

Domain-specific modeling of Cooking Recipes - Project Report

Xiaoxi Dong

McGill University, Montreal Qc Hxx xxx, Canada

Abstract. Cooking recipe is an important source for people to learn cooking. As always happening in cooking, looking for an accessible and handy recipes troubles cooking enthusiasts. Answering to this problem, the modern recipes are getting more detailed and personalized. Usually a modern recipe gives ingredients with the amount and preparation method and step-by-step instructions of cooking. However, the cooking process is very flexible and personalized. The differences exists in the physical condition of the kitchen, the condition of the ingredients, the ability of the cook and so on. This situation makes it is impossible either for cookbook writers to cover all the situations or for reader to follow the recipes. As a result, we were thinking of a cookbook that is able to meet the requirements of both sides. In this project, we implement a modeling environment for the cooking recipes. We design a meta-model, which is equivalent to a language, of the cooking recipe. It provides a platform for writers to write detailed cooking instructions. We also implement the automatic transformation from this recipe model to an interactive cooking instructor application. This application is able to modify the recipe according to the user's condition and gives timely reminder of each steps in the cooking process.

1 Introduction

Cooking to an ancient recipe was a privilege for the professionals. It was purely narrative and required great experience to understand the key words. To make recipe easier for the ordinary people, the modern recipe began to adopt detailed instructions. A modern recipe usually has most of the following parts, which are the list of ingredients with quantities and preparation instructions, the instructions on cooking process and timing, the number of people that the complete recipe will serve and sometimes the images related to the recipe. Although the modern recipe writing do make life easier, the desire of looking for a more personalized way of 'recipin' is interesting.

The cooking process is flexible when we consider the person who cooks. People developed different level of cooking skill, different speed and different cooking habit, which makes it difficult for a recipe to satisfy different type of users. The situation is especially difficult for those beginner cooks when they need to work out their tasks from others' descriptions. The cooking process also has many constraints, such as the condition of cookers, the space of the kitchen, the

number of cooks and the amount of ingredients. The recipe has to be modified so that it conforms to the physical conditions. These characteristics makes it difficult for either the cook book writers or the cookbook readers to make a recipe satisfy different conditions. Thus, in our project, we are developing a tool that creates the recipes uniformly but is able to modify the recipe according to the requirements from the users.

The tool which is called the domain specific modeling for cooking recipe, is an modeling environment for writers to graphically create models of recipes. The recipe models is then modified during model transformation process accordingly. The changes can be observed in the ingredient quantities in the textual recipe. It can represent the recipe model in multiple ways including textual and live instructions. Images can be used for ingredients and instructions by specifying a link to that file as well. From the recipe model, it can create the textual shopping list and the textual cooking instructions. It can transform the model into a process-node model to analyze the timing. Then from the process-node model, it generates code in programming language to implement the real-time, web-based animation of tasks to perform. Therefore, the application will notify the cook what to do at the same time of cooking.

To implement this domain specific modeling environment, we developed two meta-models which are Recipe model and processNode model respectively. Both of the models are built in class diagram. The former one is the language of creating recipe and the latter one is the model used for scheduling and code generation. The model and the transformations are all build in Atom3[[2]], a general purpose visual meta-modeling and model transformation environment. And the outputs of the model use .html.

The rest of this report is organized as below. Section 2 introduce the design of the recipe meta-model. Section 3 shows the design of the processNode model, which is the model for scheduling and simulation, and the transformation rule from the Recipe model to the processNode model. The last part of Section 3 explains the rule of scheduling and merging for processNode model. Section 4 discusses conclusion and future work.

2 Meta-Model for Cooking Recipe

2.1 Class Diagram Design

We use the class diagram as meta meta-model to design the 'recipeing' language. The recipe meta-model has three classes, representing the ingredient(material), the cooker and the process respectively. The process class represent the cooking agent which takes some ingredients and cookers(tools) as input and produces new material(ingredient).

As can be seen in the Figures.1 and Figure.2), the ingredients are input of a process. The ingredient class have attributes such as name, amount, unit, preparation instructions and link to image. A cooker class represents the tools required by the recipe such as bowl, pan, knife, oven and alike. It is also the input

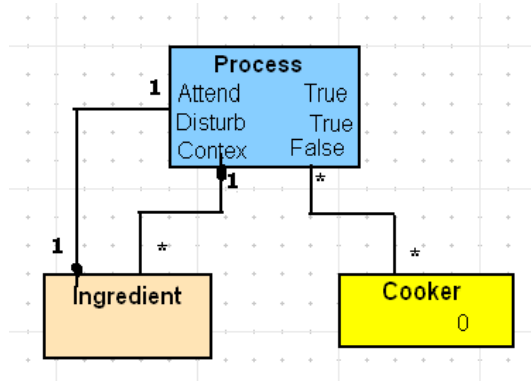


Fig. 1. Class Relations

of a process. Cooker class has attributes of cooker name, the image link and a brief description. Sometimes the cooker can influence the timing and scheduling when the tools are limited and that is why a number attribute is defined. Usually one process can have many inputs but each time there is only one material produced by one process. That is because when the material is processed, it changes to another cooked material. In the other side, one material is produced by one and only one process. We can use many cookers in one process, but no cookers are to be created in a process.

The third class in the recipe meta-model is the Process class. This class defines the actions the cook is supposed to do at each stage. It has a description of the action, the amount of time each process takes and the link to multimedia illustrations. The process class also has an attributes to specify Does this process need cook's attend? which indicates whether we can insert other process between this process or not. If a process does not need attendance, we can let it stay alone until the end of the process while we can prepare other materials. The process class also has a tag for Can this process be disturbed in the middle?. Another attribute is to indicate the relation between current process and the previous one, that is Should I start this process immediately after the previous is done?. This attribute will finally helped in scheduling process to identify the critical links.(see Figure 2)

2.2 Textual Representation Functionalities

When creating graphical environment for the meta-model, along with the three classes I also add buttons to generate scalable ingredient shopping list and textual description of the recipe instructions. The button for the shopping list calls the function at the event of 'click button'. In this function, the system searches through the graph nodes in the model and identifies those nodes has type 'Ingredient'. After having the list of ingredients, a Dialog is used to obtain number of serving and saving directory from users. Then a html code generator is called to

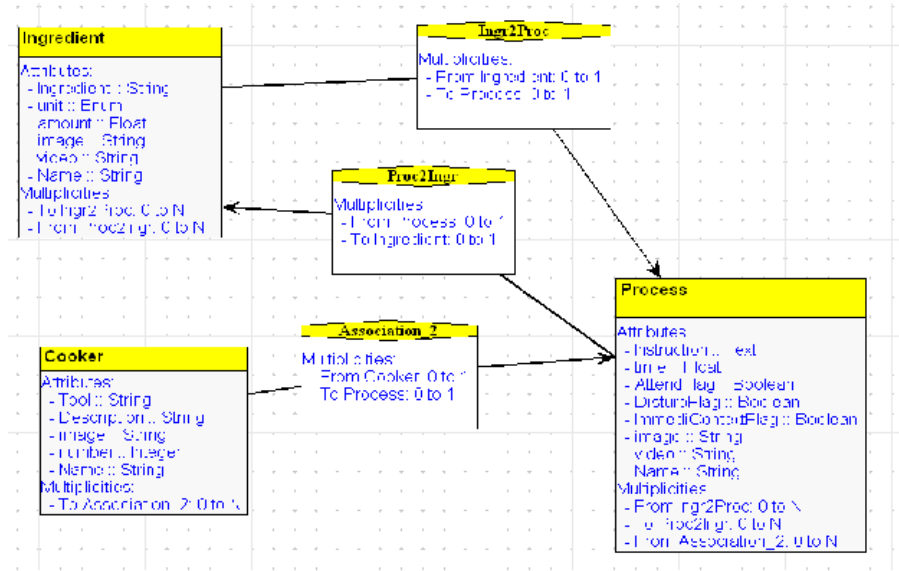


Fig. 2. class diagram of the recipe meta-model

generate file `shopping_list.html`. The instruction button one also adopts function calls and instructions are generated as `.html` file just like that of `shopping_list`. (see Figure 3 and Figure 4)

3 ProcessNode Meta-Model

To generate the easy-to-follow recipe, we need to transform the recipe model into a model which can represent the sequence of actions and the timing of actions. To achieve this goal, I designed another meta-model called ProcessNode meta-model. To execute a cooking instructor, the reader will invoke the transformation from recipe model to ProcessNode model and generate code from the processNode model. The application will use this code to produce the real-time display of the recipe.

3.1 ProcessNode Meta-model of scheduling in Class Diagram

This meta-model is also created using the class diagram. It comprises of class of process node and the association between processes. The process class has textual attributes of input items, output items and description of the current process. It also have time attributes indicating the duration of process, the earliest time a process can start. The association between process classes has one attributes 'Duration' that is the time difference between two processes. If this attribute is larger than 0, the latter process need to execute exactly that time after the former; if this attribute is 0, we are going to compare the other attribute

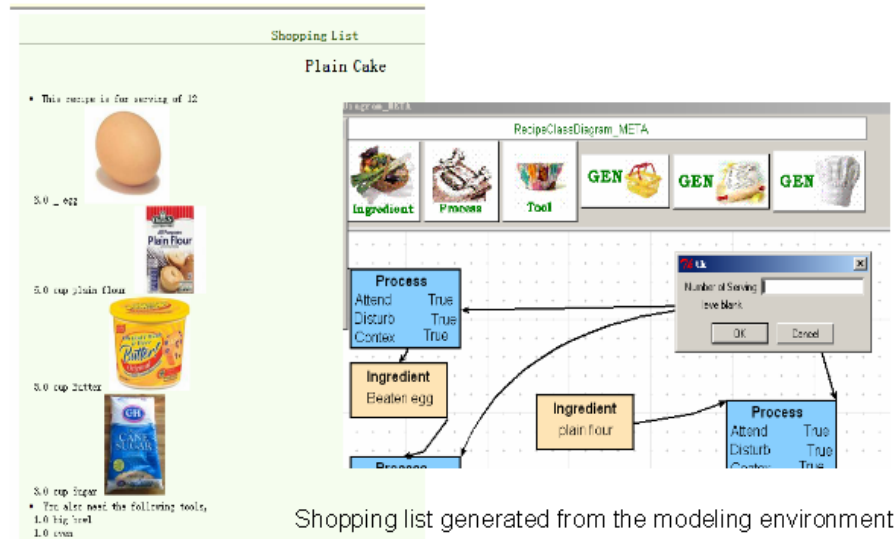


Fig. 3. Function Node 1: Generate Shopping List from Recipe Model



Fig. 4. Function Node 2: Generate Instructions from Recipe Model

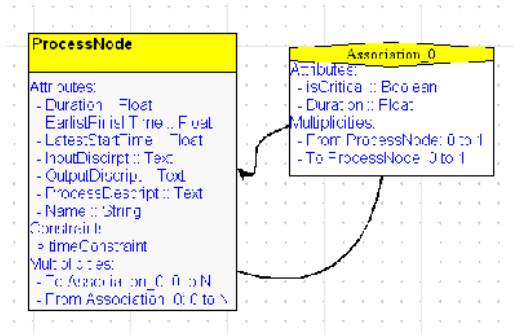


Fig. 5. ProcessNode Meta-Model in Class Diagram

'isCritical'. This attributes indicates the relation of two process. If it is true, the process will executed one after another. Otherwise, we can wait arbitrary time between two processes. The time attribute in this the ProcessNode model is important for timing and scheduling.

3.2 Model Transformation from CookBook Model to ProcessNode Model

This Transformation is the first step in generating program from the Recipe Model. There are three steps of this stage of transformation,

- 1.replace Cookbook model nodes using ProcessNode model elements;
- 2.connect neighbor process nodes;
- 3.clean up the Cookbook model.

In the first step, the application find the process node and generate an ProcessNode node which scans the input and output links of the process node, copies the duration attributes.(see Figure 6) If the process is not critical(do not require attendance), the process is divided into two process nodes with a link with time attribute of process duration. (see Figure 7)

3.3 ProcessNode Model For Timing and Scheduling

This Part is one of the most important part in this project. In this step, the ProcessNode graph is going to be flattened according to the number of cooks. By doing this, the graph becomes threads of actions where the number of threads depends on the number of cooks. Currently, the application only support one thread, that is one cook cooking. Because the cooking process is flexible, the stage involves many uncertainties. As a result, the transformation rules need coding for the constraints. It mainly involve the following actions,

- priority 1.calculating the earliest finish time for each processNode, which serves as the flag of sequence;

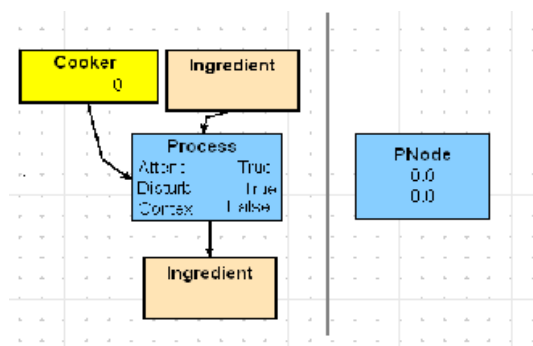


Fig. 6. The Normal Transformation from Recipe Process to Process Node

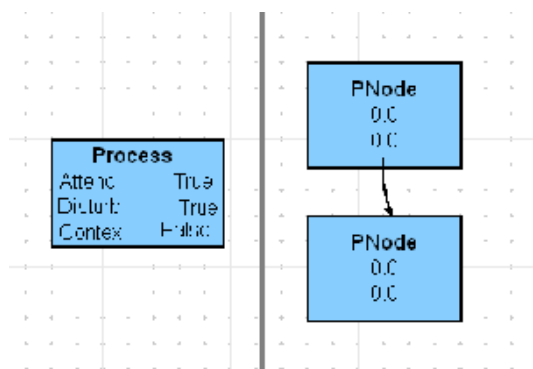


Fig. 7. The 'Non Attend' Transformation from Recipe Process to Process Node

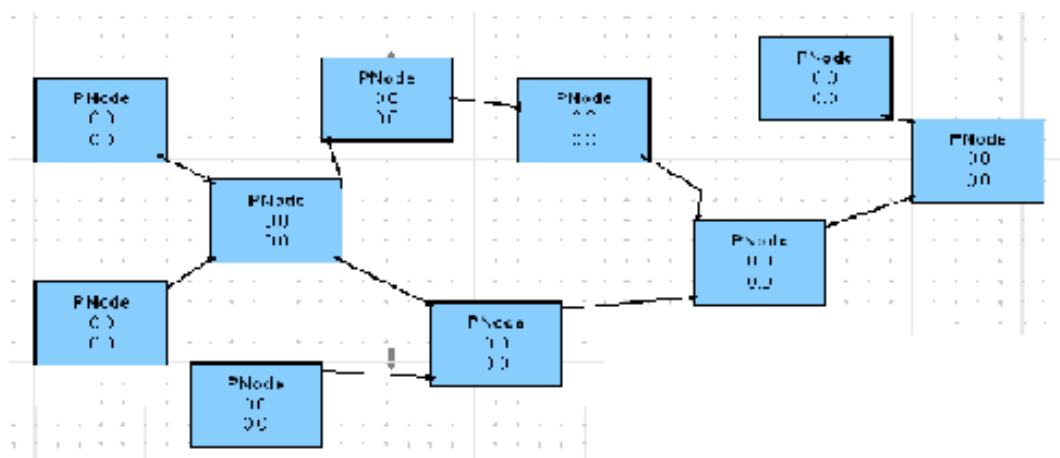


Fig. 8. Graph of ProcessNode

- priority 2.merge the critical link and non-critical links,as described below;
- priority 3.merge the non-critical links, as described below.

There are mainly two kinds of situations considering the way the nodes connect together. One situation is that only one of the two links are critical(which means it should execute immediately after the previous process), and the other is both the links are non-critical which also has two sub-types. If the two nodes links to the one node in the middle, we can move the node with non-critical link backwards to avoid disturb the critical line as shown in Figure ?? . Because the critical line, which is in orange, will not be disturbed regardless of the time or resources. This kind of situation is put in higher priority than the other.

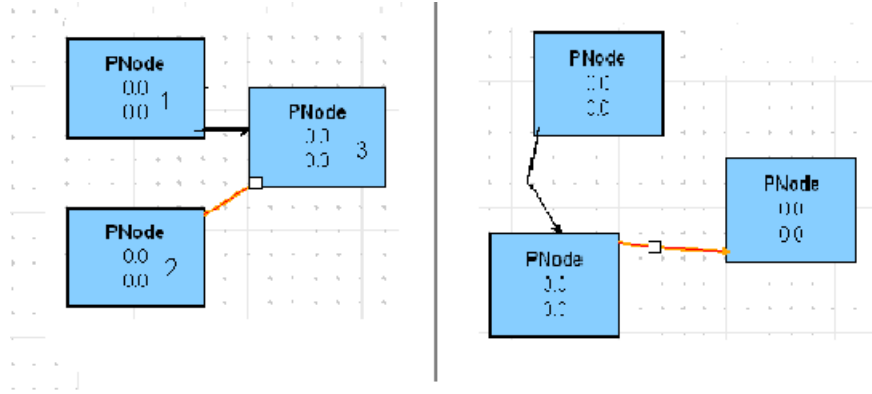


Fig. 9. one critical link : Left - Before — Right - After

The other situation, which has two non-critical link to the middle one. In this situation, there are two kinds of situation. One is the link is larger than 0 which means the link has a fixed time and it is not allowed to add any process longer than this period of time. In this situation, we compare the time of the link and that of the other node, if the link is longer, the corresponding node is inserted as shown in the graph. Otherwise if it is shorter than the period, the rule is not eligible.(see figure 10)

$$new_link_time = old_link_time - process_duration$$

The second situation is that both the links are non-critical and not fixed. When this situation matched, we will pick randomly a link is to be merged.

Every time after there rule executed, the processNode model is likely to violate the rule of calculating earliest finish time. Since the priority of these rule is higher, the time will be updated every time when the structure of graph

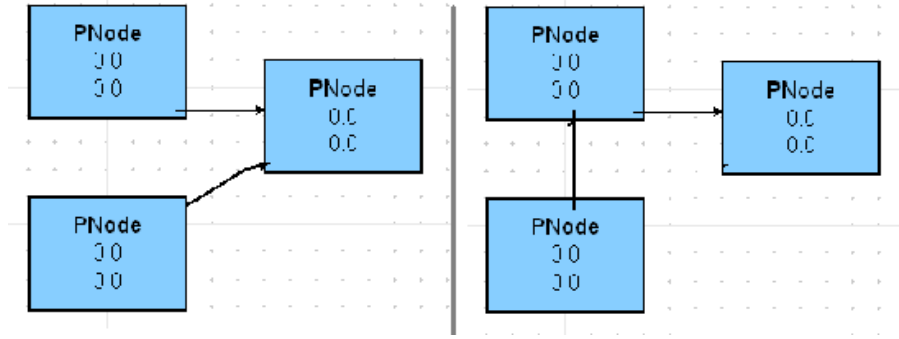


Fig. 10. No Critical Links

changes. This situation makes the scheduling updated and accurate. The similar situation happened to the splits as well.

After this part of transformation ends, we are going to generate code from the thread of processNode. This function is implemented as the finalAction in the transformation in Atom3. In the final action, a function call of instructionScheduler() is executed. It read through the processNode nodes and generate generate .html file for each nodes (i.e. the actions) in timely sequence. To use the instructor, you will need to open the file with sequence number 0 and this file is to be refreshed and navigate the the next file when action duration reaches. That is how animation is implemented.

4 Conclusion and Future Work

Instead of being a privilege of professional programmers to develop interesting applications, it is considered that domain specific modeling provides an accessible way for people in a specific field to produce domain specific applications. The cooking domain modeling environment developed in this project is one of these easy-to-learn and easy-to-produce developing environment for producing interesting applications. This cookbook domain specific modeling tool provides a language for recipe writing and transform the recipe model into a interactive real-time cooking instructor, which provides various forms of recipe. To implement this tool, I build meta-models and design transformation rules. The current model can generate textual representation of shopping list and instructions. It also generate code to refresh a .html file when a process is due. The users can follow the instructions appearing in the web page to cook.

The ingredients are with quantities and preparation instructions. It is reasonable to have images or video to illustrate the ingredients. Some of the ingredients need to be prepared if the cook do not have it handy. The recipe should address this situation by replacing that ingredient with the preparing process in current recipe if the cook asks. We are able to know the optimal number of people cooking for the recipe by counting the number of header process (the processes that

have no input connections and the splits in the graph). If there is not limited in cookers, the optimal number of cooks are able to work together. In the next transformation stage, working load distribution is able to be implemented. The current model have problems in graph with circles as it works moderate to identify the starting node of this branch. The remaining nodes in the branch will follow the same rule until all of them are merged into the thread. Note that the action with a critical link is not separable and the nodes need to be inserted all together. For the splits, the link will move forward until it finds a match.

References

1. Atom3 tutorial. In: <http://moncs.cs.mcgill.ca/people/jlara/AToM3.Programming/index.shtml>
2. The structure of Recipe In: <http://www.hertzmenn.com/articles/2007/recipes/>