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
- Communities
  - Core-developer of Seaside
  - Author of Magritte and Pier

Agenda

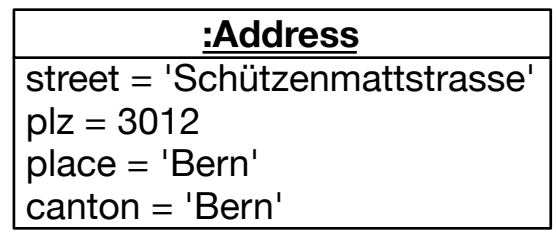
- Introduction and Example
- History and Usage
- Implementation Details
- Adaptive Models

Magritte

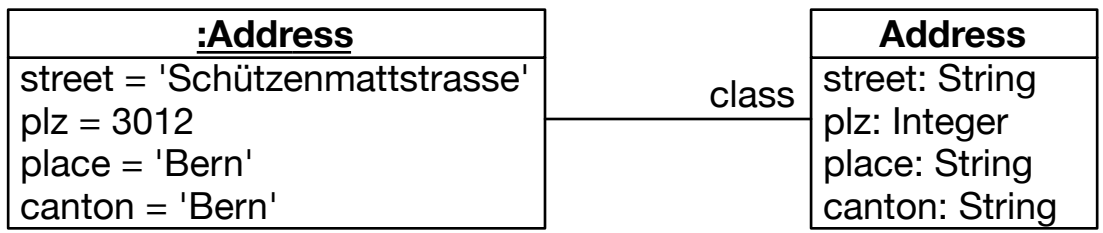
Introduction and Example

Describe once,
Get everywhere

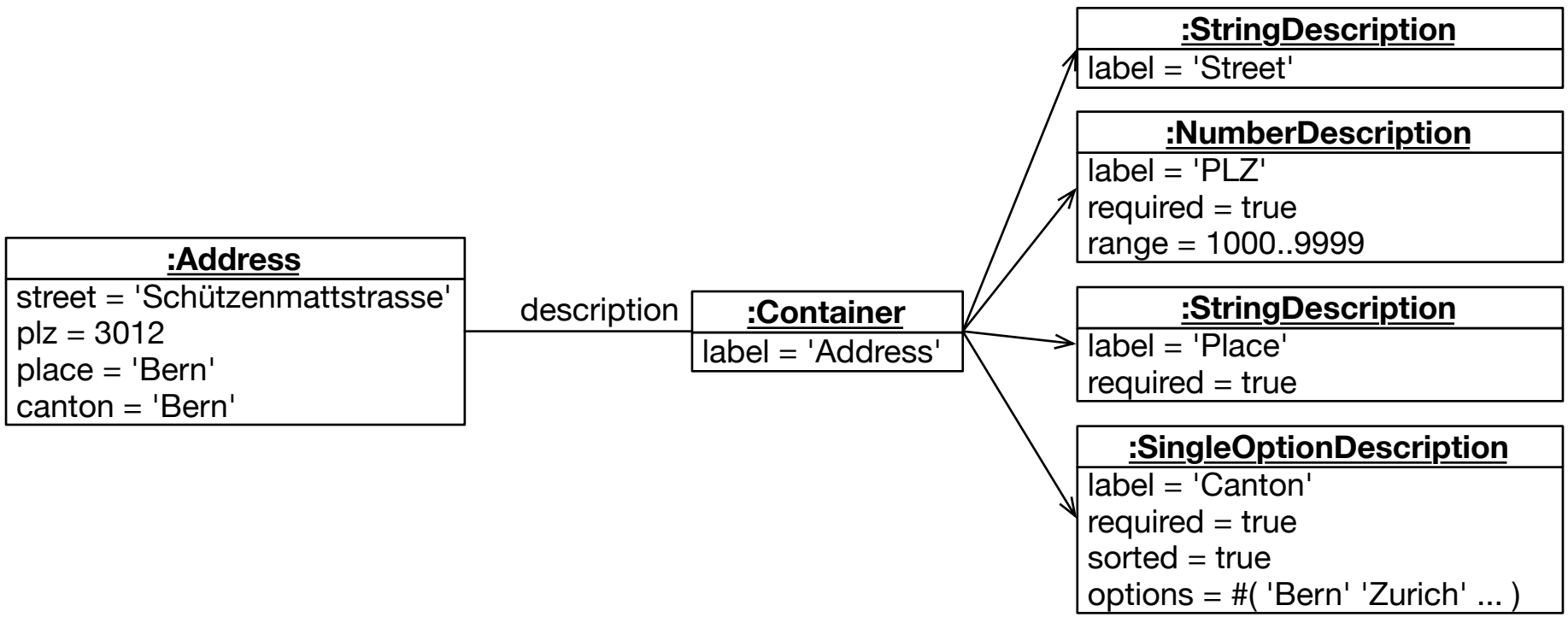
Address Object



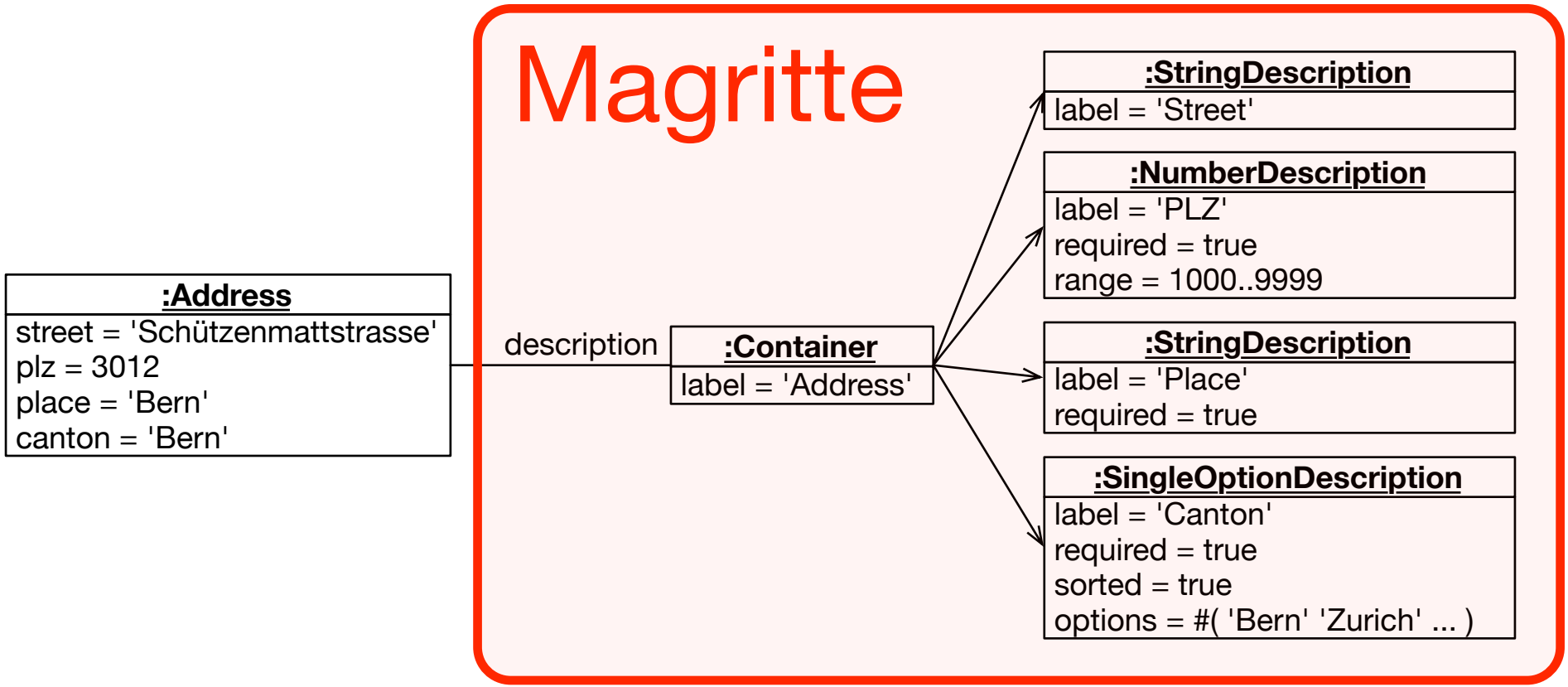
Address Class



Address Description



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 verification
- 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.

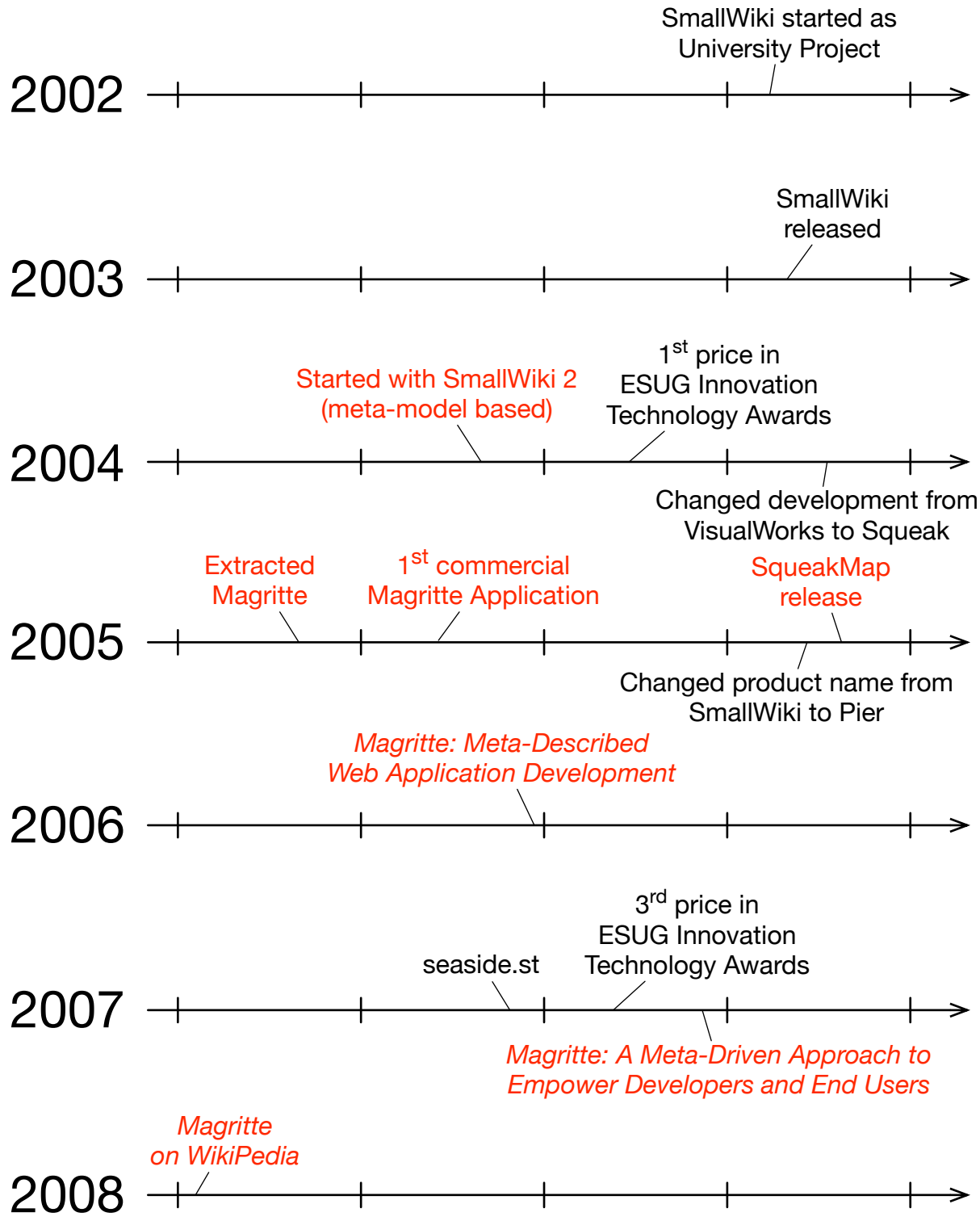
It seems a drag to me that you need to add ~~22~~ 14 class categories of Magritte, [...] I'd rather avoid Magritte. It's a ~~PhD~~ Master thesis, and so not optimized for simplicity. People who use it, love it, I suppose. Probably makes them feel smart.

— Chris Cunnington

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


seaside
The framework for developing sophisticated web applications in Smalltalk
About
Community
News
download
Seaside 2.8
Rendering Speed
Documentation
esug
navigation
participate
manage

SIG Beer - Andreas Zeller - Die Programmierumgebung der Zukunft

Language: German
Date: 1 February 2008 4:30 pm
Location: Zühlke Engineering AG, Wiesenstrasse 10a, 8952 Schlieren, Zürich

About Andreas Zeller

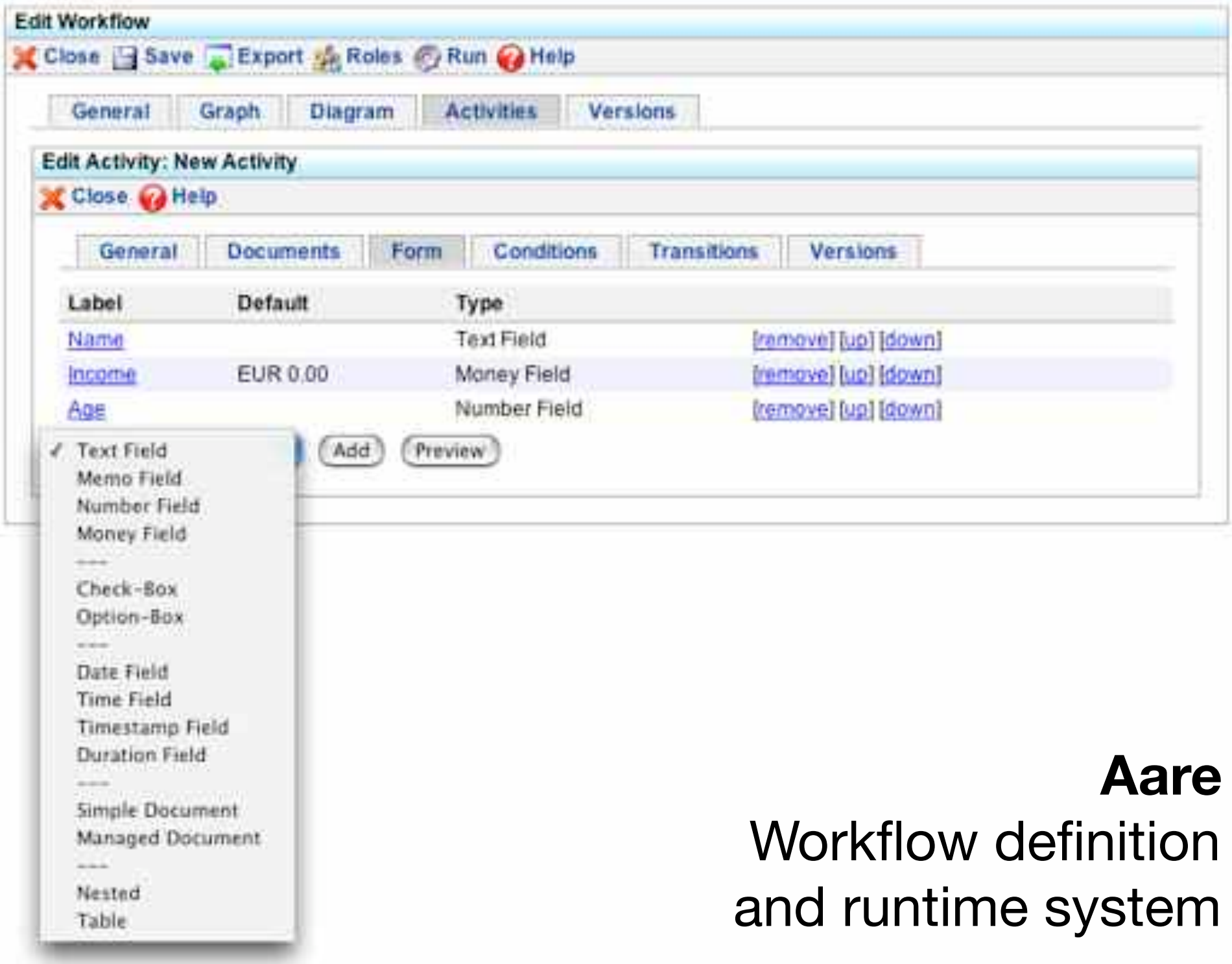
Andreas Zeller ist Professor für Softwaretechnik an der Universität des Saarlandes in Saarbrücken. Zeller erforscht grosse Software-Systeme und ihre Fehler - von Mozilla bis Microsoft. Sein Buch "Why Programs Fail - a guide to systematic debugging" wurde 2006 mit dem Software Development Magazine productivity award ausgezeichnet.

Andreas Zeller's home page.

Talk abstract

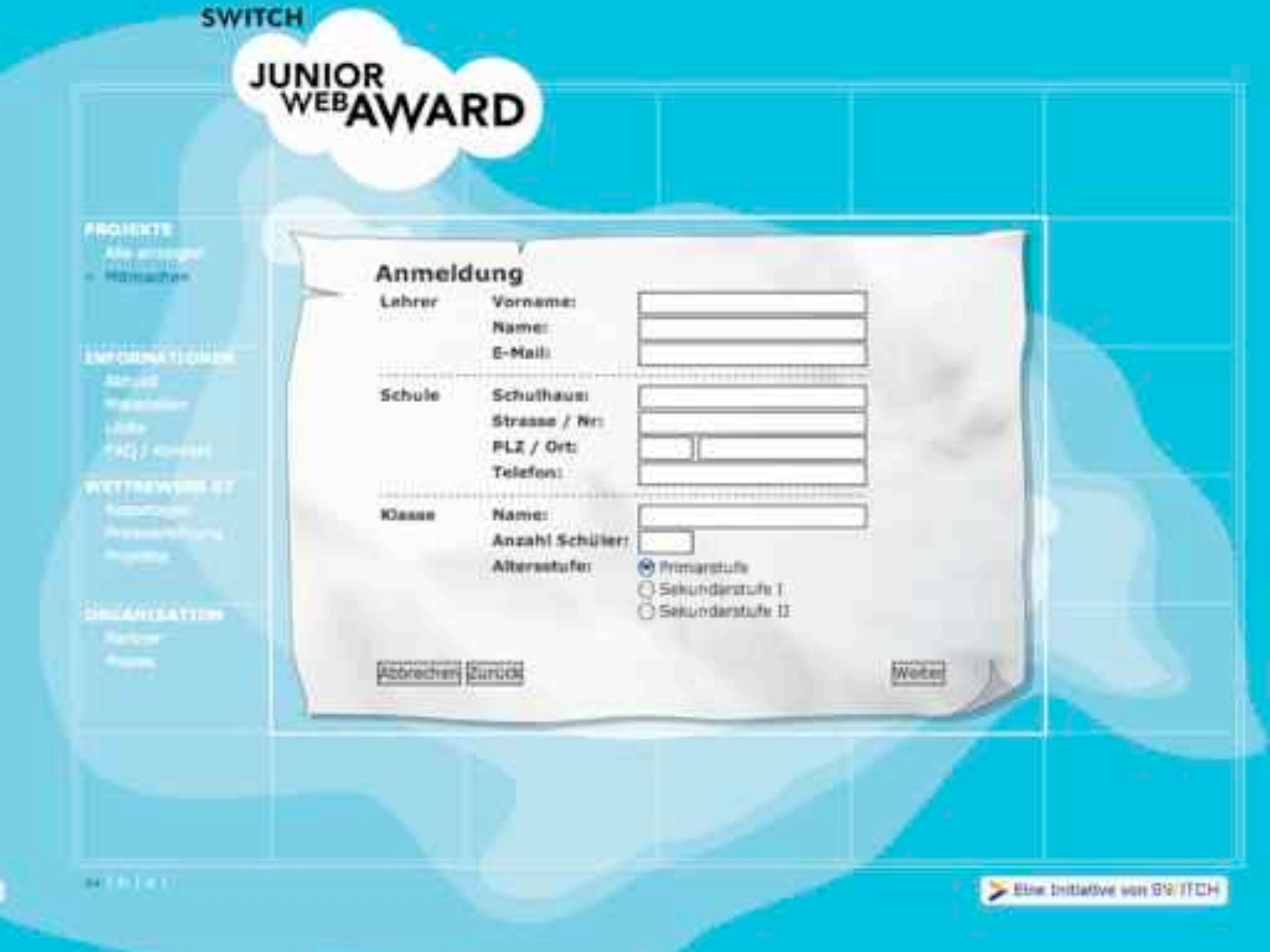
Moderne Programmierumgebungen unterstützen die Integration automatischer, erweiterbarer, und wiederverwendbarer Werkzeuge. Neue Werkzeuge können daher die verfügbare Funktionalität nutzen und automatisch Daten aus Programmen und Entwicklungsprozessen sammeln. Dies ermöglicht automatische Vorhersagen - und automatische Unterstützung für Software-Entwickler und -Entscheider. "Diese Aufgabe sollten Sie zusammen mit Karl lösen, da Sie wahrscheinlich an der 'Mailbox'-Klasse arbeiten müssen".

Slides



Aare

Workflow definition
and runtime system


SWITCH
JUNIOR WEB AWARD
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
Eine Initiative von SWITCH


CONCURSO
CLASSIC
ALBUMS
RESPONDE LA PREGUNTA
COMPLETA TUS DATOS

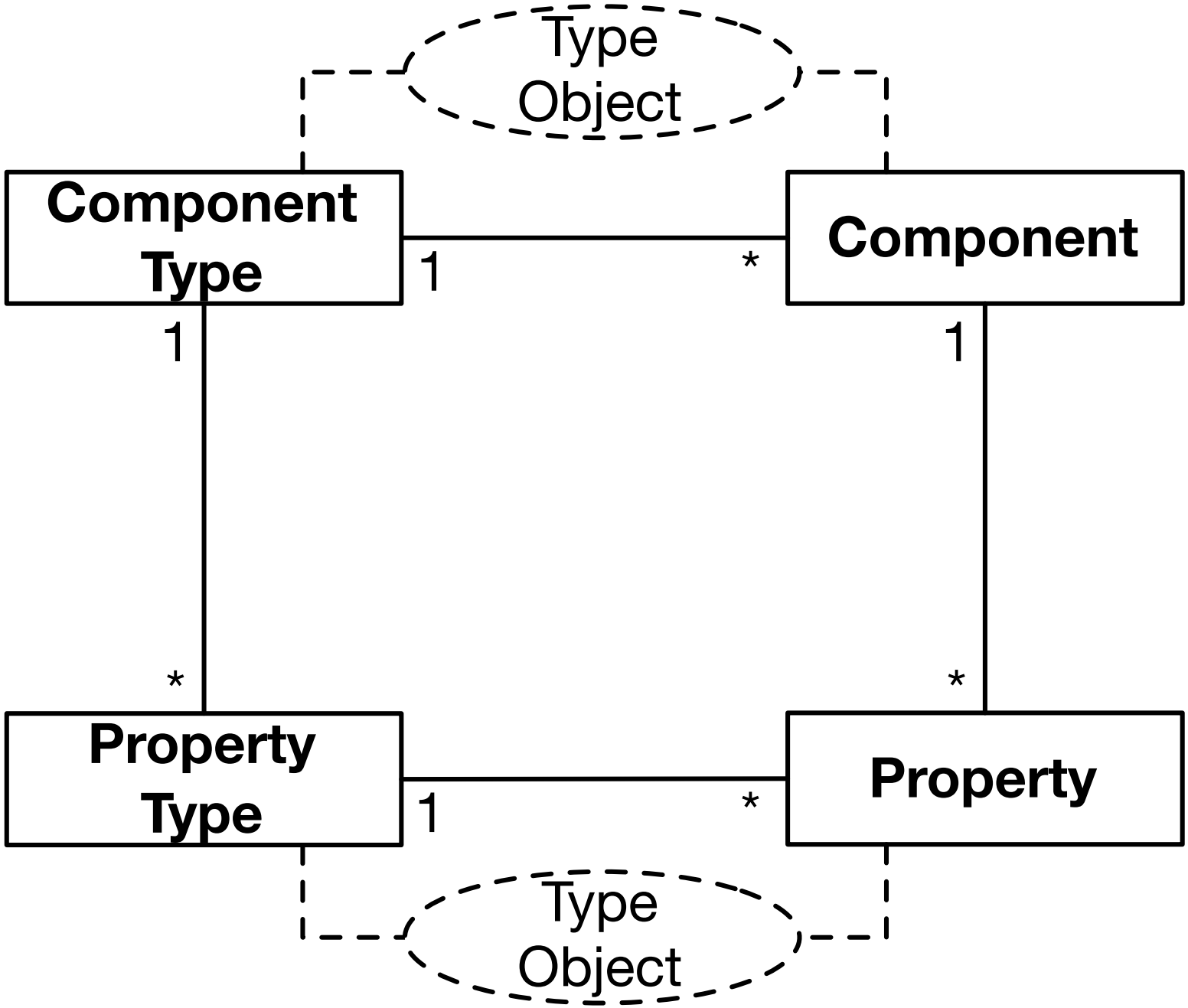
Magritte

Implementation Details

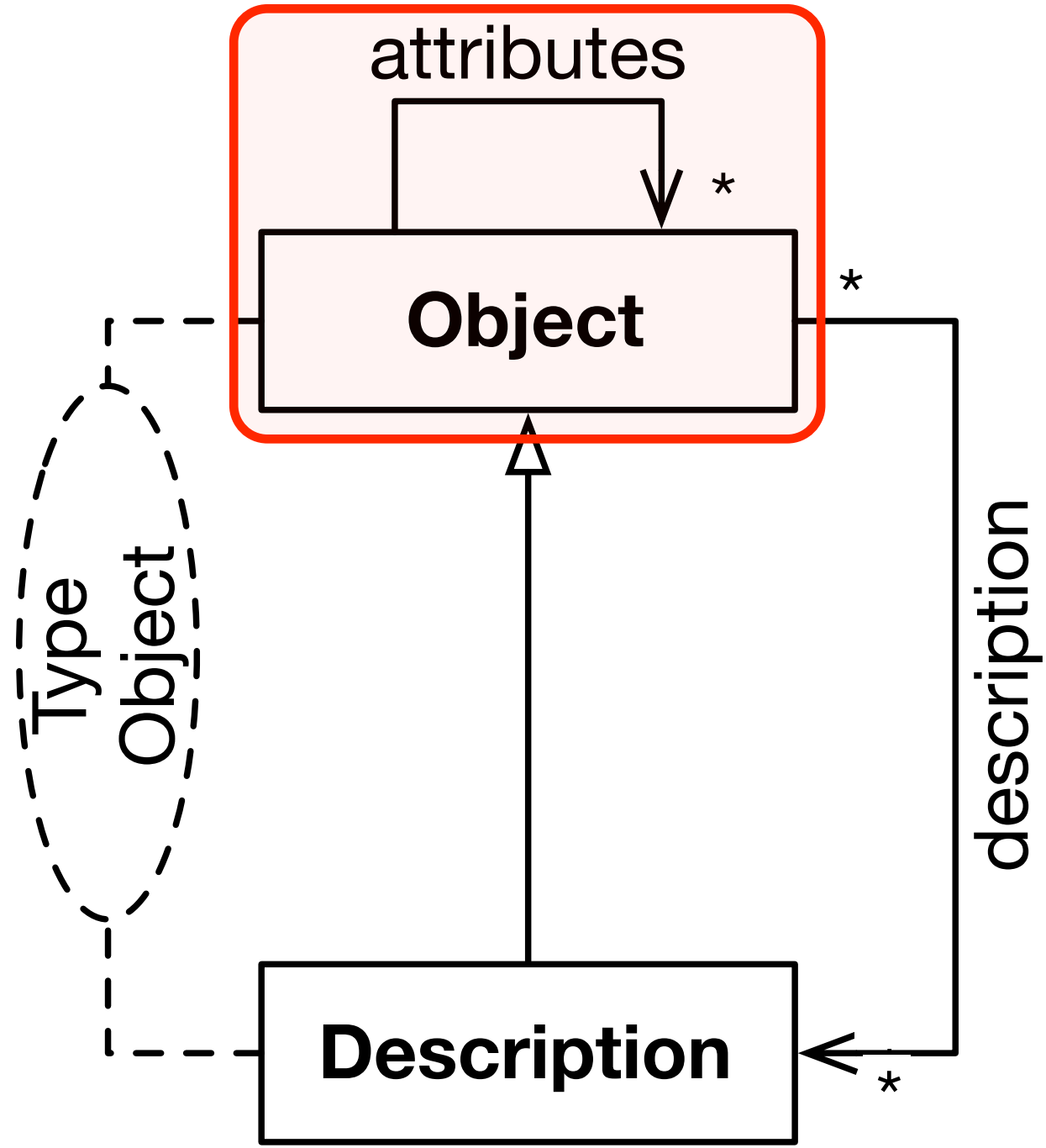
Describe once,
Get everywhere

Descriptions

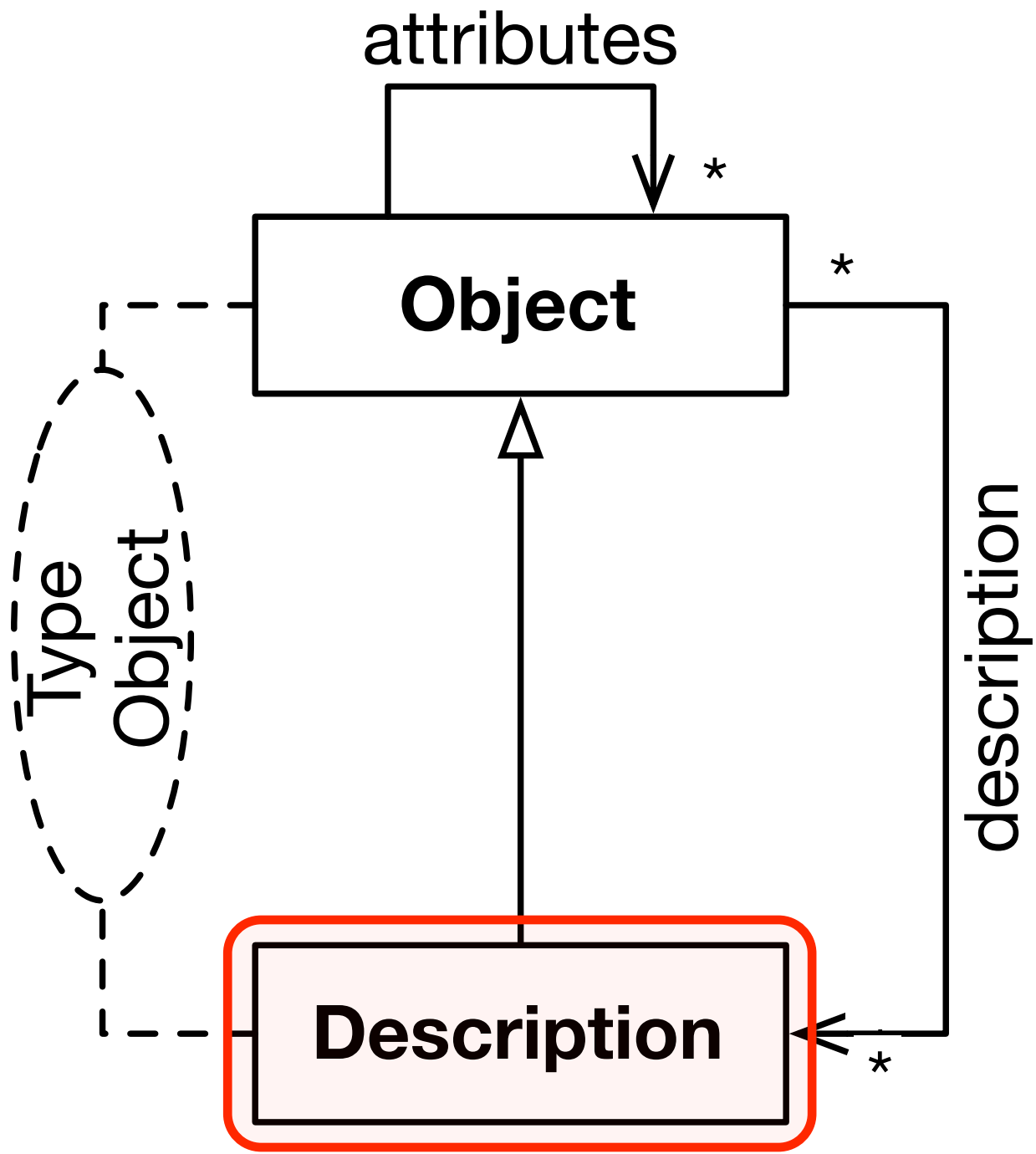
- Problem
  - Objects and their values all need to be treated differently.
- Example
  - `Boolean` and `String` are not polymorphic, therefore different code for certain operations is necessary.
- Solution
  - Introduce a descriptive hierarchy that can be instantiated, configured and composed.



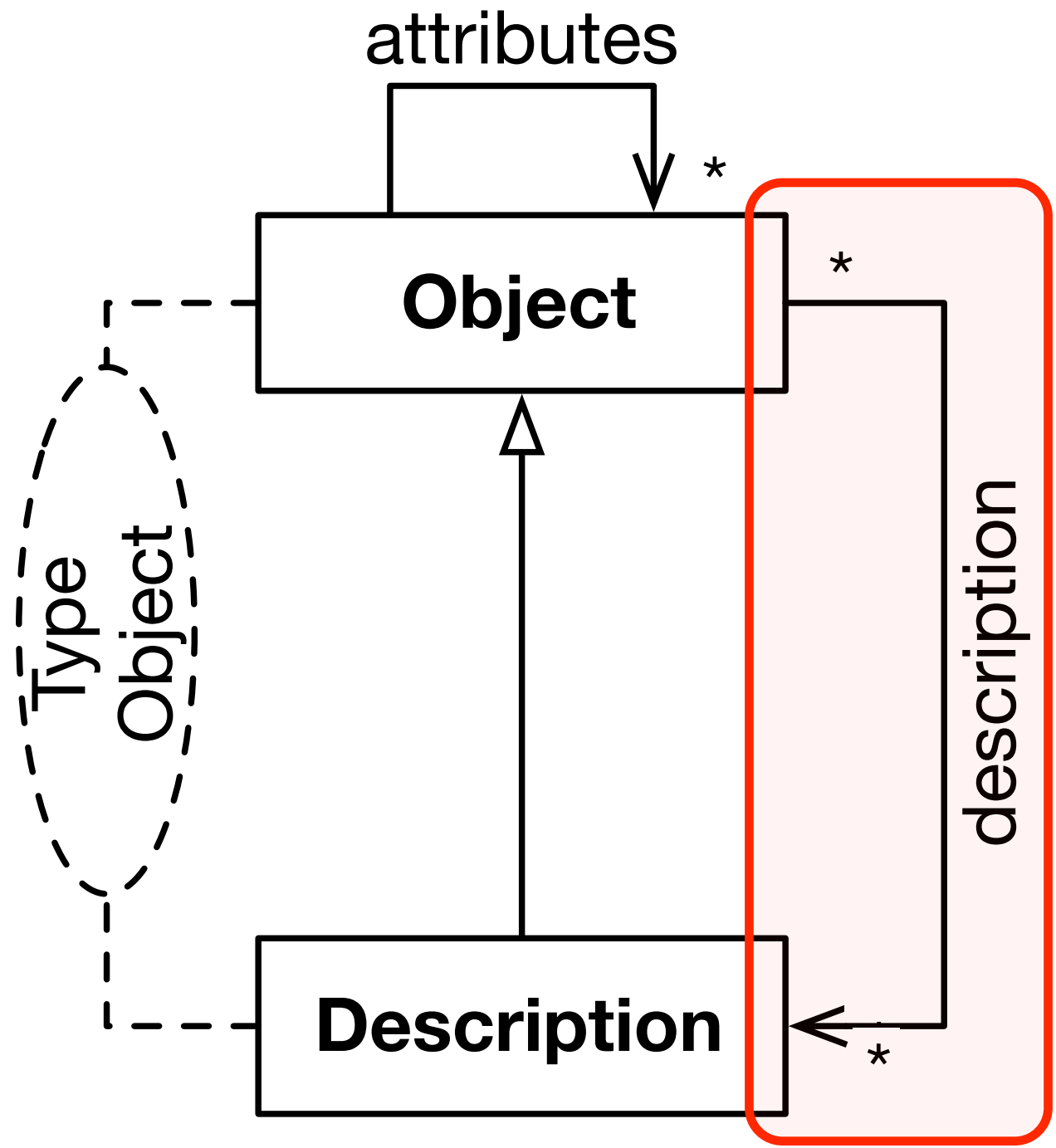
[Yoder *et al*, 2001] Architecture and design of adaptive object models



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



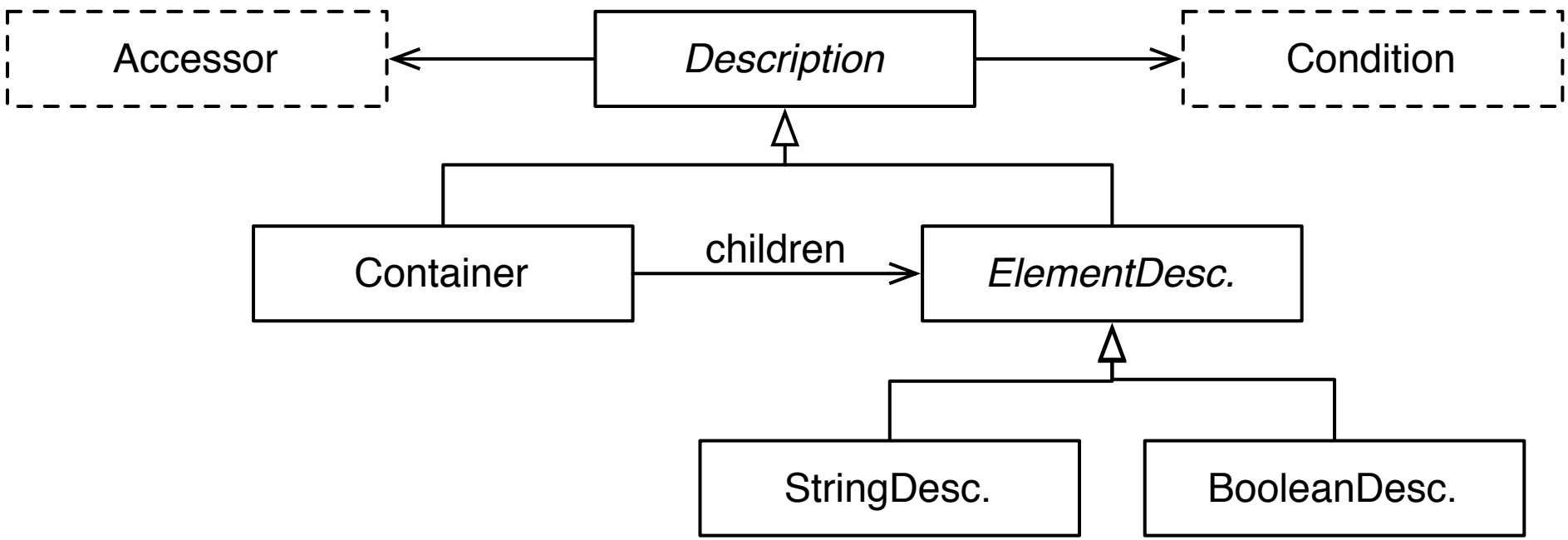
[Renggli *et al*, 2007] Magritte — A Meta-Driven Approach to Empower Developers and End Users



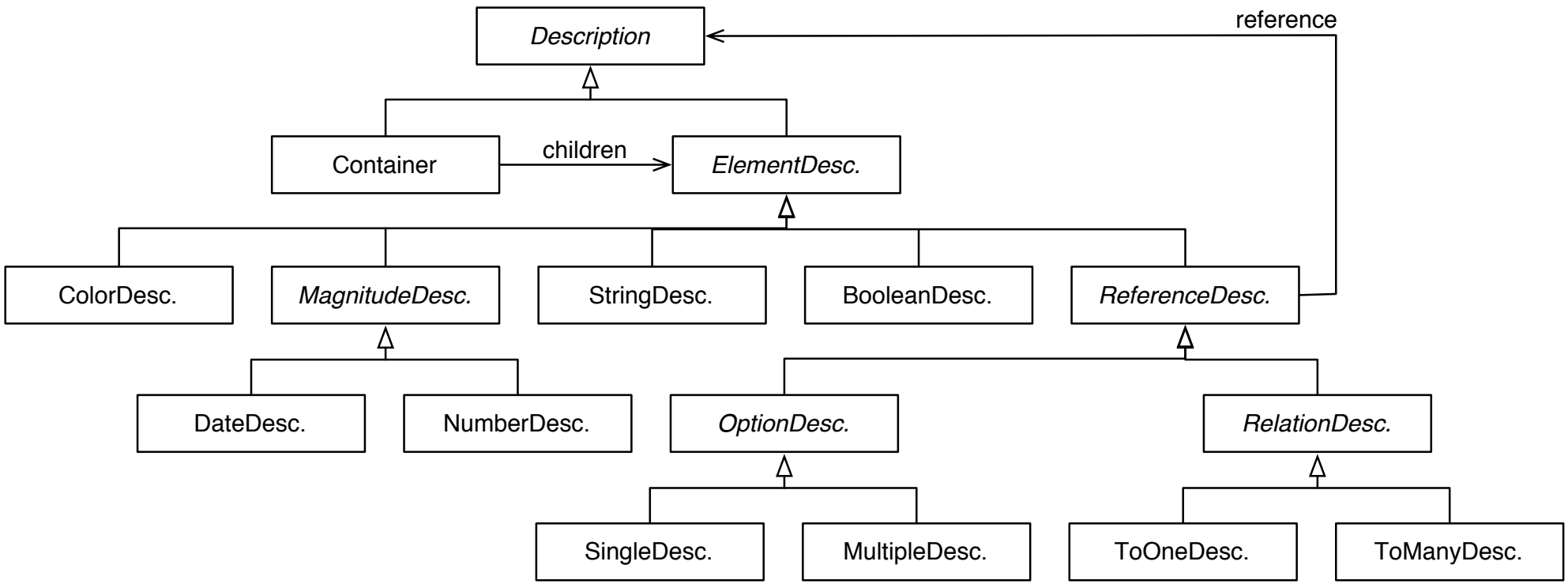
[Renggli *et al*, 2007] Magritte — A Meta-Driven Approach to Empower Developers and End Users

Descriptions

Composite pattern
to describe model-instances.



Descriptions

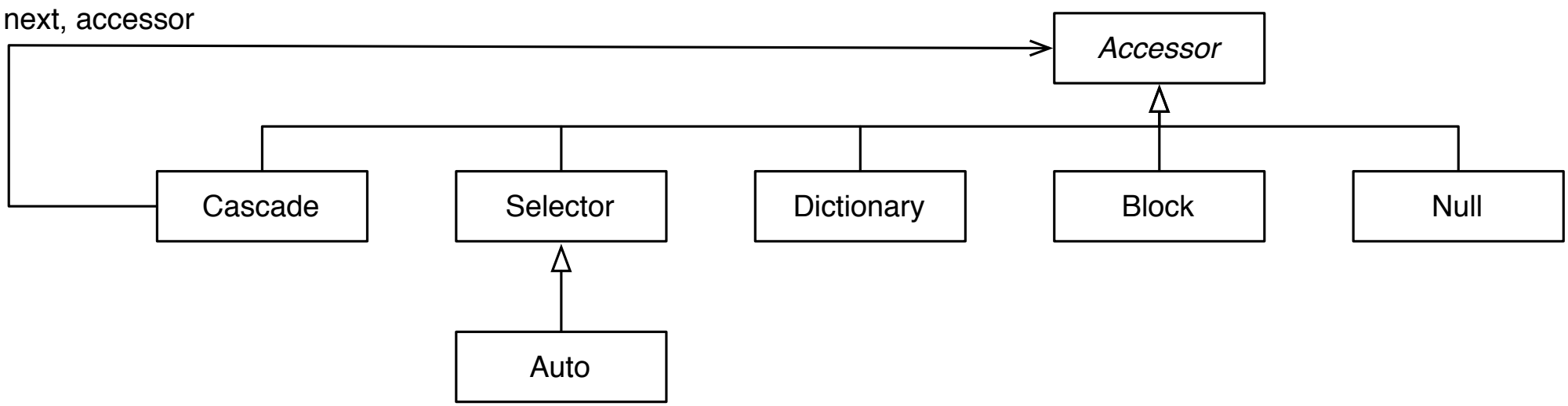


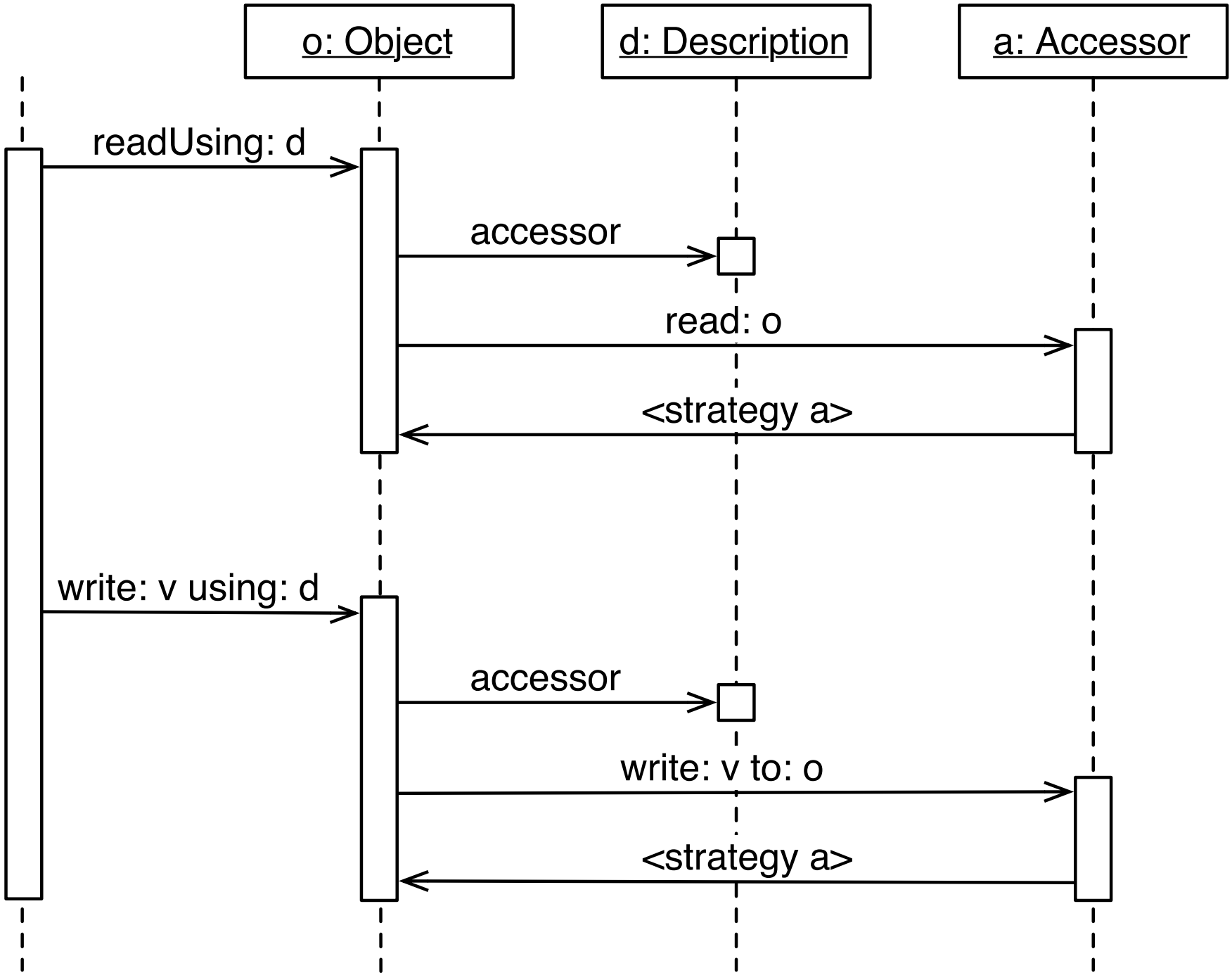
Accessors

- Problem
  - Data can be stored and accessed in different ways.
- Examples
  - Accessor methods, chains of accessor methods, instance-variables, dictionaries, derived 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.



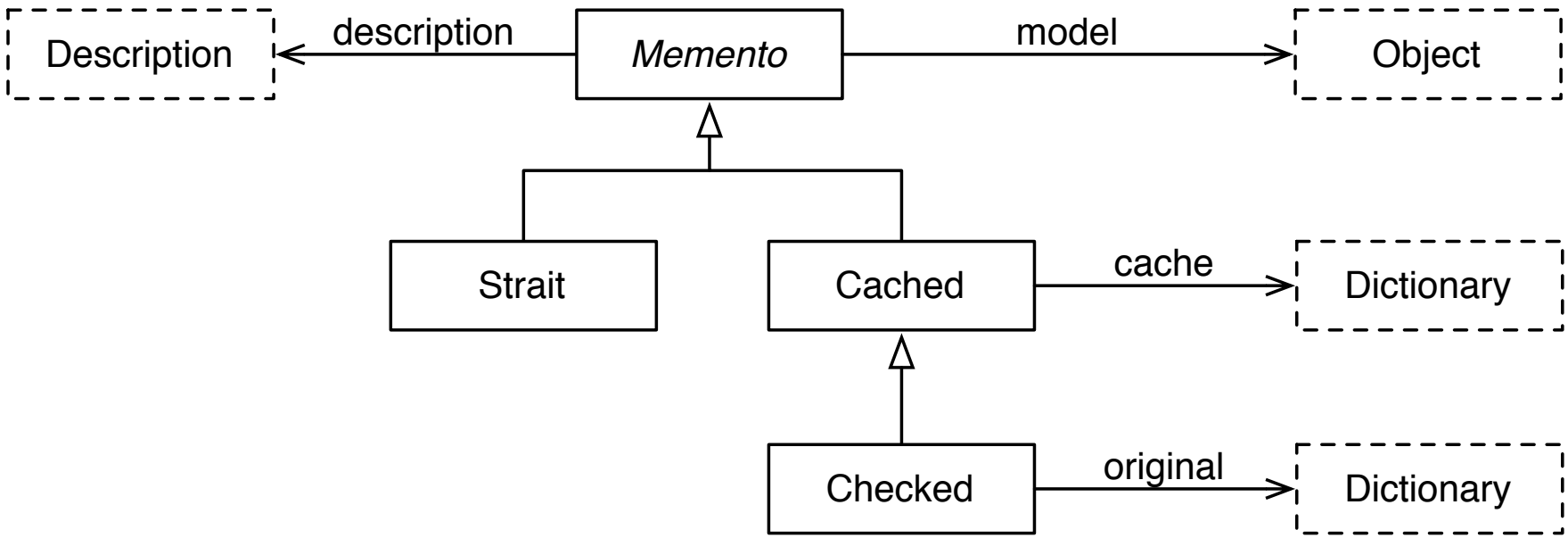

o: Object
d: Description
a: Accessor
readUsing: d
accessor
read: o
<strategy a>
write: v using: d
accessor
write: v to: o
<strategy a>

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.

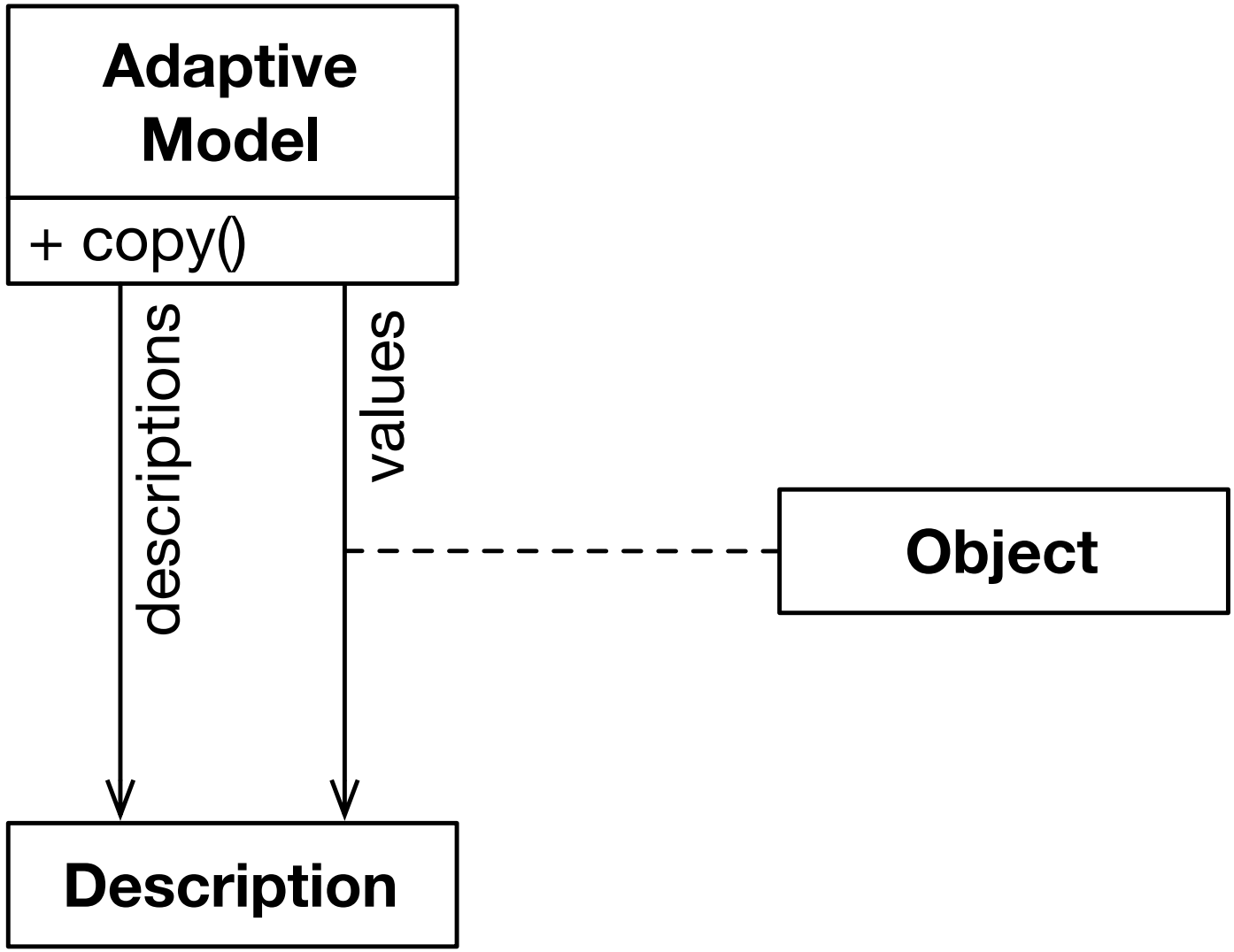


Magritte

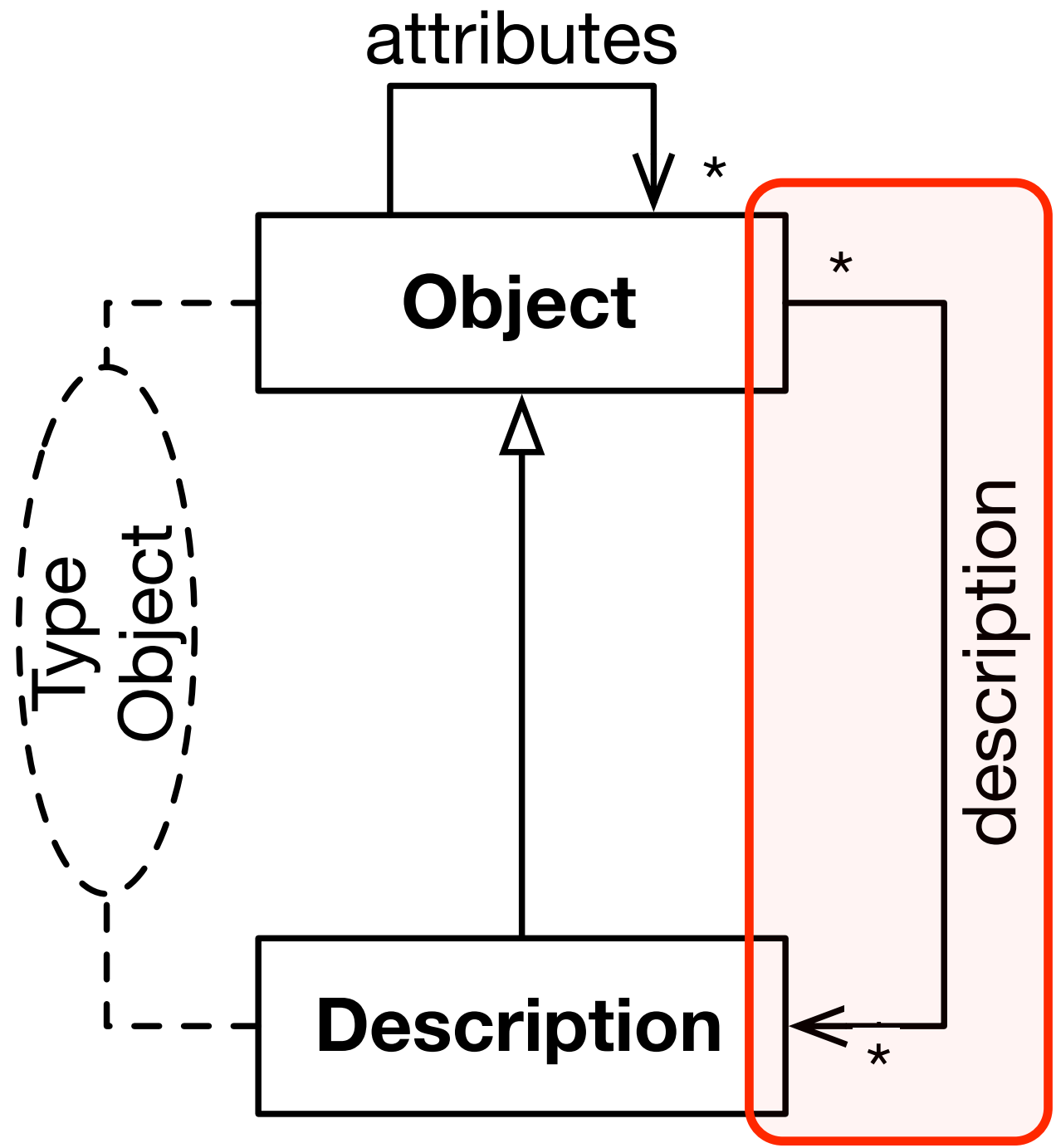
Adaptive Models

Describe once,
Get everywhere

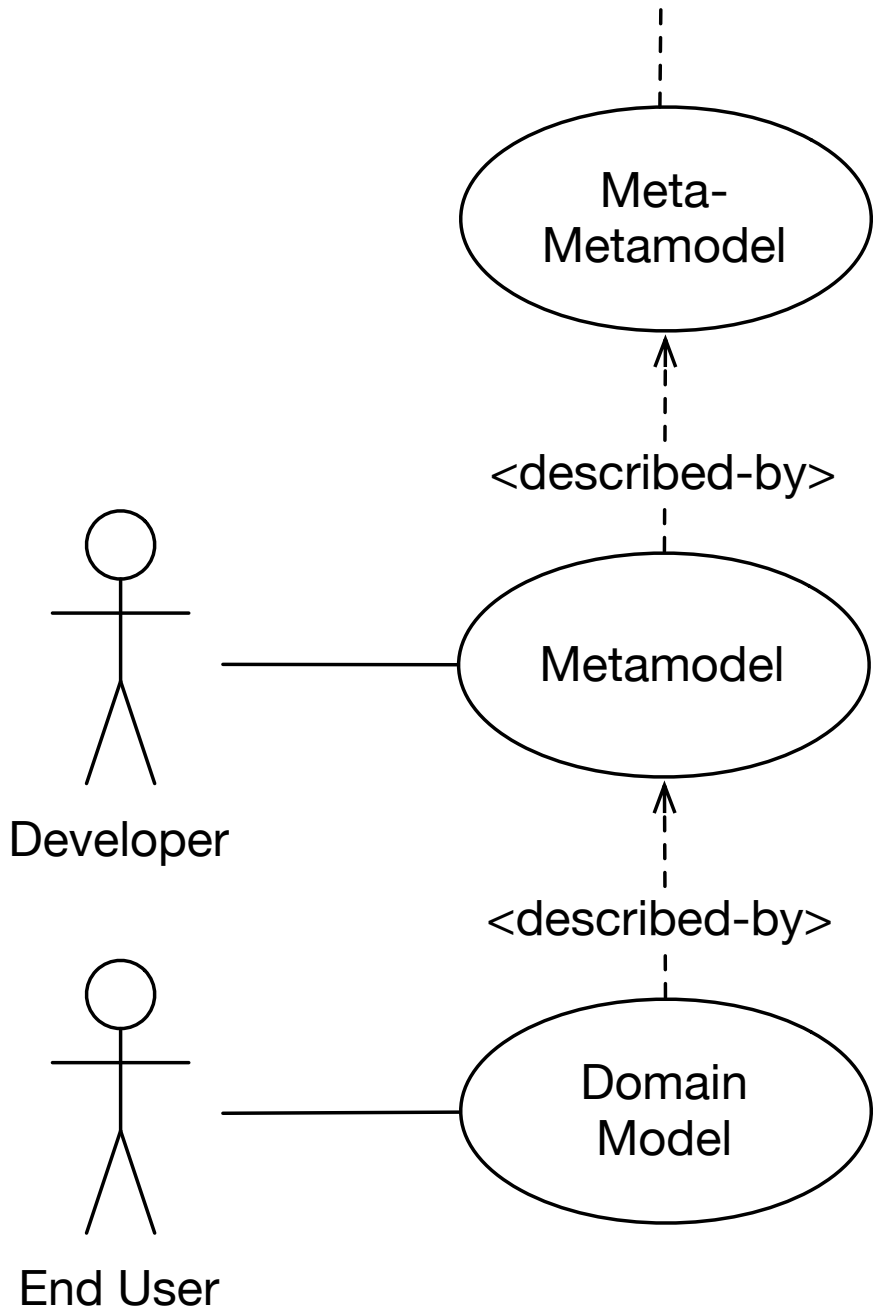
Rapidly changing requirements

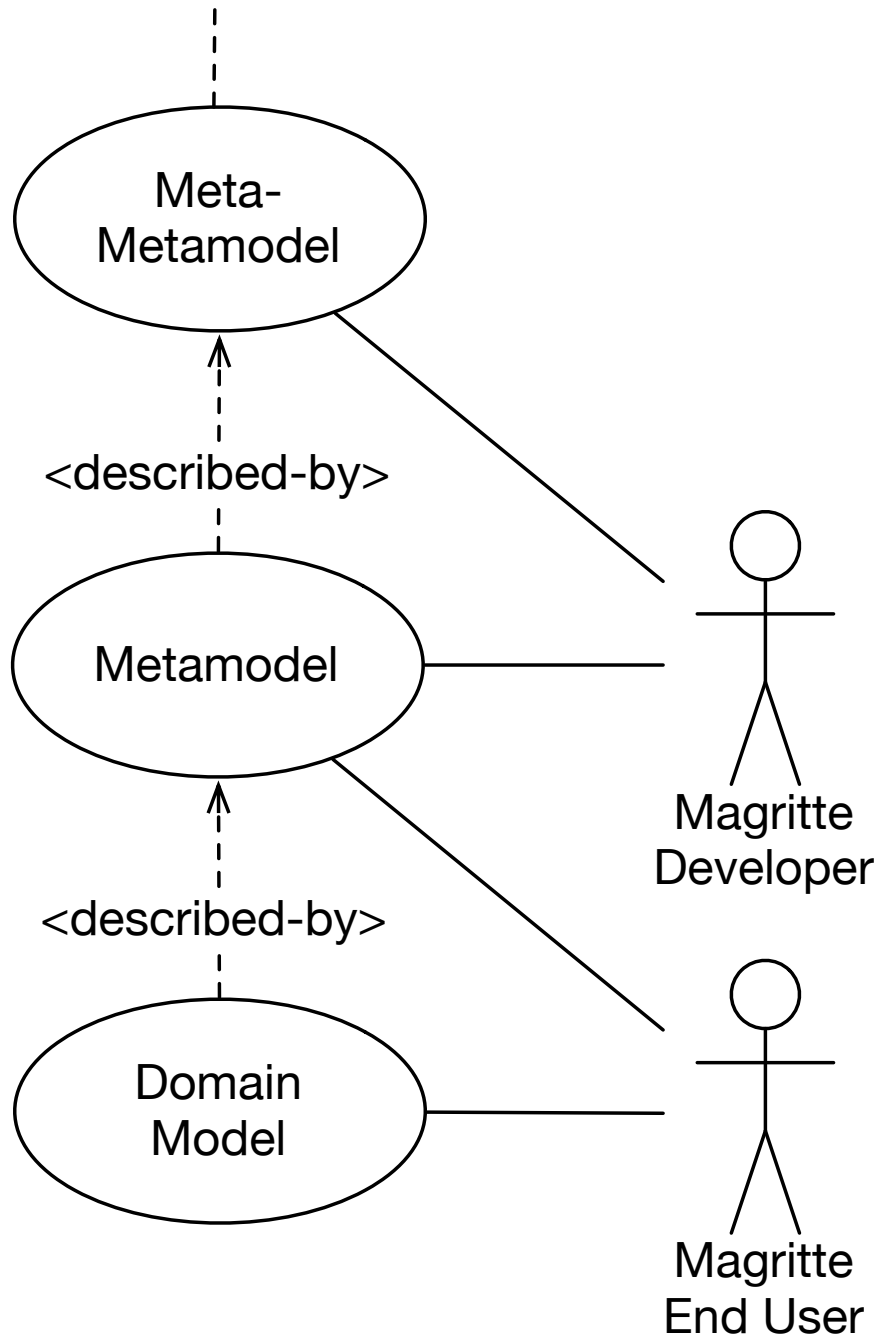

Adaptive Model
+ copy()
descriptions
values
Object
Description

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.