Automated formalisation for verification of diagrammatic models

James Williams and Fiona Polack
University of York, UK

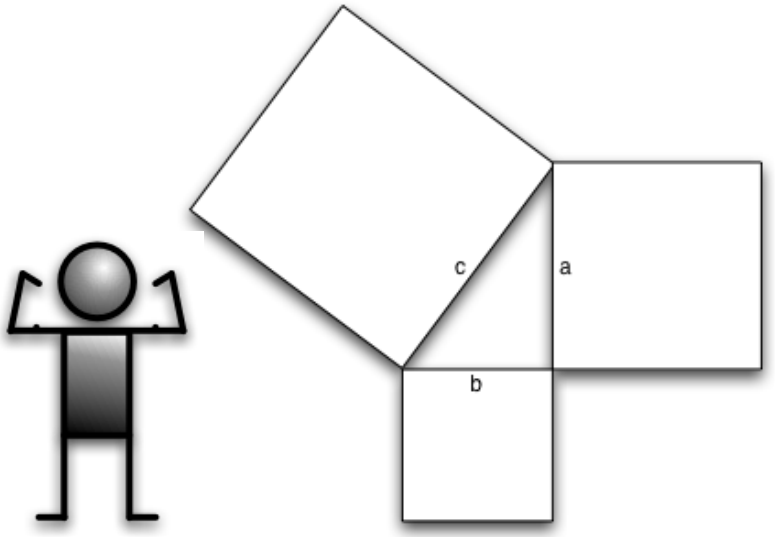
Outline

- Motivation
- Amálio's Template-based Formalisation
- The AUtoZ Tool
- Future Work

Motivation : Diagrammatic Approaches

- Diagrammatic approaches much used for designing systems
- Diagrams are intuitive and simplify the presentation of a complex idea

$a^2 + b^2 = c^2$



- However, diagrams often lack rigorous semantics and are insufficient for verification

Motivation : Formal Methods

- Formal methods used in critical systems and academia
- Mathematically state the description of a system
- Formal, unambiguous semantics
  - Amenable to verification

- Widely viewed as inaccessible by practical engineers

Method Integration (1980s -> present)

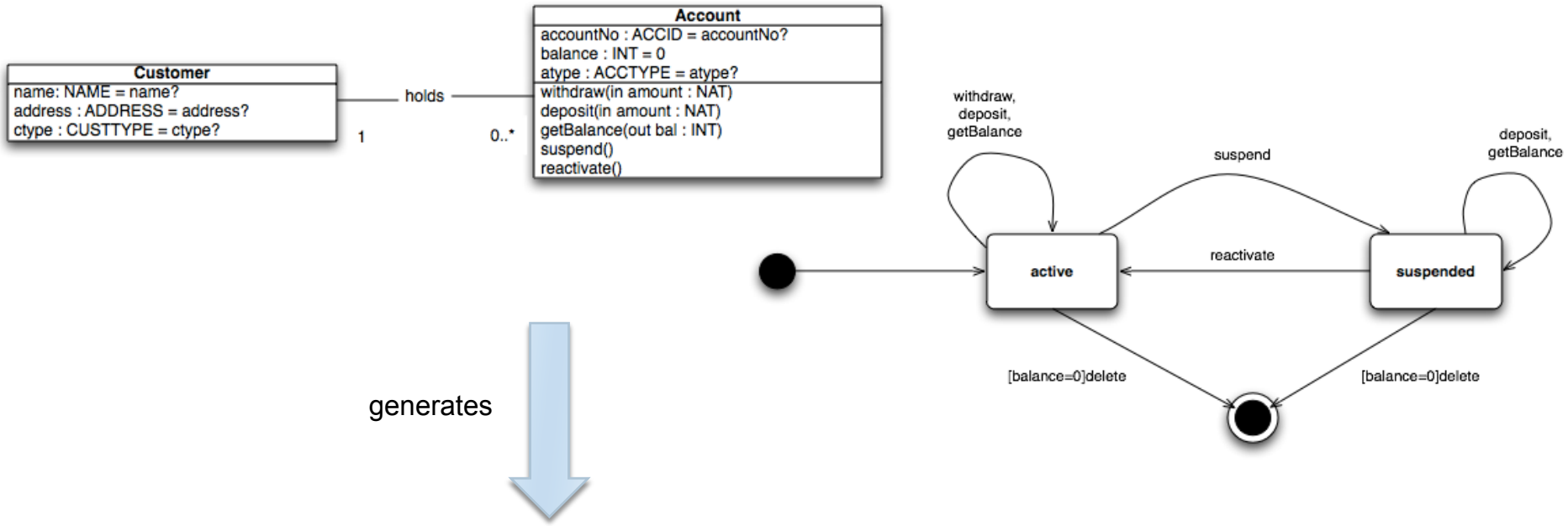
- Use the intuitive graphical design notations
  - But formally prove that they are correct
- Reduce the time taken to create formal specifications
- Better improve mainstream software
  - Fewer bugs
  - Cheaper to develop

- Many, many variants proposed
- Few integrations have found favour
  - Formalisation doesn't always align to engineering practices
  - Engineers are exposed to the formalisation

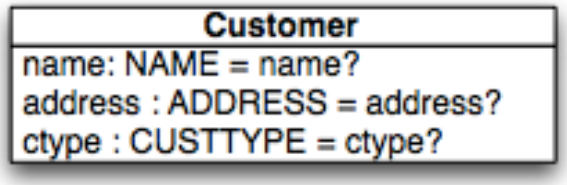
Amálio : GeFoRME (2007)

- 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

Amálio : UML+Z Framework



Amálio : UML+Z Example



Template T5: Intensional state space and initialisation with consistency conjectures

Customer

$$name : NAME$$
$$address : ADDRESS$$
$$ctype : CUSTTYPE$$

---

true

CustomerInit

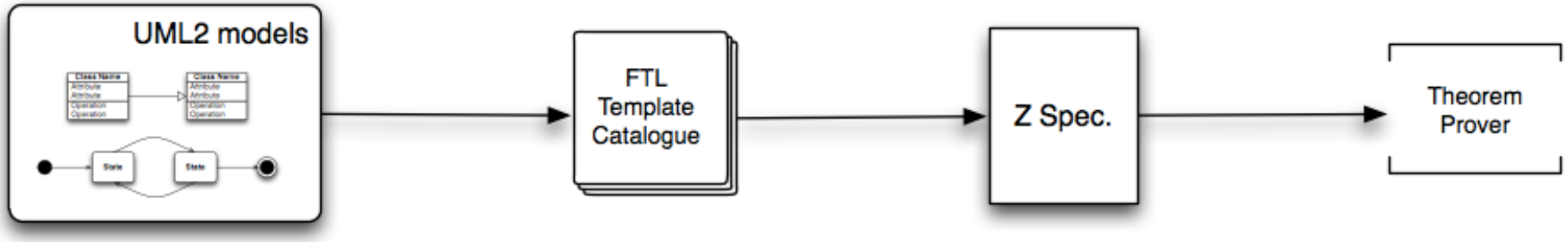
$$Customer'$$
$$name? : NAME$$
$$address? : ADDRESS$$
$$ctype? : CUSTTYPE$$

$$name' = name?$$
$$address' = address?$$
$$ctype' = ctype?$$

$$\vdash? \ \forall name? : NAME; \ address? : ADDRESS; \ ctype? : CUSTTYPE \bullet$$
$$name? \in NAME \land address? \in ADDRESS \land ctype? \in CUSTTYPE$$

$$\vdash? \ \exists CustomerInit \bullet \text{true}$$

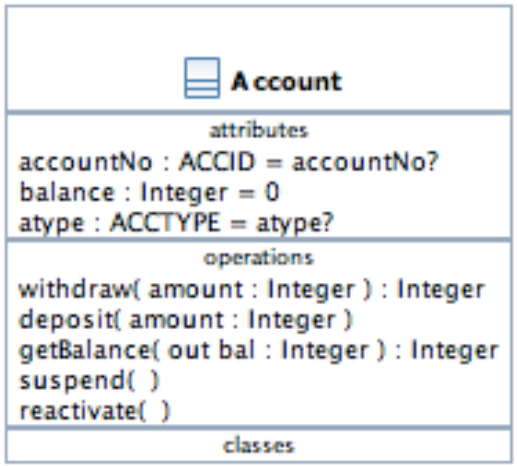
Amálio : UML+Z Method



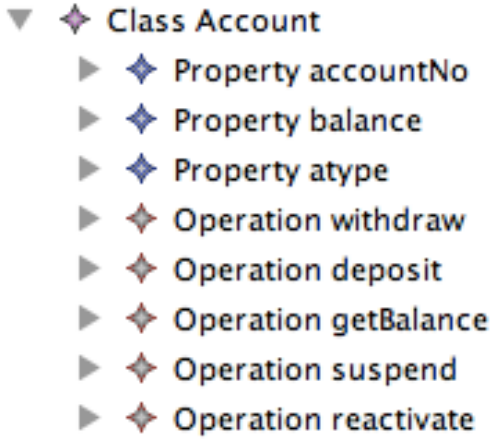
- Amálio's process is manual
- AUtoZ automates this process

Automation : Model Transformation

- Models are defined by their **meta-model**
- Graphical models can be represented textually by reference to their meta-model
- Able to access elements of the model programmatically



Class drawn using Eclipse UML2 plug-in



Class viewed in Epsilon's Exeed editor

Automation : Epsilon

- Eclipse-based model-management suite
  - Code generation
  - Model comparison
  - Merging
  - Validation
  - **Model transformation**

- The Epsilon Generation Language (EGL)
  - Model-to-text transformation
  - FTL templates written in EGL

Automation : Turning UML into Z

- FTL template:

- Example: List the properties of the Account class
  - FTL

```
Class name = <Cl>
Class attributes = [[aT]]
```

  - EGL

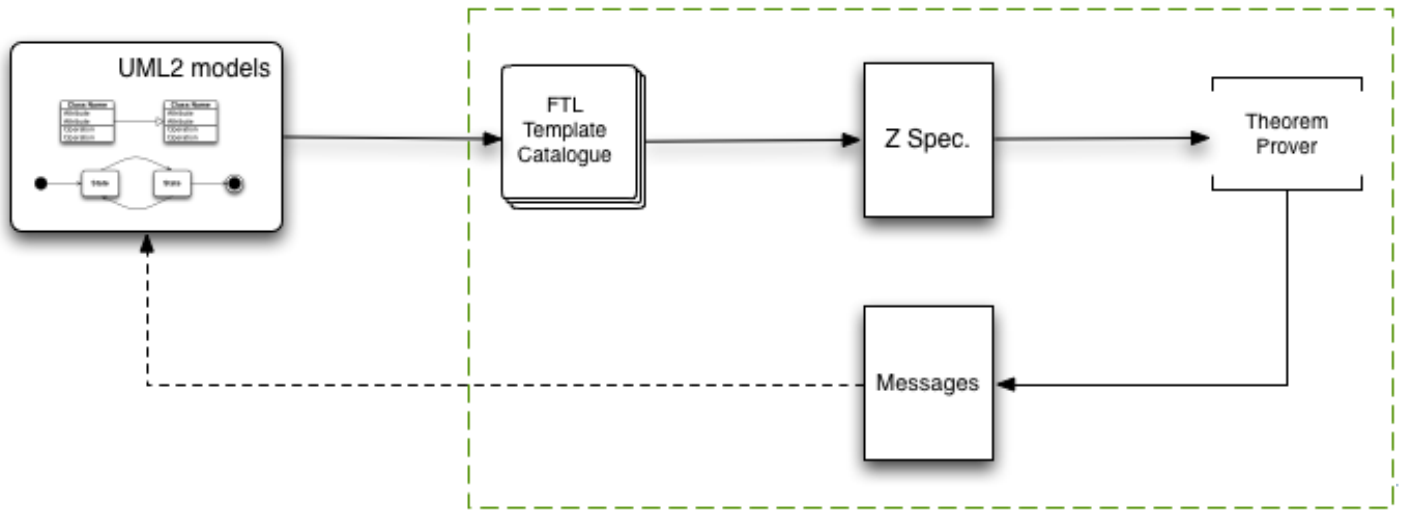
```
Class name = [%=c.name%]
Class attributes = [% for (a in c.properties){ %] [%=a.name%], [% } %]
```

  - Output

```
Class name = Account
Class attributes = accountNo, balance, atype,
```

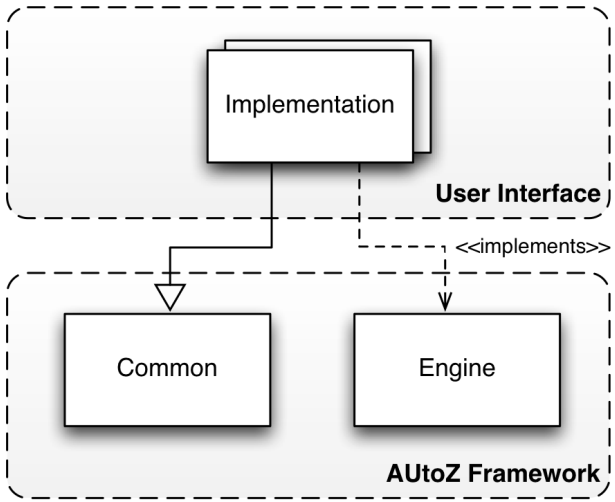
The End Result : AUtoZ

- **A**utomatic formalisation of **U**ML **to Z**
- Input : UML class and state diagrams
- Output : Messages regarding the validity of the diagrams
  - LaTeX-Z Specification
- User sees no Z notation or theorem prover messages



AUtoZ : Framework

- Developed as a plug-in for the Eclipse IDE
- Component architecture
  - Facilitates development of tool specialisations
  - Framework layer provides interfaces that specialisations must deploy



AUtoZ : Evaluation

- Better than manual?
  - Faster, with same results
- Scalability
  - Can handle multiple class and state diagrams of any size
  - Current approach is time consuming
    - Every operation of every class needs extra information to instantiate the templates, currently captured using dialogs
- Usability
  - Formality not completely hidden from user
  - Complete regeneration required if UML model changes

Future Work : Research Areas

- Operations
  - Object Constraint Language
- Error handling
  - Determine mapping between each line of the specification, and components of the UML diagrams
    - Community Z Tools
- Formal details
  - Elaborate operators available in dialogs

Future Work : Tool Extensions

- Link to CZT - AUtoCZT
- Automatic template translation
- Template catalogue
- Examples and help
- Improved run configuration
- Inclusion of remaining FTL templates
- Generalising tool support

More Information

- Tool demonstration
  - Tuesday 16:00 – 17:20
- Internet
  - www.jamesrobertwilliams.co.uk/autoz.php
- E-mail
  - jw@cs.york.ac.uk
  - fiona@cs.york.ac.uk