Decreasing your Coffee Consumption by Incremental (Code) Regeneration

Ákos Horváth, Ábel Hegedüs
IncQueryLabs Ltd.

István Ráth, Dániel Varró
Budapest University of Technology and Economics

Ádám Balogh
Ericsson Hungary Ltd.
Outline of the talk

Motivation and Background
Incrementality in transformations

Incremental transformation
• Different realizations
• Additional features

Evaluation
• Runtime performance

Conclusion
• Current status
• Future work

Main Contributors
• Zoltán Ujhelyi
• Ábel Hegedüs
• Márton Búr
• Róbert Dóczi
• Péter Lunk
• István Ráth
• Ákos Horváth
Model transformations

- Incrementality in transformations
Incrementality in model transformations: The common understanding

Motivation

- **Req. 1**: Source model sizes in the range of millions (e.g., AUTOSAR)

- **Req. 2**: Transformation times may take hours in case of complex transformations

- **Req. 3**: Chaining of model transformation
Incrementality in model transformations

Our understanding and the current aim

One way synchronization

Interpretation: when a change occurs, ...

- **Source** incrementality: “re-read as little as possible”
  - Req. 1: Source model sizes
- **Target** incrementality: “re-write as little as possible”
  - Req. 2: Complex transformation execution
- Req. 3: Transformation chaining requires both target and source incrementality
Model transformations (cont.)

- Type of incremental approaches (as we understand)
No Incrementality: Batch Transformations

1. First transformation
2. Source model changes
3. Re-execute from scratch for all source models
“Dirty” Incrementality

Pros:
• Coarse grain incrementality

Cons:
• Complex MT can be slow
• Cleanup (after an error)?
• Chaining?
• Source and target incrementality?

1. First transformation
2. Source model changes
3. Re-execute from scratch only for changed model segments
**Incrementality by traceability**

1. First transformation
2. Source model changes
3. Detect missing trace links
4. Re-execute MT only for untraceable elements

**Pros:**
- Fine grain incrementality (source and target)
- Better performance
- Chaining (req. 3.)

**Cons:**
- Highly depends on traceability links
- Smart queries needed
Query result based transformations

Pros:
• Refined context: driven by query result set changes
• Chaining (req. 3)

Cons:
• Language-level restrictions

1. First transformation
2. Source model changes
3. Detect new activations
4. Fire rule activations (in relevant context)
Realization of M2M Steps

- VIATRA
- Model-to-model transformations in
- Cyber Physical Systems demonstrator
EMF-IncQuery and Viatra based transformations

- **EMF-IncQuery**
  - Planned release **1.0: 2015.06.30 (graduation)**
  - “an incremental model query engine”

- **New VIATRA**
  - [http://eclipse.org/viatra](http://eclipse.org/viatra), “an event-driven and reactive model transformation platform”
  - High-performance model transformation engine built on IncQuery technology
  - Unique feature: supports batch, incremental transformation over a unified virtual machine infrastructure (EVM)
  - Full Java and EMF compatibility
  - API inspired by Xtend and Xbase
  - Tightly integrated into Xtend
  - Planned release **0.7: 2015.06.30**
Cyber-physical system demonstrator

The CPS system describes a set of stateful applications allocated to hosts in order to fulfill requests

- Application types and instances
  - Each type specifies a state machine with states and transitions
- Host types and communicating instances
  - Limited resources, e.g. CPU, memory
- Transitions may send or wait for signals
  - Sending is addressed to application types

▶ [https://github.com/IncQueryLabs/incquery-examples-cps](https://github.com/IncQueryLabs/incquery-examples-cps)
Batch Simple and Optimized

Based on Xtend

- Reads the SRC model and generates both Traceability and target elements
- Optimized uses
  - some caches for traceability elements
  - Optimized based on generated java code
Batch IncQuery

1. Source model changes

2. Notifications sent

3. Regeneration is triggered

3. Deletes trace and target models
Batch IncQuery

1. Source model changes

2. Notifications sent

3. Regeneration is triggered

4. Trace and Target are regenerated

3. Deletes trace and target models

Transformation

Δ

Reads query results

Reexecutes transformation

Notifications sent

Xtend
Batch IncQuery

Based on Xtend and IncQuery

- Both Traceability and Source model is queried using IncQuery
  - Cache sets are kept synchronized with changes
  - Transformations just reads results
  - Uses IncQuery only where useful
Transformation + IncQuery (dirty incremental)

1. Source model changes
2. Notifications sent
3. Calculates dirty model segment
   - Reads query results
4. Executes rules on dirty segment
   - New parts created, old modified

0. Transformation rules

Δ dirty part

Δ

Transformation

Change Monitor

Xtend
Change Monitor + IncQuery (dirty incremental)

Based on Custom Change monitor

- Calculation of dirty segment is hand code ➔ defines level of incrementality
  - Handling of deletion/creation can be cumbersome
- EMF-IncQuery is used to receive notification on changes
Explicit Traceability

1. Source model changes

0. Transformation rules

0. Rule Activations

2. Notifications sent

EVM

Δ match sets

Fires rule activation

TRACE

Δ

2. Missing Trace link “negative notifications”

SRC

Δ

TRG

3. Target model changed

3. New activations appear

4. Everything starts over
Explicit Traceability

Based on IncQuery and EVM (Event-Driven Virtual Machine)

- Model modifications defined in Xtend
- Keeps rule activation life-cycle by EVM
  - Mainly queries traceability model
  - Follows what is happening with source model
- Execution triggering is controlled separately from activation
  - Single rule
  - Delayed for multiple rules (e.g., EMF transaction end)
Query based traceability

1. Source model changes

0. Rule Activations from the start of transformation (reference point)

2. Match set changes propagated

Δ changed match sets

Fires rule activation

3. New activations appear

3. Target model changed

2. Notifications sent

EVM

Δ
Query based traceability

1. Source model changes
2. Notifications sent
3. Target model changed
4. New reference point is set
5. Transformation continues

Transformation

Δ changed match sets

2. Match set changes propagated

Δ changed match sets

Fires rule activation

0. Rule activations from the start of transformation (reference point)
3. New activations appear

3. Target model changed
Query based traceability

Traceability is kept by state of match sets

- Keeps rule activation life-cycle by EVM
  - Traceability by appeared/disappeared/updated matches based on previous "reference point"
  - Rule activation only queries source model

- Execution triggering is controlled separately from activation
Viatra (API) for model transformation definition

Provides a user-friendly API over EVM for defining transformation rules

- Supports
  - Batch execution
  - Query based traceability

- Rule definition
  - Pre condition – EMF-IncQuery pattern
  - Action – any Java compatible code, usually Xtend

Part of the upcoming Viatra 0.7 release

- Much more to see ➔ http://www.eclipse.org/viatra/
Additional features

- **Viatra**
  - *Transformation chains defined in MWE2*
    - Extensible language feature
      - API for Viatra incremental and batch transformations
    - Explicit workflow for control

- **CPS case study specific**
  - M2T code generation for Java based on *change monitor*
    - Template based
    - JDT based
  - *Not much difference in performance*
Evaluation

- Performance benchmarks
## Test scenario

### For all four transformations
- Batch
  - simple and optimized
  - IncQuery and Viatra
- Incremental
  - Change monitor based
  - EVM: explicit traceability and query result based (qbr)
  - Viatra: query based

### Executions
- First transformation execution
- Small modification + (re)execution

### Four different model scenarios
- Industrial (presented below)
- Client-Server
- Publish-Subscribe

### Environment
- New desktop machine with 16 GB RAM

### Parameters
- 10 GB Heap
- Maximum 10 minutes execution times for complete chain

### Trace model’s size similar to target model

<table>
<thead>
<tr>
<th>Scale</th>
<th>SRC Objects</th>
<th>SRC References</th>
<th>TRG Objects</th>
<th>TRG References</th>
<th>Trace Objects</th>
<th>Trace References</th>
<th>SUM Objects</th>
<th>SUM References</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>395</td>
<td>772</td>
<td>366</td>
<td>736</td>
<td>354</td>
<td>720</td>
<td>1115</td>
<td>2228</td>
</tr>
<tr>
<td>5</td>
<td>762</td>
<td>1535</td>
<td>1522</td>
<td>3056</td>
<td>3254</td>
<td>6520</td>
<td>10124</td>
<td>29739</td>
</tr>
<tr>
<td>8</td>
<td>3604</td>
<td>17111</td>
<td>6108</td>
<td>14236</td>
<td>7112</td>
<td>14236</td>
<td>22056</td>
<td>115824</td>
</tr>
<tr>
<td>16</td>
<td>7820</td>
<td>89193</td>
<td>7124</td>
<td>12395</td>
<td>7112</td>
<td>14236</td>
<td>22056</td>
<td>115824</td>
</tr>
<tr>
<td>32</td>
<td>17714</td>
<td>594181</td>
<td>16308</td>
<td>24837</td>
<td>16297</td>
<td>32605</td>
<td>50319</td>
<td>651623</td>
</tr>
<tr>
<td>64</td>
<td>43795</td>
<td>4424529</td>
<td>40960</td>
<td>50028</td>
<td>40948</td>
<td>81908</td>
<td>125703</td>
<td>4556465</td>
</tr>
</tbody>
</table>
Runtime for first transformation

Incremental approaches are within ~200%

Batch IncQuery + Batch Viatra are the fastest

Logarithmic scale!
Modification and second transformation

As expected, incremental transformation does well in (re)execution.

Logarithmic scale!
Conclusion

CPS demonstrator helps transformation engineers to

• Helps to identify key features of different approaches
• Provide hints for fine-tuning
  o Hybrid approach →
    ■ Batch+IncQuery for first transformation
    ■ Explicit traceability for (re)execution
• Viatra + EMF-IncQuery provides
  o a mature stack to execute a wide variety of incremental (and batch) transformations

Future work

• Viatra as a generic model transformation framework? →
much more is under the hood
  • Complex Event processing
  • Design Space Exploration
  • EclipseCon Europe 2015?
  • http://www.eclipse.org/viatra/
Final points

- **Viatra**
  - *0.7 will be out soon*
  - Generic transformation engine for batch and incremental transformation chains

  - The examples with all test results and more details are available form
    - [https://github.com/IncQueryLabs/incquery-examples-cps/](https://github.com/IncQueryLabs/incquery-examples-cps/)

  - Contributors:
    - Main: IncQuery Labs Ltd,
    - Auxiliary: Ericsson A.B., BME-FTSRG
    - Supporting projects: Ericsson internal project

- **Your contributions (feedback, forum posts, ideas, bugzillas, patches) are very welcome!**
  - *To what direction should we enhance the demonstrator and the Viatra transformation API?*
Evaluate the sessions at www.eclipsecon.org