# Formal Analysis in Model Management

## Exploiting the Power of CZT

James R. Williams, Fiona A.C. Polack and Richard F. Paige

Department of Computer Science, University of York, UK

February 25, 2010

# Outline

# Motivation

- Model driven engineering
  - \+ Intuitive graphical notations
  - \+ Good at abstracting away detail
  - – Often informally defined semantics
- Formal methods
  - \+ Precise, unambiguous
  - \+ Formal reasoning about specifications
  - – High entry cost, insufficient training
- Take the best from both worlds
  - Use model transformations to enable verification of diagrams
  - Diagram $\longrightarrow$ formal language

# Motivation

- Model driven engineering
  - \+ Intuitive graphical notations
  - \+ Good at abstracting away detail
  - – Often informally defined semantics
- Formal methods
  - \+ Precise, unambiguous
  - \+ Formal reasoning about specifications
  - – High entry cost, insufficient training
- Take the best from both worlds
  - Use model transformations to enable verification of diagrams
  - Diagram $\longrightarrow$ formal language

# Motivation

- Model driven engineering
  - \+ Intuitive graphical notations
  - \+ Good at abstracting away detail
  - – Often informally defined semantics
- Formal methods
  - \+ Precise, unambiguous
  - \+ Formal reasoning about specifications
  - – High entry cost, insufficient training
- Take the best from both worlds
  - Use model transformations to enable verification of diagrams
  - Diagram $\longrightarrow$ formal language

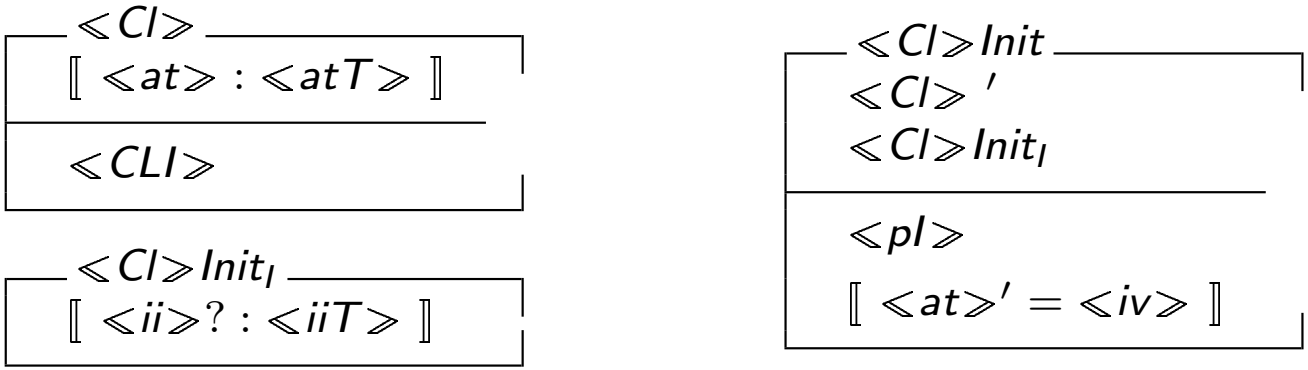
# Background

## Amálio's GeFoRME (2007)

- Generative Frameworks for Rigorous Model Driven Engineering
  - Enables construction of formal models from diagrams
  - Each diagram is a partial description of a complete model
  - Formal Template Language (FTL)
- UML+Z Framework
  - Builds Z specifications from UML class and state diagrams
  - ZOO : object-oriented style of Z
  - Catalogue of templates representing common ZOO patterns
  - Rigorous but flexible semantics

# Background

## Amálio's UML+Z Framework




$\vdash?\ \ \forall \ll Cl \gg Init_I \bullet [\![ \ll iv \gg \in \ll atT \gg ]\!]$

$\vdash?\ \ \exists \ll Cl \gg Init \bullet \text{true}$

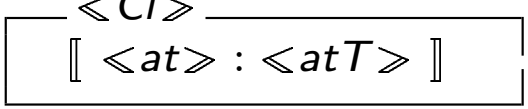
- FTL templates have associated meta-proofs
  - If the generated specification is well-formed, then the proofs hold
  - Therefore we only need to type-check the specification

# AUtoZ
## Overview

- Eclipse-based framework for integrating standard modelling techniques and formal methods
- Implementation of UML+Z framework
  - FTL templates transformed into Epsilon Generation Language
  - Template application logic file
- Allows formal methods tools to be "plugged-in"
- Goal
  - UML $\rightarrow$ Z $\rightarrow$ UML
  - Formal methods used "under the hood"

# AUtoZ
## Z Generation

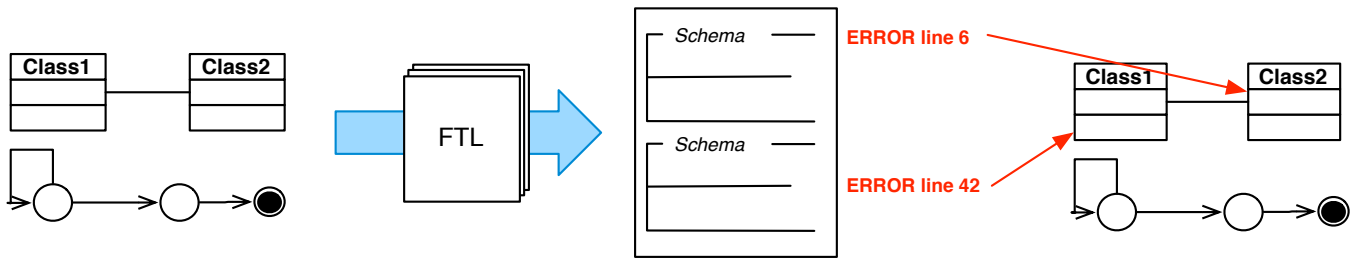
- Epsilon: a family of languages for model management
  - languages for: model transformation, model comparison, model merging, model validation, ...
  - `www.eclipse.org/gmt/epsilon/`
- The Epsilon Generation Language
  - Template-based model-to-text transformation language
- FTL templates translated into EGL




```
\begin{schema}{[%=c.name%]}
    [% for (a in c.ownedAttribute) { %]
    [%=a.name%] : [%=a.type.name%] \\ [% } %]
\end{schema}
```

# AUtoZ

## UML → Z → UML




- Many formal methods tools produce messages related to line numbers in a specification
- We need to map this back to the diagram

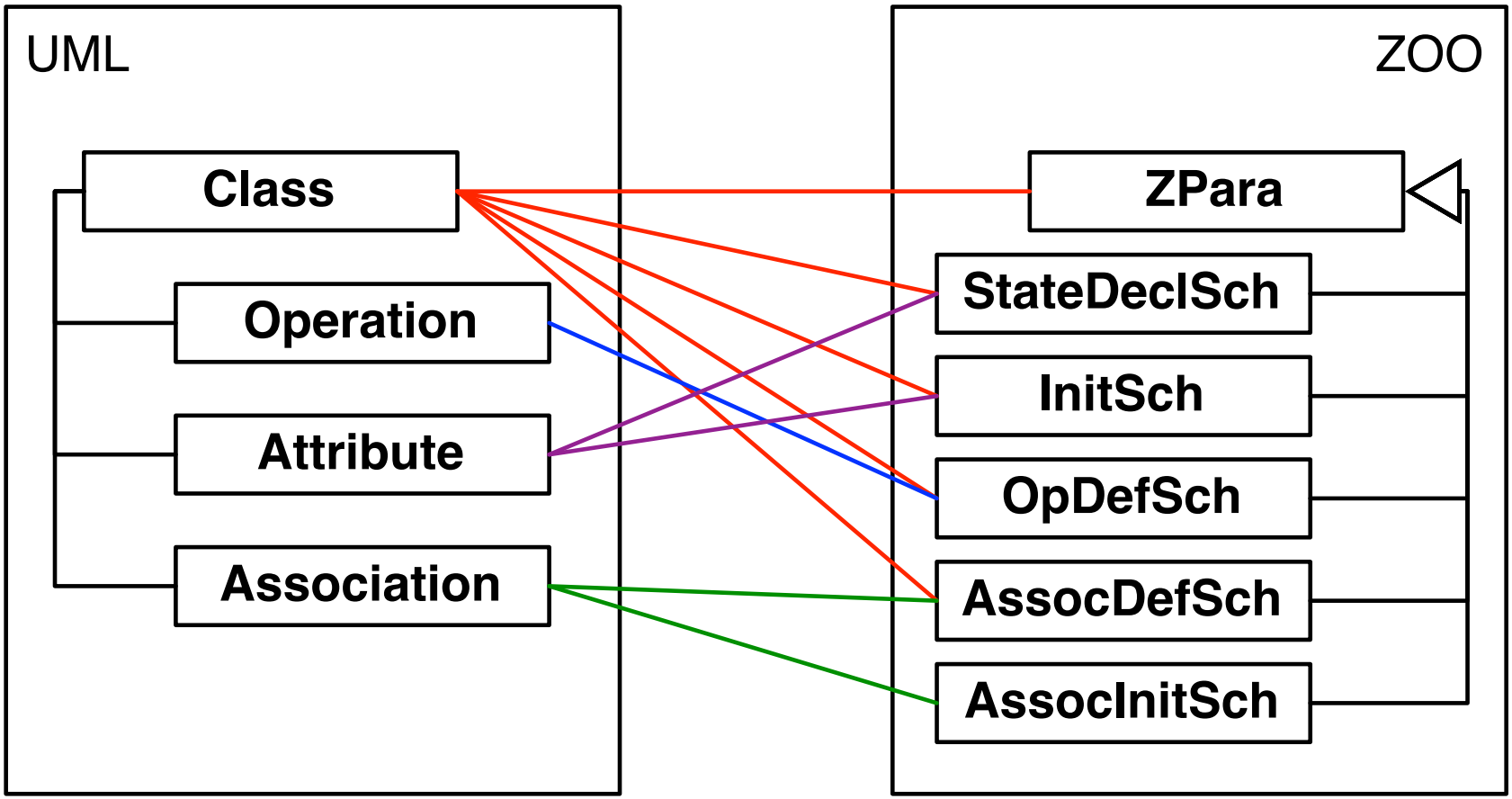
# AUtoZ
## CZT

- Community Z Tools project (`czt.sourceforge.net`)
  - Open-source project providing tool support for Z
  - Parser
  - Type-checker
- ZML: XML markup for Z
  - Annotated with messages from CZT

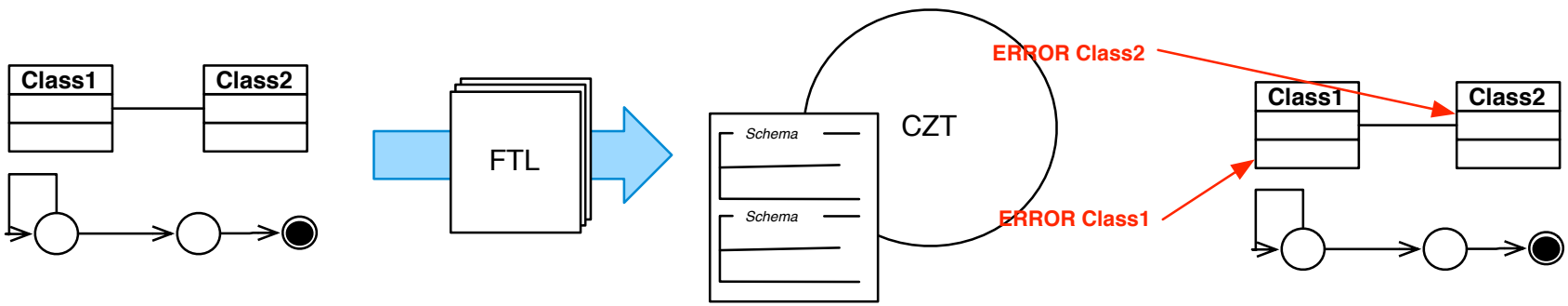
# AUtoZ
## AUtoCZT

- Both UML and ZML have a precisely defined and machine implemented abstract and concrete syntax
  - In the form of metamodels
- This allows us to create traceability links
  - Map Z elements to diagrammatic elements
- CZT used to parse and type-check the specification
- Any messages can then be related directly to the diagram

# AUtoZ
## AUtoCZT

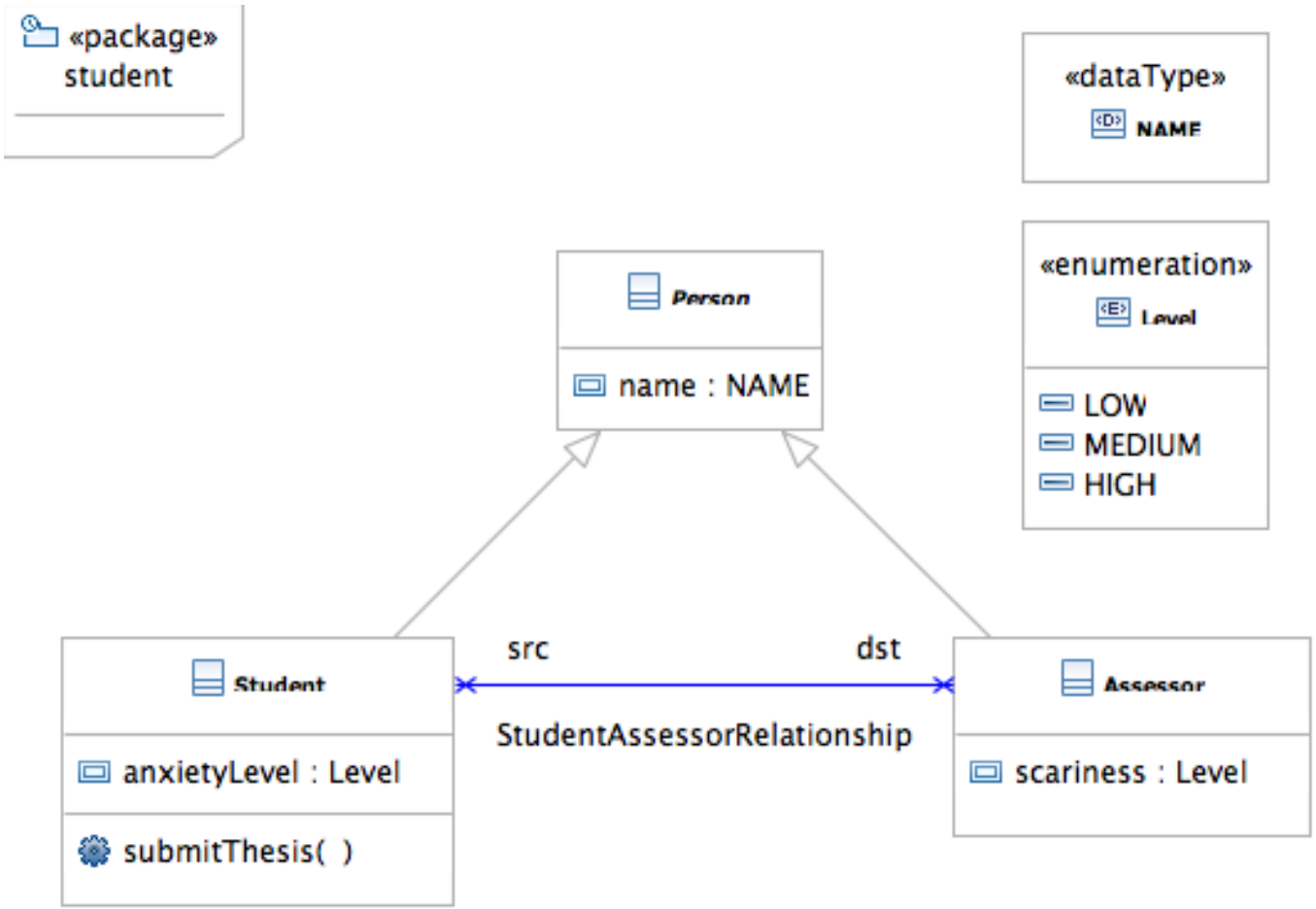


# AUtoZ
## AUtoCZT



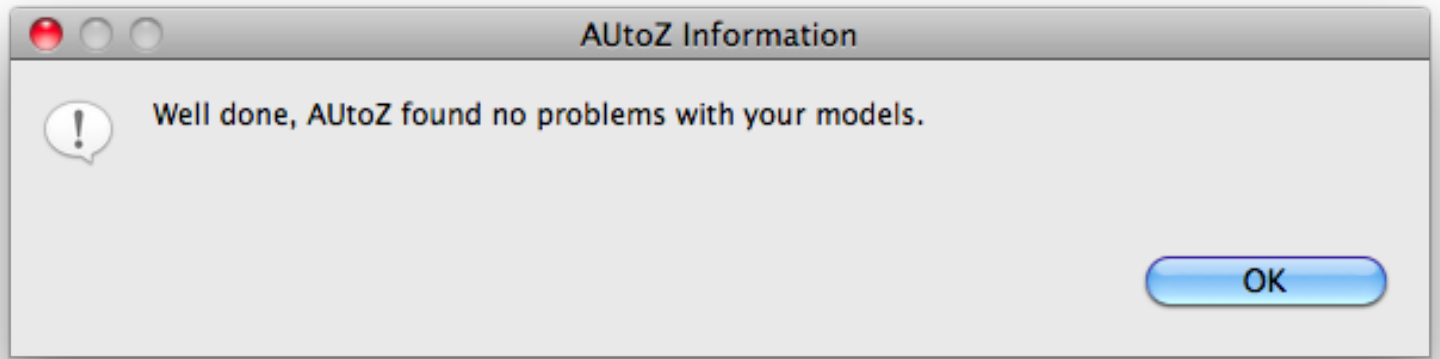
# AUtoCZT

## Example : UML Diagram



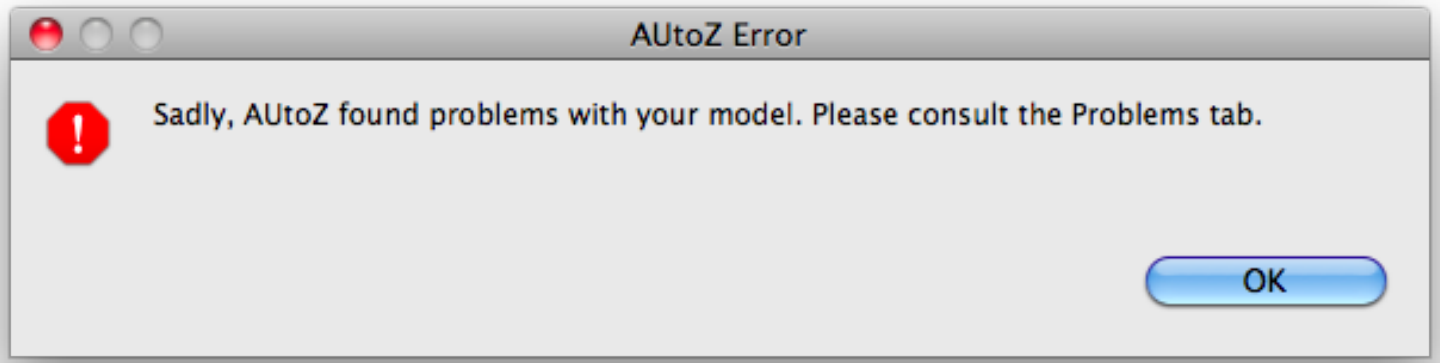
# AUtoZ

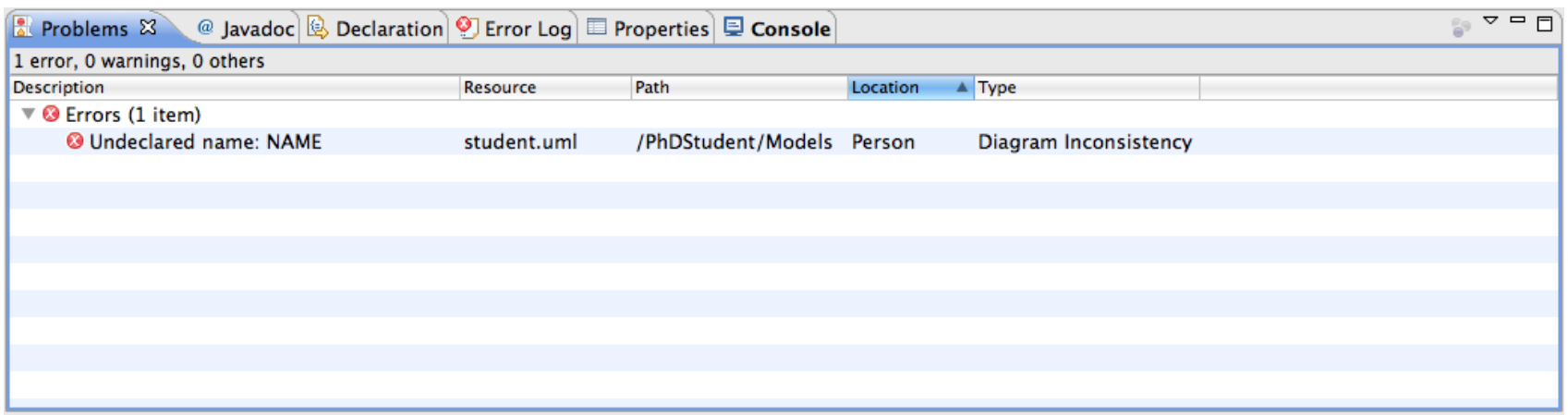
## Example : Valid Diagram



# AUtoCZT

## Example : Invalid Diagram



# Summary

- Introduced a framework for model verification
- Linked UML and ZML at the metamodel level
- Completed the tool chain to allow non formal methods experts to utilise the power of formal methods...
- ... and FM users can use MDE to quickly generate formal specifications

# Summary
## Future Work

- Specifying class operations
- Accept UML diagrams from other sources
- Improved FTL catalogue manager
  - Develop custom FTL templates
  - Automatically converted to execution language
  - Write and check meta-proofs
  - FTLogic – small language for writing template application logic
- Large case studies

# The End

- Thanks for listening
- More Information
  - `www.jamesrobertwilliams.co.uk/autoz`
  - {jw, fiona, paige} @cs.york.ac.uk