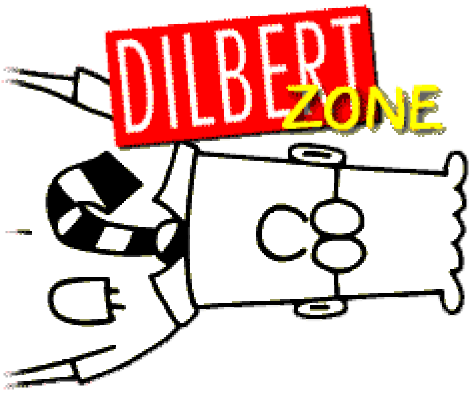
Flexible Insurance Systems

Part 5: The Pitfalls of Business Rules & Meta Systems

Agenda

- The Hyper Flexible System
- Make Your System Adaptable with Business Rules?
- Configuration Management

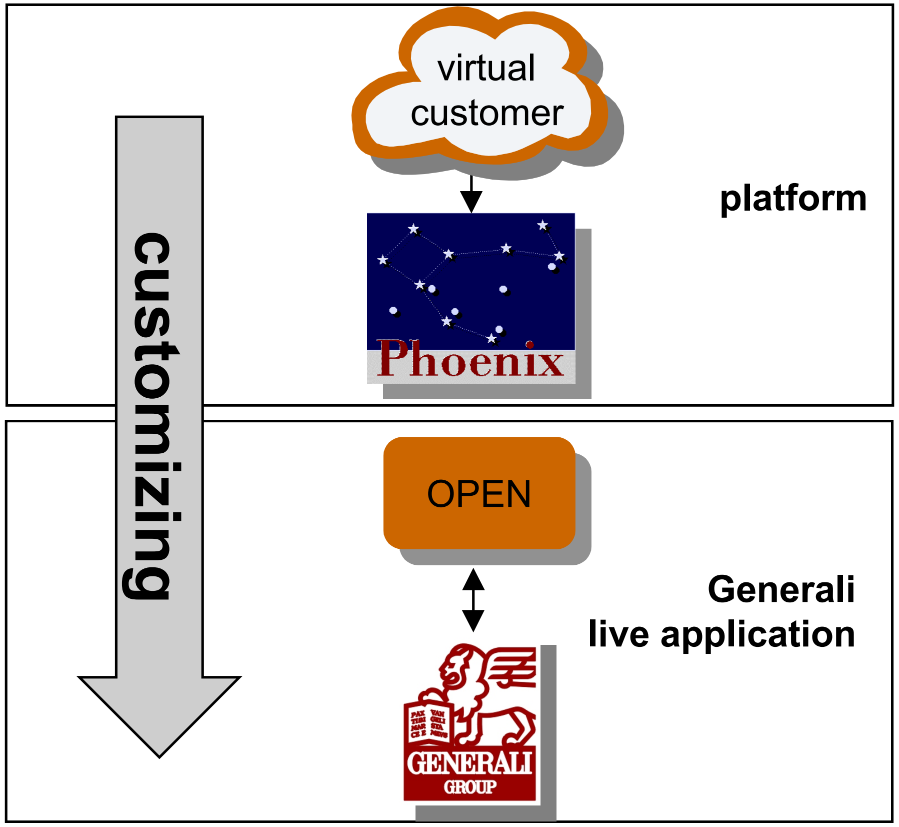
The Hyper Flexible System



- Management Speak: We want a very flexible system
- Translation: We don’t know what we want

Anti-Pattern

Example:
Part of Phoenix‘s Project Order



Phoenix is a standard platform for insurance applications

Phoenix will not implement special requirements for a special customer but will implement the average requirements of an average customer in the german speaking insurance industry.

Problems

- Who should the analysts talk to?
- How do you deal with a board of 6+ different companies that formulate requirements that differ significantly?

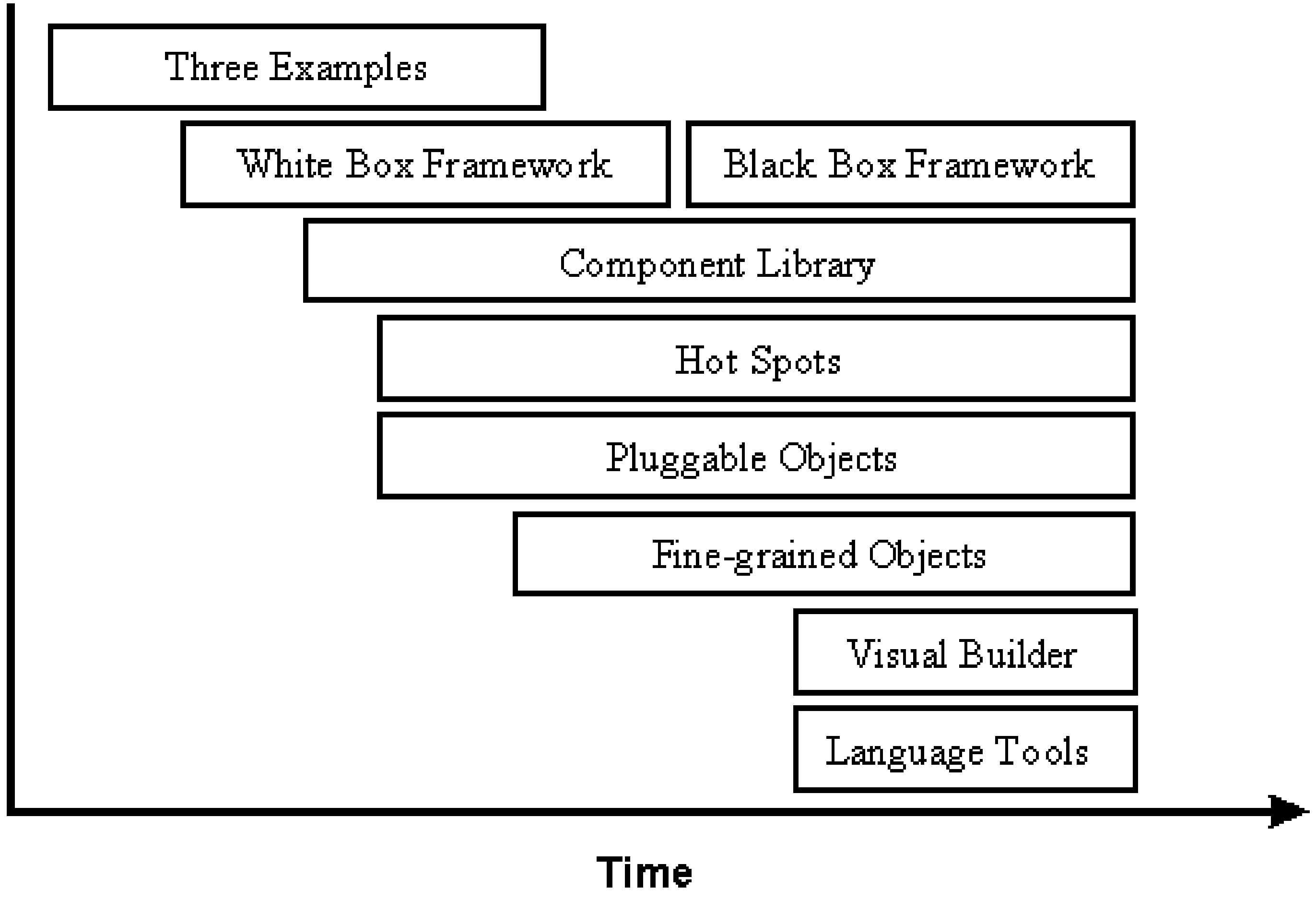
- Natural Reactions from the team:
  - Let‘s do that in the custumizing phase
  - Let‘s put it into the rule system

Normal Way to build Platforms and Frameworks

- Build an application for 1 or 2 customers
- Expand it to more and more customers

- But NEVER deal with a group of 7+ customers before you have a running solution ..

From the Frameworks Community we learn ..



Ralph Johnson, Don Roberts: Patterns for framework evolution.

http://st-www.cs.uiuc.edu/users/droberts/evolve.html

The Hyper Flexible System
A summary of facets

- Software Engineering
  - Free flying rules
  - Heap of Rules
- Performance
  - Rules in a relational database
- Security
  - Rules that are not versioned properly
  - Eidting rules in a production database
- Pseudo flexibility
  - End users fail to write rules or they need code to access object contents (like special getters)

Make Your System Adaptable with Business Rules

Anti-Pattern

The basic idea for rule systems sounds rather attractive for top management ...

From the marketing brochures of a business rule engine vendor ..

- You want a flexible system
- You want it now and don‘t want to employ programmers to change your system
- Then just factor out all business logic into business rules that can be entered by the end users themselves

Backup slide marketing speech...

PRESS RELEASE:

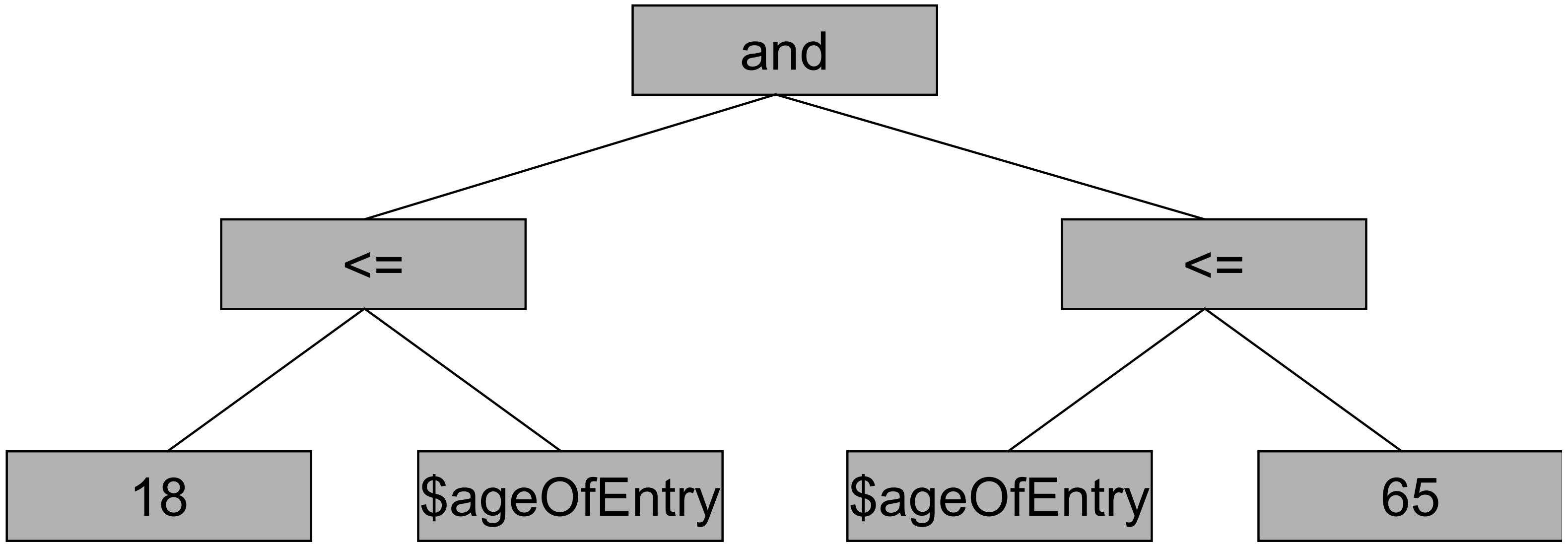
Doe Software's RuleEngine Lets Business Managers Directly Manage and Rapidly Adapt Business Rules

BILLY‘s HILL, Calif., October, 1999—Doe Software, today introduced a new version of its RuleEngine that fundamentally changes how smart customer applications are developed and maintained. Until now, these systems have been designed, built and managed solely by professional programmers.Doe Software's RuleEngine puts control of the systems into the hands of the business managers who actually run the business. And, because the underlying business rules "intelligence" resides outside the main program, managers and analysts can quickly make critical changes to respond to dynamic changes in business.

„RuleEngine is the first product I've seen that makes business rules understandable by mere mortals," said Gerry Miller, president of Mainstream Consulting, an industry consulting firm. "In apps, where rules are the key to customer interactions that are consistent and dynamic, this product is going to have immense value."

Doe‘s RuleEngine is THE business rules solution for delivering responsive, personal service to the customer. RuleEngine employs an easily adaptable, easy to understand language to create interrelated business rules for applications. These rules create a more satisfying experience for customers, and a more productive, profitable environment for enterprises.

Backup slide: Simple example for a rule in a life insurance system



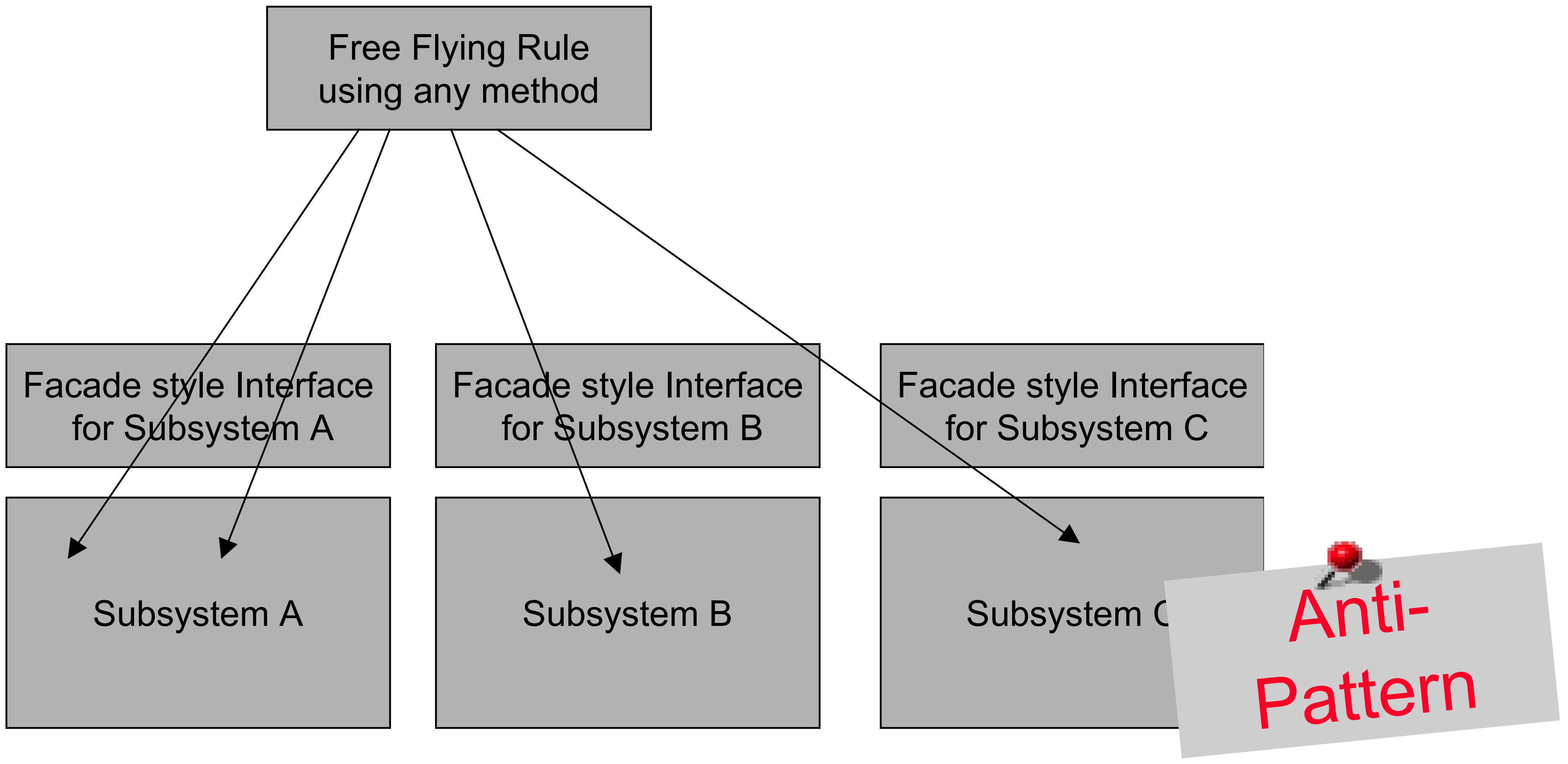
attractive for top management... But may become a nightmare for software architects

- Rules ARE code!!!

- Rules may easily break encapsulation, abstraction and subsystem borders
- Rules need to be treated like code in config management
- Rules need to be tested like code

- Rules are tempting in order to defer important analysis decisions
We don‘t know how this works – let‘s put it into the rule system and do it in the customization phase

Free flying rules



Configuration Management for a hyper flexible system

Configuration management for a hyper flexible system

- Configuration items are ...

- Code
- Database definitions
- Rules
- Print definitions
- Business process definitions
- Product definitons

- Question: How do you contain changes for the purposes of testing

Configuration management for a hyper flexible system

If you want that played really safe ...

Lessons Learned

- We removed large parts of our rule system
- We try to contain rules in a single subsystem. Therefore interfaces need to redesigned and improved.

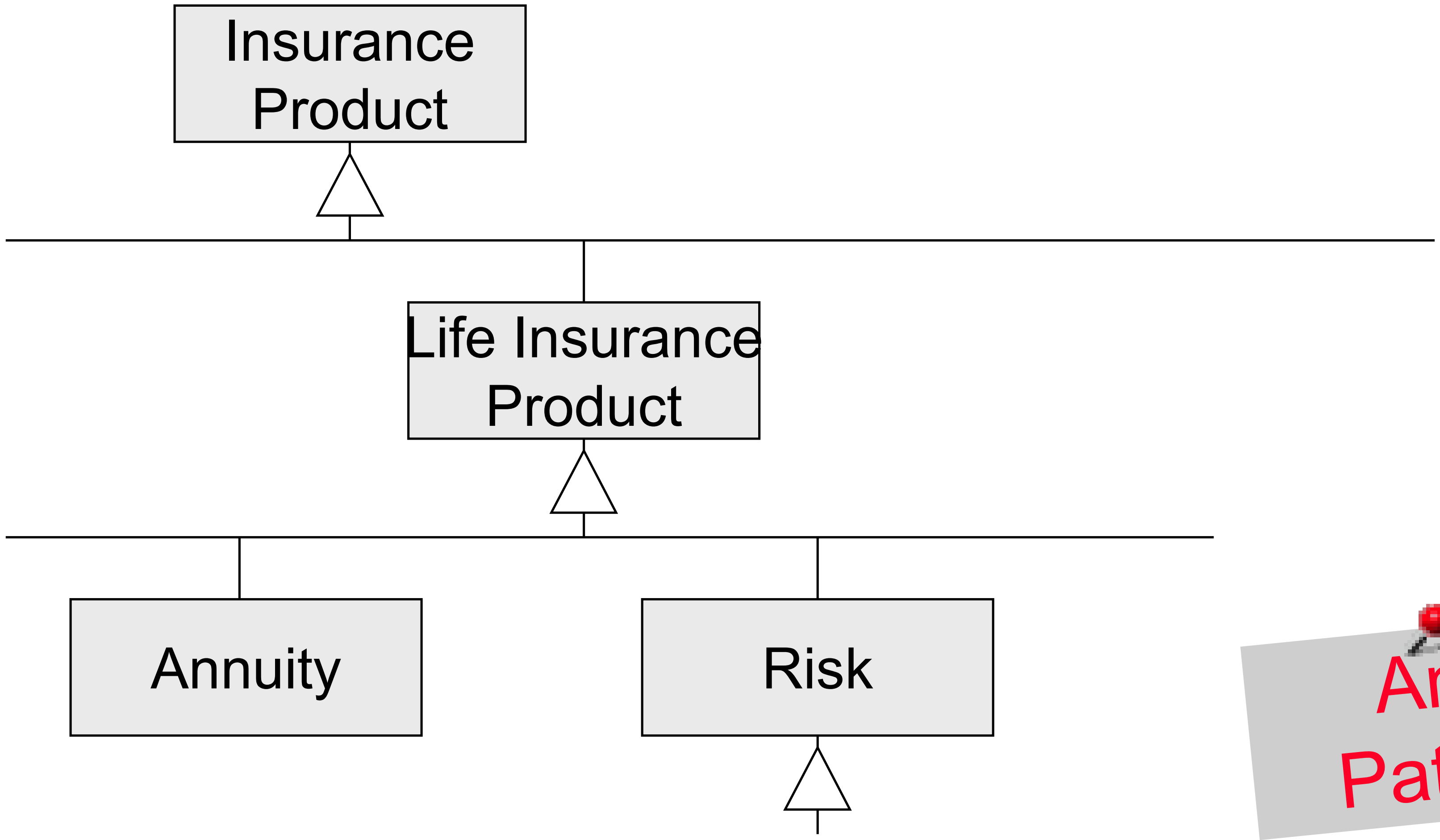
- A universal rule system has turned out to be expensive, dangerous and only pseudo flexible – as you might get lost in a heap of rules ...

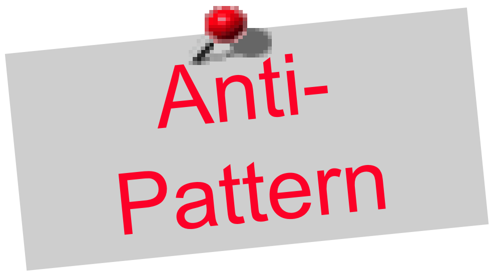
The Object-Oriented Meta-System

The Object-Oriented Meta-System

- Context: How to design a product model
- Context: I did just learn that object technology is the best way to build flexible systems that promote reuse.
- Alas – I have to be „object-oriented“ and use lots of inheritance
- But: A product driven insurance system is inherently a meta-system – and meta-systems do not REQUIRE inheritance

Excerpt from an „object-oriented“ product definition





The Object-Oriented Meta-System Diagnosis

- Product models that include product building blocks with deep inheritance hierarchy
  - You may find inheritance trees with a depth of 8+.

- Pitfall: Mixing up IS-A with HAS-A
- Pitfall: Bad use of inheritance

Anti-Pattern

Lessons

- Meta-Systems may be built using an OO language or a 3GL
  It does not really make a difference
- Meta-Systems that use semantic classes are not really meta and therefore pseudo flexible as you have to program a new semantic class whenever you need a new product element
  - change your meta-element instead of using it to define new elements).

Wrap Up for Part 5

List of Pitfalls from White Paper

- The Hyper Flexible System ✓
- The Object-Oriented Meta-System ✓
- Free Flying Rules ✓
- Rules Without Versions ✓
- Rules in a Relational Database ✓
- Heap of Rules ✓
- Semantic Product Components ✓
- Pseudo Flexibility ✓

List of Pitfalls (continued) from White Paper

- End User writes Rules ✗
- Rule Editor on the Production Database ✗
- The Meta System that does not Perform ✗

If you are interested in more detail

- The Pitfalls of Business Rules and Meta Systems (1999)

  http://www.objectarchitects.de/ObjectArchitects/papers/WhitePapers/