# Modeling Bicycle Computer Modes using Xtext and Xtend

Oleksii Kononenko & Taha Rafiq

CS 846: Model-Based Software Engineering

05/04/2012

#### Overview

- Bicycle Computer Core: Java (AWT & Swing)
- Modeling: Xtext
- Transformation: Xtend

#### **Xtext**

- Open-Source DSL development framework
  - Eclipse.org project maintained by Itemis
- Eclipse IDE integration
- Based on EMF
  - Good news: Xtext does all the heavy-lifting
- What you do: Write the grammar!
- What you get: A rich DSL 'IDE' with lots of features

### What you do

```
🖹 ModeLanguage.xtext 🔀
   grammar edu.uwaterloo.bicycle.ModeLanguage
   hidden(WS, ML COMMENT, SL COMMENT)
   import "http://www.eclipse.org/emf/2002/Ecore" as ecore
   generate modeLanguage "http://www.uwaterloo.edu/modelanguage/ModeLanguage"
 ⊖Model:
       modes+=Mode*;

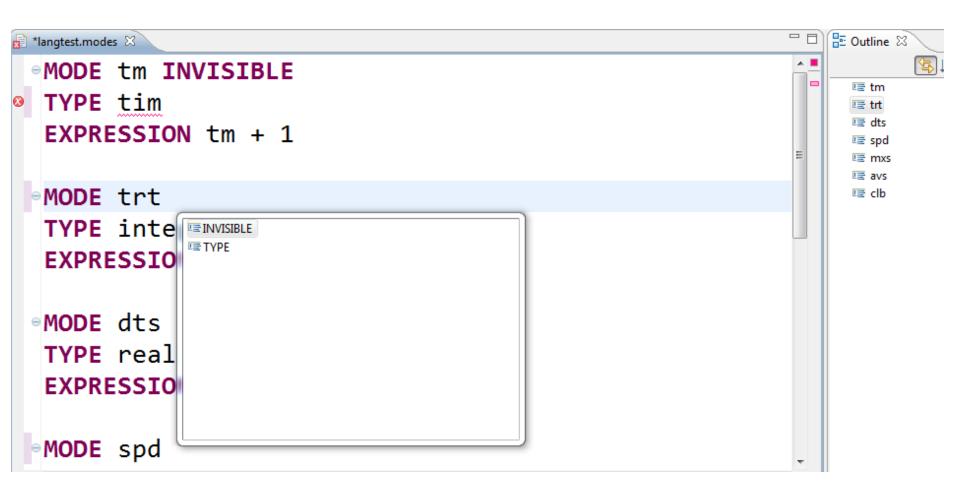
    Mode:

       'MODE' name = MNAME (inv?='INVISIBLE')? 'TYPE' type = TYPES ('CONDITION'
           expr = Expression
   j

    ○ Conditions:

       CompoundCondition
```

## What you get



#### **Xtend**

- Template language for development of <u>code</u> generators
- Comes bundled with Xtext
- What you do: Write template, iterate over AST to define transformations
- What you get: Java code generated from your model on the fly

## What you do

```
def compile(Mode m) '''
    package edu.uwaterloo.bicyclecomputer.modefunctions;
    import edu.uwaterloo.bicyclecomputer.SystemValues;
    public class «m.name» {
        public static «m.translateType» value;
        public static final String name = "«m.name»";
        public static boolean error = false;
        public static boolean overflow = false;
        public static boolean invisible = «m.isInvisible»;
        public static void calculate() {
            if (error) return;
            if («m.getOverflows») {
                error = true;
                return;
            try {
                if («m.condition.generateCondition») {
                    value = («m.translateType») «m.forReal»(«m.expr.gener
```

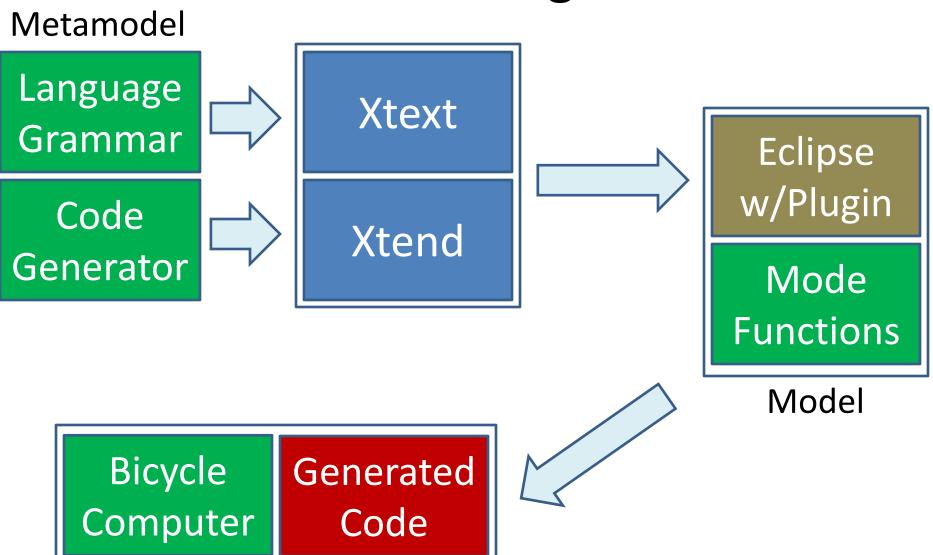
### What you get

```
package edu.uwaterloo.bicyclecomputer.modefunctions;
import edu.uwaterloo.bicyclecomputer.SystemValues;
public class avs {
    public static float value;
    public static final String name = "avs";
    public static boolean error = false;
    public static boolean overflow = false;
    public static boolean invisible = false;
    public static void calculate() {
        if (error) return;
        if (dts.overflow || dts.error || tm.overflow || tm.error) {
            error = true;
            return;
        try {
            if (true) {
                value = (float) 1.0f * ((dts.value * 3600) / tm.value);
                checkValue();
```

### Bicycle Computer Core

- Made in Java, using AWT and Swing
- Use java.reflect to dynamically detect mode classes and methods

#### How it fits together



#### Demonstration

### Insights & Lessons Learned

#### The Good

- Documentation, tutorials, webcasts available
- In active development, vibrant community
- Integrates very well with Eclipse
- Feature-rich editor
- Generates code on the fly
- Based on EMF, can be extended/integrated in many ways

### Insights & Lessons Learned

#### The Bad

- Not a lot, considering the experience of others!
- Limited support and documentation for latest version
- Need to learn Xtend; steep learning curve
- Resource hungry; needs a lot of CPU and RAM
- Inexplicable crashes while writing mode functions
  - Outline view

## Questions?