



Jasper Denkers
Delft University of Technology
Delft, The Netherlands
j.denkers@tudelft.nl



Louis van Gool
Océ Technologies B.V.
Venlo, The Netherlands
louis.vangool@oce.com



Eelco Visser
Delft University of Technology
Delft, The Netherlands
e.visser@tudelft.nl

Custom DSL Implementations

An error occurred during the generation of this view:

TYPE ERROR: Expression
0
has type integer instead of boolean

Type error, no traceability

Context: Océ develops and uses domain-specific languages (DSLs) for model-based development.

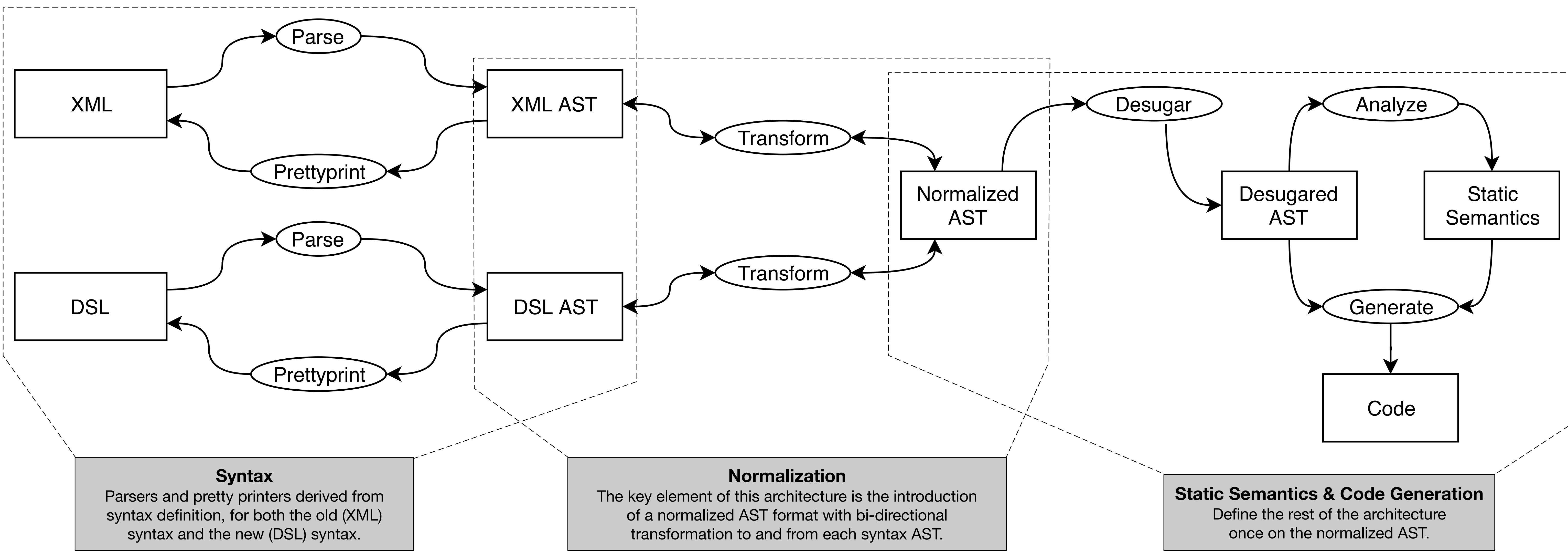
Problem: custom implementations using conventional technologies (XML for syntax, Python for static analysis and code generation) are flexible, but:

- ✗ No IDE support
- ✗ Hard to implement advanced language features
- ✗ Concise syntax is missing
- ✗ No traceable error reporting

Solution: migrate to a language workbench.

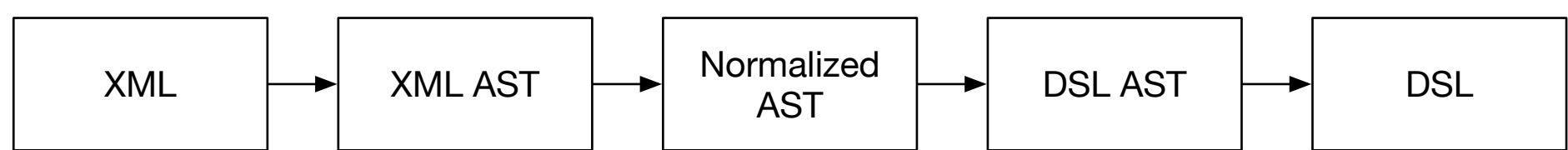
Extra requirement: automate forward and backward migration.

DSL Migration Architecture



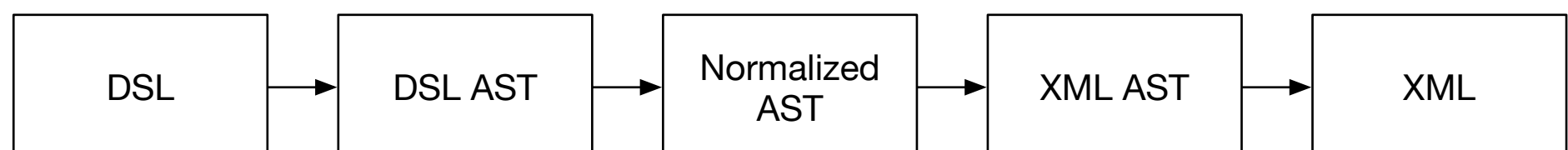
Forward Migration

For porting existing programs written in the old XML syntax to the new DSL syntax.



Backward Migration

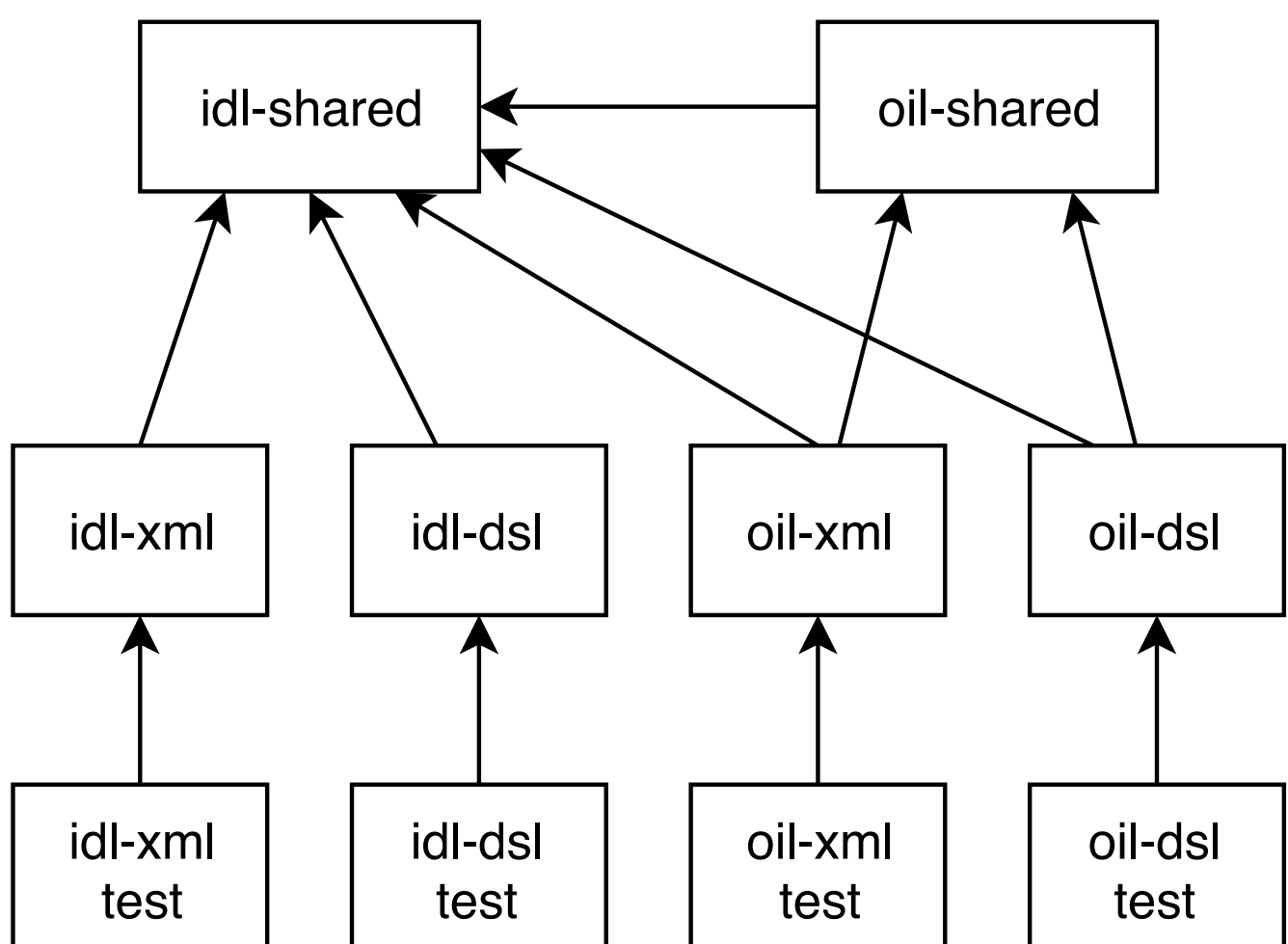
For programs written in the new DSL to be backward compatible with existing tooling still based on the XML syntax.



Modular Language Definition

Inter-language: XML and DSL variants of the languages share syntax and static semantics definition.

Intra-language: OIL re-uses syntax, static semantics, and transformations from IDL.



Implementation in the Spoofax Language Workbench

- ✓ Concise DSL syntax
- ✓ Traceable/interactive IDE feedback
- ✓ Cross-language reference resolution
- ✓ Forward/backward migration

workspace - oil.example/sle18/example.oilxml - Eclipse

printer.idl.xml

```
1 <module name="printer">
2   <interface name="printer">
3     <method name="turn_on"/>
4     <method name="turn_off"/>
5     <method name="add_job"/>
6     <method name="remove_job"/>
7     <method name="print_job"/>
8   </interface>
9 </module>
```

example.oilxml

```
1 <oil>
2   <region name="status">
3     <state name="off"/>
4     <state name="on"/>
5     <state name="ok"/>
6   </region>
7   <scope name="illegal">
8     <invariant>0</invariant>
9   </scope>
10  <group type="ca
11    <transition mType: number
12    <transition method="turn_off" source="on" target="off"/>
13    <transition method="add_job" source="on" target="ok"/>
14    <transition method="remove_job" source="ok" target="on"/>
15    <transition method="print_job" source="ok" target="ok"/>
16    <transition method="print_job" source="on" target="illegal"/>
17  </group>
18 </oil>
```

Type mismatch: invariant must be boolean, got number

printer.idl

```
1 module printer {
2   interface printer {
3     turn_on()
4     turn_off()
5     add_job()
6     remove_job()
7     print_job()
8   }
9 }

10 on [call printer.printer {
11   in off on turn_on() go on end
12   in on on turn_off() go off end
13   in ok on add_job() go ok end
14   in ok on remove_job() go ok end
15   in on on print_job() go illegal end
16 }
17
18
```