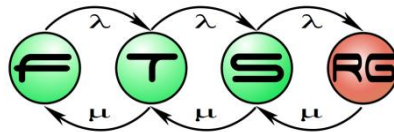


Developing and Visualizing Live Model Queries

Zoltán Ujhelyi, Tamás Szabó
István Ráth, **Dániel Varró**
Ábel Hegedüs (demonstrator)



What is a Model Query?

- $\text{ModelQuery}(A,B) \leftarrow \text{Cond}(A,B)$
 - Retrieve tuples of model elements **A,B**
 - Satisfying the query condition **Cond**
 - Enumerate one / all instances
 - With **A,B** as input or output parameters



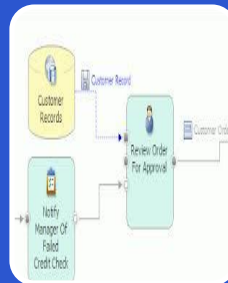
AUTOSAR

- *Select pairs of SystemSignal and its signal group which are not in the same IPDU*



IMA

- *Find instances of critical functions not running on different processors of different chassis*

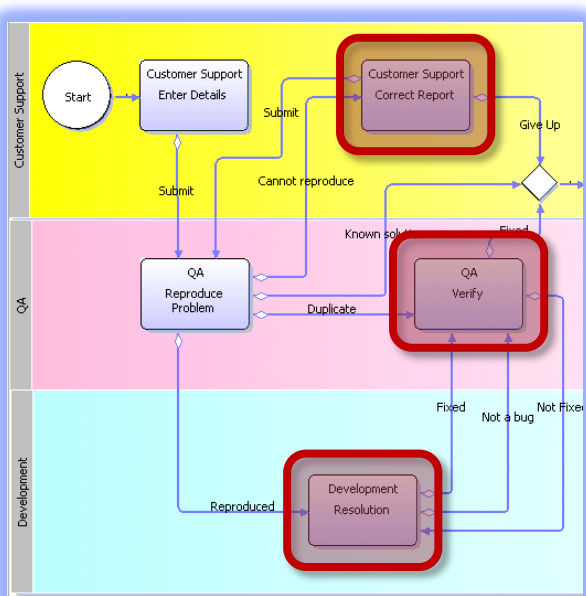


BPMN

- *Show sink activities with no outgoing flow*

Live Model Queries

- Live Query Evaluation:
Incremental cache of matches
 - Maintain a cheap cache
(only memory overhead)
 - Notify about relevant changes
 - Enable reactions to complex
structural events



Live Model Queries

- Live Query Evaluation:
Incremental cache of matches
 - Maintain a cheap cache (only memory overhead)
 - Notify about relevant changes
 - Enable reactions to complex structural events

Live Model Queries for Model Transformations

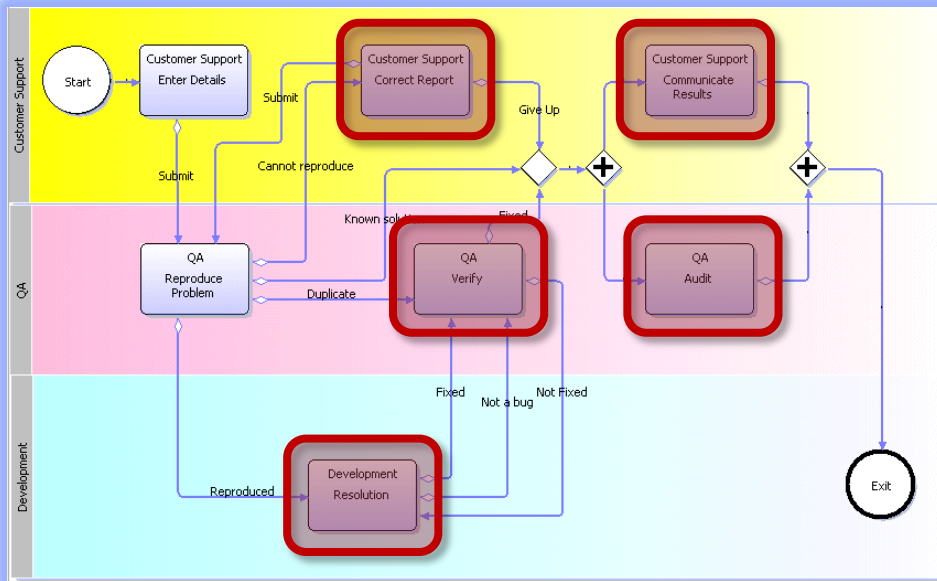
- Find all contexts a MT rule is applicable for
- Live MT rules: triggered automatically upon change

Live Model Queries for Early Validation

- Find all violations of a well-formedness constraint
- Immediate re-validation upon model change

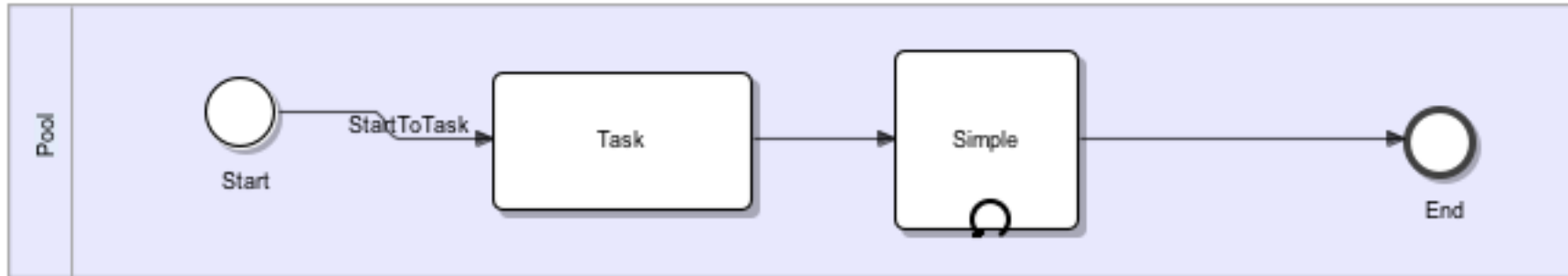
Live Model Queries for Traceability Management

- See talk tomorrow at 11:30



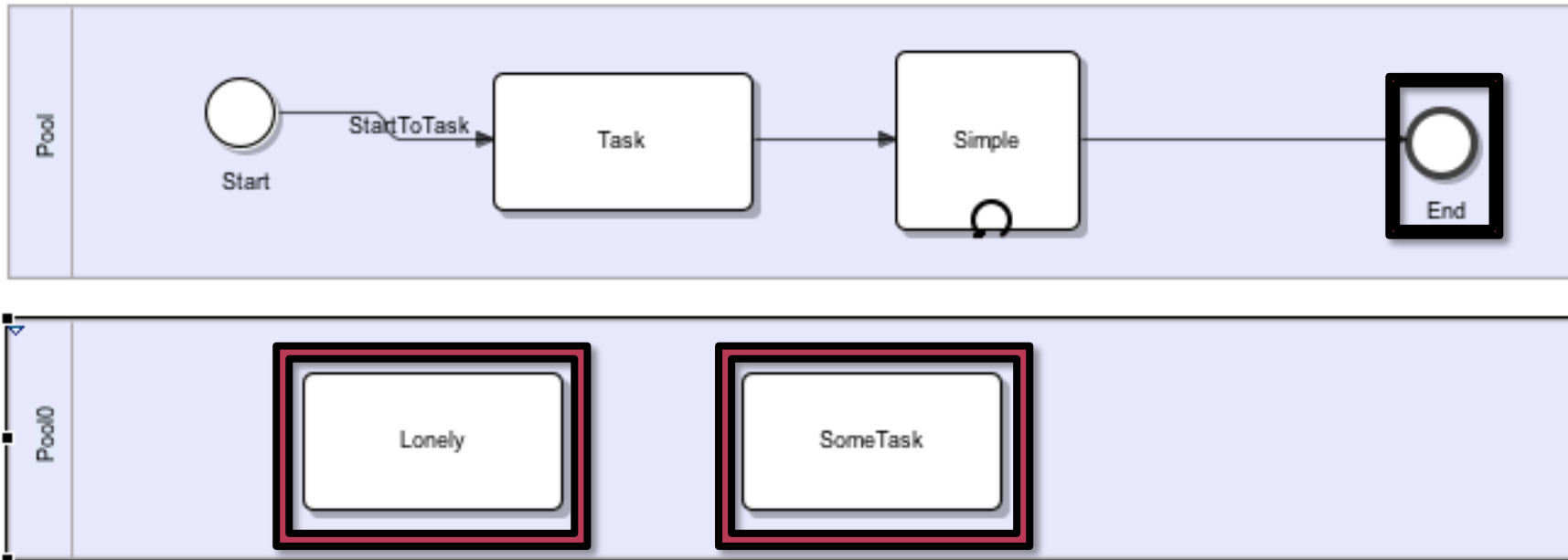
Example: Validation of BPMN Models

- BPMN models
 - Describing business processes
 - Models control and dataflow
 - Flowchart-like notation



Example: Validation of BPMN Models

- Well-formedness validation during editing
 - Sink Activity: activity without outgoing edge

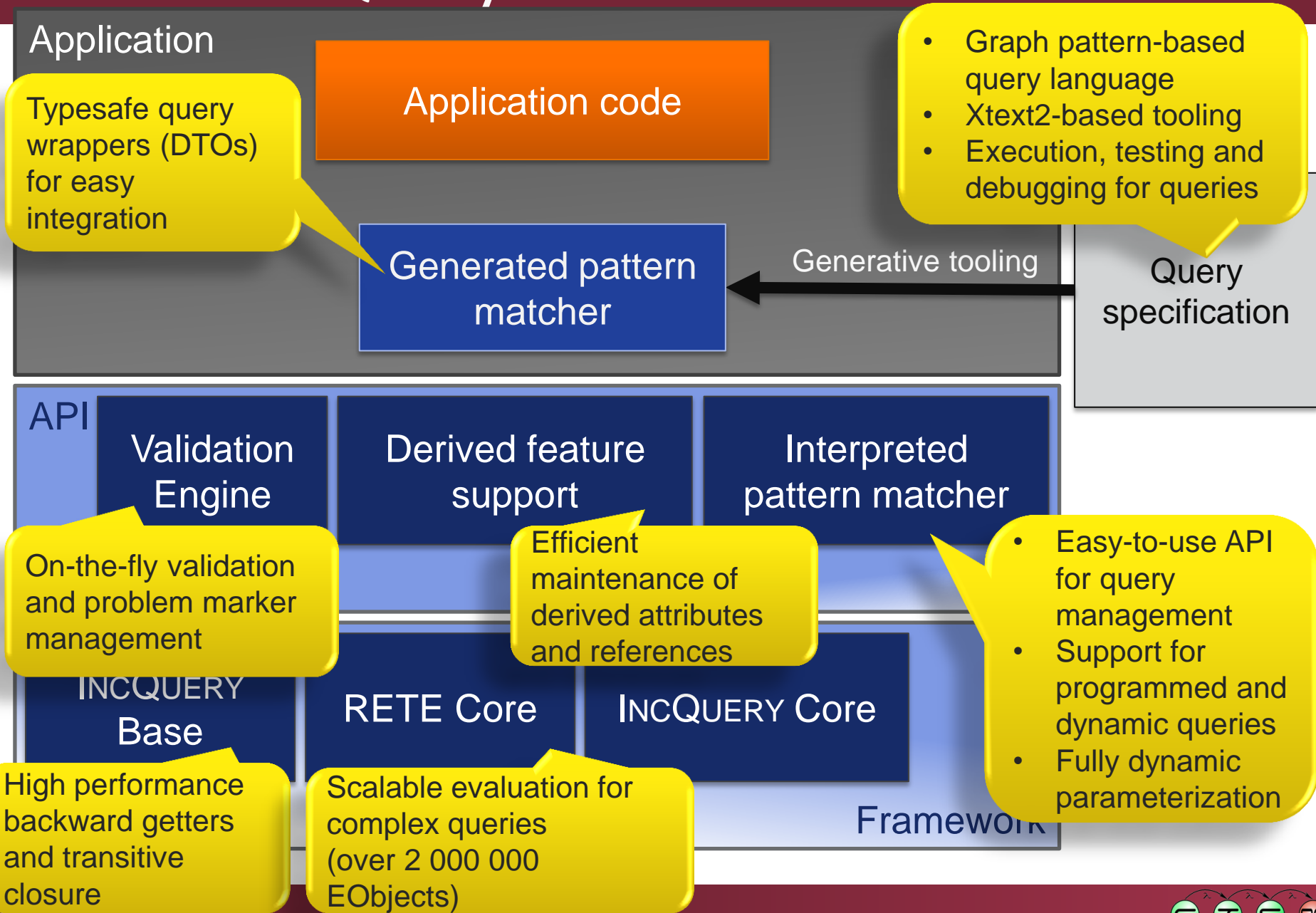


The screenshot shows a software interface with a 'Problems' window. The window displays 7 errors, 0 warnings, and 0 infos. The errors are listed in a table with columns for Description, Resource, Path, and Location.

Description	Resource	Path	Location
✖ Either a time date or a time cycle should	OMG E-Mail ...	eClarus BPMN Example	Unknown
✖ Either a time date or a time cycle should	OMG E-Mail ...	eClarus BPMN Example	Unknown
✖ Either a time date or a time cycle should	OMG E-Mail ...	eClarus BPMN Example	Unknown
✖ Either a time date or a time cycle should	OMG E-Mail ...	eClarus BPMN Example	Unknown

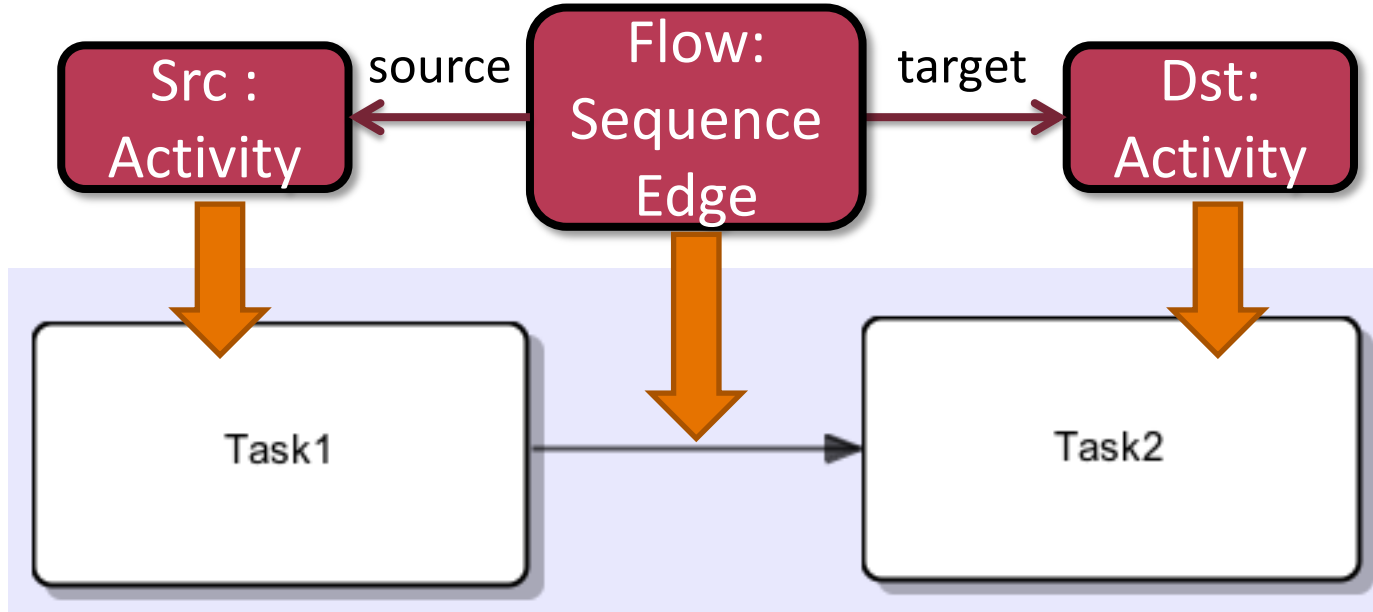
EMF-IncQuery Framework

EMF-IncQuery: Architecture Overview



Simple Graph Pattern in EMF-IncQuery

```
pattern sequenceFlowEdge
  (Flow:SequenceEdge,
   Src:Activity, Dst:Activity)= {
  SequenceEdge.source(Flow, Src);
  SequenceEdge.target(Flow, Dst);
}
```



And Some More Complex Examples...

```
pattern hasOutEdge(A: Activity) {  
  find sequenceFlowEdge(_Fr, A, _Other);  
} or {  
  find messageFlowEdge(_Fr, A, _Other);  
}
```

Pattern composition
(for reuse)

```
pattern sinkActivityNames(Name) {  
  Activity(A);  
  Activity.name(A, Name);  
  neg find hasOutEdge(A);  
}
```

Negative composition
(negation, quantification)

EMF-INCQUERY Development Tools

- Works with most EMF editors out-of-the-box
- Reveals matches as selection

Pattern Editor

EMF Instance Model

Query results recalculated on-the-fly

Query Explorer

Visualization of Live Model Queries in EMF-IncQuery

Requirements for Query Visualization

Genericity

- Multiple Model Sources
 - Model Editors
 - Current Selection

Incrementality

- Query Changes
- Model Changes

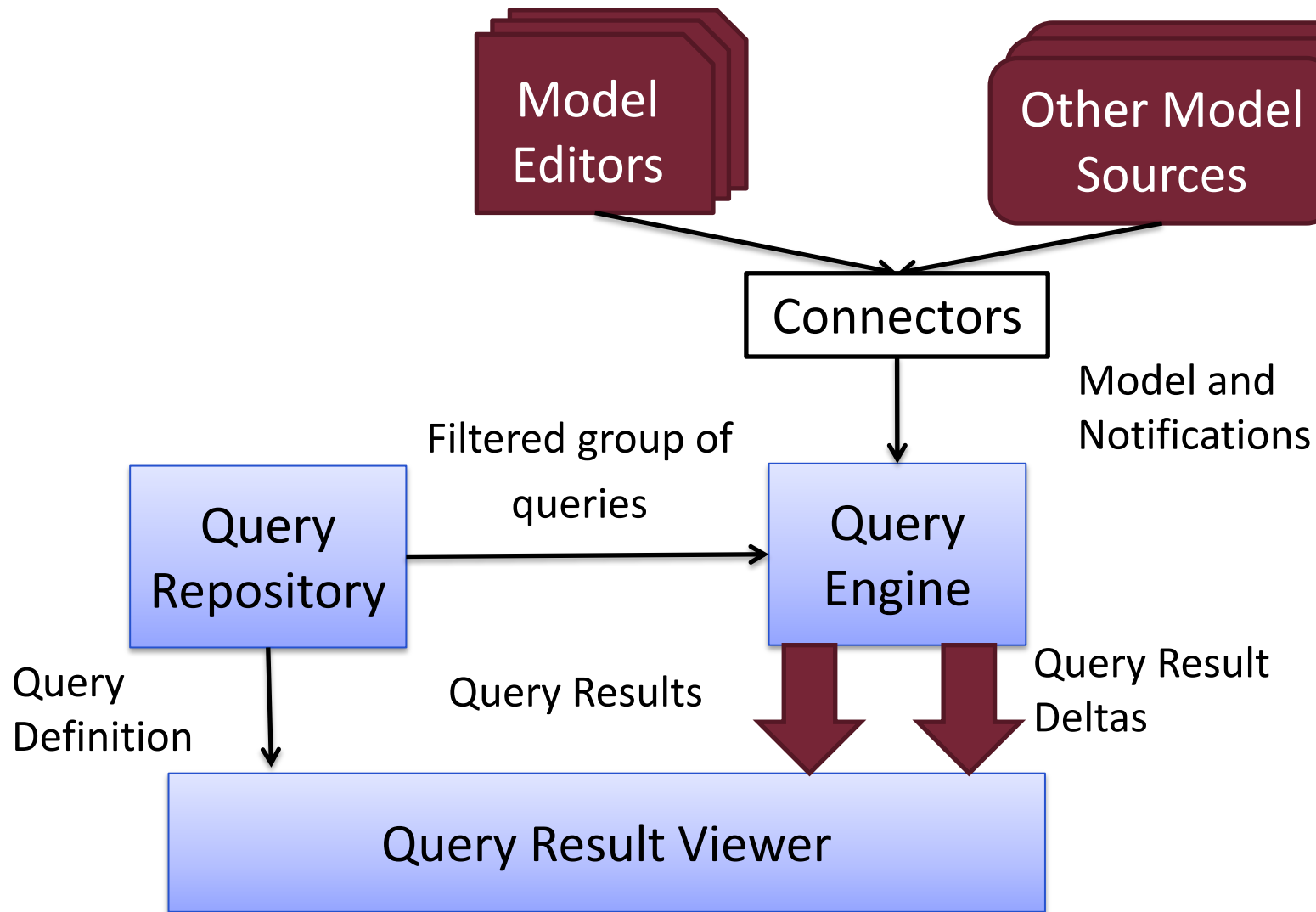
Traceability

- Query Definitions
- Input Model

Presentation

- Filtering
- Grouping

Proposed Architecture of Query Visualizer

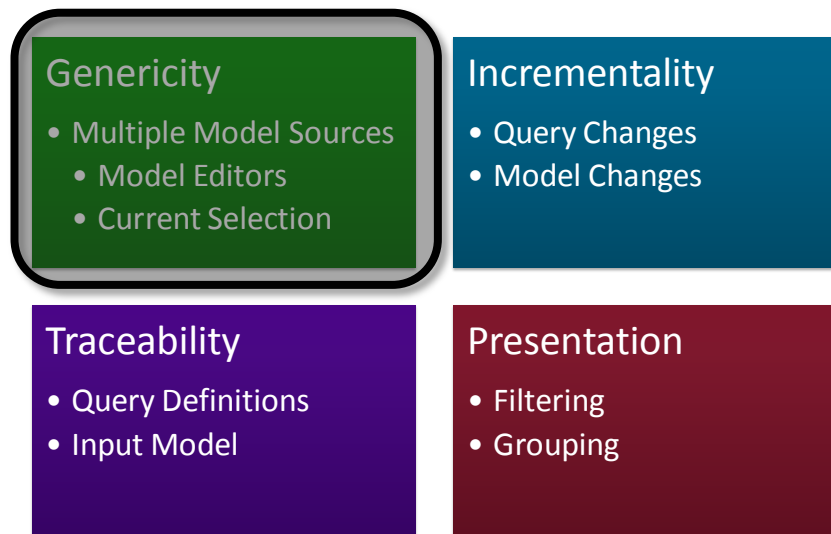
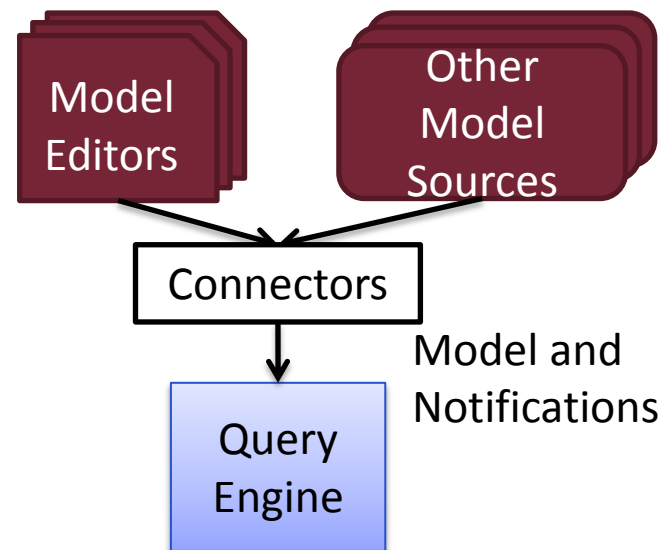


Model Source Connectors for Genericity

- Different model sources
 - Graphical (e.g. BPMN, UML)
 - Textual (e.g. OCL)
 - Different implementation technologies

➔ Model source connectors

- Operations
 - Open model
 - Send notification upon model change
 - Get current selection
 - Close editor



Query-based Indexing for Incrementality

- Update query results **incrementally** upon
 - Model updates
 - E.g. using the built-in editor
 - Engine: handles this case internally
 - Query updates
 - E.g. using the query editor
 - Engine: rebuilds internal model indexes

Genericity

- Multiple Model Sources
 - Model Editors
 - Current Selection

Incrementality

- Query Changes
- Model Changes

Traceability

- Query Definitions
- Input Model

Presentation

- Filtering
- Grouping

Query-based indexing

- Contains model elements accessed by the query
- Populated by an exhaustive model traversal (can be slow!)
- Goal: Avoid unnecessary re-traversal

Strategy 1:

Generic model indexer

- “Wildcard mode”:
index every model element
- Higher memory consumption

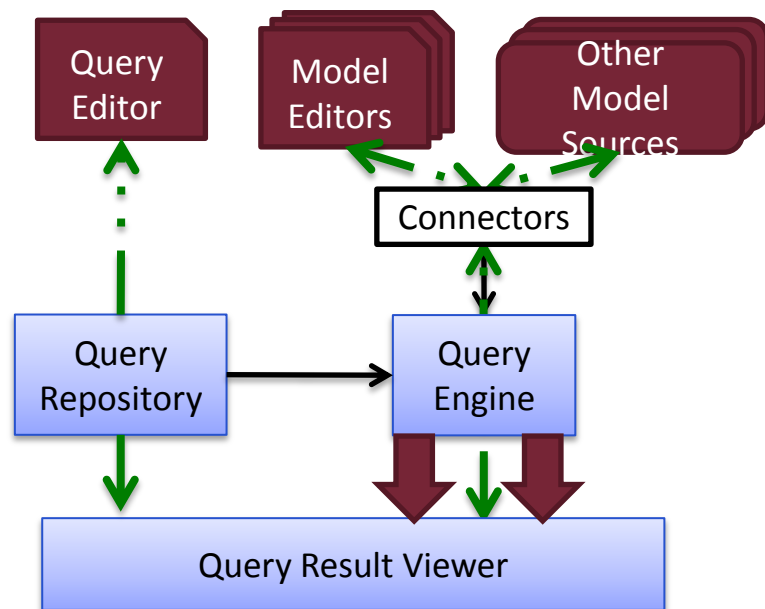
Strategy 2:

Group initialization

- Initialize several patterns together

Traceability and Navigation

- Maintaining source **models** and **queries**
- Support for navigating to/highlighting
 - Corresponding query definitions
 - Highlighting functionality in the query editor
 - Referenced model element(s)
 - Editor-dependent implementation
 - *Model source connector* handles functionality



Genericity

- Multiple Model Sources
- Model Editors
- Current Selection

Incrementality

- Query Changes
- Model Changes

Traceability

- Query Definitions
- Input Model

Presentation

- Filtering
- Grouping

Traceability in Query Visualizer

The screenshot displays the Eclipse IDE interface with three main components highlighted:

- Pattern Editor:** Shows the source code for the `schoolqueries.eiq` file. The `classesOfTeacher` pattern is highlighted with a blue box. The code defines a pattern that finds courses taught by a teacher in a specific school class.
- EMF Instance Model:** Shows the instance model for the `BUTE.school` project. The `Teacher Daniel Varro` instance is highlighted with a blue box, showing its associated courses.
- Query Explorer:** Shows the results of a query execution. The `T=Daniel Varro` result is highlighted with a blue box. The details panel shows the parameter `T` with the value `Daniel Varro`.

Additional visible elements include the `Properties` view showing the `Name` property of the selected object as `Daniel Varro`, and the `Selected Object: Teacher Daniel Varro` status bar.

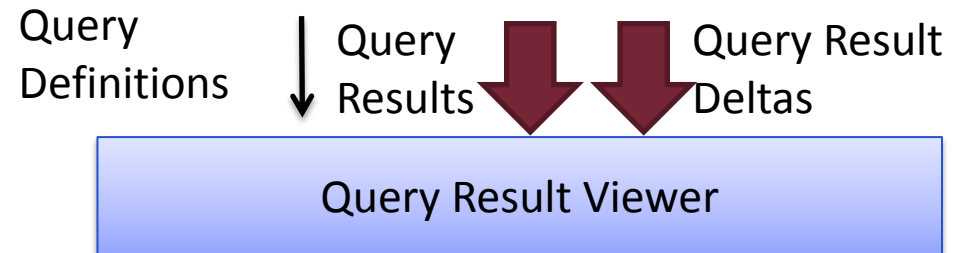
Pattern Editor

EMF Instance Model

Query Explorer

Grouping and Filtering for Presentation

- Define query groups („related” queries)
 - Static groups
 - E.g. Namespaces
 - Dynamic groups
 - E.g. Dependency based



- Filtering
 - Filter visible queries
 - Uses hints from developer
 - Manual overriding
 - Filter query results
 - By binding query parameters

Genericity

- Multiple Model Sources
 - Model Editors
 - Current Selection

Incrementality

- Query Changes
- Model Changes

Traceability

- Query Definitions
- Input Model

Presentation

- Filtering
- Grouping

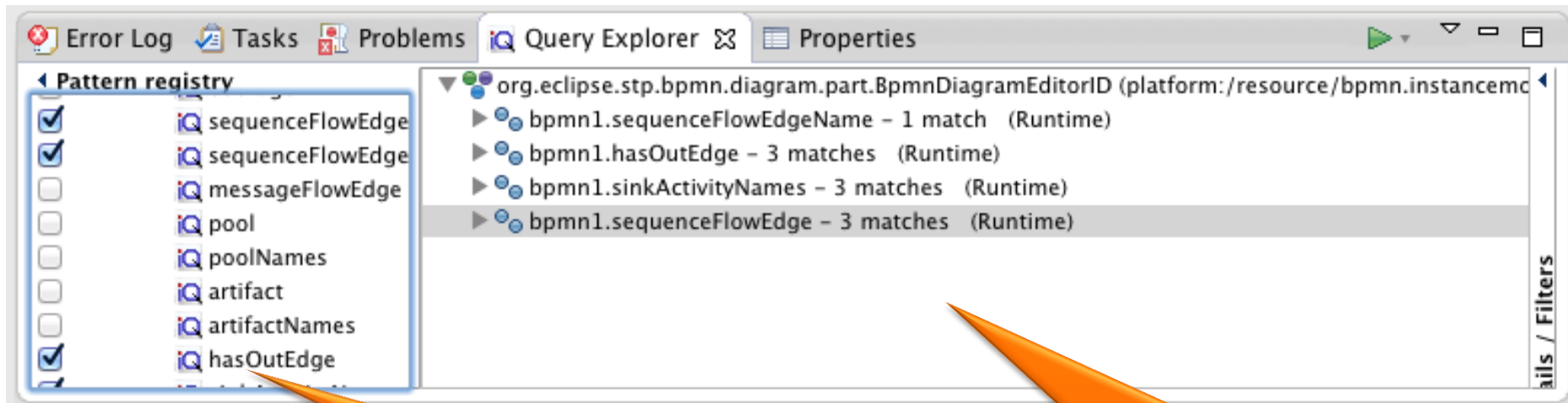
Result Viewer

The screenshot shows the Query Explorer interface with a tree view of pattern registry items. Three orange callout boxes highlight specific features:

- Different model sources:** Points to the hierarchical tree structure, including paths like `org.eclipse.stp.bpmn.diagram.part.BpmnDiagramEditorID`.
- Loaded Queries:** Points to the query names and their match counts, such as `bpmn1.hasOutEdge - 33 matches (Runtime)`.
- Query Results:** Points to the expanded view of the `A=Start` query, showing sub-items like `A=Task` and `A=Simple`.

The left sidebar is labeled "Pattern registry".

Filtering Visible Queries



Elements in Query Registry

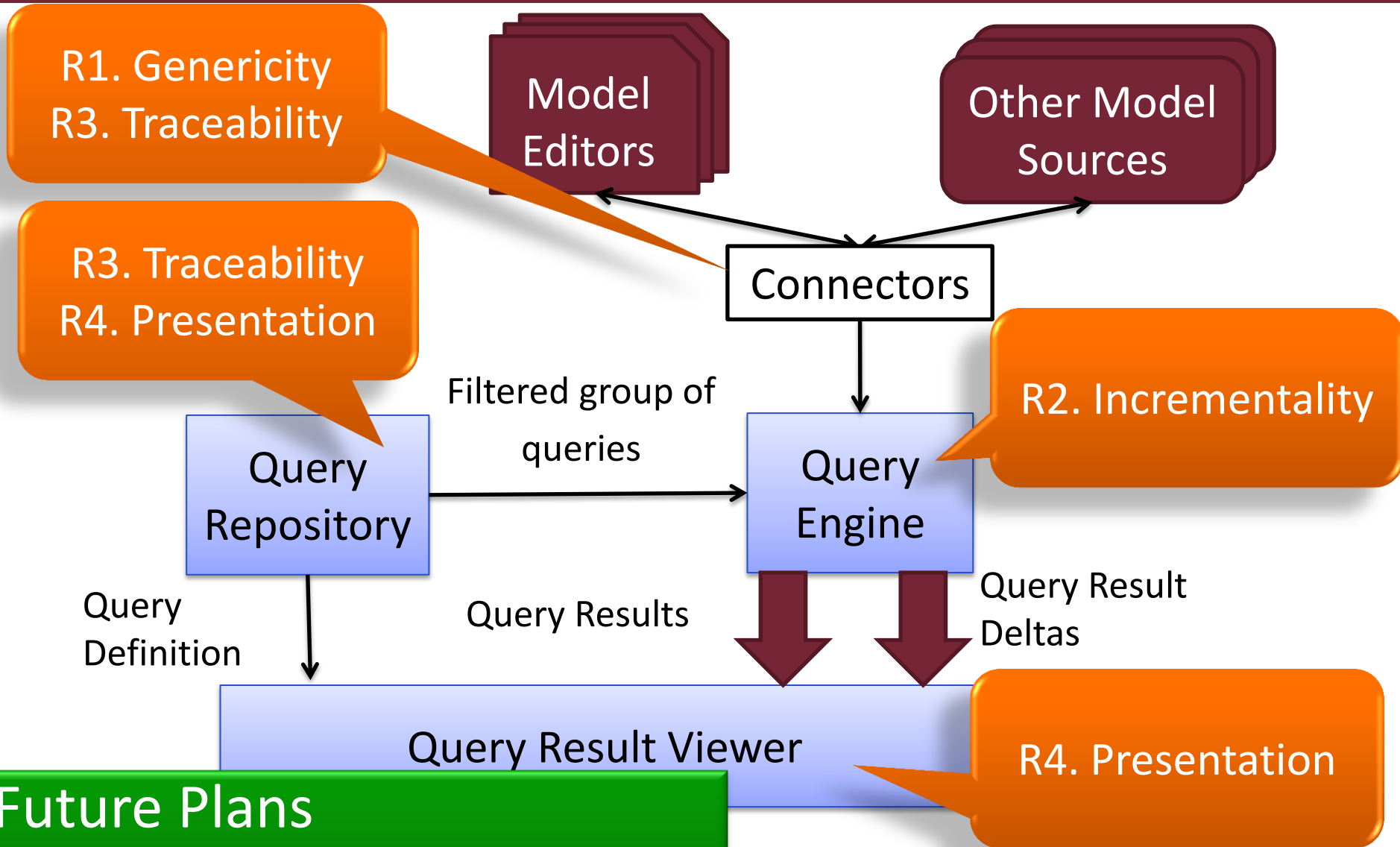
Queries Loaded

Query Result Filtering

The screenshot shows the Eclipse Query Explorer interface. The left pane displays a tree view of query results under the package `org.eclipse.stp.bpmn.diagram.part.BpmnDiagramEditorID`. The selected query is `bpmn1.sequenceFlowEdge`, which has 1 match and is filtered. The filter configuration is shown as `Flow=StartToTask, Src=Start, Dst=Task`. The right pane shows a table with two columns: `Parameter` and `Filter`. The `Src` parameter is highlighted with a blue background, and its filter value is `Activity Start`.

Parameter	Filter
Flow	
Src	Activity Start
Dst	

Conclusions and Future Work



Future Plans

- Analysis-based grouping
- Graph-based result visualization