Evaluating Tool Support for Co-Evolution of Modeling Languages, Tools and Models

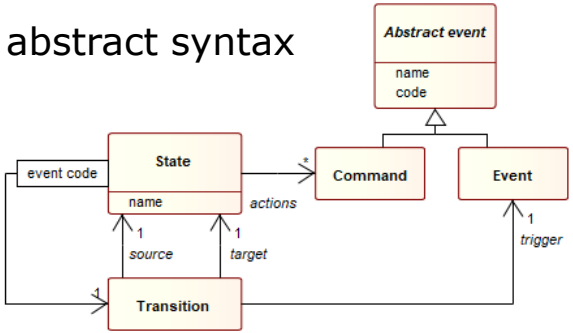
Juha-Pekka Tolvanen, Steven Kelly
jpt@metacase.com; stevek@metacase.com

Context: Co-Evolution

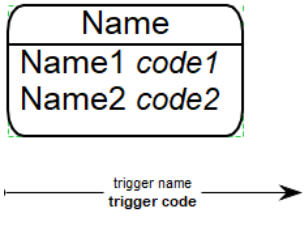
TOOL

Modeling language

abstract syntax



concrete syntax



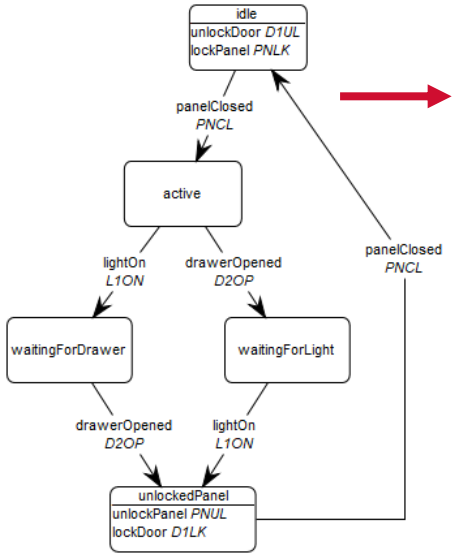
semantics

```
foreach .State; {
   do :Command {
      id;l 'State.addAction(' :Name 'Cmd);'
   }
   do ~Source>Transition {
      id;l 'State.addTransition('
      do :Trigger {:Name} ', '
      do ~Target.() {id}
      'State);' newline
   }
```

improve

co-evolve

Models generating code etc.




```
Event doorClosed = new Event("doorClose
Event drawerOpened = new Event("drawer
Event lightOn = new Event("lightOn", "L1C
Event panelClosed = new Event("panelClo

Command unlockDoorCmd = new Comma
Command lockPanelCmd = new Comman
Command unlockPanelCmd = new Comm
Command lockDoorCmd = new Command

State activeState = new State("active");
State idleState = new State("idle");
State unlockedPanelState = new State("u
State waitingForDrawerState = new State(
State waitingForLightState = new State("w

activeState.addTransition(drawerOpened,
activeState.addTransition(lightOn, waitingF

idleState.addAction(unlockDoorCmd);
idleState.addAction(lockPanelCmd);
idleState.addTransition(doorClosed, active

unlockedPanelState.addAction(unlockPan
unlockedPanelState.addAction(lockDoorC
unlockedPanelState.addTransition(panelC
```

Agenda

1. Context: Co-evolution
2. Current work and solutions
3. Research question
4. Evaluation framework
5. Validating framework
6. Conclusions and future work

Current work and solutions

- Evolution is identified as a major challenge with tools
  - 2nd most important feature: update models when language changes (1st: highlight elements and related error messages)
- Research emphasizes creating model transformations
  - Reported industry cases do not indicate their use
- Research on co-evolution has focused on metamodels and models, a narrow focus
- Few tool evaluations
  - Show that there is room for improvements
  - We applied them to MetaEdit+

	GMF [20]	**Sirius** [16]	**MetaEdit+**
1 add concrete class	x	x	o
2 add abstract class	x	o	o
3 insert superclass	o	x	o
4 delete class	x	x	o
5 rename class	x	x	o
6 add property	x	xo	o
7 delete property	x	x	o
8 rename property	x	x	o
9 move property	x	x	o
10 pull up property	x	o	o
11 change property type	x	xo	o

Research question

How to evaluate a tool's capabilities to support co-evolution of modeling languages, tools and models?

- Evaluation framework for holistic co-evolution
  - With easy to conduct evaluation method
  - With evaluation case
  - Applied it to show its viability

To be or not to be;
To deprecate or destroy?

- Where a language change reduces the set of legal models, it is rarely a good idea to adopt a strict formalist approach
  - e.g. deleting parts of models that no longer conform
- Non-conforming parts still contain info and earlier choices the modeler needs to use during model co-evolution
- Leave them: were legal when made, still generate correctly
- **Deprecate:** Allow old style but show warnings, guidance
- Follow experience with programming languages & libraries
  - Make co-evolution fully automatic if certain
  - Otherwise deprecate, provide update help

Evaluation framework: 4 aspects

2 *Location of Change* ↓	1 *Nature of Change*			
	Add	**Rename**	**Remove**	**Change**
Metamodel	1	4	7	10
Constraints	2	5	8	11
Notation	3	6	9	12

3 Location adversely impacted

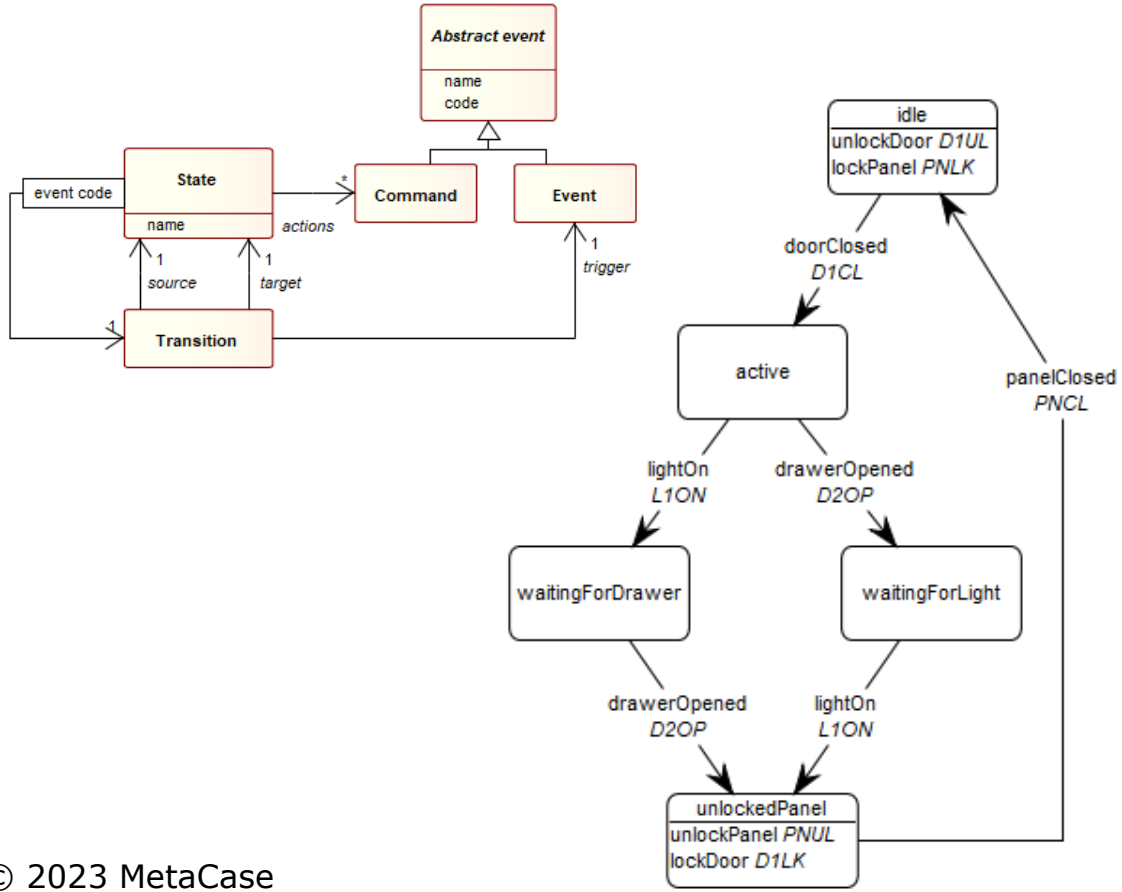
- Metamodel, Constraints, Notation
- Generators, Tool, Models

4 Scale for scoring co-evolution:

1. When creating a new artifact, editor **does not open or gives errors**
2. Editor opens **without functionality**
3. Editor allows creating a new artifact but **support for viewing and editing earlier artifacts is incomplete**
4. Editor **opens and asks for human intervention** to finalize co-evolution

   4½ if existing models behave and generate, and deprecation guidance is provided where needed
5. Editor **opens** with **fully co-evolved** earlier artifacts

A case

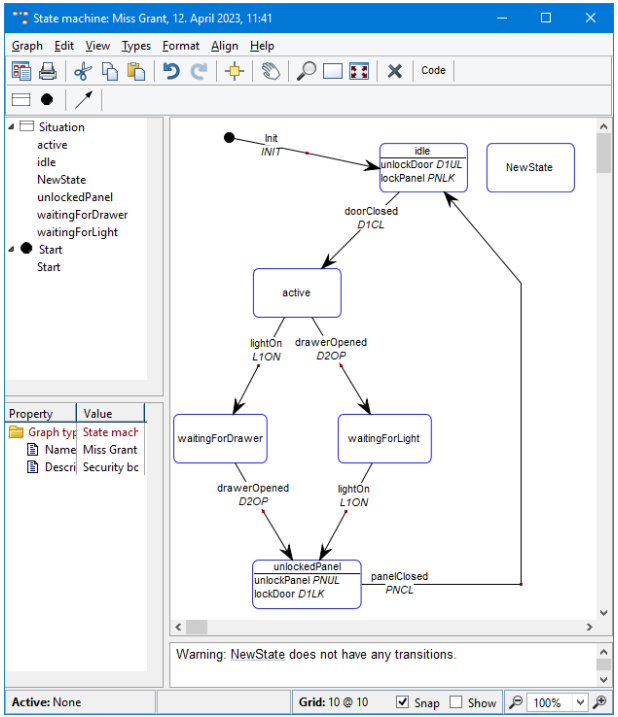


12 scenarios, evolutionary

1. Add Reset to metamodel
2. Add constraint: Only one Reset and connection to one State
3. Add notation for Reset
4. Rename State to Situation
5. Rename constraint
6. Rename symbol from notation
7. Remove Reset from metamodel
8. Remove constraint: Reset is not allowed to relate to Situation
9. Remove Reset's notation
10. Change Transition's Event property to Source role
11. Change constraint from Reset to new Start
12. Change notation refer to refer another symbol

Validating the framework

- Evaluated MetaEdit+
  - Commercial Language Workbench from MetaCase
- Implemented all 12 scenarios
  - Took 32 minutes
  - Does not require a lot of effort
- One person made the changes, other checked their correctness
  - All done in a single-user version



Each scenario available at https://github.com/mccjpt/Gothic

Evaluation results

Location of Change ↓	*Nature of Change*			
	Add	**Rename**	**Remove**	**Change**
Metamodel	1. **5** *555|555*	4. **4** *555|455*	7. **4½** *555|554½*	10. **4½** *555|554½*
Constraints	2. **4½** *555|554½*	5. —	8. **4½** *555|554½*	11. **5** *555|555*
Notation	3. **5** *555|555*	6. **5** *555|555*	9. **5** *555|555*	12. **5** *555|555*

Scores:

metamodel, constraints, notation | generator, tool, model

1. Add Reset to metamodel
2. Add constraint: Only one Reset and connection to one State
3. Add notation for Reset
4. Rename State to Situation
5. Rename constraint
6. Rename symbol from notation
7. Remove Reset from metamodel
8. Remove constraint: Reset is not allowed to relate to Situation
9. Remove Reset's notation
10. Change Transition's Event property to Source role
11. Change constraint from Reset to new Start
12. Change notation refer to refer another symbol

Summary

- Holistic framework that covers co-evolution widely
- Works well even where fully automatic is not possible
  - Deprecation approach is favored in industrial use
- Framework is viable
  - Others can repeat and validate the evaluation made
  - Other tools can be evaluated similarly
- Future work: extending what is evaluated
  - Collaboration with many people
    - both metamodeling and modeling
  - Scalability: large models, language(s), multiple users
  - Tool versions and meta-metamodel versions

Thank you

Questions?

Comments?

Counter-arguments?

Experiences?

Custom updates with MetaEdit+ API

Scenario 10: move Trigger from relationship to role

```
METype graphType      = new METype() { name = "State machine" };
METype transitionType = new METype() { name = "Transition" };
METype sourceRoleType = new METype() { name = "Source" };

MetaEditAPIPortTypeClient api = new MetaEditAPIPortTypeClient();

foreach (MEOop graph in api.allGoodInstances(graphType))
{
   foreach (MEOop transition in api.contentsMatchingType(graph, transitionType))
   {
      MEOop[] sources = api.rolesForRel(graph, transition, sourceRoleType);
      MEAny trigger = api.valueForLocalName(transition, "Trigger");
      api.setValueForLocalName(sources[0], "Trigger", trigger);
   }
}
```