OOASP: Connecting Object-oriented and Logic Programming

Andreas Falkner and Gottfried Schenner,
Siemens AG Österreich, Austria

Anna Ryabokon and Kostyantyn Shchekotykhin,
Alpen-Adria-Universität Klagenfurt, Austria

funded by FFG and KWF (grants 840242 and 3520/26767/38701)

Outline

- Motivation
- ASP and development of object-oriented software
- Object-oriented ASP
- OOASP Domain Description Language
- Overview of reasoning tasks
- Summary & future work

Motivation

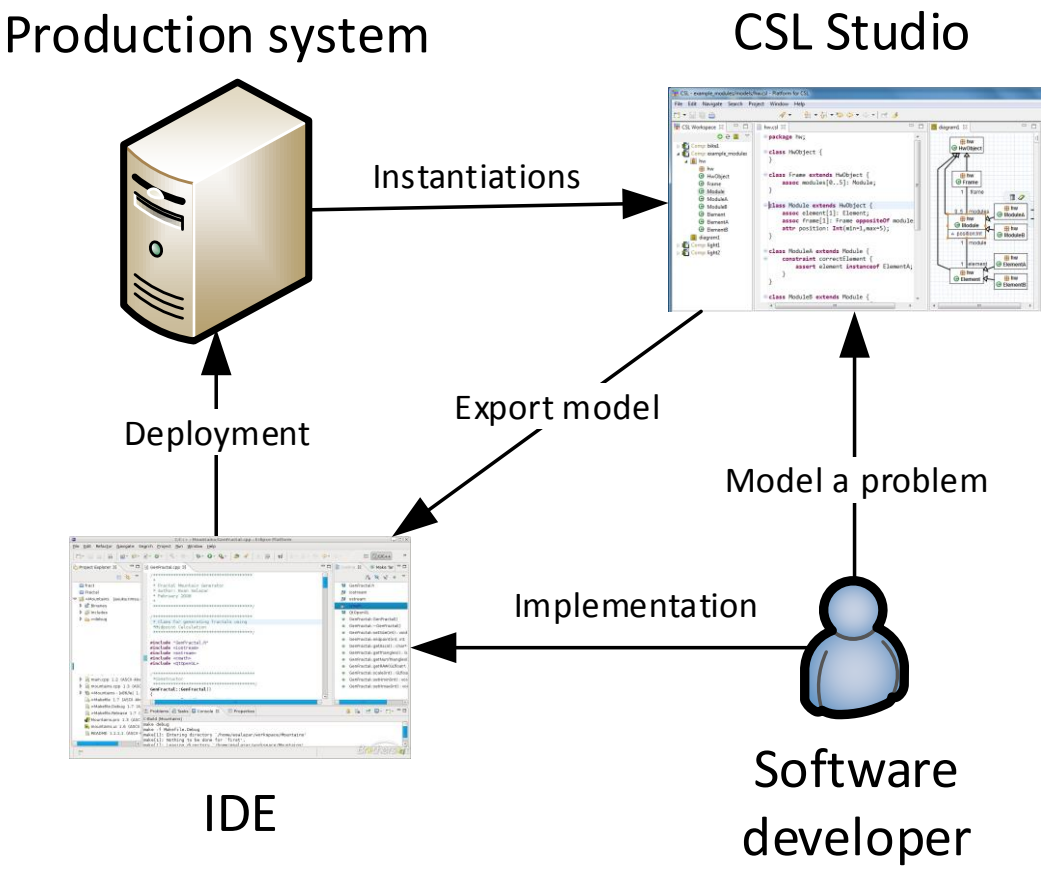
- Success of many companies depends on software systems solving complex combinatorial problems
- Development and maintenance of such software is a tedious and error-prone process
- ASP can often solve small and medium sized problems in acceptable time
- **Goal**: use ASP to reason about correctness of object-oriented models and their instantiations

Development of configurators

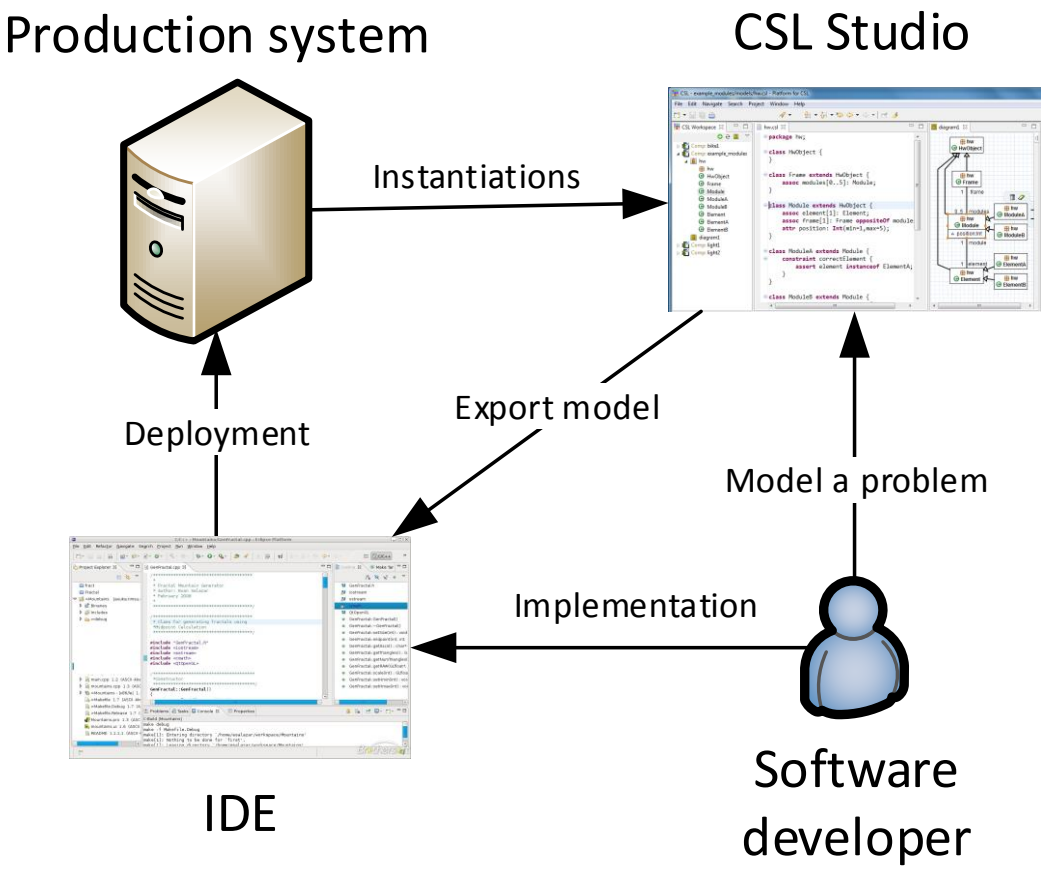
ALPEN-ADRIA UNIVERSITÄT
KLAGENFURT | WIEN GRAZ

- Configuration is an important problem of designing an artifact from a set of given components
- Siemens CSL-Studio is designed to simplify modeling and implementation of configurators

Development of configurators



Development of configurators

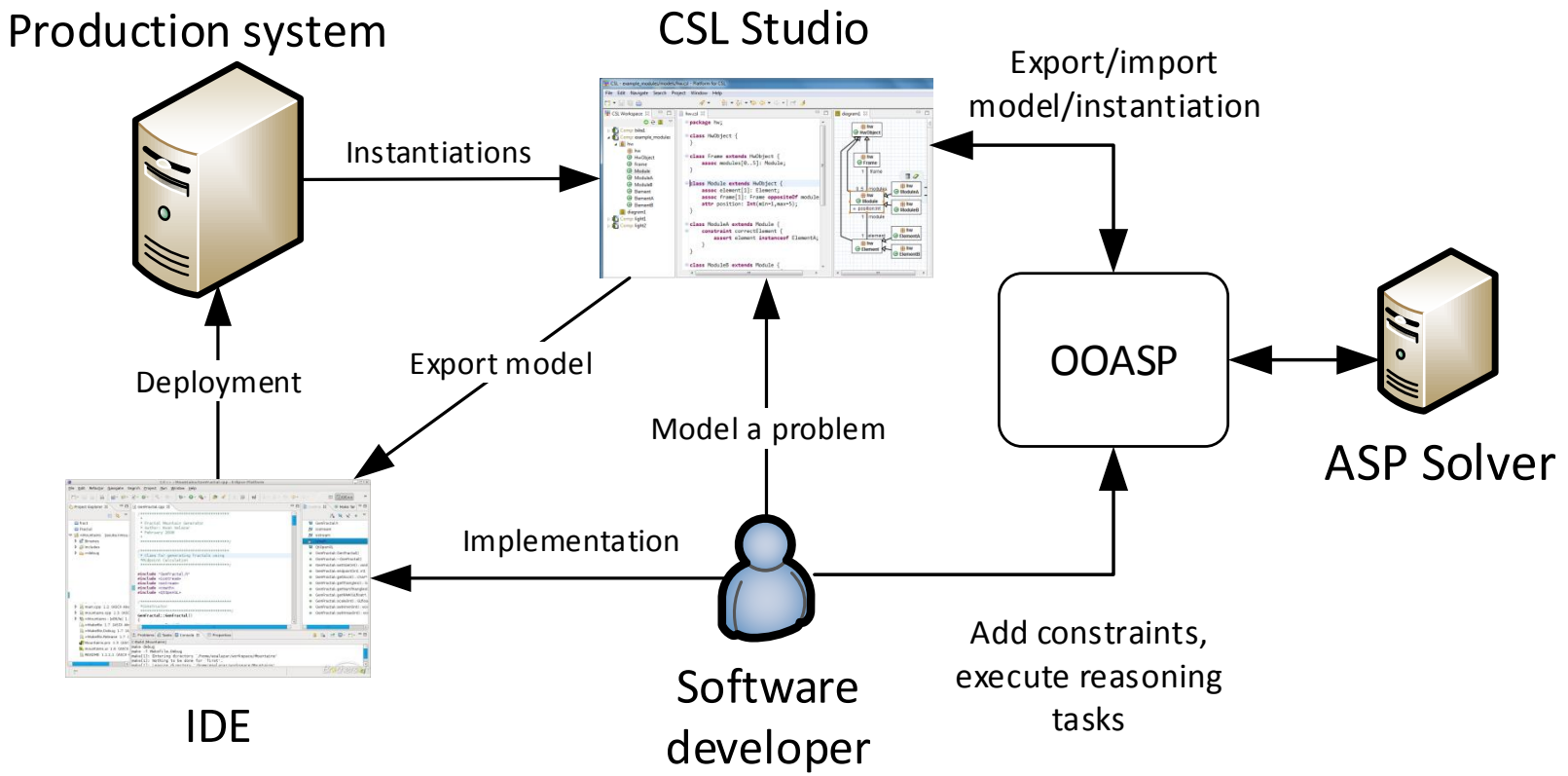


How can we support the software developer during design and testing of configurators?

ASP & configuration

- ASP solves small and medium-size problems well:
  - Partner Units Problem *[Aschinger et al. 2011]*
  - House (Rack) problem *[Friedrich et al. 2011, Aschinger et al. 2012]*
- Specific methods are required for large industrial problem instances [Teppan et al. 2012], [Ryabokon et al. 2013]
- Development and testing is often done on small examples and ASP can be used to validate the software *[Schanda and Brain 2012, Falkner et al. 2012]*

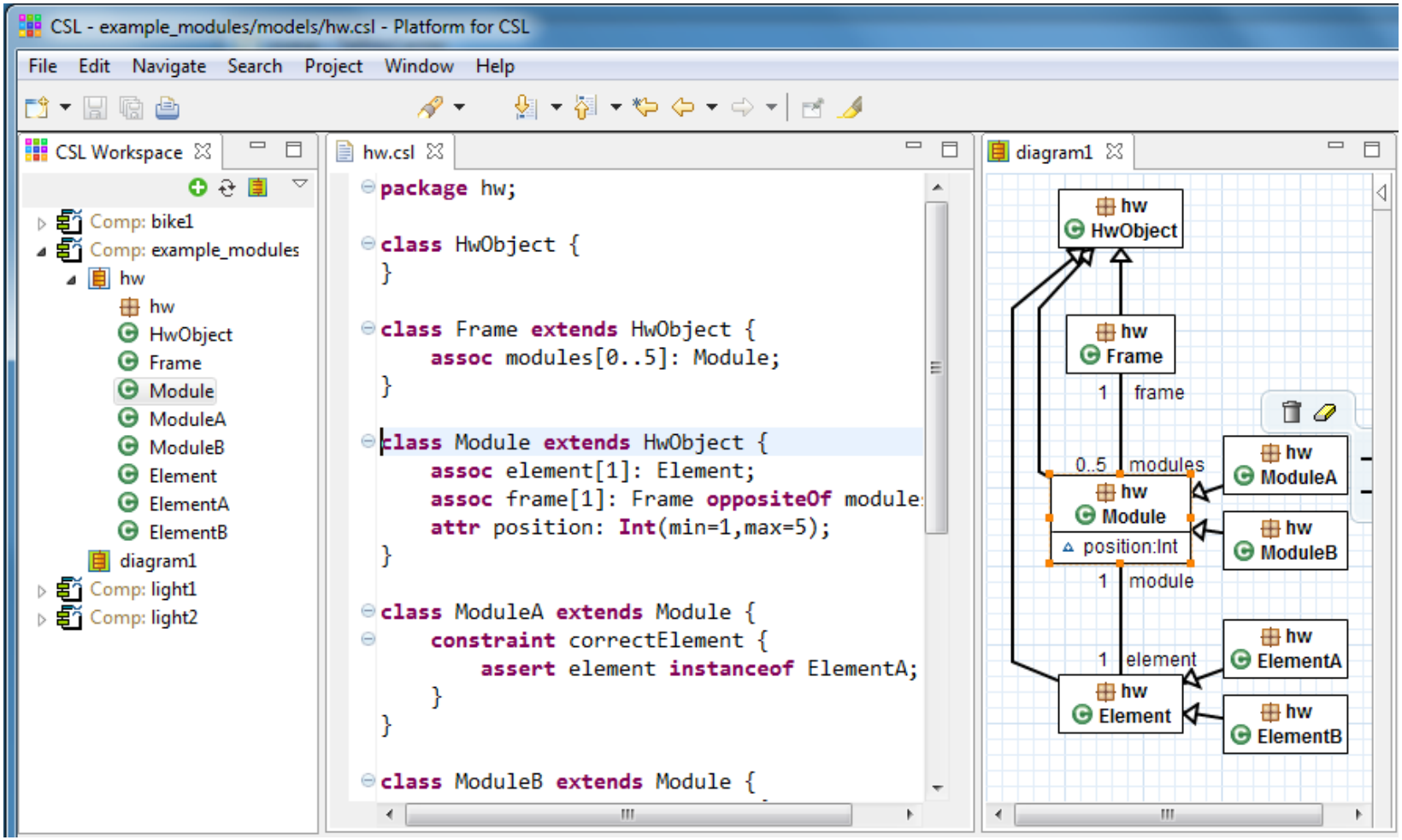
OOASP integration



OOASP-DDL

- Domain Description Language allows encoding of models and instances of configuration problems *[Dhungana et al. 2013]*
  - Multiple configuration models in one workspace
  - “Is a” hierarchy of classes within each model
  - Definition of attributes
  - Association relations with cardinality restrictions
- Experimental OOASP integration
  - Models can be exported from CSL Studio
  - CSL Studio can import problem instances encoded in OOASP-DDL

Configurator – CSL Studio



Exported model (fragment)

```
ooasp_class("v1","HwObject").
ooasp_class("v1",“Frame").
ooasp_class("v1","Module").
ooasp_class("v1","ModuleA").

ooasp_subclass("v1","Module","HwObject").
ooasp_subclass("v1","ModuleA","Module").

ooasp_assoc("v1","Frame_modules","Frame",1,1,
                                 "Module",0,5).

ooasp_attribute("v1","Module","position",
                                  "integer").
```

Model instantiations

- Instances of models are used to
  - save inputs to configuration problems
  - represent test cases for a developed configurator
  - show configuration solutions
- Instantiations saved in CSL Studio can be represented in OOASP-DDL

```
ooasp_instantiation("v1","c1").
ooasp_isa("c1", “FrameA", f10).
ooasp_isa("c1", "ModuleA", m11).
ooasp_associated("c1", "Frame_module", f10, m11).
ooasp_attribute_value("c1", “position", m11, 1).
```

Integrity constraints

- Integrity constraints are implemented in OOASP-DDL and Configurator software separately
- Diverse Redundancy – constraints are implemented manually
- Sample integrity constraint:
  - Elements of type *ElementA* require a module of type *ModuleA*

```
ooasp_cv(I,module_element_violated(M1,E1)) :-
        ooasp_instantiation(M,I),
        ooasp_associated(I,"Element_module",M1,E1),
        ooasp_isa(I,"ElementA",E1),
        not ooasp_isa(I,"ModuleA", M1).
```

OOASP framework

- OOASP uses ASP to reason about models and their instantiations
- Reasoning tasks supported by current implementation:
  - Validation of an object-oriented model and its instantiations
  - Completion of instantiations
  - Reconciliation of legacy models and their instances
- Implementation is done using meta-programming approach
- Some of the reasoning tasks, like reconciliation, can be implemented using modern ASP debuggers

Validation of a configuration

- Allows a developer to verify whether a CSL model and/or its instantiation is valid
- CSL Studio communicates with OOASP and shows the violated constraints
- Example:

```
ooasp_instantiation("v1","c1").
ooasp_isa("c1", “FrameA", f10).
ooasp_isa("c1", "ModuleA", m11).
ooasp_associated("c1", "Frame_module", f10, m11).
ooasp_attribute_value("c1", “position", m11, 1).
```

- OOASP returns:

```
ooasp cv("c1",mincardviolated(m11,“frame_module"))
```

Completion of an instantiation

- Solves two types of problems:
  1. invalid partial instantiation
     - model designed in the CSL Studio is inconsistent
     - system returned a partial instantiation that is faulty
  2. incomplete partial instantiation
- Example:

```
ooasp_instantiation("v1","c1").
ooasp_isa("c1", “FrameA", f10).
ooasp_isa("c1", "ModuleA", m11).
ooasp_associated("c1", "Frame_module", f10, m11).
ooasp_attribute_value("c1", “position", m11, 1).
```

Reconciliation I

- Goal is to restore consistency of an inconsistent (partial) instantiation given as an input
- Application scenarios:
  - an instantiation is inconsistent;
  - a model is consistent, but the given partial instantiation cannot be extended; and
  - the model is changed due to new requirements to a configurable product
- Convert OOASP-DDL into a reified form:

```
fact(ooasp(t)) :- ooasp(t).
```

Reconciliation II

- Guess the set of changes required to obtain a consistent instance

```
1{reuse(ooasp(t)), delete(ooasp(t))}1 :-
                                    fact(ooasp(t)).

ooasp(t) :- reuse(ooasp(t))
```

- A preferred solution can be found if the costs of reuse/delete actions are known

Summary

- OOASP simplifies development of the object-oriented configurators
- Three reasoning tasks are sufficient to cover most of the developer’s needs
- OOASP can be easily extended for further tasks and model types
- Experimental integration with CSL Studio showed a number of encouraging results

Future work

ALPEN-ADRIA UNIVERSITÄT
KLAGENFURT | WIEN GRAZ

The main points to be solved prior to commercial use:

- Manual maintenance of object ids too complicated, must be generated on demand [Stumptner et al. 1998]
- No automated support for computation on a lower/upper bounds of objects for an instantiation [Feinerer 2013]
- Currently no support for the integration of heuristics and symmetry breaking approaches [Gebser et al. 2013, Drescher et al. 2011]
- Performance of the meta-programming approach is limited

Thank you! Questions?

© Juha Tiihonen


© Teesaa

Corporate Technology
Research Group
Configuration Technologies
andreas.a.falkner@siemens.com
gottfried.schenner@siemens.com

Applied Informatics
Intelligent Systems and
Business Informatics Group
anna.ryabokon@aau.at
konstantin.schekotihin@aau.at