

Combinatorial Interaction Testing with CTWEDGE

Angelo Gargantini

Università di Bergamo - Italy

<http://cs.unibg.it/gargantini>

Joint work with Paolo Vavassori e Marco Radavelli

Andrea Bombarda



CTWEDGE in brief

<https://github.com/fmselab/ctwedge>

Language for CIT problems

1. with a precise formal semantics and a grammar by Xtext
2. A textual editor integrated in the eclipse IDE

Set of tools

3. for importing/exporting CIT problems
4. for generating test suites (by using external tools)

Framework

5. based on the Eclipse Modeling Framework (EMF), library to manipulate combinatorial problems in Java
6. A rich collection of Java utility classes and methods
7. A rich collection of benchmarks



INSTALLING CTWEDGE

As eclipse plugin

https://fmse-lab.github.io/ctwedge/ctwedge_update/



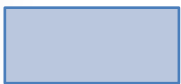
CTWEDGE EDITOR

DEMO

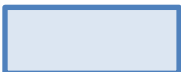




Manual
(required)



Manual
(optional)

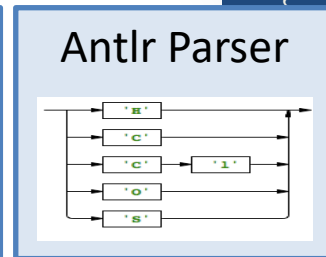
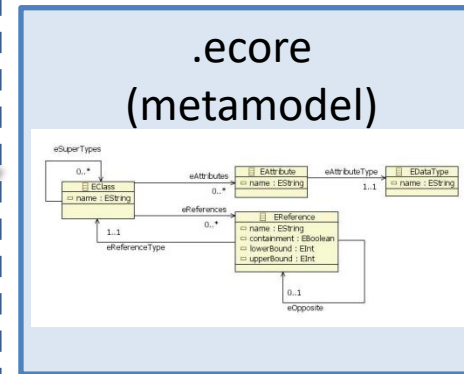


Generated

.xtext
Grammar



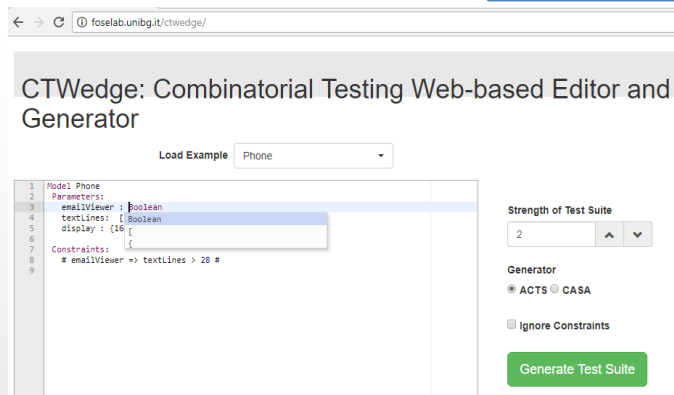
Language project



Java API

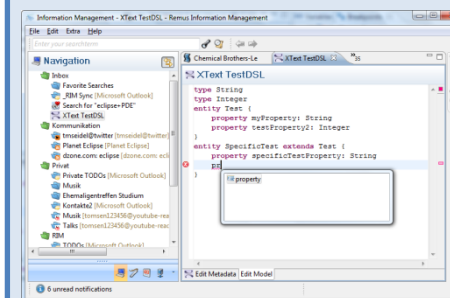
Web Editor Project

JS code



Editor Project

Editor in eclipse



Syntax coloring

Outline

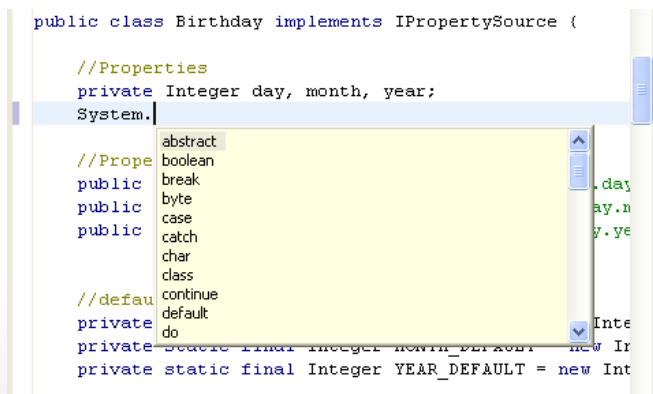
Content Assist

Formatting



Editor features

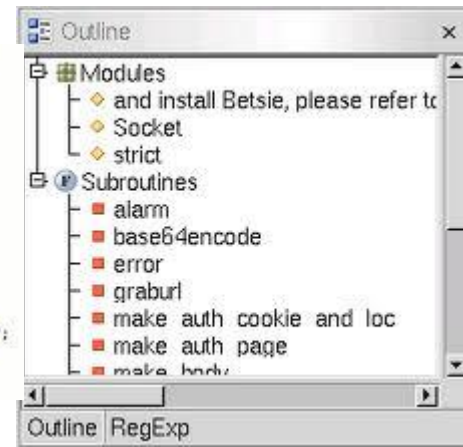
- Syntax Coloring
- Content Assist
- Template Proposals
- Rich Hover
- Rename Refactoring
- Quick Fixes
- Outline
- Folding
- Hyperlinks for all Cross References
- Find References
- Toggle Comment
- Mark Occurrences
- Formatting



```
public class Birthday implements IPropertySource {  
  
    //Properties  
    private Integer day, month, year;  
    System.  
  
    //Prope  
    abstract  
    boolean  
    break  
    public  
    byte  
    case  
    public  
    catch  
    char  
    class  
    continue  
    default  
    do  
    //defau  
    private  
    private  
    private static final Integer YEAR_DEFAULT = new Int
```



```
r path to the feature's project directory  
current version, for example '0.4.2'  
  
sion  
  
= shift;  
  
read_file('${feature_dir}/feature.xml');  
  
n/version/s*=(s*({d+}.\d+).\d+)*s)  
$!;  
ersion()])  
e version missing in ${feature_dir}/feature.xml';  
;
```



MODELLING COMBINATORIAL PROBLEMS



Grammar

- Very similar to EBNF:

CitModel:

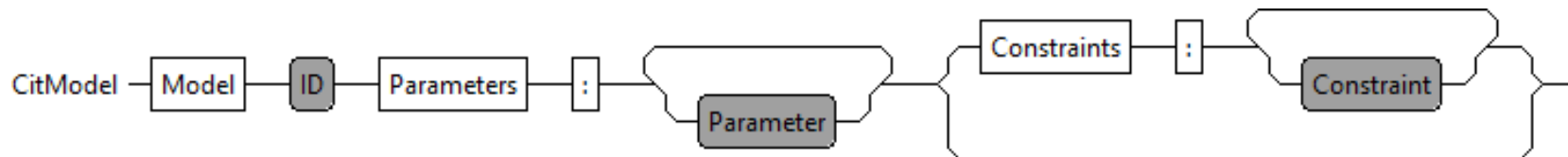
'Model' *name=ID*

//list of parameters

'Parameters' ':' (parameters+=Parameter)+

// constraints

(*'Constraints'* ':' (constraints+=Constraint)+)?;



- Translated to ANTLR



CTWEDGE Language in a glance

Model Model

Parameters:

← **Parameters**

...

Constraints:

← **Constraints**

...

Example:

*A family of phones,
that can have
several types of
cameras, display,...*



Example

```
/*  
 * This is an example model  
 */
```

Model Phone

Parameters:

```
emailViewer : Boolean  
textLines: [ 25 .. 30 ]  
display : {16MC, 8MC, BW}
```

Constraints:

```
# emailViewer => textLines > 28 #
```



Parameters and their types

- To describe a combinatorial problem would be sufficient to specify the number of variables and their cardinality.
- ctwedge language forces the designer to name parameters and to specify their types by listing all the values in their domain.
- **Choice:** explicit parameter names to facilitate the modeling of real systems and to ease the specification of constraints and seeds

Enumerative for parameters that can take a value in a set of symbolic constants.

the display of the cell phone can be colored (with 16 or 8 millions colors) or black and white,

```
display: { 16MC 8MC BW };
```



Parameters (2)

Boolean for parameters that can be either true or false.

the phone can have an email viewer

```
emailViewer: boolean;
```

Numerical values in a range for parameters that take any value in an integer range.

Phones have a number of lines between 10 and 30, but only every 5 is valid

```
textLines: [ 10 .. 30 ] step 5;
```

A list of Numbers for parameters that take any value in a set of integers.

The phone has been produced in 2012 and 2013

```
Year: {2012 2013} ;
```



Constraints

- In ctwedge, we adopt the language of propositional logic with equality and arithmetic to express constraints
- General Form (GF) constraints
 - propositional calculus and Boolean operators
 $a \text{ or } b \Rightarrow c \text{ and } d$
 - equality and inequality

If the phone has an email viewer then

 $\# \text{ emailViewer} == \text{true} \Rightarrow \text{textLines} \geq \text{threshold} \#$
 - arithmetic over the integers
 - relational and arithmetic operators for numeric terms
 $\# \text{ textLines} \geq \text{threshold} + 10 \#$
- A valid test must satisfy all the constraints



TEST GENERATION



Test generation

- CTWEDGE does not include in itself generators. Currently supports the following test generators, each defined as generator plugin:
 - AETG is a plugin developed by students following the pseudo code for the greedy algorithm of AETG.
 - IPO is a plugin developed by us following the pseudo code for IPO.
 - Random is a simple random algorithm that adds new randomly built tests until all the n-wise combinations are covered.
 - ACTS is an external test generator tool developed by the NIST.
 - CASA is an external tool for test generation based on simulated annealing by Myra Cohen and colleagues.
 - ATGT_SMT is an external tool combining heuristics and SMT solving.
- Some support constraints, seeds, ...

