TDF Tool

Agent Modelling User Guide
Version 0.3

1 Release Notes – v0.3

This is an early release of the TDF Tool.

2 Introduction to the TDF Tool

TDF[1] [1] is a methodology and tool that facilitates the engineering of dynamic decision-making systems. The modelling of dynamic decision making is important in a number of application areas including (i) capability analysis (simulation of tactical scenarios in order to identify gaps in capability); (ii) training (realistic simulation of non-player characters in a virtual environment); (iii) tactics development and evaluation (using decision-making models in a constructive simulation context to explore tactical options); (iv) autonomous systems (unmanned vehicles whose domain-specific decision-making competence needs to be on a par with that of humans); and (vi) human/machine teamwork (agents that can explain their behaviour at the level of goals and intentions). In developing TDF, we were interested in those particular application areas and so have focused on how to engineer effective and understandable dynamic decision-making systems.

TDF is grounded in the **BDI** (Belief, Desire, Intention) paradigm [3], as this is an intuitive model of human reasoning. The BDI model is a particularly parsimonious conception of rational agency, characterising agents as reasoning in terms of their *beliefs* about the world, *desires* that they would like to achieve and the *intentions* that they are committed to. Apart from its intuitive appeal to domain experts, it is a powerful computational abstraction for building sophisticated, goal-directed and reactive reasoning systems, and consequently is well suited to modelling decision making in dynamic environments.

Put very succinctly, a BDI agent performs a continuous loop in which it updates its beliefs to reflect the current state of the world, deliberates about what to achieve next, finds a plan for doing so, and executes the next step in that plan. Each time around this cycle, it effectively reconsiders its options, yielding goal-oriented behaviour that is also responsive to environmental change. Not only does the BDI model map well to how we think we think, making it an intuitive medium for conceptualising tactical decision making, it has been extensively studied in theoretical computer science, and expressed in mathematical logic, which means that the properties of BDI models can be formally verified; an important requirement for autonomous decision-making systems.

TDF is based on over 25 years of experience in the agent-based modelling of decision making and derives from Prometheus [2], a prominent **AOSE** (Agent Oriented Software Engineering) methodology. Prometheus and other AOSE methodologies are concerned with how to specify, design, implement, validate and maintain agent-based systems. However, from the perspective of modelling dynamic decision making, no single AOSE methodology collectively tackles knowledge elicitation, dynamic goal-oriented control structures or the expression of tactics at a high level of abstraction. TDF is the first methodology to focus on the full development cycle required to model dynamic decision making, from knowledge elicitation through to implementation. Note that, although TDF is primarily focused on the elicitation and modelling of **tactics**, it can also be applied to the modelling of more general behaviour that is not necessarily tactical.

So, what do we mean by **tactic**? Military science definitions of the term, tactic, tend to emphasise the adversarial aspect. More generally, we view tactics as:

[1] Tactics Development Framework

> ...the means of achieving an immediate objective in a manner that can be adapted to handle unexpected changes in the situation.

Tactics are distinguished from **strategy** in that the latter is concerned with a general approach to the problem, rather than the *specific* means of achieving a more short-term goal in a dynamic environment. A submarine commander's use of stealth to approach a target is a strategy, whereas the particular method used, for example hiding in the adversary's baffles (a blind spot), is a tactic. Tactics are concerned with the current, unfolding situation – that is, how to deflect threats and exploit opportunities to achieve one's objective. This view of tactics, as the means of achieving a short-term goal in a manner that can respond to unexpected change, seems to be common to all definitions, whether in military science, game theory, or management science.

3 Modelling in TDF

In keeping with its intended purpose as a practical methodology for modelling dynamic decision making, TDF addresses four main aspects of modelling:

- **Process.** Guidelines on the sequence of steps to be followed, what should happen in each step, and each step's purpose.
- **Artefacts and Relationships.** A TDF model comprises a number of **artefacts** and the relationships between them. Artefacts are the *named* elements that make up a model, for example, the agents, goals and tactics. A model should define the relationships between its artefacts, for example, the fact that an agent *has* goals that it can tackle and tactics it can *use* to achieve those goals.
- **Diagrams and Iconography.** The TDF methodology encourages the use of specific types of diagram that offer important views of the model. Each type of diagram can be constructed from a particular set of icons (representing artefacts and nodes) and those icons can be connected together by arcs that show the relationships between them.
- **Route to Implementation.** It is all very well having a methodology for modelling decision making, but it is difficult to develop, verify and maintain the models without software tool support. Furthermore, for engineering applications, it is useful to automate aspects of the mapping from the model to executable code.

4 TDF Stages

The TDF methodological process divides the specification and design of tactics into three main stages: **Requirements**, **Architecture** and **Behaviour**, and as you will see shortly, this structure is reflected in the layout of the **Project Overview** sidebar on the lefthand side of TDF's main window.

> Note that, although the three stages tend to be performed in sequential order, in practice one can switch back and forth between stages.

The Requirements stage can involve knowledge elicitation from domain experts along with the specification of system inputs and outputs, the main **goals** (objectives) the decision-making system will have to handle, and what high-level **strategies** it can use to achieve those goals. The Architecture stage involves specifying the roles and entities (agents) that were identified during knowledge elicitation. Finally, in the Requirements stage, the decision-making behaviour of the agents is specified. This comprises the **tactics** and **plans** the agents use to achieve their goals.

5 Overview of the Interface

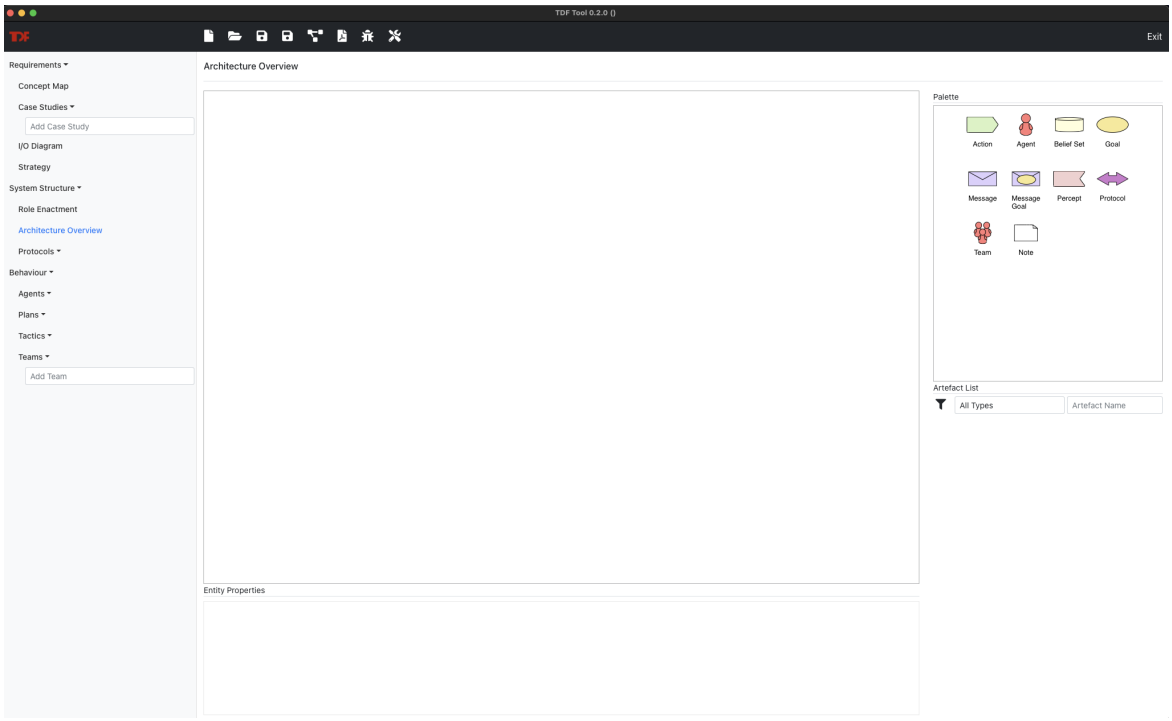

Fig. 1 Main TDF window

After launching TDF, you will see the window shown in Figure 1. This is a macOS screenshot; naturally your window borders will look different if you are running TDF on Linux or Microsoft Windows, but the window contents will be the same.

Note: TDF does not currently provide any **undo** functionality. Therefore, before you make any significant changes to your project, it is advisable that you save a version that you can revert to if you are not happy with the changes you have made.

5.1 Task-Specific Regions

Most of your interaction with the application will be through the task-specific regions of the main window (Figure 2). The central **Canvas** region is where you will create, view and edit your TDF diagrams. Each diagram will comprise a set of icons representing its entities, and arcs between the icons representing the interrelationships between those entities.

Using TDF, you will be creating a **model** of the behavioural system you want to engineer. The various TDF diagrams offer key views of the model you are creating, for example, the **I/O Diagram** shows the major interactions between the system and the environment it is embedded in. You can access the diagrams in the **Project Overview** sidebar. The various types of diagram are introduced and explained later in this document.

The **Entity Palette** is used to add icons to the Canvas, and the palette of icons changes depending on which type of diagram you are editing. To add an entity to the Canvas, drag it from the Entity Palette and drop it on the Canvas; you can then name the entity. Entities can have properties, and these can be viewed and edited in the **Entity Properties** pane at the bottom of the window. You can select artefacts to edit in the dropdown menu at the top of the **Artefact List** sidebar.

The **Toolbar** at the top of the window provides access to global functions such as loading/saving a TDF project file and modifying application settings. The **Canvas Title** shows which diagram you are editing.

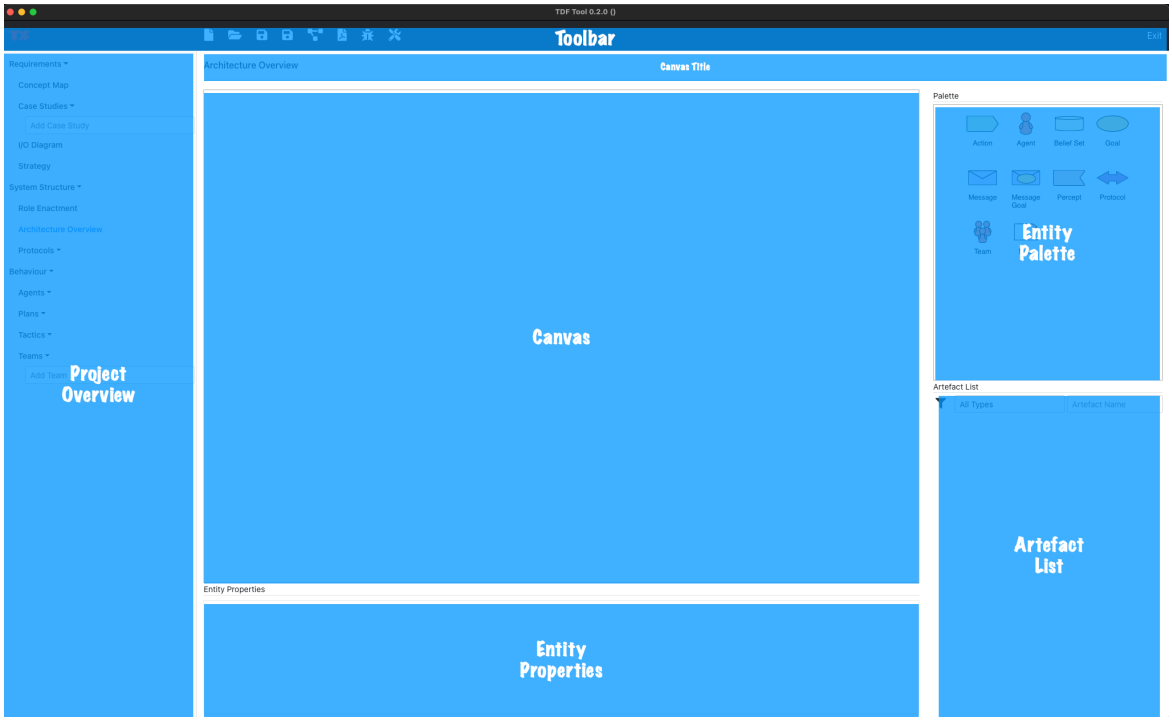

Fig. 2 Main window regions

6 The Toolbar

The Toolbar is a strip of buttons along the top of the TDF window. It provides access to a variety of commonly used functions. An Exit button can be found on the righthand side of the Toolbar; it quits the application.

The following button icons can be found on the lefthand (Figure side of the Toolbar (see Figure 3):

- **New Project.** Clears all existing diagrams and starts a new project.
- **Open Project.** Opens a project file.
- **Save Project.** Saves the project to the current file.
- **Save Project As.** Saves the project to a new file.
- **Auto Layout.** Automatically reformats the diagram.
- **Print to PDF.** Outputs to a PDF file for report generation, etc.
- **Show Console.** This is only available in special builds of the tool; it opens a console window for debugging.
- **Regenerate all Diagrams from Model.** This button can be used to rebuild the diagrams from the internal TDF model.

Note that TDF has an autosave feature that saves the project periodically. Backups are saved to the location of your saved project, in a folder matching the name of your project.

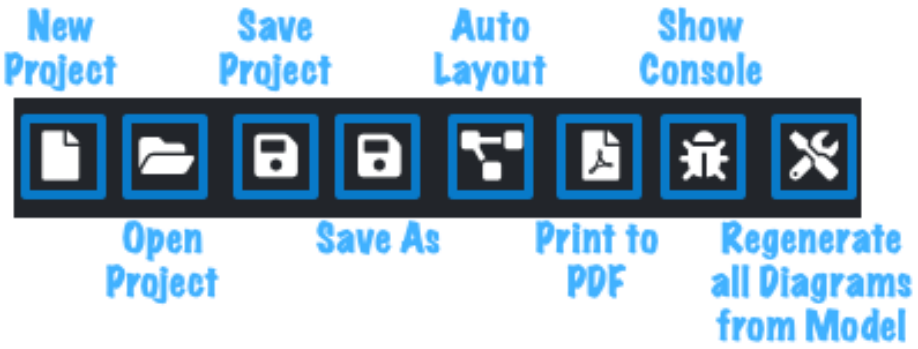

Fig. 3 Toolbar buttons

7 The Project Overview

The Project Overview region of the TDF window contains a list of diagram types grouped into the three TDF stages, namely, (i) Requirements, (ii) System Structure and (iii) Behaviour. Use this list to navigate your project, create new diagrams and select which diagram you would like to view or edit.

You can show or hide the diagram list for each TDF stage by clicking on its associated disclosure triangle. Figure 4 shows the Project Overview with all three TDF stages open; the Case Studies list is closed (no text entry box showing); the Agents list is open with a text entry box and listing the two agents therein. The I/O Diagram is currently being displayed in the Canvas, as indicated by its title bar.

A new diagram of a given type can be added to a list by typing in the name of the new entity, for example, in the Agents list in Figure 4. In this case, a new Agent artefact would be created in addition to the new diagram. Thus, there is a Marine diagram (in Project Overview) and a Marine agent artefact in the Artefact List. This Marine agent artefact can then be dragged into other diagrams (e.g the Architecture Overview).

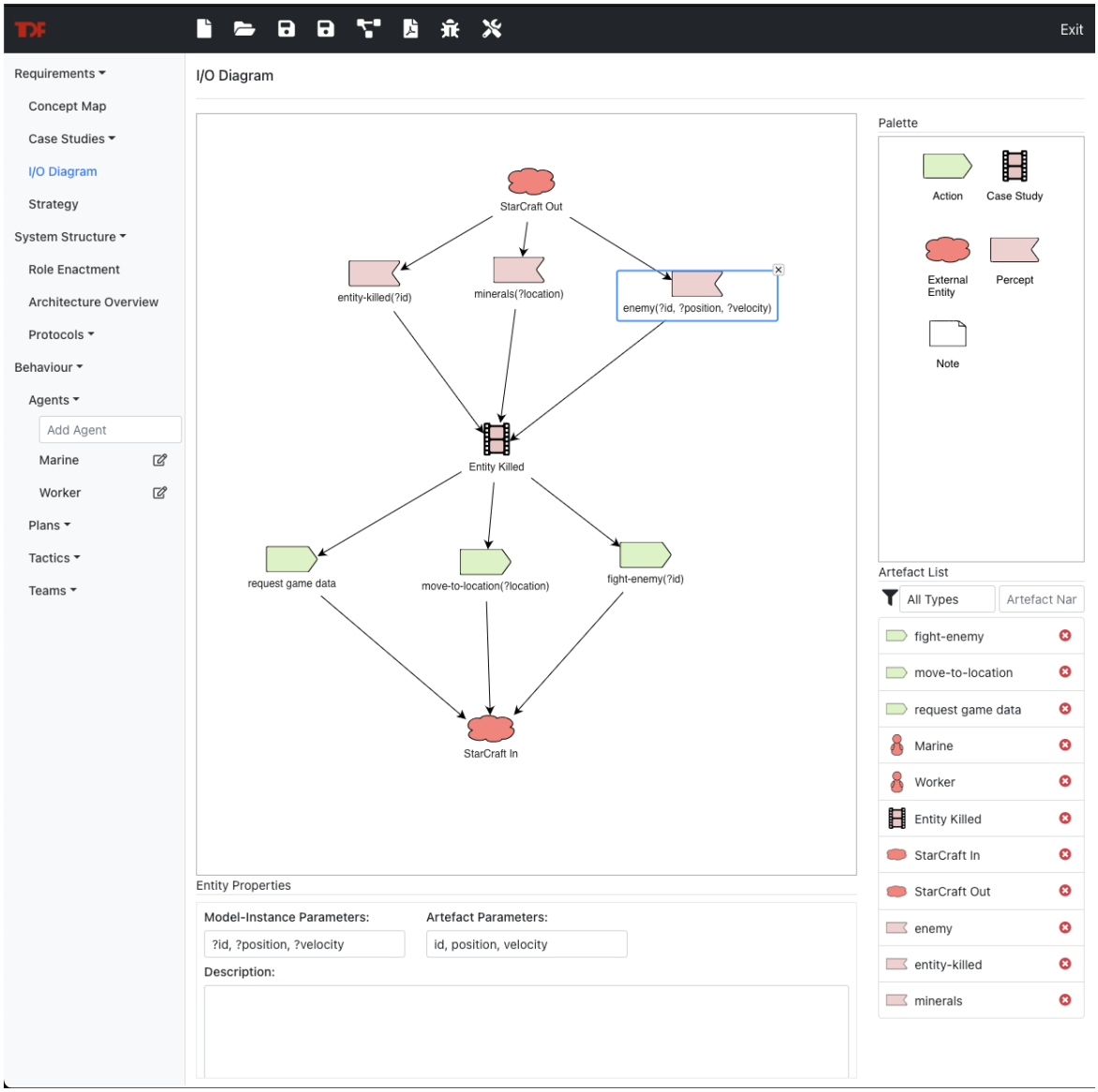

Fig. 4 Project overview

7.1 Parameters of Artefacts and Model-Instances

TDF allows you to define artefacts of various types, for example, Agents, Goals and Percepts. So, you can define a Percept called `enemy`; you can think of this a *class* of Percept. You can then add model instances of the `enemy` Percept artefact to relevant diagrams. In Figure 4, there is an `enemy` Model Instance in the I/O Diagram. The `enemy` artefact is shown in the Artefact List; you can drag it onto the Canvas to create a model-instance of the artefact. Depending on the diagram, you can have multiple instances of an artefact in one diagram.

As shown in the diagram, the `enemy` artefact has three parameters: `id, position and velocity`. There is one instance of the artefact in the diagram, and it has the model-instance parameters: `?id, ?position, ?velocity` (these are all variables, as denoted by the question mark prefix).

The following artefact types can be parameterised: percept, beliefset, goal, messagegoal, action and message. When adding an entity type for the first time, you can add parameters in the entity label. Thereafter, you can only change the parameters by editing them in the Entity Properties pane. You can edit the entity and model-instance parameters in the Entity Properties pane (they can't be changed on the Canvas). There must be the same number of model-instance parameters as entity parameters; thus, you cannot set the enemy's model-instance parameters to `?id, ?position, ?velocity, ?size`. You must first update the Entity Parameters. TDF will then add the extra model-instance parameter for you; you can then edit the parameter name that TDF automatically generated.

You can change the artefact name by editing the label of the model-instance on the Canvas. To do so, click on the label and replace the name (you cannot edit the model-instance parameters here; they can only be chaned in the Model-Instance Parameters field of the Entity Properties pane).

Note that when you edit the name of a model-instance you are actually changing the *artefact* name, thus the names of all model-instances will be updated to match the artefact name.

Model-Instance parameters can be variables (i.e. prefixed with a question mark), strings or numbers. Entity parameters must be a sequence of alphanumeric characters (can include hyphens and underscores). The list of parameters must be comma separated.

8 The Entity Palette

The righthand sidebar contains the **Entity Palette** (for example, Figure 5). As you click on different diagram types in the **Project Overview** you will notice the list of available icons in the **Entity Palette** changes accordingly; each diagram has a specific set icons.

To add a new entity to the canvas simply drag it to the **Canvas** and type in the artefact name.

If artefacts of the current type already exist in the model then the dropdown box will automatically be populated with them, and so you can select the artefact you want to reuse.

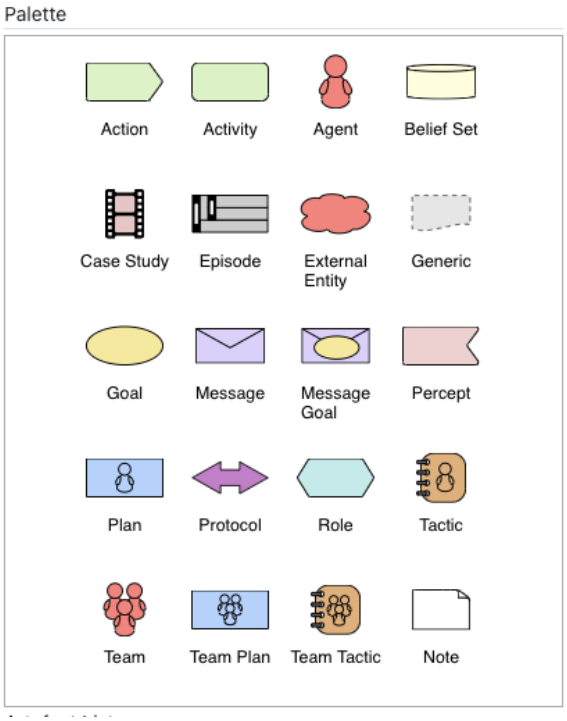

Fig. 5 Entity palette for Concept Map

If you create a new case study, agent, plan tactic, team, teamplan, teamtactic or protocol artefact via the **Entity Palette**, it will also get added to the corresponding section of the **Project Overview**.

In addition to artefacts in the **Entity Palette**, there can be **nodes**. In contrast to artefacts, nodes are *unnamed*. Nodes are used to express logical relationships within a given diagram. For example, a plan will have a **start** node and could have a **decision** and a **merge** node; these nodes act as logical transitions between the artefacts in a plan. Please refer to [1] to learn about their use.

9 The Canvas

We have already described the adding of entities to the Canvas via the **Entity Palette**. The Canvas itself has a number of other features, which we will examine in this section.

9.1 Connecting Canvas Entities

If you hover your mouse pointer over an entity in the Canvas, a pointed-finger icon will appear, as shown in Figure 6.

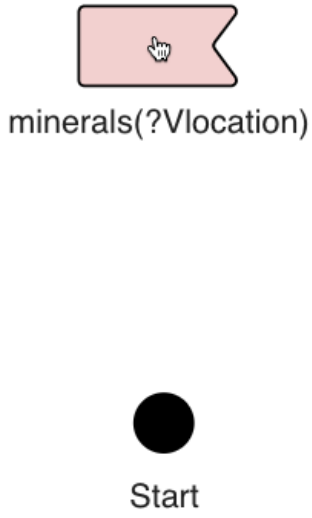

Fig. 6 Getting ready to connect two entities

To connect the source icon to the destination one, click-and-drag over the source icon, move over the destination icon and let go when it becomes highlighted under your mouse pointer (see Figure 7). If you release the mouse before the destination icon is highlighted, the action will be cancelled. If TDF does not allow the two icons to be connected, the destination icon will not be highlighted.

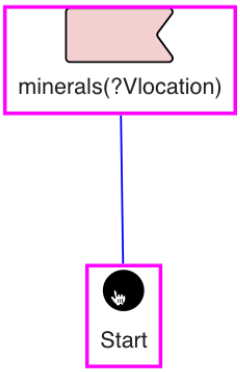

Fig. 7 Connecting two entities

Once released, an arc is created between the two entities (see Figure 8).

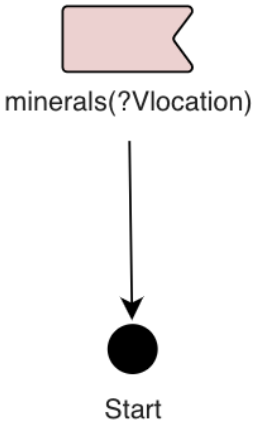

Fig. 8 Entities now connected

In some contexts, you can change the arc type by right-clicking on it and then choosing from the list of available arc types in the context menu (see Figure 9).

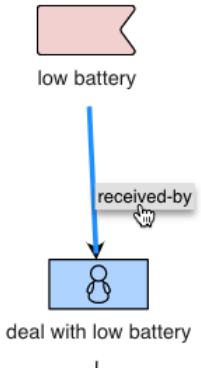

Fig. 9 Changing arc type

9.2 Moving Canvas Entities

To move an entity, hover over it and position your mouse pointer over the entity label; it changes to the arrowed-plus-shaped cursor shown in Figure 10. Click and drag the entity to the desired position.

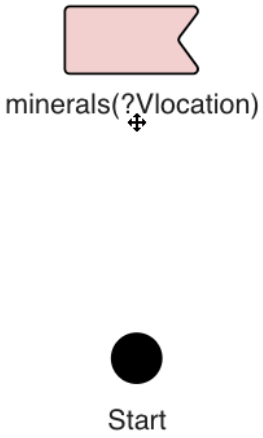

Fig. 10 Hovering over an entity move region

9.3 Selecting Canvas Entities

You can select a canvas entity by clicking on it. When selected, its properties are shown in the **Entity Properties** panel (see Figure 11). You can then edit the properties.

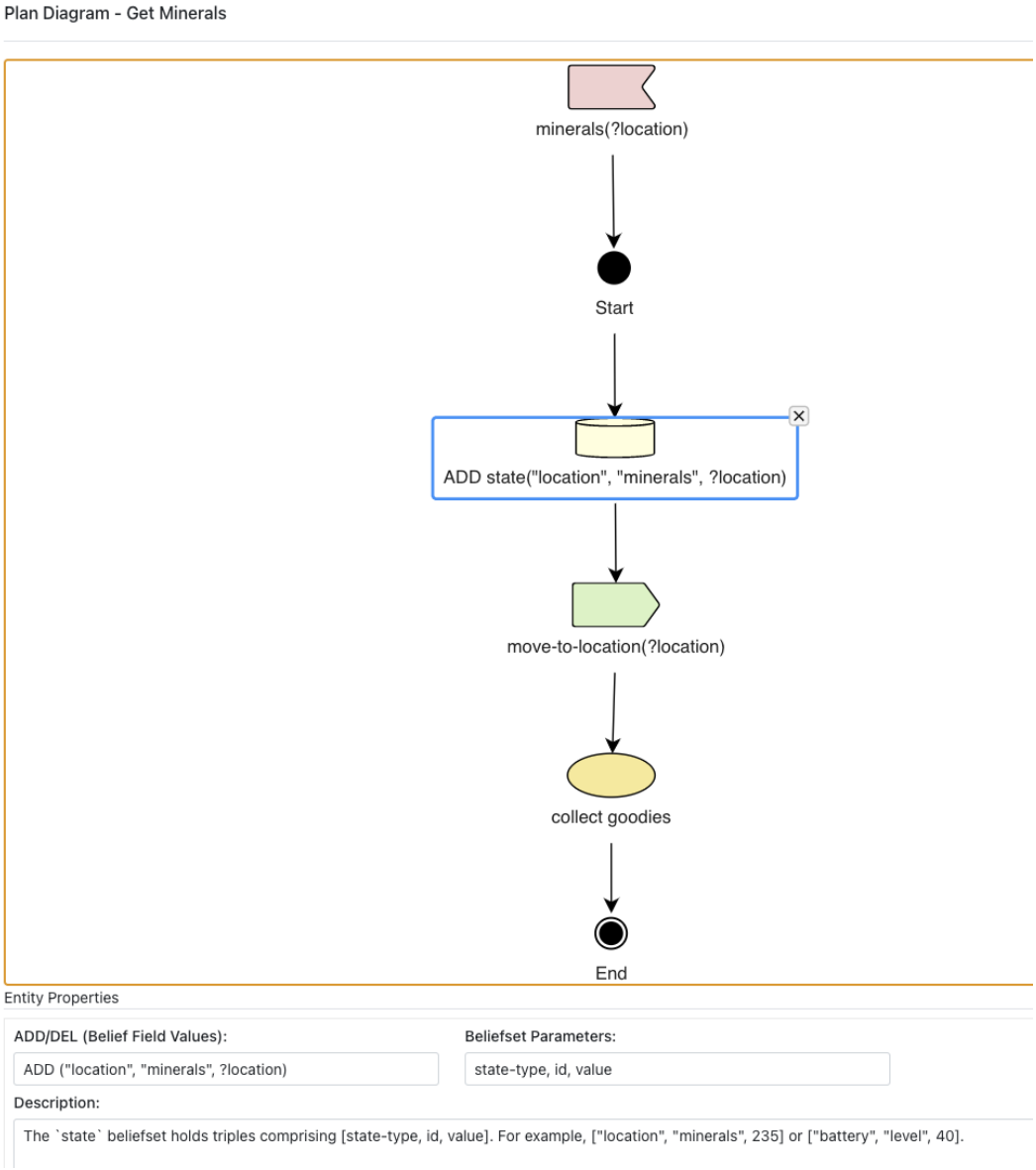

Fig. 11 Selecting an entity

You can also select multiple entities on the Canvas by dragging a selection rectangle around them, as shown in Figure 12. To select non-contiguous regions, shift-click on the relevant items.

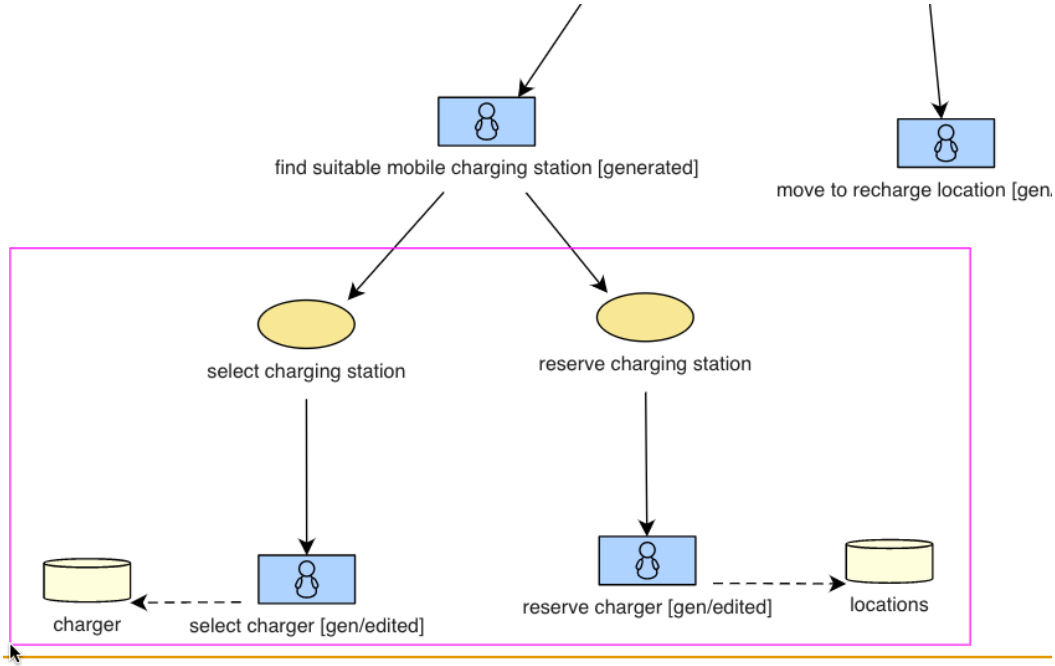

Fig. 12 Selecting multiple entities

9.4 Deleting Entities

If you want to delete one or more entities and/or arcs from a diagram, first select them and then press the **DELETE** key. Alternatively, select an entity and click on the [x] symbol that is on the top-right of the entity (see Figure 13).

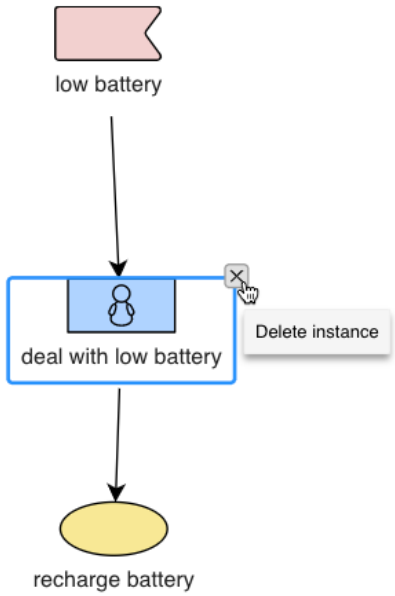

Fig. 13 Deleting an entity

10 Artefacts List Panel

The **Artefacts List** panel at the bottom of the righthand sidebar allows you to drag artefacts to the Canvas. You can also delete the artefact from /emph all diagrams by clicking on the [x] next to the artefact name (Figure 14)). Because this impacts multiple diagrams, you need to confirm that you want all instances of the artefact to be deleted (Figure 15).

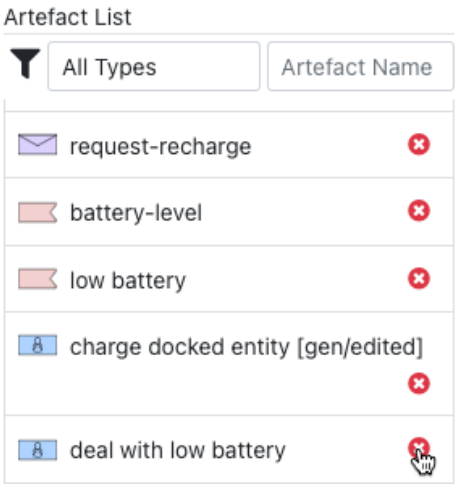

Fig. 14 Delete artefact

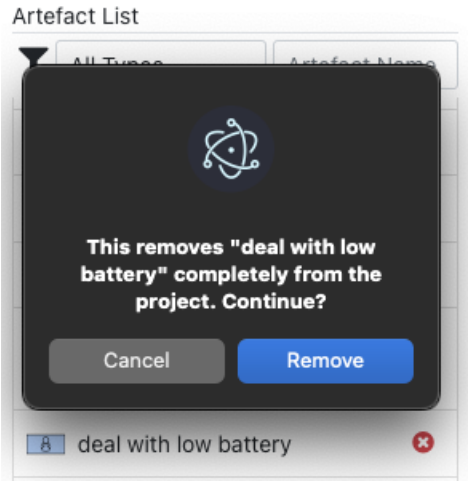

Fig. 15 Confirm artefact deletion

When you select an artefact in the list, it will be highlighted on the Canvas if it is in the current diagram. You can filter the list using the dropdown menu or by typing a search string into the adjacent search box (see Figures 16 and 17).

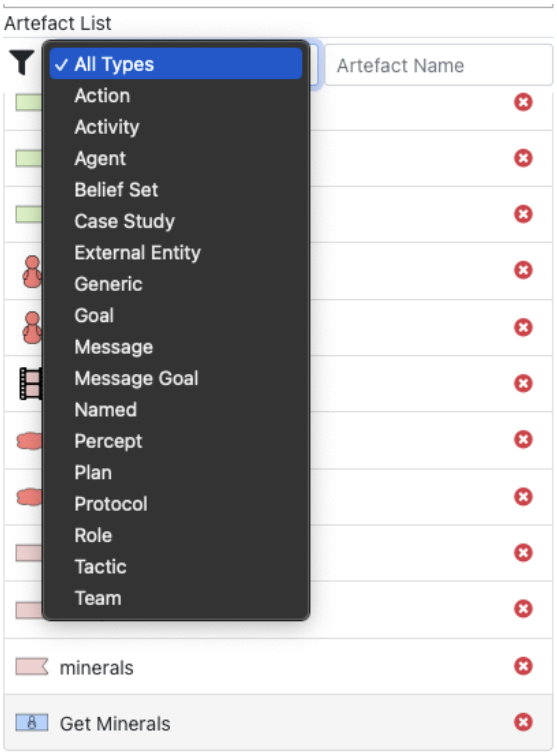

Fig. 16 Filter artefacts list by type

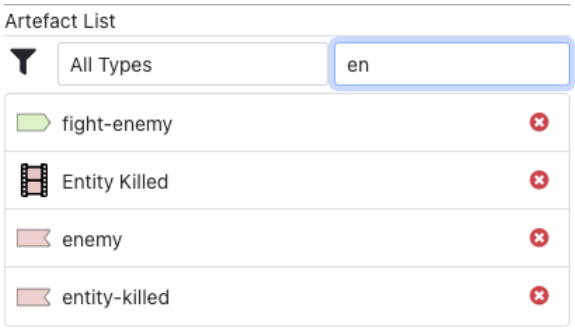

Fig. 17 Filter artefacts list by name

11 Other Functionality

• **Change Artefact Name** Double-click on the icon's label to change the name of the artefact. This name change is inherited by all instances of the artefact in the project. You can also change artefact name by clicking on the Pencil icon in the Project Overview.

• **Propagation.** Propagation of relationships from one diagram to another has been implemented, but propagation as a result of relationship deletion is not supported.

• **Beliefset Updates.** Beliefset updates are an experimental feature in TDF. You can specify that a belief is added or deleted in a plan diagram by using ADD or DEL respectively (see Figure 11 for an example of adding a belief to a beliefset.)

• **Beliefset Access.** Beliefset access is denoted via arc labels (either in a Strategy diagram or a Plan diagram). Figure 18 shows a plan with various types of arc label. The label between the invocation condition (the percept, `navigation-parameters`) and the Start node is the plan **context condition**; it is true if the beliefset `waypoint` contains an entry matching `?id, ?x, y`. The guard on the arc from the Start node to the goal `g1` is passed if the beliefset `state` does not contain an entry matching `"location", "self", ?x, ?y`. The arc from `g1` to `g2` passes if there is no `"location", "self", ?x, ?y` entry in the `state` beliefset, and the goal `g2` will be pursued as long as a matching belief does not exist (this is a **preserve condition** on the goal). The arc between the goal `g2` and the action `move-to` is annotated with a **functional predicate** (essentially, a boolean function), as denoted by the `'IS:'` prefix. The final arc is a natural language entry; if you assign a label that is *not* a syntactically correct `IF` or `IF/WHILE`, TDF will prefix it with two hyphens to denote that it is not a formal annotation.

• **Teams.** Currently, the teams functionality is experimental.

• **JSON Output.** When you save a TDF file, a JSON version is also written. The JSON file contains all of the entities and their properties, including a textual representation of graphical plan diagrams. The JSON file can be used as input for code generation for the agent platform of your choice. The JSON file is unformatted; format it in your IDE if you want to read the contents.

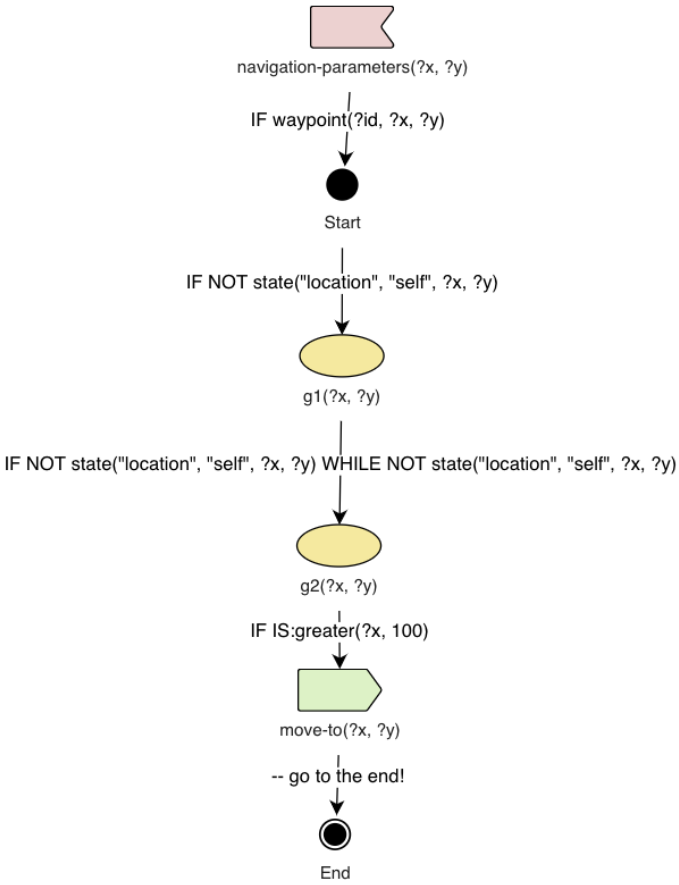

Fig. 18 Plan arc labels

References

1. Evertsz, R., Thangarajah, J., Ly, T.: Practical Modelling of Dynamic Decision Making. Springer International Publishing (in press)
2. Padgham, L., Winikoff, M.: Developing intelligent agent systems: a practical guide. Wiley (2004)
3. Rao, A.S., Georgeff, M.P.: BDI agents: From theory to practice. In: Proceedings of the first ICMAS (95), pp. 312–319. San Francisco (1995)