PAVO: A Framework for the Visualization of Performance Analyses Results

Jürgen Walter
University of Würzburg

Maximilian König
University of Würzburg

Simon Eismann
University of Würzburg

Samuel Kounev
University of Würzburg

Abstract

Awareness of application performance can be derived through various quantitative analyses, model-based and measurement-based approaches. A suitable visualization supports the understanding of application performance. Usually, analysis tools include a tool specific visualization or no visualization at all. In this paper we proposed to decouple the result visualization from the analysis approach. We present the Performance VisualizatiOn (PAVO) framework which provides result visualization tailored to a given performance analysis result. To show benefits and usability of PAVO, we present our integration in Descartes Query Language (DQL).

1 Introduction

Performance is of particular relevance to software system design, operation, and evolution because it has a major impact on key business indicators. While most analysis approaches offer complex result log files, a suitable visualization of the results of performance analyses is helpful to raise performance awareness. Declarative Performance Engineering (DPE) [6] promises to automate the elaboration of a previously defined performance concern. Using OMG's Structured Metrics Meta-Model (SMM) [8], Measurement Architecture for Model-Based Analysis (MAMBA) [2] or the Descartes Query Language (DQL) [3] a user can specify performance information needs. This opens the opportunity for a visualization tailored to a given quantitative analysis. Dependent on the result type, different visualizations can be drawn. For example, series of continuous values can be represented as line charts while discrete values should not be connected and can be shown as bar chart. In general there are multiple alternative visualization, e.g., continuous and discrete series both could be depicted as a box plot. To provide tailored visualizations we designed PAVO [5]. PAVO encapsulates the complexity of visualizations into a ready to use component.

2 State of the Art

Visualizations in performance engineering include graph-based and chart-based approaches. Graph-based approaches visualize structures. These can be obtained from execution traces. Examples include the ExplorViz [4] approach to visualize software landscapes using a city metaphor. ExploreViz visualizes the whole system while charts usually focus on a certain aspect. Chart-based visualizations for quantitative analyses are included in several APM and model-based analysis tools. Further, there are commercial visualization tools independent of performance engineering, e.g., Tableau [7]. The user selects a subset of the analysis data and the visualization type, but they do not automate the selection of applicable approaches. We identified a lack of a generic visualization component designed to be reused in multiple performance analysis tools.

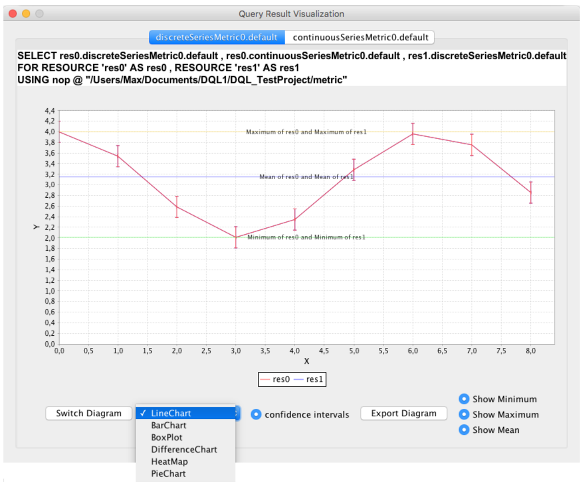
3 Usage and Features

The goal of PAVO is to provide a *generic* visualization building block for all stages of quantitative analysis approaches of performance engineering. To incorporate PAVO in your solution approach, either mount your solution approach into DQL or use PAVO as a standalone. To perform the latter, one has to transform the results to the result data format and pass it to the visualization interface. Then the user receives a visualization GUI. On performance result retrieval, PAVO decides on the visualization type. Currently supported visualization type include line charts, bar chart, box plots, pie charts and difference chart. Moreover, PAVO provides the following features

Automated Selection Based on the passed result, PAVO chooses one or more suitable diagram types for the visualization.

Slicing PAVO subdivides the visualization in multiple diagrams if the result does not suit to be depicted in one diagram.

Diagramm Switch In case the user prefers a different visualization, the user may be able to switch



```
SELECT res0.discreteSeriesMetric0.default , res0.continuousSeriesMetric0.default , res1.discreteSeriesMetric0.default
FOR RESOURCE 'res0' AS res0 , RESOURCE 'res1' AS res1
USING nop @ "/Users/Max/Documents/DQL1/DQL_TestProject/metric"
```

Query Result Visualization
discreteSeriesMetric0.default
continuousSeriesMetric0.default
Maximum of res0 and Maximum of res1
Mean of res0 and Mean of res1
Minimum of res0 and Minimum of res1
Y
X
res0
res1
Switch Diagram
✓ LineChart
BarChart
BoxPlot
DifferenceChart
HeatMap
PieChart
confidence intervals
Export Diagram
Show Minimum
Show Maximum
Show Mean

Figure 1: Several Features of PAVO GUI: Enrichments, slicing into two diagrams, diagram type selection dialogue, and image export

diagram types. PAVO filters diagram types to the ones applicable for the given result.

Updates The visualization of an online result updates itself on receiving new values which enables an integration into performance dashboards and is useful in log replays.

Enrichment Where applicable, PAVO provides various enrichments for basic diagrams that can be turned on and off. Graphics can be enriched using derived values like minimum, maximum and mean. In case the result is equipped with confidence intervals, they are displayed as error bars.

Image Export PAVO provides a button to export an image of the selected chart.

4 Library Design

PAVO receives a result of a quantitative performance analysis and returns a visualization GUI. The basic structure follows the model-view-controller pattern, as depicted in Figure 2. The `Controller` contains the `VisualizationSelector` component which controls instances of the model elements `VisualizationType` and `ResultModel`. The `View` component contains the view logic. To update the visualization of changing results, e.g., from a live system, we apply the observer pattern.

Result Model The choice of visualization is based on the result model for which we set up two requirements. PAVO requires a reduced model easily to adopt and meaningful to decide between different visualization types. Based on these requirements we decided against reuse existing models, e.g., SMM. SMM's current version 1.1.1 provides no differentiation if the measures are continuous or discrete. Further, we considered it to be to complex for our application scenario. Besides this, our model is compatible with results yielded by SMM. Figure 3 presents the result model. A `ResultContainer` contains possible multiple results. `AbstractQuantitativeResult` describes the surveyed element, metric, and statistic type. It subdivides into `ValuesResult` and `SeriesResult`. Series can be continuous or discrete (`isContinuous` set to false).

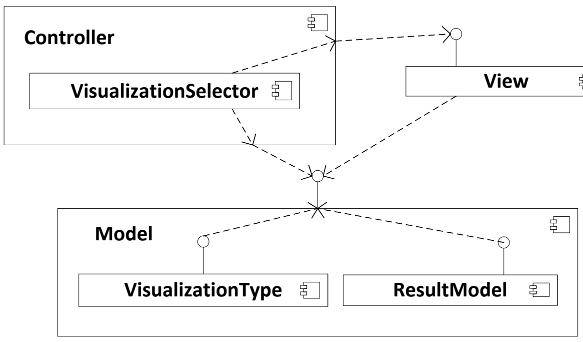


Figure 2: UML component diagram

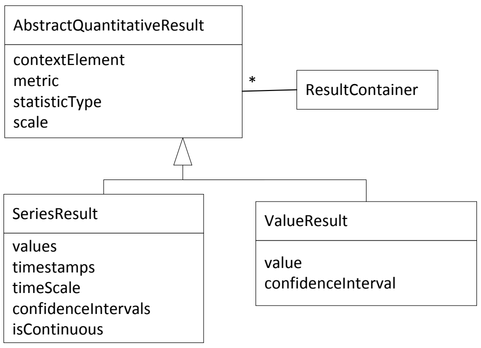


Figure 3: Result model

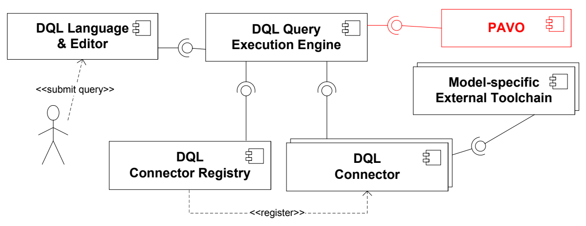
Visualization Types All visualization types implement the `IVisualizationType` interface which requires two methods. The `visualize` method creates for a result the corresponding visualization. The `canVisualize` is used to select applicable visualization types that can be chosen from the drop down menu. For, example a pie chart requires two or more elements of `ValueResult` with same metric. A line chart requires one or more `AbstractQuantitativeResult` that are either continuous `SeriesResult` or `ValueResult`. For bar chart, there is no restriction on `SeriesResult` to be continuous. Besides these two methods, the interface enables to specify various enrichment functionality, all following an identical construction. If the `supports` method returns true, PAVO creates a radio buttons within the GUI. A `draw` method enriches the visualization accordingly. The same applies also for other enrichment features like minimum, maximum, and mean. The `IVisualizationType` provides empty default implementations of the enrichment methods so that the developer has only to specify the applicable ones.

Controller The controller subdivides the elements within the passed result container according to metrics. PAVO creates one visualization tab per metric. The selection algorithm, located in `VisualizationSelector`, walks for each subset through all visualization alternatives and checks whether they can be applied or not using the `canVisualize` method of visualization types. So far, we randomly chose one of the applicable visualization types for intial visualization. For the future, we intend to implement a more sophisticated selection that considers data distribution. For example, box plots provide better insights for asymmetric distributions with outliers than for normal distributions.

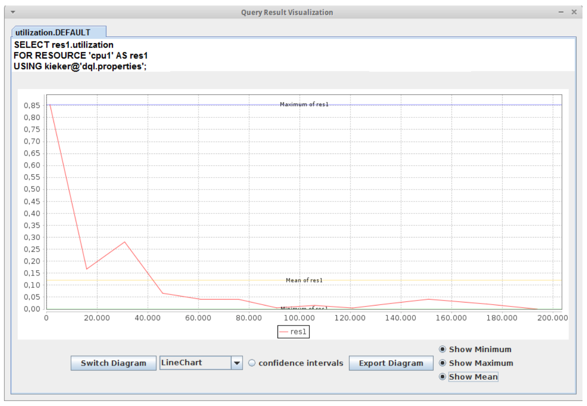
View So far, we focused on Eclipse integration. In the future, additional web interfaces for system operators are possible. For the `View` component our implementation uses JFreeChart to draw diagrams [1]. Among the common Java chart libraries, JFreeChart offers substantially more features and documentation compared to e.g., JOpenChart or Chart2D.

5 Integration in DQL

Following the DPE vision [6], the main evaluation goal is to demonstrate applicability in all stages of performance engineering. To evaluate the PAVO library we integrated it for result visualization of DQL queries. Due to the modular design, the integration caused low overhead. On result retrieval, the `Query Execution Engine` forwards results to PAVO as depicted in the component diagram below.



Exemplary, Figure 4 shows a visualization for measurements triggered by DQL using a JPetstore instantiation presented in [caseStudySubittedToSSP].



```
SELECT res1.utilization
FOR RESOURCE 'cpu1' AS res1
USING kieker@'dql.properties';
```

Figure 4: Visualization Sample

6 Conclusion and Future Work

This paper presents a visualization framework for quantitative analyses. The PAVO library provides ready-to-use visualization for quantitative performance analyses. The source code of the library is open-source and publicly available [1]. PAVO can be used as a building block for the DPE idea. It automates the visualization of performance analyses. The user specifies *what* he wants to know, receives a processed result, and PAVO automates *how* to visualize it. PAVO includes several features like an automatic choice of chart type, a diagram merge for composed visualizations, diagram type switch, enrichments, and image export. The future development of the library will focus on integrating additional visualization approaches, e.g., violin plots and a more sophisticated selection of visualization types from applicable ones.

Acknowledgments: This work is supported by the German Research Foundation (DFG) in the Priority Programme "DFG-SPP 1593: Design For Future—Managed Software Evolution" (KO 3445/15-1)

[1] `www.descartes.tools/dql`

References

[1] D. Gilbert. "The jfreechart class library". In: *Developer Guide. Object Refinery* 7 (2002).

[2] S. Frey et al. "MAMBA: Model-Based Software Analysis Utilizing OMG's SMM". In: *Softwaretechnik-Trends* 32.2 (2012).

[3] F. Gorsler, F. Brosig, and S. Kounev. "Performance Queries for Architecture-Level Performance Models". In: *Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering (ICPE 2014)*. New York, NY, USA: ACM, March 2014.

[4] F. Fittkau, S. Roth, and W. Hasselbring. "ExplorViz: Visual Runtime Behavior Analysis of Enterprise Application Landscapes". In: *Proceedings of the 23rd European Conference on Information Systems (ECIS)*. 2015.

[5] M. König. *Result Visualization for Performance Analyses.* Bachelor Thesis. 2016.

[6] J. Walter et al. "Asking "What?", Automating the "How?": The Vision of Declarative Performance Engineering". In: *Proceedings of the 7th ACM/SPEC International Conference on Performance Engineering (ICPE 2016)*. Mar. 2016.

[7] .

[8] I. A.-D. M. (Object Management Group. *Structured Metrics Meta- Model (SMM) , version 1.1.1.* Tech. rep.