggli

- Academics
  - PhD Student, University of Bern
- Industry
  - Independent Software Consultant
- Open-Source Communities
  - Core-developer of Seaside
  - Author of Magritte and Pier

Agenda

- Introduction
- Example
- History and Usage
- Implementation Details
- Adaptive Models

Magritte

Introduction

Describe once,
Get everywhere

Adapted from Oscar Nierstrasz, 2008
http://scg.unibe.ch/Teaching/MM/Slides/01Intro.ppt.pdf

What is a model?

What is a meta-model?

Example

System	Real World	You
Model	Tables and Tuples	(Andreas, Muster, ...)
Meta-model	Database schema	Student, Course, ...
Meta-meta-model	Relational data model	Tables, Attributes, Tuples, ...

Metaprogramming

A program that manipulates
a program (possibly itself)

Reflection

Introspection

- The ability of a program to observe and reason about its own state.

Intercession

- The ability for a program to modify its own execution state or alter its own interpretation or meaning.

Example

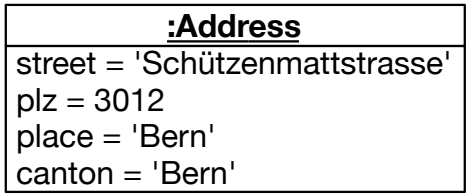
```
// Without introspection
World world = new World();
world.hello();

// With introspection
Class class = Class.forName("World");
Object object = cls.newInstance();
Method method = cls.getMethod("hello", null);
method.invoke(object, null);
```

Magritte

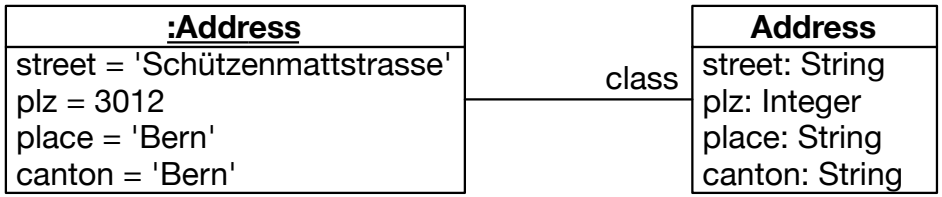
Example

Describe once,
Get everywhere

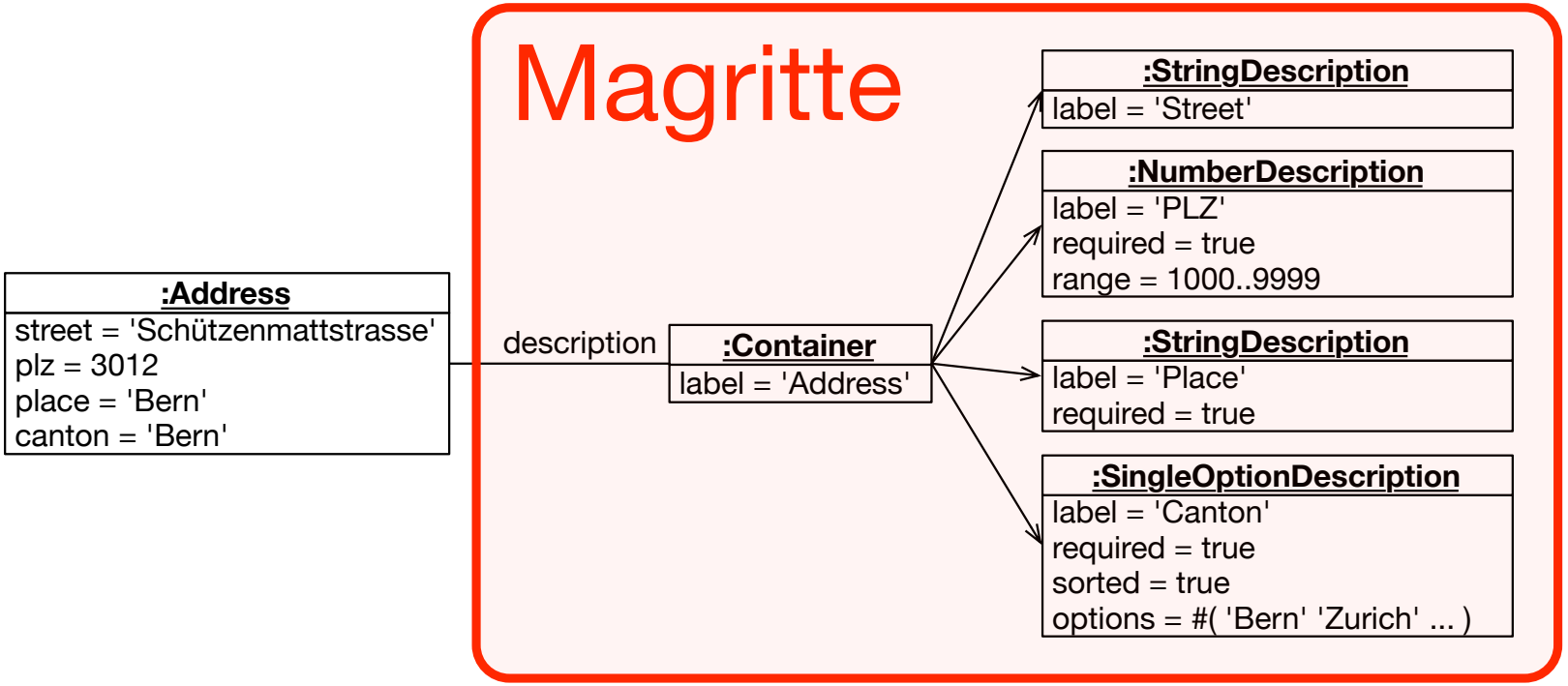
Address Object



Address Class



Address Description



Describe (1)

Address class>>#descriptionStreet

```
^ MAStringDescription new
    autoAccessor: #street;
    label: 'Street';
    priority: 100;
    yourself
```

Address class>>#descriptionPlz

```
^ MANumberDescription new
    autoAccessor: #plz;
    priority: 200;
    label: 'PLZ';
    beRequired;
    min: 1000;
    max: 9999;
    yourself
```

Describe (2)

Address class>>#descriptionPlace

```
^ MAStringDescription new
    autoAccessor: #place;
    label: 'Place';
    priority: 300;
    beRequired;
    yourself
```

Address class>>#descriptionCanton

```
^ MASingleOptionDescription new
    options: #('Zuerich' 'Bern' 'Luzern' ...);
    autoAccessor: #canton;
    label: 'Canton';
    priority: 400;
    beRequired;
    beSorted;
    yourself
```

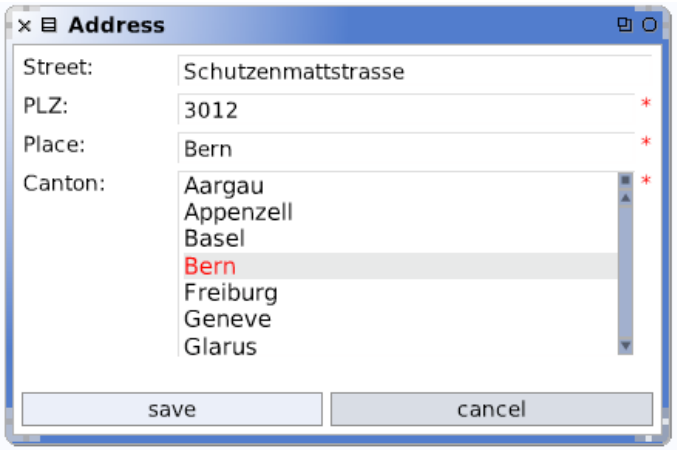
Interpret (1)

```
anAddress description do: [ :description |
    Transcript
        show: description label; show: ':'; tab;
        show: (description toString: (anAddress readUsing: description));
        cr ]
```

Street: *Schutzenmattstrasse*
PLZ: *3012*
Place: *Bern*
Canton: *Bern*

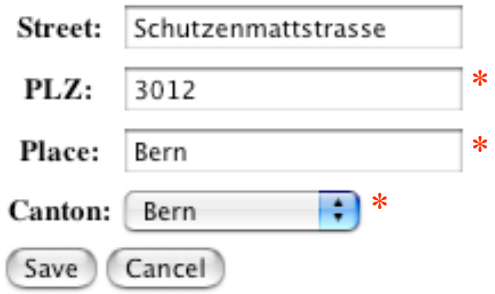
Interpret (2)

```
result := anAddress asMorph
        addButtons;
        addWindow;
        callInWorld
```



Interpret (3)

```
result := self call: (anAddress asComponent
        addValidatedForm;
        yourself).
```



What is it used for?

- Reflection
  - Introspection
  - Intercession
- Viewer building
- Editor building
- Report building
- Documentation
- Data validation
- Query processing
- Object filtering
- Object serialization
- Object copying
- Object indexing
- Object initialization
- Object extension
- Object adaption
- Object customization

and much more ...

Why is it useful?

- Describe once, get everywhere.
- Extensibility of classes is ensured.
- Context dependent descriptions.
- End-user customizable.
- Developer configurable.

Why is it cool?

- Describe once, get everywhere.
- Be more productive.
- Lower coupling.
- Empower your users.
- Do more, with less code.
- Do more, with less hacking.

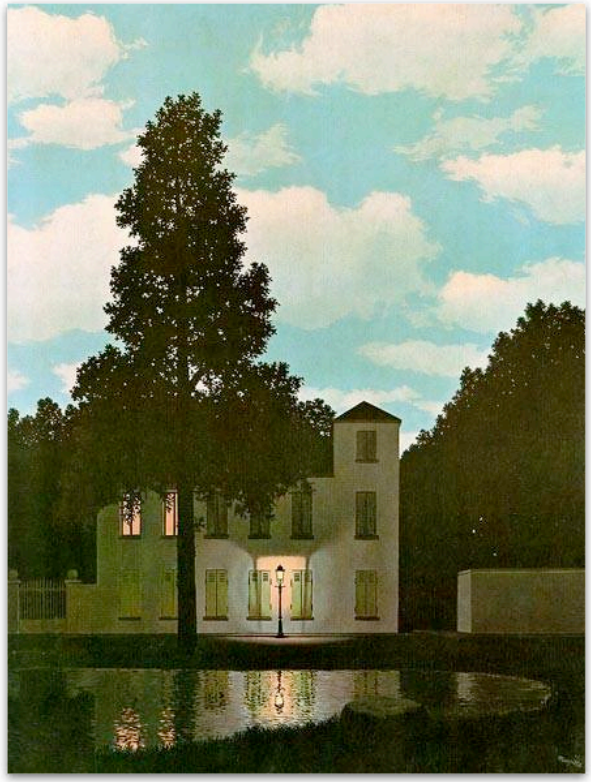
It seems a drag to me that you need to add ~~22~~ 14 class categories of Magritte, [...] I'd rather avoid Magritte. It's a ~~PhD~~ Master thesis, and so not optimized for simplicity. People who use it, love it, I suppose. Probably makes them feel smart.

— Harry Hamington [1]

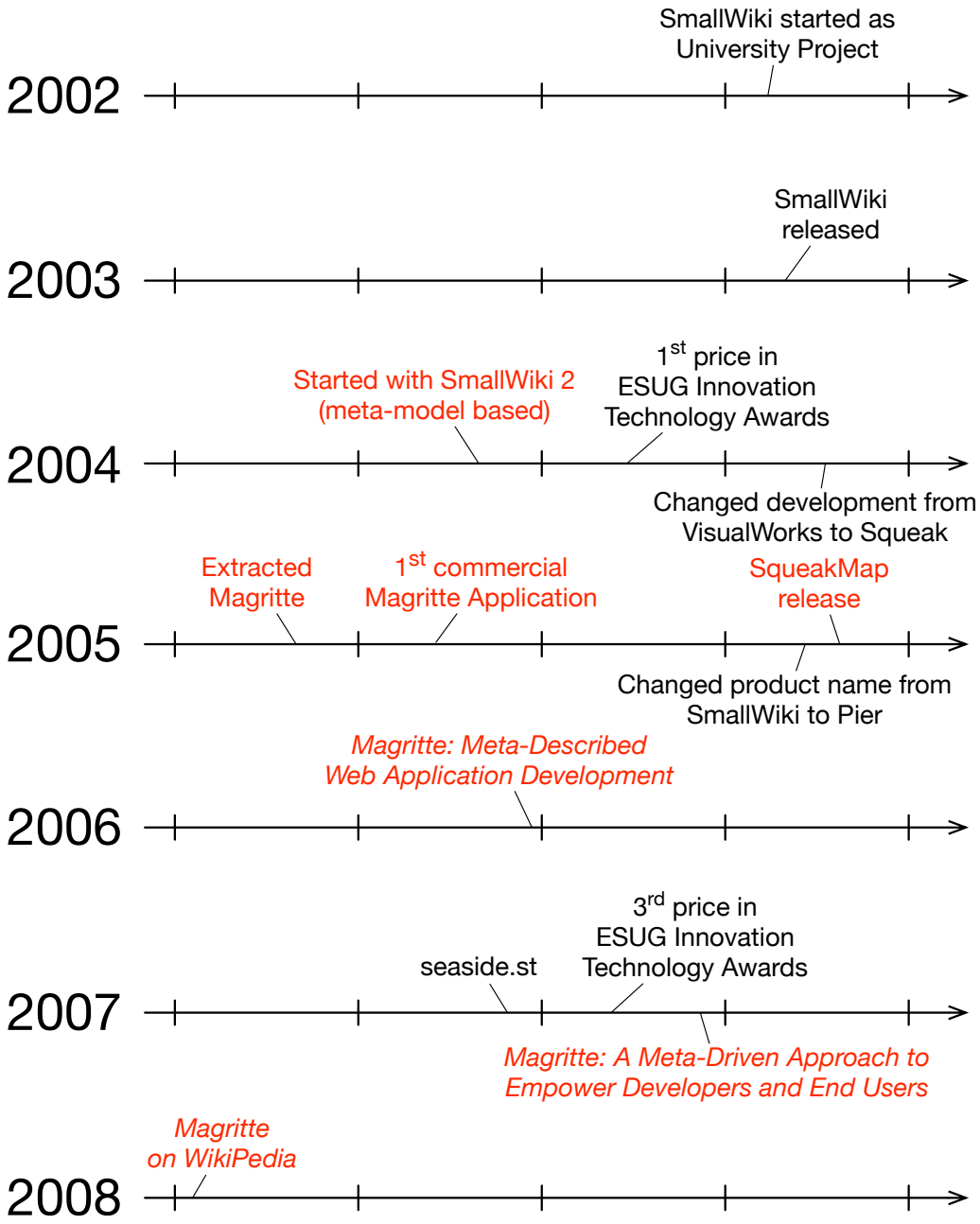
[1] Name d. Redaktion bekannt, geändert

Magritte

History and Usage



Describe once,
Get everywhere


2002
SmallWiki started as University Project
2003
SmallWiki released
2004
Started with SmallWiki 2 (meta-model based)
1st price in ESUG Innovation Technology Awards
Changed development from VisualWorks to Squeak
2005
Extracted Magritte
1st commercial Magritte Application
SqueakMap release
Changed product name from SmallWiki to Pier
2006
Magritte: Meta-Described Web Application Development
2007
seaside.st
3rd price in ESUG Innovation Technology Awards
Magritte: A Meta-Driven Approach to Empower Developers and End Users
2008
Magritte on WikiPedia

Search the Seaside

seaside

The framework for developing sophisticated web applications in Smalltalk

About

- Screenshots
- Users
- Examples
- What others think about Seaside
- Hosting

Documentation

- FAQ
- Tutorials
- Videos

Community

- Weblogs
- Mailing List
- Development
- Contribute
- Merchandise
- Extensions
- Projects

Seaside 2.8

News

Industry Misinterpretations 75: Invoking the Demon *17 February 2008*
This week, we interviewed Randal Schwartz, who has recently come back to the Smalltalk world (by w...

Remote Seaside Debugging with Persistent Continuations *14 February 2008*
If you've read the section on Hyper in the Terse Guide to the GLASS Tools, then you realize t...

Podcast of James Foster's Ottawa STUG Talk *12 February 2008*
James Robertson has posted the audio for James Foster's talk on GLASS at the Ottawa Carleton S...

Jcrawler for Seaside Testing *10 February 2008*
jcrawler is a good tool for load testing Seaside applications - in theory. Unfortunately, it takes a...

download

open source

Seaside is a free and Open Source™ web application framework distributed under the MIT License.

Seaside is available on the following Smalltalk platforms:

- Squeak (download)
- Cincom Smalltalk
- Dolphin
- GemStone

esug

navigation

About
Community
Documentation
Download

participate

Mailing List: Ask questions and talk with Seaside experts.
Weblogs: Read the latest news about the Seaside community.
Contribute: Help to improve code and documentation of Seaside.

manage

View Browse Login

powered by seaside*

This website is run on Squeak, Seaside and is powered by the Pier CMS. The design was kindly provided by Samuel Morello. The site is hosted on a server of netstyle.ch.

SIG Beer - Ralf Lämmel - Not quite a sales pitch for C# 3.0 and .NET's LINQ

Language: English
Date: 5 March 2008 5:30 pm
Registered participants: 15
Location: Institut für Informatik und angewandte Mathematik (IAM), Universität Bern, Engehalde 8, Room 003.

About Ralf Lämmel

Ralf Lämmel serves on the faculty of the Department of Computer Science at the University of Koblenz-Landau. He was appointed a professorship per 1 July 2007. Prior to this appointment, in the years 2005-2007, Ralf Lämmel was affiliated with Microsoft Corp., where he served on a research and development position with focus on XML technologies. In the years 2001–2004, Ralf Lämmel served on a permanent faculty position, at the Free University of Amsterdam, in the Software Engineering department, and he was also affiliated with the Dutch Center for Mathematics and Computer Science (CWI) -- starting in 1999. Ralf Lämmel obtained his PhD in Computer Science (dissertation in programming theory) from the University of Rostock (Germany) in December 1998.

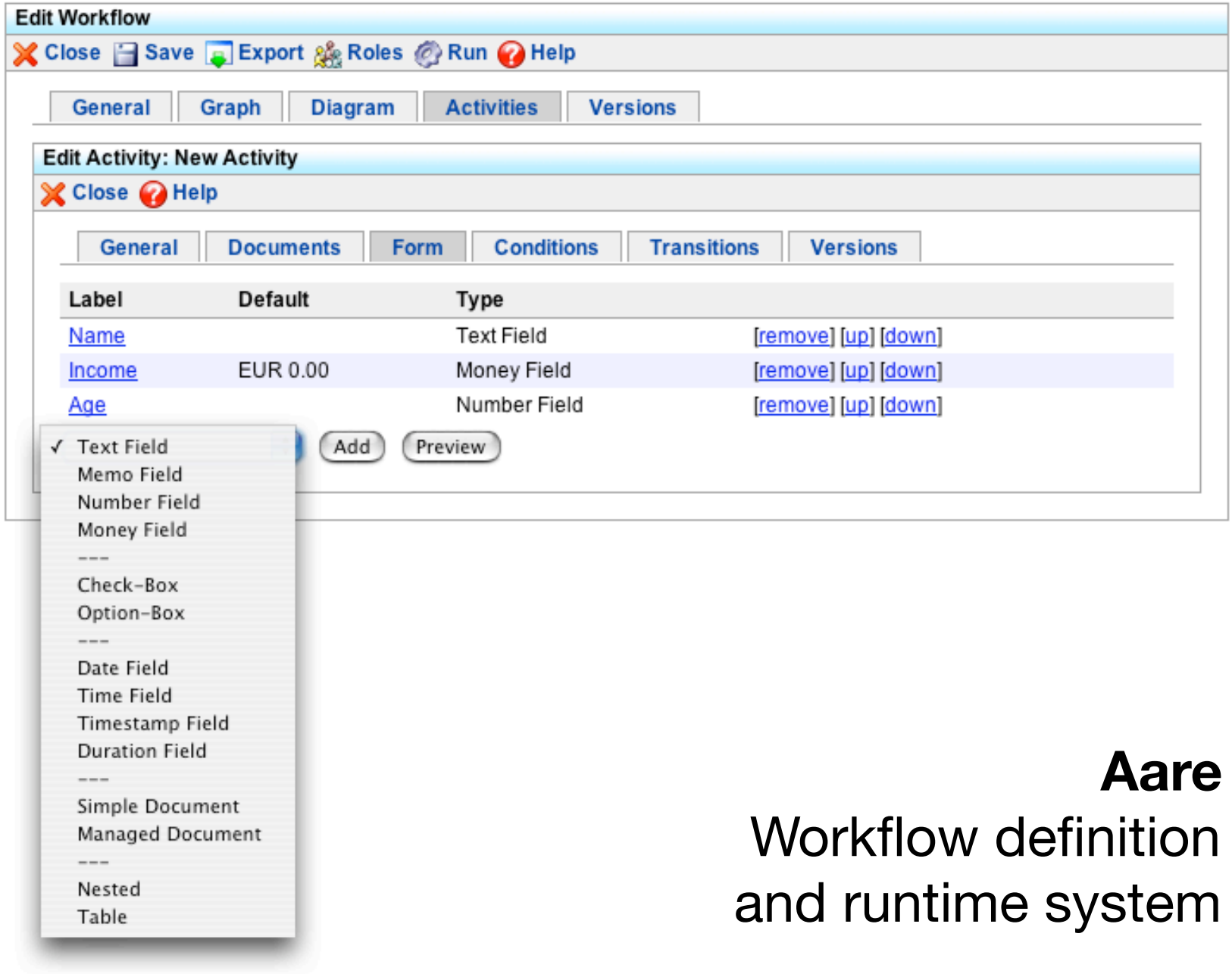
Ralf Lämmel's speciality is "software language engineering" but he is generally interested in many themes of software engineering and programming. His

Talk abstract

Suppose you are an academic Haskell and Prolog nerd as well as a grammar hacker with a sympathy for the real IT world (which equals Cobol, SQL, C\\++, and more recently, VB, C#, XML). Temporarily bored by the academic treadmill and Haskell's purity, you end up working at Microsoft for some time, which is good timing because Microsoft's LINQ is just taking shape. (I worked in the SQL/XML/LINQ teams at Microsoft, Redmond, 2005-2007.)

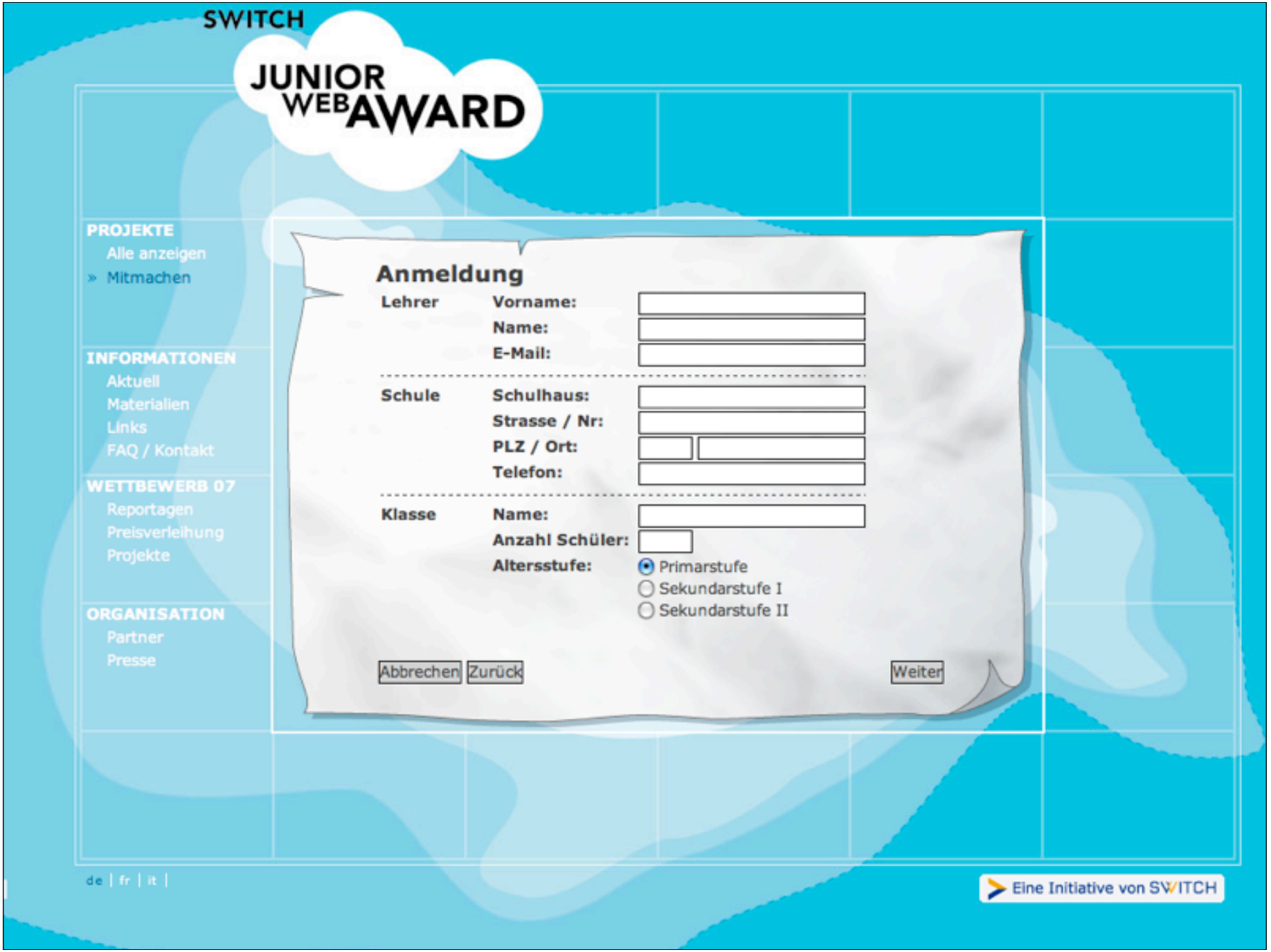
What is LINQ? Here is my short, unofficial explanation: LINQ (for Language-INtegrated Query) is a programming technology designed to keep it's head above water in the Bermuda triangle of data processing (X/O/R); technically, it is a combination of some cheap OO and functional programming language extensions, an almost unrecognizable stretch of Haskell's list comprehensions, a clever exploit of interface polymorphism, a statically typed + quotation-free form of

Slides



Aare

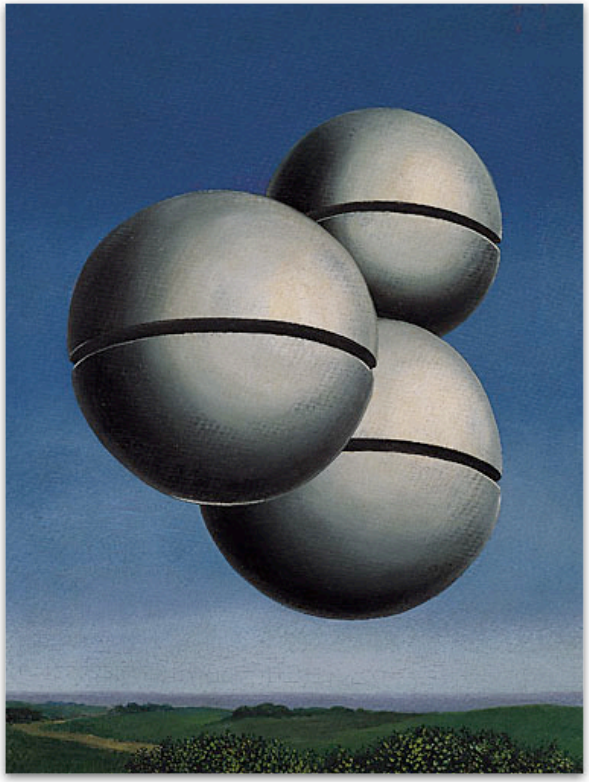
Workflow definition
and runtime system


SWITCH
JUNIOR WEB AWARD
PROJEKTE
Alle anzeigen
» Mitmachen
INFORMATIONEN
Aktuell
Materialien
Links
FAQ / Kontakt
WETTBEWERB 07
Reportagen
Preisverleihung
Projekte
ORGANISATION
Partner
Presse
Anmeldung
Lehrer
Vorname:
Name:
E-Mail:
Schule
Schulhaus:
Strasse / Nr:
PLZ / Ort:
Telefon:
Klasse
Name:
Anzahl Schüler:
Altersstufe:
Primarstufe
Sekundarstufe I
Sekundarstufe II
Abbrechen
Zurück
Weiter
de | fr | it |
Eine Initiative von SWITCH


THE HISTORY CHANNEL
CONCURSO
CLASSIC
ALBUMS
RESPONDE LA PREGUNTA
1) Pregunta número 1 acerca de classics albums
COMPLETA TUS DATOS
Nombre
Apellido
Cédula de Identidad
Teléfono
Email
Dirección
Ciudad
Estado
Código Postal

Magritte

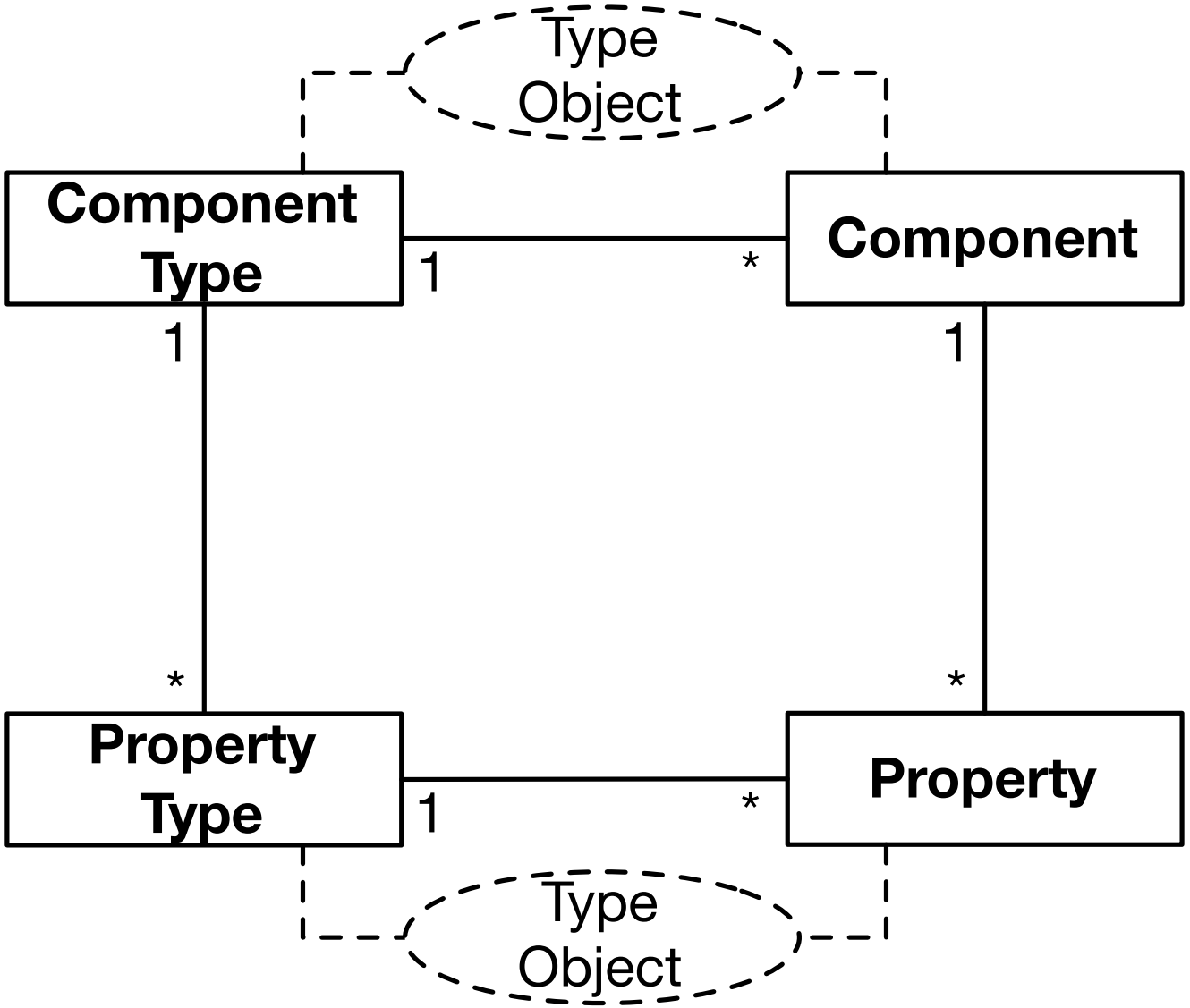
Implementation Details



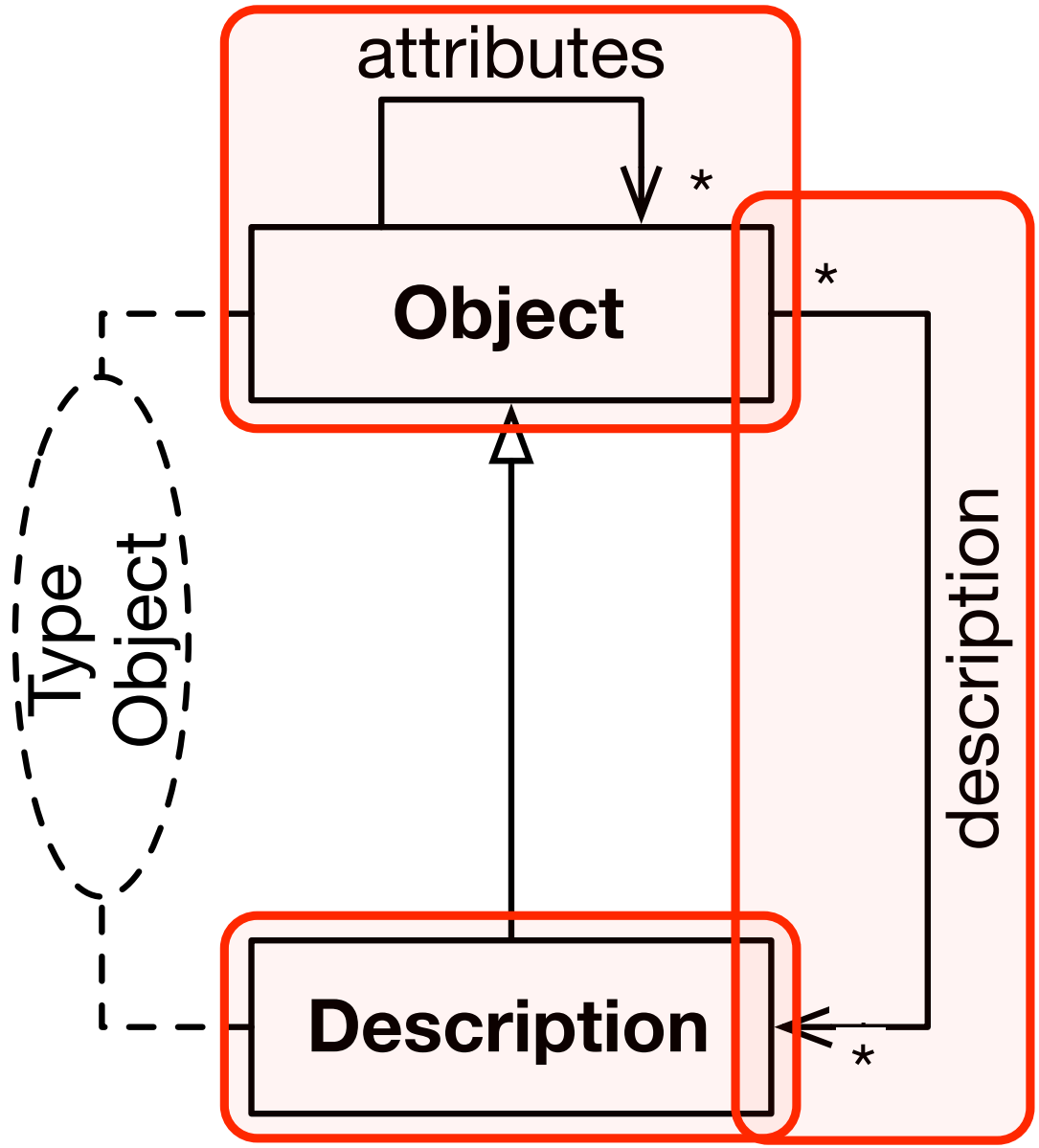
Describe once,
Get everywhere

Descriptions

- Problem
  - Objects need to be treated differently.
  - Types do not provide required information.
- Examples
  - Default value, print string, validation condition, ...
- Solution
  - Introduce a descriptive hierarchy that can be instantiated, configured and composed.



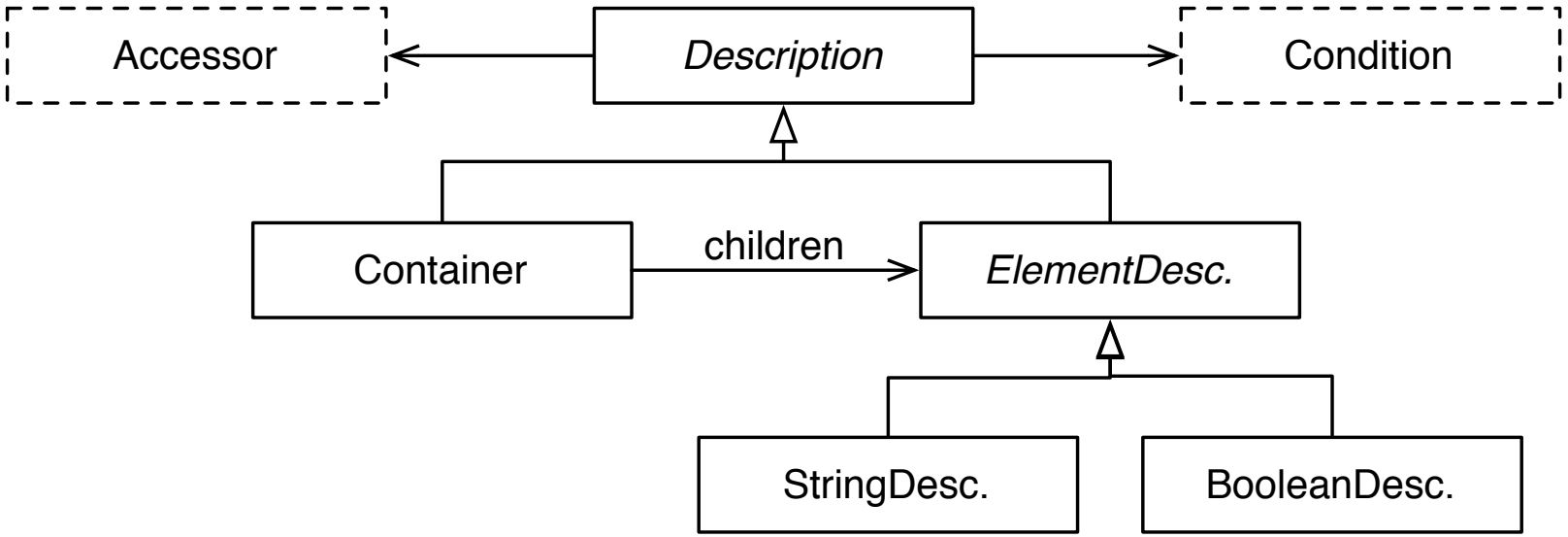
[Yoder *et al*, 2001] Architecture and design of adaptive object models



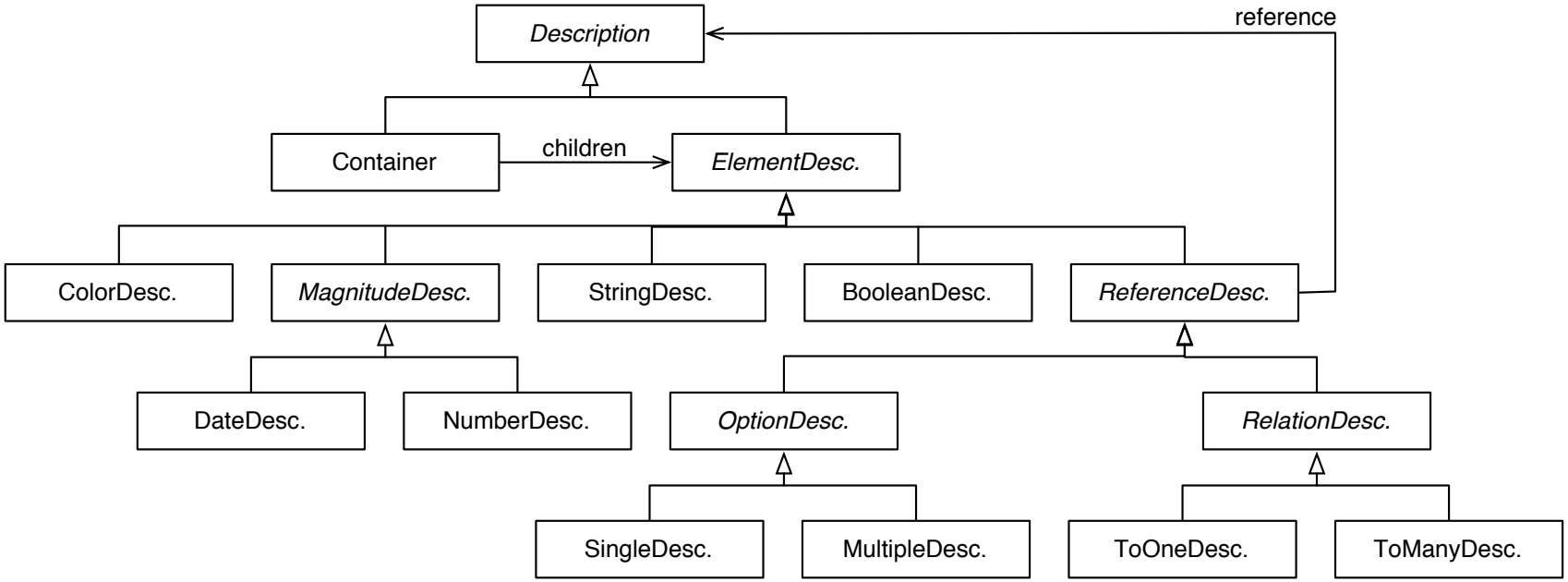
[Renggli *et al*, 2007] Magritte — A Meta-Driven Approach to Empower Developers and End Users

Descriptions

Composite pattern to describe objects



Descriptions

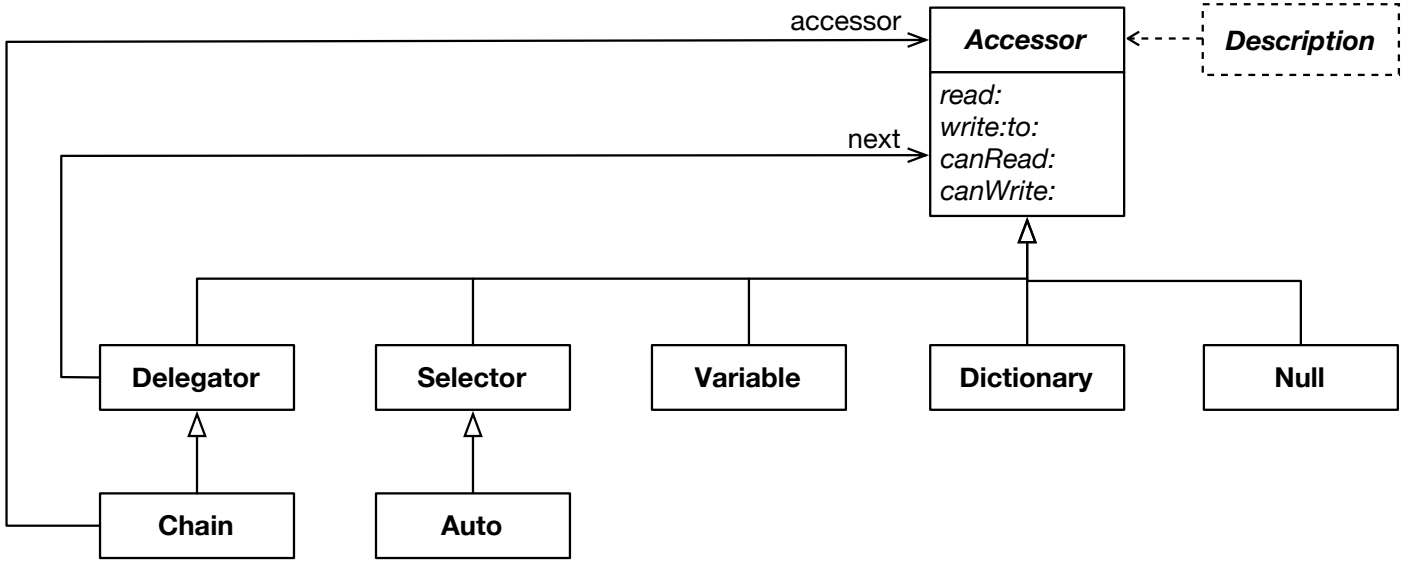


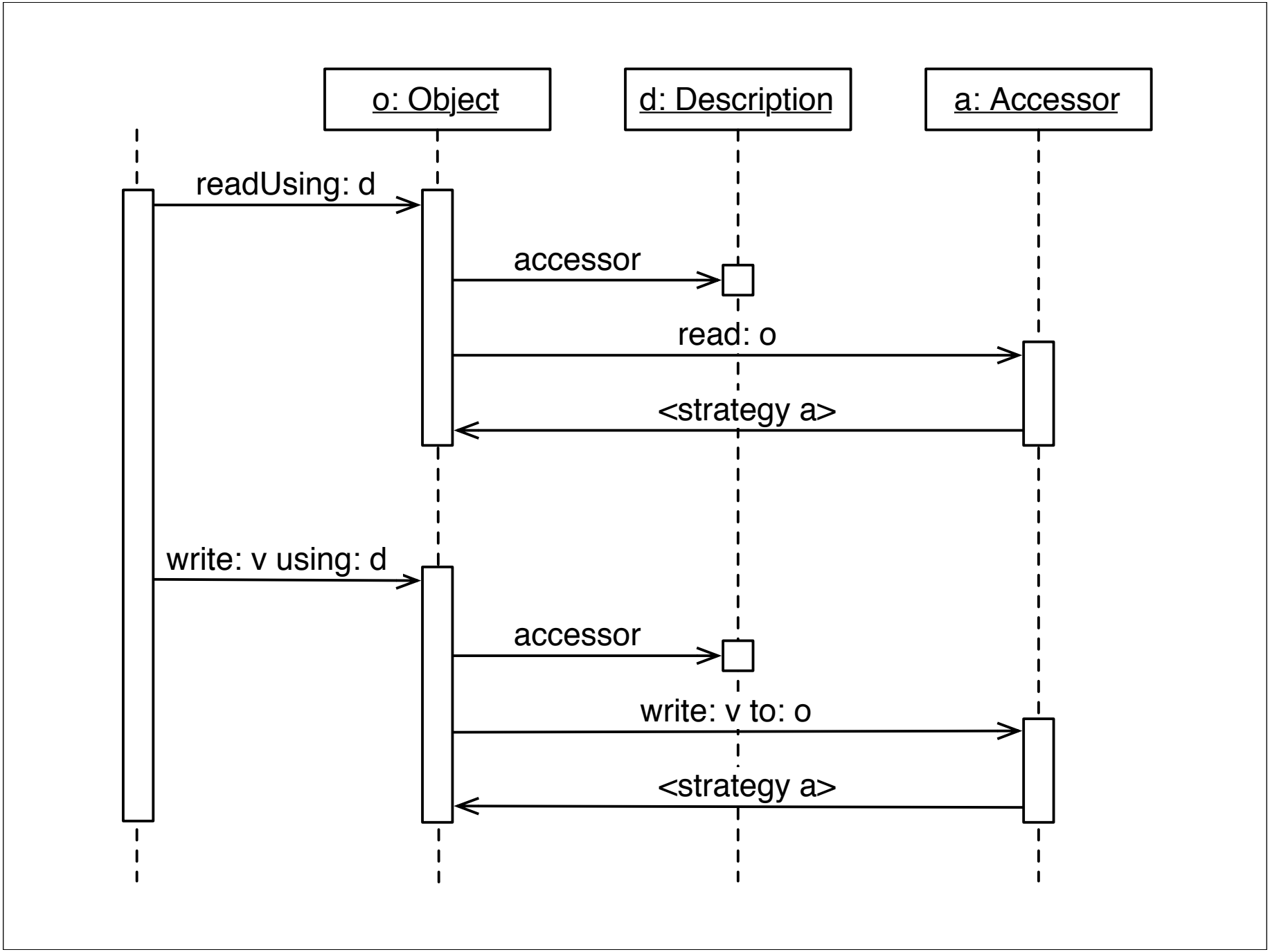
Accessors

- Problem
  - Data can be stored and accessed in different ways.
- Examples
  - Accessor methods, instance-variables, hash-tables, calculated values, external values, etc.
- Solution
  - Provide a strategy pattern to be able to access data through a common interface.

Accessors

Strategy pattern
to access model-entities.



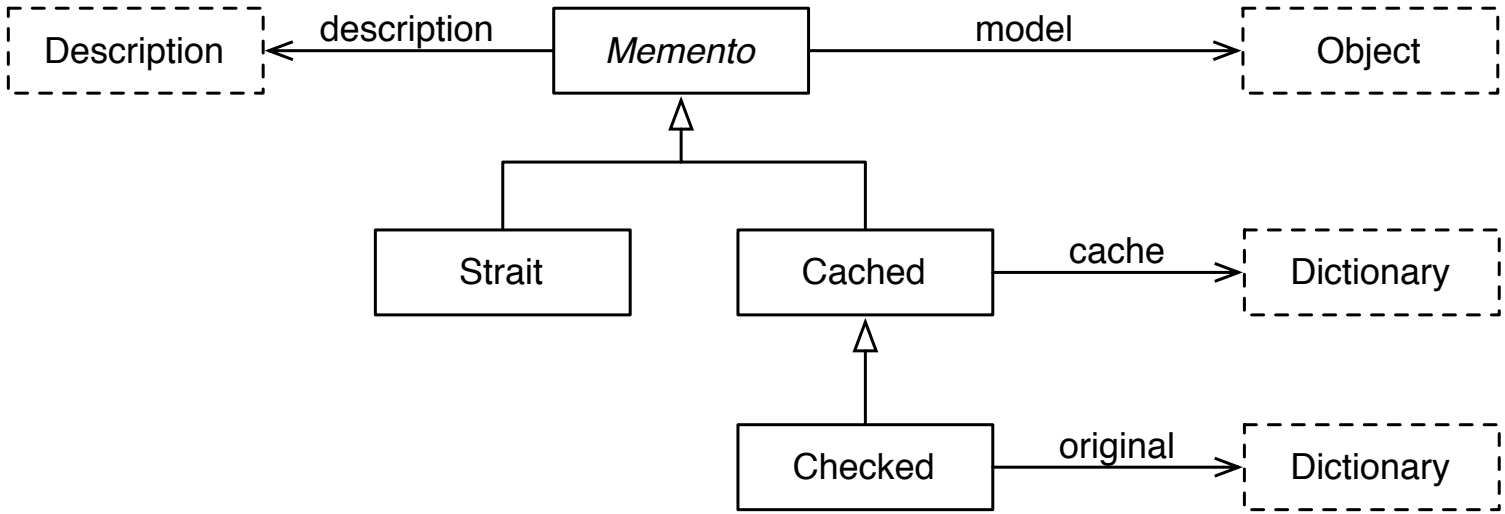

o: Object
d: Description
a: Accessor
readUsing: d
accessor
read: o
<strategy a>
write: v using: d
accessor
write: v to: o
<strategy a>

Mementos

- Problems
  - Editing might turn a model (temporarily) invalid.
  - Canceling an edit shouldn’t change the model.
  - Concurrent edits of the same model should be detected and (manually) merged.
- Solution
  - Introduce mementos that behave like the original model and that delay modifications until they are committed.

Mementos

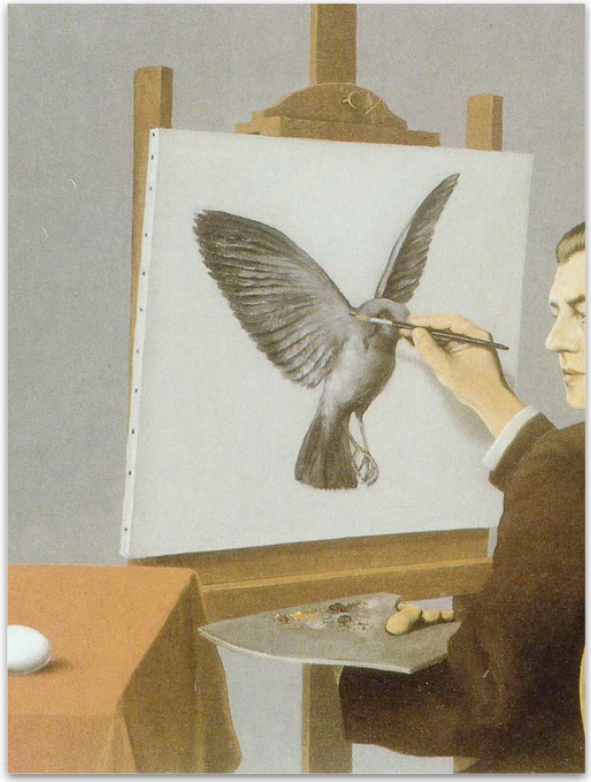
Memento pattern
to cache model-entities.



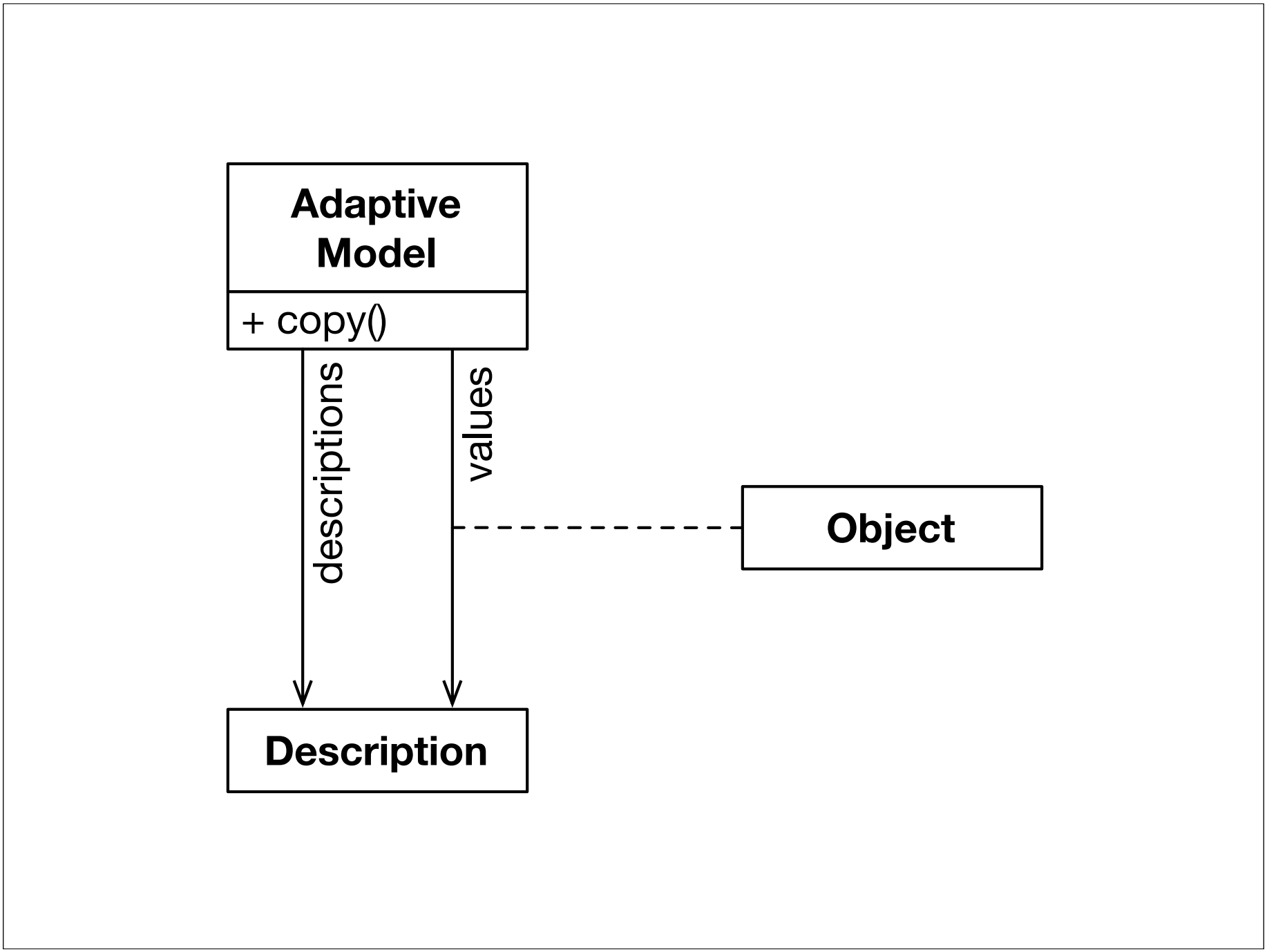
Magritte

Adaptive Models

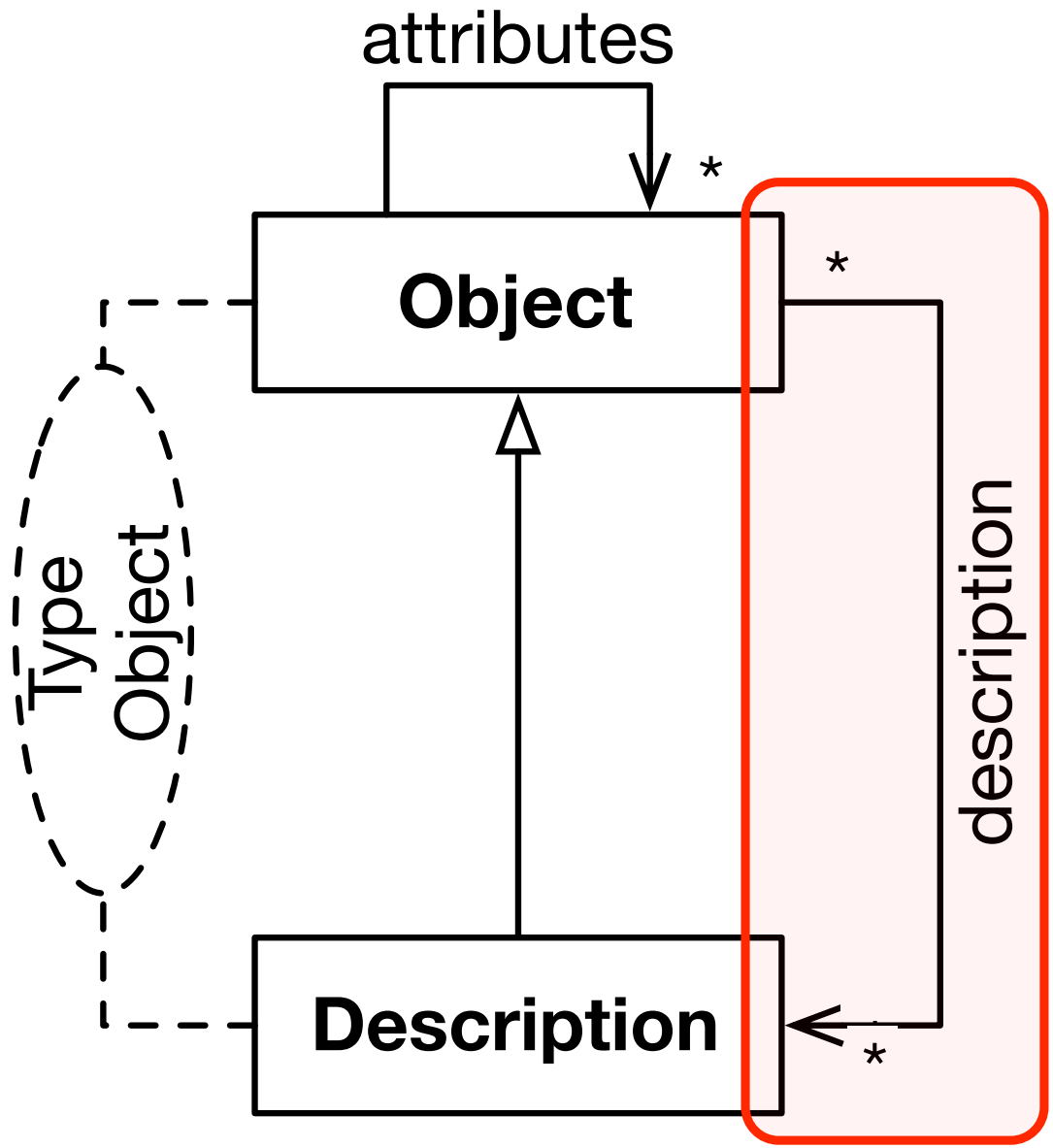
Describe once,
Get everywhere



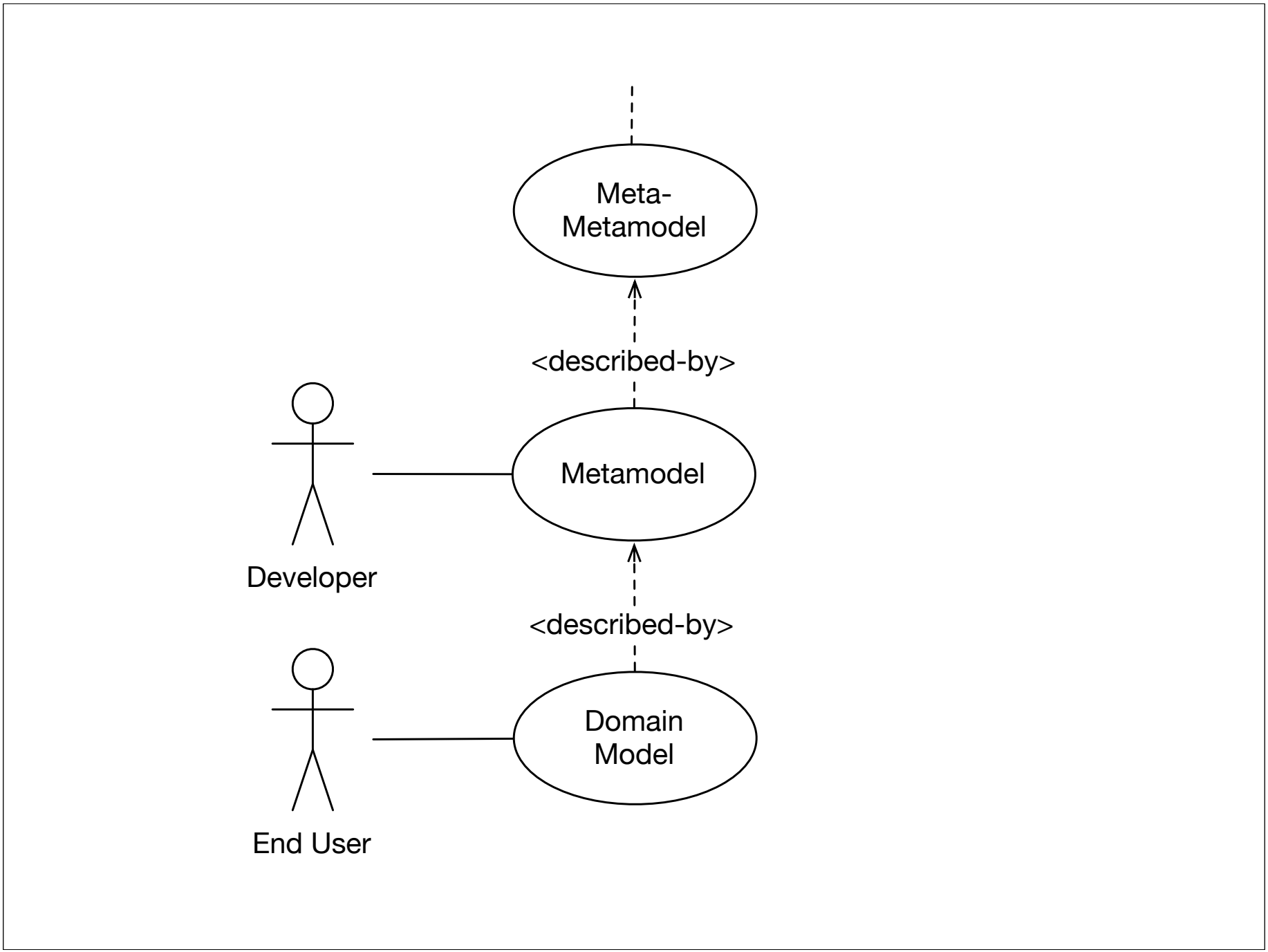
Rapidly changing requirements

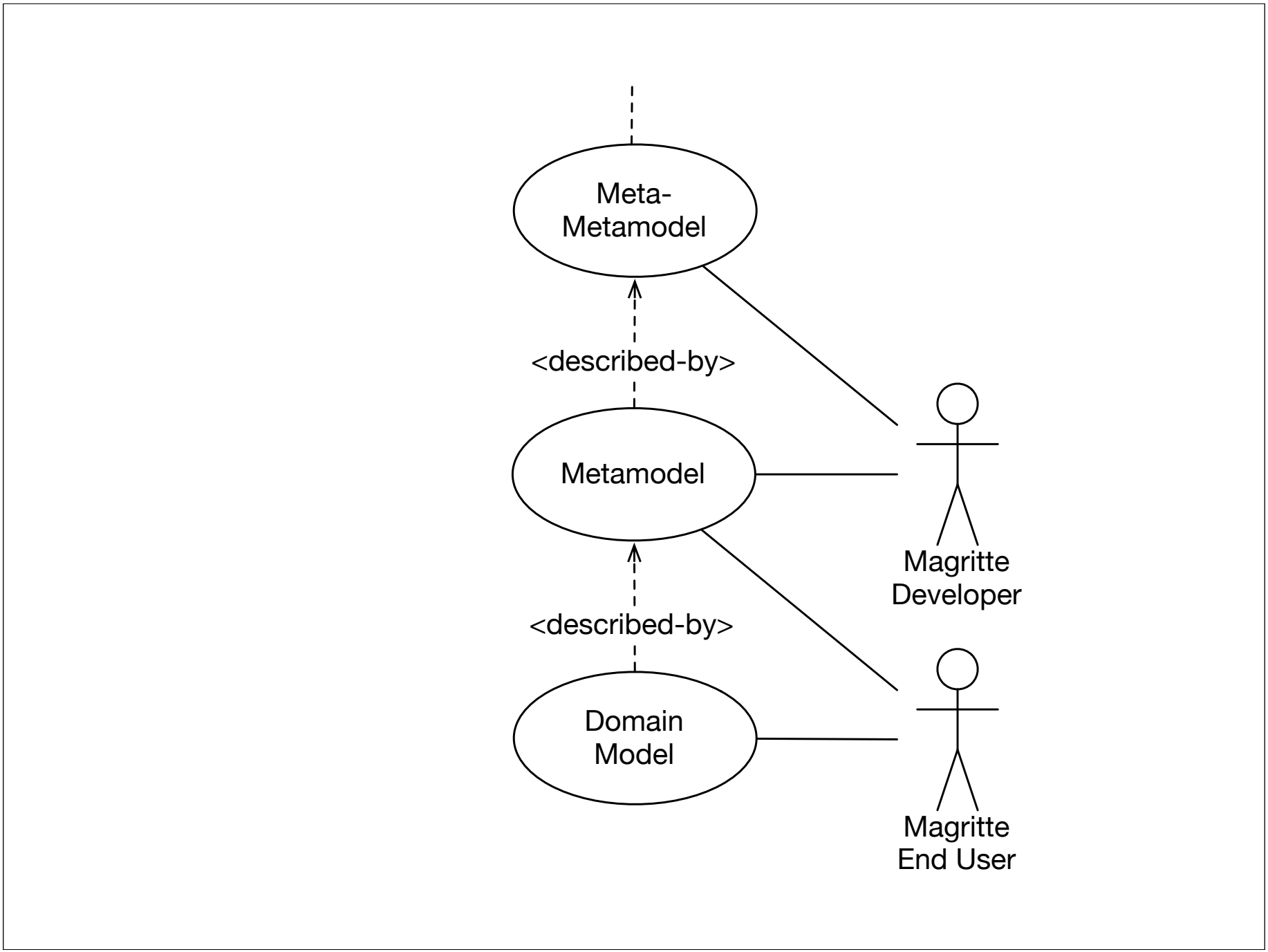

Adaptive Model
+ copy()
descriptions
values
Object
Description

End users would
know their requirements



[Renggli *et al*, 2007] Magritte — A Meta-Driven Approach to Empower Developers and End Users


Meta-
Metamodel
<described-by>
Metamodel
Developer
<described-by>
Domain
Model
End User


Meta-
Metamodel
<described-by>
Metamodel
<described-by>
Domain
Model
Magritte
Developer
Magritte
End User

Demo

Conclusion

- Describe once, get everywhere.
- Ensure extensibility and maintainability.
- Automate repetitive tasks.
- End-user customizability.
- (Run-time) adaptive object model.