Model Driven Architecture: What's in it for Patterns

Ashraf Gaffar

Concordia University

Montreal

What are Models

-Models are abstractions of the real world

- Skip unnecessary details to amplify our manipulation capability
  - Meteorology mathematical models, parallel/grid processing
- Offer processing power and anticipate response
  - Simulation
- Abstract large body of data to accentuate key points
  - Comprehension and control
- Express particular views of a system
  - Incremental design models

Well-defined models are written in well-defined languages

- Bubble diagrams , OMT, UML, Source code

What are Models (cont.)

Carleton's University:

5 types of models

Conceptual, Physical, Mathematical, Statistical, Visualization

"*All models are wrong, but some are useful*", George E.P. Box

-Three mounds in the modeling complexity scale:

1-Unconstrained representation (simple throwaways)

Sketches, low fidelity prototypes, Bubble diagrams

2-Constrained representation (complex throwaways)

Images, flow charts , pie charts, visualization models

3-Processable representation (machineable artifacts)

Digital maps, math equations, UML diagrams, ER diagrams

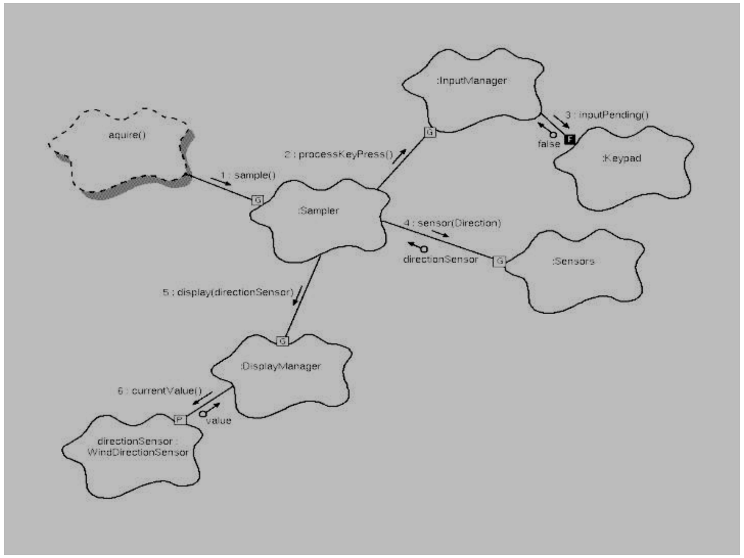
Not Reinventing the Wheel

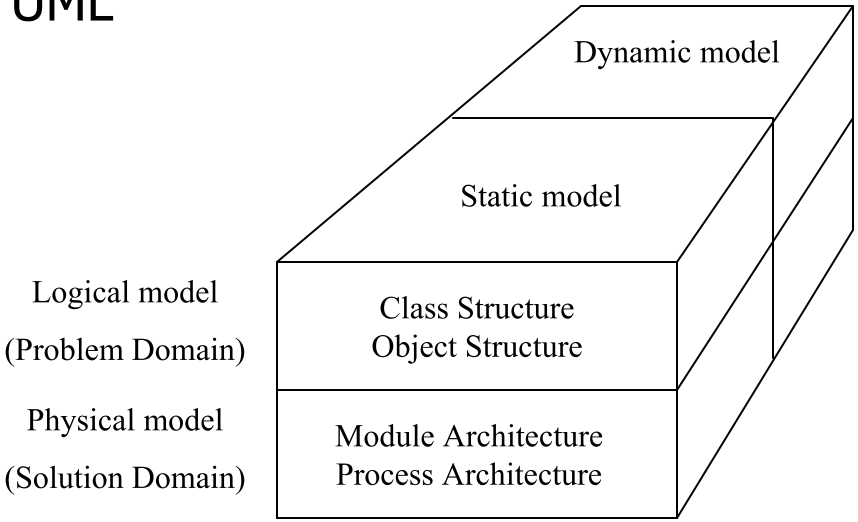
Hand made models (bubble diagrams)

A “thing”, and “ another thing”, and a “thing to connect them”

Standard models and methods

Coad/Yourdon, Booch method, OMT, UML





Dynamic model

Static model

Logical model

(Problem Domain)

Class Structure
Object Structure

Physical model

(Solution Domain)

Module Architecture
Process Architecture

Booch Method

The Antithesis of Models

Code is the ultimate goal

- minimize effort and time (valuable resources) directed towards building narrative models (documentations)
- Deliver a working code, not “loads of documents”

Even when documents have knowledge, they are byproducts

it is code that will do the job

documents only help designers communicate

Similarly, in User Interface Design:

When everything else fails, read the user manual

Perception: (Murphy’s law)

*The primary function of the design engineer is to make things difficult for the fabricator [programmer] and impossible for the serviceman [maintainer]

Racing Against Murphy’s Laws

More Murphy’s Laws

*A meeting is an event at which the *minutes* are kept and the *hours* are lost

*After all is said and done, a […] lot more is said than done

Have we “wrongfully” stereotyped documents and left them behind?

One Goal:

-Avoid doing the same work twice

Two approaches:

1- Reduce documentation

2- Use documentation that can evolve into code

Racing Against Murphy's Laws (cont.)

Approach 1: Reduce documentation.

Challenge: how to do it the right way

Just reducing or eliminating documents is not a solution

*If builders built buildings the way programmers wrote programs, then the first woodpecker that came along would destroy civilization

-Software complexity has grown far beyond the size of one-man comprehension, it is a TEAM WORK

-Effective communication between team members is a key to project success

Racing Against Murphy's Laws (cont.)

Extreme Programming, Agile Software Development:

Reduce documentation + **increase communication**

It is all about code, it is all about communication

Cockburn Metaphor: Game Theory

software development is " a cooperative game of invention and communication"

1- To deliver code

2- To prepare for continued play (a great way to say " *maintenance*")

*A complex system that works in invariably found to have evolved from a simple system that works → build a small system and grow it iteratively.

But beware: The reverse is not true

mind DOS-to-Windows bumpy evolution

We need to learn the rules

Racing Against Murphy’s Laws (cont.)

Approach 2: Use documentation that can evolve into code

Indeed, it is all about code, BUT good design matters

“the sooner you start, the longer it takes”
Fred Brooks, The Mythical Man Month

Beware of “how”!
*Nothing is as easy as it looks

The greatest perceived problem with modeling is its overhead

A typical modeling approach that we (still) use is “backward learning” :
Step 3: first start with your design
Step 2: Learn how to write “some” models for your current design
Step 1: learn the art of modeling and meta modeling (if you still have time)

MDA (Model Driven Architecture)

*To spot the expert, pick the one who predict the job will take the longest and cost the most

MDA solves the problem of experts by teaching them the “science of modeling” the right way

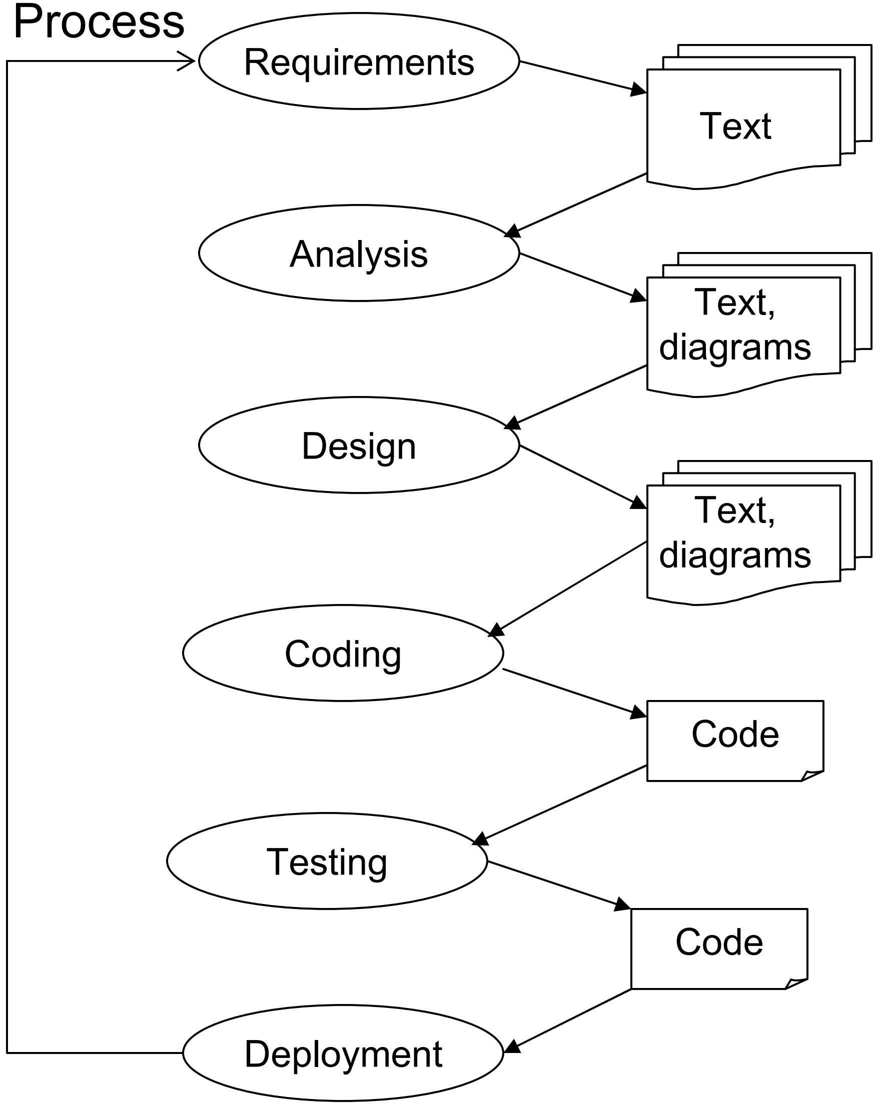
-It is a natural extension to the OO paradigm, but it is different
-It is based on UML, but it is much more
-It is about modeling, meta modeling, automatic transformations and code generation straight from our models
-We did many of the techniques, several times, in different approaches
It is time to unify these intellects and experiences
-The size of the effort, and the names of participating companies is promising

The Ultimate Goal:

To do the job once, right, and at high level: Executable Models

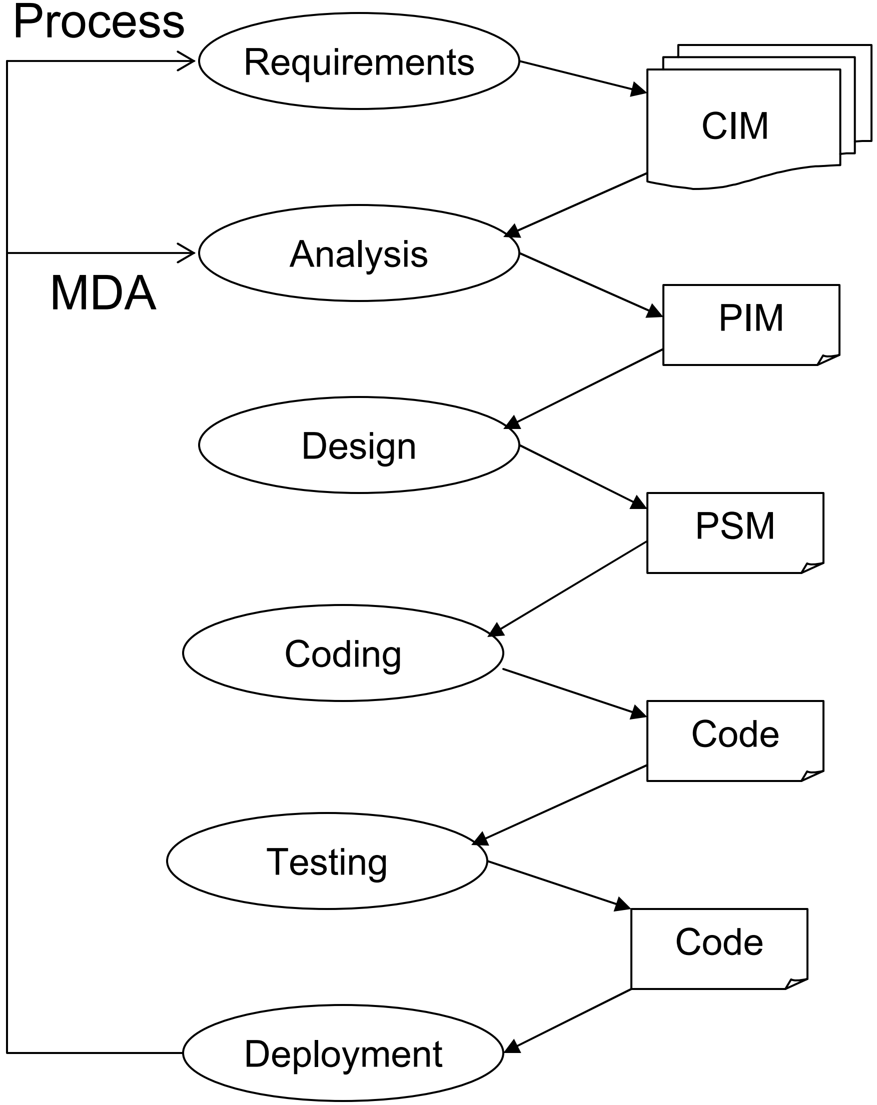
BUT it is a real challenge

Iterative Process



MDA Explained: The Model Driven Architecture (TM): Practice and Promise

Iterative Process



MDA

Requirements

CIM

Analysis

PIM

Design

PSM

Coding

Code

Testing

Code

Deployment

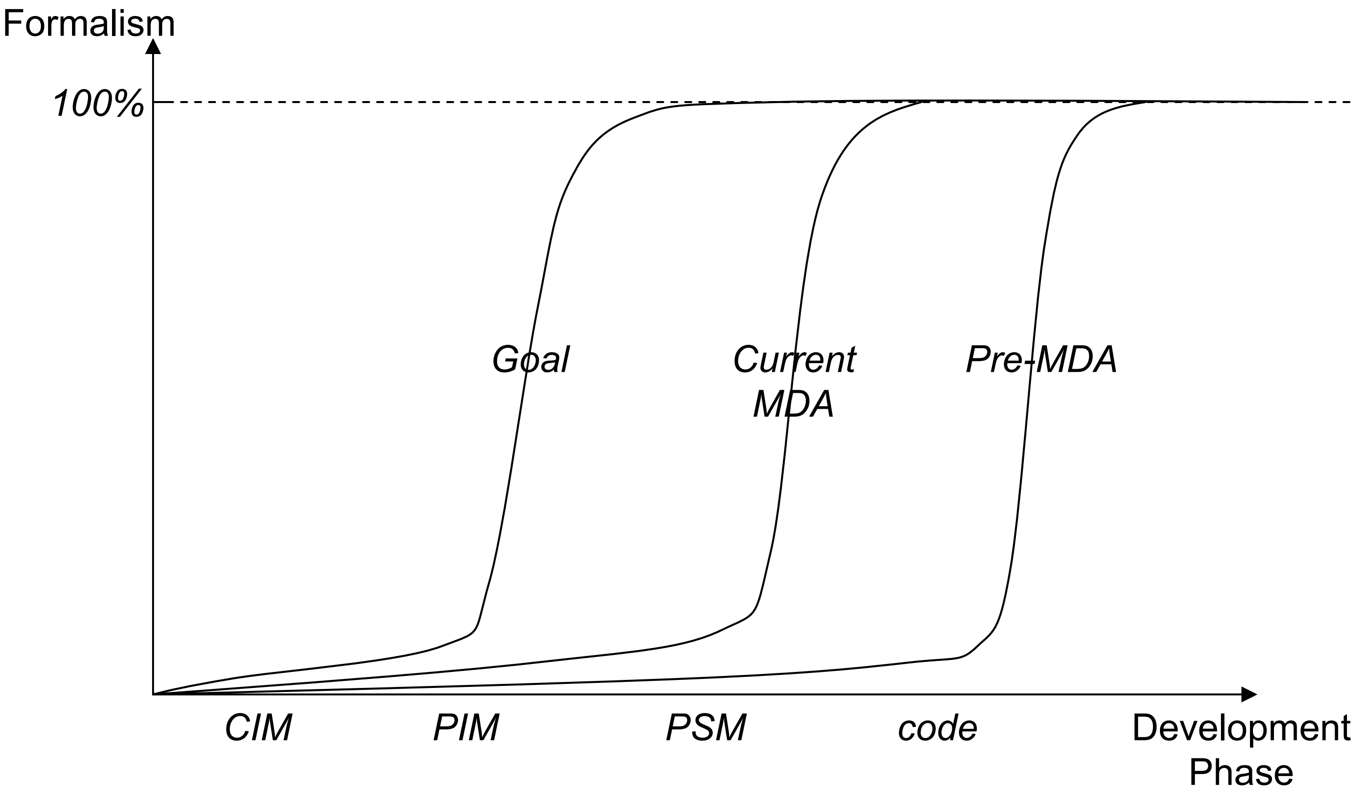
Problem Domain,
Business Use Cases,
Concepts, goals

Solution Domain,
Functionality, classes,
Computations

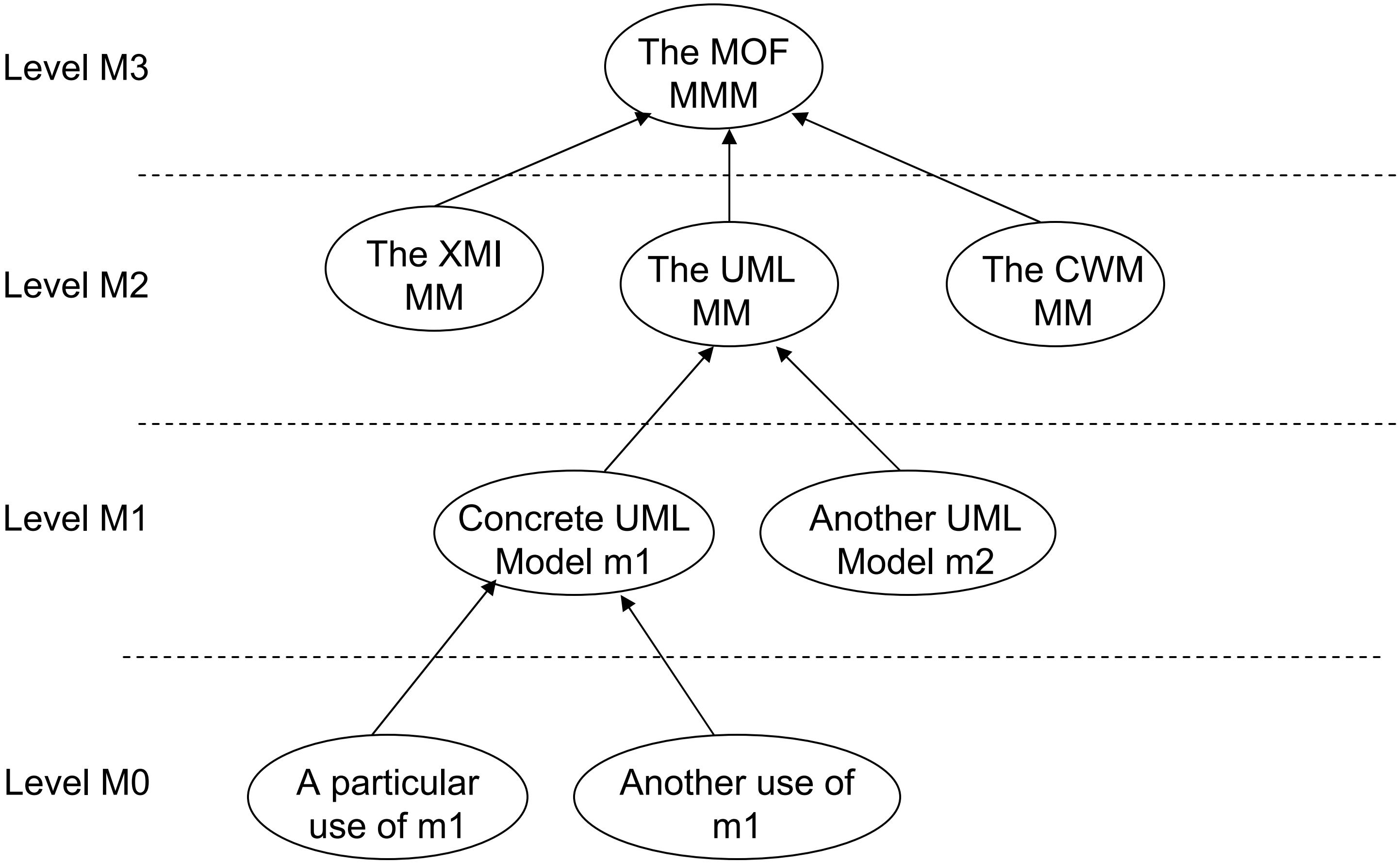
Platform details,
Concrete Classes,
objects, Interactions

MDA Explained: The Model Driven Architecture (TM): Practice and Promise

MDA Goal


Formalism
100%
Goal
Current MDA
Pre-MDA
CIM
PIM
PSM
code
Development Phase

MDA Components



Bezivin, J. From Object Composition to Model Transformation with the MDA

MDA Vision

Short Term Vision:

Efficient and seamless interoperability between applications, tools, databases

Shared Metadata

Standardization of Services

Platform Specifications

Long Term Vision:

Software ability to automatically discover its environment, dynamically adapt to it, and interact with it

Reflections, introspections, agents, …

-BUT in a universal approach

Applying MDA to Patterns

-Patterns are effective way of communication

Concept: Wrapped collection of preprocessed knowledge

Contents: Proven experience for reuse

Alternatives: Direct interaction with a “Mentor”

-Patterns (could be) everywhere

We focus on HCI domain:

Understanding the user is difficult

Satisfying the user is even more difficult

Patterns can certainly help

Patterns and Reuse

-Context today

Several platforms, languages, interfaces

Numerous pattern collections

Heterogeneity is growing

-Popularity creates a problem of scalability

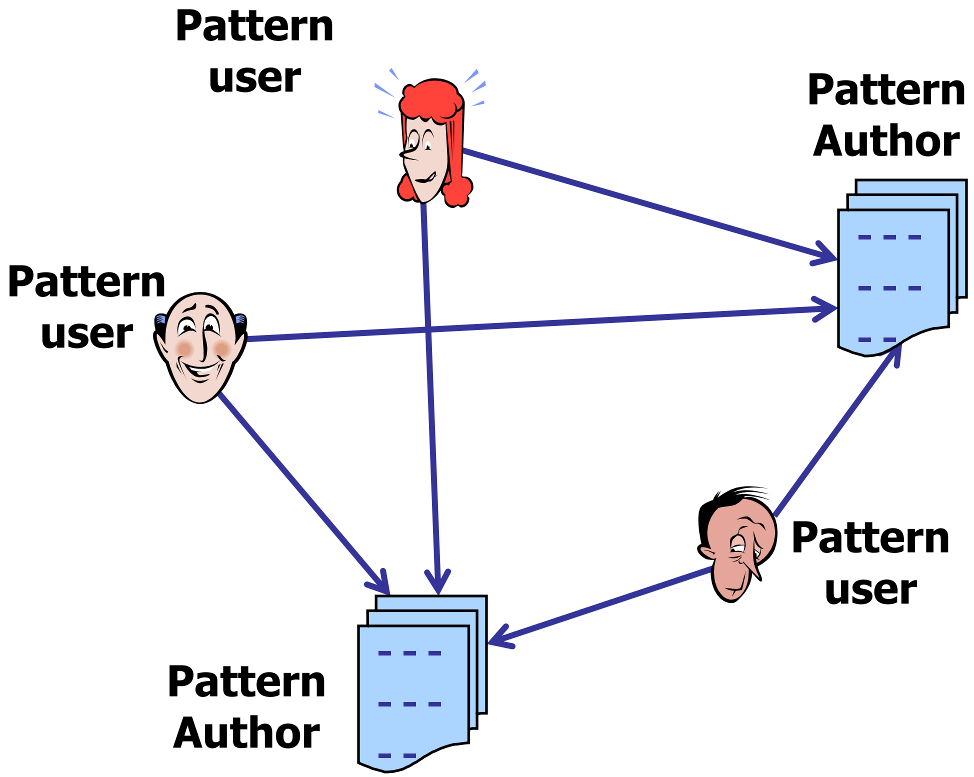
Magnitude, redundancies, contradictions between patterns

Improvised techniques for searching and applying

Negative effect on reuse

Pattern Sharing: Ideal Situation

-How to effectively use patterns from different collections?



Pattern user

Pattern Author

Pattern user

Pattern user

Pattern Author

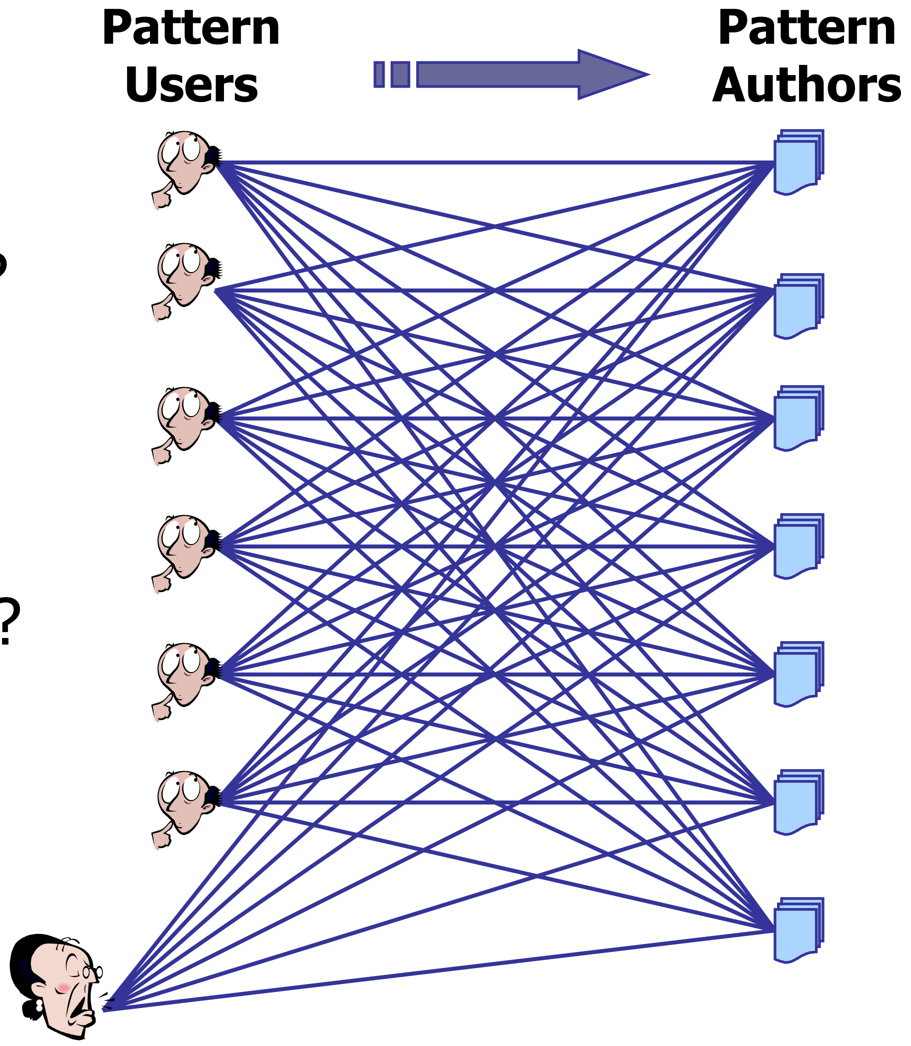
Pattern Sharing: Actual Situation

-Dissemination or mining?

-User responsibility?

-Scalability?

-Visibility of new patterns?



Goal: Effective Pattern Reuse

-Integration of disparate efforts

Preserving knowledge

A unifying platform with modular “plug ins”

-Model based techniques

Design creativity vs. standards

can they coexist?

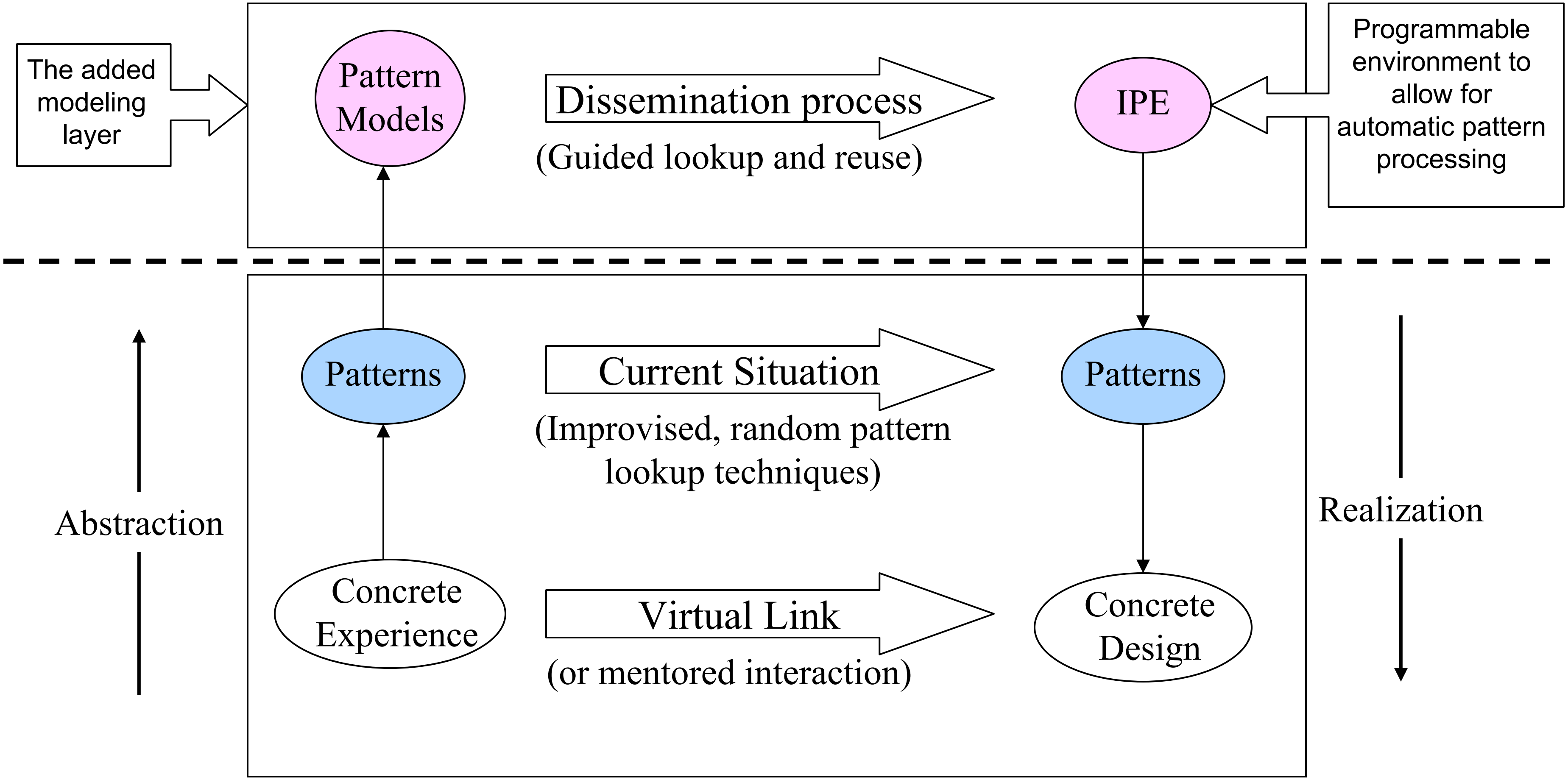
Interoperability

Abstraction, forward compatibility

-A process to support a complete pattern lifecycle

Generation, dissemination, assimilation

The Pattern Lifecycle, adding a Modeling layer



The added modeling layer

Pattern Models

Dissemination process

(Guided lookup and reuse)

IPE

Programmable environment to allow for automatic pattern processing

Abstraction

Patterns

Current Situation

(Improvised, random pattern lookup techniques)

Patterns

Realization

Concrete Experience

Virtual Link

(or mentored interaction)

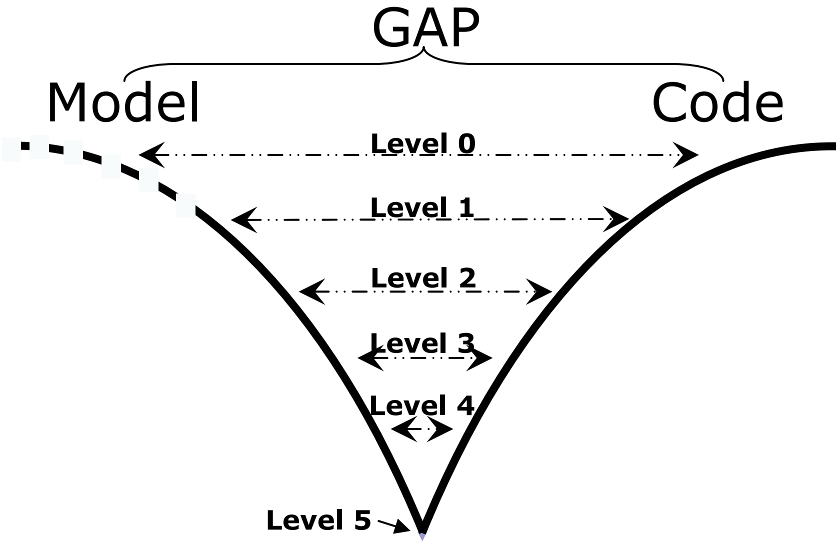
Concrete Design

Tools for Building Pattern Models

Borland:

Modeling Maturity Levels

- Level 0: No specification
- Level 1: Textual
- Level 2: Text with Diagrams
- Level 3: Models with Text
- Level 4: Precise Models
- Level 5: Models only



http://www.borland.com.cn/news/download/files/JBuilder.pdf

MDA Benefits to Patterns

-Portability

From Author to User

-Interoperability

Platform independence

-Integration

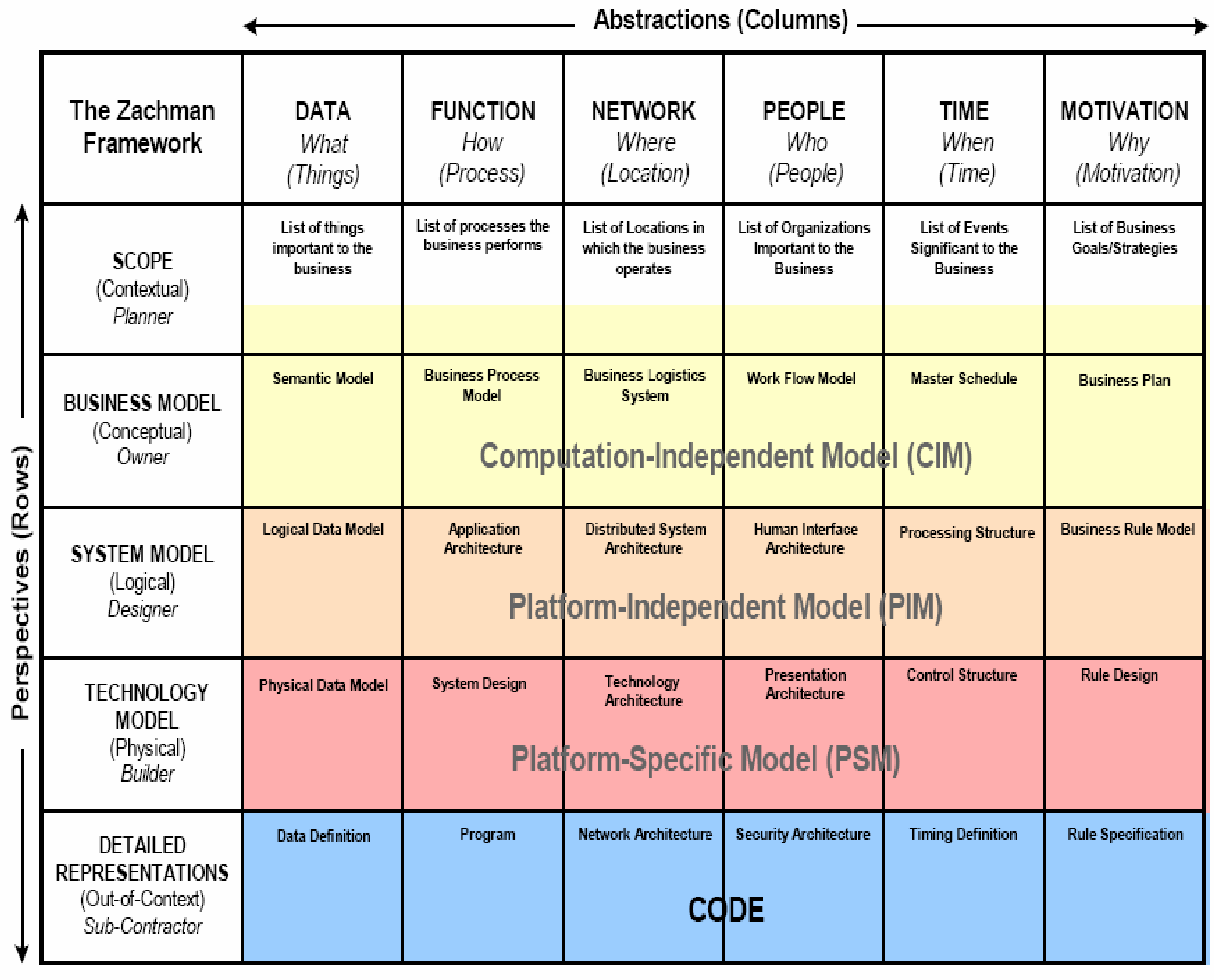
Into User’s Design Process

Vision:

Code Generation

Both Dissemination and Assimilation

The Start: MDA and Zachman Framework



Abstractions (Columns)

Perspectives (Rows)

The Zachman Framework	DATA *What* *(Things)*	FUNCTION *How* *(Process)*	NETWORK *Where* *(Location)*	PEOPLE *Who* *(People)*	TIME *When* *(Time)*	MOTIVATION *Why* *(Motivation)*
SCOPE (Contextual) *Planner*	List of things important to the business	List of processes the business performs	List of Locations in which the business operates	List of Organizations Important to the Business	List of Events Significant to the Business	List of Business Goals/Strategies
BUSINESS MODEL (Conceptual) *Owner*	Semantic Model	Business Process Model	Business Logistics System	Work Flow Model	Master Schedule	Business Plan
SYSTEM MODEL (Logical) *Designer*	Logical Data Model	Application Architecture	Distributed System Architecture	Human Interface Architecture	Processing Structure	Business Rule Model
TECHNOLOGY MODEL (Physical) *Builder*	Physical Data Model	System Design	Technology Architecture	Presentation Architecture	Control Structure	Rule Design
DETAILED REPRESENTATIONS (Out-of-Context) *Sub-Contractor*	Data Definition	Program	Network Architecture	Security Architecture	Timing Definition	Rule Specification

Computation-Independent Model (CIM)

Platform-Independent Model (PIM)

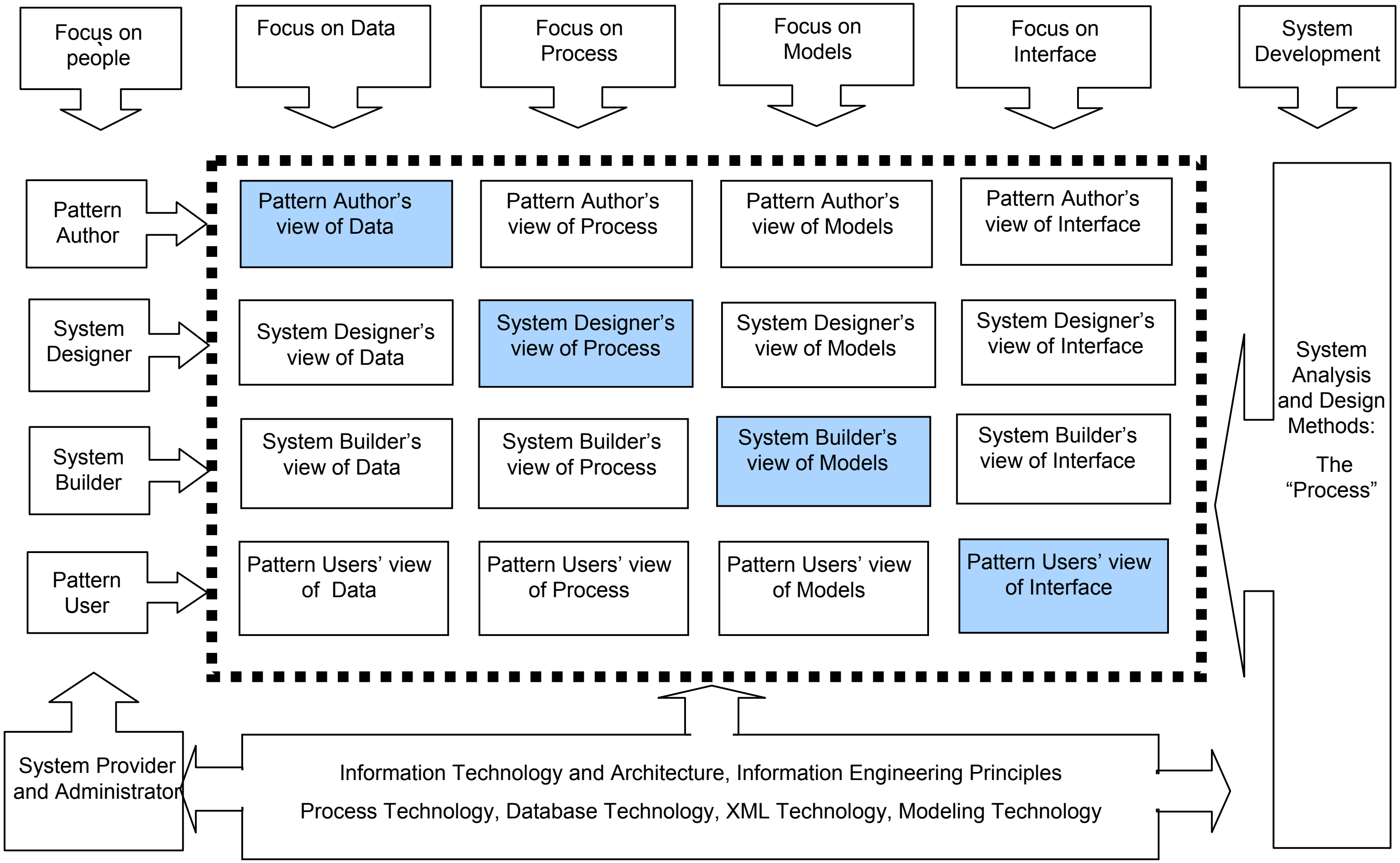
Platform-Specific Model (PSM)

CODE

OMG Group at http://www.omg.org

Using Zachman Framework (partially)

To start with: Patterns are data, seen differently by different stakeholders



Focus on people	Focus on Data	Focus on Process	Focus on Models	Focus on Interface
Pattern Author	Pattern Author's view of Data	Pattern Author's view of Process	Pattern Author's view of Models	Pattern Author's view of Interface
System Designer	System Designer's view of Data	System Designer's view of Process	System Designer's view of Models	System Designer's view of Interface
System Builder	System Builder's view of Data	System Builder's view of Process	System Builder's view of Models	System Builder's view of Interface
Pattern User	Pattern Users' view of Data	Pattern Users' view of Process	Pattern Users' view of Models	Pattern Users' view of Interface

System Development

System Analysis and Design Methods:

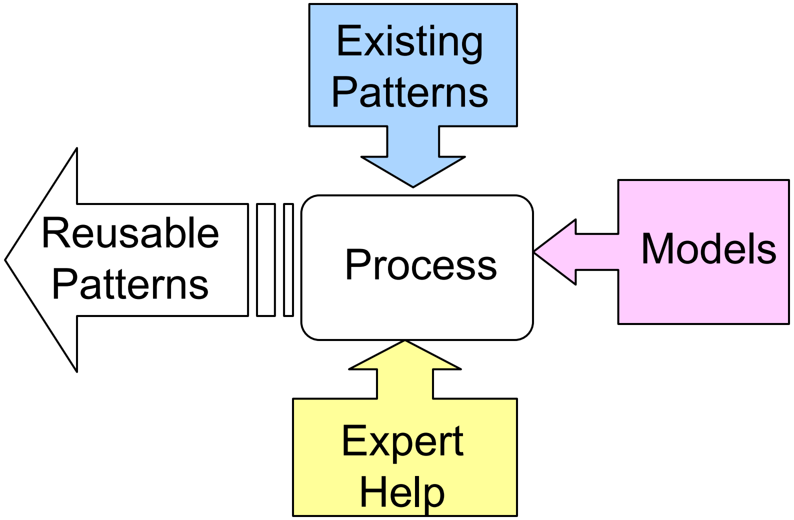
The "Process"

System Provider and Administrator

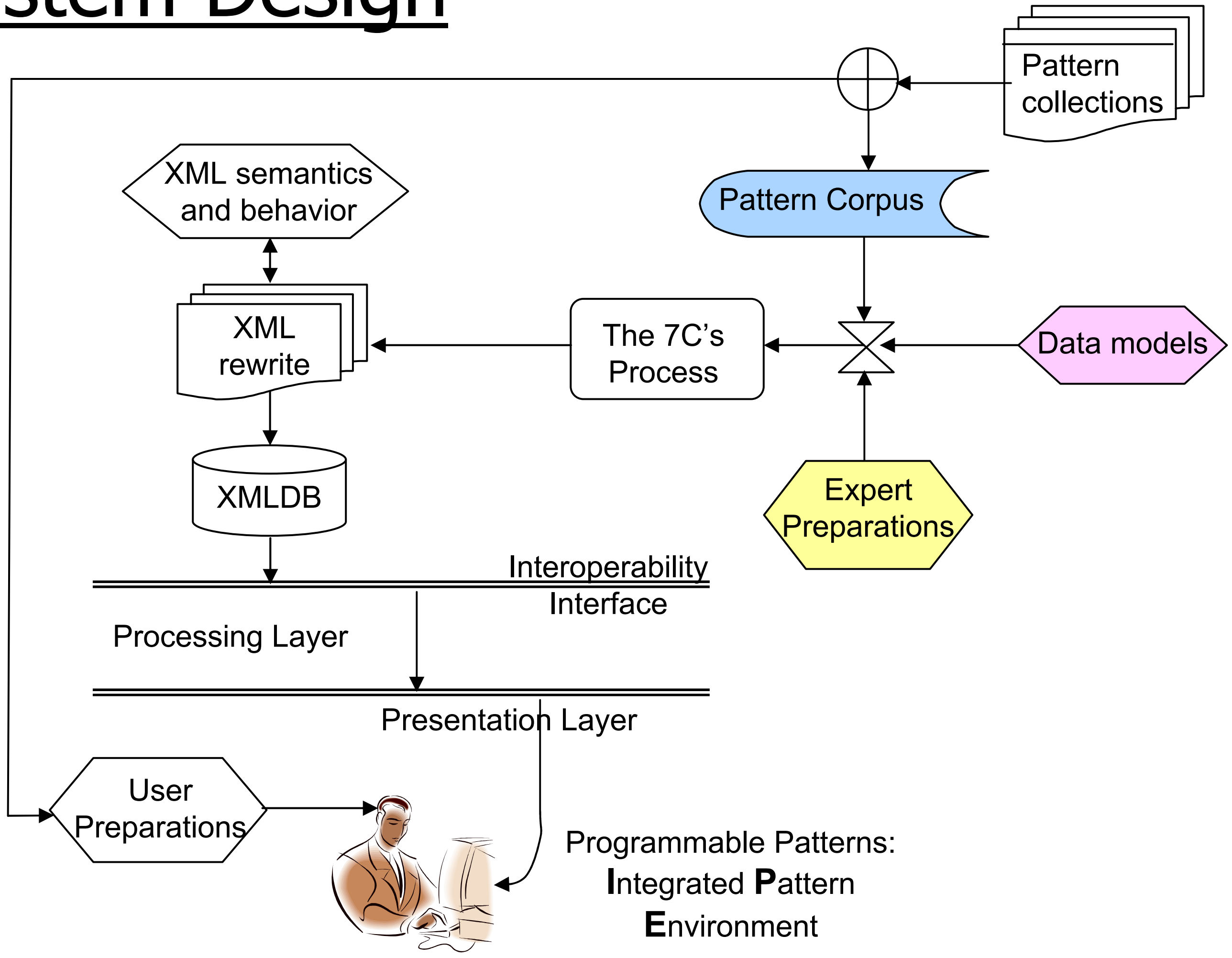
Information Technology and Architecture, Information Engineering Principles

Process Technology, Database Technology, XML Technology, Modeling Technology

Workflow to Automate the Pattern Process



The System Design



Pattern collections

Pattern Corpus

XML semantics and behavior

XML rewrite

The 7C's Process

Data models

XMLDB

Expert Preparations

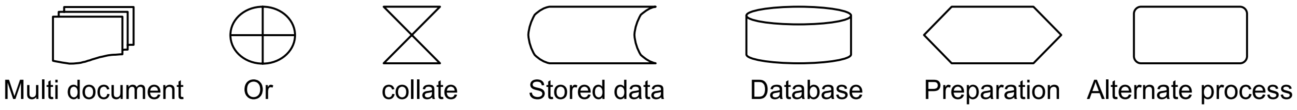
Interoperability Interface

Processing Layer

Presentation Layer

User Preparations

Programmable Patterns:
Integrated **P**attern
Environment



Multi document

Or

collate

Stored data

Database

Preparation

Alternate process

References

-OMG Group at http://www.omg.org

-Bezivin, J. From Object Composition to Model Transformation with the MDA, Proc. TOOLS'USA, Volume IEEE TOOLS-39, Santa Barbara, August 2001

-Poole, J. D., MDA: Vision, Standards, and Emerging Technologies, Position paper submitted to ECOOP 2001, Workshop on Metamodeling and Adaptive Object Models, April 2001

-Kleppe, A., Warmer, J., Bast, W. MDA Explained: Practices and Promises, Addison Wesley,2003

-Murphy's Laws Web sites:

http://www.murphys-laws.com/murphy/murphy-laws.html

http://dmawww.epfl.ch/roso.mosaic/dm/murphy.html

Questions?

***All things are possible, except skiing through a revolving door**

If you have any suggestion or comments, please contact me
gaffar@ieee.com