SOLVING THE FAMILIES TO PERSONS CASE USING EVL+STRACE

Model-Driven Software Engineering Research Group

Department of Software Engineering
University of Isfahan
Leila Samimi-Dehkordi
samimi@eng.ui.ac.ir
Bahman Zamani
zamani@eng.ui.ac.ir
Shekoufeh Kolahdouz-Rahimi
sh.rahimi@eng.ui.ac.ir

SOLUTION OVERVIEW

EVL+Strace

Epsilon Validation Language (EVL)

\+

Specific trace metamodel (Strace)

Source Metamodel

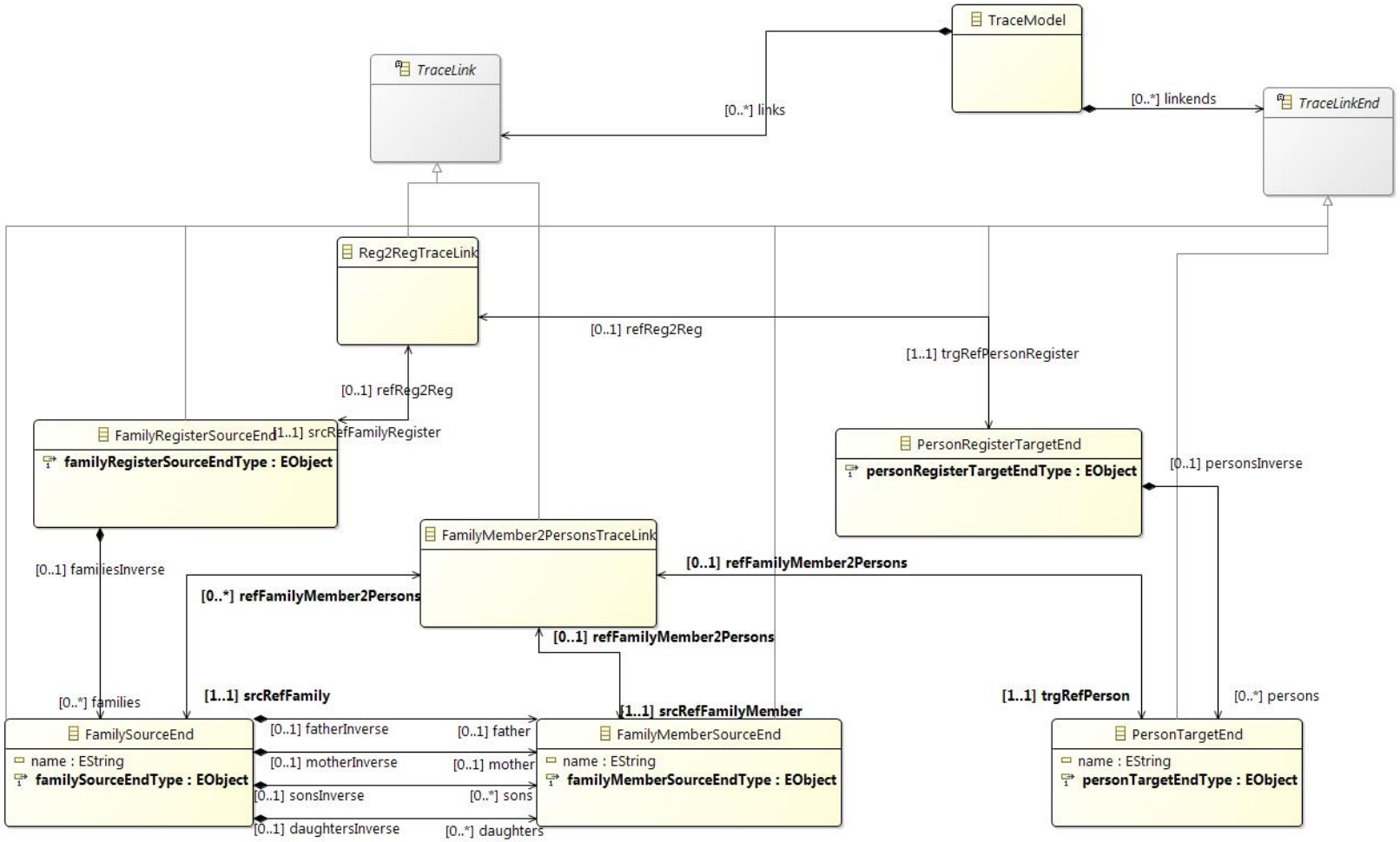
Target Metamodel

http://lsamimi.ir/EVLStrace.htm

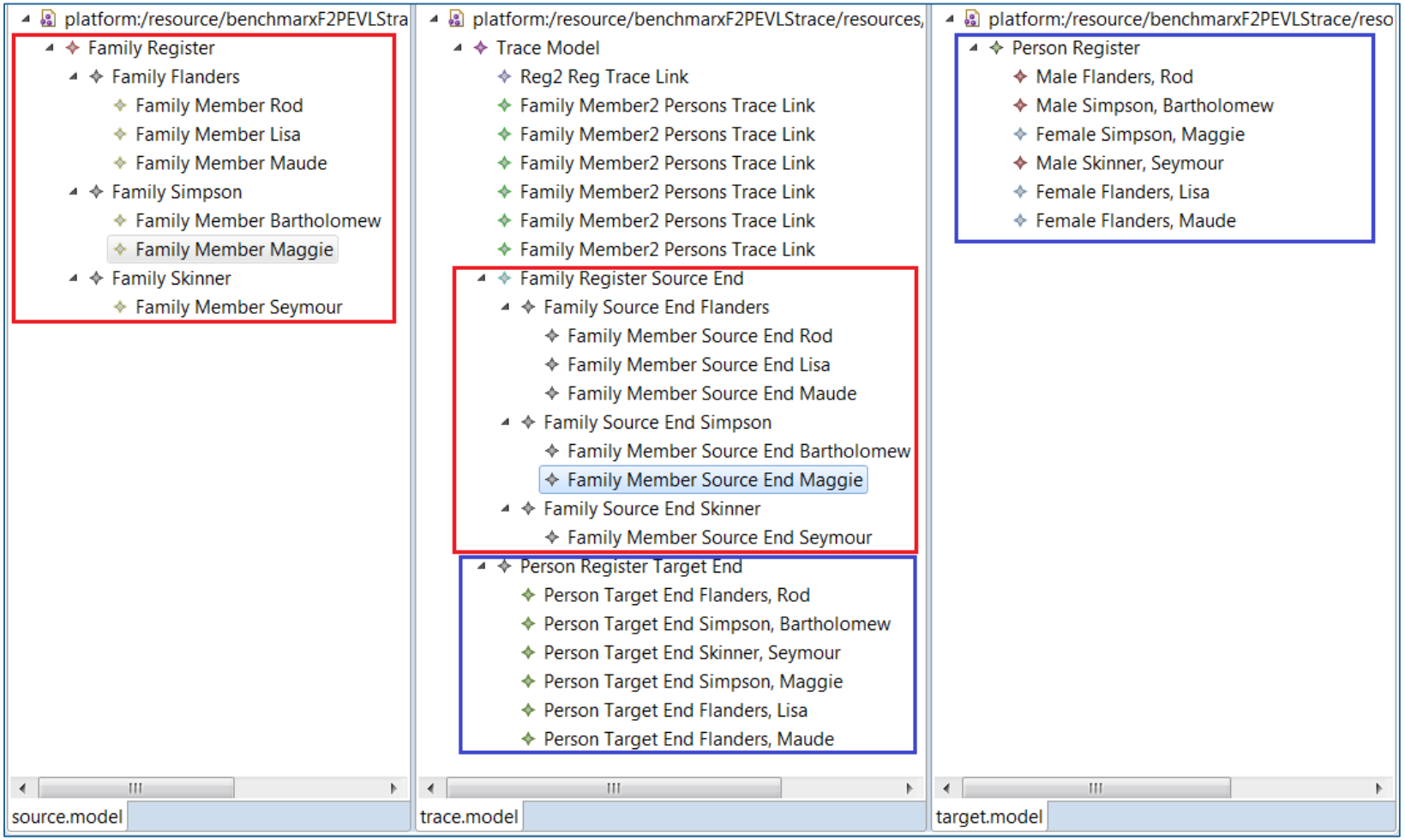
EVL+Strace

- EVL expresses constraints between heterogeneous models and evaluates them to resolve the occurred violations.
- An EVL constraint contains two main parts including **check** and **fix** blocks.
- The Epsilon Object Language (EOL) specifies the checking expressions and fixing statements.
- The defined constraints in EVL+Strace are applied on the elements of three metamodels including source, specific trace, and target.
- A specific trace metamodel consists of strongly typed trace links.
- EVL+Strace can recognize four atomic updates including element deletion, addition, relocation, and value modification.

SPECIFIC TRACE METAMODEL (SOLUTION)



THREE CONSISTENT MODELS



TRANSFORMATION CODE

```
1  context Families2Persons!FamilyMemberSourceEnd{
2    guard: not self.isRemoved() and not self.refFamilyMember2Persons.endTypeIsRemoved()
3    constraint nameIsModified{
4      check: not self.nameIsModified()
5      message: 'name of '+self +' is modified'
6      fix{
7        title:'Propagate the modification'
8        do{ self.namePropagates();}
9      }}}
```

TRANSFORMATION CODE

context

```
1  context Families2Persons!FamilyMemberSourceEnd{
2    guard: not self.isRemoved() and not self.refFamilyMember2Persons.endTypeIsRemoved()
3    constraint nameIsModified{
4      check: not self.nameIsModified()
5      message: 'name of '+self +' is modified'
6      fix{
7        title:'Propagate the modification'
8        do{ self.namePropagates();}
9      }}}
```

Conditions for guard

Conditions for check block

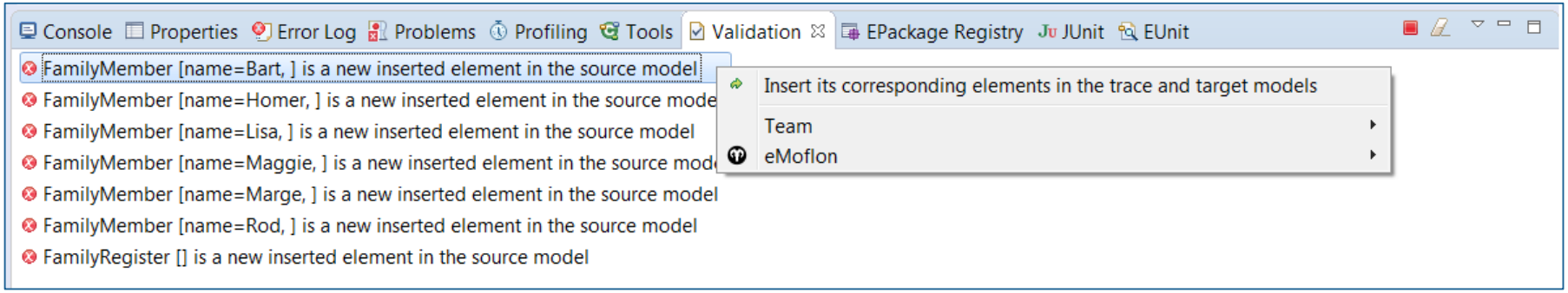
Message shows a modification is occurred

Statements for fix block

Constraint for recognizing modification

EOL operation propagate the recognized modification

EXECUTION (INTERACTIVE FIXING)



INTERACTIVE VS. AUTOMATIC EVL+STRACE

```
constraint Foo {

    check : <some expression>

}
```

```
constraint Foo {

    check {
        var condition = <some expression>;
        if (condition == false) {
            // Code that automatically fixes the problem
        }
        return true;
    }
}
```

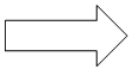
http://lsamimi.ir/MoDEBiTE.htm

INTERACTIVE VS. AUTOMATIC EVL+STRACE

```
constraint Foo {

    check : <some expression>

}
```



```
constraint Foo {

    check {
        var condition = <some expression>;
        if (condition == false) {
            // Code that automatically fixes the problem
        }
        return true;
    }
}
```

- To test the approach, we use auto-fix code for EVL+Strace
- EUnit and Workflow tools of the Epsilon framework are used for testing.
- It is possible to automatically produce the trace metamodel and some parts of transformation code with the use of our toolkit, **MoDEBiTE**.

http://lsamimi.ir/MoDEBiTE.htm

RESULTS

#	direction	Policy	Change Type	Test Case Name	Result
1	fwd	fixed	-	testInitialiseSynchronisation	expected pass
2	fwd	fixed	attribute	testFamilyNameChangeOfEmpty	expected pass
3	fwd	fixed	add	testCreateFamily	expected pass
4	fwd	fixed	add	testCreateFamilyMember	expected pass
5	fwd	fixed	add	testNewFamilyWithMultiMembers	expected pass
6	fwd	fixed	add	testNewDuplicateFamilyNames	expected pass
7	fwd	fixed	add	testDuplicateFamilyMemberNames	expected pass
8	bwd	runtime	add $(e \wedge p)$	testCreateMalePersonAsSon	expected pass
9	bwd	runtime	add $(e \wedge p)$	testCreateMembersInExistingFamilyAsParents	expected pass
10	bwd	runtime	add $(e \wedge \neg p)$	testCreateMalePersonAsSon	expected pass
11	bwd	runtime	add $(e \wedge \neg p)$	testCreateMembersInExistingFamilyAsParents	expected pass
12	bwd	runtime	add $(e \wedge p)$	testCreateDuplicateMembersInExistingFamilyAsChildren	expected pass
13	bwd	runtime	add $(\neg e \wedge p)$	testCreateMalePersonAsParent	expected pass
14	bwd	runtime	add $(\neg e \wedge p)$	testCreateMembersInNewFamilyAsParents	expected pass
15	bwd	runtime	add $(\neg e \wedge p)$	testCreateDuplicateMembersInNewFamilyAsParents	expected pass
16	bwd	runtime	add $(\neg e \wedge \neg p)$	testCreateMalePersonAsSon	expected pass
17	bwd	runtime	add $(\neg e \wedge \neg p)$	testCreateFamilyMembersInNewFamilyAsChildren	expected pass
18	bwd	runtime	add $(\neg e \wedge \neg p)$	testCreateDuplicateFamilyMembersInNewFamilyAsChildren	expected pass
19	fwd	fixed	add	testIncrementalInserts	expected pass*
20	fwd	runtime	del	testIncrementalDeletions	expected pass*
21	fwd	fixed	attribute	testIncrementalRename	expected pass*
22	fwd	fixed	move	testIncrementalMove	expected pass*
23	fwd	fixed	add+del	testIncrementalMixed	expected pass*
24	fwd	fixed	move	testIncrementalMoveRoleChange	expected pass*
25	fwd	fixed	-	testStability	expected pass
26	fwd	fixed	-	testHippocraticness	expected pass
27	bwd	fixed	add	testIncrementalInsertsFixedConfig	expected pass
28	bwd	runtime	add	testIncrementalInsertsDynamicConfig	expected pass
29	bwd	runtime	del	testIncrementalDeletions	failure
30	bwd	runtime	attribute	testIncrementalRenamingDynamic	expected pass
31	bwd	runtime	del+add	testIncrementalMixedDynamic	failure
32	bwd	runtime	add	testIncrementalOperational	expected pass
33	bwd	runtime	-	testStability	expected pass
34	bwd	runtime	-	testHippocraticness	expected pass

CONCLUSION

- A bidirectional model-to-model transformation solution to the TTC 2017 Families to Persons case study based on a novel approach named EVL+Strace.
- The trace metamodel (correspondence metamodel) is specific to the domains of the Families and Persons case studies.
- The approach defines constraints to check user updates with the use of EVL.
- It is possible to program more than one fixing ways, and interactively ask user to restore the consistency.
- To test the solution, we change the constraints to fix the violations automatically.
- The evaluation presents that from all 34 test cases, automatic EVL+Strace has 32 expected pass and two failures.

http://lsamimi.ir/EVLStrace.htm