

Introduction to Aspect-Oriented Software Development

Prof. Jean-Marc Jézéquel

(Univ. Rennes 1 & INRIA)

Triskell Team @ IRISA

Campus de Beaulieu

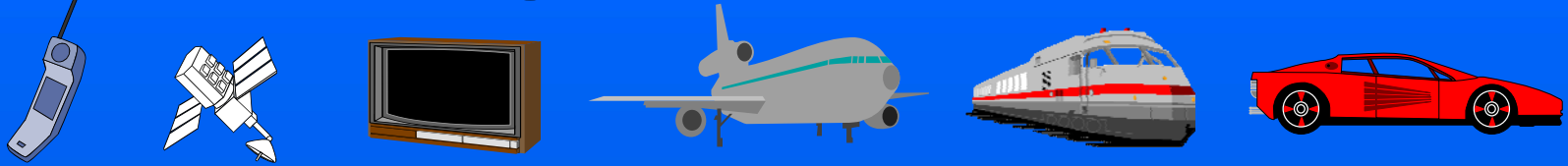
F-35042 Rennes Cedex

Tel : +33 299 847 192 Fax : +33 299 847 171

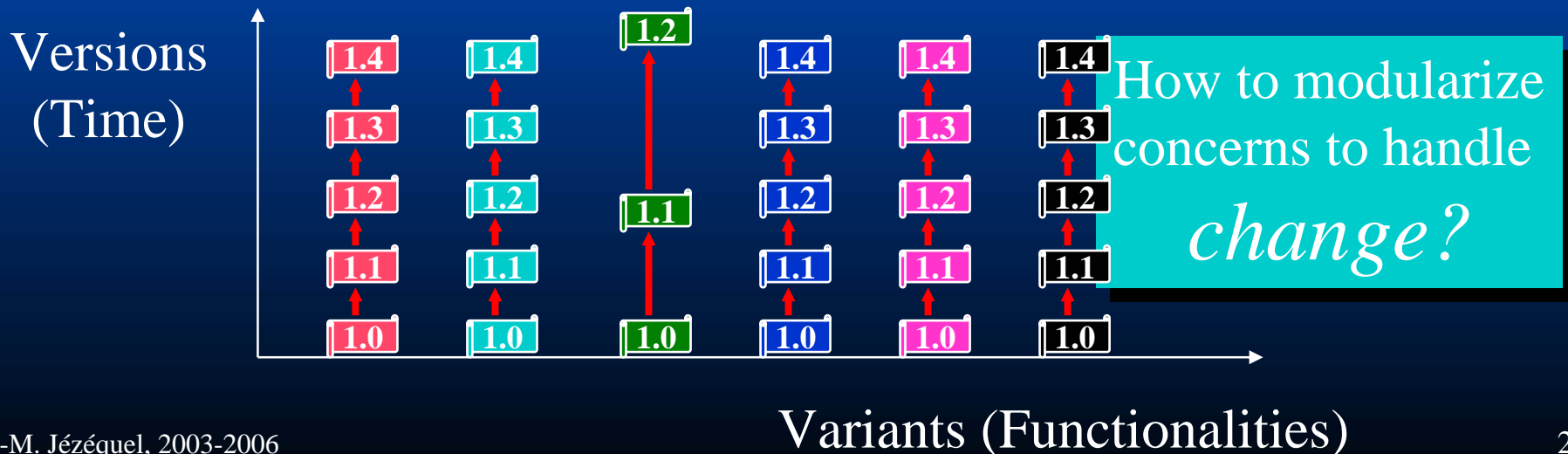
e-mail : jezequel@irisa.fr

<http://www.irisa.fr/prive/jezequel>

Modern Software Problems

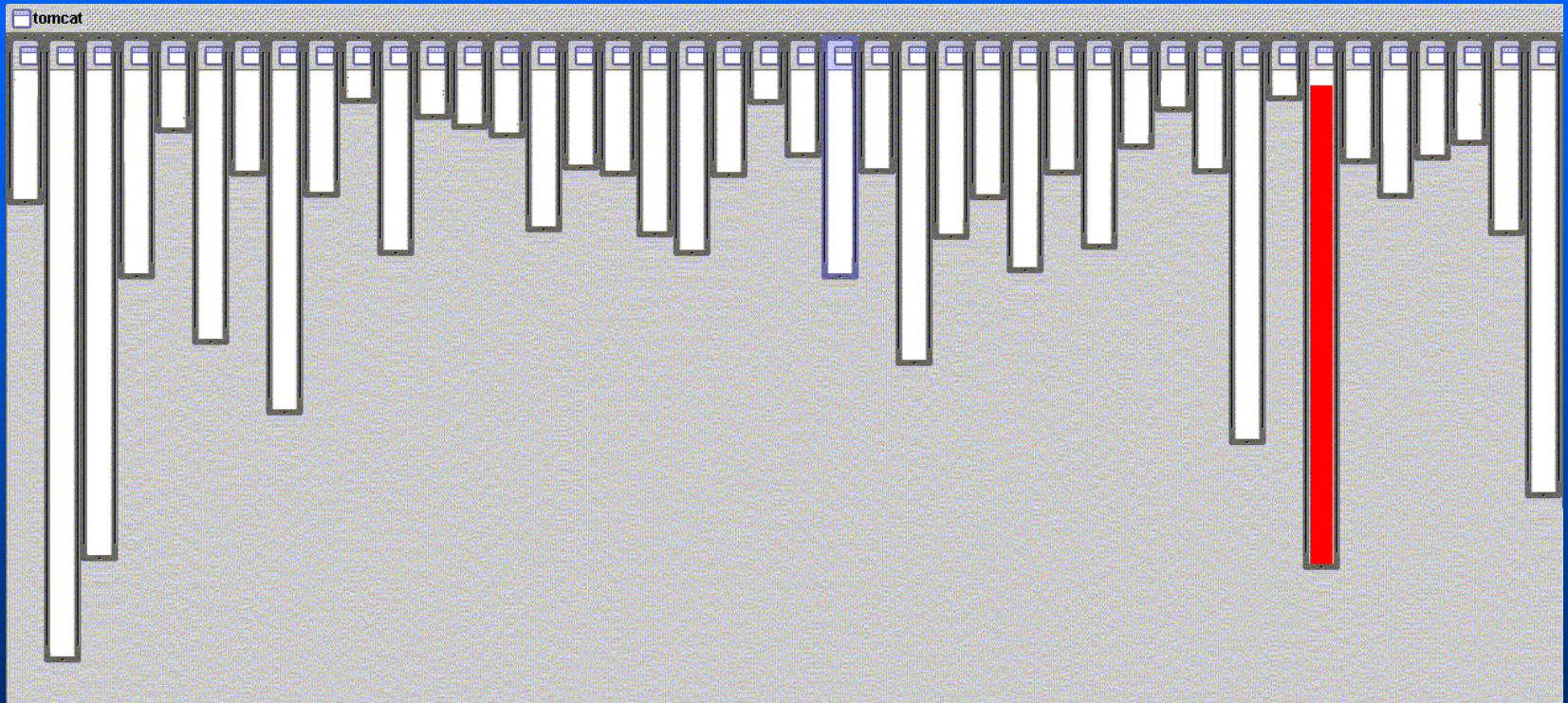


- Importance of non-functional aspects
 - Persistency, distribution, monitoring, etc.
 - » Several possible solutions
 - quality of service : reliability, latency, performance...
- Flexibility of functional aspects: Product Lines
 - notion of *product lines* (space, time)



Example: good modularity

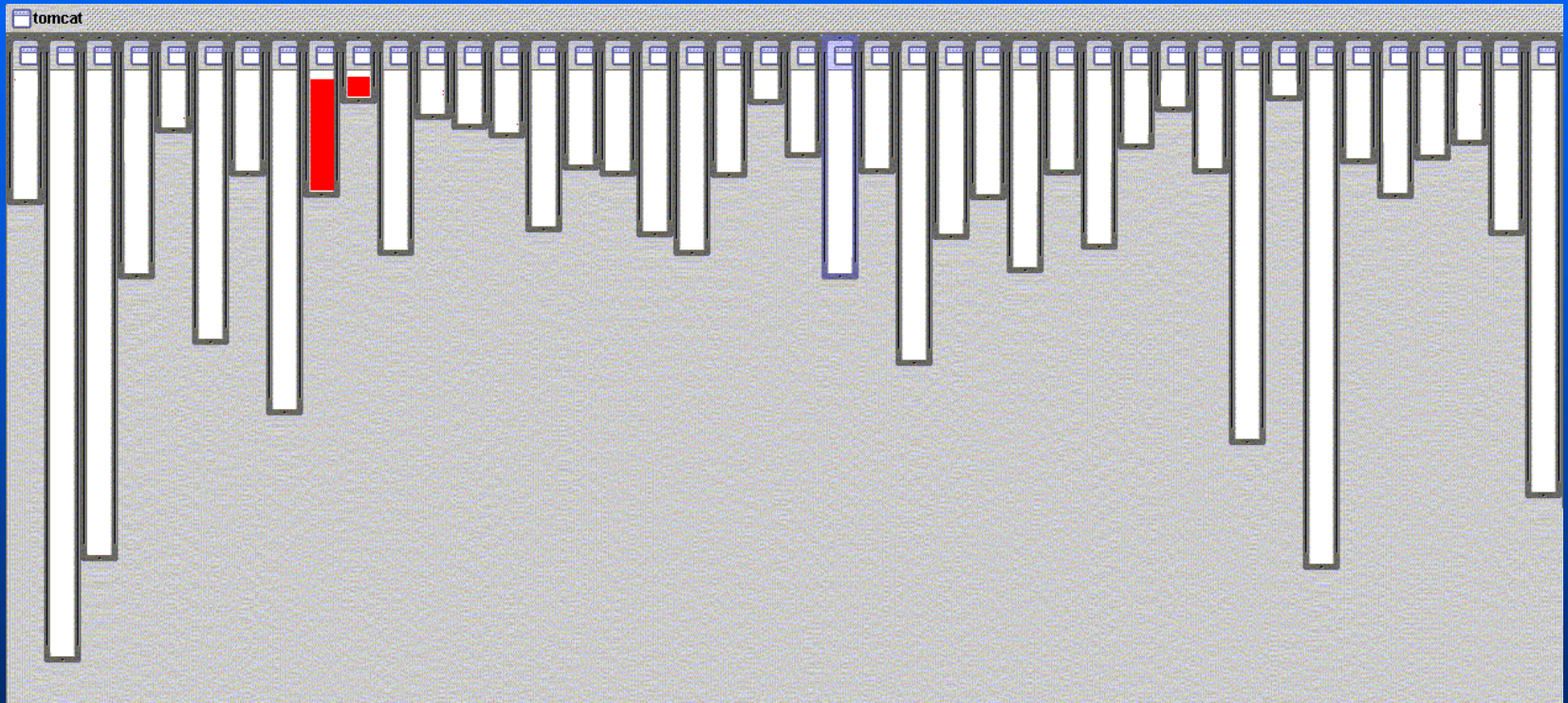
XML parsing



- XML parsing in `org.apache.tomcat`
 - red shows relevant lines of code
 - nicely fits in one box

Example: good modularity

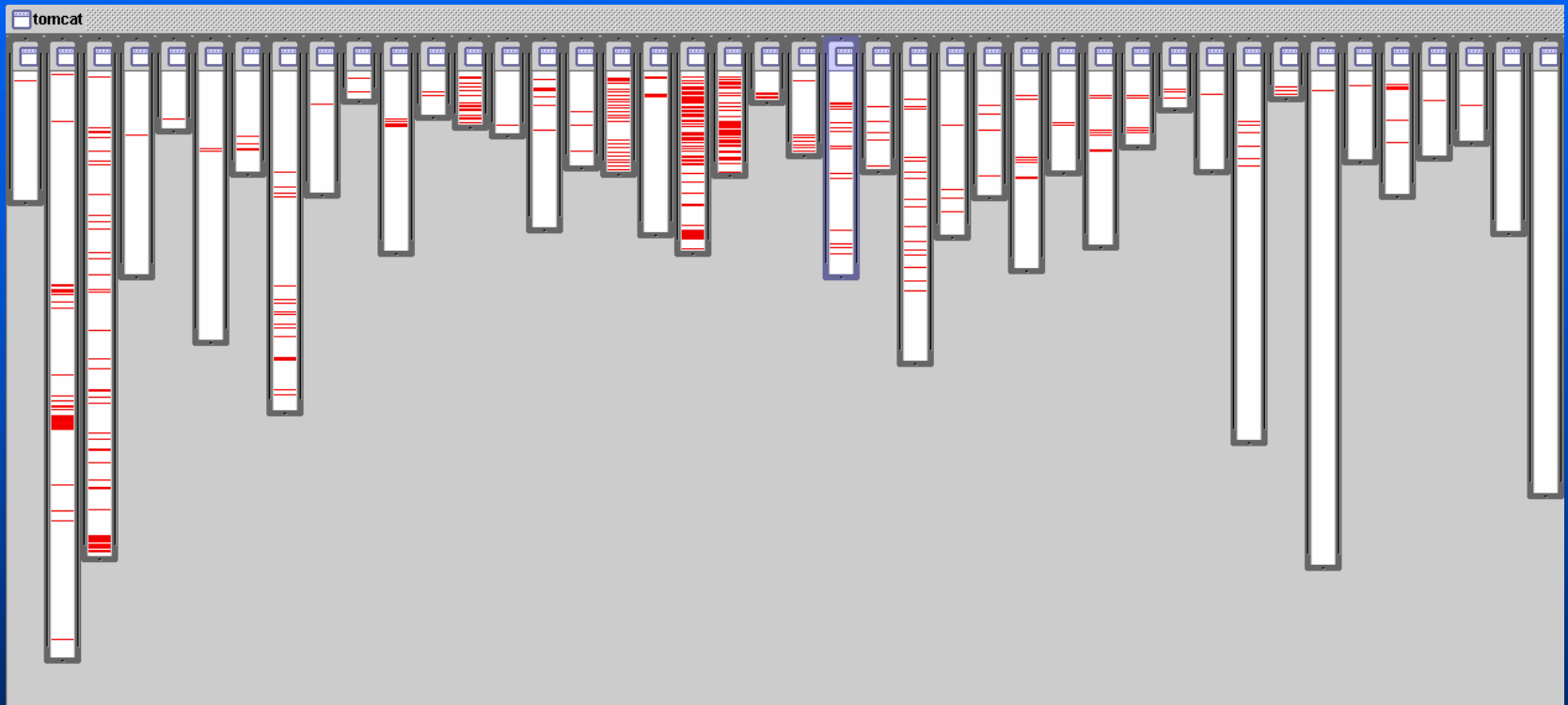
URL pattern matching



- URL pattern matching in `org.apache.tomcat`
 - red shows relevant lines of code
 - nicely fits in two boxes (using inheritance)

problems like...

logging is not modularized



- where is logging in `org.apache.tomcat`
 - red shows lines of code that handle logging
 - not in just one place
 - not even in a small number of places

The World and the Model

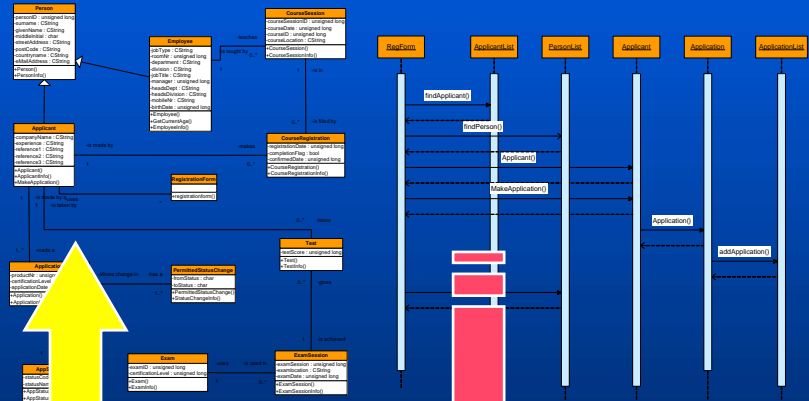
- A Model is a **simplified** representation of an **aspect** of the World for a specific **purpose**
 - Modeling is separating aspects
 - UML paved the way from OOP to Model Based Engineering (MDE)

*Specificity of Engineering:
Model something not yet
existing (in order to build it)*

M_1
(modeling
space)

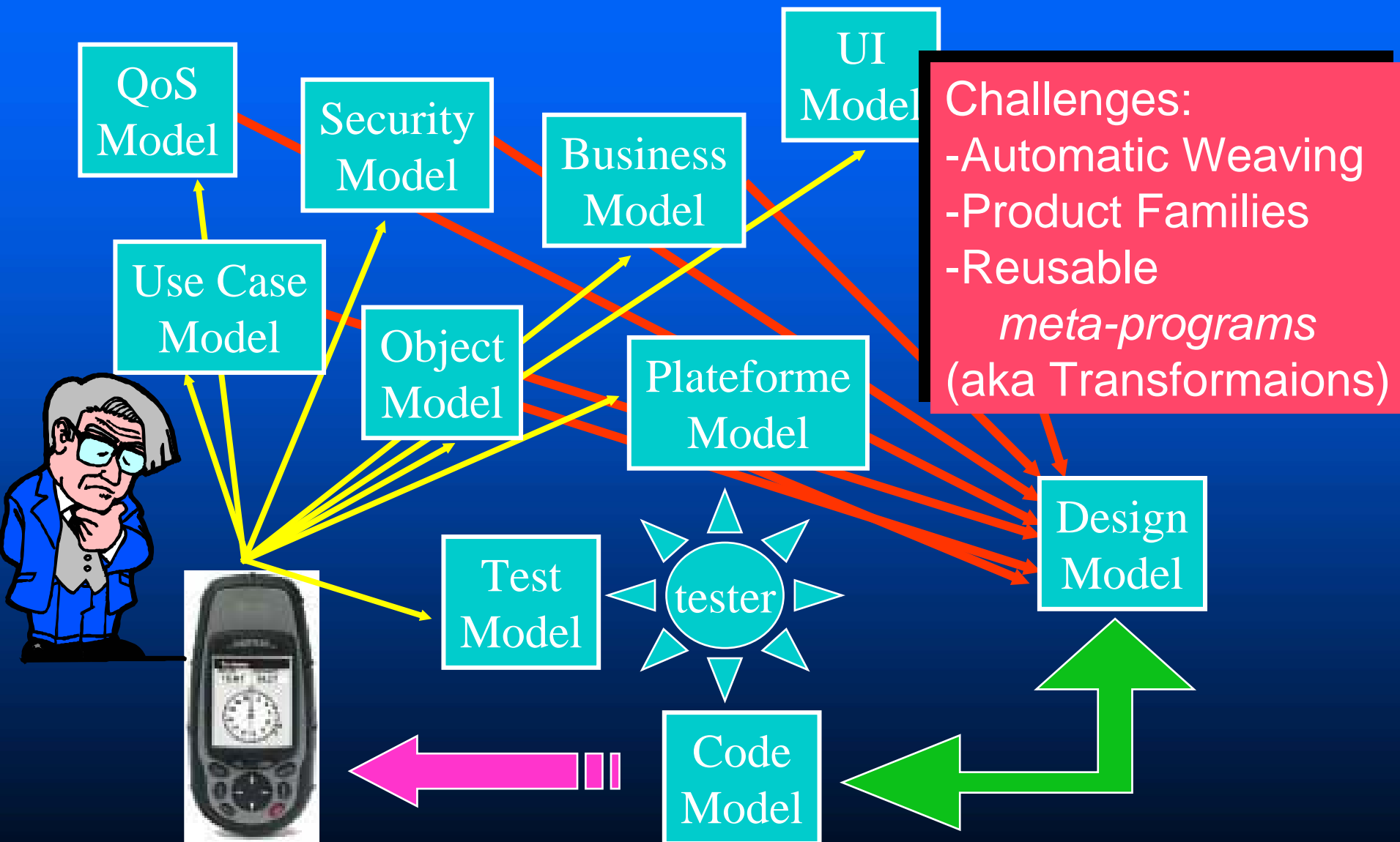
M_0
(the world)

Is represented by



MDE

Modeling and Weaving



Aspects in a Nut Shell

- Aspect Oriented Software Development
 - Avoid the tyranny of a dominant decomposition
 - » Which makes it impossible to modularize some concerns
- AOSD Concepts
 - Modularize these concerns in *Aspects*
 - An aspect defines a set of *join points*
 - *Weavers* to weave aspect logic in the core application
- AOP is a subset of AOSD popularized by AspectJ
 - Kiczales et al., ECOOP'97
 - » MIT's one of 10 key technologies for 2010

Expected benefits of using AOP

- good modularity,
even for crosscutting concerns
 - less tangled code
 - more natural code
 - shorter code
 - easier maintenance and evolution
 - » easier to reason about, debug, change
 - more reusable
 - » library aspects
 - » plug and play aspects when appropriate

Beyond AOP: Open issues in AOSD

■ Theoretical issues

- Aspect composability
 - » Aspects are non commutative, non-associative in the general case
- Semantic point cuts (to avoid syntax dependencies)

■ Software Engineering Issues

- From Requirements to Tests, through Analysis & Design
- Relationship with MDE

■ Technological Issues

- Tool set, efficiency...

■ => AOSD-Europe Network of Excellence

- <http://www.aosd-europe.net/>