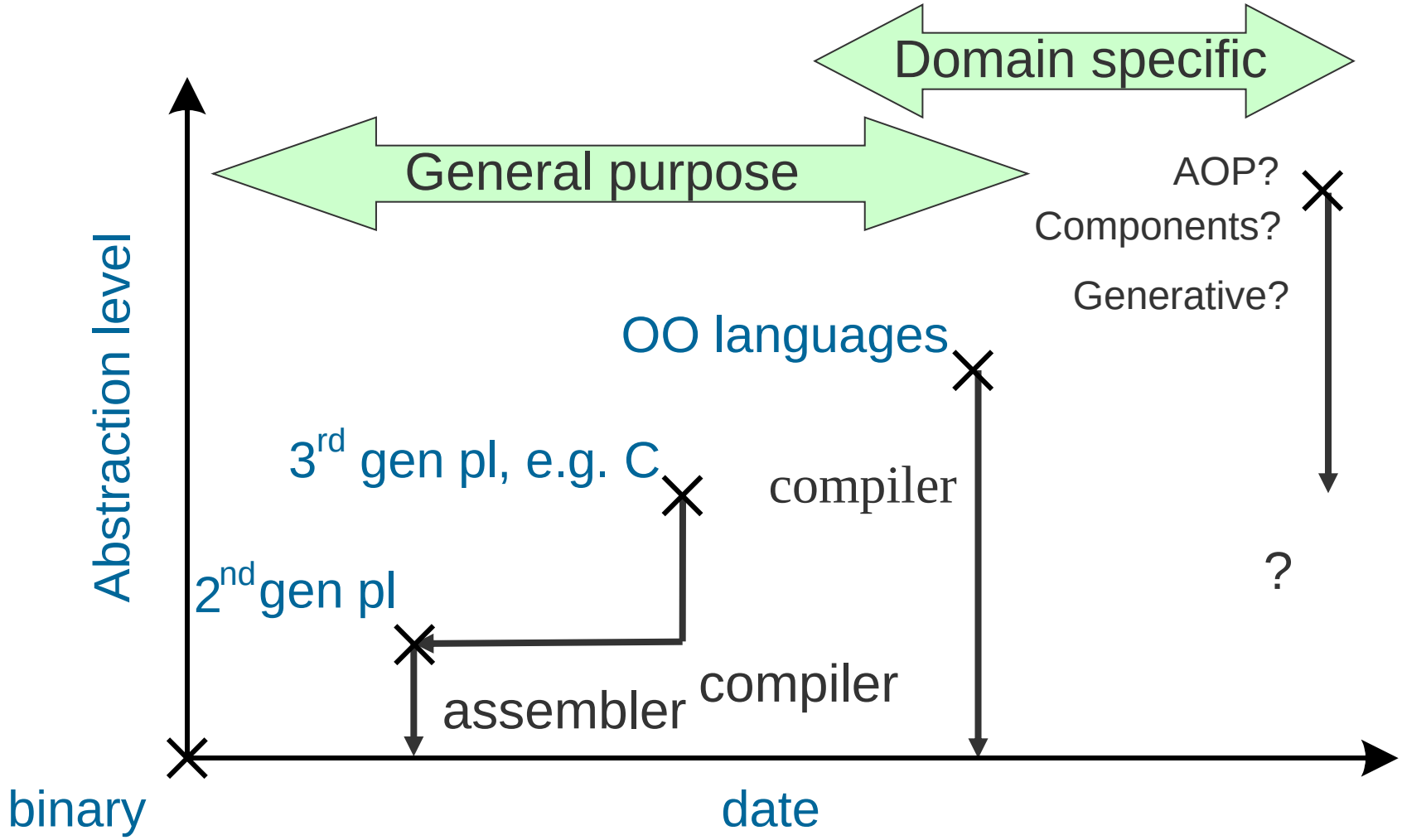
MANCHESTER 1824
The University of Manchester

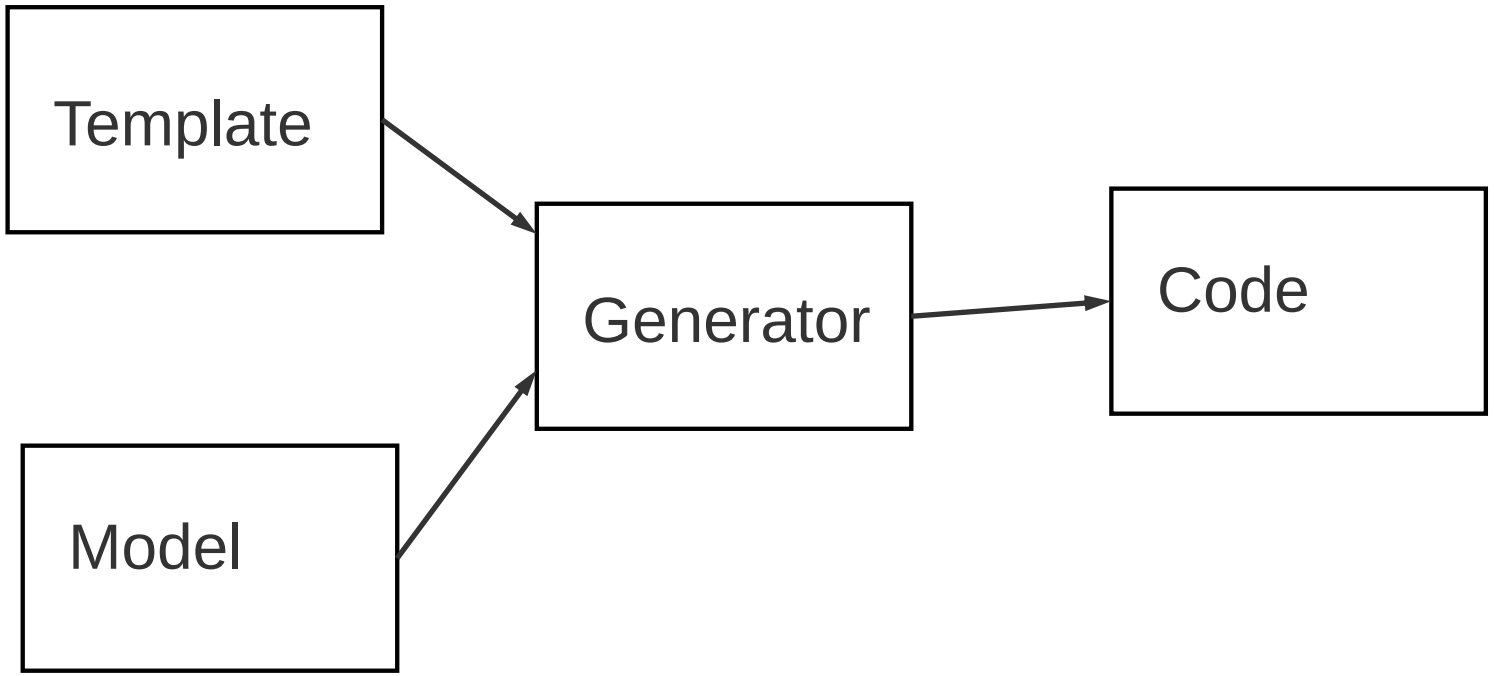
Generative Programming

Andy Carpenter
School of Computer Science
(Andy.Carpenter@manchester.ac.uk)

Abstraction in Software Development



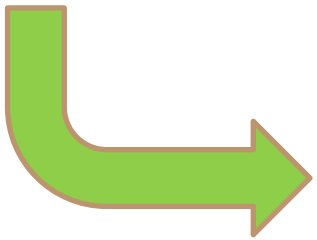
Principle



SASS

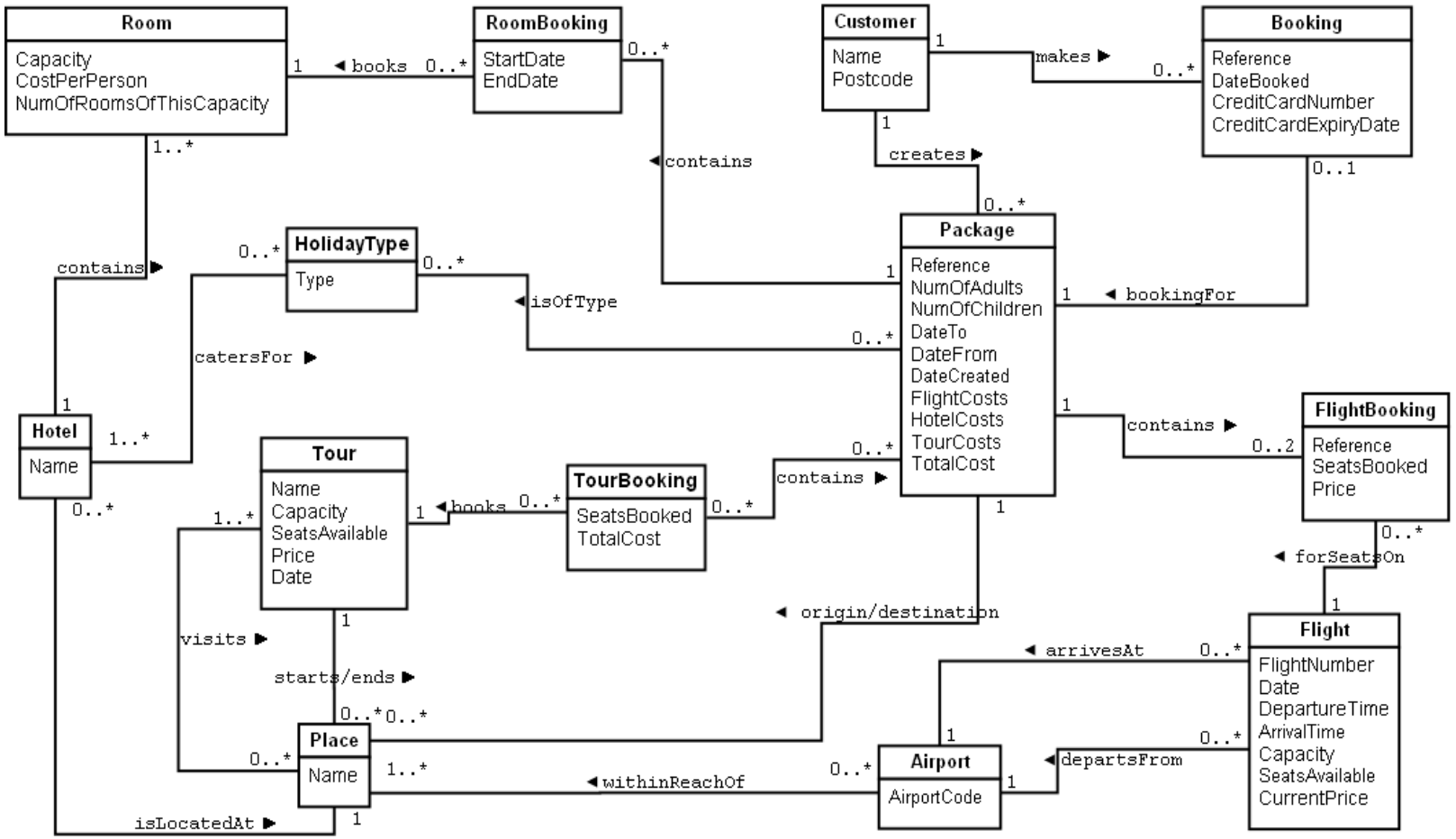
```
$font-stack: Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}
```



```
body {
  font: 100% Helvetica, sans-serif;
  color: $primary-color;
}
```

Model Example

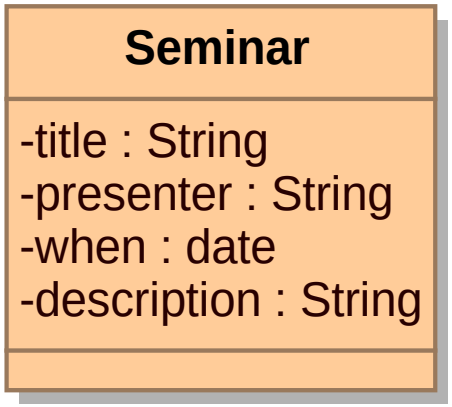


MANCHESTER 1824
The University of Manchester

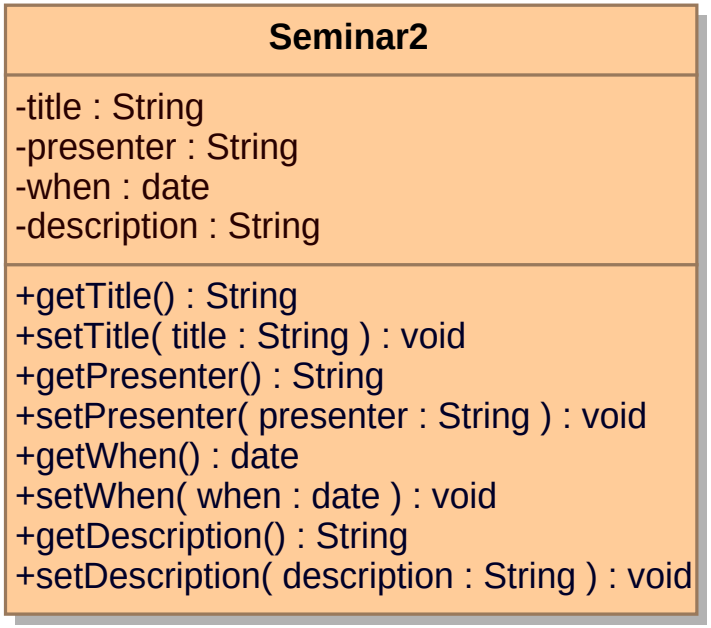
Implementation of Class Diagram

Types of Implementations

PIM



PSM



MANCHESTER 1824
The University of Manchester

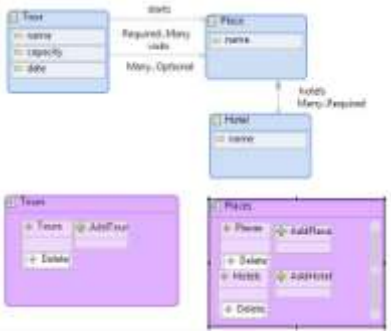
Implementing Design Patterns

Generate Alternatives

Entities

Associations

Pages Units



JSF/Facelets

JPA

Validation

Caching

Spring

Resource Bundles

PHP

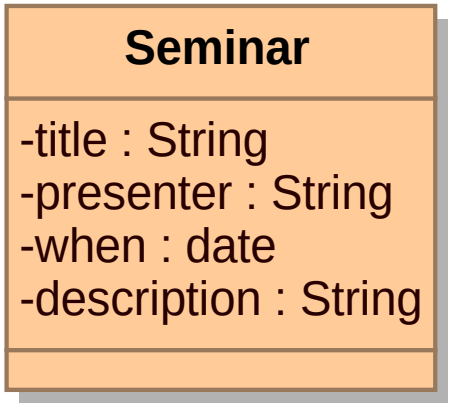
CakePHP

Validation

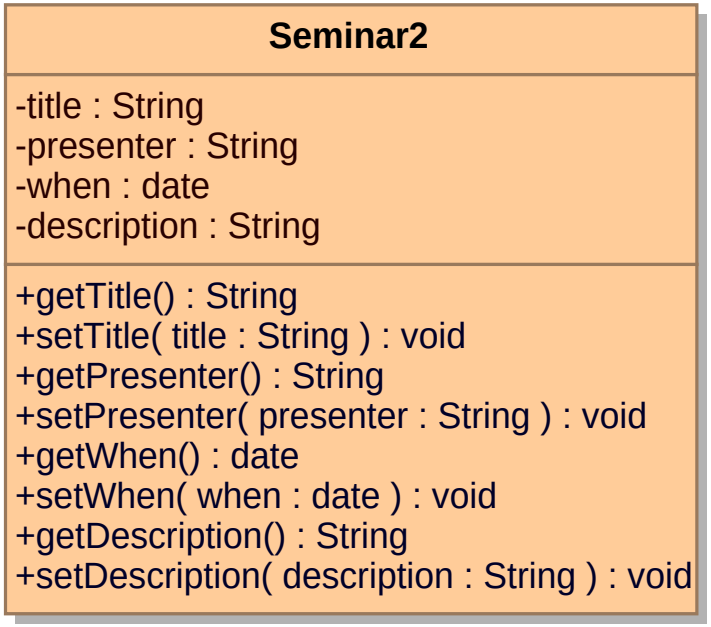
Types of Models

- Platform Independent Models (PIMs)
  - no implementation details
- Platform Specific Models (PSMs)
  - have implementation details
- Transformations to convert PIMs into PSMs

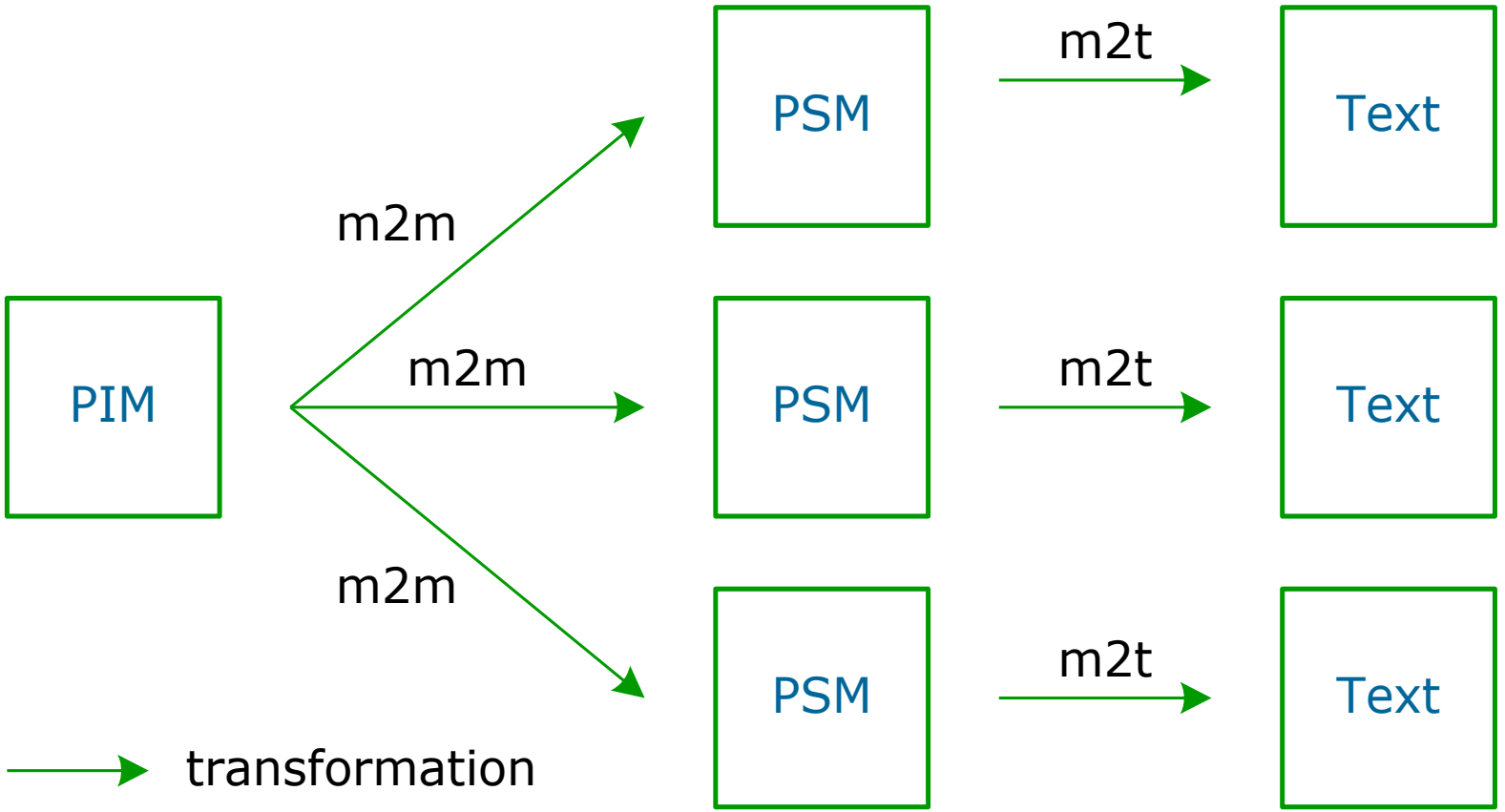
PIM



PSM

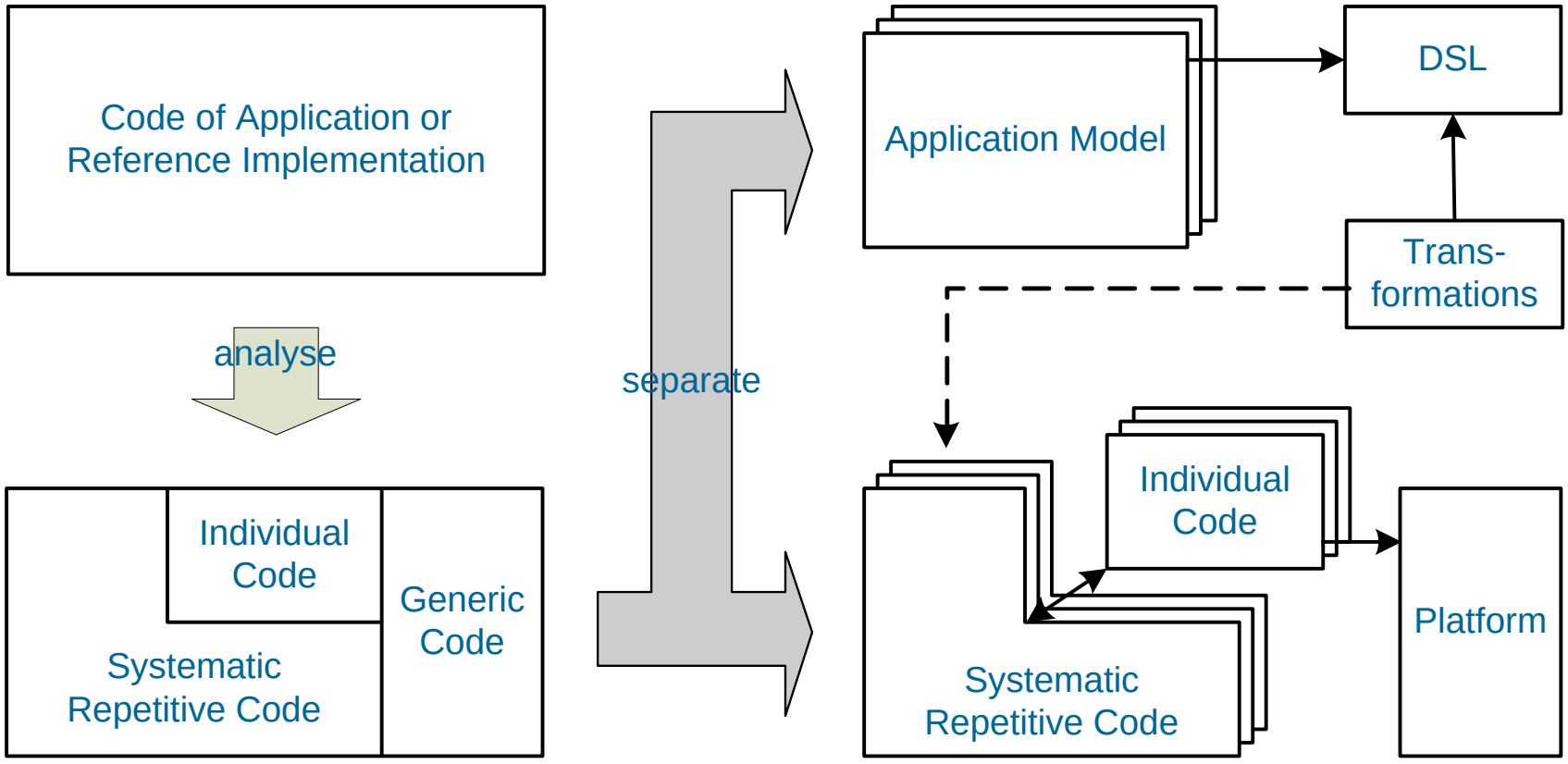


MDSD Development Flow



Model 2 text (m2t)
Model 2 model (m2m)
Text 2 model (m2t)

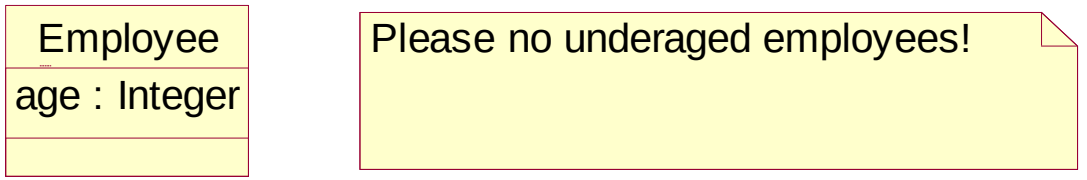
Use of Models



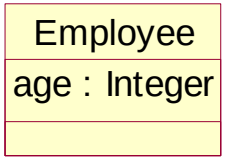
MDSD Figure 2.1

Validation

- Annotations in natural language:
  - ambiguous, imprecise,



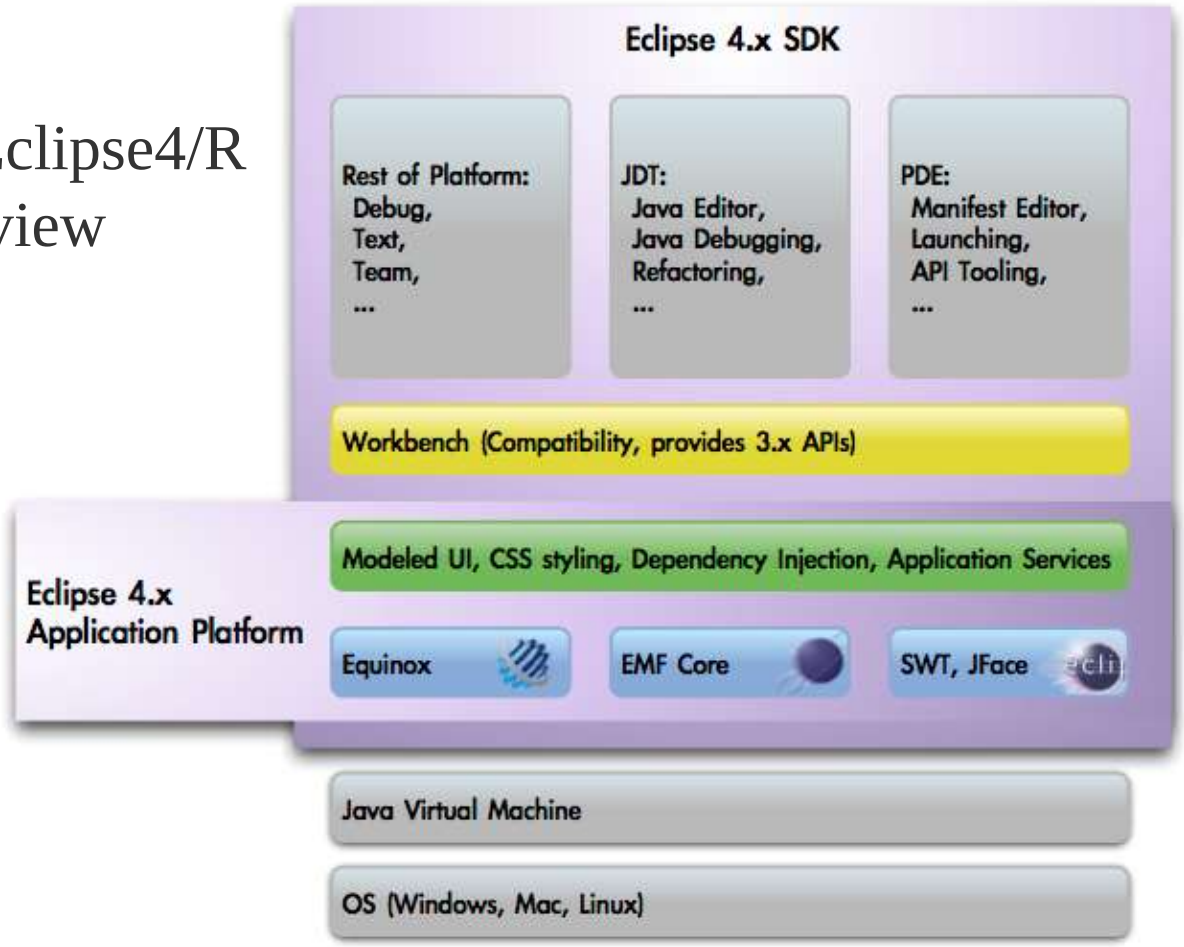
  - not automatically checkable
- Traditional formal languages, e.g. Z:
  - require good understanding of mathematics
  - do not scale to large systems
- Object Constraint Language (OCL):
  - formal, precise, unambiguous
  - tool support is available



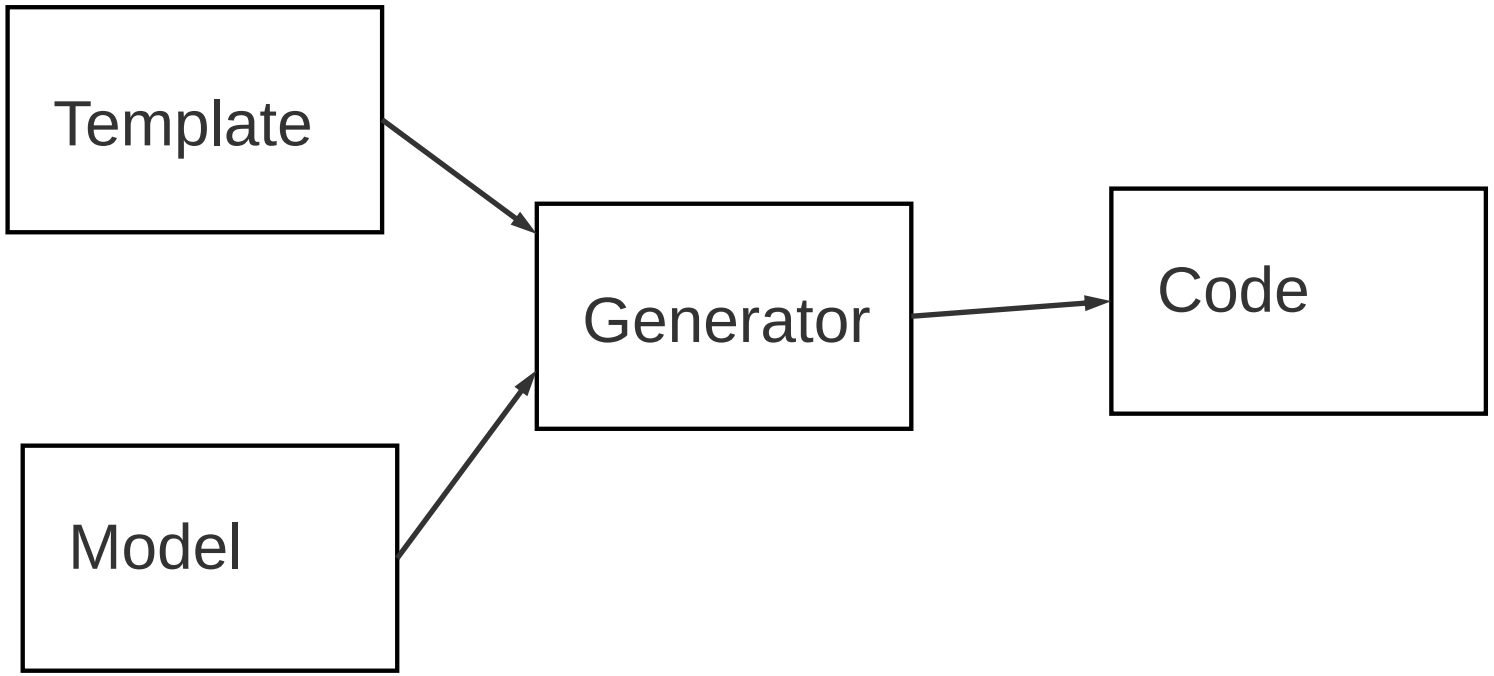

```
context Employee inv:
self.age > 18
```

Eclipse 4

http://wiki.eclipse.org/Eclipse4/RCP/Architectural_Overview



Principle

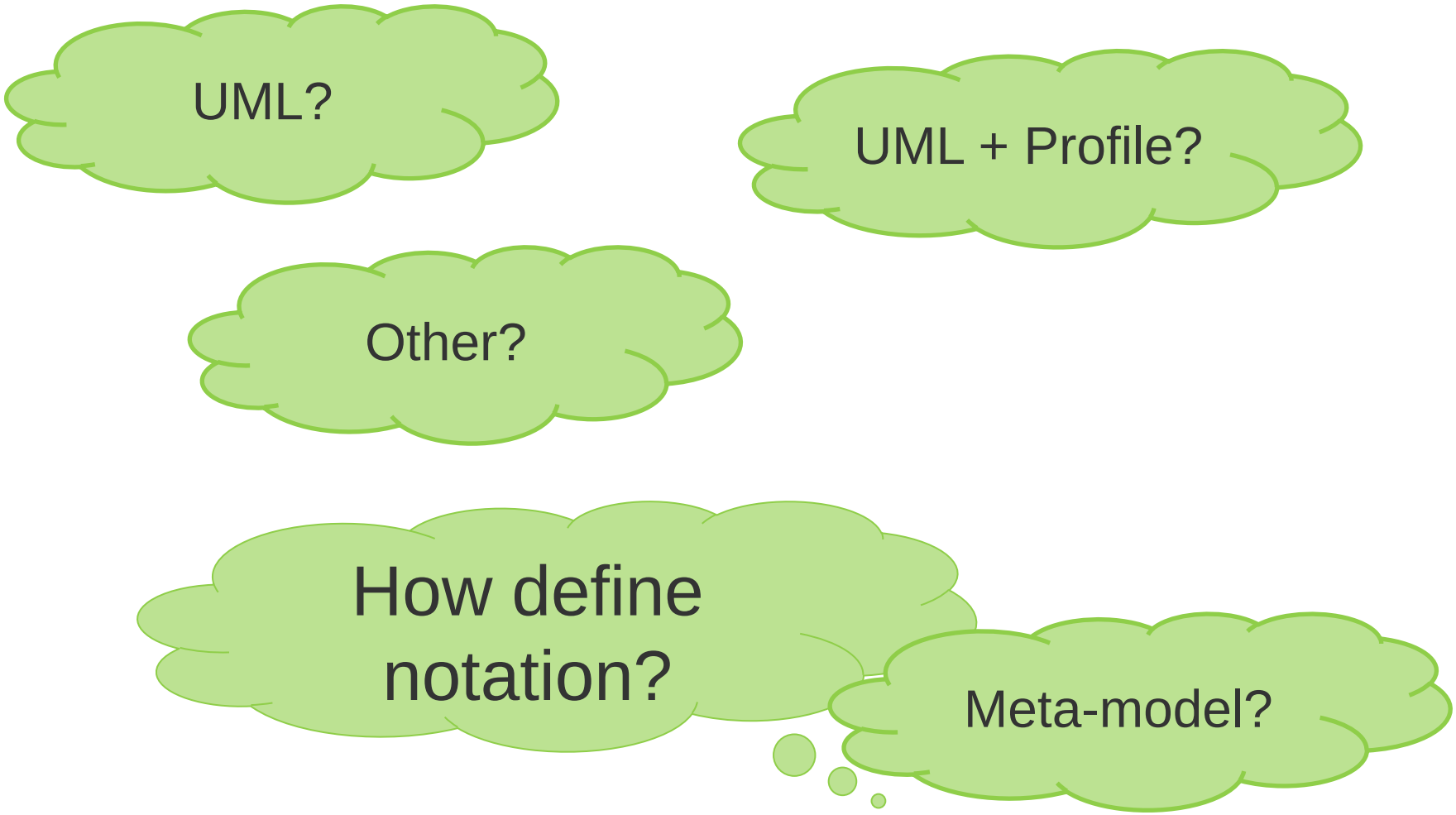


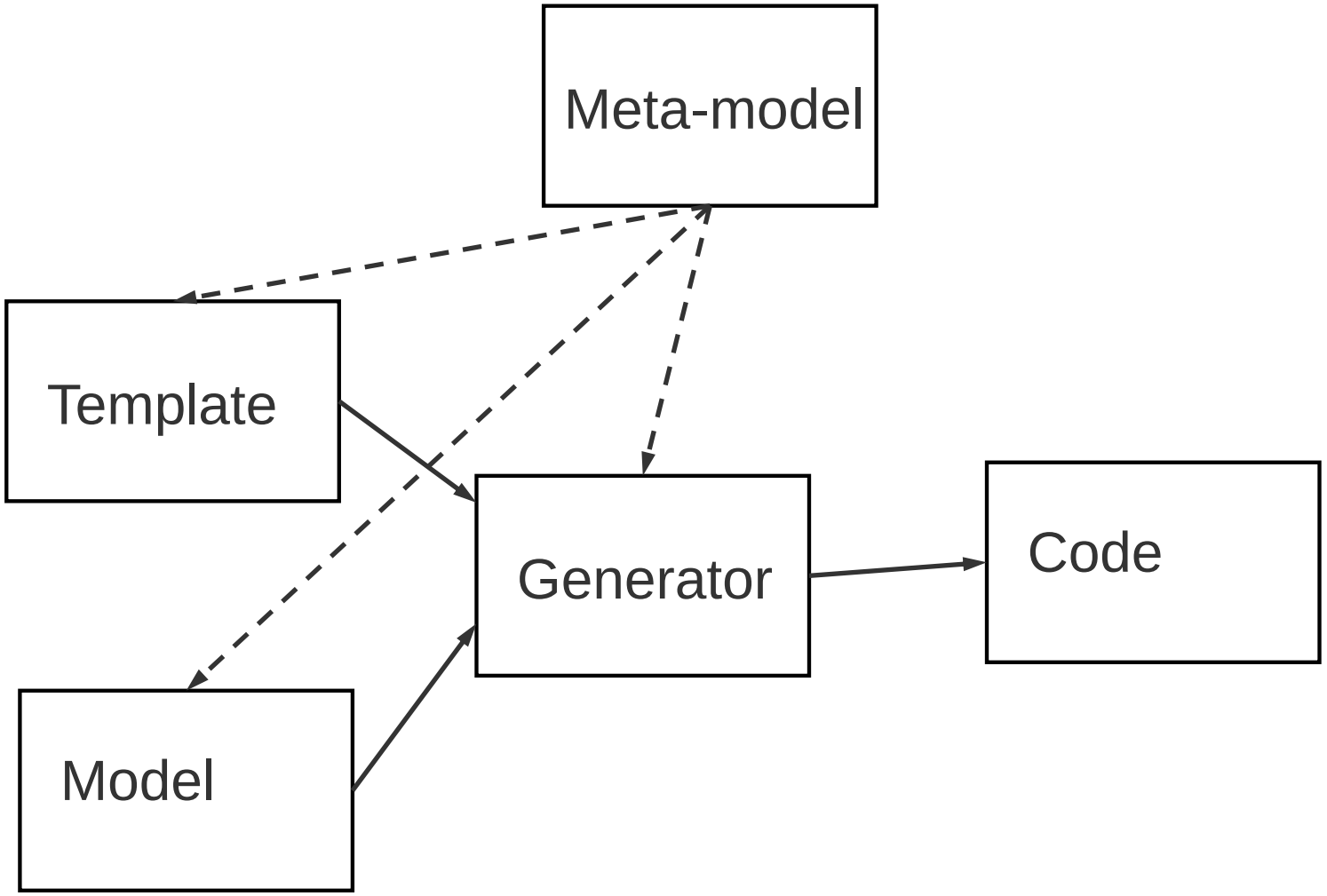
MANCHESTER 1824
The University of Manchester

What is needed?

- Modelling paradigm (DSL)
  - Modelling language with elements that have defined semantics; e.g. class in UML class diagram is container with attributes that have type
- Target platform
  - Programming language, software libraries,
  - Hardware
- Generator for transforming model to target
  - able to read and understand model
  - able to read and process template

Model Notation




Meta-model
Template
Generator
Code
Model

Eclipse Modeling Tools

Model-Driven Software Development

Model-Driven Engineering

Model-Driven Architecture

lopment

Software Fac

About using models to simplify software development

gn

Model-Driven

Model-Driven Software Development

Model-Driven Development

Everything is a Model

Ecore model

UML model

ER model

Source code

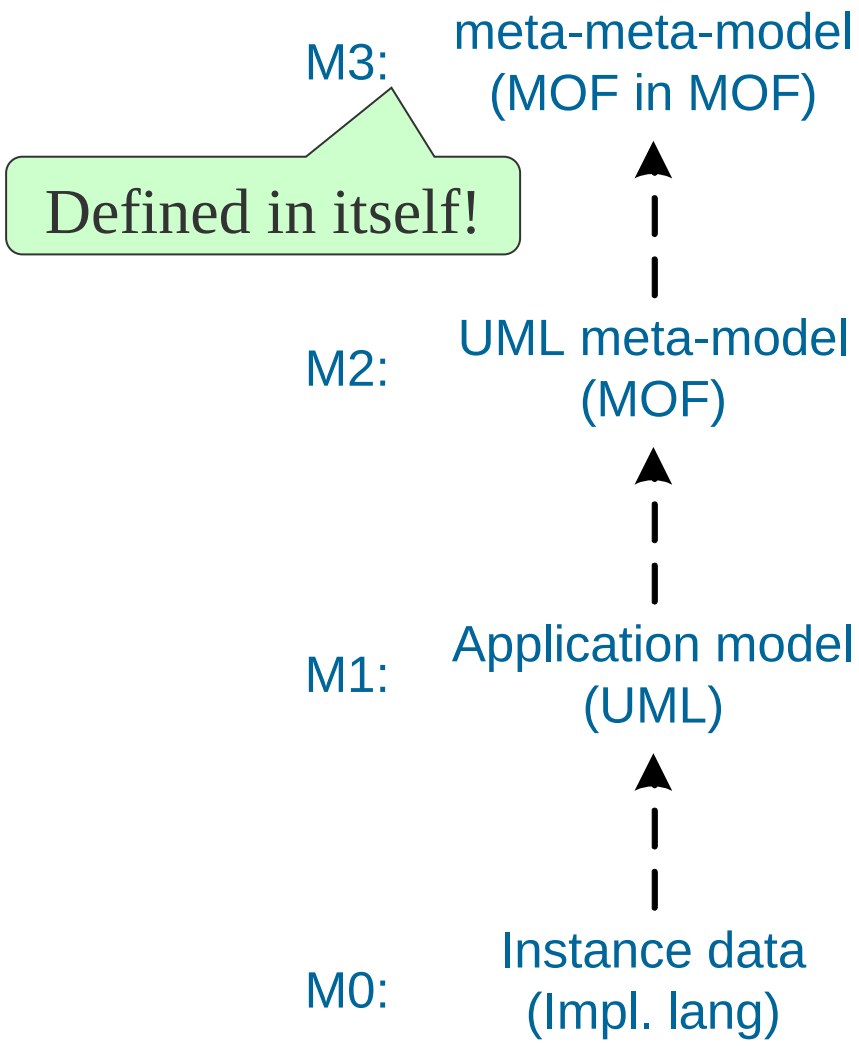
XML schema

Application instance data

XML

MDA Modelling Levels

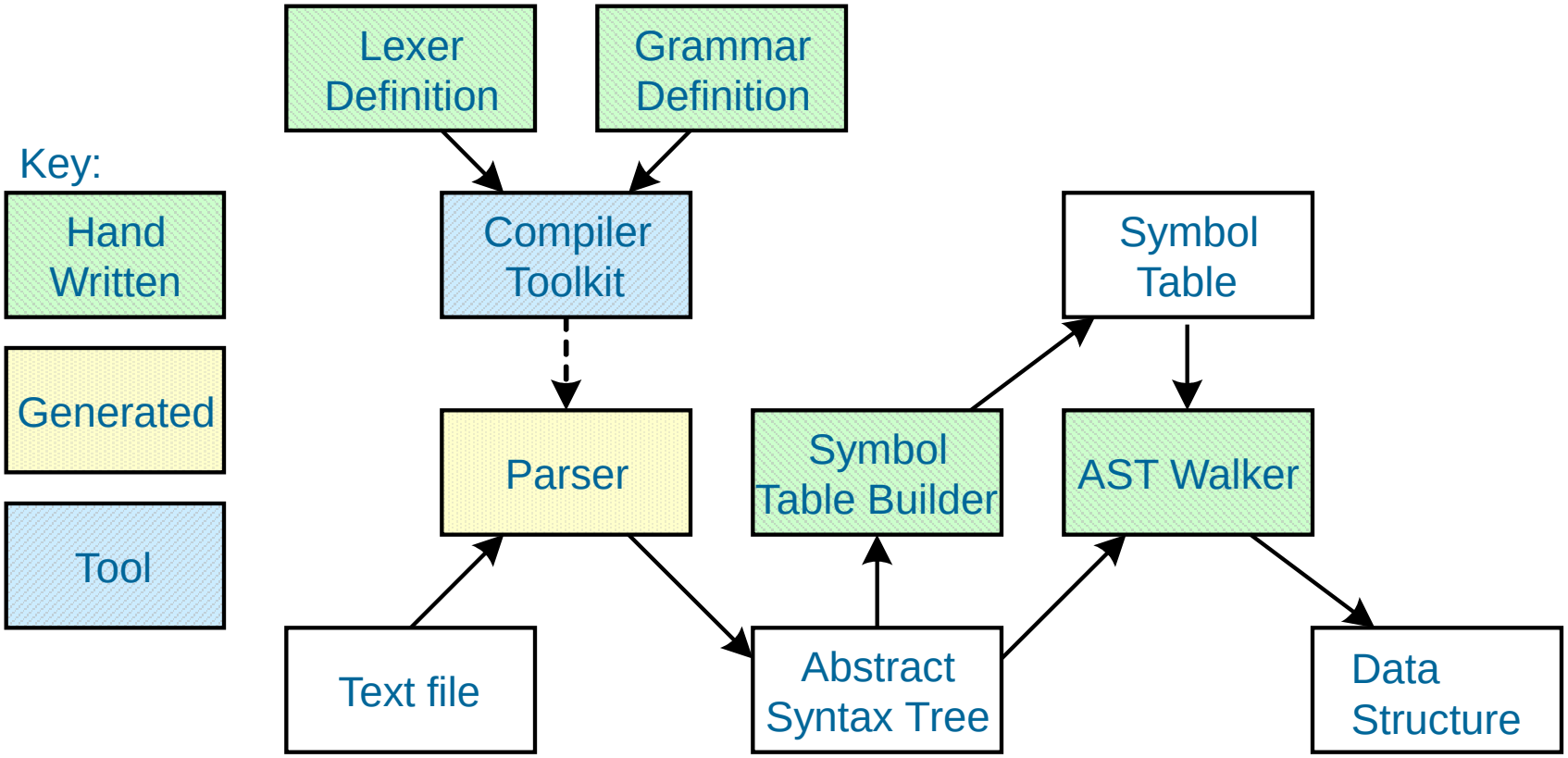
- M1: model of application
  - e.g. UML model, definition of Java class
- M2: model of modelling notation (meta-model)
- M3: model of meta-model (meta-meta-model)
- M0: application instance data
- Say model conforms to (is instance of) meta-model



A Meta-Model Defines a DSL

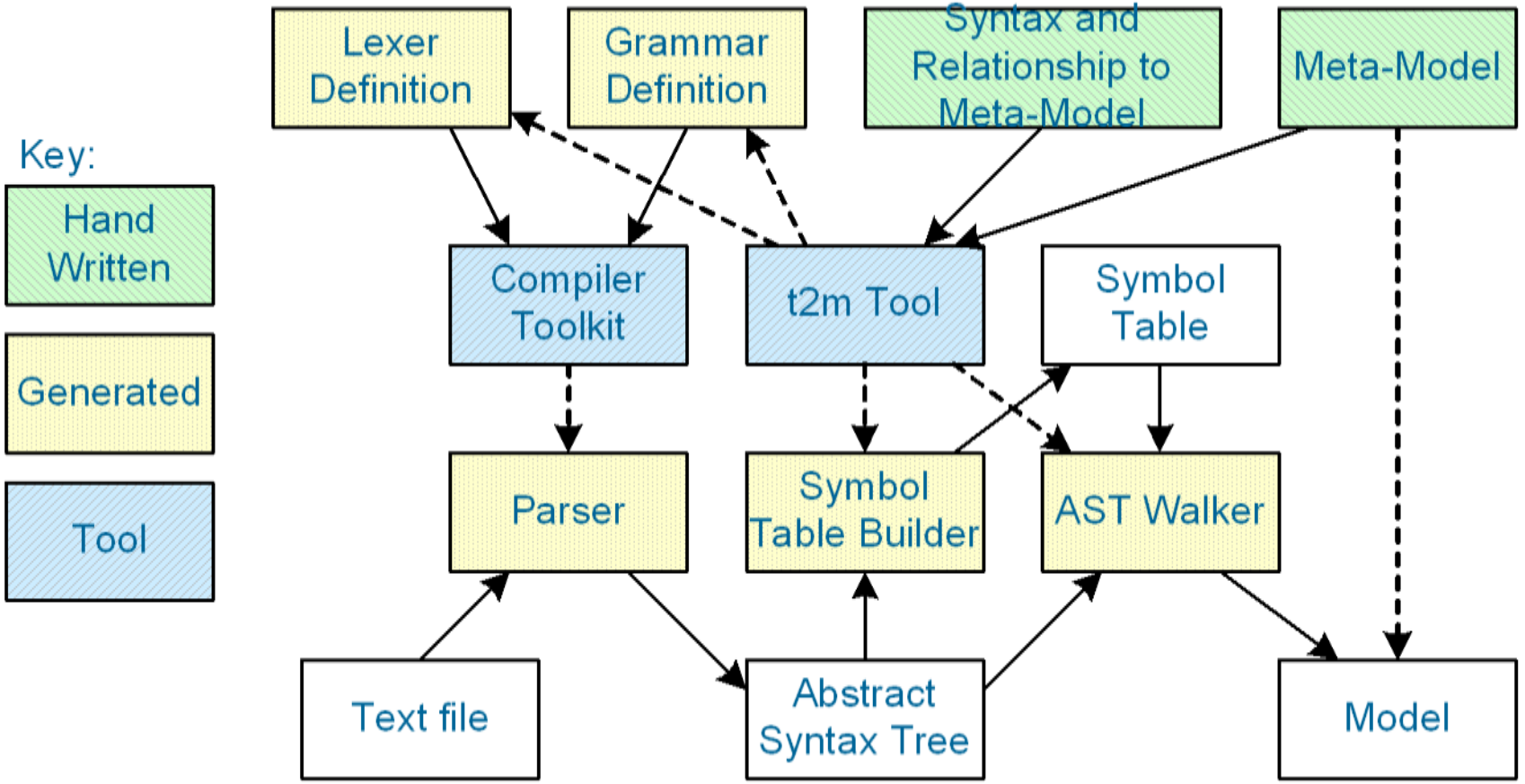
Textual Implementation of DSL

- Load instance of meta-model from textual syntax
- Implementation without any modelling tools:



Textual Implementation of DSL

- Aim with tool



Graphical Implementation of DSL

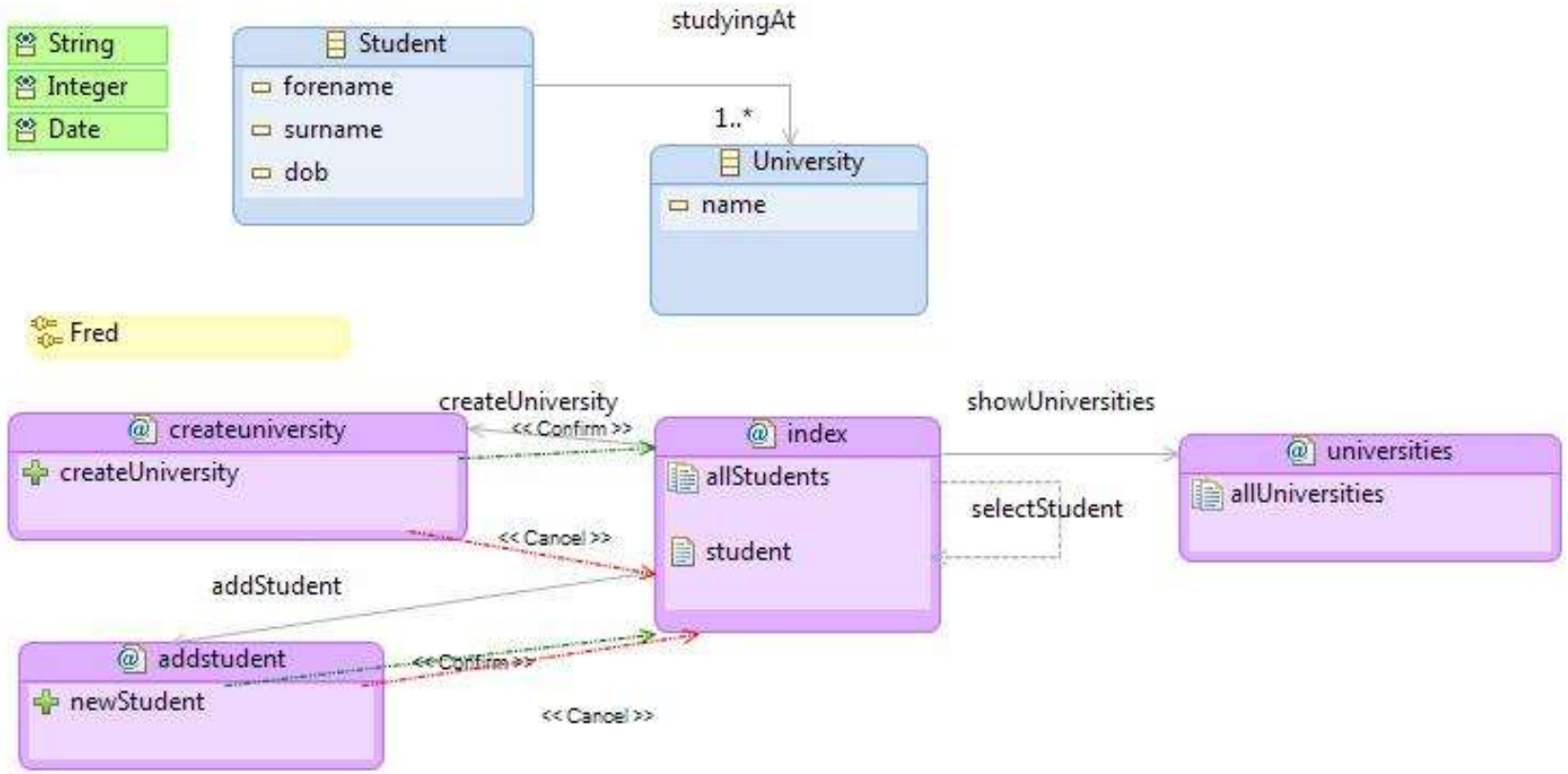
MANCHESTER 1824
The University of Manchester

Use of DSL

Development Times

- Learning curve to use hard-crafted approach
- Learning curve to use model-driven approach
  - bigger for simple technology
- What about when target complex technology?
- Still need to understand target
  - artefacts will be reflected within model
- For experience modeller targeting complex technology, model-based approach faster
- Real benefit when target upgraded technology

Web Model


String
Integer
Date
Student
forename
surname
dob
studyingAt
1..*
University
name
Fred
createUniversity
<< Confirm >>
createuniversity
createUniversity
<< Cancel >>
index
allStudents
student
showUniversities
universities
allUniversities
selectStudent
addStudent
addstudent
<< Confirm >>
newStudent
<< Cancel >>