Hasso Plattner Institut
Digital Engineering · Universität Potsdam

# BACHELOR PROJECT
# Test Data Generator Tooling
## Winter 2019/2020

## Parallel Data Generator Framework (PDGF)

With the current rise of Machine Learning libraries and frameworks as well as general data processing and data analytics engines there is bigger and bigger need for good quality test data. For that reason bankmark UG (haftungsbeschränkt) has developed the Parallel Data Generator Framework (PDGF). The main goal of PDGF is to provide the means to create massively parallel and scalable data generation programs without the need of any low-level programming. For that PDGF provides an XML-based specification language and many generators, e.g., long number generator or data time generator, that can be used to specify a custom schema and along the logic and rules for generating test data for this particular schema.

The current way for developing such data generator programs involves writing schema specifications in XML. For the most part using XML works just fine; there is no need for an IDE or compiler tool-chain, the programs are verbose such that they can be understood by other people, and XML as a language is standardized and well-defined, which means there is a plethora of tools available aiding in the creation of XML documents. But the use of XML also has its shortcomings. Data schemas, for instance, are inherently typed, that is fields have types and records in turn consist of typed fields. Using XML does not make it easy to enforce these types. That means typing related bugs that could potentially be found in advance lead to run time exceptions. Another case where the usage of XML can be cumbersome is when more complicated logic is needed. There are mechanisms to inject Java code into the XML specifications but doing so without an IDE or the typical tooling is more inefficient than simply writing a program in Java with an IDE.

A solution is to provide a Java-API additional to the existing XML-API. But this means all the parsing logic needs to be duplicated; once for XML and once for Java. A good fit for resolving the duplication issue could be the Java Architecture for XML Binding (JAXB). With JAXB it is possible to simply annotate existing code to specify and identify the relevant interfaces, classes, and attributes and JAXB will then handle all the marshalling and unmarshalling including parsing. The PDGF code should be almost free of XML related code.

## Project Outline

The main goal is to use and showcase the usefulness of the a Java-API for PDGF. The participants should develop several applications that leverage the new API. Some examples for these applications are:

- a graphical user interface (GUI) for defining data generator programs,
- a web-service for generating data,
- a framework for unit testing that will generate appropriate data based on the type signature.

Such an application should then be showcased with a runnable prototype. Even though all application are independent of each other, they should use a common API, which should be developed as part of the project. Ultimately, the goal is to identify problems with the PDGF architecture that make the use of JAXB complicated or even prevent its use. Ideally the XML-API itself should be left untouched from the perspective of the end-user for backwards compatibility.

## External Partners

The project will be executed in cooperation with bankmark UG (haftungsbeschänkt) and potentially additional partners.

## Skills

This project is a software engineering project, participants need some experience in software engineering and experience with at least one programming language, preferably Java. Additional knowledge of software development processes and build tools, such as Maven, would be preferable. Student should be comfortable in documenting their work with tools like JavaDoc, Wiki, and others.

## Suggested Reading

- PDGF - https://www.bankmark.de/products-and-services/pdgf/
- JAXB - https://docs.oracle.com/javase/tutorial/jaxb/intro/index.html
- PDGF Papers
    - Rabl et al. “A Data Generator for Cloud-Scale Benchmarking”. TPCTC’10
    - Rabl et al. “Just can’t get enough – Synthesizing Big Data”. SIGMOD 2015. http://msrg.org/papers/SIGMOD15-DBSynth

## Contact

Tilmann.Rabl@hpi.de