StreamGen: A UML-Based Tool For Developing Streaming Applications

Michele Guerriero
Politecnico di Milano
Milano, Italy
michele.guerriero@polimi.it

Alessandro Nesta
Politecnico di Milano
Milano, Italy
alessandro.nesta@mail.polimi.it

Elisabetta Di Nitto
Politecnico di Milano
Milano, Italy
elisabetta.dinitto@polimi.it

ABSTRACT

Distributed streaming applications, i.e. applications that process massive and potentially infinite streams of data, are becoming increasingly popular in order to tame at the same time the velocity and the volume of Big Data. Designing and developing distributed streaming applications is currently difficult because it involves the employment of 1) complex programming paradigms to deal with the unboundedness of data streams together with 2) distributed streaming engines, each coming with its own APIs. To address the above shortcomings, in this tool demo paper we present StreamGen, a model-driven tool aiming at simplifying the development of distributed streaming applications. StreamGen provides (i) a UML profile to add streaming-specific concepts to standard UML Class Diagrams and (ii) a model-to-text transformation to automatically generate the application code starting from UML models.

KEYWORDS

Model-Driven Development, Streaming Applications, Big Data, Domain-Specific Modeling Languages, UML

ACM Reference Format:
Michele Guerriero, Alessandro Nesta, and Elisabetta Di Nitto. 2018. StreamGen: A UML-Based Tool For Developing Streaming Applications. In *MiSE'18: MiSE'18:IEEE/ACM 10th International Workshop on Modelling in Software Engineering , May 27, 2018, Gothenburg, Sweden.* , 2 pages. https://doi.org/10.1145/3193954.3193964

1 INTRODUCTION

Over the last years we have witnessed the development of many new *data-intensive technologies* that enable the exploitation of Big Data in order to extract its hidden value. In particular, various *distribute streaming engines* (e.g. Apache Storm[1], Apache Flink[2], etc.) have been proposed. These allow to develop and execute distributed streaming applications in order to simultaneously tame the volume and the velocity of Big Data. A distributed streaming application is able to process massive and potentially infinite streams of data, continuously updating its output.

However only few organizations have achieved full-scale production for their applications exploiting Big Data [2, 5]. This is mainly due to (i) the inherent complexity of developing Big Data solutions and to (ii) the huge variety of Big Data frameworks and technologies currently available. Hence, there is a need for creating languages, methodologies and tools to speed up the development of Big Data applications by assisting and guiding developers in the adoption and usage of the many frameworks and concepts introduced by this new field.

In this article we present StreamGen[3], a tool that makes one step in line with this investigation by enabling the model-driven development of distributed streaming applications. StreamGen offers (i) a Domain-Specific Modeling Language (in the form of a UML profile [6]) which abstract from platform-specific details and (ii) a model-to-text transformation to automatically generate executable code starting from profiled UML models.

2 TOOL OVERVIEW

Streaming application can be seen as a *directed graph* in which edges represent data streams and nodes represent operators that acquire, transform or store data streams. Each data stream is essentially an infinite and ordered (typically by means of a timestamp) set of tuples, which are structured into a finite set of *fields*. The computational model is inherently parallel and well suited for distributed and large-scale data processing. In fact, being the functional operators independent from each other, they can execute concurrently as soon as their respective input streams are available, thus enabling *task parallelism*. *Data parallelism* can also be exploited by splitting a data stream into disjoint partitions, which can be then processed concurrently. Finally, in order to deal with the infiniteness of data streams, distributed streaming engines typically provide a notion of *windowing*, i.e. chopping up a data stream into finite pieces along some temporal boundaries [3].

MiSE'18, May 27, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5735-7/18/05…$15.00
https://doi.org/10.1145/3193954.3193964

[1] http://storm.apache.org/
[2] https://flink.apache.org/
[3] https://github.com/MicheleGuerriero/UML2JavaFlink

Figure 1: StreamGen example model: the streaming Word Count

StreamGen provides a UML profile (implemented in Eclipse UML Papyrus) that captures the concepts needed to model distributed streaming applications as described above. The profile is intended to specialise the semantics of Class Diagrams. It provides stereotypes that represent the core concepts of streaming applications, such as data sources, data sinks, data streams and stream transformations. Figure 1 shows the streaming *Word Count* application modeled with StreamGen. The application processes in parallel an incoming stream of text lines and every 10 minutes emits the count of each distinct word it has seen.

Table 1: A summary of StreamGen provided operators.

Stereotype	Type
$\langle\langle MapOperator \rangle\rangle$	Transformation
$\langle\langle ReduceOperator \rangle\rangle$	Transformation
$\langle\langle TextFileDataSource \rangle\rangle$	Data Source
$\langle\langle TextFileDataSink \rangle\rangle$	Data Sink
$\langle\langle SocketDataSource \rangle\rangle$	DataSource
$\langle\langle SumOperation \rangle\rangle$	Transformation
$\langle\langle WindowFunction \rangle\rangle$	Transformation
$\langle\langle FlatmapOperation \rangle\rangle$	Transformation

The modeling method is such that *UML Classes* are used to model the nodes of the graph (sources, transformations and sinks – see Table 1 for a summary of the currently provided operators), while *UML Information Flow Links*[4] are used to model the data stream produced and consumed by the various nodes. Each Information Flow Link must *convey* objects (i.e. tuples, in streaming terminology) of a specific type. The conveyed data type can be either a primitive type (e.g. String or Integer) or a complex (user-defined) data type.

The profile allows to model data streams that are partitioned into sub-stream according to a *key* field of the conveyed data type (see the $\langle\langle KeyedDataStream \rangle\rangle$ stereotype). Moreover, it is possible to apply a window on a data stream (see the $\langle\langle WindowedDataStream \rangle\rangle$ stereotype), which can be of a specific size (minutes, seconds, etc) and type (fixed, sliding). It is then possible to specify (using a small snippet of Java code) how the operator that consumes a windowed data stream has to process the incoming windows of tuples.

Given a Class Diagram defined according to its profile, StreamGen is then able to automatically generate the corresponding application code. In particular, we have developed an Acceleo[5] model-to-text transformation that generates code runnable on the Apache Flink distributed streaming engine [4]. Nevertheless, our modeling approach is generic and other code generators could be plugged-in, to support also other engines (e.g. Apache Spark).

[4]https://www.uml-diagrams.org/information-flow-diagrams.html
[5]https://www.eclipse.org/acceleo/

3 DISCUSSION AND CONCLUSION

At the time of writing StreamGen is at its early stages, with Flink being the only supported platform and with a still limited set of provided operators (e.g. it is possible to model only data coming from file or socket connections). Nevertheless, we have already started an empirical evaluation of our approach. We took a group of 37 students and, after a 3-hour introduction to distributed streaming concepts, we let them use StreamGen in order to develop some example streaming applications. The results seems to confirm that StreamGen does simplify the development of distributed streaming applications. We plan to continue the development of StreamGen (e.g. adding support for more operators) and to strengthen our evaluation with further experiments. Moreover, in order to show and confirm the effectiveness of the provided abstractions, we plan to develop other code generation modules to support, using the same modeling language, other popular Big Data streaming frameworks (e.g. Apache Spark). Finally, we plan to integrate StreamGen with DICER [1], a model-driven tool, outcome of our previous research, which simplifies and speeds up the *deployment* (i.e. installation, configuration, etc.) of modern data-intensive infrastructures. By integrating the two tools and their modeling languages we would obtain a complete model-driven environment for both developing and deploying distributed streaming applications.

ACKNOWLEDGMENT

This work is supported by the European Commission grant no. 644869 (H2020, Call 1), DICE.

REFERENCES

[1] Matej Artač, Tadej Borovšak, Elisabetta Di Nitto, Michele Guerriero, and Damian A. Tamburri. Model-driven Continuous Deployment for Quality DevOps. In *QUDOS 2016 Proceedings*.
[2] Mathieu Colas, Ingo Finck, Jerome Buvat, Roopa Nambiar, and Rishi Raj Singh. 2015. *Cracking the Data Conundrum: How Successful Companies Make Big Data Operational*. Technical Report. Capgemini consulting. URL: https://www.capgemini-consulting.com/cracking-the-data-conundrum.
[3] Akidau et al. 2015. The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in Massive-Scale, Unbounded, Out-of-Order Data Processing. *Proceedings of the VLDB Endowment* (2015).
[4] Carbone et al. 2015. Apache Flink™: Stream and Batch Processing in a Single Engine. *IEEE Data Eng. Bull.* (2015), 28–38.
[5] Michele Guerriero, Saeed Tajfar, Damian A. Tamburri, and Elisabetta Di Nitto. Towards a Model-driven Design Tool for Big Data Architectures. In *BIGDSE '16 Proceedings*.
[6] François Lagarde, Huáscar Espinoza, François Terrier, and Sébastien Gérard. Improving UML profile design practices by leveraging conceptual domain models. In *ASE 2007*. 445–448.