

Modelling with Templates in UML

Matthew Hause
ARTiSAN Software Tools



Réf: xx 30/01/2005
p1



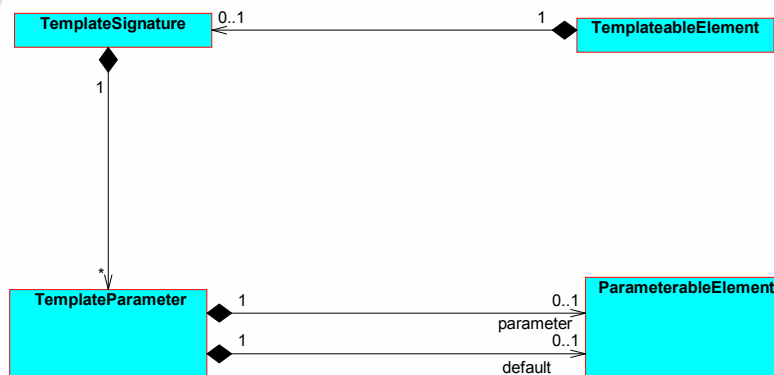
Agenda

- History
- Metamodel and notation
- Templates in Pattern Implementation
- Templates in Aspect Design
- The role of templates in Model-Driven Architecture (MDA)
- Implementation experience

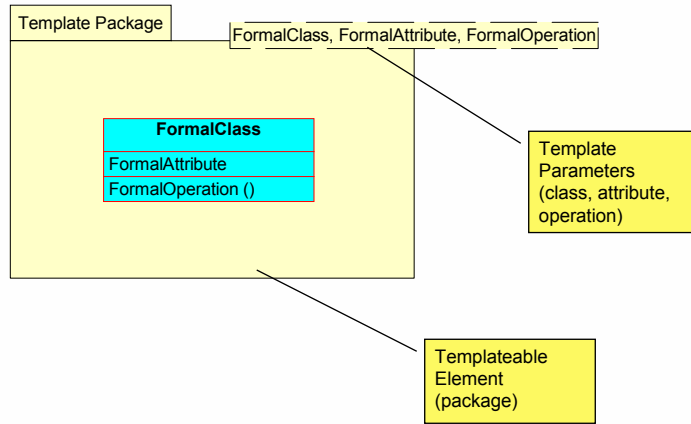
History

- Underused Template metamodel in UML 1.X
- 2U Infrastructure Submission
 - Defined package templates, based on Catalysis concepts
 - Used templates to construct metamodel
 - Text substitutions in names using <>
 - Implemented by prototype tool
- U2P Superstructure Submission
 - 2U Metamodel Adapted by Stuart Kent et al
 - Added class and operation templates
 - Model elements as parameters
 - More general substitution mechanism
- MOF Queries, Views and Transformations (QVT)
 - Many submissions use template-like approach.

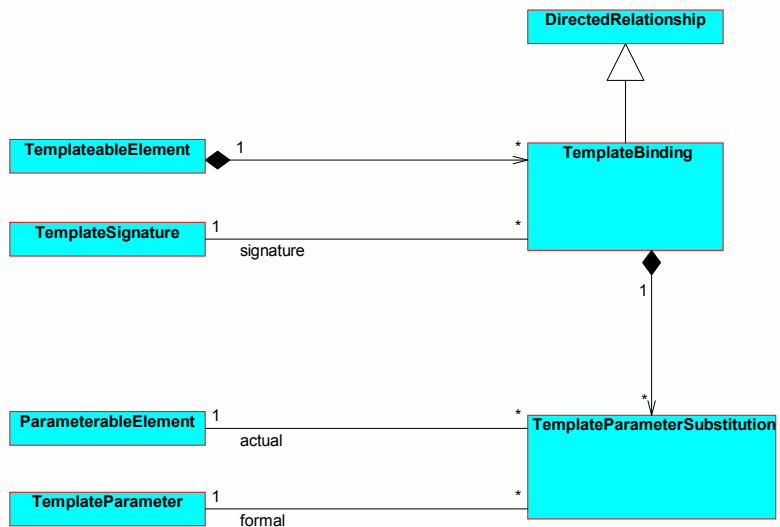
Template MetaModel



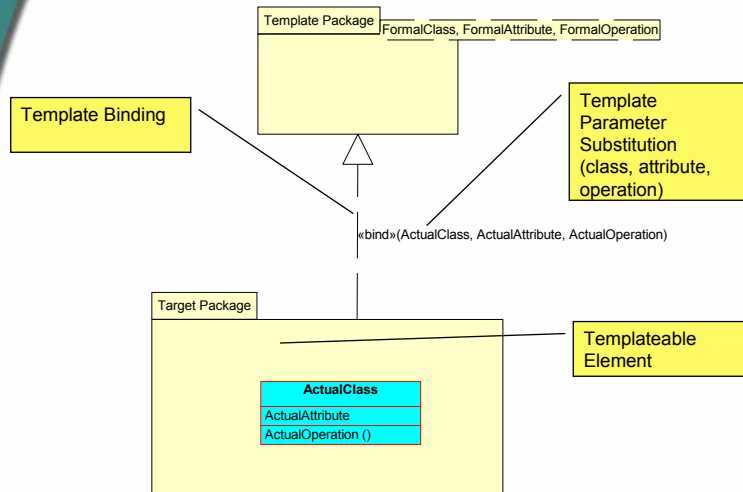
Template Notation



Binding MetaModel



Binding Notation



Pattern Language & Design Patterns

The Problem:

- “We keep reinventing the wheel”
 - *Some organizations solve the same problem differently over and over again!*
 - *No one is learning from mistakes!*
 - *We are wasting valuable time and money!*

The Solution:

- The same recurring problem has a single solution!
 - *Let’s document the solution to the problem in such a way that it can be reapplied – reused over and over again!*
 - *If the initial solution is not optimal we can modify it separately from our development effort and reapply it – reuse!*
 - *Let’s call the documented solution a Design Pattern!*

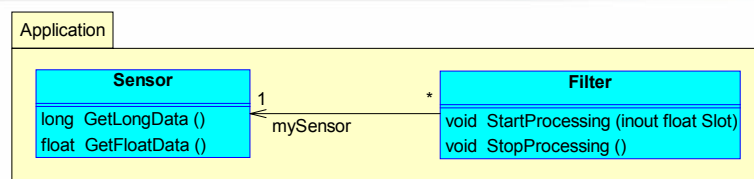
Design Patterns are a flexible and inherently reusable approach to Standardization.

Easy to adopt and easy to adapt!

How is a Pattern Implementation Expressed and Used in UML?

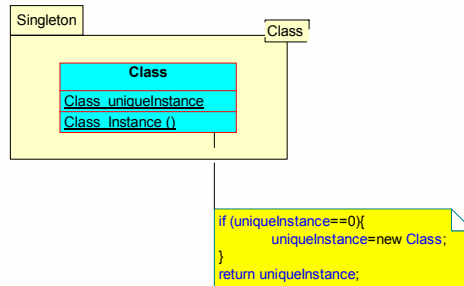
- Expressed as a UML (template) package with formal parameters
 - *Template package contents refer to formal parameters*
- Used by “binding” a client package to a template package
 - *Bind provides the actual parameters*
 - *Template contents are merged into the client*
 - *Formal parameters are replaced by actual parameters*

Example Application



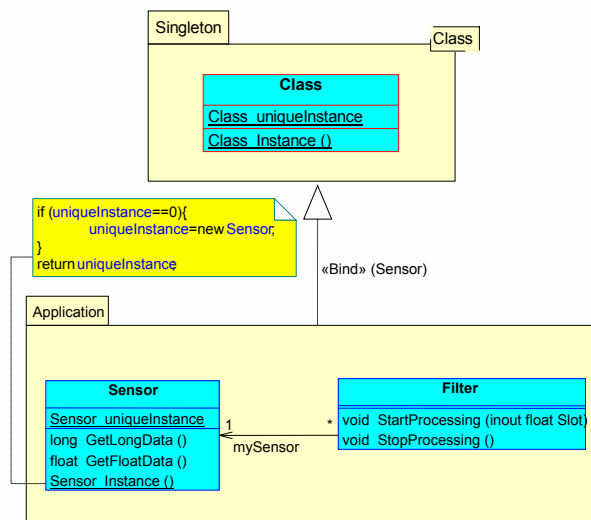
- **Sensor makes available data in two formats, long and float;**
- **Filter has a reference to a Sensor and can be asked to process filter data for some fixed time-slot**

Singleton Pattern

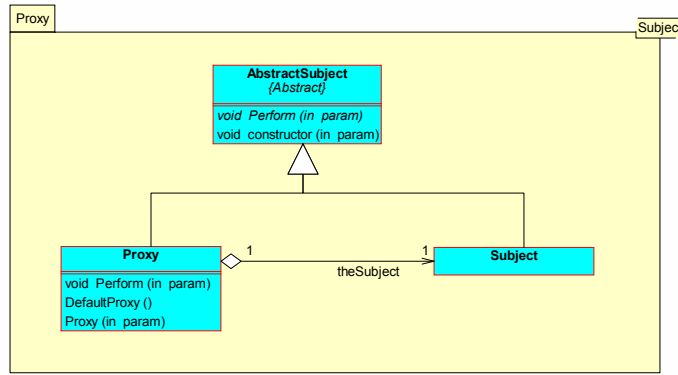


In our system, we want there to be only one Sensor object, which should be globally accessible. This problem is suitable for solution using the Singleton pattern.

Applying the Pattern

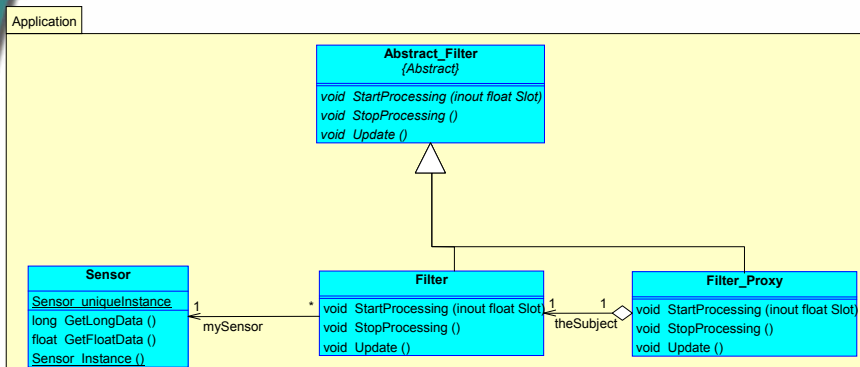


Proxy



Often an application may not want external applications to access an object directly – we may need to perform some error checking, or perhaps the services are accessed remotely, or maybe some methods of the class are dealt with differently from others. In these situations, the ‘Proxy’ pattern may be used.

Proxy Applied

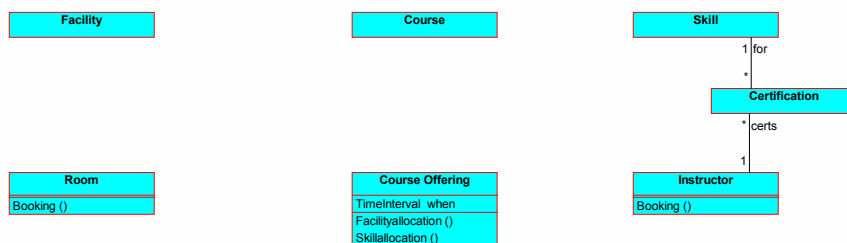


Templates and Aspects

- **Aspects** are *cross-cutting* requirements
 - Distribution
 - Security
- **Problem**
 - The support for an aspect is scattered across multiple classes in a system
 - Individual classes quickly become entangled with many cross-cutting requirements
- **Solution**
 - The design for each requirement is described in a *Composition pattern*.
 - Composition patterns are expressed using templates
 - The composition patterns are merged or *woven* together in an application

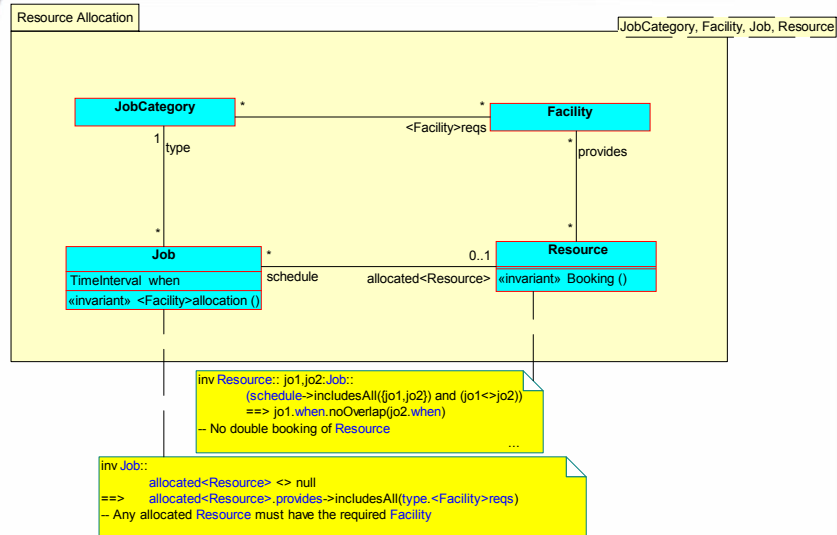
Seminar Booking Application

- Have resources such as rooms and instructors.
- Need to allocate courses to rooms and instructors based on room facilities and instructor skills

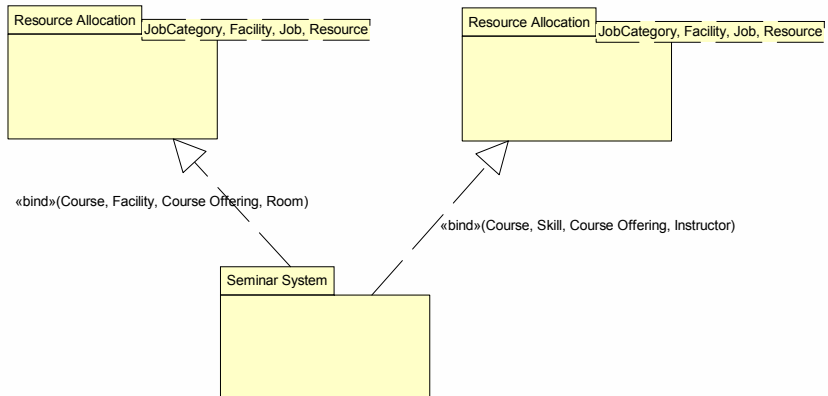


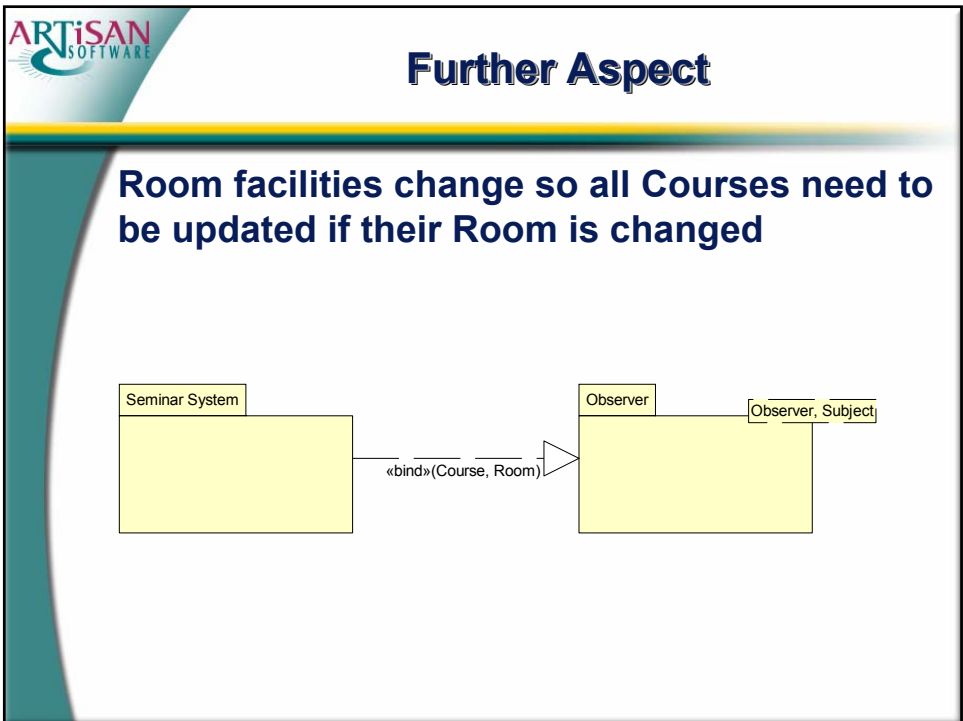
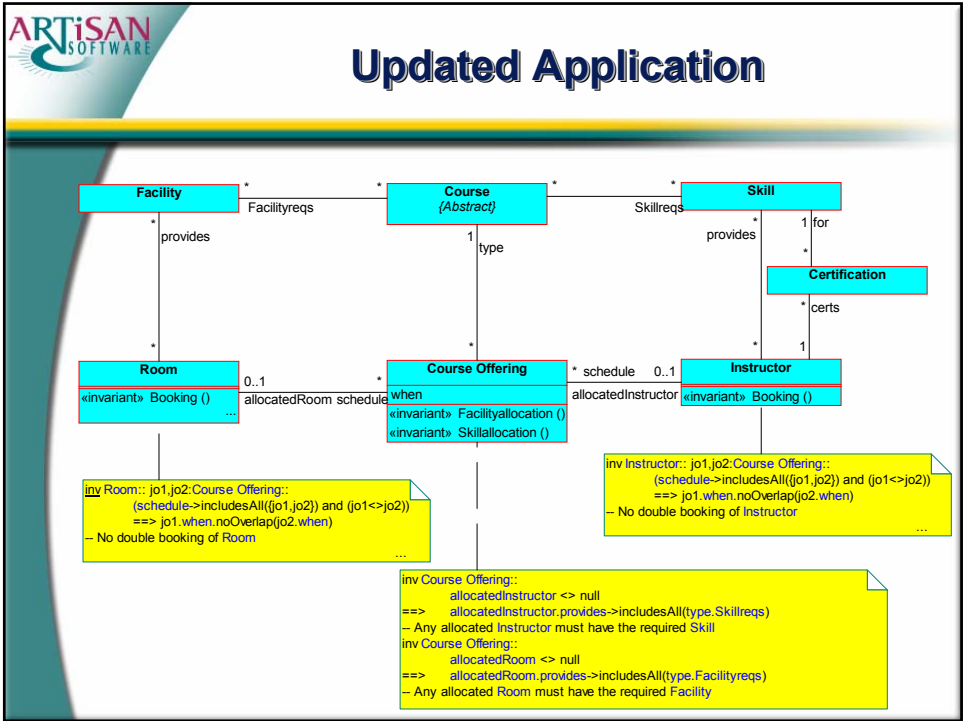
Adapted from an example in Catalysis (D'Souza and Wills)

Resource Allocation Template

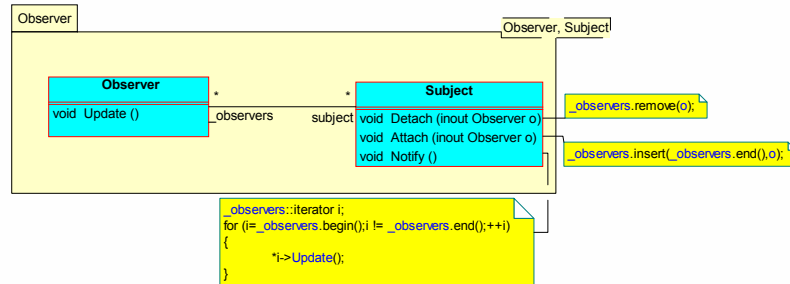


Two uses of Resource Allocation



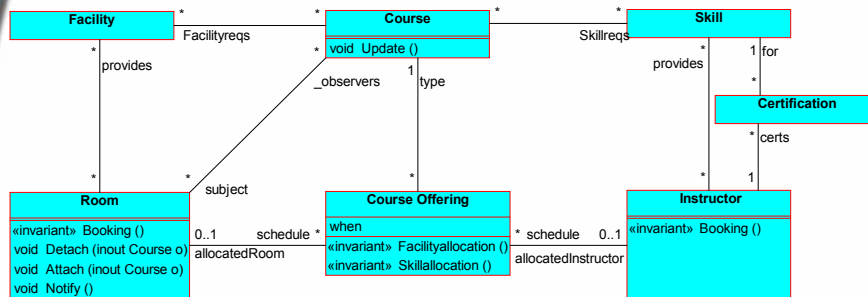


Observer Design

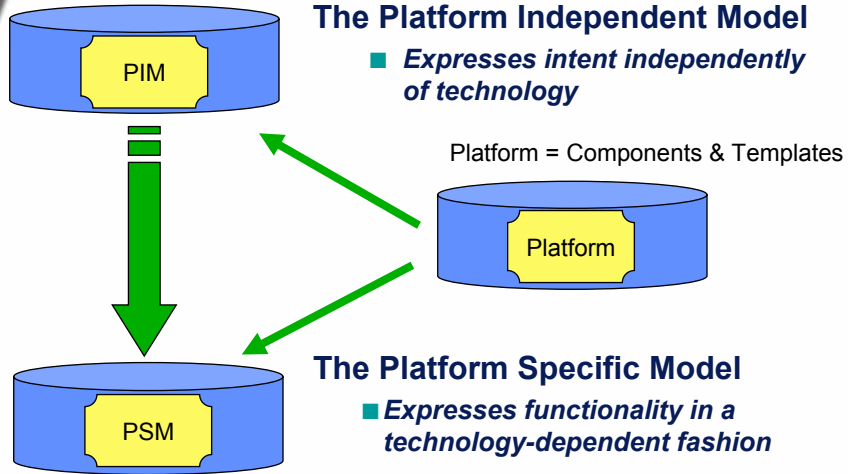


- Observers register themselves with a Subject
- When the subject is updated it notifies the Observer that an Update has occurred

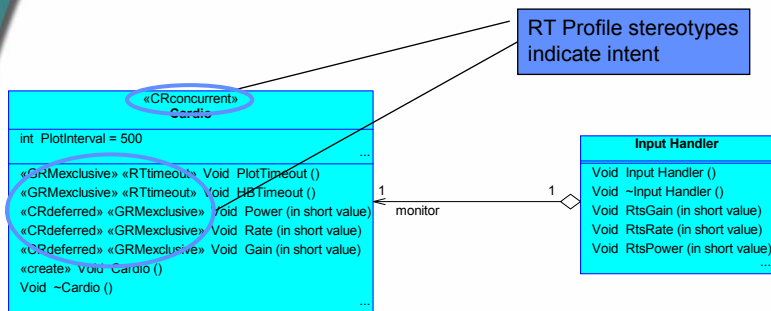
Updated Application



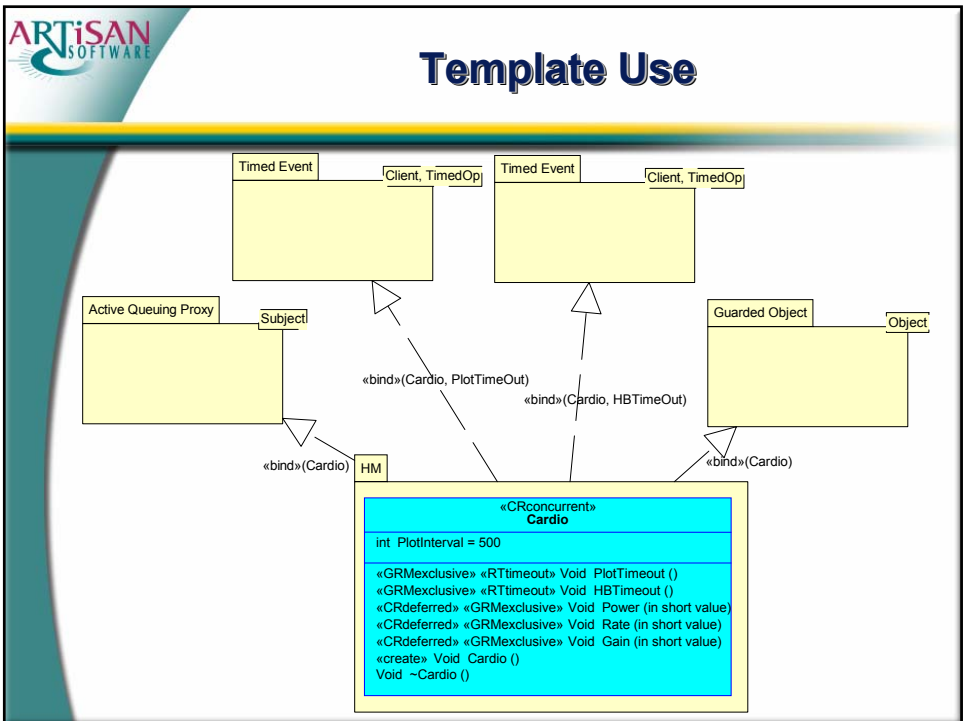
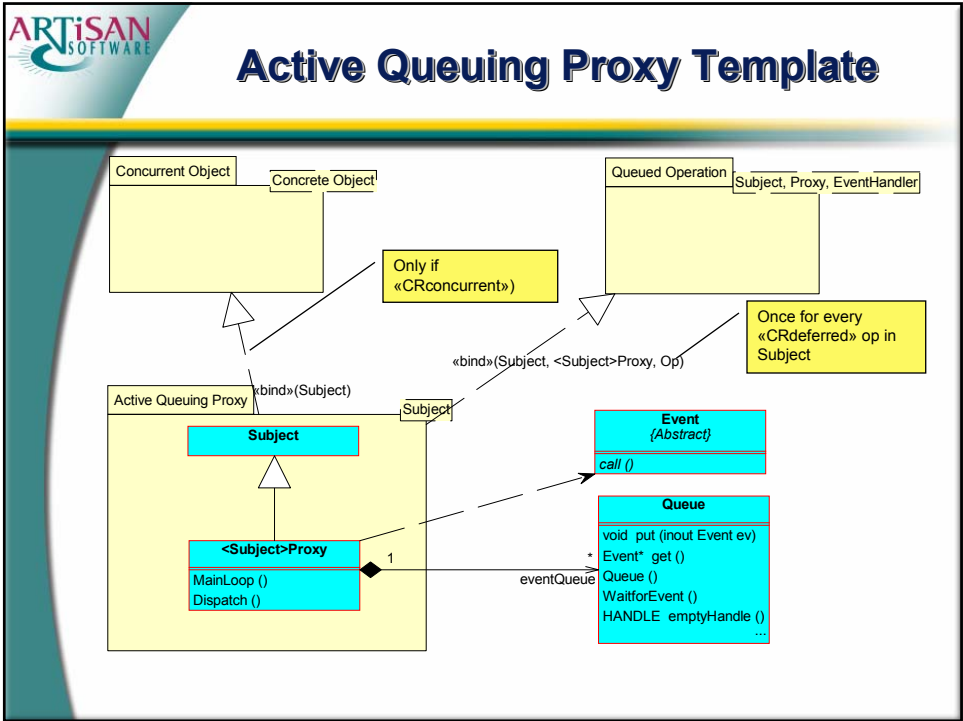
Use of Templates for MDA



Abstract Heart Monitor



AKA Platform Independent Model



The Future

- **Templates in QVT**
 - *MOF 2.0 Queries Views and Transformations*
 - ◆ *Queries on models.*
 - ◆ *Views on metamodels.*
 - ◆ *Transformations of models*
 - *Key technology for supporting MDA*
 - *Templates and Patterns are a critical part of many QVT submissions*
- **Templates in other languages**
 - *Templates may be added to EDOC, CWM*
 - *Harmonise with Aspect/J ...*
 - *Add to Java?*

Conclusion

- **Templates have finally reached maturity in UML 2.0**
- **They have a variety of uses including:**
 - *Pattern implementation*
 - *Aspect implementation*
 - *MDA transformations*
- **Implementation experience will improve the technology**
 - *Support for automation and scripting*
- **QVT will trigger further evolution**



Questions?

Web: www.artisansw.com

E-mail: MatthewH@artisansw.com