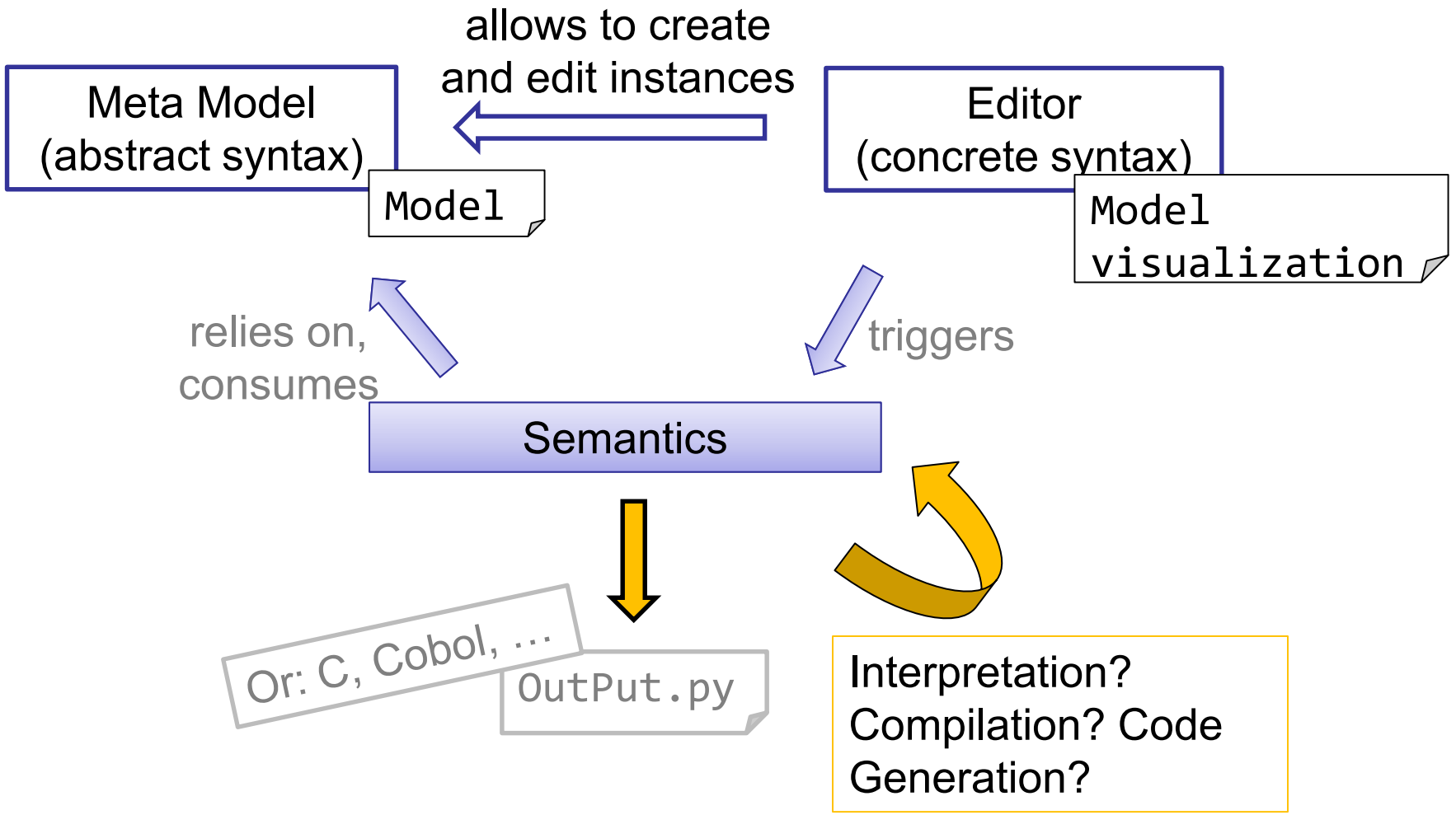
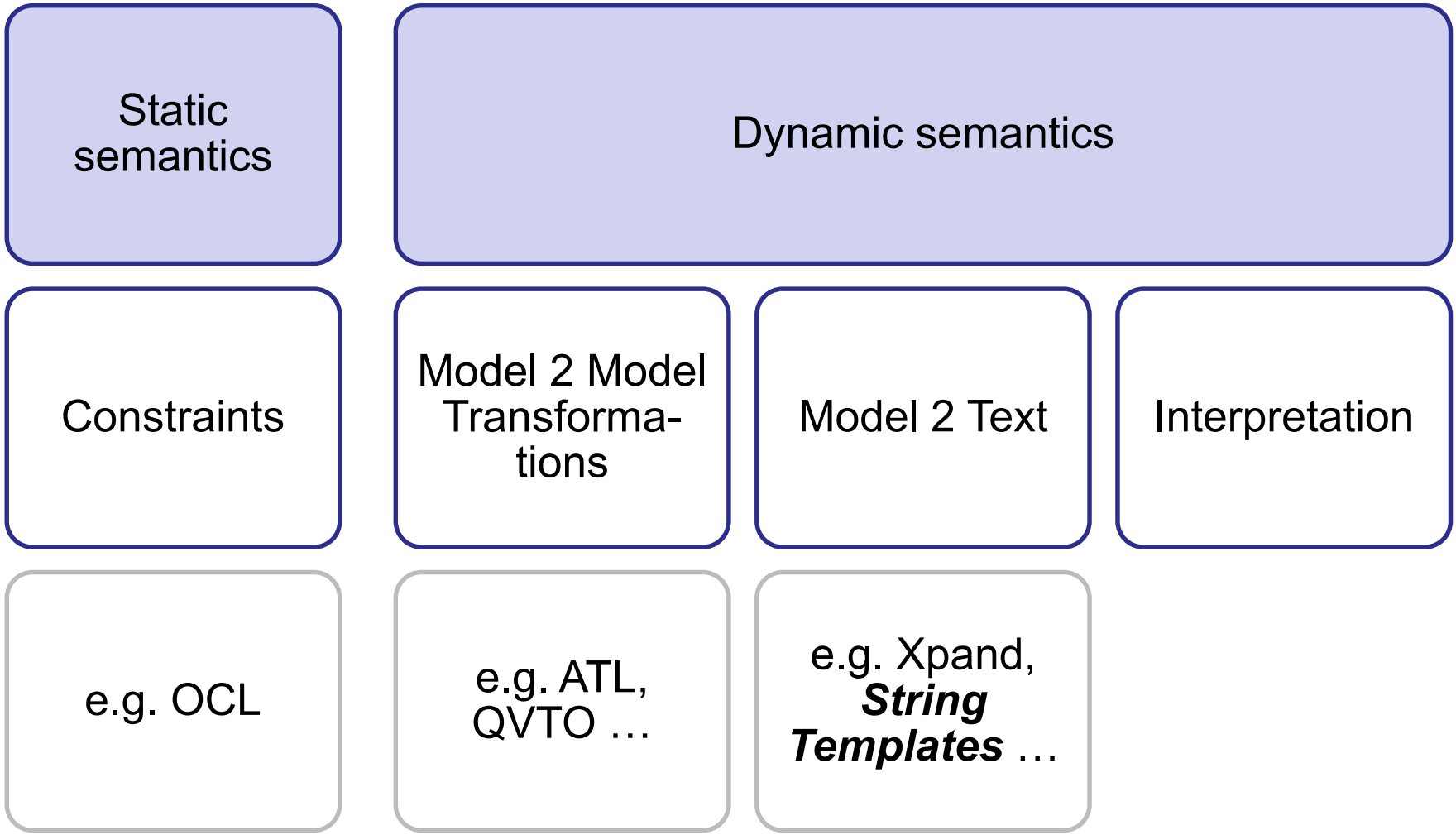
Model-Driven Engineering (MDE)
Lecture 2: Semantics

Regina Hebig, Thorsten Berger

Semantics

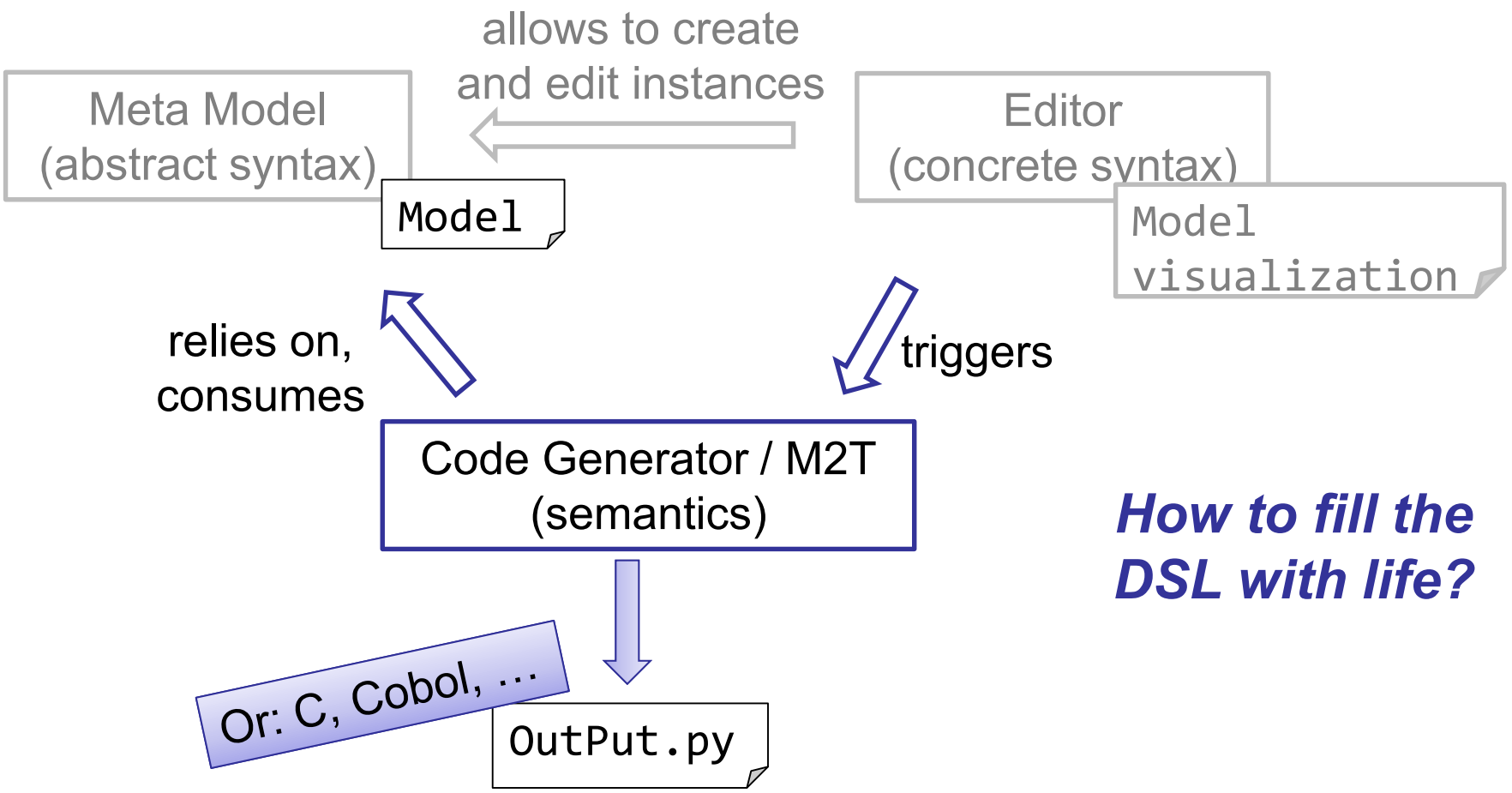

allows to create
and edit instances
Meta Model
(abstract syntax)
Model
Editor
(concrete syntax)
Model
visualization
relies on,
consumes
triggers
Semantics
Or: C, Cobol, …
OutPut.py
Interpretation?
Compilation? Code
Generation?

Semantics


Static semantics
Dynamic semantics
Constraints
Model 2 Model Transforma-tions
Model 2 Text
Interpretation
e.g. OCL
e.g. ATL, QVTO …
e.g. Xpand, ***String Templates*** …

Creating a DSL's Semantic

Model to Code with String Template



How to fill the DSL with life?

How to fill the DSL with life?

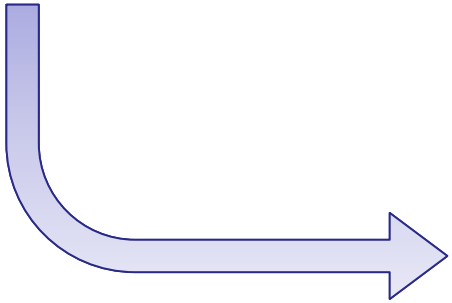
- Options:
  - Interpretation
  - Transformation / Code generation

- Today: focus code generation
  - How to get access to code (through MM - ecore)
  - Often done with templates; e.g. StringTemplate (http://www.stringtemplate.org/)

String Template

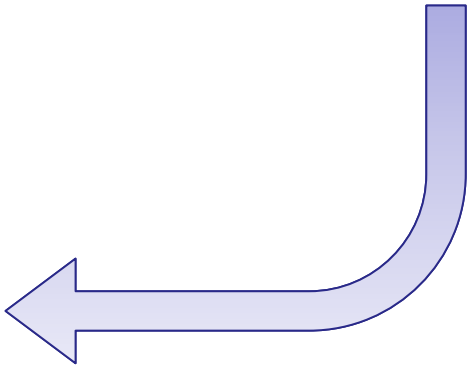
SimpleTemplate.stg

```
Hello, <name>
```



Instantiate template and fill it

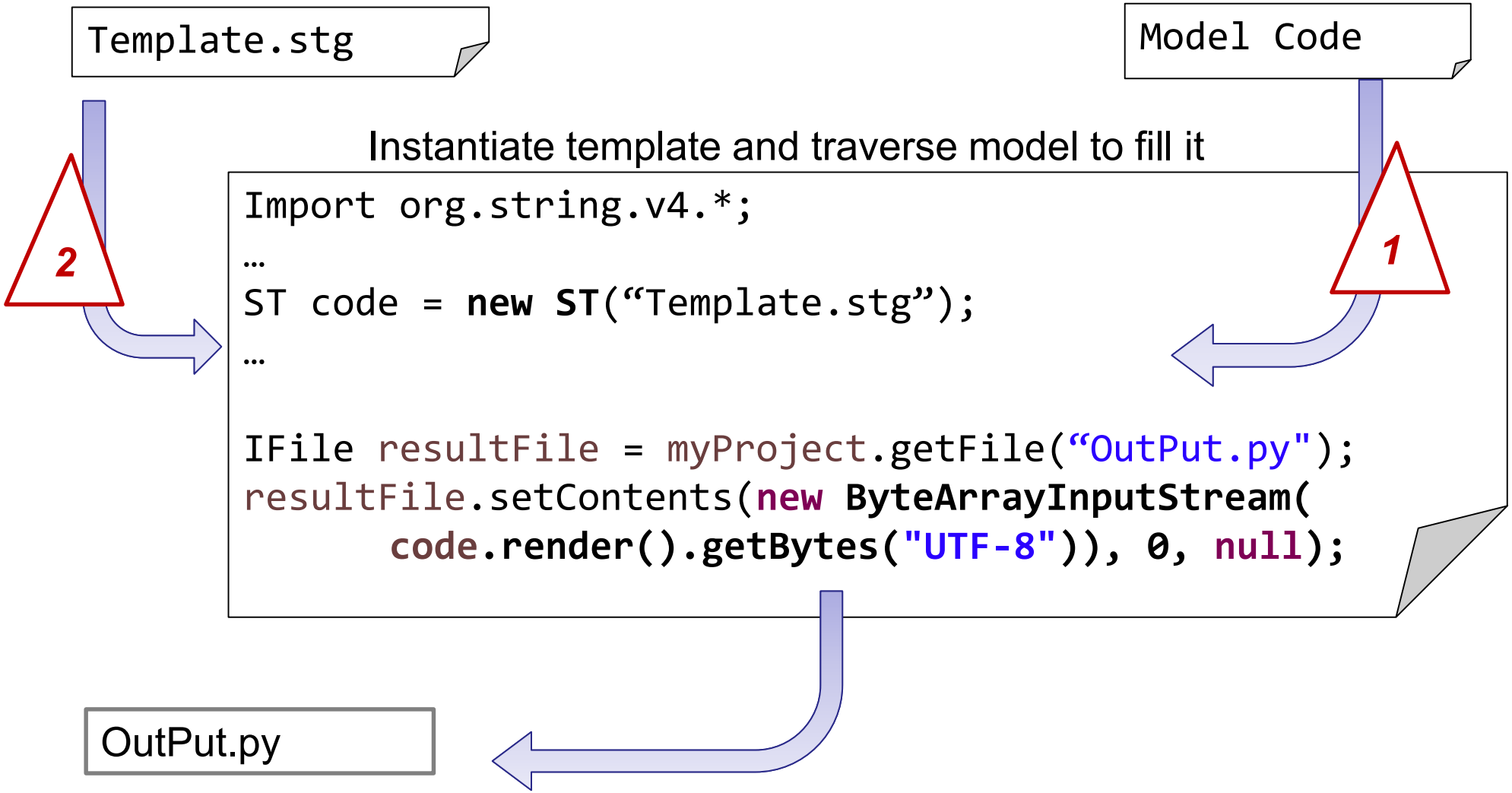
```
Import org.string.v4.*;
…
ST hello = new ST("SimpleTemplate.stg");
hello.add("name", "World");
System.out.println(hello.render());
```



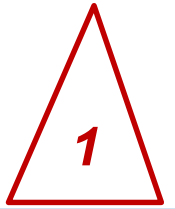
Render filled template

```
Hello, World
```

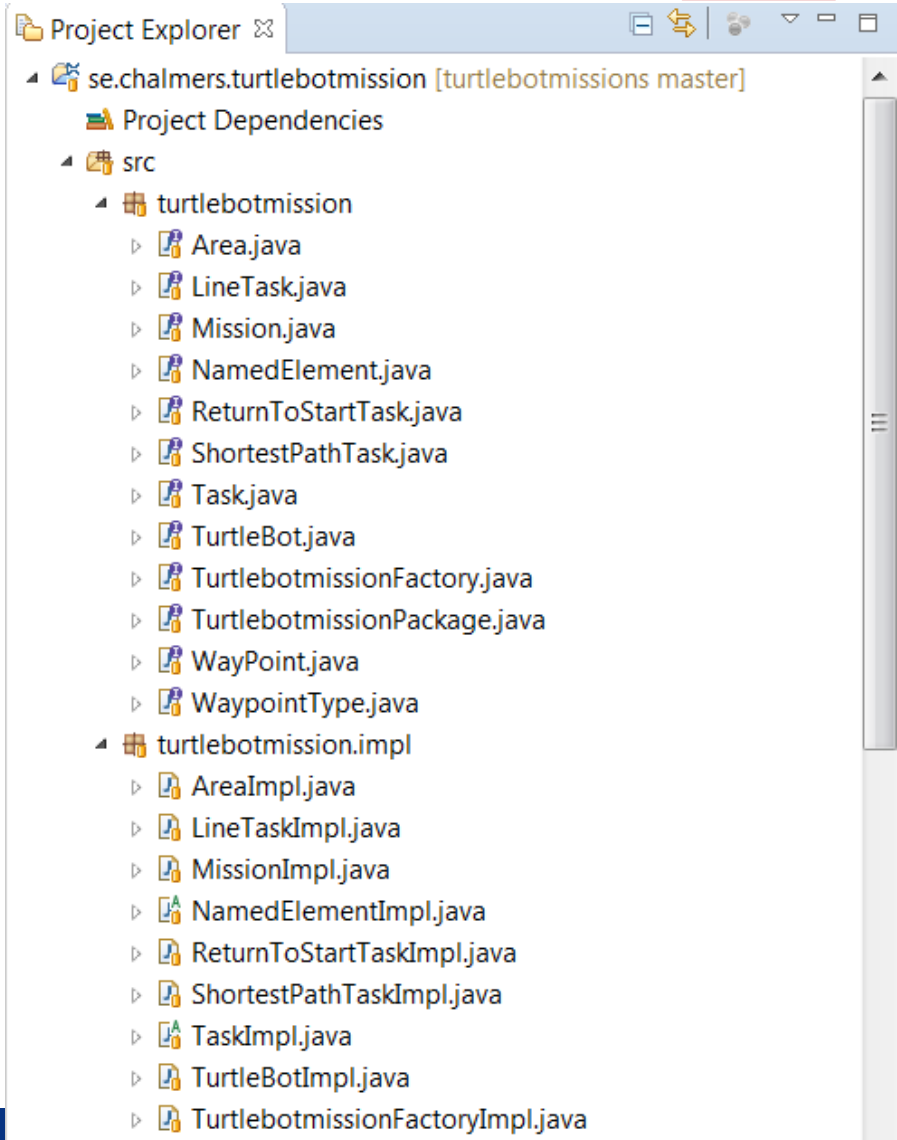
Model to Code with String Template



Generator and editor

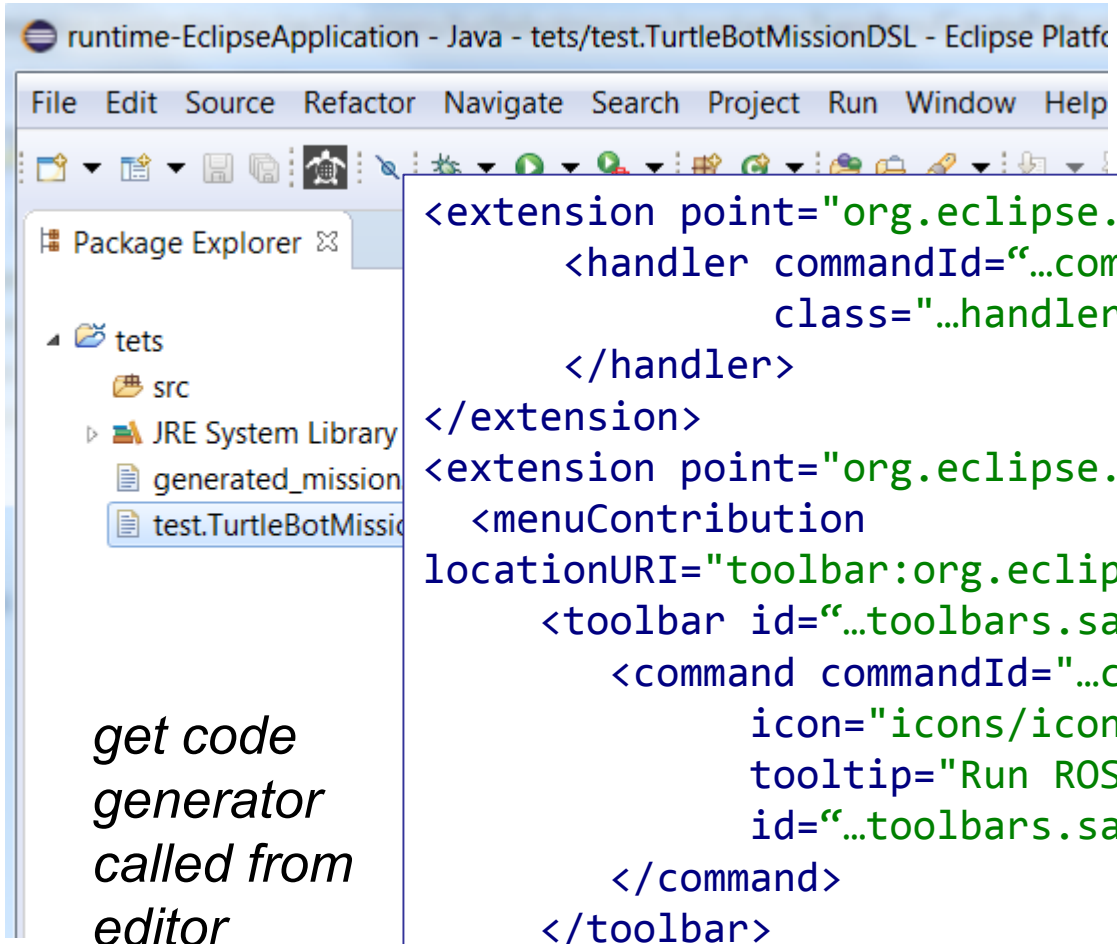


- How to:
  - Get code generator called from editor?
  - Get access to the model?
- Options:
  - Integration to save functionality
  - Addition of extra menus



Code-Generator – Hook generator to editor

- Option: Addition of extra menus



get code generator called from editor

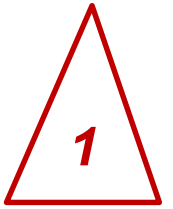
Plugin.xml

```
<extension point="org.eclipse.ui.handlers">
      <handler commandId="…commands.sampleCommand"
               class="…handlers.CreatePythonHandler">
      </handler>
</extension>
<extension point="org.eclipse.ui.menus">
  <menuContribution
locationURI="toolbar:org.eclipse.ui.main.toolbar?after=additions">
     <toolbar id="…toolbars.sampleToolbar">
        <command commandId="…commands.sampleCommand"
              icon="icons/icon48.png"
              tooltip="Run ROSJava"
              id="…toolbars.sampleCommand">
        </command>
     </toolbar>
  </menuContribution>
</extension>
```

Code-Generator – Hook generator to editor

- Option 2: Addition of extra menus

get access
to the model



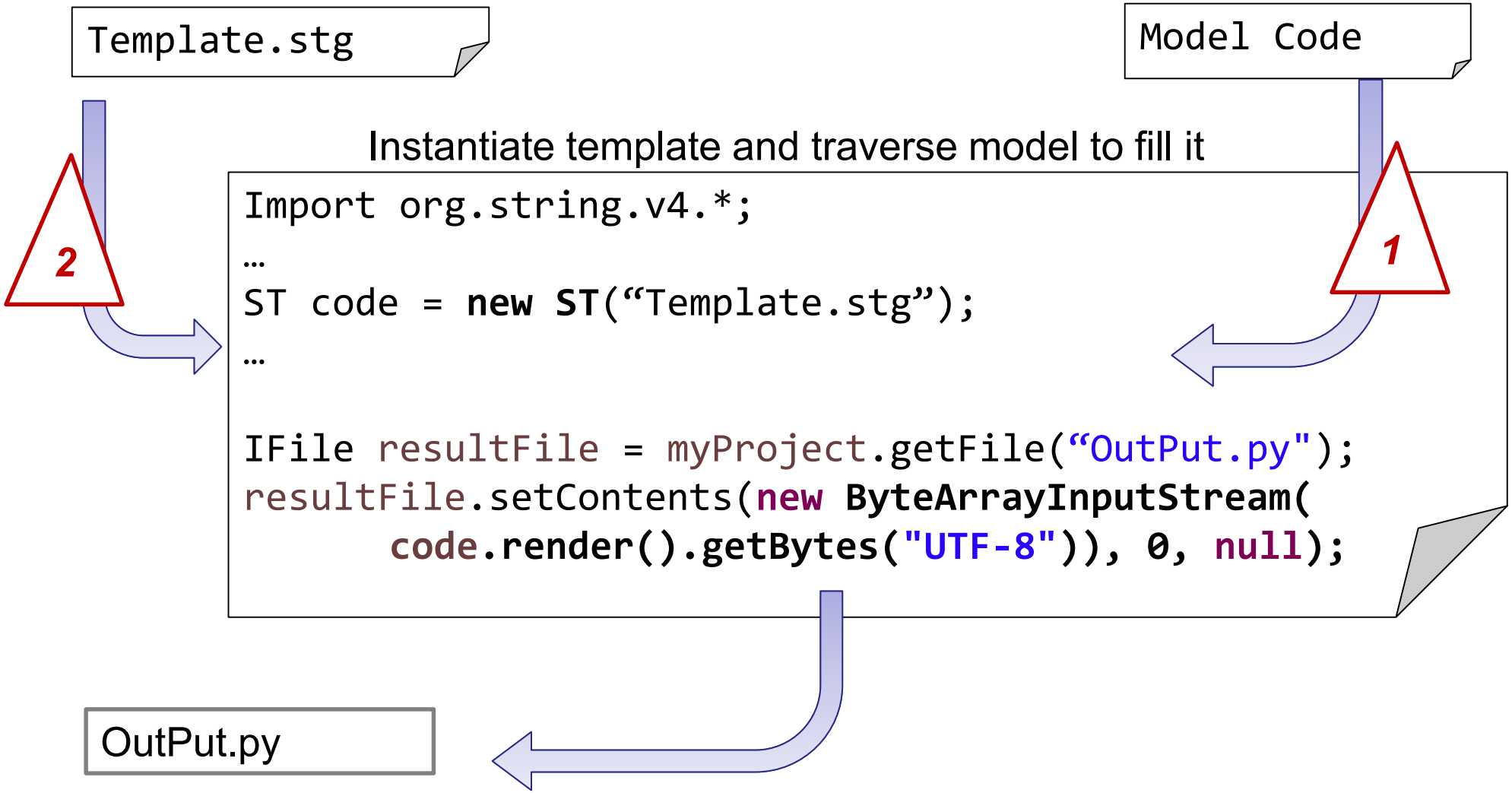
New Handler

```
public class CreatePythonHandler extends AbstractHandler {

@Override
public Object execute(ExecutionEvent event) throws ExecutionException {
 IWorkbenchWindow window = HandlerUtil.getActiveWorkbenchWindowChecked(event);
 IEditorPart editor = window.getActivePage().getActiveEditor();
 if (editor instanceof XtextEditor) {
   IXtextDocument doc = ((XtextEditor) editor).getDocument();

 doc.modify(new IUnitOfWork<Void, XtextResource>() {
  @Override
  public java.lang.Void exec(XtextResource archimodel) {
   // now we access the model
   for (EObject modelObject : archimodel.getContents()) {
    if (modelObject instanceof OurModelImpl) {
     … = … (OurModel) modelObject;
     …
   return null;
  }
 });
```

Model to Code with String Template



Filling a Template

Template

```
Hello, <name>
```

How to fill

```
ST hello = new ST("SimpleTemplate.stg");
hello.add("name", "World");
```

```
setGridSizeAndInitialize(x,z,ul,uh,uw) ::= <<...>>
createSquareObject(objectName, length, width) ::=
<<...>>
slice(name, object, portions) ::= <<...>>
alignObject(object) ::= <<...>>
turnIntoLegoBrick(object) ::= <<...>>
initializeHelperClasses() ::=
```

```
STGroup hello = new
STGroupFile("Template.stg");

ST st = group.getInstanceOf("alignObject");
// configuring the variable
st.add("object", …);
```

Filling a Template

Template

How to fill

```
main(states, initial, events, actions, tgroups) ::=
<<
...
<actions:action(); format="lower"; separator="\n">
...
>>

action(a) ::= <<...>>
```

```
Set<String> actions = …
…
ST main = group.getInstanceOf("main");
main.add("actions", actions);
…
```

- Templates can reference each other

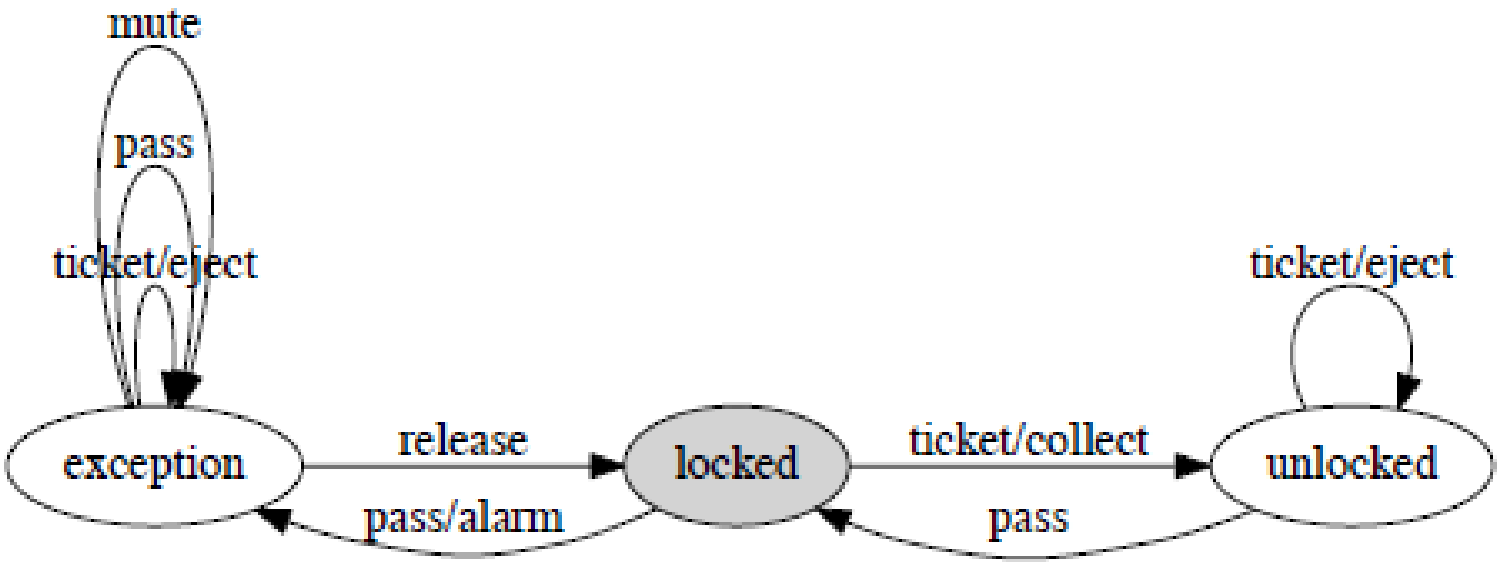
Filling a Template

Syntax	Description
tmpl(p) ::=“…”	Defines a template with parameter
tmpl(p) ::=<<…>>	Multiline template with indentation and linebreaks
tmpl(p) ::=<%…%>	Multiline template without indentation and linebreaks
<*attribute*>	Replaced with value of *attribute*.ToString() (or empty string if missing).
<*attribute.property*>	Replaced with value of *property* of *attribute* (or empty string if missing).
<*attribute*.(*expr*)>	Indirect property lookup. Same as *attribute.property* except value of *expr* is the property name.
<*multi-valued-attribute*; separator=*expr*>	Concatenation of ToString() invoked on each element separated by *expr*.
<*attribute*:*template*(*argument-list*)>	Template *application*.
<if(!*attribute*)>*subtemplate*<endif>	If *attribute* has no value or is a bool object that evaluates to false, include *subtemplate*. These conditionals may be nested.
<! comment !>, $! comment !$	Comments, ignored by StringTemplate.

For more, see: https://theantlrguy.atlassian.net/wiki/display/ST/Five+minute+Introduction

Quiz FSM (based on an example from Ralf Lämmel (1/5))

An FSM for a turnstile in a metro



- **States** (nodes): locked, unlocked, exception
- **Events**: ticket, pass, release, mute
- **Actions**: collect, eject, alarm
- **Transitions** (edges)

Quiz FSM (based on an example from Ralf Lämmel (2/5))

```
main(states, initial, events, actions, tgroups) ::= <<
enum State { <states; format="upper", separator=", "> };
enum State initial = <initial; format="upper">;
enum Event { <events; format="upper", separator=", "> };
<actions:action(); format="lower", separator="\n">
enum State next(enum State s, enum Event e) {
    switch(s) {
<tgroups:tgroup(); separator="\n">
        default: return UNDEFINED;
    }
}>>

action(a) ::= "void <a>() { }"

tgroup(g) ::= <<
        case <g.stateid; format="upper">:
            switch(e) {
                <g.ts:transition(); separator="\n">
                default: return UNDEFINED;
            }>>

transition(t) ::= <%
case <t.event; format="upper">:
<if(t.action)><t.action; format="lower">(); <endif>
return <t.target; format="upper">;%>
```

The templates for generating C code

Quiz FSM (based on an example from Ralf Lämmel (3/5))


```
import org.stringtemplate.v4.ST;
import org.stringtemplate.v4.STGroup;
import org.stringtemplate.v4.STGroupFile;
import org.stringtemplate.v4.StringRenderer;
import java.io.File;
import java.util.*;

public class FsmlCGenerator {
        private static class TGroup {
                public String stateid;
                public List<Transition> ts;
        }
        public static String generateC(Fsm fsm) {
                // Build list of states with extra "UNDEFINED"
                List<String> states = ...
                // Build set of events
                Set<String> events = ...
                // Build set of actions
                Set<String> actions = ...
                // Group transitions by state
                List<TGroup> tgroups = ...
                // Load template group and retrieve top-level template
                STGroup group = new STGroupFile(... + "Fsm.stg");
                group.registerRenderer(String.class, new StringRenderer());
                ST main = group.getInstanceOf("main");
                // Set template parameters and render
                main.add("states", states);
                main.add("initial", fsm.getInitial());
                main.add("events", events);
                main.add("actions", actions);
                main.add("tgroups", tgroups);
                return main.render();
        }
}
```

Auxiliary data structure

Translation of FSM

Actual template processing

Outline of the Java code for **program generation**

Quiz FSM (based on an example from Ralf Lämmel (4/5))


```
// Build list of states with extra "UNDEFINED"
List<String> states = new LinkedList<>();
for (State s : fsm.getStates()) states.add(s.getStateid());
states.add("UNDEFINED");
// Build set of events
Set<String> events = new HashSet<>();
for (Transition t : fsm.getTransitions()) events.add(t.getEvent());
// Build set of actions
Set<String> actions = new HashSet<>();
for (Transition t : fsm.getTransitions())
    if (t.getAction()!=null) actions.add(t.getAction());
// Group transitions by state
List<TGroup> tgroups = new LinkedList<>();
for (State s : fsm.getStates()) {
    TGroup tg = new TGroup();
    tg.stateid = s.getStateid();
    tg.ts = new LinkedList<>();
    for (Transition t : fsm.getTransitions())
        if (tg.stateid==t.getSource()) tg.ts.add(t);
    tgroups.add(tg);
}
```

Translation of FSM

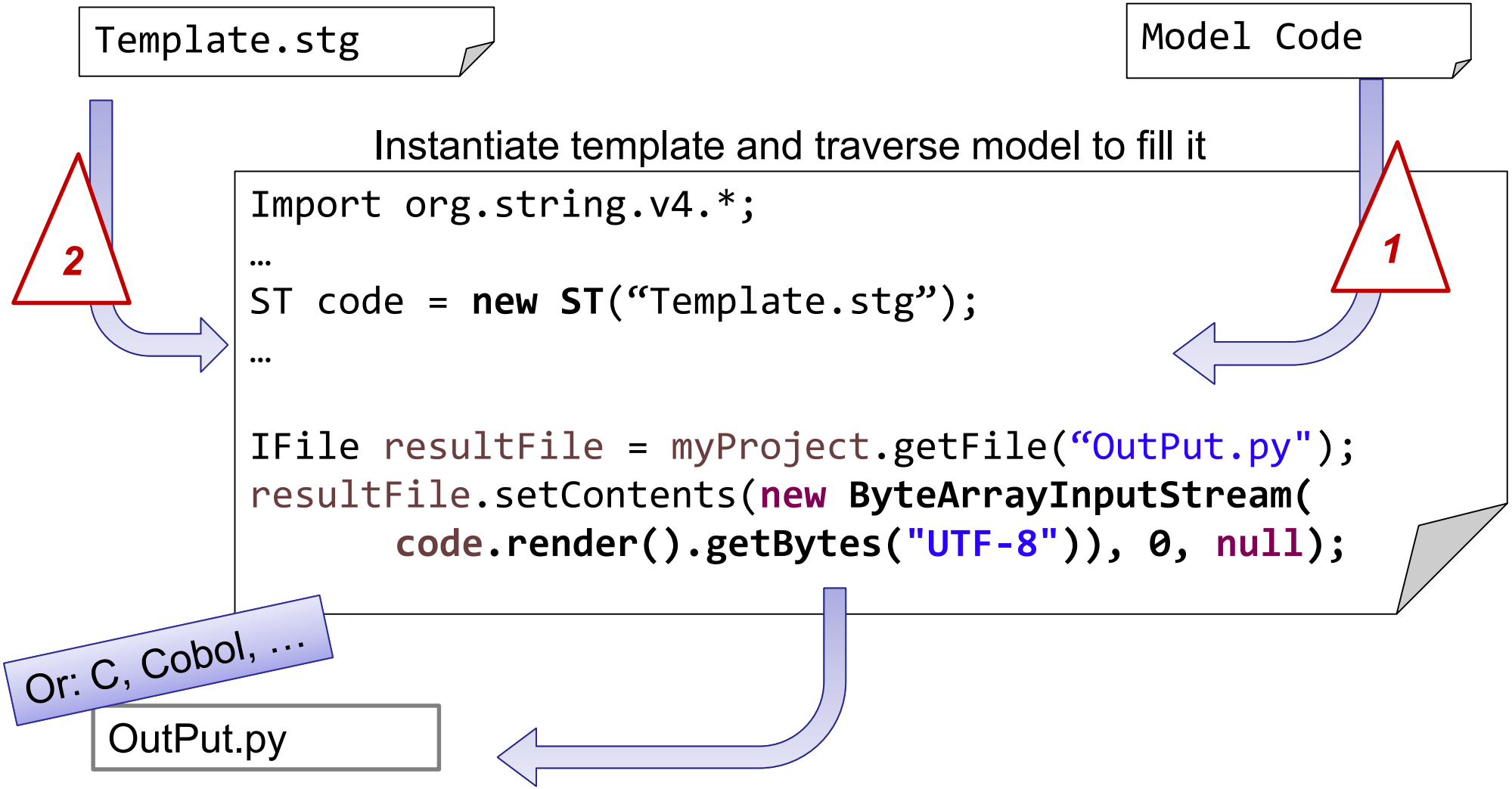
N.B.: We aim at a strict separation of model and view and thus, we do not try to 'compute' anything in the template.

Quiz FSM (based on an example from Ralf Lämmel (Solution 5/5))

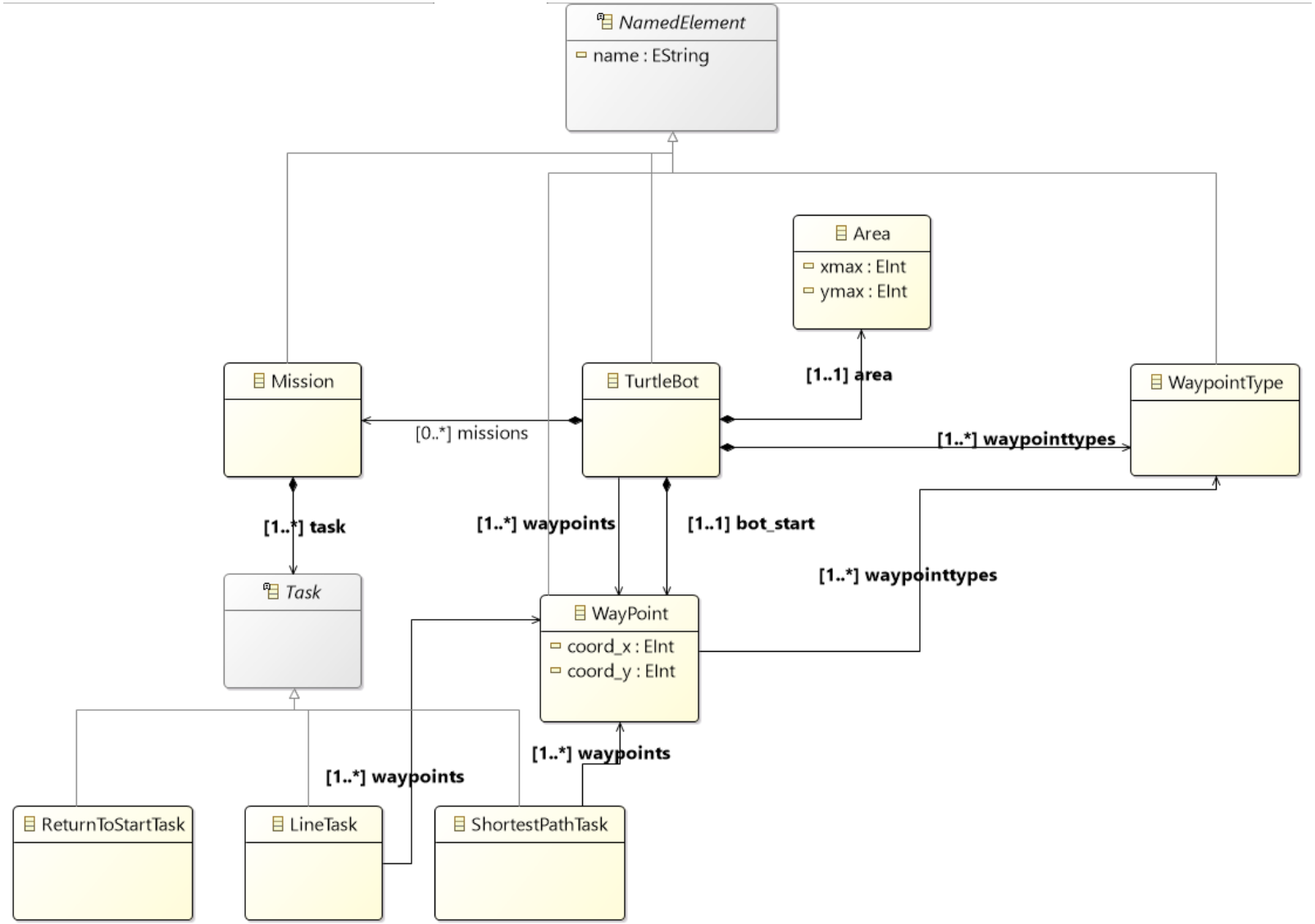

```
enum State { LOCKED, UNLOCKED, EXCEPTION, UNDEFINED };
enum State initial = LOCKED;
enum Event { TICKET, RELEASE, MUTE, PASS };
void alarm() { }
void eject() { }
void collect() { }
enum State next(enum State s, enum Event e) {
    switch(s) {
        case LOCKED:
            switch(e) {
                case TICKET: collect(); return UNLOCKED;
                case PASS: alarm(); return EXCEPTION;
                default: return UNDEFINED;
            }
        case UNLOCKED:
            switch(e) {
                case TICKET: eject(); return UNLOCKED;
                case PASS: return LOCKED;
                default: return UNDEFINED;
            }
        case EXCEPTION:
            switch(e) {
                case TICKET: eject(); return EXCEPTION;
                case PASS: return EXCEPTION;
                case MUTE: return EXCEPTION;
                case RELEASE: return LOCKED;
                default: return UNDEFINED;
            }
        default: return UNDEFINED;
    }
}
```

C code for the turnstile FSM

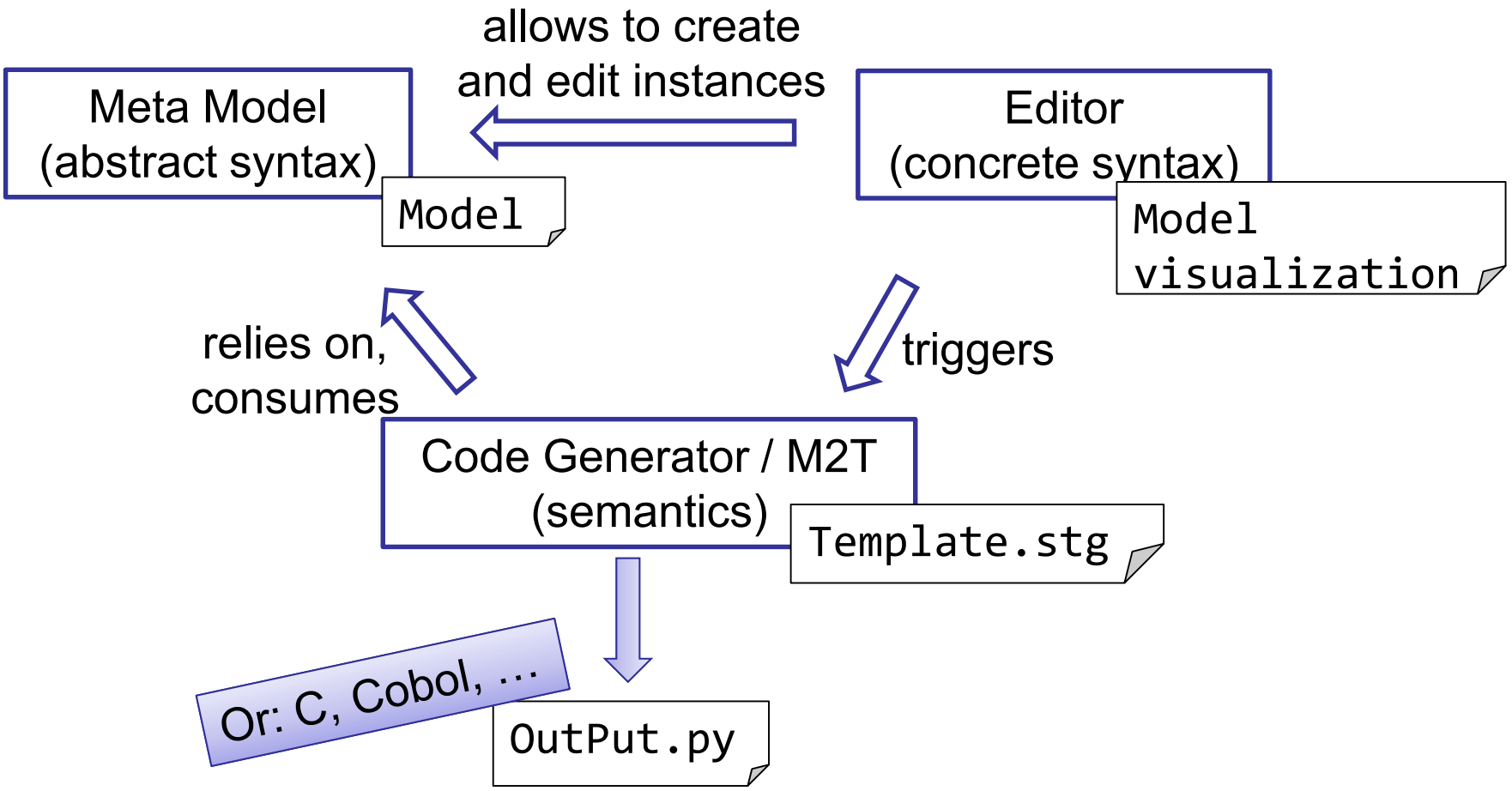
Model to Code with String Template



Assignment: Model-based Software Engineering



Wrap-up


allows to create
and edit instances
Meta Model
(abstract syntax)
Model
Editor
(concrete syntax)
Model
visualization
relies on,
consumes
triggers
Code Generator / M2T
(semantics)
Template.stg
Or: C, Cobol, …
OutPut.py

Links to Tutorials

- Eclipse/Modeling Framework:
  - An introductory video for eclipse (done by Grischa Liebel, who is one of our PhD students): https://www.youtube.com/watch?v=4yGGHzwPqpA
  - [Berger] Appendix C: Using the Eclipse Modeling Framework
  - http://www.vogella.com/tutorials/EclipseEMF/article.html
  - http://help.eclipse.org/kepler/index.jsp?topic=%2Forg.eclipse.emf.doc%2Ftutorials%2Fclibmod%2Fclibmod.html
- Xtext:
  - [Berger] Appendix E: Xtext in A Nutshell
  - https://eclipse.org/Xtext/documentation/101_five_minutes.html
  - https://eclipse.org/Xtext/documentation/102_domainmodelwalkthrough.html
- String templates:
  - An introduction to code generation with string templates by Ralf Lämmel https://www.youtube.com/watch?v=EWDVo3zRr1E&feature=youtu.be
  - https://theantlrguy.atlassian.net/wiki/display/ST/Five+minute+Introduction
  - https://github.com/antlr/stringtemplate4/blob/master/doc/index.md

[Berger] “Introduction to Model-Driven Software Engineering with Domain-Specific Languages”, Andrzej Wasowski and Thorsten Berger, Appendix E:Xtext in A Nutshell https://www.dropbox.com/s/mjaqcwmcpjq6mqb/mdsebook-draft-wasp.pdf?dl=0

Model-Driven Engineering (MDE)

Regina Hebig, Thorsten Berger