
# The Cha-Q Meta-Model: A Comprehensive, Change-Centric Software Representation

Vrije Universiteit Brussel

Universiteit Antwerpen

Coen De Roover, Christophe Scholliers, Angela Lozano, Viviane Jonckers

Javier Perez, Alessandro Murgia, Serge Demeyer

# Cha-Q Meta-Model

➜ to be shared by prototypes for analyzing, repeating, tracing changes

first interconnected representation of:

- ✓ state & evolution of the different entities of a software system
- ✓ each change to an entity that results in a new entity state
- ✓ system snapshots under control of a VCS

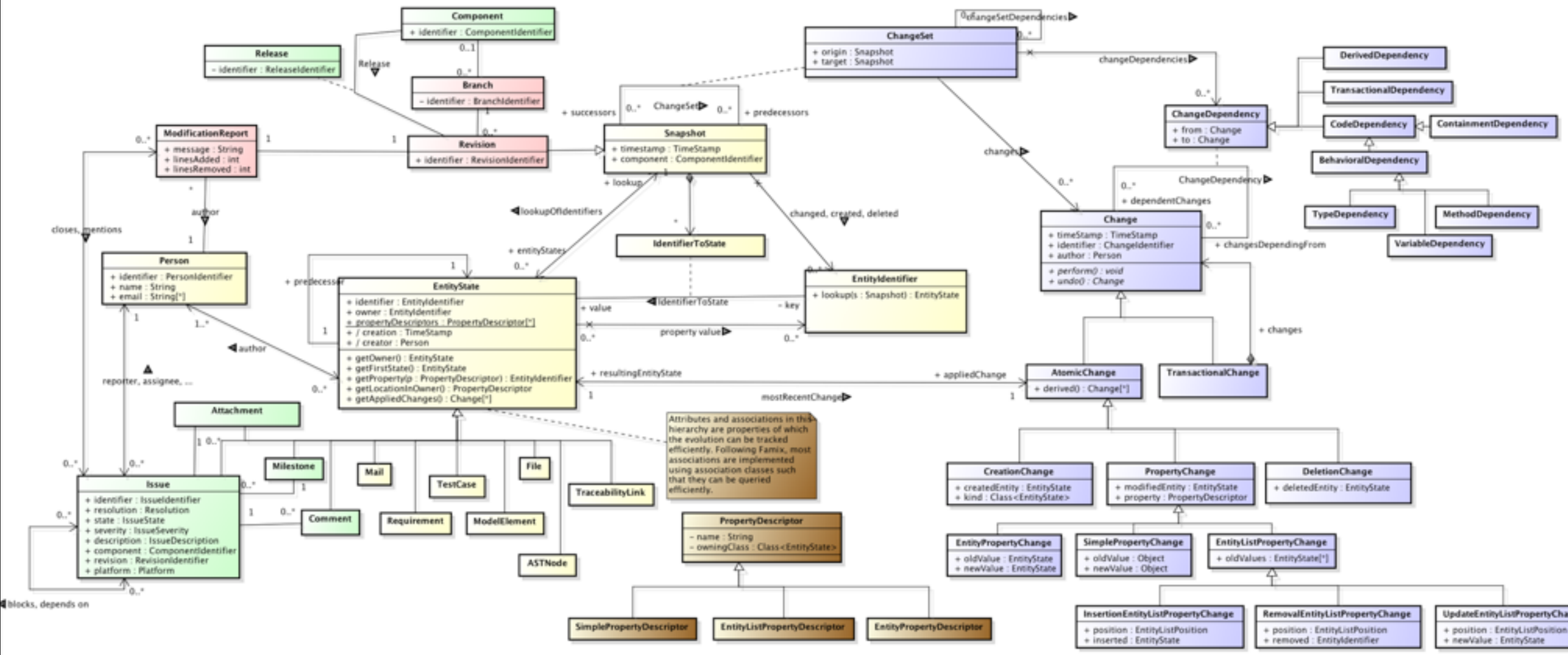
object-oriented (i.e., each concept and relation is represented by a class)

driven by positive experience with FAMIX, RING/C/H, Cheops

memory-efficient tracking of states

driven by poor scalability reported for Hismo and Syde

# Cha-Q Meta Model: Overview & Inspiration

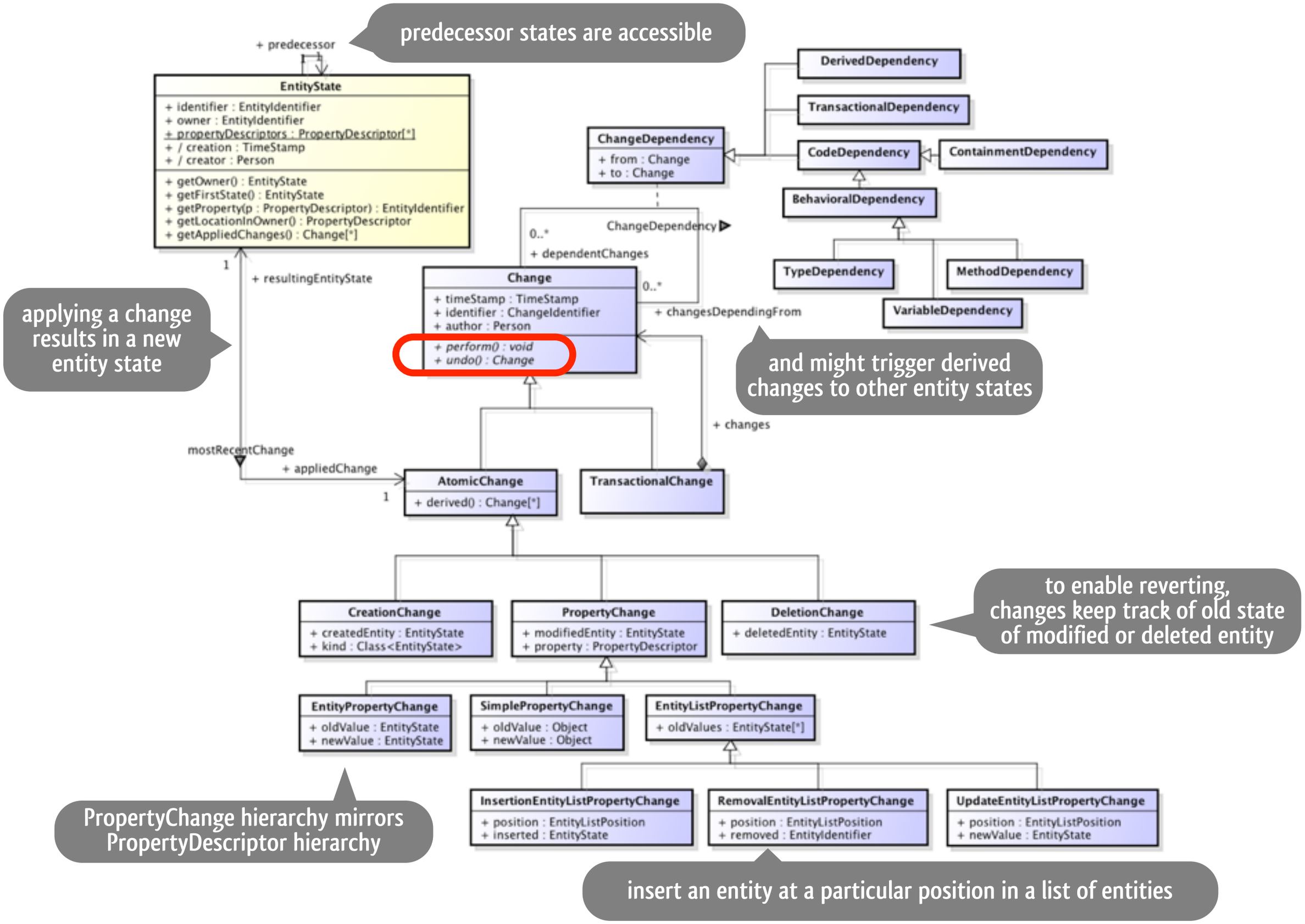



**state & evolution** — Ring/H, FAMIX, Hismo

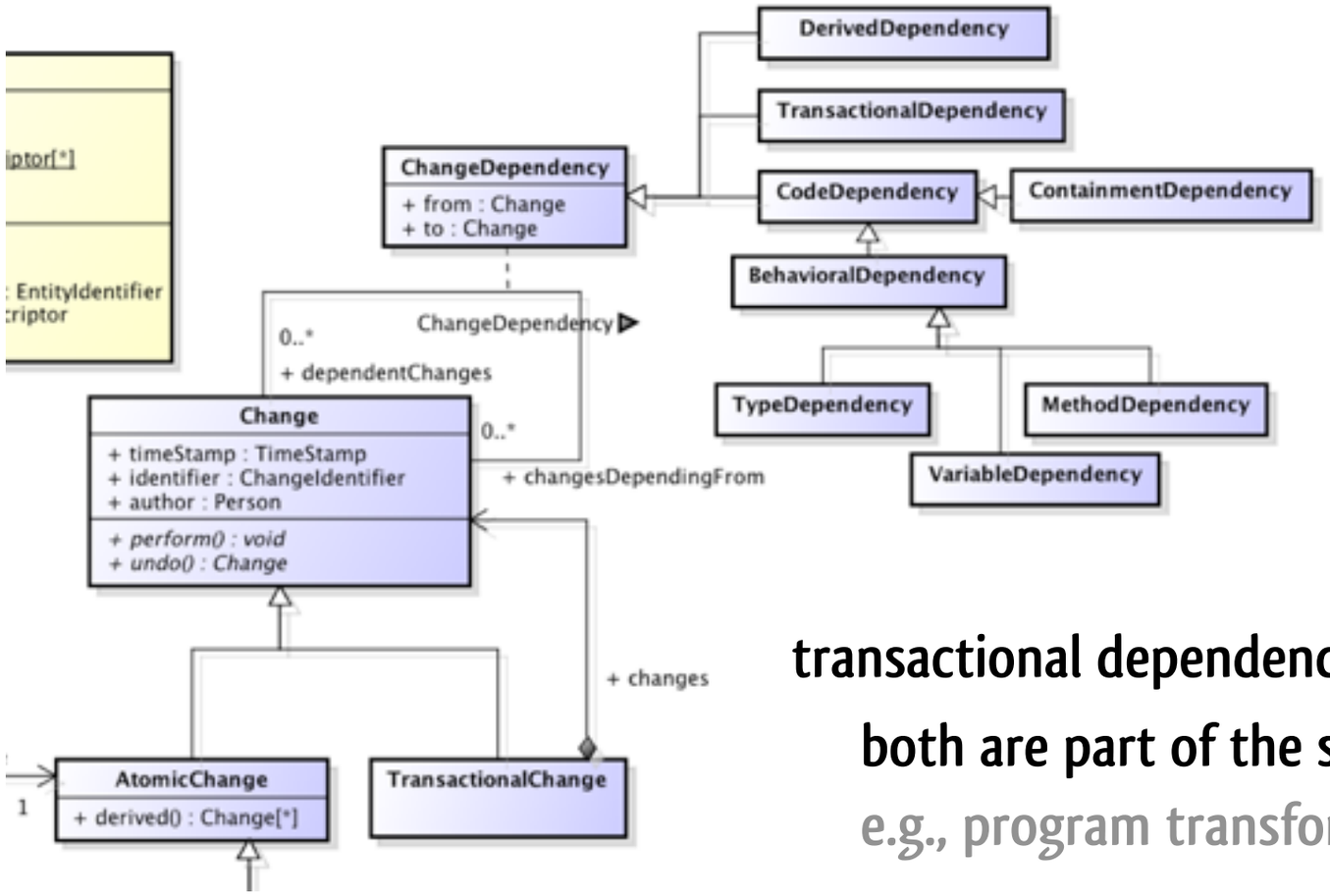
**change** — ChEOPS, UniCase, Ring/C, Spyware, Syde

**versioning** — Evolizer, STNACockpit

# Change: Overview of Representation



# Change: Change Dependencies




a change object c2 depends upon another change object c1 if applying c2 without c1 would violate invariants of the meta-model

transactional dependency:

both are part of the same transaction which schedules c1 before c2

e.g., program transformations

containment dependency:

the subject of c1 is the owner of the subject of c2

e.g., owner of a field declaration is its declaring class

type/method/variable dependency

c1 creates a type/method/variable declaration referred to by c2

e.g., type of a parameter declaration

# Evolution: Overview of Representation

accumulates the effect of one or more changes

represents the state of the system at a particular point in time, as seen by a particular developer

each EntityState knows in which Snapshot to start looking up EntityIdentifiers

enables tracking the evolution of an entity throughout its lifetime

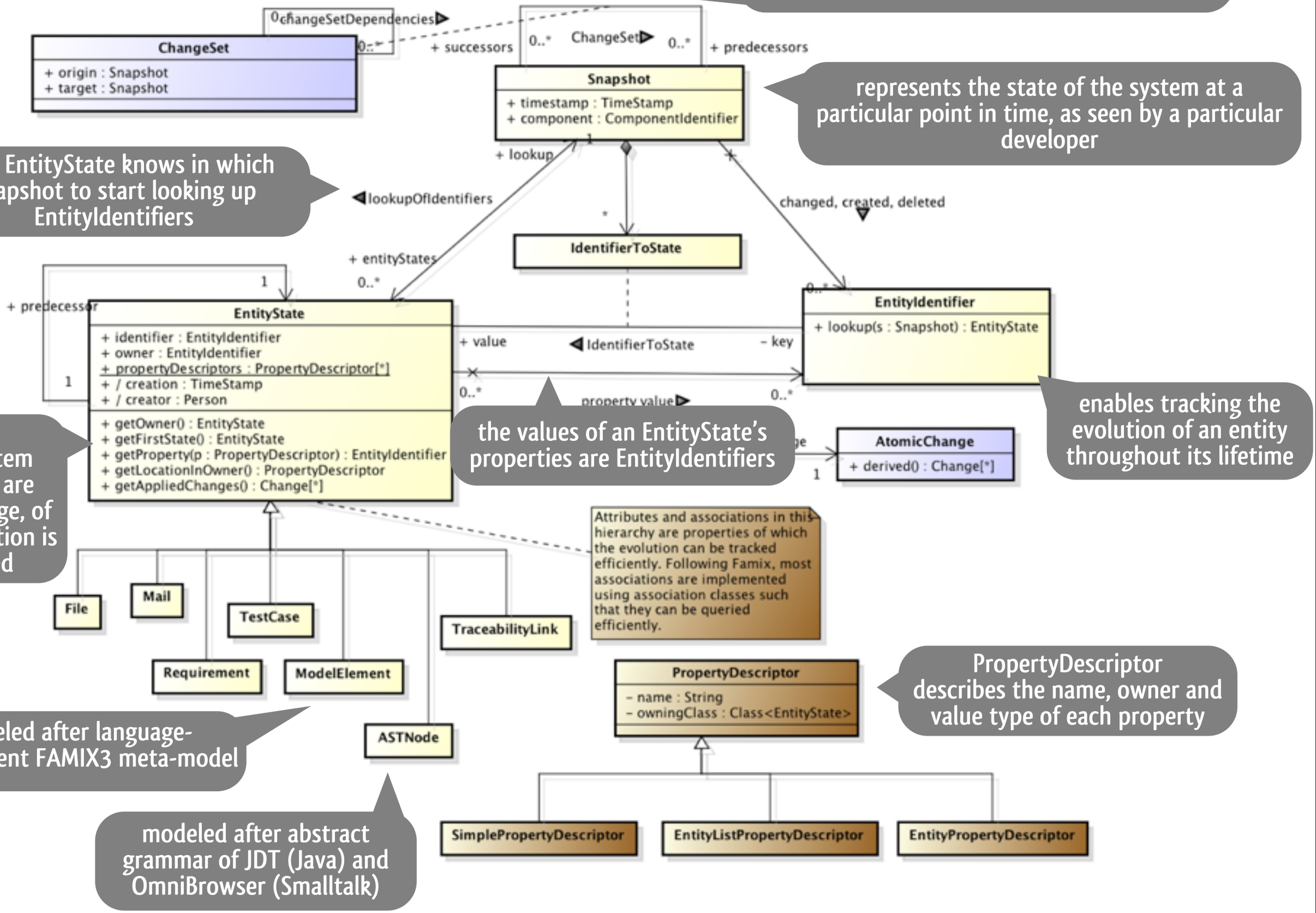
the values of an EntityState's properties are EntityIdentifiers

subclasses represent system artefacts that are subject to change, of which the evolution is to be tracked

PropertyDescriptor describes the name, owner and value type of each property

modeled after language-independent FAMIX3 meta-model

modeled after abstract grammar of JDT (Java) and OmniBrowser (Smalltalk)



# Evolution: Memory-Efficient State Tracking

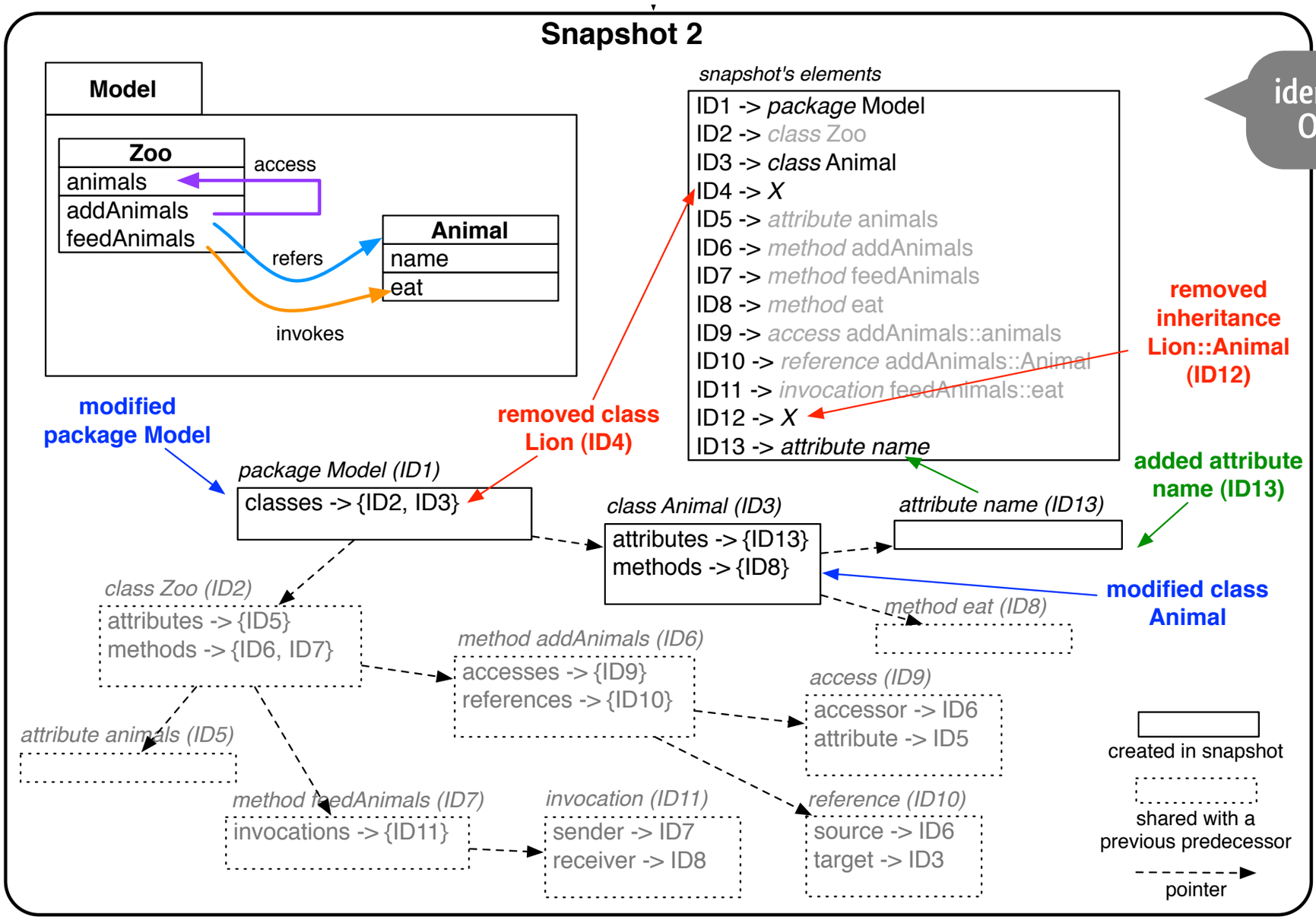
**copying an entity each time its state changes is expensive**

for snapshot-centric Hismo model: 70min for full snapshot copy of 350MB

for change-centric Syde model: 3GB for SVN repo of 78MB

**selective cloning is difficult to implement**

all entities are transitively interconnected




identifier-based technique used by Orion and Ring/H meta-models

**performing a PropertyChange:**

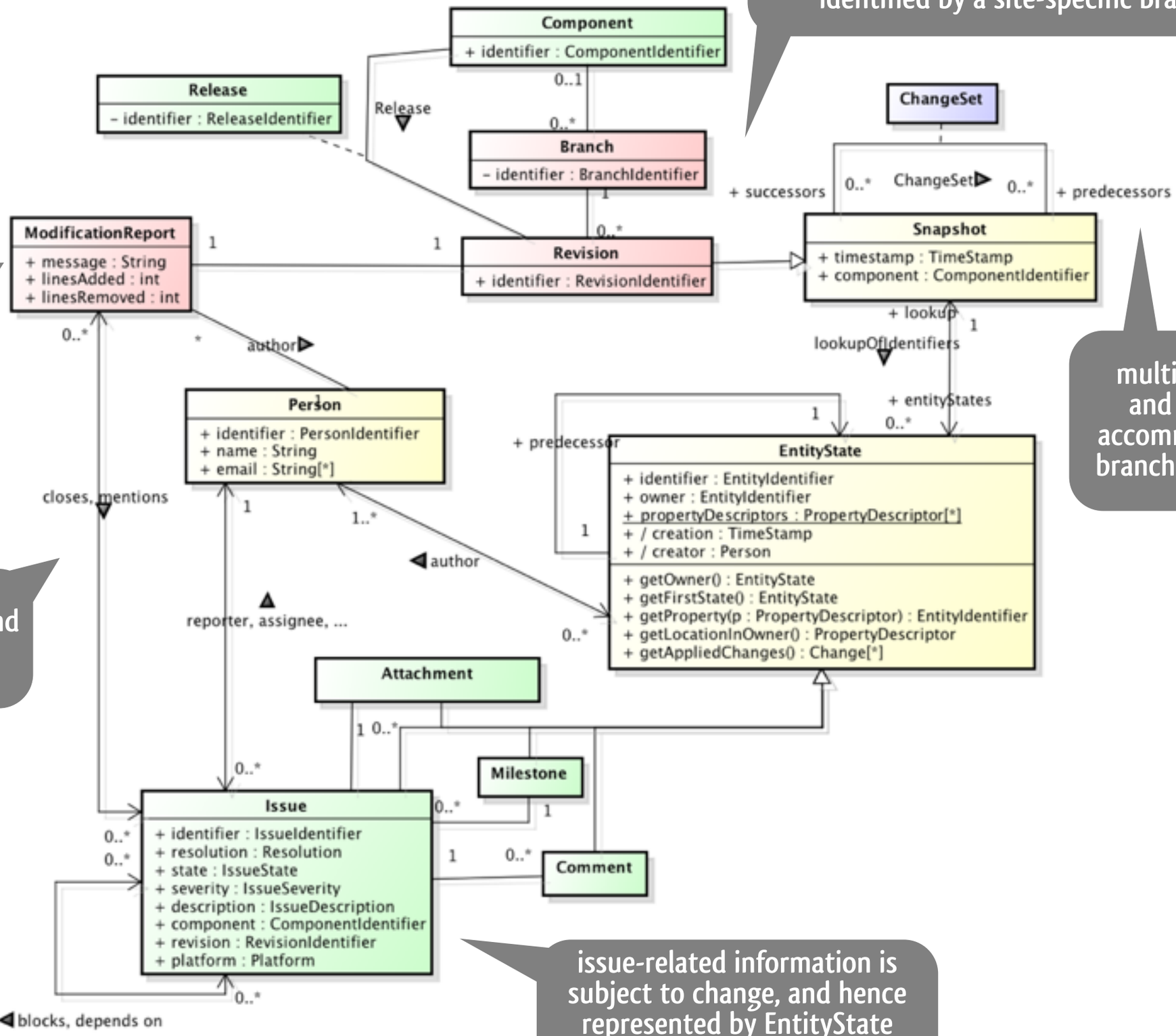
1/ clone EntityState

2/ update snapshot's mapping from ID to clone

3/ update property value in clone

[Uquillas 12]

# Versioning & Issues: Overview of Representation




revisions can be associated with a logical branch, identified by a site-specific BranchIdentifier

a modification report is associated with each snapshot placed under version control

multiple successors and predecessors accommodate physical branching and merging

reports can mention and close reported issues

issue-related information is subject to change, and hence represented by EntityState subclasses

# Implementation Highlight: Annotation Metadata

**v1**

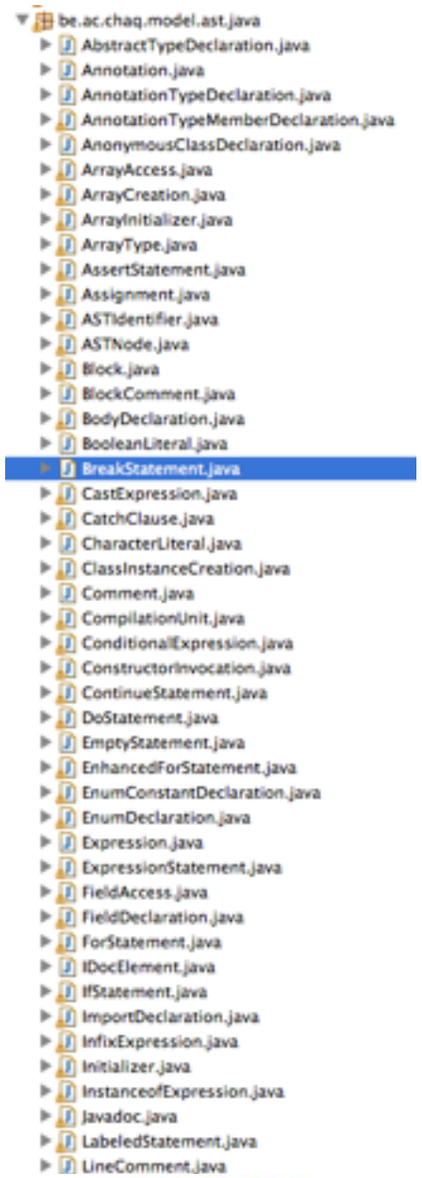
```
public class BreakStatement extends Statement {
    @EntityProperty(value = SimpleName.class)
    private EntityIdentifier label;

    public EntityIdentifier getLabel() {
        return label;
    }

    public void setLabel(EntityIdentifier label) {
        this.label = label;
    }
}
```

annotation for properties of which evolution is to be tracked

**v2**

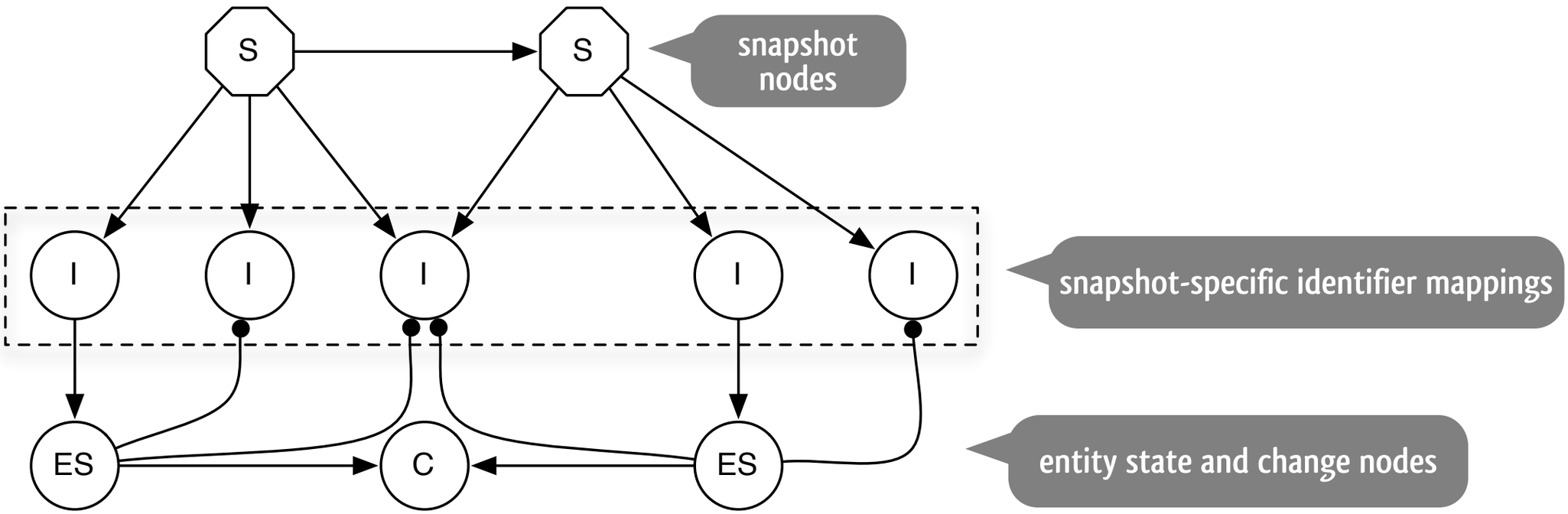
```
public class BreakStatement extends Statement {
    @EntityProperty(value = SimpleName.class)
    private EntityIdentifier<SimpleName> label;

    public EntityIdentifier<SimpleName> getLabel() {
        return label;
    }

    public void setLabel(EntityIdentifier<SimpleName> label) {
        this.label = label;
    }
}
```

more typesafe version,
obtained through our own rewriting tool!



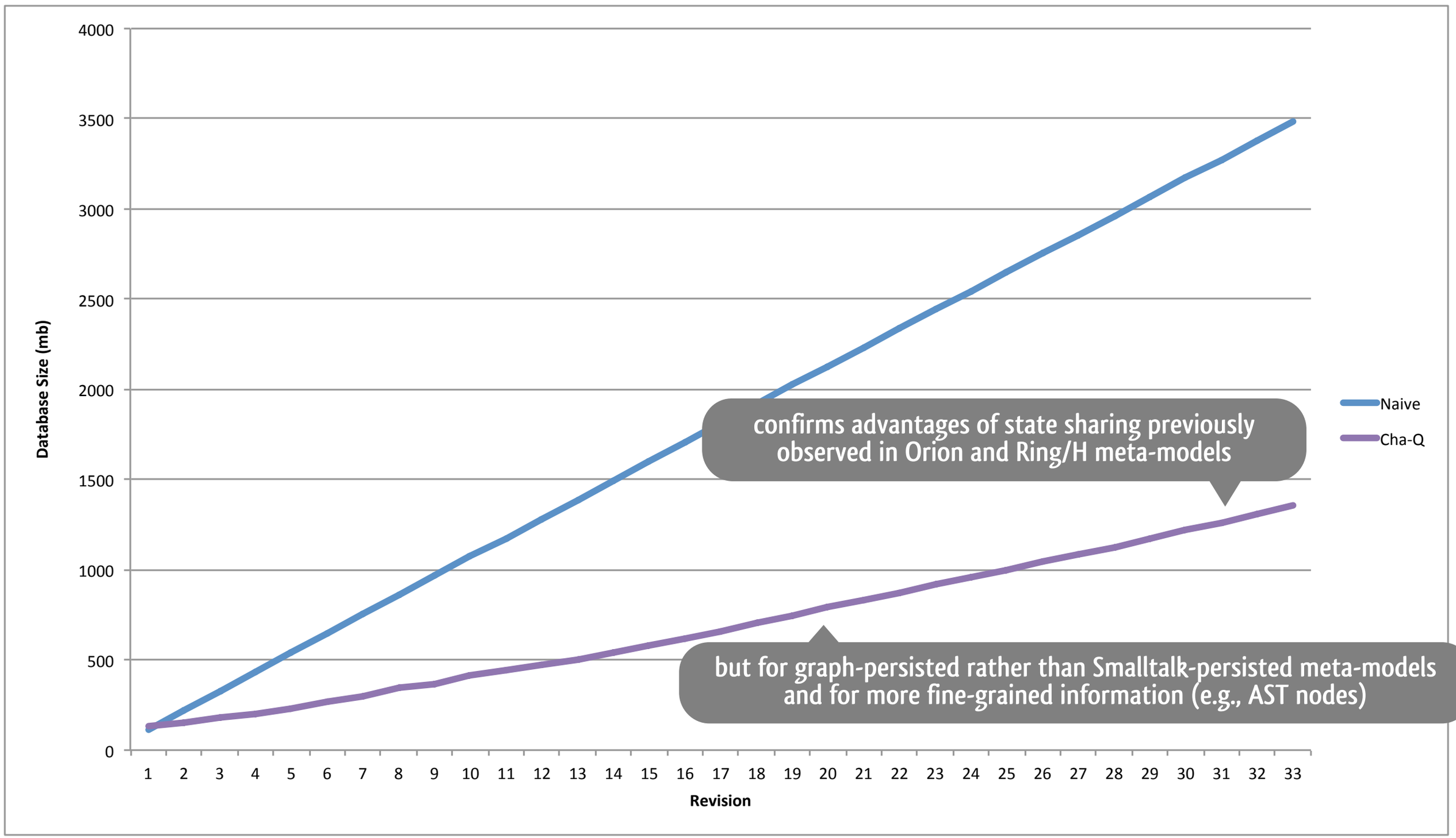

# Implementation Highlight: Persistence

@EntityProperty + Neo4j + weak references



# Implementation Highlight: Disk Footprint

evolution of Exapus project (http://github.com/cderoove/Exapus)
single revision: 194149 nodes, 223979 properties, and 194147 relationships of 32 distinct types
on average: 22,5 files per revision changed

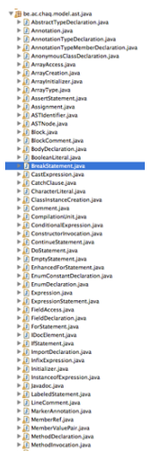


# Conclusions

defines the first interconnected representation of:

- ✓ state & evolution of the different entities of a software system
- ✓ each change to an entity that results in a new entity state
- ✓ system snapshots under control of a VCS

implementation highlights:



+

=

- ✓ memory-efficient
- ✓ familiar OO API for tool builders

http://soft.vub.ac.be/chaq/ for upcoming tools that share our meta-model!