Case Studies – DSVL Tools

- **Design Patterns – MaramaDPTool**

  Maplesden, D., Hosking, J.G. and Grundy, J.C. A Visual Language for Design Pattern Modelling and Instantiation, Chapter 2 in *Design Patterns Formalization Techniques*, Toufik Taibi (Ed), Idea Group Inc., USA, 2007.

- **Performance Engineering – Marama/MTE**

  Draheim, D., Grundy, J.C., Hosking, J.G., Lutteroth, C. and Weber, G. Realistic Load Testing of Web Applications, In Proceedings of the 10th European Conference on Software Maintenance and Re-engineering, Berlin, 22-24 March 2006.

- **Enterprise Modelling Language**

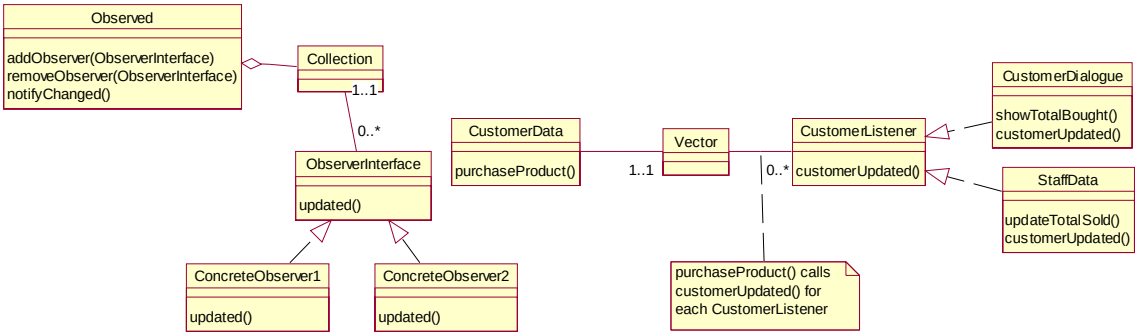
  Li, L. Hosking, J.G. and Grundy, J.C. Visual Modelling of Complex Business Processes with Trees, Overlays and Distortion-Based Displays, In Proceedings of the 2007 IEEE Symposium on Visual Languages and Human-Centric Computing, USA, Sept 23-27 2007, IEEE CS Press.

Design Patterns

- **Common ways of solving programming problems when designing & coding**
- **Many examples in Java APIs:**
  - **Delegation, Observer e.g. event subscription/notification**
  - **Iterators e.g. over collections**
  - **Immutable, Flyweight e.g. String objects**
  - **Singleton e.g. java.lang.Runtime object**
  - **Filters e.g. java.io Reader subclasses**
  - **Adaptors e.g. in AWT for various kinds of events**
  - **Command e.g. Button, MenuItem**
  - **Single Threaded Execution e.g. Vector class method synchronization**

Example: Observer Pattern

- **Name: Observer**
- **Synopsis: way to decouple handling of events; viewing of changing data**
- **Context: like delegation pattern - concept of "observers" on "observed" object [is actually a form of extension to Delegation]**
- **Forces: have object with others needing to be told about changes to it**
- **Solution:**

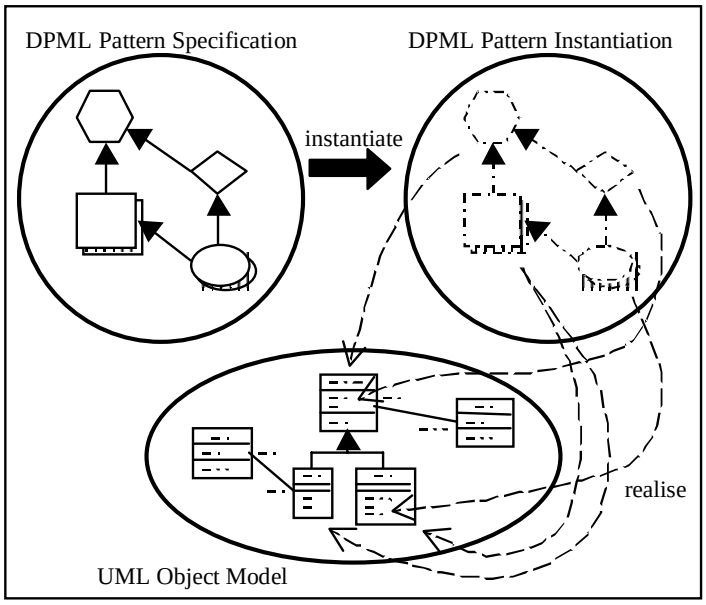


Motivation for DPTool

- **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**

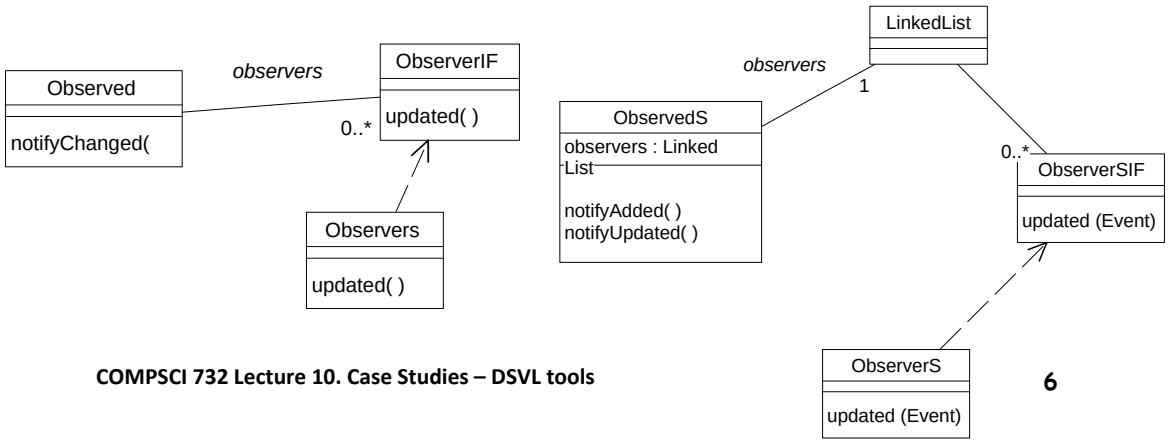
Usage in Design Process

- **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…)**



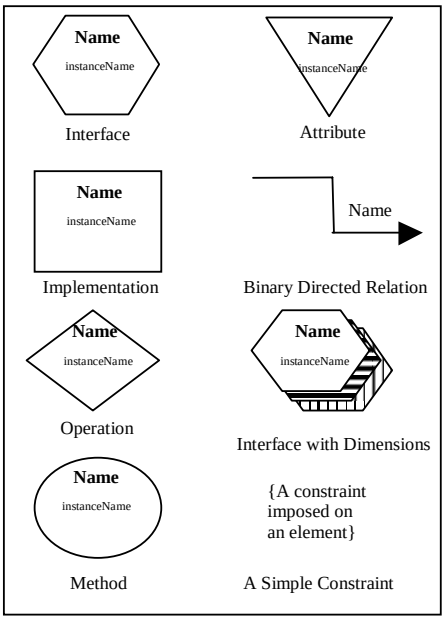
Design Patterns vs Design Pattern Solutions

- **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…**



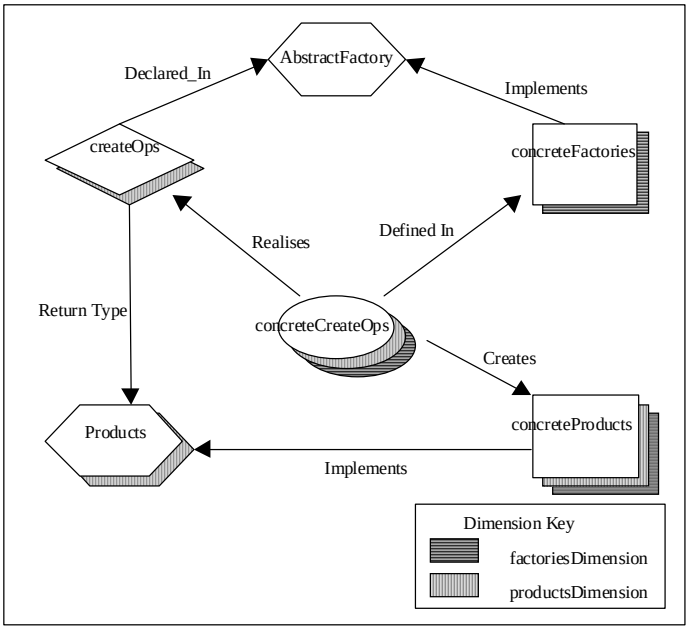
DPML

- **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)**

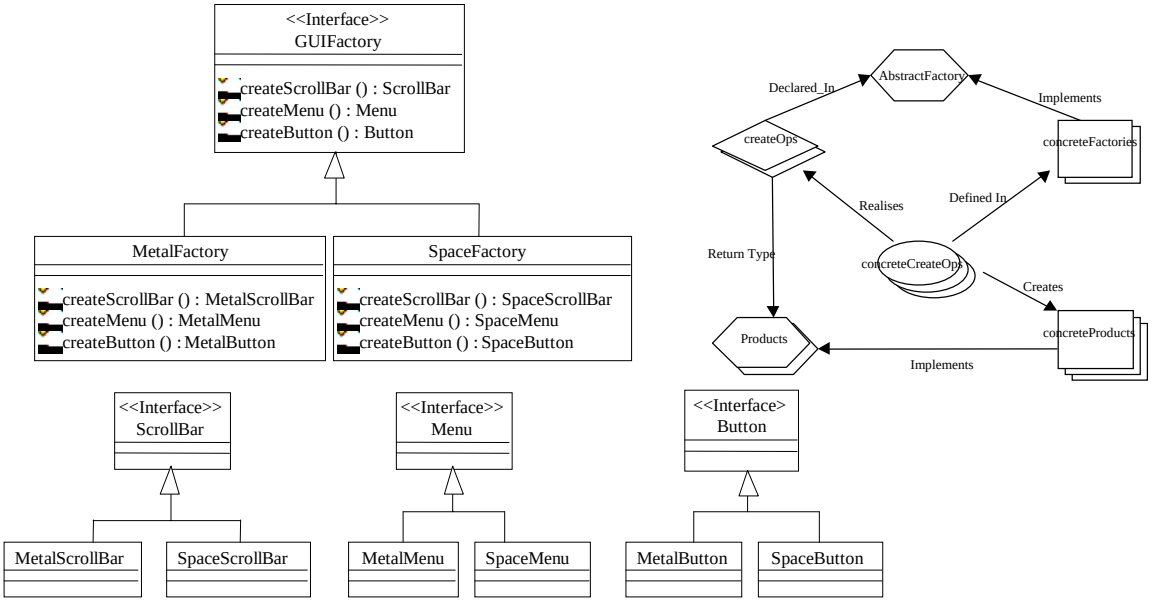


Example: Abstract Factory Pattern

- **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**

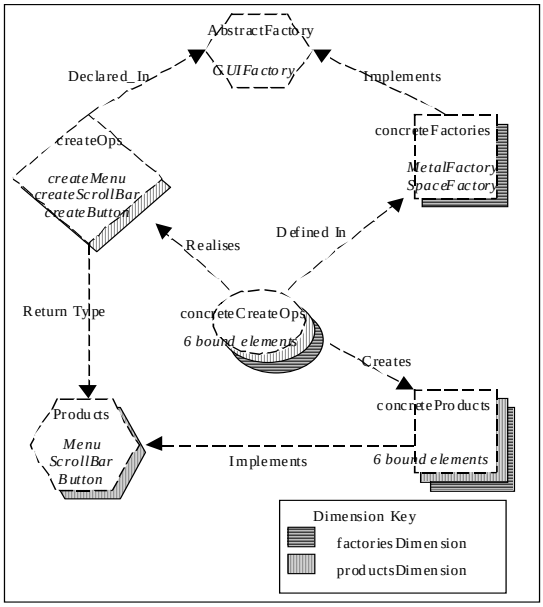


UML Model



Instantiation into UML Designs

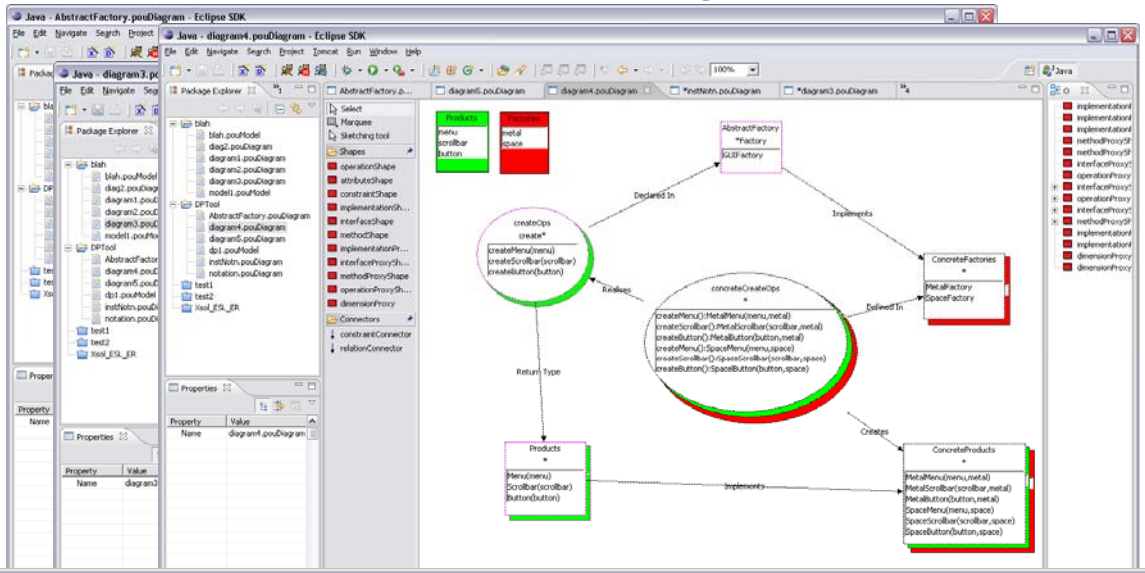
- **Have instantiation diagrams that refer to UML classes, opns, etc**
- **Instantiation diagram elements from DPML DP models linked to UML design elements**
- **Allows tracking of usage, validation of usage**
- **Possible to abstract DPs from UML models…**
- **Eg instantiation of abstract factory into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



DPTool

- Tool support for DPML
- Design pattern solution models
- UML Models
- Instantiation diagrams
- Rule checking, multiple pattern projects etc

DPTool - Examples



Violations in Pattern: UML Model Base

Error Type	Description	ModelElement	Pattern Instance
Model Warning	No view of base model element	MetalFactory.createCar():Car	
Model Warning	No view of base model element	MetalFactory.createScrollBar...	
Pattern Instance Error	The FROM bound element 'MetalFactory.createCar():Car' does not satisfy this relation.	Proxy for Method: createMeth...	TestInstance
Pattern Instance Error	The element "car():Car" has a name which does not match the participant's 'instancename' pattern	Proxy for Operation: createOp	TestInstance

Evaluation

- Two approaches we used:
  - 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 hidden dependencies between elements in different models

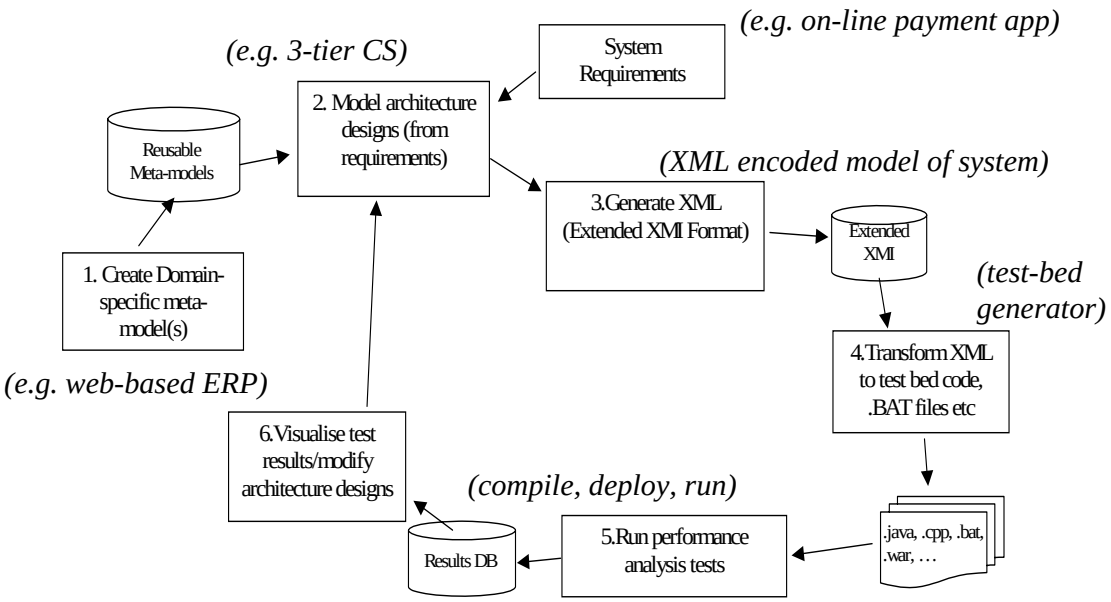
Performance Engineering

- All software systems, but especially distributed systems, have performance requirements e.g.
  - Must support X # users simultaneously
  - Must be able to perform Y # transactions per second
  - Must provide response to user/other system in Z # seconds/ms
- Other performance issues also important e.g. data size, network load, processing load, memory usage, …
- Quote from Auckland IT company CEO: *"The thing that keeps me awake at night is not (1) can we implement this system, not (2) can we get investment capital/sell system, but (3) will it scale to support buyers' needs?"*
- Its VERY hard to answer this question!

Distributed system performance evaln

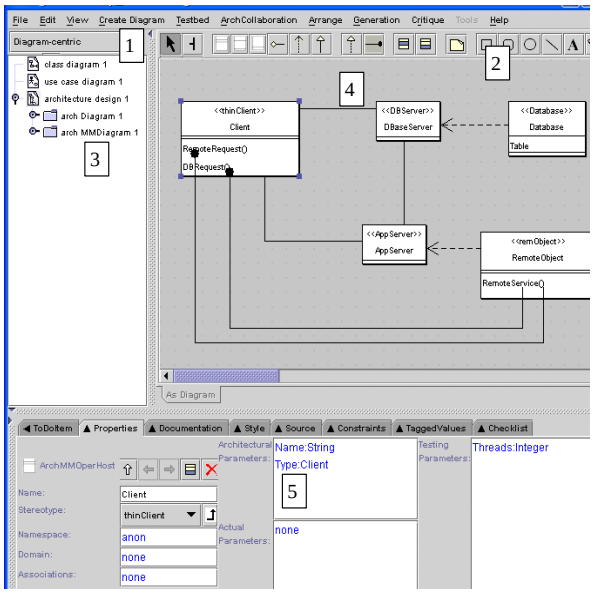
- **Difficulties faced by software engineers:**
  - **Complexity of today's software architectures**
  - **New middleware to use/middleware performance**
  - **Database management/server performance**
  - **Hardware and network performance variation**
  - **Huge variation in deployment situations for our software**
  - **Unpredictability of 3rd party components for system**
- **Ways can we evaluate system performance:**
  - **Existing system profiling**
  - **Benchmarks**
  - **Software architecture-based simulation**
  - **Rapid prototyping**
- **Bottom line: its HARD and time-consuming…**

Automated Rapid Prototyping e.g. Argo/MTE, Marama/MTE

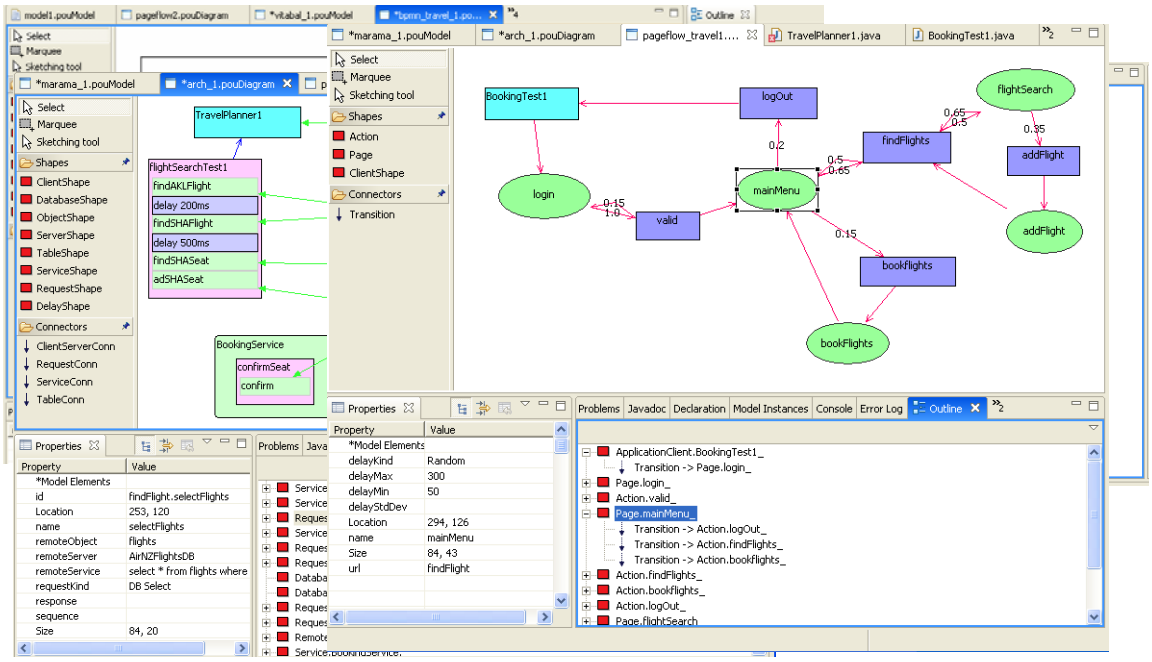


Domain-specific Meta-models

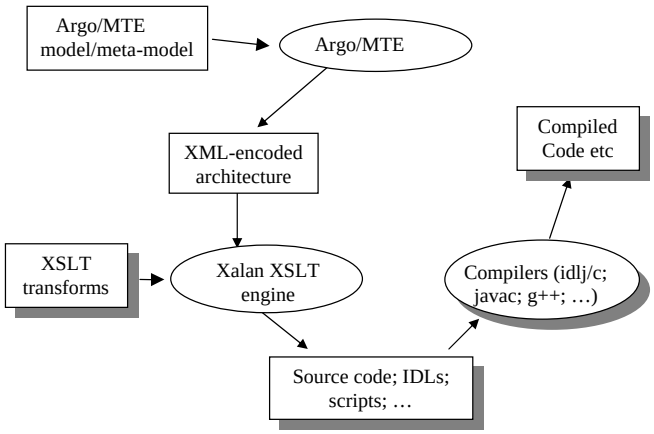
- **Meta-model: client, server, request, service, DB etc**
- **For RMI, CORBA, EJBs, SOAP RPC, JSPs/ASPs, …**
- **Use MM abstractions in visual architectural models = "building blocks" for target system architecture & technologies**



Modelling in MaramaMTE



Code Generation



- **Model architecture using Argo/MTE visual language**
- **XML encoding of architecture generated**
- **XSLT scripts used to generate .java, .bat, .xml etc files (see egs in paper…)**

XSLT Example

Client_Clie

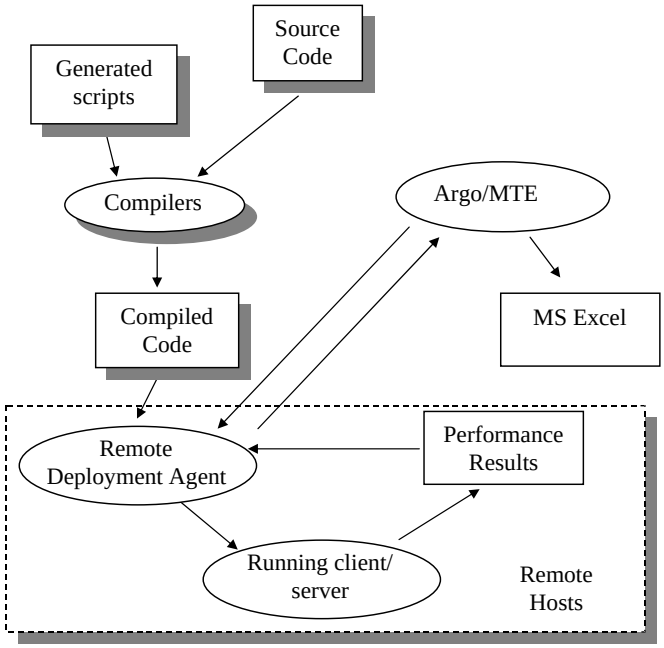
```
<?xml version="1.0" e
<Client>
<Name>ClientTest</Nam
<Hosts>LocalHost</Hos
<Threads>1</Threads>
<Request>
<Type>CorbaRequest</T
<Name>findVideo</Name
<RemoteObject>VideoMa
<TimesToCall>10</Time
<RecordTime>yes</Reco
</Request>
<Request>
<Name>rentVideo</Name
<Type>CorbaRequest</T
<RemoteObject>VideoMa
<TimesToCall>4</Times
<RecordTime>yes</Reco
</Request>
<Request>
…
</Client>
```

```
…
<!-- CORBA reque
<xsl:template ma
    public stati
(<xsl:value-of s
        int iter
  String name =
        String r
        System.g
        long sta
        int i=0;
        while(i
            ser
          i++;
        }
        if(recor
            long
            doub
            Stri
            Syst
            Syst
        }
  }
</xsl:template>

…
```

ClientTest.java

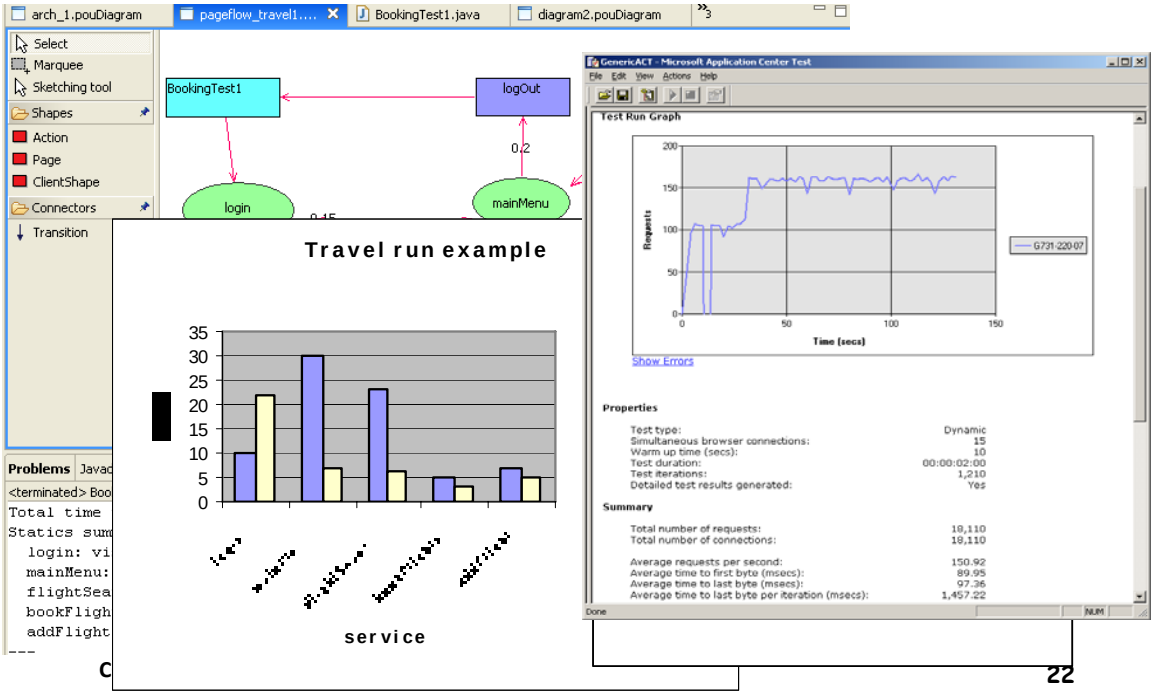
```
public class ClientTest
{
…

   public static void findVideo(VideoManager server){
        int iter = 10;
  String name = "findVideo";
        String recordTime = "yes";
        System.gc();
        long start = System.currentTimeMillis();
        int i=0;
        while(i != iter){
             server.findVideo_service ();
            i++;
        }
        if(recordTime.equals("yes")){
            long time = System.currentTimeMillis() - start;
            double elapse = (double)(time) /
(double)(Math.max(1,iter));
            String perf =
name+"\t"+time+"\t"+iter+"\t"+elapse;
            System.out.println(perf);
            System.err.println(perf);
        }
  }
…
}
```

Running Performance Tests and Viewing Results



- **Upload code to hosts (clients and servers)**
- **Deploy (EJBs)**
- **Start (RMI, CORBA)**
- **Run clients**
- **Results sent back to Argo/MTE**
- **Results displayed**

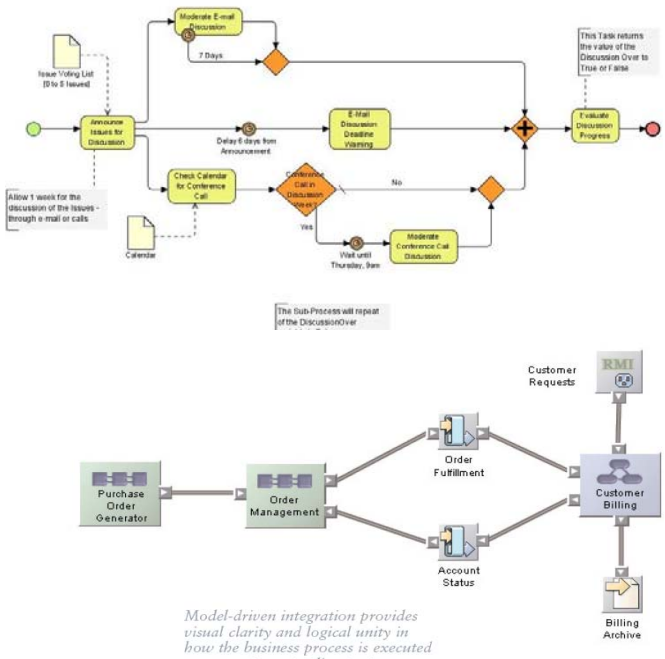
Viewing Performance Results…



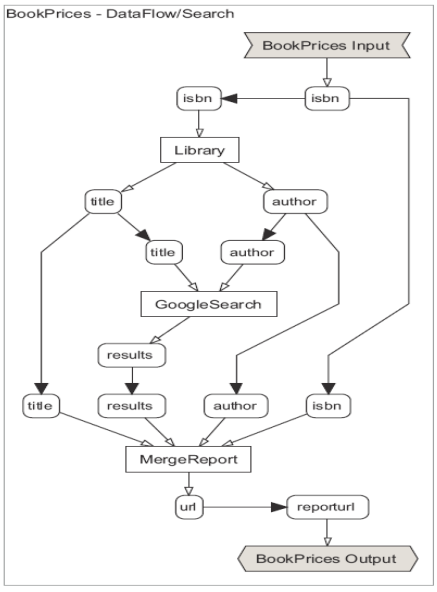
Business process modelling

- **Since the early 1970s many languages, standards, methodologies and tools for business modelling have been created**
- Methodologies**: ER Models, DFDs, Flow Charts, Scenarios, Use Cases, IDEF, etc.**
- Notations**: UML, BPMN, BioOpera, WTD, AOM etc.**
- Tools**: JOpera, T-Web, ZenFlow, ARIS, WebSphere, Visio etc.**

Box-and-line Style Diagrams



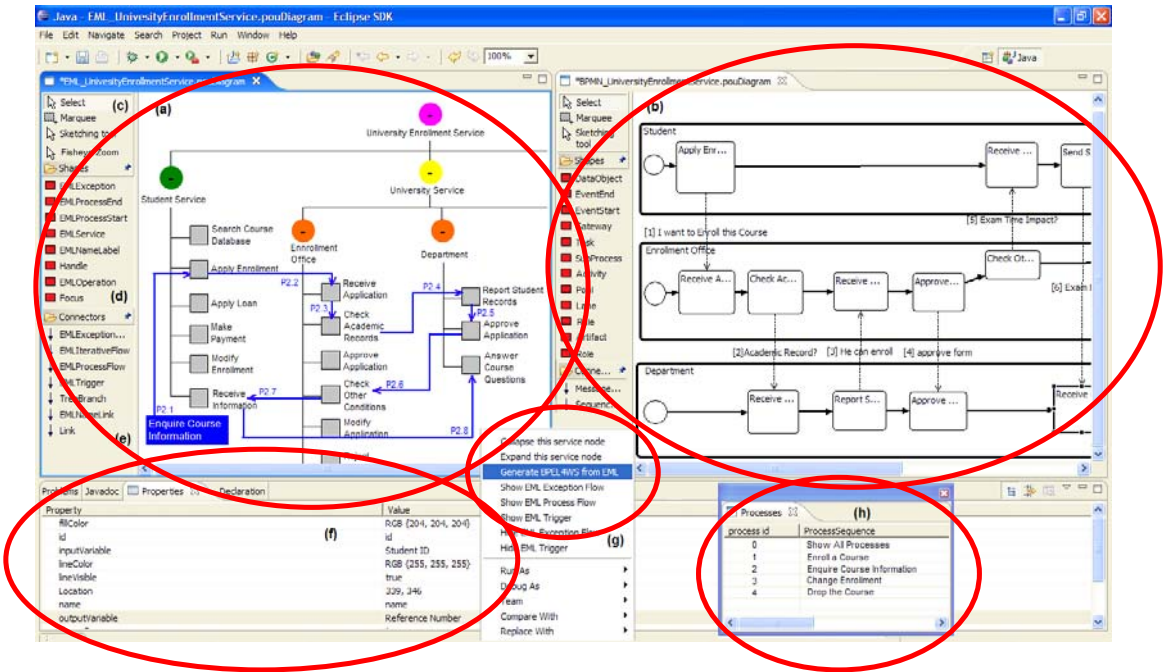
Model-driven integration provides visual clarity and logical unity in how the business process is executed across systems, trading partners, and workflow.



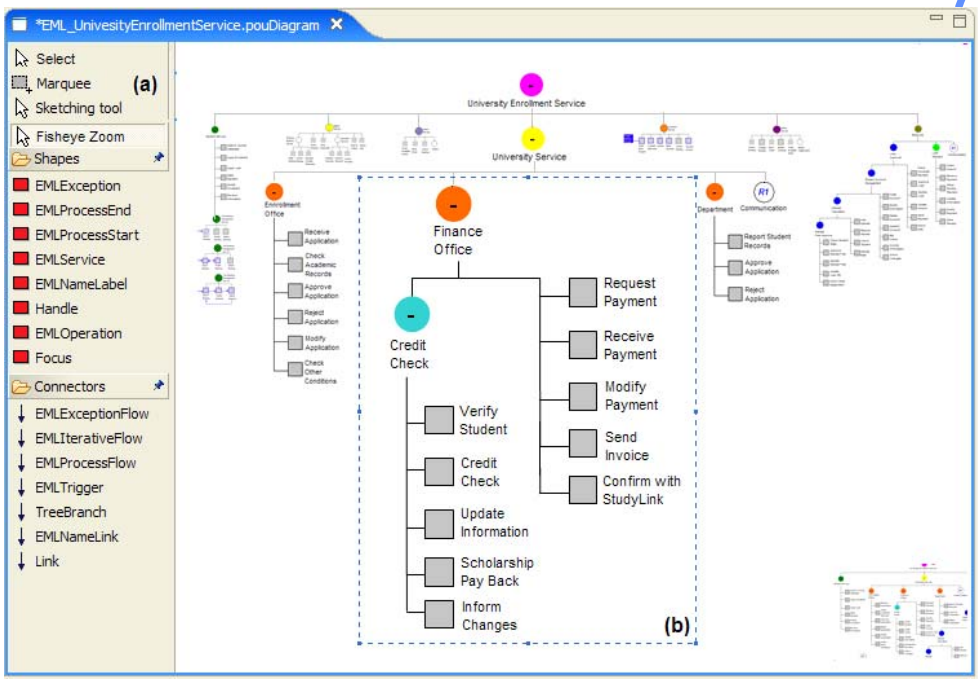
Motivation for MaramaEML

- Most of these approaches only emphasize process modelling, missing the ability to model system functional architecture
- Common source of difficulty: appropriate visual methods to reduce the complexity of large business modelling diagrams
- Existing modelling technologies are:
  - effective in only limited problem domains or
  - have major weaknesses when attempting to scale to large systems modelling
    - e.g. “cobweb” and “labyrinth” problems

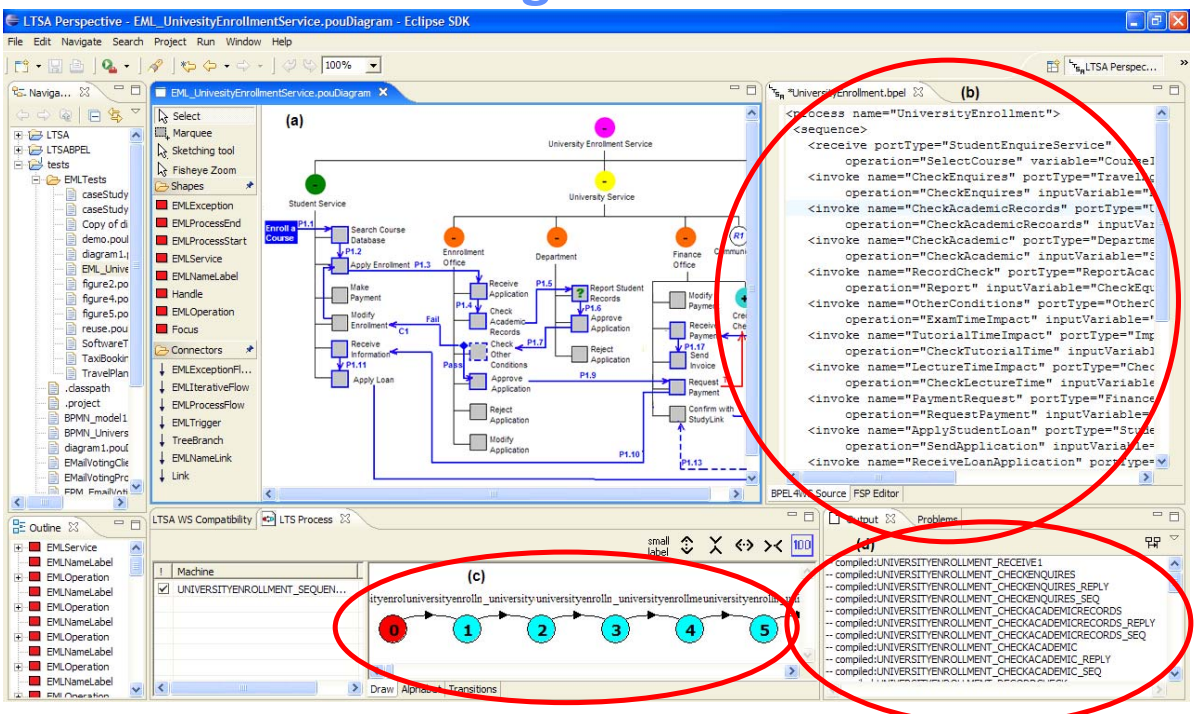
MaramaEML



Distortion-based view for scalability



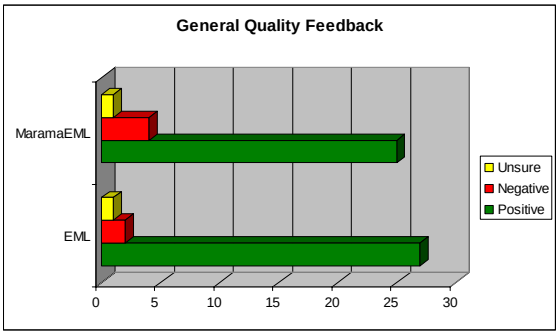
Code generation

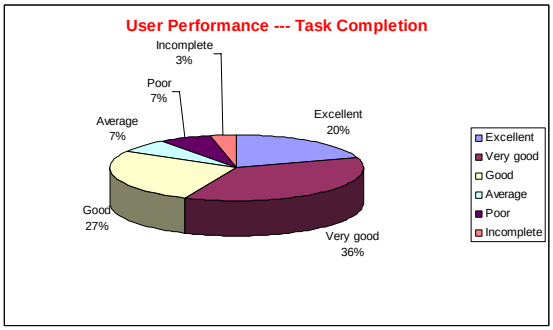


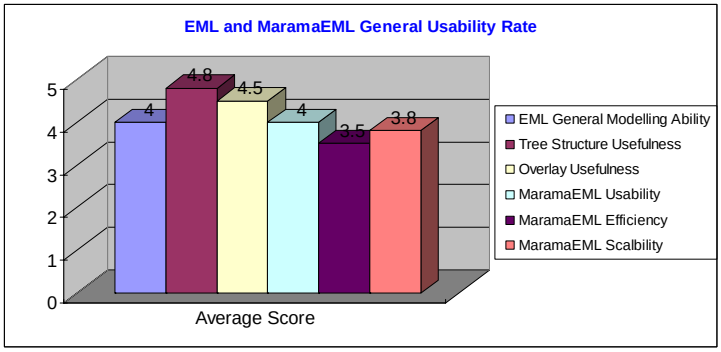
Evaluation

- Versus Requirements
  - All met
- Cognitive dimensions
  - Strong emphasis on:
    - closeness of mapping
    - hidden dependency mitigation
- Task-based end user evaluation
  - Small scale
  - Good support for EML over BPMN for both pen and paper and computer based modelling
  - Some criticism of environment
    - Speed of response for fisheye view
    - Lack of traceability support
- Large end user evaluation
  - Approx 30 users

Large Evaln Results Summary







Large Evaln Result Summary

Participants were divided into two groups to answer different questionnaires (General Usability and Cognitive Dimensions Walkthrough)

- Very positive results for EML modelling ability and tree-overlay methodology
- Good comments on software tool support: easy to use, provides efficient modelling, inspection and code generation functions, etc.
- Very good performance feedback on Visibility enhancement, Viscosity maintenance, Diffuseness simplification, Hard Mental Operation reduction, Consistency awareness, Hidden dependency mitigation and Closeness of mapping.

- Trade offs for Premature Commitment, Abstraction Gradient and Secondary Notation support
- Strong demand for adding UML view into framework
- An achromatopsia participant became totally lost in the overlay integration view
- Lack of F1 help function in system
- Speed improvements needed when modelling large tree structure

Summary

- **Visual languages offer new metaphors for constructing models of complex software systems to support requirements, design, coding, testing, reverse engineering, interaction with stakeholders, even non-software tools e.g. Statistics Design Language (SDLTool – see A Suite of Visual Languages for Statistical Survey Specification, Kim et al)**
- **Domain-specific visual language solutions currently in vogue – usually as a part of Model-Driven Engineering – highly stereotyped UML replaced slowly by more appropriate visual languages & tools**
- **Eclipse Graphical Modelling Framework (GMF), MS Visual Studio DSL Tools, AndroMDA, other MDE toolsets evolving rapidly**
- **Evaluation of the DSVLs used still immature**