Assignment 1
Building modelling in metaDepth

Cláudio Gomes
claudio.gomes@uantwerp.be

July 29, 2017

1 Practical Information

The goal of this assignment is to design a domain-specific modelling language (formalism) for representing the movement of people in a building floor.

The language is called *Bmod*.

This section describes the tasks that you need to complete, and the next gives the requirements about each individual task.

The solution to each tasks depends on the way you solved the other tasks. For example, Task 4 (below) is strongly related to Task 1. So *read carefully the whole document* and make a solution sketch of each task, before implementing the solution.

The tasks are purposefully ambiguous. Use common sense or ask Cláudio if you are unsure about what a task means.

Each task may have a simple version, and an "expert" version. *To get a full mark*, the expert version must be solved correctly.

Tooling. The tools used in this assignment are prototypes still under active development. Therefore, except bugs and usability issues. Feel free to report them to Cláudio, or to the tool's developers. Make sure to solve each task incrementally, and either make regular backups of the input files, or use version control software. These tools can crash and corrupt the input files.

Section 4 describes known bugs, and advice regarding the tools.

1.1 Task Overview

Task 1 *Describe the abstract syntax of the Bmod language in the metaDepth tool.*

Task 2 *Describe validity constraints that any model must obey in order to conform to Bmod.*

Task 3 *Describe the operational semantics of Bmod.*

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 Bmod language is to be used by anyone (that is, non computer experts) that wants: a) to design a new floor plan, or b) to study an existing floor plan. To accomplish this, it should be possible to use Bmod to model rooms, doors, and how those rooms ought to be used during a normal day.

Room usage information is important because a good floor plan should avoid crowding, underused rooms, etc...

To model the room usage, it must be possible to represent different people and their movements along the different rooms during the day. For example, John usually works in the office, but occasionally (e.g., every 50 minutes) goes to the restroom, while Gabriel usually goes around the floor (e.g., first to the corridor, then to the office, then to the corridor again, then to the secretary, etc...) for most of the day.

The routine of each person should be as concise as possible. For example, it should not contain the detailed path followed from one room to the adjacent one. Instead, it should contain high level descriptions of where (and when) the person needs to be.

For this assignment, you can assume that each person has a different routine. Therefore, to model a floor which is used by 50 people, you need to describe 50 schedules.

Expert. Instead of assuming that each person has a different routine, introduce the means to model different kinds of people, such that any Bmod user can represent large numbers of people easily. Note however that each individual person must still be able to have its own custom routine.

2.2 Task 2

Since Bmod is made to model a single building floor, you must make sure that any valid Bmod model obeys the following constraints:

1. It should be possible for a person to move from any room to any other room.

Expert. Create two more validity constraints that makes life easier for Bmod users.

Tooling. To implement this task, use the validation functionalities provided by metaDepth. Since the tool is a prototype, there can be bugs in the validation. If you suspect that a bug is causing your validation code to fail (or crash), take note of the validation code, remove it, and implement the validation in a `checkValidity.eol` script, that throws an exception if the model is not valid. Finally, mention this incident in the report, and paste the validation code that did not work.

2.3 Task 3

Bmod users should be able to get a sense of how the rooms are going to be used throughout the day, according to the people they modelled.

In this task, you must implement the operational semantics of a given Bmod model, in a `operationalSemantics.eol` script, which essentially simulates the people moving around the floor.

The simulator keeps track of simulated time, moves people around the floor according to their schedule, and gives sensible output about people's movements and room occupation.

People cannot teleport (move in zero time) from a room to a non-adjacent room. They must follow a path along adjacent rooms. It is permissible that a person arrives late at a room.

Expert. People cannot be late, and they always follow the shortest path to the destination.

2.4 Task 4

It should be possible to model, and simulate, a building plan similar to the one in Figure 1. Create routines for the relevant people, and analyze room occupation.

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).

2.5 Task 5

The grading process depends on a well written, and concise, report.

A good report should:

1. Explain the general approach to each task, and in case of ambiguities, describe your decisions and rationale.
2. Explain the structure of your submission, and the purpose of each individual file.
3. Describe any issues encountered with the tools, and how they were overcome.

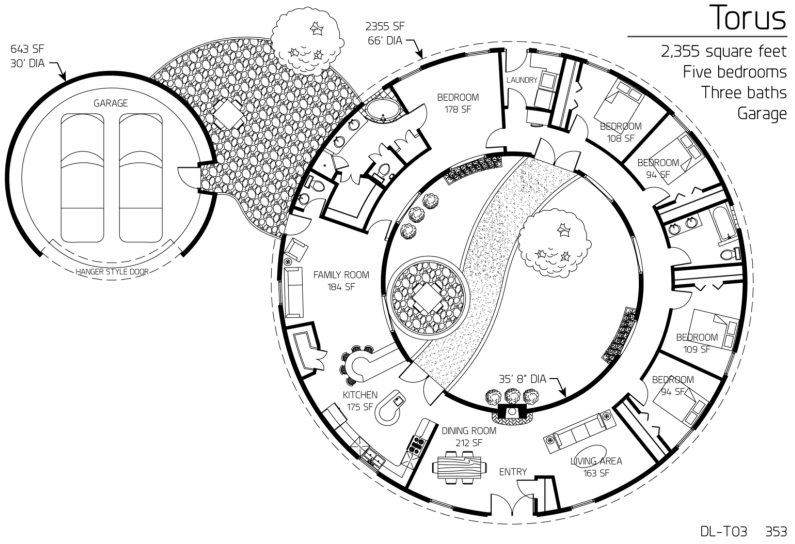

Figure 1: Sample floor plan.

4. Give an estimate of how many hours in total the group spent to solve the assignment. This measure is not to evaluate the efficiency of the group, but rather to have an estimate of the relative difficulty of the assignment, and provide a fairer grade.

3 Tools and Documentation

- MetaDepth main page, documentation, examples, and download: `http://metadepth.org/`

4 Tips, Tricks, Pitfalls, and Issues

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

Advice. Build your metamodel (and models) incrementally, checking whether they work at each point, and make backups.

Bug. Avoid having nested models. They are not parsed correctly by metaDepth.

Bug. Avoid using derived attributes. If you need derived attributes, these can be computed in the .eol script, using assignments of the form

```
element.~derivedAttr = expression;
```

Bug. Any instruction after the first operation declaration in EOL is ignored. So, code instructions first, and then operation declarations.

Bug. If a model element has attributes that do not correspond to the corresponding metamodel element, metaDepth will not throw an error mesage, and will most likely not parse the other attributes of the element correctly. For example, if the metamodel declares a person to have a name and an age attribute, and the model instantiates a person with `pname = John` and `age = 20`, it is likely that John's attributes will not be parsed.