Assignment 2
Building modelling in AToMPM

Cláudio Gomes
claudio.gomes@uantwerp.be

July 29, 2017

1 Practical Information

The goal of this assignment is to repeat the modelling task of the previous assignment, in the AToMPM visual metamodelling environment.

The assignment structure is the same as the previous assignment, and the advice with respect to tooling applies to this assignment as well.

1.1 Task Overview

Task 1 *Implement the abstract syntax of Bmod in AToMPM using the* `/Formalisms/__LanguageSyntax__/SimpleClassDiagram` *formalism.*

Task 2 *Describe the concrete visual syntax of Bmod.*

Task 3 *Enhance the usability of Bmod by adding actions to automatically move/snap/resize model elements.*

Task 4 *Use Bmod to model a building floor.*

Task 5 *Write a report.*

1.2 Deadline and Logistics

Complete this assignment in **groups of 2**.

One, and only one, person in the group must submit the solution on Blackboard before TBA.

Contact Cláudio Gomes (`claudio.gomes@uantwerp.be`) if you have questions.

2 Requirements

2.1 Task 1

The requirements for this task are the same as in assignment 1, except the tool used is AToMPM.

The formalism should be called Bmod. The abstract syntax model file should be called BmodMM.model. The metamodel (a.k.a. compiled abstract syntax) should be called Bmod.metamodel.

At the end of this task, the following folder should exist in atompm installation directory:
`atompm/users/yourName/Formalisms/Bmod/`
And it should include every file that you used to solve this assignment.

2.2 Task 2

A suitable concrete syntax should clearly show the relevant information to anyone reading a Bmod plan model. This includes at least:

1. The name of each room;
2. Each person's name, and in which room the person is in;
3. Each person's schedule.

The concrete syntax file should be called
`Bmod.defaultIcons.model`
And the compiled concrete syntax should be called
`Bmod.defaultIcons.metamodel`

2.3 Task 3

Enhance the usability of Bmod by adding an action that automatically positions a person roughly in the middle of the room, when it is placed in the room.

Expert. Make the necessary changes to Bmod, such that the number of people in a room is displayed visually. This number should be automatically updated whenever people get into or leave the room. Additionally, add actions that snap doors to the edges of rooms.

2.4 Task 4

It should be possible to model a building plan similar to the one in Figure 1. Create routines for the relevant people.

Expert. Create your own floor plan. But beware that there must be at least two ways of getting from a room to some other room (in other words, some circular structure).

3 Tools and Documentation

- AToMPM git repository (with installation instructions):
  `https://msdl.uantwerpen.be/git/simon/AToMPM`
- AToMPM documentation, accessible from the UI, is in:
  `https://msdl.uantwerpen.be/documentation/AToMPM/index.html`

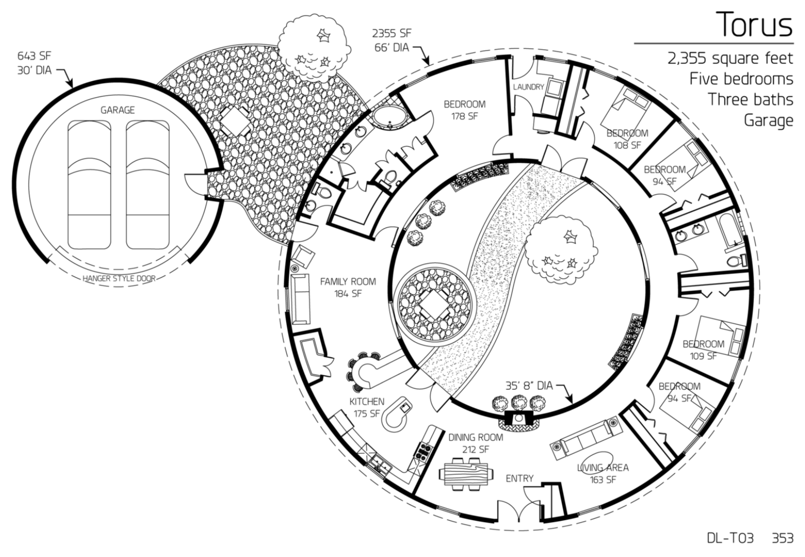

Figure 1: Sample floor plan.

4 Tips, Tricks, Pitfalls, and Issues

Below is a list of known obstacles, and advice, while working with AToMPM:

Advice. The installation of AToMPM in windows requires specific versions of python, igraph, and visual studio. If you do not want to "pollute" your machine with old versions of these dependencies, install AToMPM in a Linux virtual machine (the installer works almost flawlessly). With a linux virtual machine, you use your browser if you set up a "host-only" network adapter.

Advice. There are plenty of examples shipped with AToMPM. Of particular relevance is the PacMan language, which includes action code for snapping elements together, and other features that you may need.

Advice. While working in AToMPM, keep an eye in the browser console, as warnings are shown there. For example, when an element is dropped inside some other element, but no containment relationship was created, a warning is issued. This is important since sometimes the containment relationships fail to be created.

Advice. For containment relationships, it is advised that these are not invisible links, but instead barely visible ones. Containment relationships between an element A and an element B are only created if the element A is moved into element B. If element A is pasted or created directly on top of element B, the relationship is not created, resulting in hard to debug problems. The containment links are a visual cue that the containment relationship was created.

Bug. After performing a window switch with ALT+TAB, if a right arrow key is pressed while editing a text area, a tab is inserted, instead of just moving the arrow to the right. This is indented to facilitate indentation of code.

Advice. Backup the formalisms folder, or keep it under source control. This is because AToMPM can corrupt the model files if it crashes.

Advice. Keep in mind that any change to the abstract or concrete syntax automatically invalidates any Bmod model that you have made. Therefore, make sure you made very small models to test the abstract and concrete syntax, before solving Task 4.

Advice. A possible workflow while creating a language in AToMPM is as follows:

1. Create abstract syntax element and its attributes.
2. Save and compile the abstract syntax.
3. Add the icon or link description of the newly created element to the concrete syntax.
4. Save and compile the concrete syntax.
5. Test the concrete syntax by creating a simple model with the newly created element.
6. Backup, and go to step 1.