HoWCoM Case Study - TGRL Tool

Rijul Saini and Gunter Mussbacher

October 11 2021

1 Prerequisites

During the hands-on sessions, participants have two options to build TGRL models using our tool, i.e., TGRL (Textual Goal-oriented Requirement Language) Visual Studio (VS) Code extension [1]. First, participants can use VS Code desktop-based application to create or join a collaboration session. Second, participants can join the collaboration session using web-based VS Code views in a browser. We highly recommend participants to use the first option due to the limited support available in web-based VS code views in a browser. For example, the code highlighting feature and visualization of corresponding diagrams are not supported in a browser. However, participants will be still able to edit the textual TGRL models collaboratively using the second option.

In this section, we provide the steps to create or join a collaboration session. Next, we provide excerpts from the TGRL grammar so that participants can refer them during the hands-on session to build TGRL models. For further details, participants can refer to the GitHub repository of TGRL grammar.

1.1 Steps for Installation

1. Install the desktop-based VS Code application on your system from the VS Code site.

2. Download our tool in the form of vsix file (vscode-xtext-turn-0.0.2.vsix) from the GitHub repository of our tool. We provide further details in our repository to develop or debug the extension.

3. Open a Terminal to create a project run directory (workspace) and launch VS code extension using the below commands:

   ```
   mkdir workspace
   cd workspace
   code .
   ```

4. Install the downloaded VSIX file (vscode-xtext-turn-0.0.2.vsix) of our proposed TGRL extension in VS Code by using shortcut key CTRL + SHIFT + X or navigating in menu bar:

   View → Extensions → Install from VSIX... → Select the downloaded vsix file.

5. Install the VS Code Live Share extension using the "Search extensions in Marketplace" or using the below command after using CTRL + P (VS Code Quick Open) shortcut key:

   ```
   ext install MS-vsliveshare.vsliveshare-pack
   ```

6. Click on the "LiveShare" option in the bottom left corner of VS Code to start a collaboration session.

7. Modeller can create TGRL Models (file with .turn extension) collaboratively. For quick start, we have provided a sample TGRL model "example.turn" in the "examples" folder of the repository.

1.2 Excerpt from TGRL Grammar [2]

```
URNspec:
'urnModel' name=QualifiedName
info=(ConcreteURNspec)? &
actors+=Actor*

Actor:
'actor' name=QualifiedName
longName=LongName '{'
('importance' (importance=ImportanceType |
importanceQuantitative=QuantitativeValue))?
elems+=IntentionalElement*
'}';

IntentionalElement:
type=IntentionalElementType name=QualifiedName
longName=LongName '{'
('importance' (importance=ImportanceType |
importanceQuantitative=QuantitativeValue))?
('unit' unit=STRING)?
linksSrc+=ElementLink*
'}';

ElementLink:
Contribution | Decomposition | Dependency;

Contribution:
(name=QualifiedName longName=LongName)?
'contributesTo' dest=[IntentionalElement|QualifiedName]
(correlation?='correlated')? 'with' (contribution=ContributionType |
quantitativeContribution=QuantitativeValue);
```

2 Goals

In this section, we present our research questions which we have improved based on reviewers' feedback and the overall objectives of the HoWCoM workshop.

1. Which editors or IDEs are preferred by participants for performing modelling and development together?
2. What is the satisfaction level of participants for handling conflicts with our proposed tool?
3. What is the performance of the collaborative environment facilitated by our solution in terms of scalability and time?
4. What other potential features and use case scenarios are possible with our proposed tool?

3 Tasks

In this section, we present the tasks which we identify to answer the above research questions and to evaluate the features supported by our tool.

1. Create X collaboration sessions based on the total number of groups of participants, i.e, X collaboration sessions for X groups. One member in each group leads the collaboration session by creating a live share collaboration session and sharing the URL of the created session with other participants through any communication medium such as email and slack. While accepting participants' requests to join the session, access model can be changed from read-only to write or vice-versa.
2. After joining the collaboration session, each participant joins the chat group in Live Share inside VS Code. In addition, each participant ensures that the TGRL VS code extension is installed on their systems (replicas).
3. Participants verify the error messages due to empty TGRL models (validation). In addition, participants verify that auto auto-suggestions are populated by our tool by using CTRL + SPACE keys.
4. Participants give a name to the URN model (urnModel <name>) and assigns responsibilities in the chat widget for different actors and intentional elements.
5. Create Y actors where each participant is assigned to create at least one actor by referring to the excerpt of TGRL grammar provided in Section 1.2. For quick start, participants can use the examples provided in the "examples" folder of the repository.
6. Create Z intentional elements where each participant is assigned to create at least one intentional element by referring to the excerpt of TGRL grammar provided in Section 1.2.
7. Participants select the name of an actor (not the "longName"), right click using mouse to open the context menu, and click "Generate Diagram". In case, "Generate Diagram" option is clicked without selecting the name of an actor then a pop-up will open where participants can input the name of actor. Participants verify that the diagrams are synchronized with the textual models.

8. Participants can build models individually yet collaboratively (different model views based on actors). While building models, participants verify the cursor positions of other participants. In addition, participants can focus their attention on a particular participant.

9. Participants can perform delete and undo operations by selecting a part of the model to ensure the recovery of the desired state of their models.

4 Questionnaire

In this section, we present the questionnaire to evaluate the above research questions and to assess the benefits and limitations of our tool.

Participants are requested to provide comments to the below questions.

1. Which existing features require further improvement?

2. Which other features can be supported by our tool?

3. Which other tools you are aware of that provide similar or better support for the mentioned features?

Participants are requested to give a score to the below questions based on their experience on the scale of 1 to 5: 1 (Very unsatisfied), 2 (unsatisfied), 3 (neutral), 4 (satisfied), and 5 (very satisfied).

1. How is your experience in downloading and installing our tool in the form of a vsix file?

2. How is your experience in launching and joining a collaboration session in VS Code?

3. How is your experience in locking the models by changing the access mode of participants to read-only from write access?

4. How is your experience in using the chat channel inside VS Code during collaboration?

5. How is your experience in following the cursors of other participants and in using the focus feature while building TGRL models?

6. How is your experience in building the textual models using the auto-completion feature?

7. How is your experience in selecting or providing the actor name in the pop-up while generating their corresponding diagrams?

8. How is your experience in observing the changes made to the textual models are synchronized in corresponding diagrams simultaneously?

9. How is your experience while performing the delete and undo operations for recovering the desired state of models?

10. How is your experience while observing the changes made by other participants in textual models and their corresponding graphical models?

11. How is your experience in working on individual model fragments based on actors as well as their own graphical views?

12. How is your experience in resolving conflicts manually by using the chat communication channel or in avoiding conflicts by following the cursor positions of other participants?

13. How is your experience in identifying and managing users during collaboration?

References

[1] R. Saini and G. Mussbacher, "Towards conflict-free collaborative modelling using vs code extensions," in *2021 ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C) (to be published)*, 2021.

[2] R. Kumar and G. Mussbacher, "Textual user requirements notation," in *International Conference on System Analysis and Modeling*. Springer, 2018, pp. 163–182.