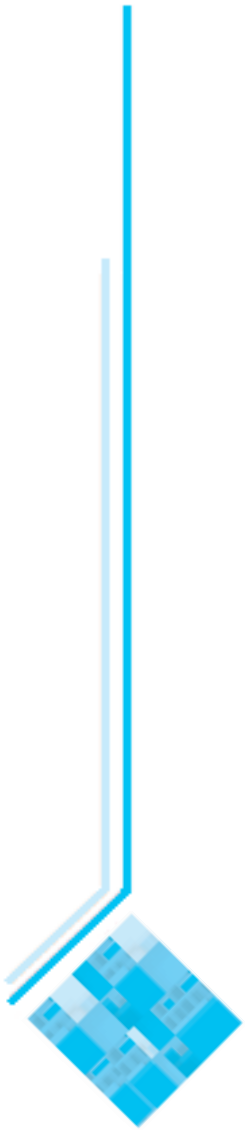


Model-Oriented Programming: Bridging the Model-Code Divide

Modeling in Software Engineering (MiSE 2013)

Omar Badreddin, Timothy C. Lethbridge

University of Ottawa

tcl@eecs.uottawa.ca

http://www.eecs.uottawa.ca/~tcl

http://www.umple.org

Key Ideas of Model-Oriented Programming (MOP)

It should be possible to

- Use a 'nice' textual form for any modeling construct
- Embed code in model
- Embed model in code

Umple allows this transparently for

- Java
- C++
- PhP
- And even all at the same time!

Why Model-Oriented Programming

Real programmers still like code

Outside the critical systems domain, most models are just documentation

- And tend to be:
  - —Wrong
  - —Not executable anyway

Those of us who have done real model-driven development know doing so is a 'no brainer' for productivity + quality

- We just have to bridge the model-code divide

Why not just use a ‘UML Action Language’ like Alf?

We don’t really need yet another language for such elements

People want to

- Leverage their existing code and libraries
- Do this in small increments

Umple as a MOP tool

It 'eats its own dogfood'

- The Umple compiler is developed with itself
- http://code.google.com/p/umple/source/browse/trunk/cruise.umple/src/Umple.ump
- http://metamodel.umple.org

Umple allows embedding of code for

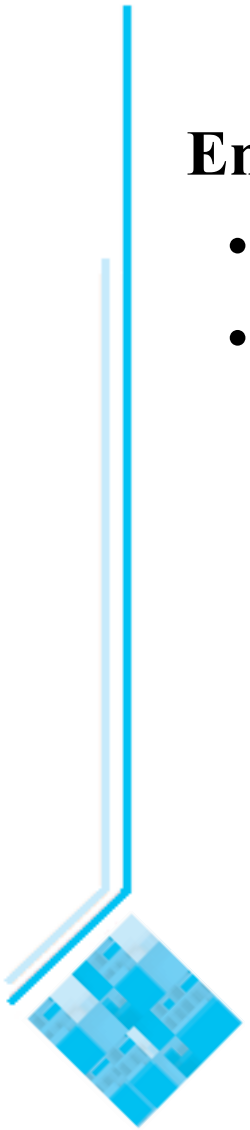
- Methods
- State machine actions, activities, guards
- Arbitrary code before/after any generated API for manipulating associations, attributes etc.

Examples

Small example: http://helloworld.umple.org

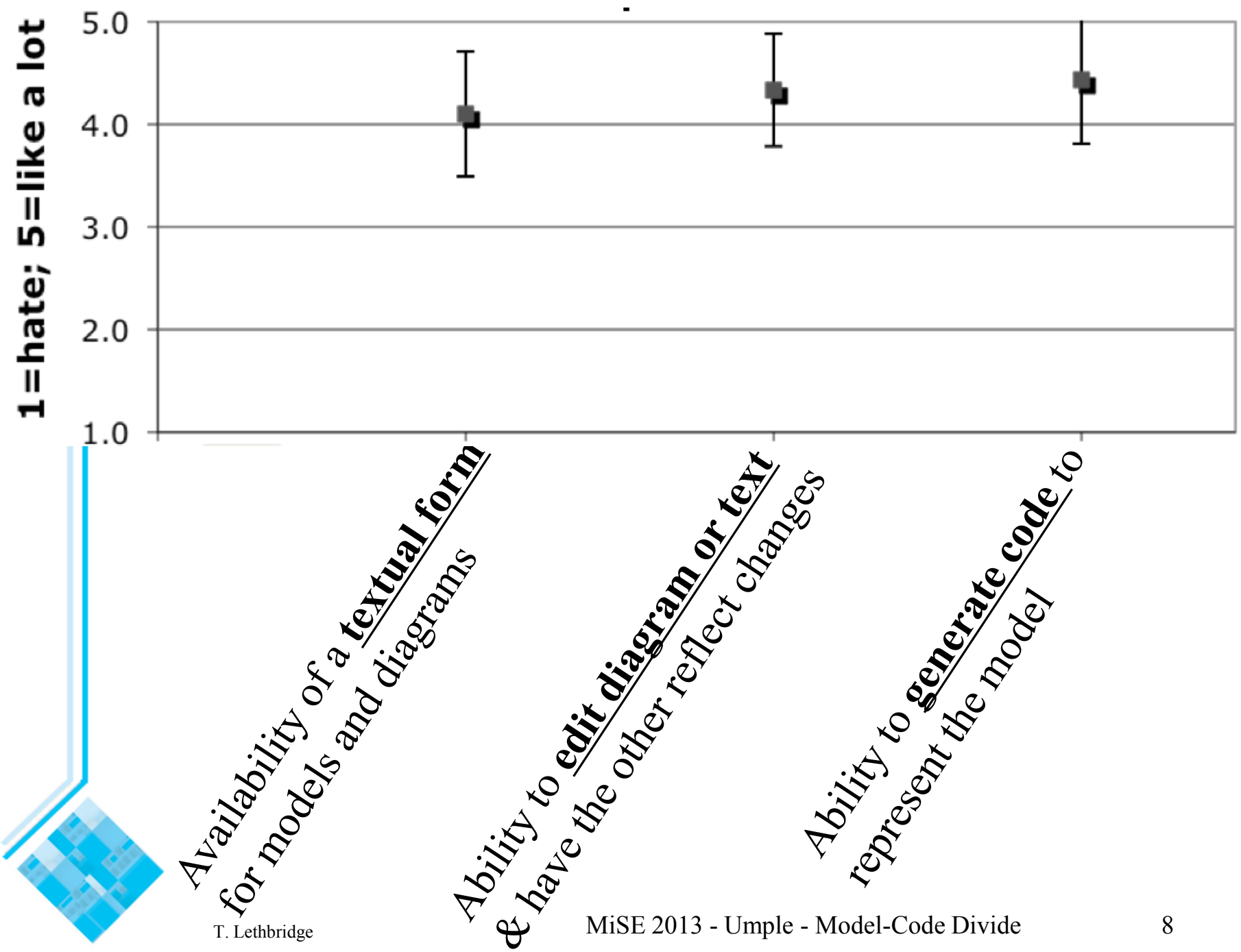
State machines: http://statemachines.umple.org

Evidence of benefit



Empirical studies show

- Umple code is easier to understand than Java
- Helps students learn to model: CSEE&T


1=hate; 5=like a lot
5.0
4.0
3.0
2.0
1.0
Availability of a textual form for models and diagrams
Ability to edit diagram or text & have the other reflect changes
Ability to generate code to represent the model

Questions?