HyLiMo: A Textual DSL and Hybrid Editor for Efficient Modular Diagramming

Niklas Krieger[1]

Abstract: Diagramming with precise layouting for scientific publications and technical documentations is time-consuming and cumbersome. Therefore, this work briefly presents HyLiMo, a tool for blended graphical and textual diagramming including live-synchronizing. This allows diagrammers to define diagrams textually and then adjust the layout graphically. An evaluation via two case studies confirmed the tool's practicality in creating class diagrams with precise layouts. However, feedback suggests several features for future work.

Keywords: Hybrid diagramming; Graphical-textual diagramming; UML class diagram

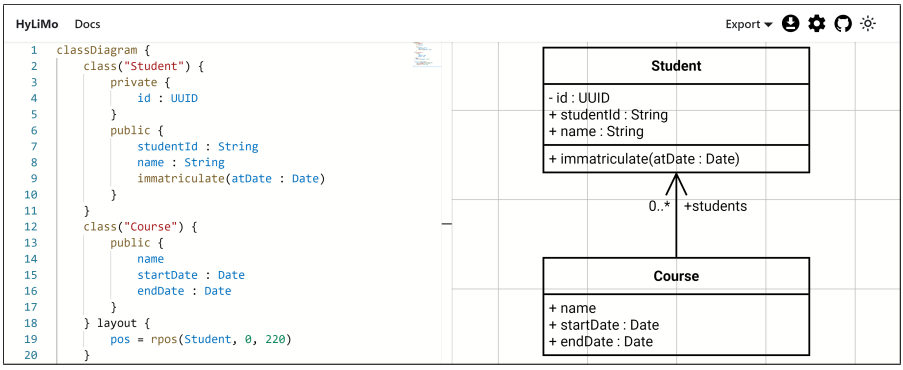

Fig. 1: Web-based hybrid graphical-textual editor.

Introduction and Goals: In Software Engineering, a variety of diagrams are used to create visual representations of systems, processes or concepts. In particular, UML class diagrams are a type of structural diagram used to model classes and their relationships in object-oriented systems. For diagramming, two main approaches exist: First, visual tools like diagrams.net and Visio allow graphically creating diagrams. Users place diagram elements on a canvas, creating the layout of the diagram manually. In contrast, tools like PlantUML and Mermaid are used to create diagrams using a textual syntax. For most such tools, the diagrammar does not specify the layout, instead, elements are auto-layouted. Therefore, this approach is insufficient when manual and precise layouting is required, e.g., for scientific publications and technical documentation. While textual tools supporting manual layouting exist, in particular TikZ UML, defining positions textually can be cumbersome and time-consuming. Hence, users benefit from graphical tools' ability to directly drag diagram elements to the desired location.

[1] University of Stuttgart , Institute of Software Engineering, niklas.krieger@iste.uni-stuttgart.de

doi:10.18420/sw2024-ws_19

Hybrid graphical-textual approaches allow us to bridge this gap. Recent research in modeling introduces the blended modeling concept, which uses multiple notations, including textual and graphical, to manipulate an underlying model [Ci19]. Depending on the task, modelers choose the view most beneficial, thus improving efficiency. Our goal is to bring the benefits of hybrid graphical-textual modeling to diagramming. In particular, users should be able to define diagrams textually initially, but then modify the layout graphically. To achieve this, we envision a hybrid live-synchronized editor, with side-to-side textual and graphical views.

Method: First, we collect and evaluate requirements with different stakeholders. Following, we worked out the concept in detail and created an architecture. After implementing the framework and editor, we evaluated the result in two case studies, as described in *Results*. After collecting initial requirements, to identify further requirements, we conducted expert interviews with 14 current and former software engineering researchers. Based on a survey, where 13 of them participated, we prioritised requirements, and finalized design decisions.

Concept: Fig. 1 shows the hybrid editor[2] created based on these requirements. It is split into two parts: On the left side, the diagram is defined using a textual notation. To improve flexibility and extensibility, we decided to incorporate programming language features, in particular custom functions and control flow constructs. Thus, we decided to implement our textual notation as an internal DSL in a custom general-purpose programming language, SyncScript. The textual notations serve as the single source of truth, including styling and layout information, allowing diagrammers to use tooling made for regular code, in particular version control to store and share diagrams easily. To give the user immediate feedback, textual and graphical views are live-synchronized. On each edit of the textual definition, the code is executed, and the diagram is updated if the execution is successful. As in graphical tools, users can manipulate the layout of the diagram by interacting with the graphical view. In particular, users can move diagram elements, or rotate such elements. On each edit, the textual definition is updated, and as a result, the diagram is rerendered. Yet, in practice, executing the code and generating the diagram introduced lag, worsening user experience. Consequently, we introduced predictions updating the graphical diagram before the code is executed. While the modular architecture of our framework allows us to support different diagram types in the future, currently, only UML class diagrams are supported.

Results: To evaluate our tool, we conducted two case studies in which a doctoral researcher and an undergraduate student created several class diagrams. Used features include custom styling, and a variety of diagram elements, including classes, associations, etc. Both case studies showed that our tool can be used in practice to create class diagrams with precise layouting. However, given feedback includes several feature requests, e.g., pdf export.

Bibliography

[Ci19] Ciccozzi, Federico et al.: Blended Modelling - What, Why and How. In: 2019 ACM/IEEE MODELS Companion. pp. 425–430, 2019.

[2] https://hylimo.github.io/