A Visual Language for Design Pattern Modelling and Instantiation

THE UNIVERSITY OF AUCKLAND
www.auckland.ac.nz

David Maplesden, John Hosking and John Grundy

Department of Computer Science,
University of Auckland, New Zealand
john@cs.auckland.ac.nz

TOOLS Pacific 2002

Outline

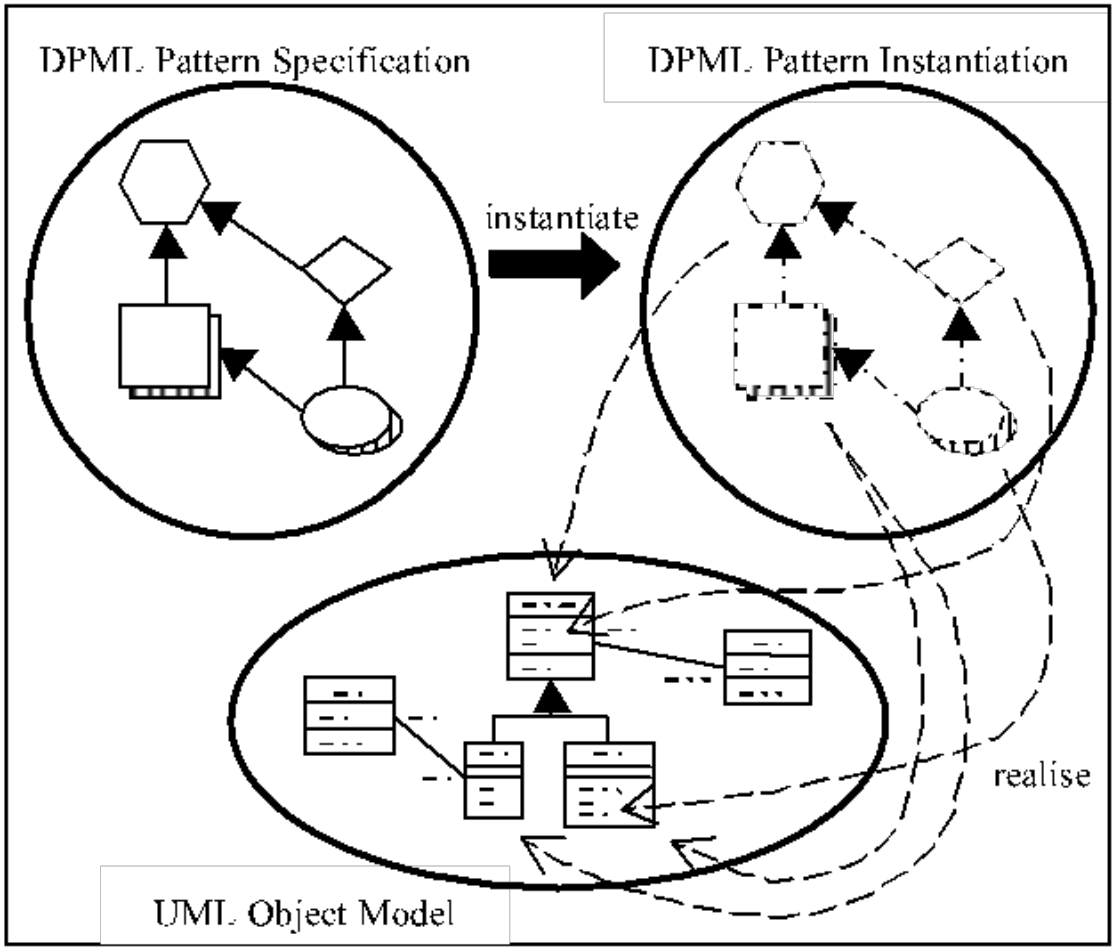
- Motivation
- Design Patterns & Design Pattern Solutions
- Design Pattern Modeling Language
- Instantiating DPML patterns
- DPML+UML
- DPTool
- Evaluation of DPML & DPTool
- Future work

Support for Design Patterns

- **Use patterns to help reuse design/implementation approaches**
- **Use with UML (or other) OODs + code**
- **Want to better-support:**
  - Modeling of design pattern "solutions" i.e. particular approaches to implementing patterns
  - Tracking usage of pattern solutions in designs
  - Validating patterns are correctly used
  - Abstracting new patterns from design models
- **Our approach:**
  - Design Pattern Modeling Language (DPML)
  - DPTool

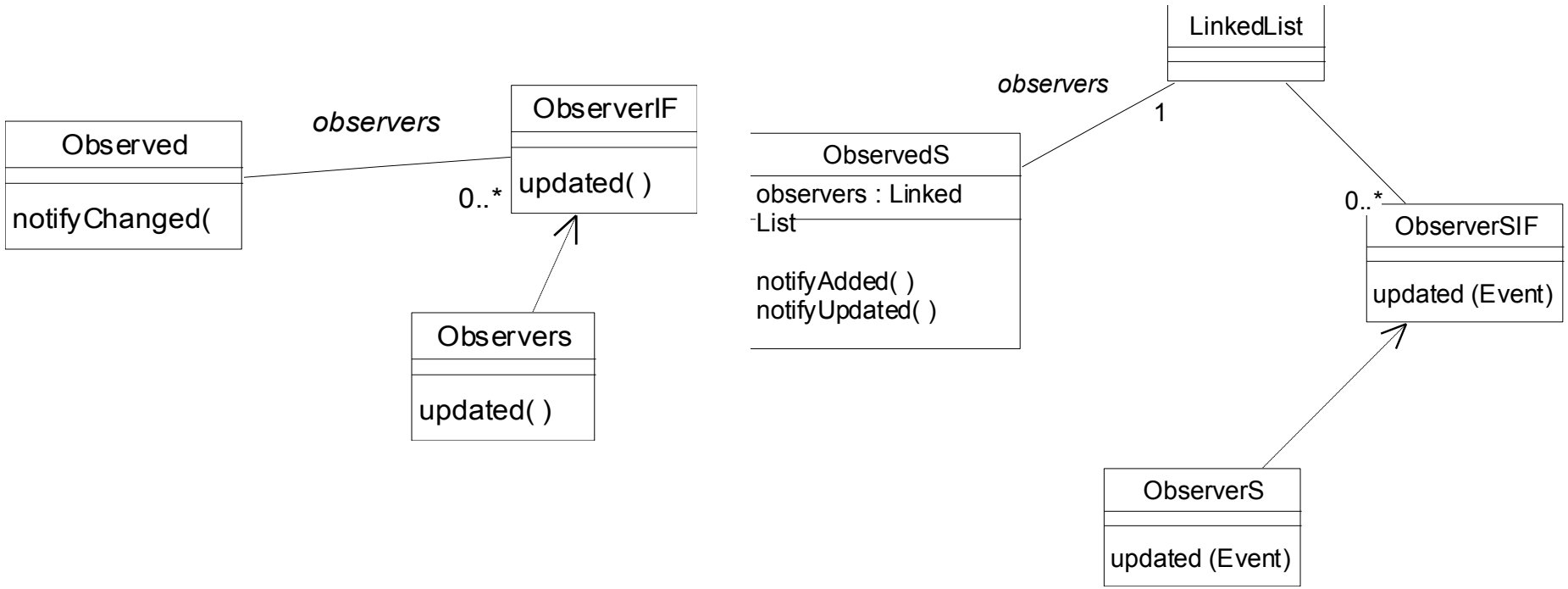
Usage in Design Process

- Modeling with UML
- Design pattern specifications (using DPML)
- Instantiate DPs from DPML
- Link instantiated DP model elements to UML design elements
- (Abstract DP instantiations & DPML DP models from UML…)



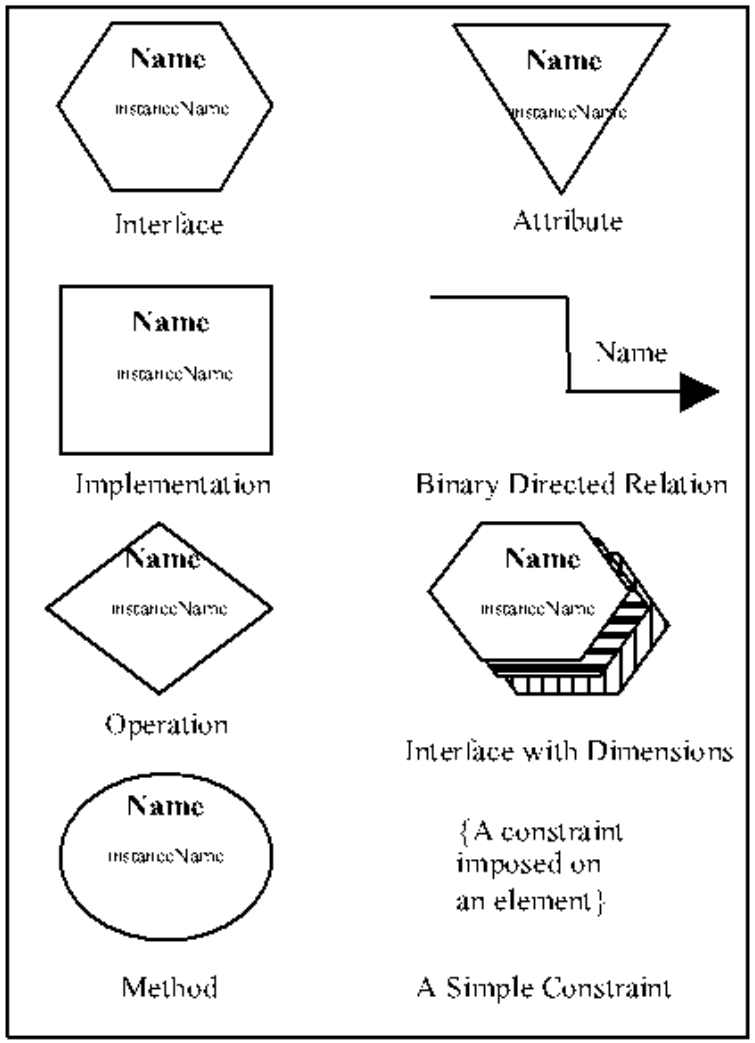
Design Patterns vs Design Pattern Solutions

- Design pattern models abstract problem solution
- Design pattern solution specifies actual approach to solving problem (classes, methods, relationships etc)
- May have >1 solution for a particular design pattern…



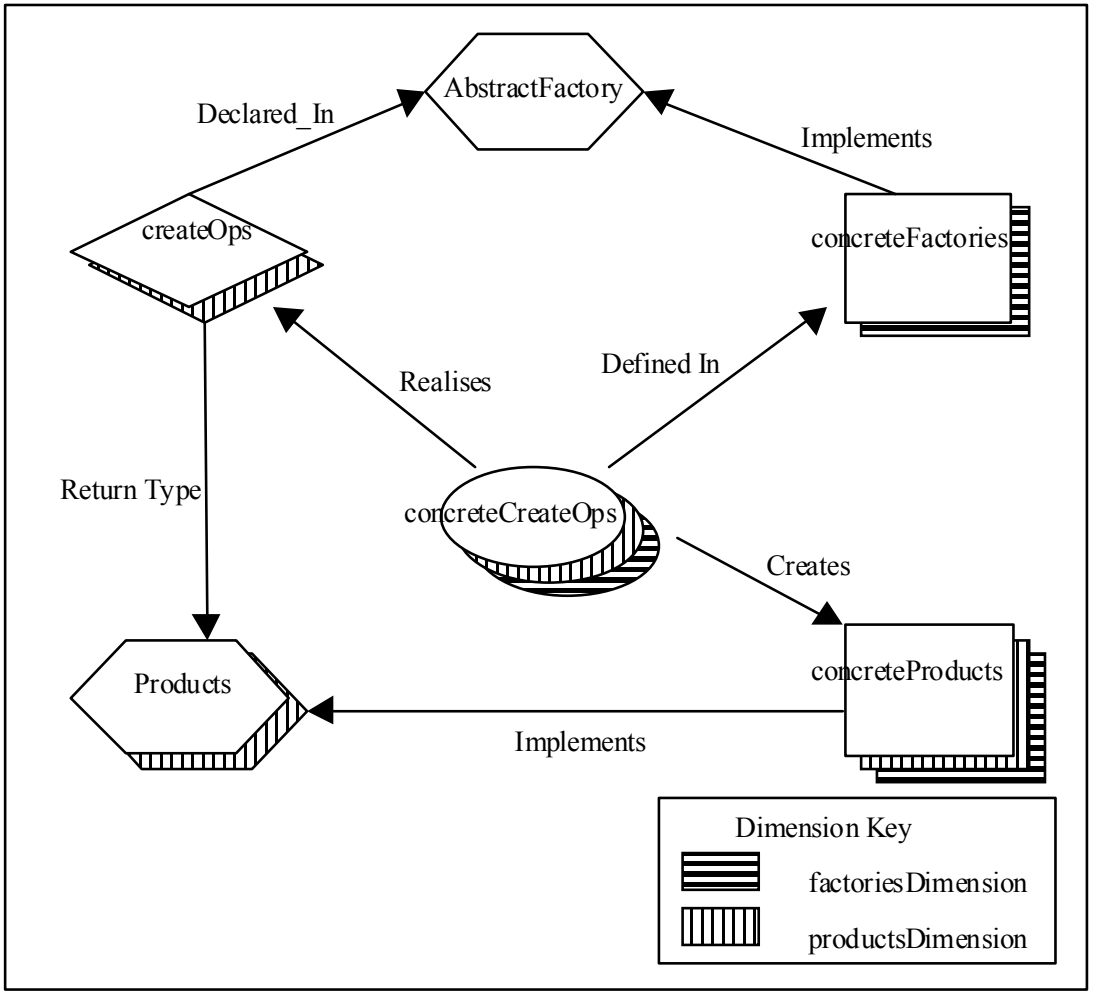
DPML

- **DPML - Design Pattern Modelling Language**
- **Abstract representation of design pattern solutions**
- **Supports instantiation of patterns into UML designs**
- **Basic notation represents important participants**
  - interfaces & implementations
  - operations and methods
  - attributes
  - relations & constraints
  - abstract cardinality (dimensions)

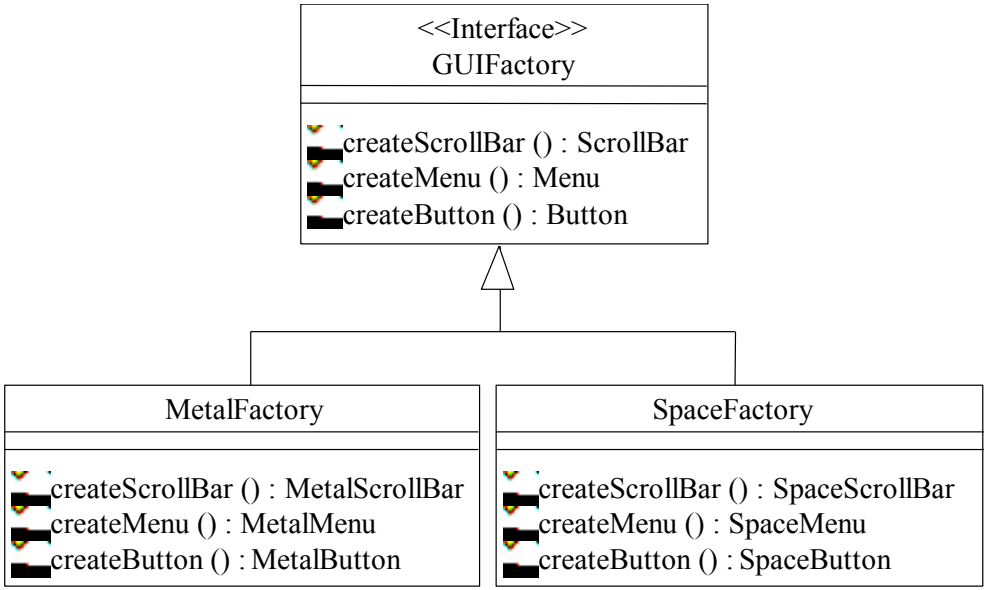


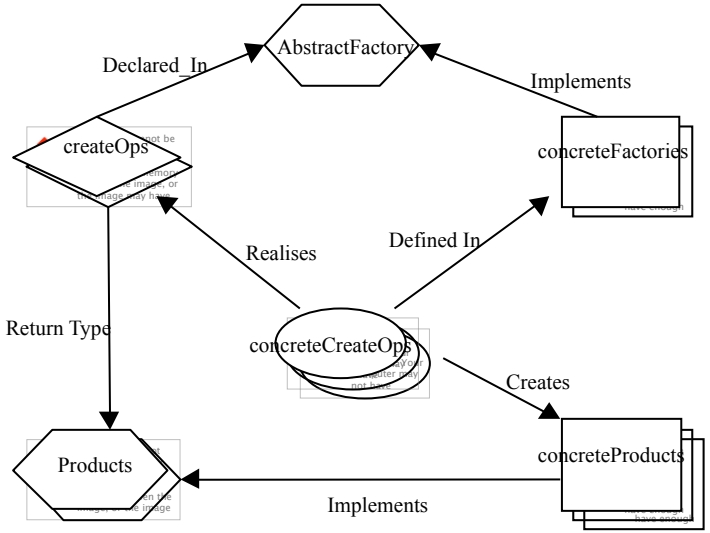
Example: Abstract Factory Pattern

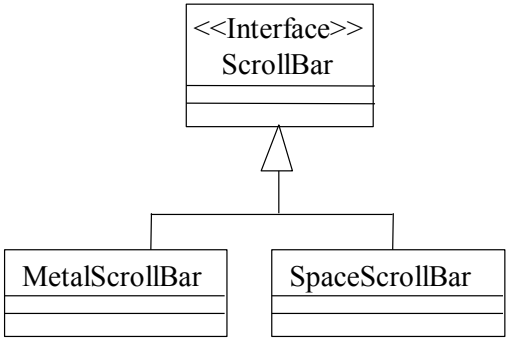
- Each dimension represents cardinality of the set of participants
- Eg same number of createOps as Products (one for each Product)
- Eg no of concrete CreateOps is no of factories times no of products

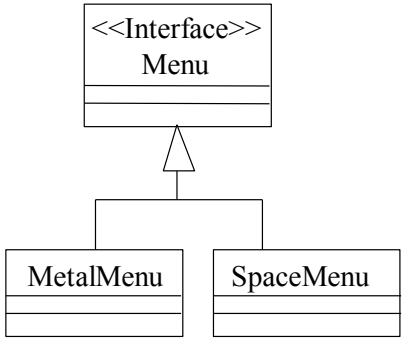


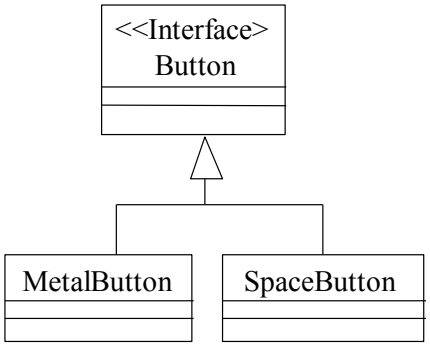
UML Model


<<Interface>>
GUIFactory
createScrollBar () : ScrollBar
createMenu () : Menu
createButton () : Button
MetalFactory
createScrollBar () : MetalScrollBar
createMenu () : MetalMenu
createButton () : MetalButton
SpaceFactory
createScrollBar () : SpaceScrollBar
createMenu () : SpaceMenu
createButton () : SpaceButton


AbstractFactory
Declared_In
Implements
createOps
concreteFactories
Realises
Defined In
Return Type
concreteCreateOps
Creates
Products
concreteProducts
Implements

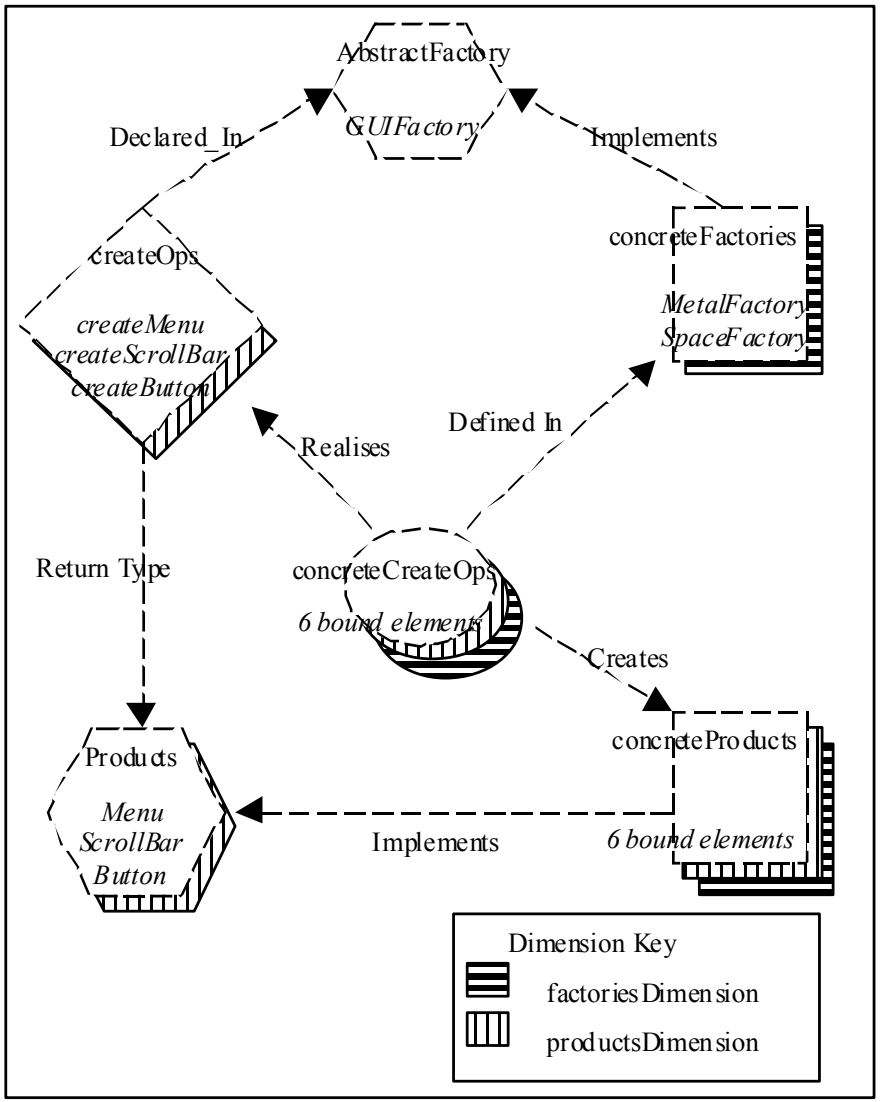

<<Interface>>
ScrollBar
MetalScrollBar
SpaceScrollBar


<<Interface>>
Menu
MetalMenu
SpaceMenu


<<Interface>
Button
MetalButton
SpaceButton

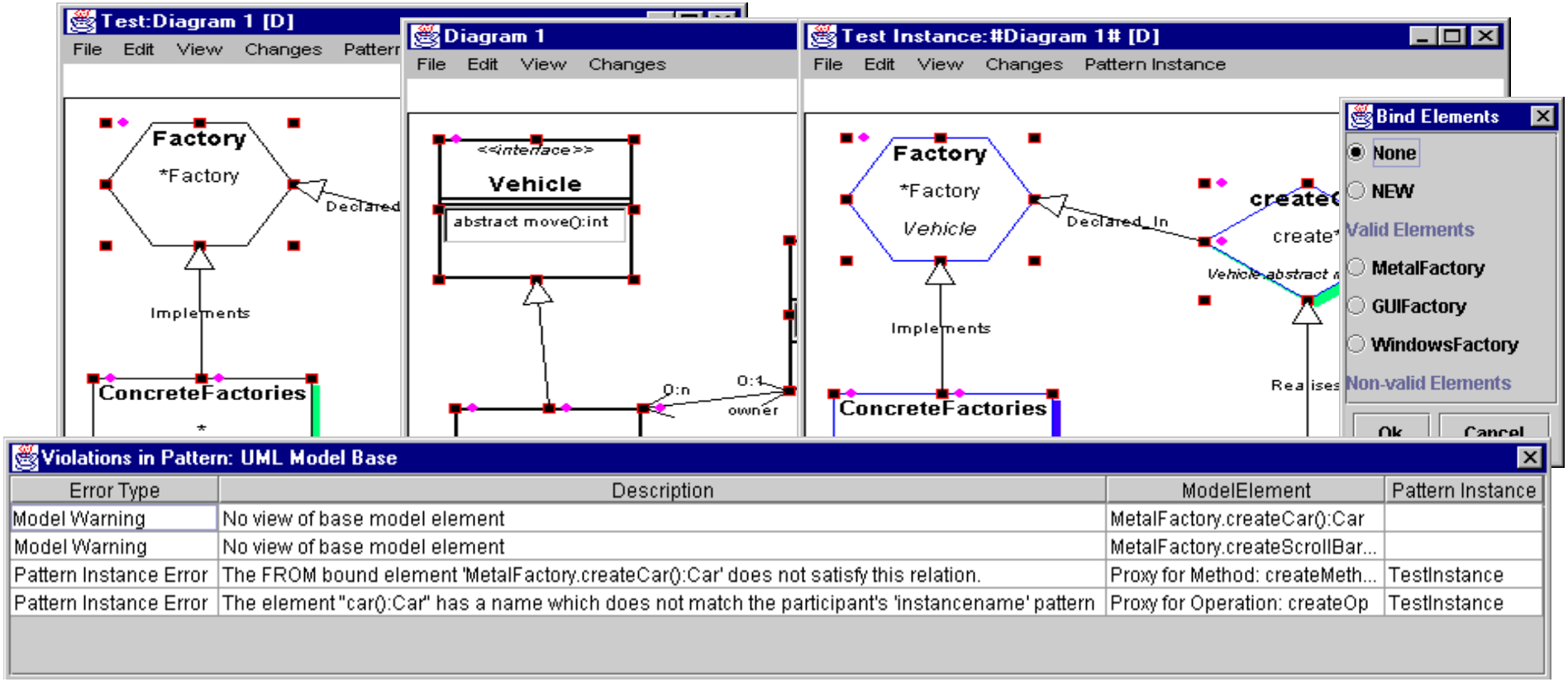
Instantiation into UML Designs

- Have instantiation diagrams that refer to UML classes, opns, etc
- Instantiation diagram elements from DPML DP models linked to UML design elements
- Allows tracking of usage, validation of usage
- Possible to abstract DPs from UML models…
- Eg instantiation of abstract factory into GUI factory



DPTool Examples

- DPML models
- UML Models
- DP Instantiation diagrams
- Validation



Evaluation

- **Two approaches:**
  - Empirical (several experienced designers)
  - Cognitive ("cognitive dimensions")
- **Empirical:**
  - Half a dozen experienced industry and academic designers
  - Use DPTool to model, instantiate several patterns and (simple) UML designs
  - Very good feedback on usefulness of DPML + tool support
- **Cognitive:**
  - Assessed DMPL visual language on several dimensions
  - Generally rates well, though quite "abstract"
  - Some problems with hiding links between elements in different models

Future Work & Conclusions

- Alternative visual representations of DPML elements
- Link visualisation; further work with dimensions
- Abstraction of DP solutions from UML models
- Overlapping patterns – analysis & visualisation support
- Applying to software architecture patterns/styles…
- Plug-in to commercial CASE product

- Can model design patterns using their own language (DPML)
- Can associate UML elements with DPML instances to track pattern usage, validate usage
- Surveyed experienced designers like this kind of support
- Questions over appropriate visualisation, other applications

References

- Maplesden, D., Hosking, J.G. and Grundy, J.C. A Visual Language for Design Pattern Modelling and Instantiation, In Proceedings of Human-Centric Computing 2001, IEEE CS Press.
- Grundy, J.C., Mugridge, W.B. and Hosking, J.G. Constructing component-based software engineering environments: issues and experiences, Information and Software Technology Vol 42, No. 2, Special Issue on Constructing Software Engineering Tools, Elsevier Science Publishers.
- Grundy, J.C., Hosking, J.G., Mugridge, W.B., Apperley, M.D. A decentralised architecture for software process modelling and enactment, IEEE Internet Computing: Special Issue on Software Engineering via the Internet, Vol. 2, No. 5, September/October 1998, IEEE CS Press, pp. 53-62.
- Grundy, J.C., Hosking, J.G., Mugridge, W.B., Apperley, M.D. A decentralised architecture for software process modelling and enactment, IEEE Internet Computing: Special Issue on Software Engineering via the Internet, Vol. 2, No. 5, September/October 1998, IEEE CS Press, pp. 53-62.
- Grundy, J.C. and Hosking, J.G. Serendipity: integrated environment support for process modelling, enactment and work coordination, Automated Software Engineering: Special Issue on Process Technology, Vol. 5, No. 1, January 1998, Kluwer Academic Publishers, pp. 27-60.
- Grundy, J.C., Mugridge, W.B. and Hosking, J.G. A Java-based Componentware Toolkit for Constructing Multi-view Editing Systems, In Proceedings of the 2nd Component Users' Conference (CUC'97), Munich July 15-18, 1997, SIGS Books.