GRAPE: A Guiding RML Authoring Projectional Editor in Action and User's First Impressions

Jakub Duchateau[1], Christophe Debruyne[1]

[1]Montefiore Institute, University of Liège, Belgium

Abstract

RDF Mapping Language (RML) facilitates Knowledge Graph construction but presents significant authoring challenges. GRAPE, an open-source projectional editor, aims to simplify RML authoring. GRAPE offers a guided, text-like interface preventing syntax errors while maintaining RDF flexibility. We showcase its core features, including its language-oriented design and specialized notation. A user study with 20 MSc students revealed diverse user preferences: some felt reassured by the guided approach, while others preferred traditional text editors. These findings inform future GRAPE development.

1. The Challenge: RML Authoring for all

Data integration is an essential activity in creating Knowledge Graphs (KGs), often requiring collaboration between domain experts and KG engineers. The RDF Mapping Language (RML) is a standard for this data transformation task; however, its graph-based nature and Turtle syntax create a steep learning curve for both domain experts and KG engineers. Moreover, we hypothesize that KG engineers often prefer the freedom provided by textual editors.

This paper demonstrates the use of GRAPE [1], a tool designed to find a compromise between a guided authoring experience for domain experts whilst retaining the flexibility of RML and RDF authoring for advanced users and use cases. The goal is to provide a tool that caters to a broad audience.

2. The GRAPE Approach: Main Features and Innovations

As reported in GRAPE [1], existing approaches rely either on text editors allowing one to write arbitrary RDF graphs or GUIs for authoring mappings, thus constraining the permitted constructs. GRAPE introduces a novel editing experience for RML by using a language-oriented approach with a projectional editor, and provides guidance to ensure well-formed RML, without the rigid constraints often found in many visual (e.g., GUI) tools.

- Projectional editing: Users manipulate a projected Abstract Syntax Tree (AST) representing the RML vocabulary (triples map, term map, etc.), not the text itself, which contrasts with tools like YARRRML (https://w3id.org/yarrrml/) . Contextual actions (move, create) are used to change the AST, preventing syntax errors as only structurally valid edits are possible.
- Composable language ecosystem: Because GRAPE is built on a language-oriented foundation, it is not limited to RML. The editor can be extended to support other vocabularies, or RML extensions, as RML itself is modularized. GRAPE allows users to mix RML with generic Turtle, or even dedicated syntaxes for OWL or SHACL, all within a single document. While this extensibility requires implementing each new language's structure within the projectional framework, it enables a unified authoring experience. While RMLEditor (https://rml.io/tools/rmleditor/) can be argued to be projectional, it does not provide, to the best of our knowledge, the capability of being extended.

Cooperative Information Systems - Early Research Achievement and Demos, 2025.
Jakub.Duchateau@uliege.be (J. Duchateau); C.Debruyne@uliege.be (C. Debruyne)
0009-0009-5090-8192 (J. Duchateau); 0000-0003-4734-3847 (C. Debruyne)

- Novel, extensible notation: GRAPE uses a more explicit, readable notation for RML constructs, inspired by the clarity of languages such as YARRRML. For example, function calls use a familiar `function(parameter=value)` syntax. Because it is a projectional editor, a concept can have multiple notations, and they can be extended with hints or altered without modifying the core language. YARRRML and RML in Turtle are embedded DSLs (also known as internal DSLs), YARRRML because it is based on YAML, and RDF as it is the use of a vocabulary inside a Turtle document. In contrast, GRAPE offers an external DSL, which means it is a stand-alone language.

GRAPE furthermore provides context-aware auto-completion and suggestions, guiding the user through the mapping process. However, unlike many wizard-based tools, it does not lock users into a rigid workflow, retaining the ability to create complex mappings and embed arbitrary RDF. This is achieved not only by analyzing the AST, but also by processing the vocabularies used in the mappings.

GRAPE is not just an editor but an integrated environment. Users can execute RML with BURP[1] from within the tool, allowing the testing of mappings in the mapping authoring process. GRAPE also provides support for importing and exporting RML in Turtle format.

As for its maturity level, GRAPE is a functional prototype developed based on user feedback, including the study presented in this paper. It currently supports RML-Core and some RML-FNML and RML-IO.

3. See GRAPE in Action

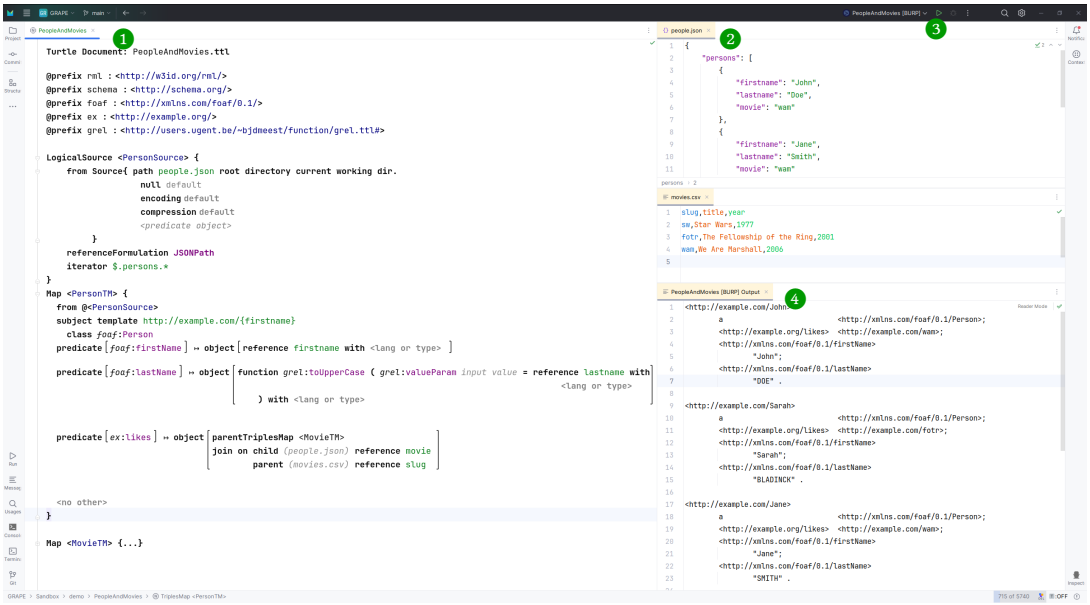


Figure 1: GRAPE Editor Screenshot: ❶ The turtle projectional document with turtle and RML mappings, in the right panel the input file ❷ and once run with ❸ the output RDF ❹ can be opened for analysis.

As for its availability, GRAPE is open-source and can be downloaded and explored from its Git repository: https://gitlab.uliege.be/JakubDuchateau/GRAPE. GRAPE is available with an EUPL-1.2 license. A tutorial demonstrating the workflow in GRAPE, from loading a data source to generating an RDF graph, is available at https://jakubduchateau.gitlabpages.uliege.be/GRAPE. A short demonstration video is available at https://kutt.to/grape-coopis-demo.

4. First Impressions of GRAPE via a User Study

To evaluate GRAPE's usability and gather initial feedback, we conducted a user study with 20 MSc students in Computer and Data Science (S1 to S20) enrolled in a course where (R2)RML is covered. The

[1]BURP https://github.com/kg-construct/BURP (version 0.1.2)

study was designed to observe how users introduced to RML would interact with GRAPE's editing paradigm. Students had exercises on RDF prior to the (R2)RML module.

The study adopted the protocol proposed by Crotti Junior and Debruyne [2]. The process involved several steps: a pre-survey to gauge initial knowledge and motivation, a short tutorial on GRAPE, solving 5 mapping tasks, and a post-study questionnaire to assess perceived usability and cognitive load. We requested that, for each mapping task, participants assess the difficulty using a 7-point Likert scale. Three additional participants (U1 to U3), who had previously completed the course, were then recruited. In addition to the original protocol, these participants performed the five tasks in a think-aloud setting, completed a sixth task using RML-FNML, and participated in a follow-up interview for richer qualitative data. Participants were informed that the lead author of this paper was the main developer of the tool.

4.1. Key Qualitative Findings

Whilst the protocol allows for some limited quantitative analyses, we will focus on preliminary qualitative findings due to space limitations. The following observations are based on an analysis of qualitative user feedback, using open coding and affinity diagramming to identify themes and patterns.

Participants' observations about the protocol, as construed for this experiment, were a lack of clarity in the provided tutorial (S2, S3, S6, S10), and stress relating to not knowing whether the mapping was correct to move to the next task (U1).

A central theme emerged from the analysis: a tension in the user experience regarding the projectional editing paradigm. Some participants found the guided approach reassuring. The editor's structured nature prevented syntax errors, allowing them to focus on the mapping logic rather than the syntax. As one participant noted, the tool was "More constrained but with clear error messages" (U1), while another found they "didn't have much syntax errors as the tech was taking care of it all" (S12) or they "really enjoy the placeholder feature allowing us to fill the missing pieces of the puzzle" (S7).

Conversely, participants also expressed frustration with the unfamiliar editing model offered by projectional editing. Accustomed to text editors, they found the restricted cursor movement and editing capabilities unintuitive. One remarked, "I feel like the interface is not very intuitive [...] in a text editor environment it is expected that we have full control on the cursor position [...] and being restricted on that seems frustrating" while also recognizing "suggestions and completions are indeed useful" (S19). This sometimes led to inefficient workarounds, with one student explaining they "needed to delete the entire line an[d] (sic) restart" (S6) because they could not discover a more direct editing method. And deletion in itself was mentioned as unintuitive, e.g., "sometimes huge chunks being deleted in a blink of an eye" (S7). These observations align with known adoption challenges for projectional editors [3].

We speculate that users had to navigate distinct learning curves simultaneously: the concept of projectional editing, the application of RML, and GRAPE's specific notations (the RML and Turtle DSLs). While not a critique of GRAPE per se, the output of BURP is n-triples, and some found the use of turtle (input) and n-triples (output) confusing, which was easily addressed in the current version.

Beyond these general sentiments, several usability issues were observed:

- Users struggled with constants in expression maps, particularly language maps and datatype maps. Users often attempted to add these as constants as "values" instead of selecting the appropriate expression map. After identifying the problem with the students, we solved it by automatically constructing the correct expression map, depending on whether an IRI or a string was entered for the second group. The second group no longer appeared to have the issue.
- Users were able to change the identifier of a triple map from within a referencing object map. Some found this confusing as they expected to change the reference to another triple map. With this feedback, we limited the option to rename a triples map only in its definition.
- The tool supports the provision of hints to combine sources in referencing object maps, but that feature had to be enabled. When it was activated for some students, those hints were reported "it instantly became very easy. Before that, I wasted a lot of time trying to understand why the program~~me~~ (sic) couldn't find attributes that I thought should be findable"(S13).

- It was surprising to observe that students preferred copying and pasting (parts of) triple maps, and subsequently adapting them, instead of using autocompletion or the editor's other contextual actions. This may highlight issues with discoverability or a mismatch of the mental model of the editor. For instance, S15 noted, for some tasks, they "didn't understand how I had managed to do them," and previously, "I don't know how parentTriplesMap appeared, I wasn't trying to make it appear when it did."

- Participants reported that the error messages reported by the engine were lacking a position within the source file, and they had to infer where the error was. They also noted that the errors are difficult to understand and are only shown at compile time (effectively run-time). This problem does not stem from GRAPE, but its underlying RML engine.

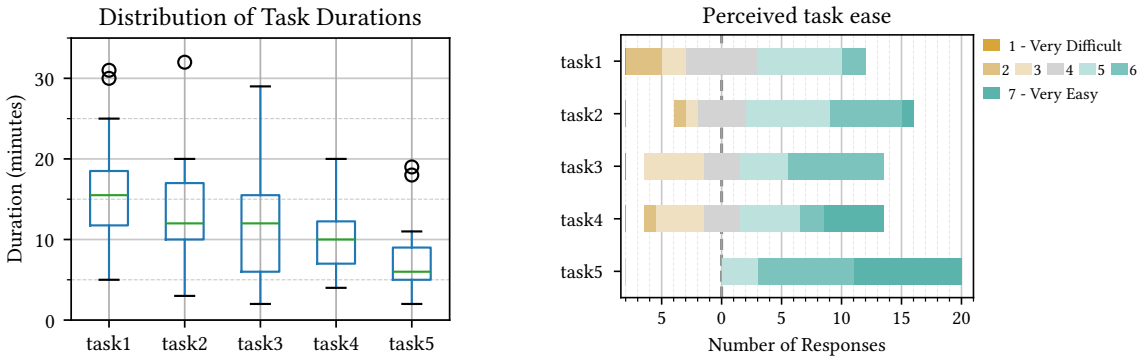
We recognize that these students are not representative of what we would deem domain experts. Even with this limitation, this study was useful for identifying bugs that have since been addressed, as well as usability issues that are being improved.

4.2. Quantitative Insights

The quantitative data seem to support the qualitative findings. Figure 2a shows the learning curve; the mean time to complete tasks decreased as participants gained familiarity with the editor. Task 3, which introduced the first join condition, shows the highest variability, reflecting various usability issues related to referencing object maps. However, tasks 4 and 5, which were similar in complexity to 2 and 3, were completed much faster. This suggests a learning curve that flattens out with practice. As participant S3 noted, "slow at the beginning, fast at the end with repetitive tasks" (S10) and "once you get used to it, it is much easier and faster to perform tasks compared to the previous [R2RML module]".

The perceived ease of each task, rated on a 7-point Likert scale, reflects a similar progression (Figure 2b). Task 1 was rated as the most difficult, aligning with one student's comment: "I think the difficulty mainly came to discovering how to apply what I had just learned to a new interface" (S7). In contrast, Task 5 was found to be (very) easy by all participants, demonstrating that once the initial hurdles were overcome, the tool became effective for them: "Getting used to the tool and I start to see its interest: very easy to use and intuitive!" (S8).

6.26894in



(a) Mean task completion times with standard deviation. Task 3 shows high variability, but subsequent tasks are faster, indicating a learning effect.

(b) Distribution of perceived ease for each task. Task 1 was perceived as most difficult, while Task 5 was rated as easy by most.

Figure 2: User Study Quantitative results of first round with 20 students.

While GRAPE's projectional nature prevents syntactical errors, the study also revealed several instances of semantic or logical errors in the final mappings submitted by the participants. While all mappings were runnable, out of the 20 student participants, 7 submitted mappings with errors. These included conceptual misunderstandings of RML, such as one participant (S11) not using parent triples

maps for joins, and another (S13) omitting datatypes. Other errors were minor, such as two participants (S18, S19) making the same mistake in a class definition, or three more (S8, S12, S20) having spelling mistakes in IRIs or using incorrect namespaces. Further study is warranted to ensure that these errors are due to the participants' comprehension of the task or the tool being misleading.

Threats to Validity All participants of the study were computer science students enrolled in a course on Knowledge representation and reasoning. Three facilitators were present in the class to assist participants with technical issues, but ended up answering questions about RML's principles as well.

5. Discussion and Future Directions

The study confirmed that GRAPE, in its current form, presupposes a foundational understanding of RDF and RML concepts. A key takeaway is the need to better support users in forming a high-level mental model of the target mapping, as the editor's step-by-step guidance can sometimes obscure the overall structure. Notwithstanding these observations, participants were able to generate well-formed RML even though they had little experience with this mapping language. Features that were appreciated were the autocompletion of both mapping structures (i.e., "the skeleton") and the values (e.g., the use of predicates in predicate maps).

Looking ahead, two primary directions emerge. First, it is crucial to conduct a similar study with the tool's other target audience: domain experts. Evaluating how non-developers approach an editor that retains RML's conceptual complexity but abstracts away the syntax will be critical. Second, the strong preference of some for textual programming suggests that a one-size-fits-all approach may be insufficient. This highlights the need to decouple GRAPE's core architectural benefits, such as language composition, from the projectional editing interface. Future work should investigate delivering these features through alternative paradigms (e.g., enhanced text editors or GUIs) to broaden adoption. Furthermore, exploring more intelligent, user-driven assistance, such as through contextualized AI-powered suggestions [4], could offer a promising path to bridging the usability gap for all user groups.

Acknowledgments

This work was supported by the Fonds de la Recherche Scientifique – FNRS under Grant n° MIS F.4016.24.

Declaration on Generative AI

During the preparation of this work, the authors used Gemini and Grammarly to improve grammar and spelling check, paraphrase and reword, abstract drafting. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] J. Duchateau, C. Debruyne, GRAPE: Guiding RML authoring with a projectional editor, in: [5], 2025. URL: https://ceur-ws.org/Vol-3999/paper3.pdf.

[2] A. Crotti Junior, C. Debruyne, A Protocol for KG Construction Tasks Involving Users, in: [5], 2025. URL: https://ceur-ws.org/Vol-3999/paper1.pdf.

[3] T. Berger, M. Völter, H. P. Jensen, T. Dangprasert, J. Siegmund, Efficiency of projectional editing: A controlled experiment, in: Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2016, Association for Computing Machinery, New York, NY, USA, 2016, pp. 763–774. doi:10.1145/2950290.2950315.

[4] J. Stütz, O. Karras, A. Oelen, S. Auer, A user-driven hybrid neuro-symbolic approach for knowledge graph creation from relational data, in: Cooperative Information Systems - 30th International Conference, CoopIS 2024, Porto, Portugal, November 19-21, 2024, Proceedings, volume 15506 of *LNCS*, Springer, 2024, pp. 169–185. doi:10.1007/978-3-031-81375-7_10.

[5] D. Chaves-Fraga, I. Dasoulas, C. Debruyne, A. Dimou, U. Serles, D. V. Assche (Eds.), Proceedings of the 6th International Workshop on Knowledge Graph Construction, number 3999 in CEUR Workshop Proceedings, Aachen, 2025. URL: http://ceur-ws.org/Vol-3999.