Intelligent Monitoring of Software Components

Sabine Moisan

INRIA Sophia Antipolis Méditerranée, FR
Artificial Intelligence ↔ Software Engineering

Cross disciplinary approach in two directions

Software Engineering

Components, frameworks,
models...

Components, MDE
- Complement model-based approach for deployment and run time adaptation of component processing chains

Knowledge bases,
supervision rules...

Knowledge-based systems
Supervision techniques
- Develop adaptable toolkits to design and customize KBS elements (GUIs, languages, KB verification tools, inference engines ...)

Artificial Intelligence
Supervision for (dynamic) software component adaptation

Knowledge-based approach to component configuration (select, assemble, tune...) at deployment and at run time

→ At runtime: Changing environment
  - Control execution of running components
  - Adapt configuration in case of changes
  - Handle failures: detect and fix
  - Provide maximum autonomy (embedded systems)

Supervision => means to express and operationalize this knowledge
Supervision Ontology

- Generic concepts & relations necessary to select, assemble, monitor, control... components
- Knowledge Representation Language
Policy Rules - Evaluation & Repair

- Know-how to take (run time) decisions
  - Involves different categories of rules
  - Translated into executable code: part of the run time system
- For **dynamic adaptation**
  - Evaluation: assess results, detect problems
  - Repair/adjustment: propagate problems, fix problems (tune parameters, reorganize configuration plan)

<table>
<thead>
<tr>
<th>Choice</th>
<th><strong>If</strong> object attribute ( a ) has value ( v ) <strong>Then</strong> use operator ( op_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialization</td>
<td><strong>If</strong> object attribute ( a ) has value ( v ) <strong>Then</strong> set parameter ( p ) to value ( v_1 )</td>
</tr>
<tr>
<td>Evaluation</td>
<td><strong>If</strong> result ( r ) has property ( p ) <strong>//detected automatically or manually</strong> <strong>Then</strong> declare problem ( pb ) for ( op_1 ) and call <strong>repair</strong></td>
</tr>
<tr>
<td>Repair/Adjustment</td>
<td><strong>If</strong> operator ( op_1 ) has problem ( pb ) <strong>Then</strong> transmit ( pb ) to ( op_2 ) or increase/decrease parameter ( p_1 ) or choose another sub-operator ( op_3 ) ... for future execution</td>
</tr>
</tbody>
</table>
Repair Example

If assess_op Threshold size_ambiguous
Then decrease parameter pthreshold
adjustment_method percentage
adjustment_step 0.05

If assess_op Box size_ambiguous
Then send_up size_ambiguous

If assess_op Detection size_ambiguous
Then send Threshold

If assess_op Compute_thresholds
Then parameters

Galaxy detection

Isolate object

Build contours

Compute thresholds

Threshold

Detect

Visualize

E Evaluation
R Repair
A Adjustment
Conclusion & Future Work

Used in AI applications
• Customizable mechanism
• Explicit way to express evaluation/repair policies
• Performance: Limited overhead

But... evaluation/repair rules and policies difficult to specify... and to automate

Now: Complement more formal/but limited methods (logical constraints)
Models@Runtime + Supervision

Introduce evaluation/repair supervision mechanism to guide exploration of search space and fine tune solution.
Any question?