Integrating Efficient Model Queries in State-of-the-art EMF Tools*

Gábor Bergmann, Ábel Hegedüs, Ákos Horváth, István Ráth, Zoltán Ujhelyi and Dániel Varró

Budapest University of Technology and Economics,
Department of Measurement and Information Systems,
1117 Budapest, Magyar tudósok krt. 2
{bergmann,hegedusa,ahorvath,rath,ujhelyiz,varro}@mit.bme.hu

Abstract. Model-driven development tools built on industry standard platforms, such as the Eclipse Modeling Framework (EMF), heavily use model queries in various use cases, such as model transformation, well-formedness constraint validation and domain-specific model execution. As these queries are executed rather frequently in interactive modeling applications, they have a significant impact on the runtime performance of the tool, and also on the end user experience. However, due to their complexity, they can also be time consuming to implement and optimize on a case-by-case basis. The aim of the EMF-IncQuery framework is to address these shortcomings by using declarative queries over EMF models and executing them effectively using a caching mechanism.
In the current paper, we present the new and significantly extended version of the EMF-IncQuery Framework, with new features and runtime extensions that speed up the development and testing of new queries by both IDE and API improvements.
We demonstrate how our high performance queries can be easily integrated with other EMF tools using an entirely new case study in which EMF-IncQuery is deeply integrated into the EMF modeling infrastructure to facilitate the incremental evaluation of derived EAttributes and EReferences.

1 Introduction

As model management platforms are gaining more and more industrial attention, the importance of automated model querying techniques is also increasing. Queries form the underpinning of various technologies such as model transformation, code generation, domain-specific behaviour simulation and well-formedness validation that are all essential in state-of-the-art modeling tools and toolchains.

The leading industrial modeling ecosystem, the Eclipse Modeling Framework (EMF [1]), provides different ways for querying the contents of models. These

* This work was partially supported by the SecureChange (ICT-FET-231101) European Research Project, the Certimot (ERC_HU-09-01-2010-0003) Project, the grant TÁMOP (4.2.2.B-10/1–2010-0009) and the János Bolyai Scholarship.

approaches range from manually coded model traversal to high-level declarative constraint languages such as Eclipse-OCL [2]. However, industrial experience [3] shows strong evidence of scalability problems in complex query evaluation over large EMF models, taken from the various modeling domains; and manual query optimization is time consuming to implement on a case-by-case basis.

In order to overcome this limitation, the EMF-IncQuery[1] framework [3] proposes to use declaratively specified queries over EMF models, executing them efficiently *without manual coding* using incremental graph pattern matching techniques [4]. The benefits of EMF-IncQuery with respect to the state-of-the-art of querying EMF models [2,5] include: (i) high performance querying of models in the range of millions of elements, (ii) efficient addressing of instance enumeration and backward navigation (which are both frequently encountered shortcomings of the EMF API); and (iii) a user friendly yet powerful declarative graph pattern based formalism.

In the current tool demonstration paper, we present the next evolutionary step of the EMF-IncQuery framework, focusing on novel query and execution features. As a complex case study, we illustrate how EMF-IncQuery can be deeply integrated into the EMF modeling layer to facilitate the efficient evaluation of *derived features* (virtual attributes and references that represent indirectly calculated structural information).

The paper is structured as follows: first, Section 2 gives a brief architectural and feature-oriented overview of EMF-IncQuery, with a focus on novel contributions. Section 3 shows how incremental queries can be integrated into the EMF modeling layer for the evaluation of derived features. Section 4 gives an overview of related work, and Section 5 concludes the paper.

2 Overview of EMF-IncQuery

2.1 Model queries by graph patterns

Graph patterns [6] are an expressive formalism that can be used for various purposes in model-driven development, such as defining declarative model transformation rules, capturing general-purpose model queries including model validation constraints, or defining the behavioral semantics of dynamic domain-specific languages [7]. A graph pattern (GP) represents conditions (or constraints) that have to be fulfilled by a part of the instance model. A basic graph pattern consists of *structural constraints* prescribing the existence of nodes and edges of a given type, as well as *expressions* to define *attribute constraints*. A *negative application condition* (NAC) defines cases when the original pattern is *not* valid (even if all other constraints are met), in the form of a negative sub-pattern. A match of a graph pattern is a group of model elements that have the exact same configuration as the pattern, satisfying all the constraints (except for NACs, which must not be satisfied). The specification for the complete query language of the EMF-IncQuery framework was described in [6], the current tool paper presents its implementation.

[1] `http://viatra.inf.mit.bme.hu/incquery/new`

Example. We illustrate our approach on a simple demonstration domain of *Schools* (encoded in EMF's ECore language as illustrated in Figure 1) that manage *Courses* involving *Teacher*s, and enroll their students assigned to *Year*s and *SchoolClass*es. Aside from simple EAttributes and EReferences, it also features *derived features* that are marked as volatile and transient, i.e. not stored explicitly in instance models but rather calculated on-demand by hand-written code. Such attributes or references usually represent a (simple) computed view the model and are frequently supported by ad-hoc Java implementations integrated into the EMF model representation.

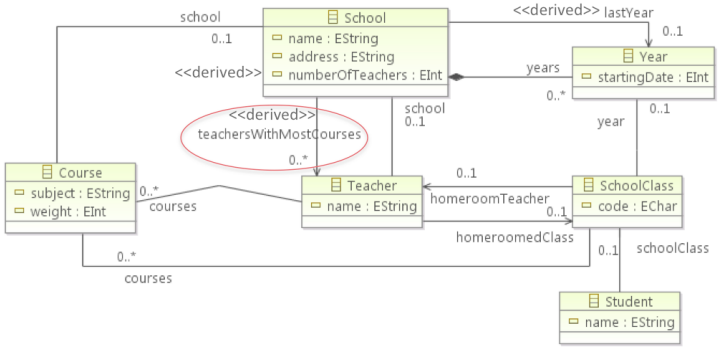


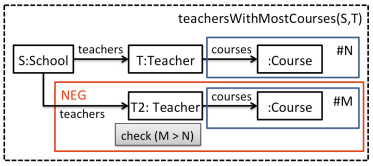
Fig. 1. The domain metamodel of the case study

In this paper, we show how graph patterns and EMF-IncQuery as the underlying execution engine can be used to ease the specification and automate the efficient evaluation of such features. The graph pattern *teachersWithMostCourses(S,T)* (Figure 2) is used to express the semantics of the *teachersWithMostCourses* derived EReference (connecting *School* and *Teacher* in Figure 1, highlighted with an ellipse), that is to identify those teachers who have the maximum number of *Course* instances assigned (through the *Teachers.courses* reference).

This graph pattern defines the target set of teachers by combining a negative application condition (NAC) and cardinality constraints. It expresses that a teacher T belongs to this set iff there is no other teacher $T2$ whose number of courses M (the actual cardinality, i.e. number of elements connected through the *courses* reference) would be larger than the number of courses N assigned to T.

2.2 Execution of incremental queries

The overall development workflow of the EMF-IncQuery framework focuses on the language tooling for specifying queries and then automatically generat-




```
1  pattern teachersWithMostCourses(S, T)=
2  {
3   School.teachers(S,T);
4   neg pattern moreCourses(S,T) = {
5    Teacher.courses(T,C) # N;
6    School.teachers(S,T2);
7    Teacher.courses(T2,C2) # M;
8    check(M > N);
9   }
10 }
```

Fig. 2. Graph pattern example in graphical and textual syntax

ing integration code that plugs into any existing EMF-based application. As a novelty targeted towards simplification, EMF-IncQuery now also features an interpretative query execution facility that allows the developer to specify ad-hoc queries directly from Java code, without involving the tooling and the code generator.

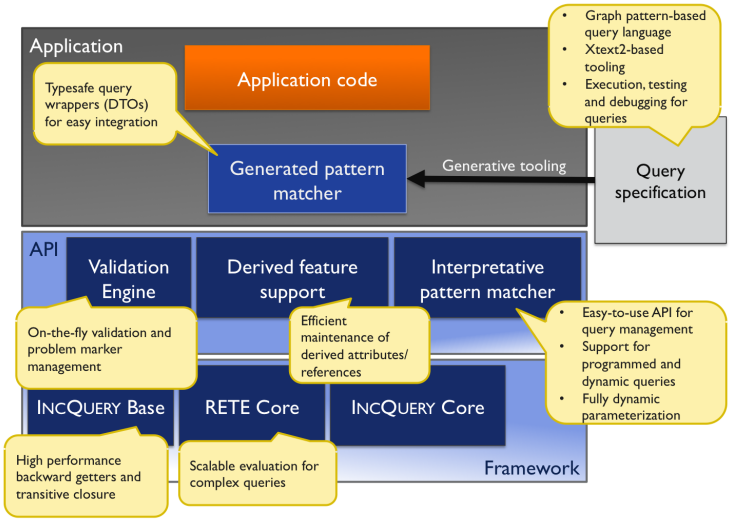


Fig. 3. Overview of the novel EMF-IncQuery architecture

The overall architecture of and EMF-based application built in EMF-IncQuery is overviewed in Figure 3. Based on the query specification (supported by an Xtext 2-based [8] editor, featuring syntax highlighting, code completion and well-formedness validation), pattern matcher plugins are generated that can be easily integrated to an existing Eclipse-based application. These plugins access the core functionality of the system through the EMF-IncQuery API that exposes three key novel services: (1) the *Validation Engine* provides a wrapper

to the EMF Validation service, to provide EMF-IncQuery-based on-the-fly well-formedness validators using standard Eclipse Error Markers; (2) the *Interpretative pattern matcher* provides an access point to quickly execute ad-hoc queries directly from Java code; (3) the *BASE*[2] component provides frequently used low-level incremental queries such as the instant enumeration of all instance elements belonging to a given EClass, or reverse navigation along unidirectional EReferences. BASE also provides a novel incremental transitive closure query algorithm that can be used to incrementally compute reachability regions.

Benefits. At the core, the incremental evaluation and lifecycle management of queries is facilitated by the RETE engine, originally developed for the Viatra2 model transformation framework [4]. Using this approach, the query results (the match sets of graph patterns) are cached in memory, and can be instantaneously retrieved when queries are issued. These caches are automatically and incrementally maintained upon model updates, using automatic notifications provided by EMF. There is a slight performance overhead on model manipulation, and a memory cost proportional to the cache size (approx. the size of match sets). These special performance characteristics make incremental techniques suitable for application scenarios such as on-the-fly well-formedness checking, live model transformation and other complex use cases.

3 Integrating incremental queries to the EMF modeling layer

In this section, we outline how the efficient querying features of the EMF-IncQuery framework can be integrated to EMF-based applications in a deep and transparent way, through the incremental evaluation and maintenance of derived features. The overall architecture of our approach is shown in Figure 4.

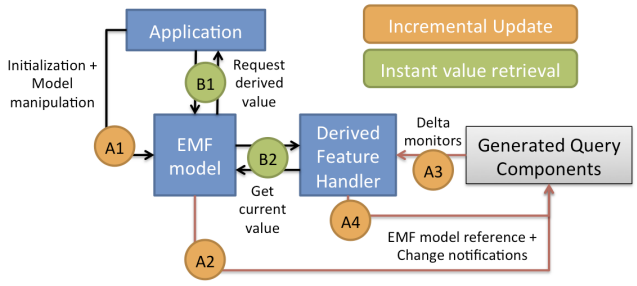


Fig. 4. Overview of the integration architecture

Here, the application accesses both the model and the query results through the standard EMF model access layer (query results are represented as the values

[2] `http://viatra.inf.mit.bme.hu/incquery/base`

of derived attributes or references) – hence, no modification of application source code is necessary. In the background, *Derived feature handlers* (novel features of the EMF-IncQuery API) are attached to the EMF .model plugin that integrate the generated query components (pattern matchers). This approach follows the official EMF guidelines of implementing derived features and is identical to how ad-hoc Java code, or OCL expression evaluators are integrated.

Challenges of using derived features in EMF. In using derived features with EMF-based applications, developers may encounter two key challenges. First, depending on the complexity of the semantics of derived features, their evaluation may impose a *severe performance impact* (since complex calculations and extensive model traversal may be necessary for execution). Unfortunately, this scalability issue will affect all other software layers using the .model code, including the user interface, model transformations, well-formedness validators etc. Second, due to the *lack of propagating notifications* for derived features, model changes will not trigger e.g. user interface updates.

Our approach provides a solution for both of these challenges. As the performance characteristics of the EMF-IncQuery engine have been shown to be practically agnostic of query complexity and model size [3], derived features of complex semantics and inter-dependencies can be used without severe evaluation performance degradation. Additionally, as shown in Figure 4, the update propagation mechanism of the RETE network (*delta monitors*) are connected to the EMF Notification layer so that the application software components are automatically kept up-to-date about the value changes of derived features.

Implementation details. In our prototype implementation[3], we augmented the architecture outlined above with a code generator that supports the automatic generation of integration code (*derived feature handlers*) based on a simple specification model that encodes the *core semantics* of backing queries, that can either be (i) a reference with a multiplicity of one (mapped to a scalar derived reference value) or $*$ (mapped to an unmodifiable EList as a derived reference value); (ii) the cardinality (match set size) of the backing query (e.g. to support the *School.numberOfTeachers* derived attribute in Figure 1).

The lifecycle of such handler objects is tied to the host EObjects, to enable their garbage collection together with the instance model itself. Additionally, they can be parameterized to use the EMF-IncQuery engine in the *batch evaluation mode*, which disables incremental update propagation, but may be more efficient overall for rarely used queries, or queries whose incremental maintenance would require too much memory.

4 Related work

EMF-IncQuery is not the first tool to apply graph pattern based techniques to EMF [9, 10], but its incremental pattern matching feature is unique.

[3] `http://viatra.inf.mit.bme.hu/incquery/examples/derivedfeatures`

Model queries over EMF. There are several technologies for providing declarative model queries over EMF. Here we give a brief summary of the mainstream techniques, none of which support incremental behavior.

EMF Model Query 2 [5] provides query primitives for selecting model elements that satisfy a set of conditions; these conditions range from type and attribute checks to enforcing similar condition checks on model elements reachable through references. Unfortunately, the expressive power of Model Query 2 is weaker than first order logic (and thus that of OCL and EMF-IncQuery). For example, more complex patterns involving circles of references or attribute comparisons between nodes cannot be detected.

EMF Search [11] is a framework for searching over EMF resources, with controllable scope, several extension facilities, and GUI integration. Unfortunately, only simple textual search (for model element name/label) is available by default; advanced search engines can be provided manually in a metamodel-specific way.

OCL evaluation approaches. OCL [12] is a standardized navigation-based query language, applicable over a range of modeling formalisms. Taking advantage of the expressive features and wide-spread adoption of OCL, the project Eclipse OCL provides a powerful query interface that evaluates OCL expressions over EMF models. However, backwards navigation along references can still have low performance, and there is no support for incrementality.

Cabot et al. [13] present an advanced three-step optimization algorithm for incremental runtime validation of OCL constraints that ensures that constraints are reevaluated only if changes may induce their violation and only on elements that caused this violation. The approach uses promising optimizations, however, it works only on boolean constraints, and as such it is less expressive than our technique.

An interesting model validator over UML models is presented in [14], which incrementally re-evaluates constraint instances whenever they are affected by changes. During evaluation of the constraint instance, each model access is recorded, triggering a re-evaluation when the recorded parts are changed. This is also an important weakness: the approach is only applicable in environments where read-only access to the model can be easily recorded, unlike EMF. Additionally, the approach is tailored for model validation, and only permits constraints that have a single free variable; therefore, general-purpose model querying is not viable.

5 Conclusions

Previously [3] we presented EMF-IncQuery as prototype framework for efficiently executing complex queries over EMF models, which adapts incremental technologies [4] for graph pattern matching. In the current paper, we present an evolved tool that includes two key improvements compared to previous versions: (i) an Xtext2-based tooling that fully implements the extended graph

pattern language [6] and (ii) a new runtime architecture that features several novel services including the on-the-fly validation engine and the interpretative ad-hoc query evaluator, built on a rewritten core that provides core queries and efficient transitive closures.

The secondary focus of this paper was a novel feature whereby queries can be deeply and transparently integrated into EMF-based applications to facilitate the efficient evaluation of derived features. The two key advantages of this approach are: (i) complexity-agnostic performance characteristics that allow developers to easily integrate derived references and attributes with complex semantics, without a severe scalability impact, even over very large instance models; (ii) transparent and automatic notification propagation that simplifies the integration to already existing user interfaces, model transformations and any other code that uses EMF models.

References

1. The Eclipse Project: Eclipse Modeling Framework. `http://www.eclipse.org/emf`.
2. The Eclipse Project: MDT OCL. `http://www.eclipse.org/modeling/mdt/?project=ocl`.
3. Bergmann, G., Horváth, Á., Ráth, I., Varró, D., Balogh, A., Balogh, Z., Ökrös, A.: Incremental Evaluation of Model Queries over EMF Models. In: Model Driven Engineering Languages and Systems, MODELS'10. Volume 6395 of LNCS., Springer (2010)
4. Bergmann, G., Ökrös, A., Ráth, I., Varró, D., Varró, G.: Incremental pattern matching in the VIATRA model transformation system. In Karsai, G., Taentzer, G., eds.: Graph and Model Transformation (GraMoT 2008), ACM (2008)
5. The Eclipse Project: EMF Model Query 2. `http://wiki.eclipse.org/EMF/Query2`.
6. Bergmann, G., Ujhelyi, Z., Ráth, I., Varró, D.: A Graph Query Language for EMF models. In: Proc. of ICMT'11, 3rd Intl. Conference on Model Transformation, Springer (2011)
7. Syriani, E., Vangheluwe, H.: Programmed graph rewriting with DEVS. Applications of Graph Transformations with Industrial Relevance (2008) 136–151
8. The Eclipse Project: Xtext. `http://www.eclipse.org/xtext`.
9. Biermann, E., Ermel, C., Taentzer, G.: Precise Semantics of EMF Model Transformations by Graph Transformation. In: MoDELS '08, Springer (2008)
10. Giese, H., Hildebrandt, S., Seibel, A.: Improved flexibility and scalability by interpreting story diagrams. In: Proceedings of GT-VMT 2009. Volume 18., ECEASST (2009)
11. The Eclipse Project: EMFT Search. `http://www.eclipse.org/modeling/emft/?project=search`.
12. The Object Management Group: Object Constraint Language, v2.0. (May 2006) `http://www.omg.org/spec/OCL/2.0/`.
13. Cabot, J., Teniente, E.: Incremental integrity checking of UML/OCL conceptual schemas. J. Syst. Softw. **82**(9) (2009) 1459–1478
14. Groher, I., Reder, A., Egyed, A.: Incremental consistency checking of dynamic constraints. In: FASE 2009. Volume 6013 of LNCS., Springer (2010)