

Design Patterns Application with MDE

Prof. Jean-Marc Jézéquel

(Univ. Rennes 1 & INRIA)

Triskell Team @ IRISA

Campus de Beaulieu

F-35042 Rennes Cedex

Tel : +33 299 847 192 Fax : +33 299 847 171

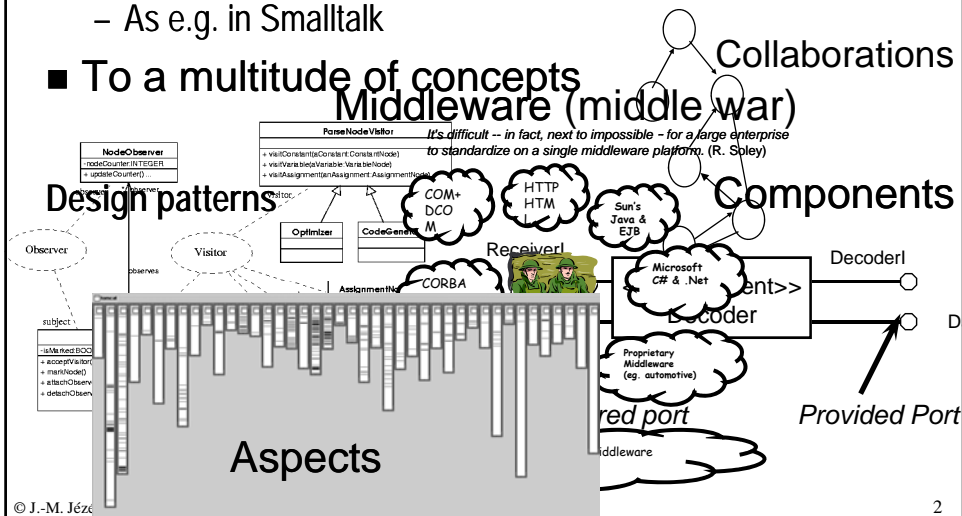
e-mail : jezequel@irisa.fr

<http://www.irisa.fr/prive/jezequel>

Once upon a time... software development looked simple

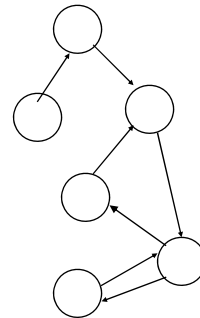
- From the object as the *only* one concept
 - As e.g. in Smalltalk

- To a multitude of concepts



Collaborations

- Objects should be as simple as possible
 - To enable modular understanding
- But then where is the complexity?
 - It is in the way objects interact!
 - Cf. Collaborations as a standalone diagram in UML
(*T. Reenskaug's works*)



© J.-M. Jézéquel, 2003

3

Design Patterns

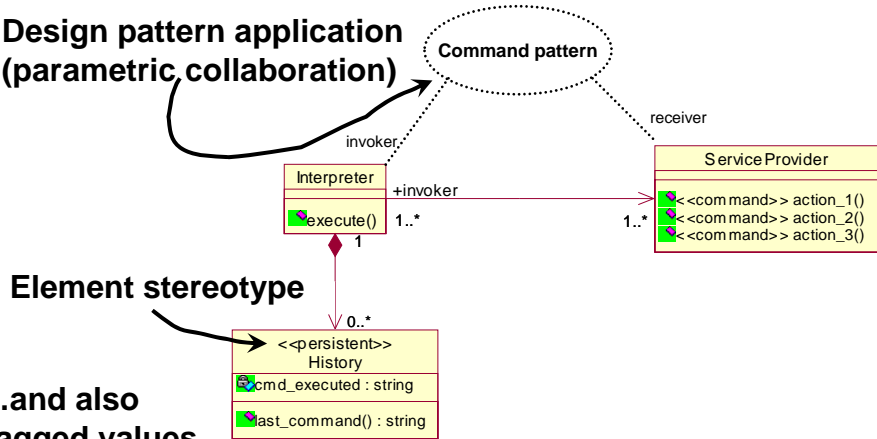
- Embody *architectural know-how* of experts
- As much about problems as about solutions
 - pairs problem/solution in a context
- About non-functional forces
 - reusability, portability, and extensibility...
- Not about classes & objects but *collaborations*
 - Actually, design pattern applications *are* parameterized collaborations

© J.-M. Jézéquel, 2003

4

Embedding implicit semantics into a model

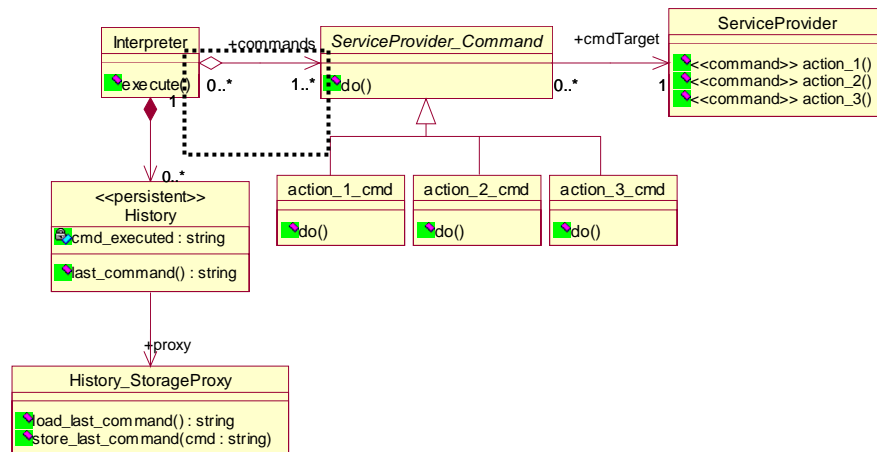
Design pattern application
(parametric collaboration)



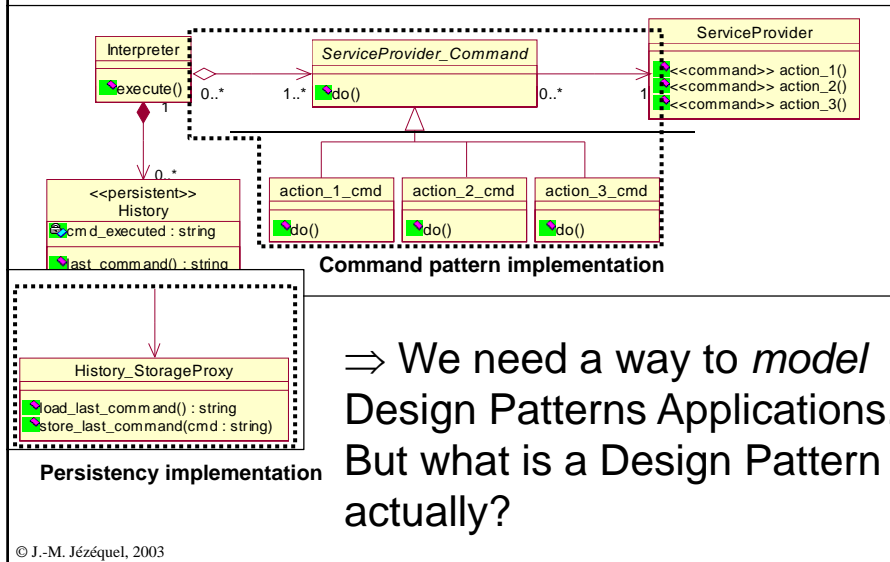
Element stereotype

...and also
Tagged values
& Contracts

...and the result we want...



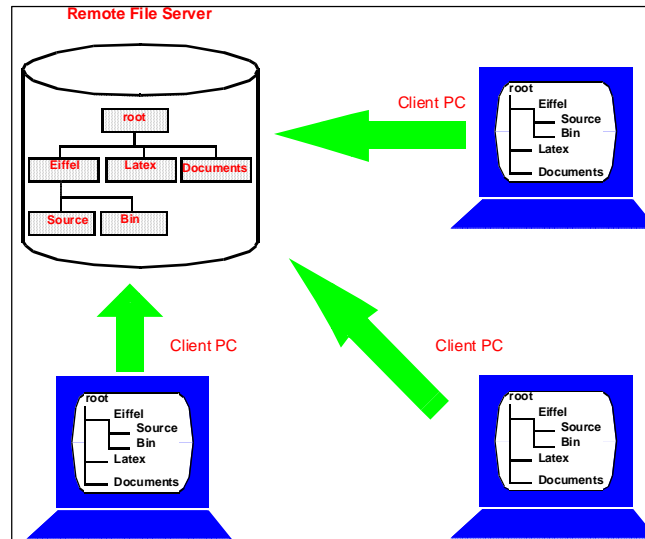
How To: Automatic Model Transformations



Origin of Design Patterns

- GoF's Book: A catalog
 - *Design Patterns: Elements of Reusable Object-Oriented Software* (Gamma, Helm, Johnson, Vlissides). Addison Wesley, 1995
- Earlier works by Beck, Coplien and others...
- Origin of Patterns in Architecture (C. Alexander)
 - *Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to this problem in such a way that you can use this solution a million times over, without ever doing it the same way twice.*

Example: A Distributed File System



© J.-M. Jézéquel, 2003

9

The Observer Pattern

■ Intent

- *Dependency from a subject to observers so that when the subject changes state, observers are notified*

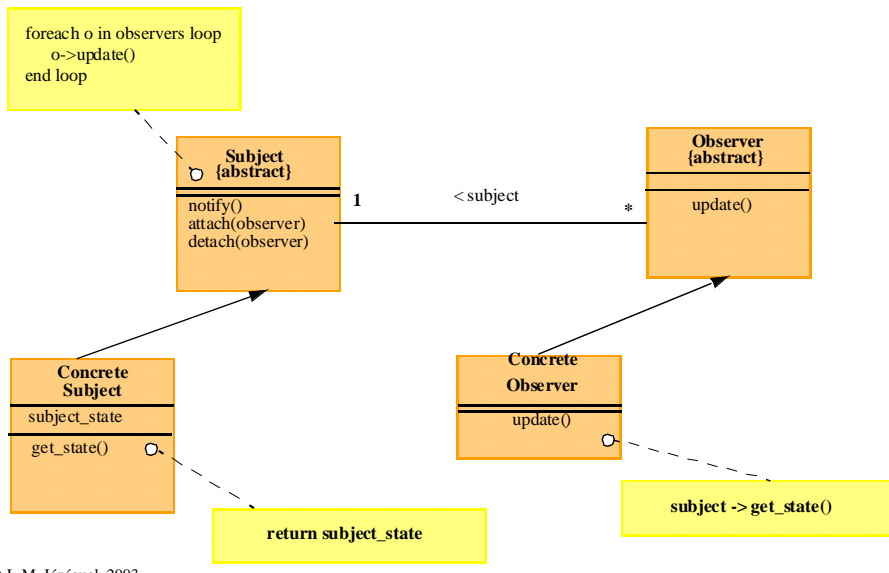
■ Key constraints

- *Any number of observers*
- *Each observer can react specifically to the notification of change*
- *The subject should be decoupled from the observers (dynamic add/remove of observers)*

© J.-M. Jézéquel, 2003

10

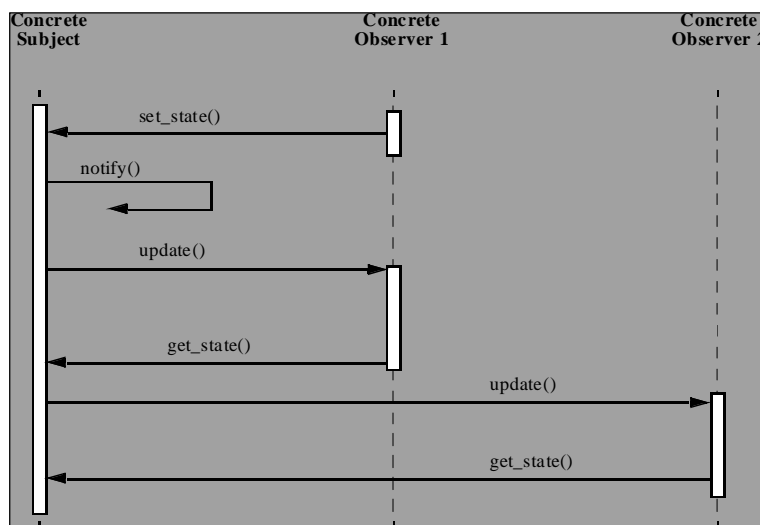
Structure of the Observer Pattern



© J.-M. Jézéquel, 2003

11

Collaborations in the Observer Pattern



© J.-M. Jézéquel, 2003

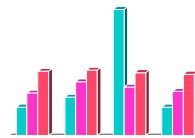
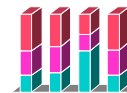
12

Another Problem...

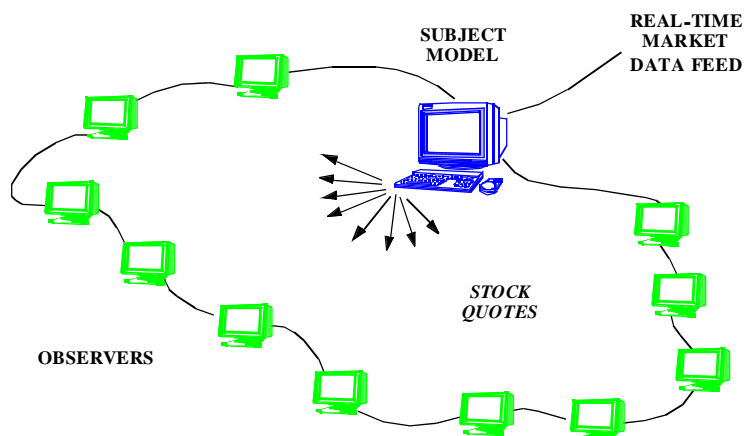
- Any number of views on a Data Table in a windowing system...

- close, open views at will...
- change the data from any view
 - » ... and the other are updated

	1er trim.	2e trim.	3e trim.	4e trim.
Est	20,4	27,4	90	20,4
Ouest	30,6	38,6	34,6	31,6
Nord	45,9	46,9	45	43,9



Yet Another Problem...



What Design Patterns are all about

- As much about *problems* as about *solutions*
 - pairs *problem/solution in a context*
- Not about classes & objects but *collaborations*
- About *non-functional* forces
 - reusability, portability, and extensibility...
- Embody *architectural* know-how of experts

Interest of Documenting Design Patterns

- Communication of architectural knowledge among developers
- Provide a common vocabulary for common design structures
 - Reduce complexity
 - Enhance expressiveness, abstractness
- Distill and disseminate experience
 - Avoid development traps and pitfalls that are usually learned only by experience

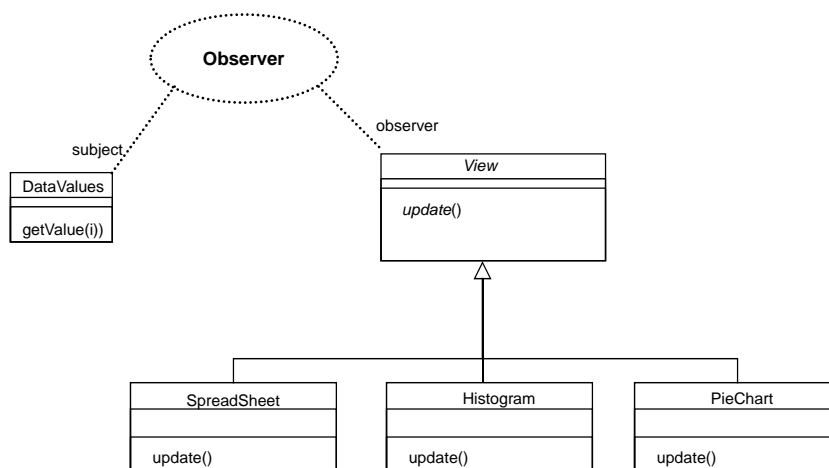
Precise Modeling of Design Pattern Applications

- Go beyond mere documentation
- Specifying reusable applications of design patterns
 - Structural properties
 - Behavioral properties
- Using design patterns in a model
 - Pointing out or detecting pattern occurrences
 - Checking for missing structural properties

© J.-M. Jézéquel, 2003

17

Example in UML



© J.-M. Jézéquel, 2003

18

UML Current Solution (UML 1.4)

- Patterns in UML rely on collaborations
 - That is, sets of collaborating roles
 - A role in a collaboration is a placeholder for objects conforming to the role's base classifier
 - Additional constraining elements can be used
- Collaboration diagrams or sequence diagrams are used to represent expected interactions among participant objects

UML Current Solution (continued)

- Reusability of the pattern is obtained by turning the bases of roles into formal template parameters
- A pattern occurrence is then a template instantiation (a.k.a. binding) providing the actual participants for each template base
- The binding is represented using the intuitive ellipse notation

Using Templates to Express Structural Constraints?

- Using templates as “prototypical” structural constraints was a good idea:
 - Placeholders share the same notation as the “real” modeling elements
 - No need to introduce M2 (Meta-Model) level entities
- But...

Limitations of the approach (1/2)

- Template parameters provide only a fixed number of placeholders for modeling elements
 - Problem in Composite, Visitor, etc.
- Almost everything must be parameterized (including so-called constraining elements)...
- ... which leads to numerous parameters.
- Some parameters are “compound”.

Limitations of the approach (2/2)

- Template expansion is only used to link and check the conformance of the actual modeling elements to this set of “prototypical” structural constraints.
- Moreover, no conformance rules are specified in the UML documentation.

Patterns as Meta-Level Constraints

- Use explicit constraints at the M2 level
 - instead of implicit template constraints
- A pattern is modeled as a set of constraints
 - similar to UML Well-Formedness Rules, but with
 - » Pre-conditions stating the initial situation
 - » Post-conditions to describe the result of the pattern application
 - These supplementary constraints apply only to the participants in the pattern occurrences
- The profile mechanism could be used as a way to build a repository for pattern definitions

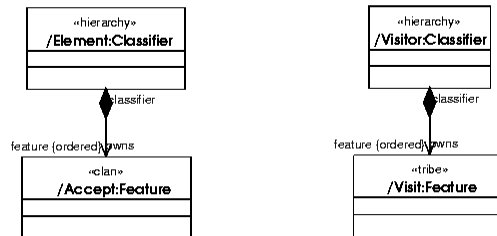
About Collaborations and Constraints

- Collaborations as contexts for OCL expressions
 - Some constraints involve several elements
 - The context of an OCL expression is normally made of a single element - "self"
 - Collaborations and their roles help describe complex contexts

Collaborations of modeling elements

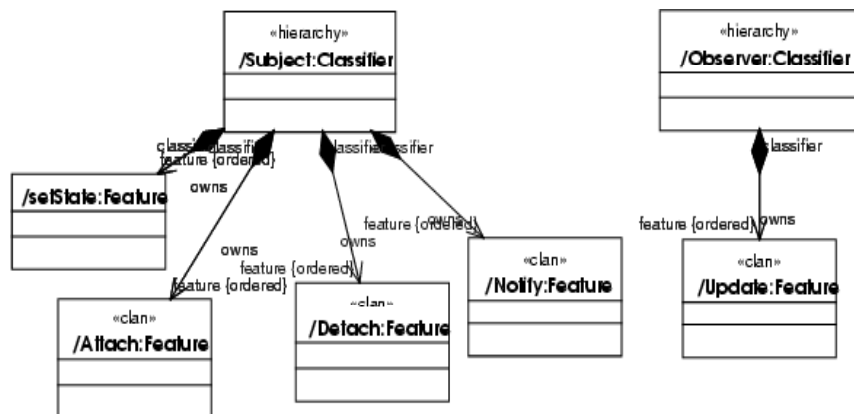
- A pattern can be thought of as a constrained collaboration of UML modeling elements.
- Refinements can be specified by specializing the collaboration and adding new constraints
- Each occurrence of the pattern in the model corresponds to a M2 collaboration occurrence

Model of the Visitor Pattern



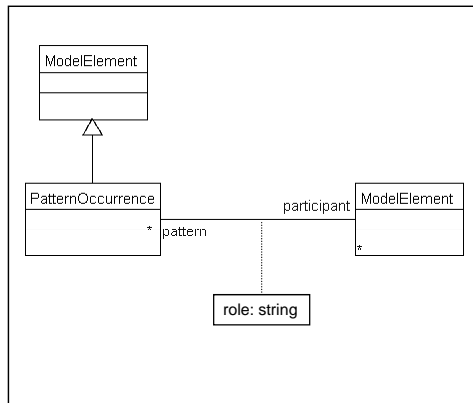
- visit->size() = element->size()
 and visit->forall(v | visitor->feature->includes(v)
 and v.parameter->size() = 1
 and element->exists(e | v.parameter.type = e)
 and ...
)

Model of the Observer Pattern

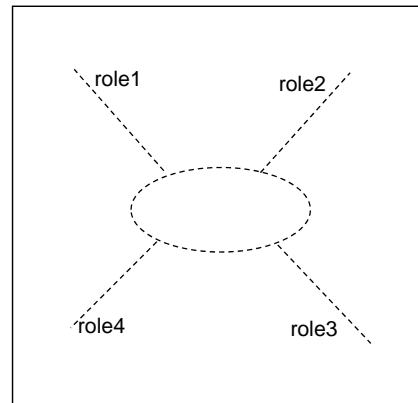


Meta-Model for Pattern Occurrences

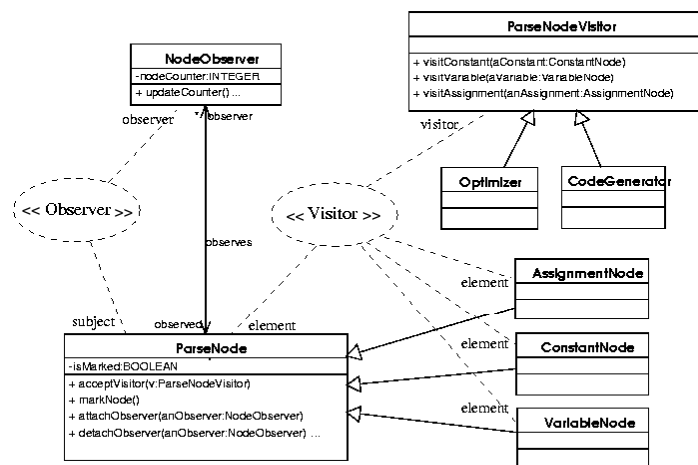
Meta-Model



Notation



Pattern Occurrences



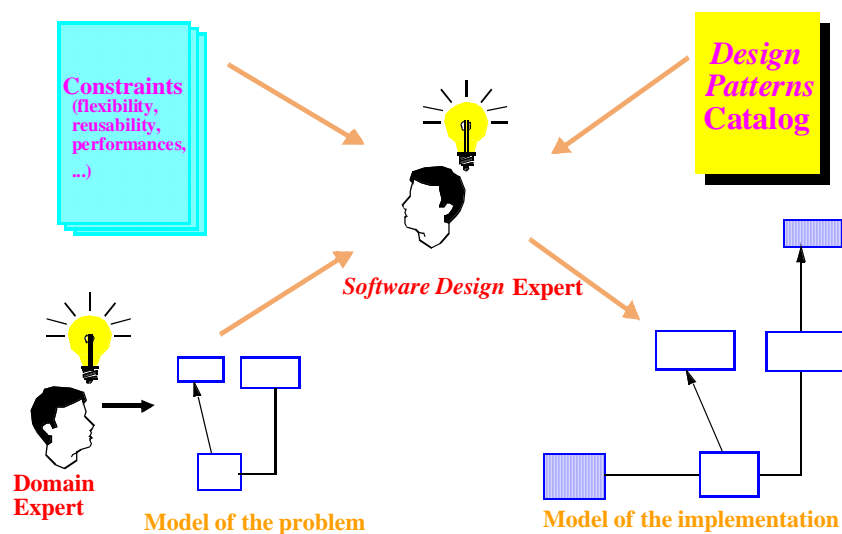
About Behavioral Properties

- Interactions (sequence diagrams) are representations of expected behavior
 - They are to be interpreted as properties
- Precise specification requires a model of the execution semantics
 - HMSCs, Action Semantics
- Behavioral properties should be constraints restraining the set of possible executions

© J.-M. Jézéquel, 2003

31

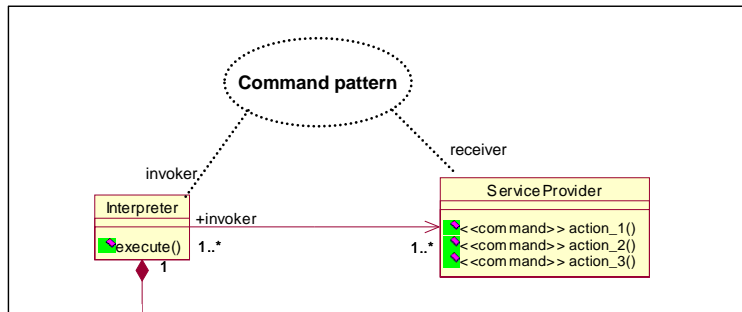
Design Patterns in the Model Driven Architecture



© J.-M. Jézéquel, 2003

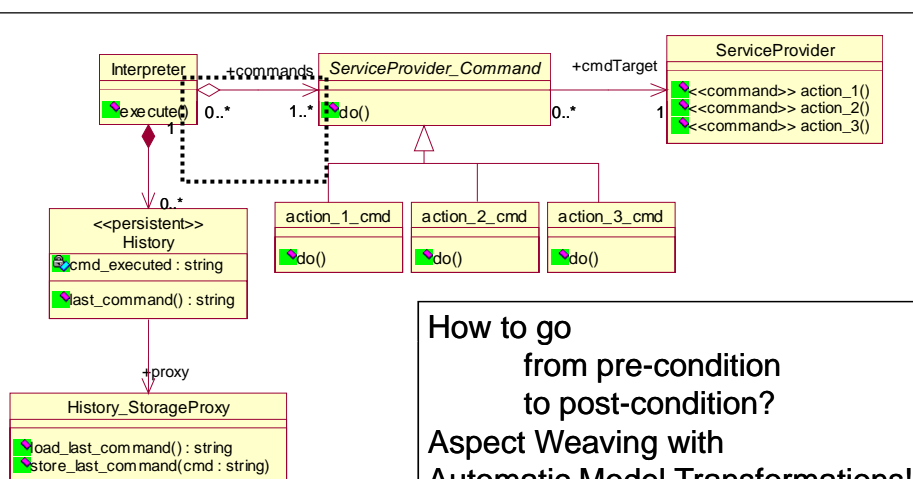
32

Let's look at a business model... (pre-state)



- Instantiate command design pattern.
- Implement persistence that stores the command string to a given database.

...and the wanted design model... (post-state)



How to go
from pre-condition
to post-condition?
Aspect Weaving with
Automatic Model Transformations!

Design Patterns: Summary



- Design Pattern applications as constrained collaborations
- Many different variants of applications
 - E.g. observer push or pull
- Identification of occurrence
 - UML ellipse notation
- Weave the pattern application through model transformations
 - Correspondance with PIM vs PSM

References

- “Design Patterns and Contracts”
 - Addison-Wesley, 1999. ISBN 0-201-30959-9
- “Making Components Contract Aware”, *IEEE Computer*, July 1999
 - A. Beugnard, J.-M. Jézéquel, N. Plouzeau, D. Watkins
- “Precise modeling of design patterns”, - In Proc. UML2000, LNCS 1939
 - A. Le Guennec, G. Sunyé, and J.-M. Jézéquel
- “A toolkit for weaving aspect oriented UML designs.” – In Proc. of AOSD 2002,
 - W.M. Ho, J.-M. Jézéquel, F. Pennaneac'h, and N. Plouzeau.

