Towards a coordinated approach to software transformation

Anthony Sloane and Shirley Goldrei

Programming Languages Research Group
Department of Computing, Macquarie University
Sydney, Australia

http://www.comp.mq.edu.au/plrg/
Our Thesis

STS focus: "architecture, reuse, implementation (data representations and algorithms), application models and benchmarks".

Our slant: generation of software transformation systems from high-level specifications.

<table>
<thead>
<tr>
<th>Formalism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification notations</td>
</tr>
<tr>
<td>Generation algorithms</td>
</tr>
<tr>
<td>Generated components</td>
</tr>
<tr>
<td>Infrastructure</td>
</tr>
</tbody>
</table>

For the STS field to advance most effectively, we need a coordinated approach to these different levels.
Are we talking to ourselves?

Do we just do our own thing?

Or can we at least share infrastructure?
Are we talking to ourselves?

Do we just do our own thing?

Or can we at least share infrastructure?

<table>
<thead>
<tr>
<th>Specification notations</th>
<th>Specification notations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation algorithms</td>
<td>Generation algorithms</td>
</tr>
<tr>
<td>Generated components</td>
<td>Generated components</td>
</tr>
</tbody>
</table>

Infrastructure
Incompatible specification notations

Are we able to reuse our specifications with more than one system for porting or comparison?

Can we understand each other's examples?

<table>
<thead>
<tr>
<th>Specification notations</th>
<th>Specification notations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation algorithms</td>
<td>Generation algorithms</td>
</tr>
</tbody>
</table>
Incompatible specification notations

Are we able to reuse our specifications with more than one system for porting or comparison?

Can we understand each other's examples?
Incompatible specification notations

Are we able to reuse our specifications with more than one system for porting or comparison?

Can we understand each other's examples?

<table>
<thead>
<tr>
<th>Specification notations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation algorithms</td>
</tr>
</tbody>
</table>
Incompatible specification notations

Are we able to reuse our specifications with more than one system for porting or comparison?

Can we understand each other's examples?
Hiding and reinvention of the wheel

What is really going on inside our tools?

How easily can we improve on or experiment with the algorithms without having to build a whole new tool?

<table>
<thead>
<tr>
<th>Specification notations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation algorithms</td>
</tr>
<tr>
<td>Generated components</td>
</tr>
</tbody>
</table>
Hiding and reinvention of the wheel

What is really going on inside our tools?

How easily can we improve on or experiment with the algorithms without having to build a whole new tool?

Tool

- Specification notations
- Generation algorithms
- Generated components
Hiding and reinvention of the wheel

What is really going on inside our tools?

How easily can we improve on or experiment with the algorithms without having to build a whole new tool?

- Specification notations
- Generation algorithms
- Generated components
Hiding and reinvention of the wheel

What is really going on inside our tools?

How easily can we improve on or experiment with the algorithms without having to build a whole new tool?

Specification notations

Generation algorithms

Generated components
Interoperability and optimisation

At which levels can we mix two different systems?

Just joining or joining and optimisation?

Specification notations

Specification notations

Specification notations
Interoperability and optimisation

At which levels can we mix two different systems?

Just joining or joining and optimisation?

Generation algorithms

Generation algorithms

Generation algorithms
Interoperability and optimisation

At which levels can we mix two different systems?

Just joining or joining and optimisation?
Where do we go from here?

Coordinate:

designs of specification notations,

high-level specifications of generation algorithms,

language-independent generated components,

integration and optimisation at all levels, and

reusable infrastructure.