

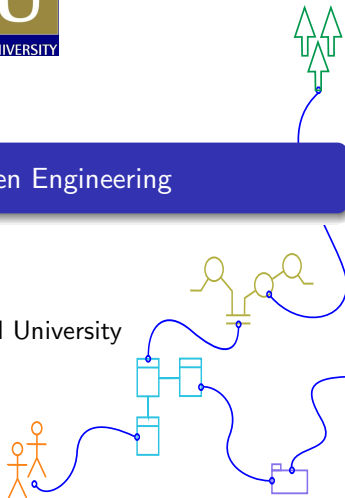


## COMP2510 - Model Driven Engineering

**Shayne Flint**

The Australian National University

October 2009



# Outline

- 1 Background
- 2 Current Approaches to Model-Driven Engineering
  - Conceptual overview
  - Current approaches to MDE
- 3 Current ANU research
  - MDE and Underlying Assumptions
  - The Aspect-Oriented Thinking (AOT) approach
  - Tool Demonstration
- 4 Wrapping up
  - Key points
  - Current AOT Projects

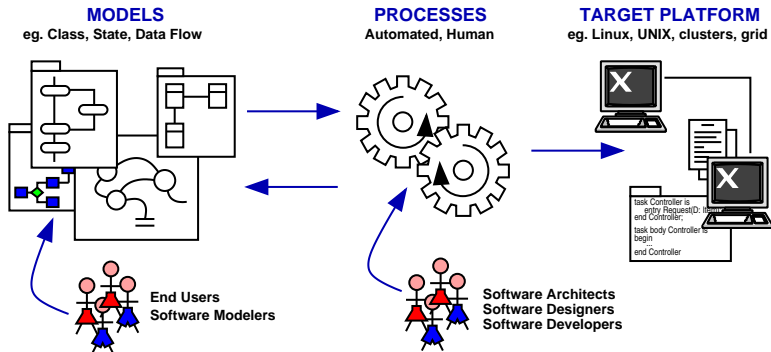
## Why am I interested in MDE?

- **Increasing demands on software**
  - We need to quickly build software systems in response to rapid social, economic and environmental change.
- **Declining or stable student numbers**
  - Demand is increasing
  - Current output is not meeting demand
  - The pipe is empty
- **Plus...**
  - A long standing interest in Specification, Modeling and Automation



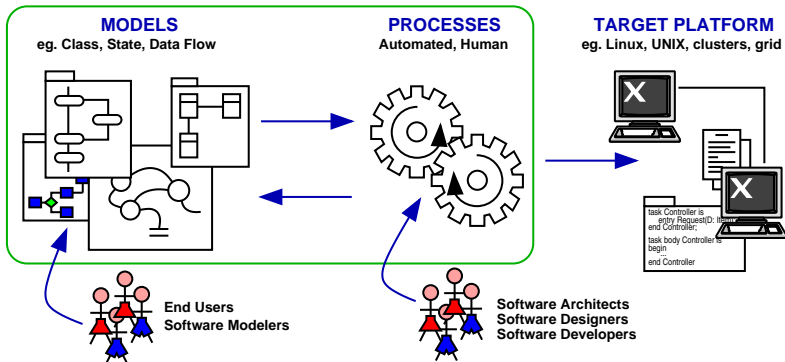
# Current Approaches to MDE

# Conceptual overview - Separation of Concerns

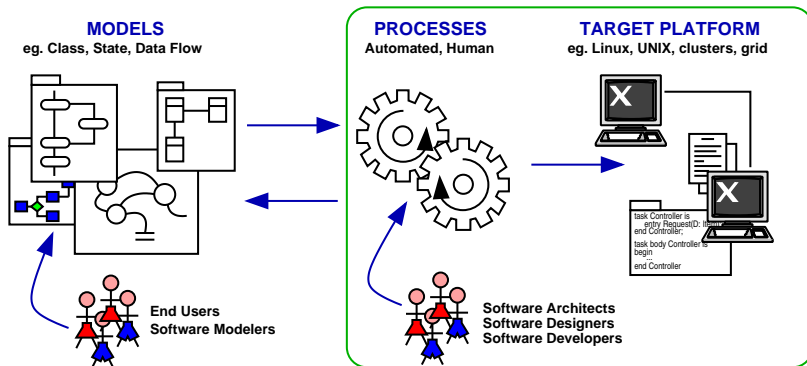


## Conceptual overview - Separation of Concerns

**INTELLECTUAL EFFORT  
IS CAPTURED IN THE MODELS & PROCESSES  
RATHER THAN SOURCE CODE**



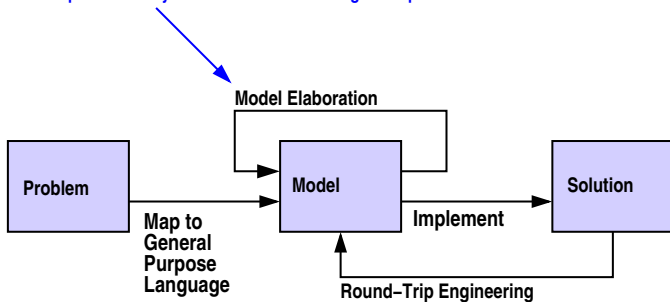
# Conceptual overview - Separation of Concerns



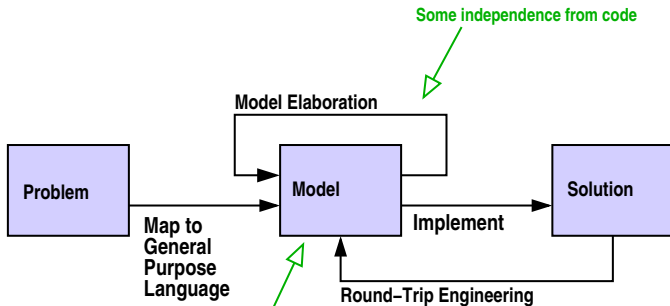
**DIFFERENT PLATFORMS CAN BE  
TARGETED WITH DIFFERENT PROCESSES  
WITHOUT CHANGING THE MODELS**

## Elaborative approach to MDE

Models 'morph' from analysis to architecture to design to implementation



## Elaborative approach to MDE - Advantages

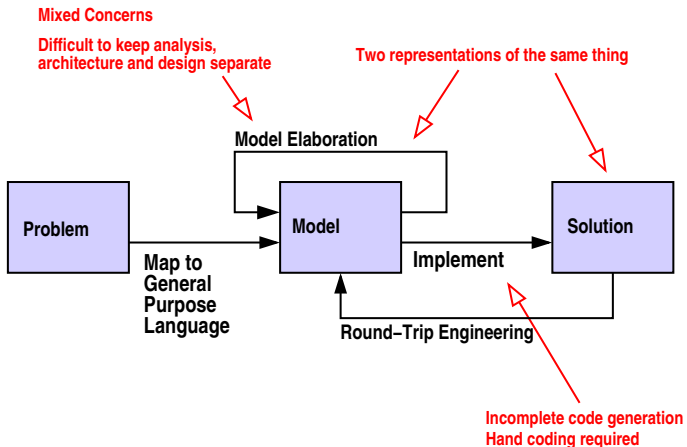


Some independence from code

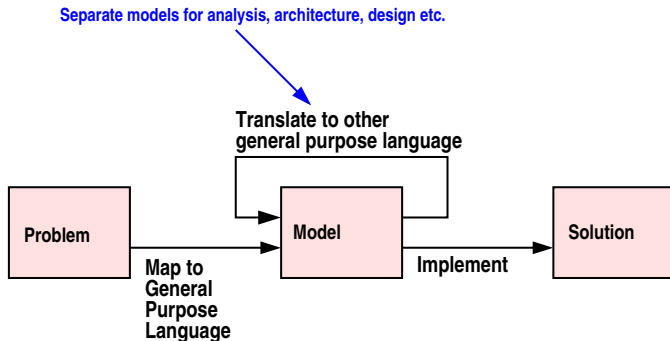
Common modeling language throughout life-cycle

Object-oriented paradigm throughout life-cycle

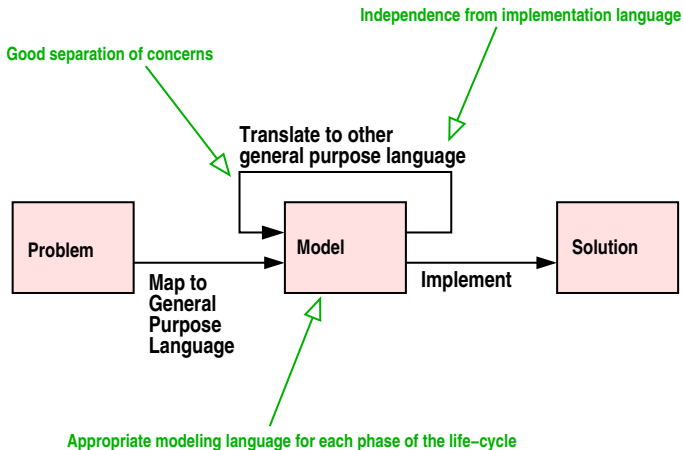
## Elaborative approach to MDE - Limitations



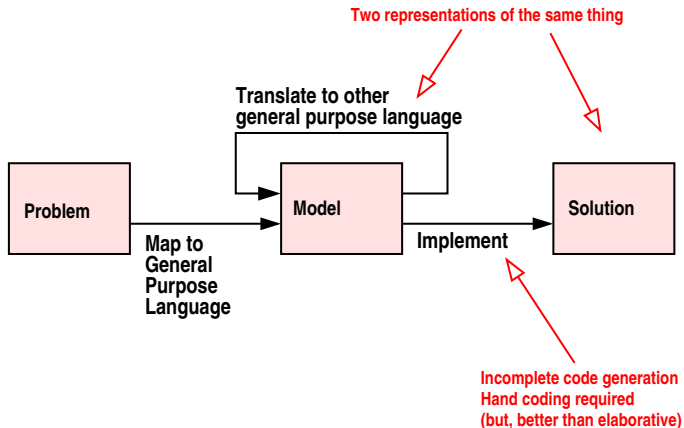
## Translative approach to MDE



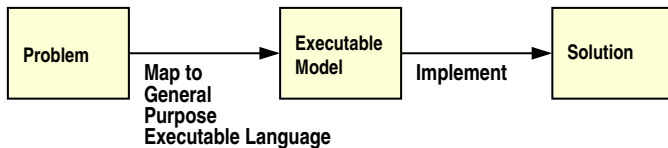
## Translative approach to MDE - Advantages



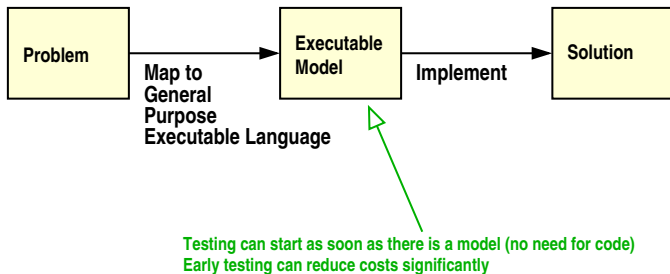
## Translative approach to MDE - Limitations



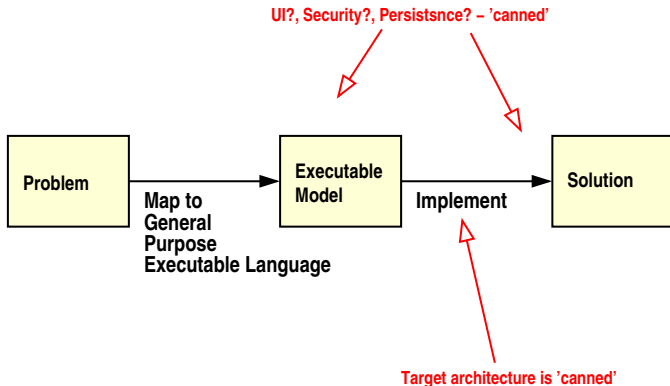
## Executable Modeling approach to MDE



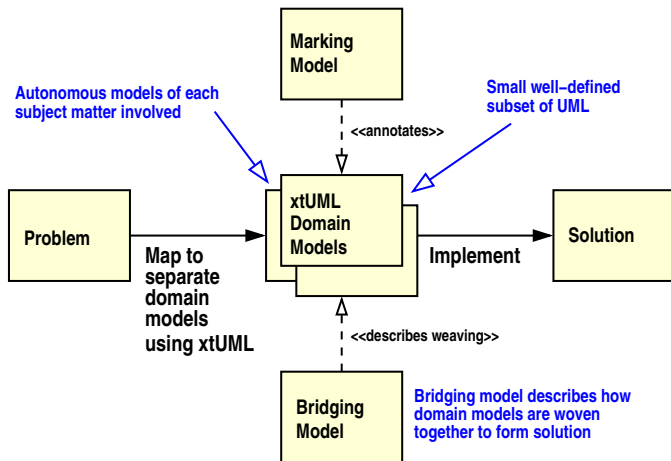
## Executable Modeling approach to MDE - Advantages



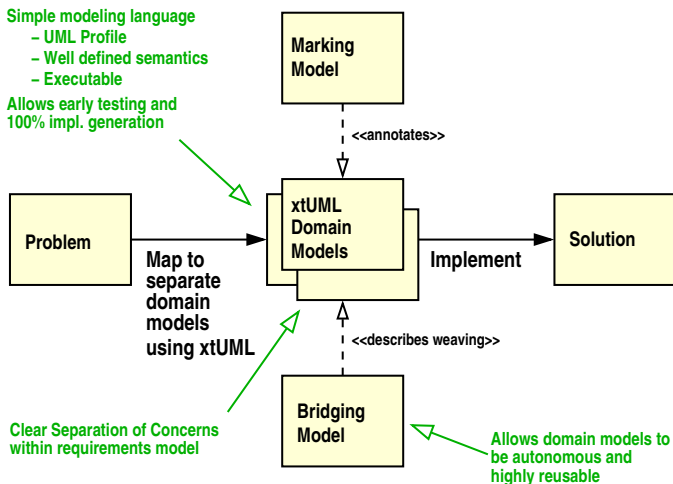
## Executable Modeling approach to MDE - Limitations



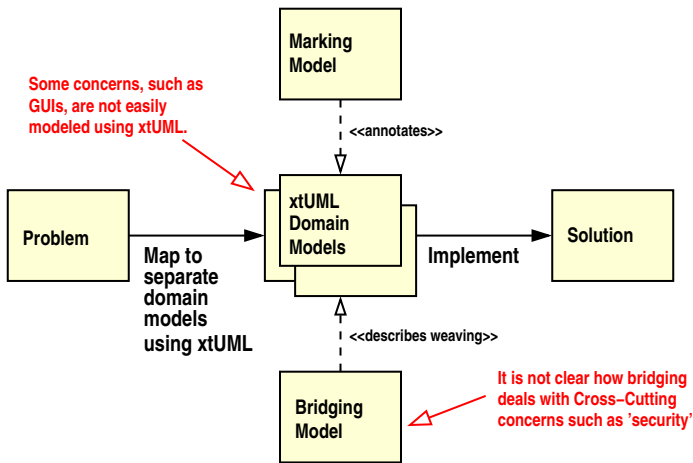
## Executable/Translatable UML approach to MDE



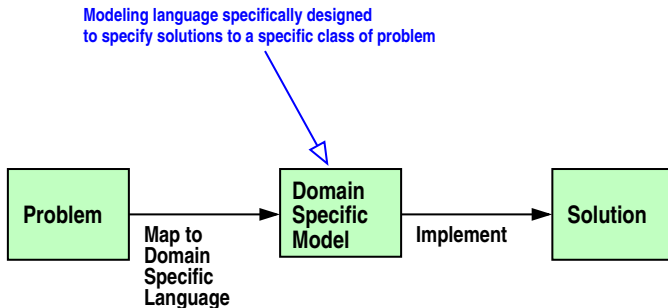
## Executable/Translatable UML approach to MDE - Advantages



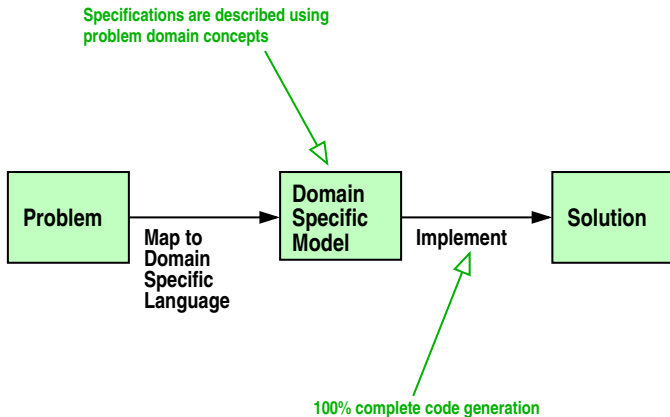
## Executable/Translatable UML approach to MDE - Limitations



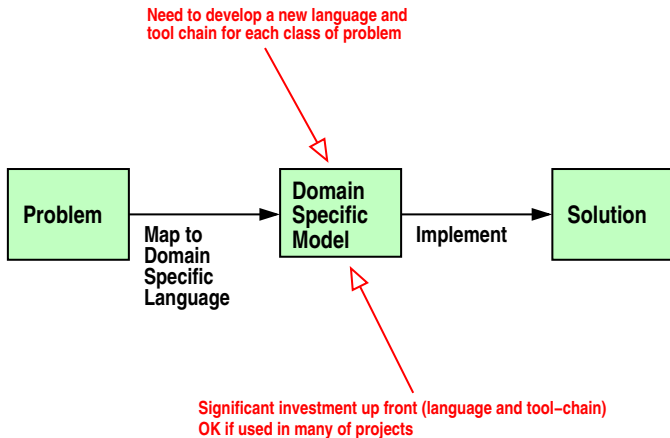
## Domain Specific Modeling (DSM) approach to MDE



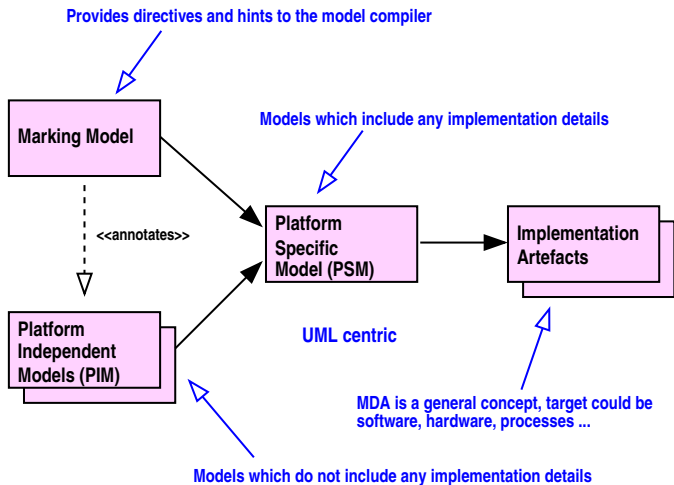
## Domain Specific Modeling (DSM) approach to MDE - Advantages



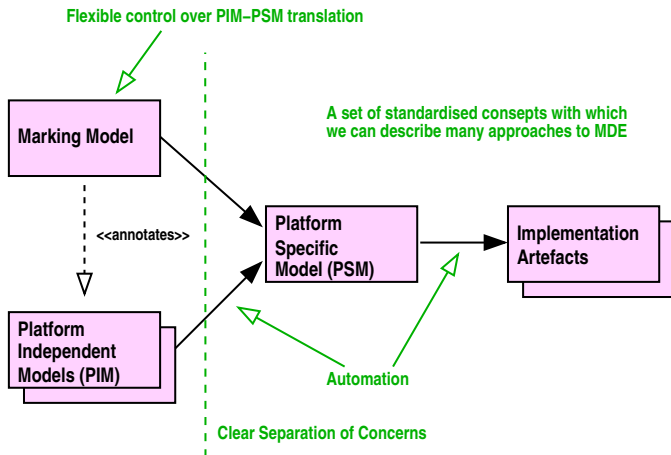
## Domain Specific Modeling (DSM) approach to MDE - Limitations



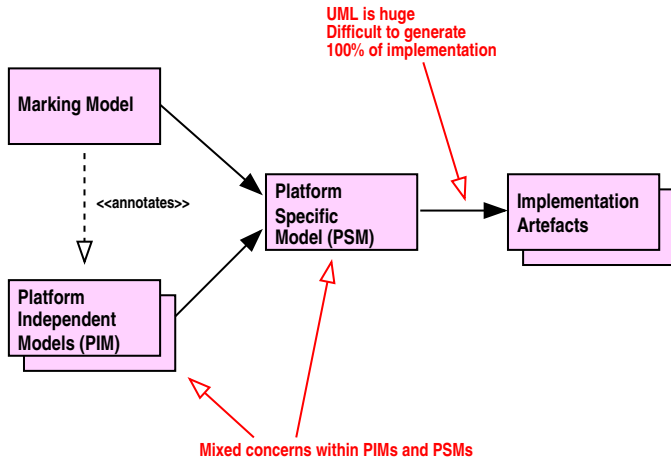
# The OMG Model-Driven Architecture (MDA)



## The OMG Model-Driven Architecture (MDA) - Advantages



## The OMG Model-Driven Architecture (MDA) - Limitations



# Current ANU research

## Underlying Assumptions made by existing approaches to MDE

- **Models are used as Specifications**
  - Models are used to specify part of a specific system
  - Models are translated to form implementation artefacts
- **A 'system' is always built**
  - And it will usually be a 'Software System'
  
- **These assumptions also apply to programming languages**
  - 3GL
  - 4GL
  - Functional
  - Procedural

## Underlying Assumption leads to specific challenges

- **These assumptions lead to a long list of ‘interesting’ research challenges [1][2]**
  - poor industrial uptake
  - requirements and stakeholder management
  - integrating multiple viewpoints
  - cross-cutting concerns
  - modelling languages and paradigms
  - variations in architecture and implementation
  - model semantics
  - translation
  - model synchronisation
  - evolution and reuse
  - the presence of uncertainty, imperfection and ambiguity
  - verification
  - scalability
  - visualisation

[1] R. France and B. Rumpe. *Model-driven Development of Complex Software: A Research Roadmap*. In *Proceedings, Future of Software Engineering, 2007. FOSE 07*, pages 37-54, May 2007.

[2] R. V. D. Straeten, T. Mens, and S. V. Baelen. *Challenges in Model-Driven Software Engineering*. In *Models in Software Engineering*, volume 5421 of *Lecture Notes in Computer Science*, pages 35-47. Springer-Verlag, 2008.

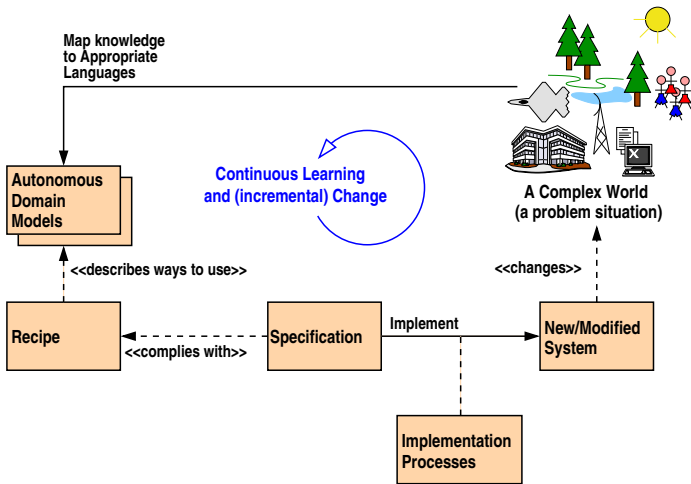
## What if we make a fundamentally different assumption

- **Models as Knowledge**
  - To capture and communicate knowledge about the world
  - To understand the world
  - To identify, understand and evaluate required changes to the world
  - Independent assets in their own right
- **Most disciplines already use models in this way**
  - Disciplinary experts would continue capturing knowledge in models using a wide range of languages and techniques (eg. UML, System Dynamics, differential equations, data flow and even Natural Language)
- **How would this change the way we do MDE?**
  - and how would it impact our list of 'interesting' research challenges?

# Aspect-Oriented Thinking

Bad (historical) name and not to be confused with  
*'Aspect-Oriented Programming'*  
*'Aspect-Oriented Modeling'*  
*'Aspect-Oriented Analysis and Design'*

## Conceptual model



# Conceptual model

- **Problem Situations**

- An environment of interacting software, hardware, people, processes and ideas

- **Domain Models**

- Separate aspects of a problem situation are captured in domain models
- Various languages and approaches
- Autonomous - can be independently developed by disciplinary experts

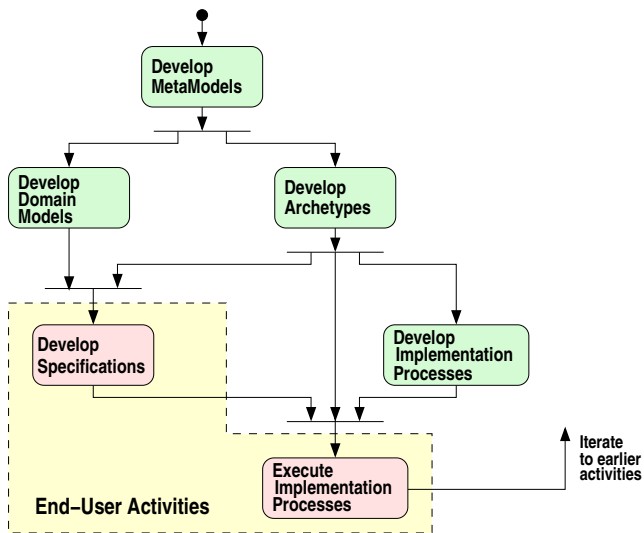
- **Specifications**

- Describe existing and new systems involved in a problem situation
- Comprise a set of **constraints** (requirements) defined in terms of knowledge captured in domain models.

## Conceptual model

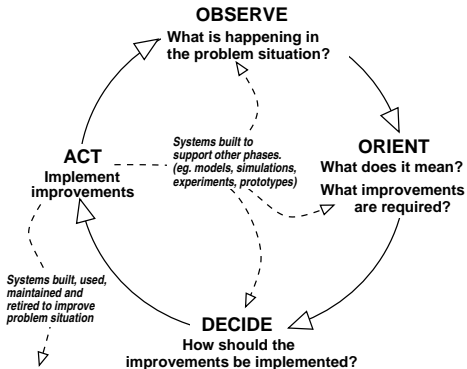
- **Recipes (Specification Archetypes)**
  - Specifications are formed in accordance with **Recipes**
  - These recipes comprise an integrated set of **Constraint Prototypes**
  - Deal with functional, architectural, implementation and other aspects of system development
  - In effect, they describe the ways in which domain model knowledge can be used to specify systems - *patterns, templates, guidelines*
- **Implementation Processes**
  - Automate the generation of systems (source code, documentation etc.)
  - Built against a particular specification archetype
  - Able to process any specification built against such an archetype
- **New/Modified Systems**
  - Deployed into the problem situation in the hope that they change the situation in ways that lead to desirable improvements

## Process model



## Process model

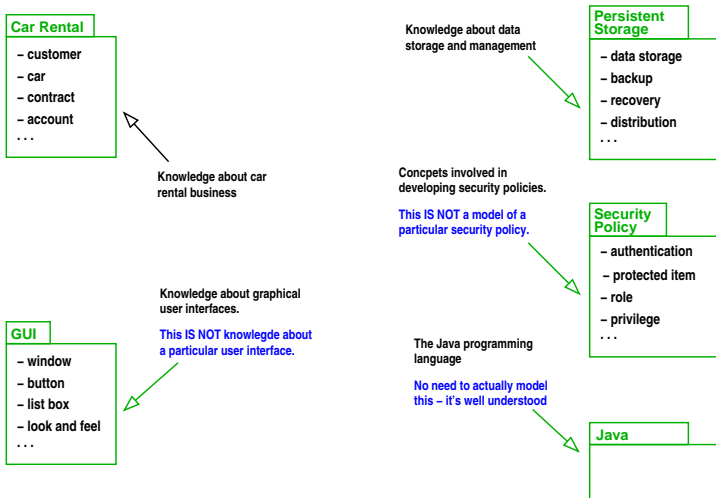
- The approach is applied within a 'Systems Thinking' framework based on Boyd's OODA loop



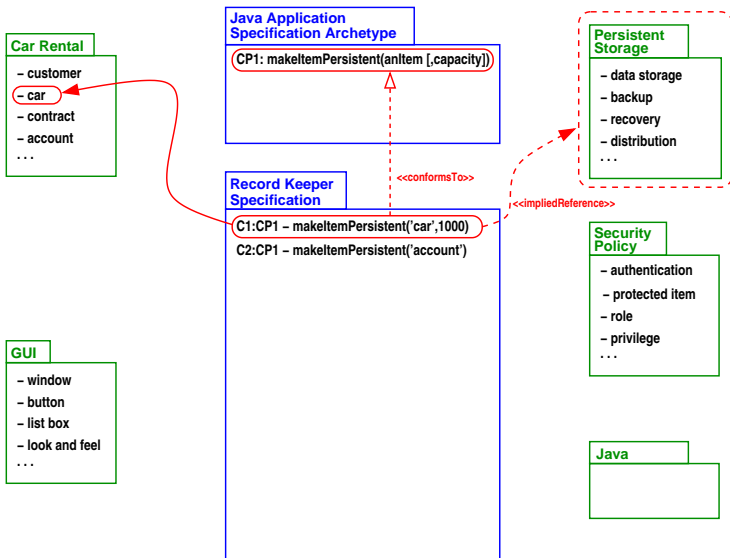
- Not the subject of this lecture

# Car Rental Example

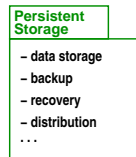
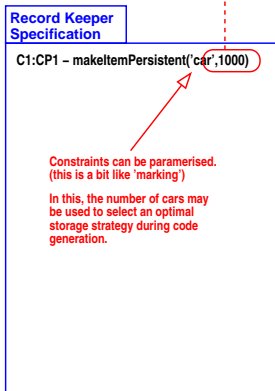
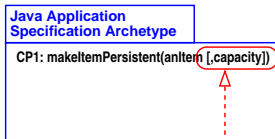
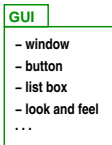
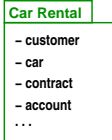
## Car Rental Example



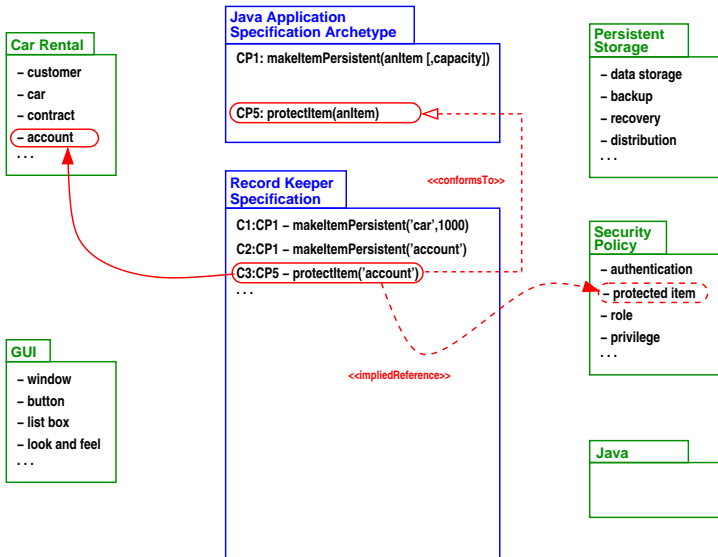
## Car Rental Example - persistence



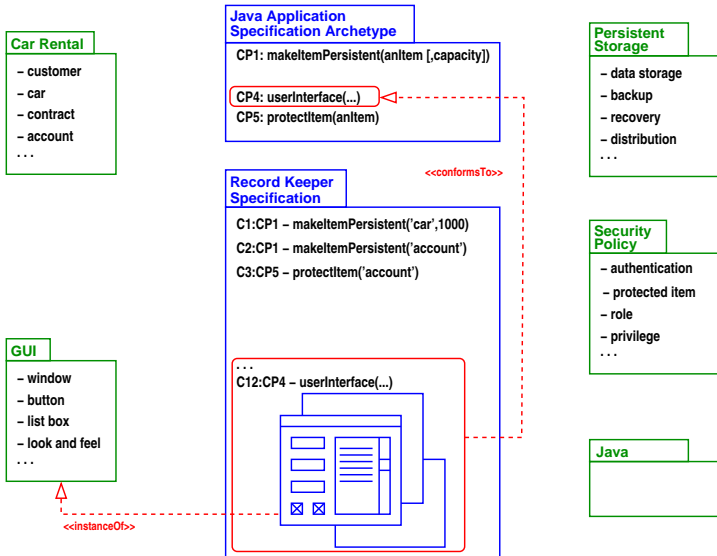
## Car Rental Example - optimisation



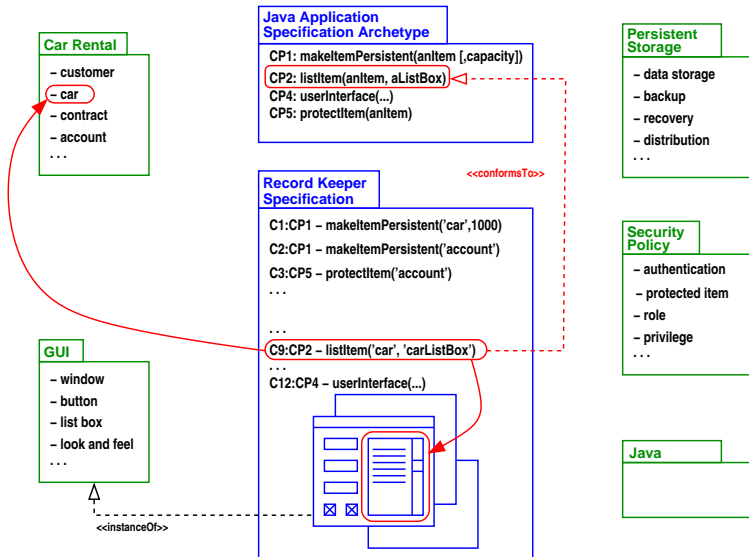
## Car Rental Example - cross-cutting security



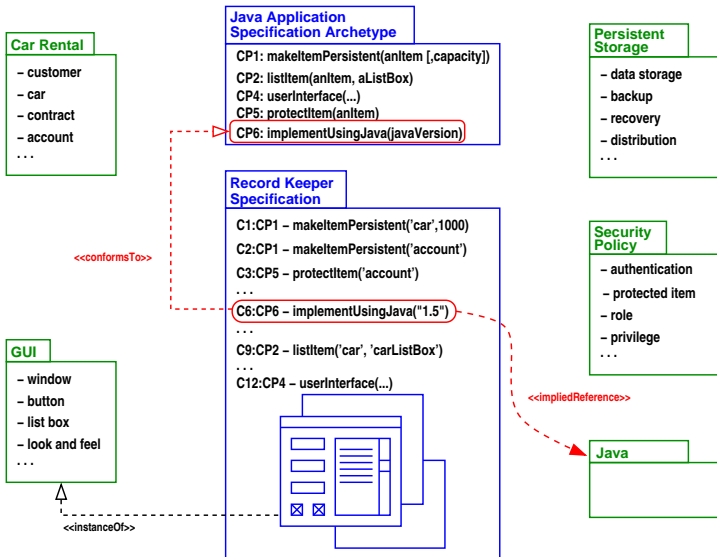
## Car Rental Example - user interface



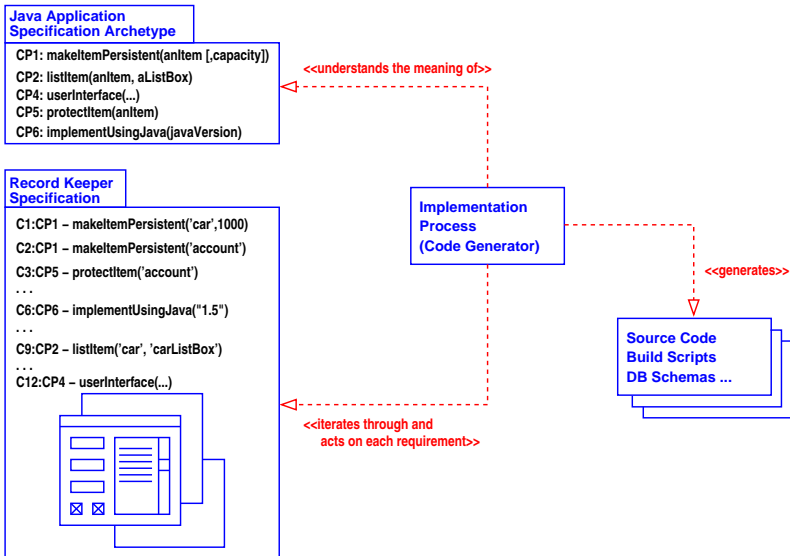
## Car Rental Example - user interface



# Car Rental Example - implementation technology



## Car Rental Example - code generation



# How does AOT address traditional MDE research challenges

## How does AOT address traditional MDE research challenges

- **Industrial uptake**
  - The value of traditional MDE models is limited (because they are specs for a single or small number of systems)
  - ROI on traditional models is low, except in specialised applications
  - AOT models represent highly reusable and valuable knowledge - Intellectual Assets
- **Requirements and stakeholder management**
  - Stakeholder knowledge and views are captured in autonomous **domain models**
  - **Recipes** used to capture stakeholder consensus for a class of system
  - **Specifications** capture stakeholder agreement regarding a specific system
- **Integrating multiple viewpoints**
  - **Recipes** capture the ways in which multiple (stakeholder) viewpoints can be integrated for a specific purpose
- **Cross-cutting concerns**
  - Cross-cutting concerns (eg. Security Policy) are captured in autonomous domain models
  - **Recipes** describe how they can be woven together with other concerns
- **Modelling languages and paradigms**
  - AOT is inherently multi-language and multi-paradigm
  - **Domain Models** can be developed in any language or paradigm
  - **Recipes** describe how such models can be integrated **for a specific purpose**

## How does AOT address traditional MDE research challenges

- **Variations in architecture and implementation**

- **Recipes** often comprise a separate set of requirement prototypes for specifying functional and non-functional requirements, as well as architectural and implementation details.
- Architectural and implementation related prototypes can be swapped out for alternatives

- **Model semantics**

- Reaching agreement as to semantics at the model level is usually difficult due to a lack of context and resulting large numbers of potential stakeholders.
- In AOT, the semantics of a model may change with context - *ie. eyes of the beholder*
- In effect, model semantics are captured in **Recipes** and **Implementation Processes**
- Reaching agreement at the **Recipe** level is much more practical due to a narrower context and fewer stakeholders

- **Translation**

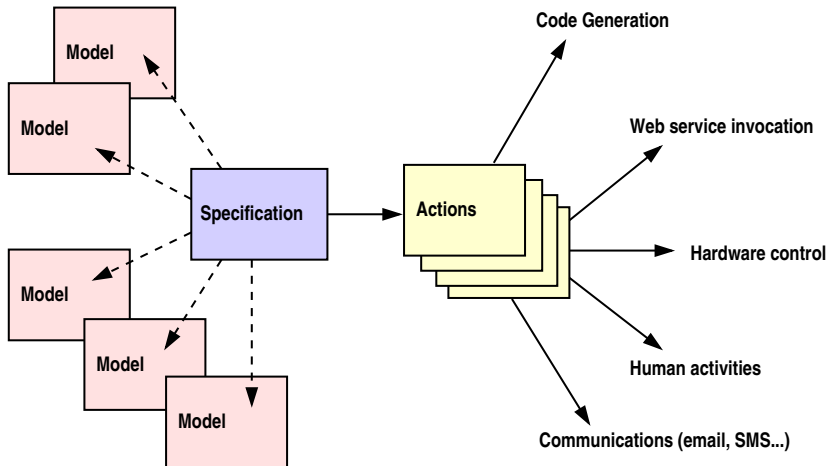
- We do not often translate models.
- Better to think of **executing actions** required to implement **specifications**

- **Synchronisation**

- Only required when **translating** models - we don't do that
- Nonsense in most situations in any case

## How does AOT address traditional MDE research challenges

### Actions vs Model Translation



## How does AOT address traditional MDE research challenges

- **Evolution and reuse**
  - As models evolve, the ways in which they are used evolves
  - That is, the **recipes** are updated.
  - Because they are autonomous and developed as bodies of disciplinary knowledge, **Domain Models** are highly reusable.
- **The presence of uncertainty, imperfection and ambiguity**
  - **Domain Models** only need be good enough for the purposes of a specific recipe
  - Different **recipes** can interpret models in different ways
- **Verification**
  - **Problem moves to Recipes, Specifications and Implementation Processes.**
- **Scalability**
  - The multi-dimensional separation of concerns inherent in the AOT approach should deal with scalability issues
- **Visualisation**
  - **Always a problem**
  - But we have '*a cunning plan*'
  - Hopefully we will explore this in the 3Worlds project

# Tool Demonstration

# Wrapping up

## Key Points

- **Traditional Model-Driven Engineering**

- Models are built to:
  - (sometimes) understand a '*problem*'
  - specify new systems
- Assumes that a system is to be built
- Assumes that requirements are '*out there*' to be collected

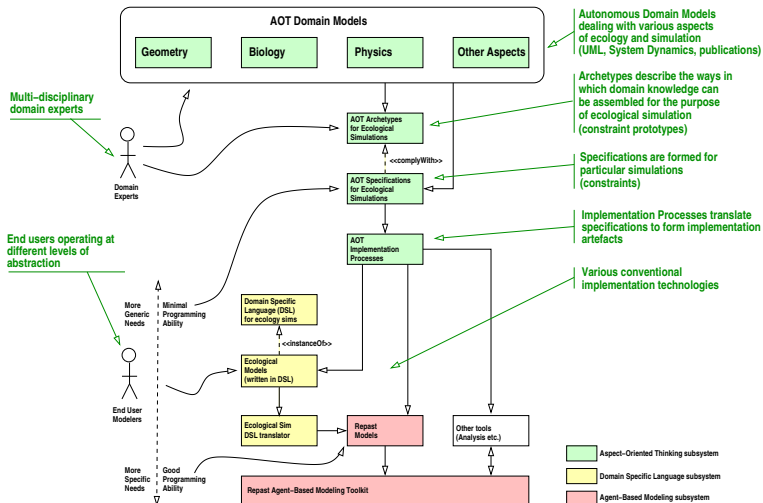
- **Aspect-Oriented Thinking**

- **Models** are built to:
  - capture and communicate knowledge about the world
  - understand the world
  - identify, understand and evaluate required changes to the world
- **Recipes (Specification Archetypes):**
  - capture the ways in which knowledge (models) can be used to describe systems
- **Specifications**
  - formed in accordance with recipes
  - can be followed to produce system implementations by machine, people, or nature (?)
  - models are not necessarily translated

## Current ANU MDE Projects

- **Helping scientists do more science**
  - Collaborators
    - ANU Fenner School of Environment and Society
    - Department of Biology, École Normale Supérieure (ENS), Paris
  - 3Worlds - new generation platform for ecological simulation
  - Re-engineering for Linux clusters
- **(Web) Services Orchestration**
  - With Jack Xiao (MComp)
  - Archetypes for orchestration of autonomous services
  - Similar issues to systems-of-systems engineering
- **Web sites implementing complex business rules**
  - With Lizhi Huang (MComp)
  - Local industry
  - ARC Linkage Pilot
- **AOT Tool Support**
  - Multi-Language support (user defined)
  - Automated Implementation Process support
  - Demonstrated in software development domain
  - 2<sup>nd</sup> generation web-based tool under development

# An ecological modeling system





## COMP2510 - Model Driven Engineering

Shayne Flint

**Software-Intensive Systems Engineering Group**  
College of Engineering and Computer Science  
The Australian National University

Tel: +61-(0)2-6125-8183  
shayne.flint@anu.edu.au

