Proposal: Creating a web-based model transformation UI (with GLSP)

Florian Weidner
Philipps-University Marburg, Germany
Department of Mathematics and Computer Science, Software engineering group
May 20, 2025

I. Motivation and introduction

In software engineering, often Model-Driven Engineering (MDE) is used to increase development productivity and quality. Concepts are modeled closer to the domain so that they describe important aspects of a solution with human-friendly abstractions. The models can also be used to generate application fragments, that can be directly used as source code. In the process of MDE, many activities need to transform source models into different target models, while following a set of transformation rules. This model transformation process is based on algebraic graph transformations. A metamodel is used to model the structure and rules of the concept. The resulting transformation language can provide automatic model creation, development, and maintenance activities. [2] One framework to use MDE is Eclipse Modeling Framework (EMF) by the Eclipse Foundation. It provides a basis for application development, using modeling and code generation facilities. Many frameworks build upon EMF, providing various MDE tools like code generators, graphical diagramming, model transformation or model validation. [3] One model transformation framework is Henshin. [5] It tries to provide model transformation capabilities with a high level of usability. [4] For metamodels it uses EMF Ecore files and for instance models EMF XMI files. The framework enables transformations on XMI instance files using a defined transformation language. It provides a graphical and textual syntax to create these transformation rules. [5] Henshin can be used as an eclipse plugin. Eclipse makes it easy to access, but especially for new users, the heavy editor makes using Henshin unintuitive.

Therefore the goal exists to create a graphical option to use the Henshin model transformations without the overhead of the heavy eclipse editor. A web-based graphical editor would make using Henshin even more accessible and intuitive.

Graphical Language Server Platform (GLSP) is an open-source framework by the Eclipse Foundation to develop custom diagram editors for distributed web applications. [1] It can be used in Eclipse Desktop IDE, Eclipse Theia, Visual Studio Code, and embedded in any website. With these functionalities, GLSP fits to create an accessible, intuitive application to create and apply Henshin model transformations.

II. Project Requirements

The following requirements are a list of the functional and non-functional requirements (that I came up with). The client should be integrated as an Eclipse Theia client. Additional clients can be added in the future without impacting other parts of your implementation. [**eclipseGLSP**] The backend should be implemented with Java.

Functional requirements:

- EMF XMI instance files should be displayed in a graphical editor
- Henshin rule files should be displayed in a graphical editor
- EMF Ecore metafiles should be displayed in a graphical editor
- The instance editor should display all rules that can be applied to the instance model.
- Parameters of the rules should be editable when applying a rule.
- After applying a rule, the instance model should be updated and displayed in the instance editor as a temporal file that can be used to apply multiple rules. The initial instance model should not be changed.
- The instance editor should provide editing functionality for the instance model.
- The Henshin rule editor should provide editing functionality for the transformation rules.
- The Ecore editor should provide editing functionality for the Ecore metamodel.

Once all functional requirements are implemented, the application should fully support a basic model transformation workflow. Its functionality should be equivalent to using the Henshin plugin within the Eclipse editor. It should be possible, that additional Henshin functionalities like State Space analysis or conflict and dependency analysis can be added in the future.

Non-functional requirements:

- The application should be web-based and accessible via a web browser.
- The application should be responsive and work on different screen sizes.
- The application should be user-friendly and intuitive to use.
- The application should be performant and handle large models efficiently.

III. Basic architecture

GLSP uses a client-server architecture. The client and server communicate via a websocket connection and JSON-RCP. The backend will be implemented in Java, which is the main programming language for EMF and especially the Henshin SDK. The GLSP backend supports integration with EMF models as the underlying source model for the diagrams. That makes the integration of Henshin easier because all files needed are based on EMF. The *HenshinRessourceSet* can be loaded directly over the EMF integration of GLSP into the *ModelState*. For the XMI instance files, the Henshin rule files, and the Ecore metamodel files, a mapping to the GLSP internal graphical model needs to be implemented. [**eclipseGLSP**]

For providing editing functionality, all commands need to be implemented as Actions and Handlers in the GLSP backend. They are manipulating the source files directly. The GLSP architecture is built that after manipulating the source model, the new state is mapped to the graphical model of GLSP again and then displayed in the client. [**eclipseGLSP**]

The GLSP client is implemented in TypeScript. It uses Sprotty, an SVG-based diagramming framework, to render the diagrams. Different diagram languages can be defined for multible file types. For each file type, a different backend module must be registered. The ecore metamodel, the Henshin rule file and the XMI instance files should be in the same folder in a workspace. There you can open the corresponding file and EMF model should be displayed in a graphical editor.

IV. Preliminary table of contents

The preliminary table of contents is on the full next page.

CONTENTS

V. Time schedule

The goal is to finish the project and the thesis at the end of August. From now on that includes 14 weeks. The three main tasks are the development of the application, writing the scientific work about the thesis, and writing the final thesis. Here is a list of the main tasks that need to be done with the estimated time needed and the planned completion date.

TABLE I
PROJECT DEVELOPMENT TASKS

No.	Description	Estimated Time Needed	Planned Completion
1	EMF XMI instance files should be displayed in a graphical editor	2 weeks	24. Mai
2	The instance editor should display all rules that can be applied to the instance model.	1 week	31. Mai
3	Parameters of the rules should be editable when applying a rule.	1 week	31. Mai
4	After applying a rule, the instance model should be updated and displayed in the instance editor as a temporal file that can be used to apply multiple rules. The initial instance model should not be changed.	1 week	31. Mai
5	The instance editor should provide editing functionality for the instance model.	2 weeks	14. Juni
6	Henshin rule files should be displayed in a graphical editor	2 weeks	28. Juni
7	The Henshin rule editor should provide editing functionality for the transformation rules.	1 week	28. Juni
8	EMF Ecore metafiles should be displayed in a graphical editor	1 week	?
9	The Ecore editor should provide editing functionality for the Ecore metamodel.	1 week	?

TABLE II
THESIS WRITING TASKS

No.	Description	Estimated Time Needed	Planned Completion
1	Writing basics chapter about MDE and model transformations	1 week	07.Juni
3	Writing basics chapter about EMF and the Eclipse Foundation	1 week	14.Juni
4	Writing basics chapter about Henshin and GLSP	1 week	21.Juni
5	Writing related work and existing tools	1 week	05.Juli
6	**Finishing scientific work**	1 week	12.Juli
7	Writing requirements chapter	1 week	19. Juli
8	Writing System Design and Architecture chapter	2 weeks	02. August
9	Writing Implementation chapter	1 week	09. August
10	Writing Testing and Evaluation 4 chapter	1 week	16. August
11	Writing Discussion chapter	0.5 week	23. August
12	Writing introduction and conclusion chapters	0.5 week	30. August

References

[1] Philip Langer and Tobias Ortmayr. *Eclipse GLSP*. https://github.com/eclipse-glsp/glsp. 2025.

[2] S. Sendall and W. Kozaczynski. "Model transformation: the heart and soul of model-driven software development". In: *IEEE Software* 20.5 (2003), pp. 42–45. DOI: 10.1109/MS.2003.1231150.

[3] David Steinberg et al. *EMF: Eclipse Modeling Framework 2.0*. 2nd. Addison-Wesley Professional, 2009. ISBN: 0321331885.

[4] Daniel Strüber et al. "Henshin: A Usability-Focused Framework for EMF Model Transformation Development". In: *Graph Transformation*. Ed. by Juan de Lara and Detlef Plump. Cham: Springer International Publishing, 2017, pp. 196–208. ISBN: 978-3-319-61470-0.

[5] Daniel Strueber. *Henshin*. https://github.com/eclipse-henshin/henshin/wiki. 2025.