Grader model

Course project for Modelling of critical systems

Ólafur Dagur Skúlason

IY11249

10 December 2018

Introduction

In this paper I will detail an implementation of the software for a grading machine, as developed for the meat processing industry. This is a course project for a university course, and as such the emphasis of the project is not on feature completion and perfection, but rather on learning to apply the benefits of using VDM to create a model. The type of machine in question is based on work I have made for a previous employer several years ago, and proprietary elements have kept out of the solution.

Project Description

The grading machine is an industrial machine that sorts and batches pieces of meat prior to packaging or further processing, as such these machines can be found at any stage in the factory process flow and any flaws in their operation can have large financial consequences. The physical machine is conceptually rather simple, a conveyor with pieces on top surrounded on both sides by receivers, called gates, that are capable of routing material off the conveyor belt. In Illustration 1 we can see a simplified view of what a grader consists of. There is the central conveyor with pieces (A,B and C) on the belt. To the side are six(6) locations, each behind a gate. Each location is also equipped with a button (not shown in the image) where an operator can accept a finished batch, opening the location to resume production, for testing purposes these buttons should be pressed immediately after a batch is competed. In the illustration the gate for location 6 is open and piece A is being routed. As the machine is intended for factory production, it must be able to manage the two major roles it shall play in the production process, sorting pieces into different productions, and batching pieces into consumer packaging. To this end the machine shall receive orders from a back-office production terminal (e.g. Scada) and assign the production order to some of the locations on the machine.

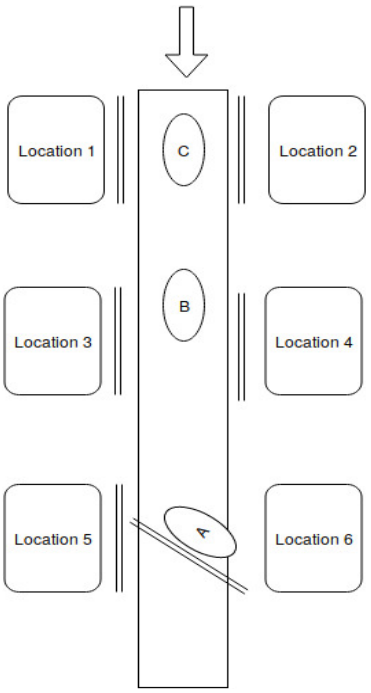

Illustration 1: Conceptual grader

Requirements Specification

The requirements as specified cover the rough structure of what a machine of this sort should do. They are specified in Table 1, note that not all of the requirements have been implemented. The cause for this is both time pressure and only following prior consideration that adding that feature or constraint would not add much value to the learning goals of the project. The original requirements were written from memory based on the device I was a part of designing several years ago.

When I started building the model and adding the various conditions to the system I started to see a need for minor additional features that a system of this sort needs. As such quite a few minor features were added to the system to satisfy these constraints, as an example of such a feature would be pre-emptively closing a gate in case a second piece comes too closely behind it. As such not all the functionality of the system is covered by the requirements specification as written, but are hidden in minor adjustments made in the model.

ID	Name	Description	Implemented
1	configurable number of bins	The machine shall have a configurable number of bins, this is to make the software conform to each machine and not the other way around	Yes
2	Time is a function of conveyor movement	The machine shall treat time as distance moved on the conveyor, this is to allow for changes in belt speed without complex reconfiguration of the software	Yes
3	A bin can contain a limited amount of material	The physical bin is limited in the amount of material contained	No
4	Gates facing each other can not be open at the same time	Since the gates open out to the belt, having gates open into each other will cause damage to the hardware	No
4	The tracker must maintain and track piece events	Piece events are the engine driving the machine’s behavior. These events include grading, opening/closing gates and routing of pieces	Yes
5	Pieces can not go to locations different from their destination	A piece going to the wrong destination can be caused by a failure to close the gate early enough.	Yes
6	The machine must maintain a list of orders	The machine must be able to receive new orders overriding existing production.	Yes
7	A piece can only be graded to an order	The machine must only use open production orders to grade pieces into.	Yes
8	Grading algorithm	An algorithm that provides size grading, no batching	Yes
9	Batching algorithm	An algorithm that provides simple batching, no optimization	Yes
10	Rate algorithm	An algorithm that takes pieces at a set rate	No
11	Round robin for Grading/Batching	A method for sharing produce between multiple grading/batching locations of the same order	No
12	Prioritized production	Grading shall take order priority into account while grading	Yes
13	Remove finished orders	Once an order has been produced, it should be removed from locations and other lower priority orders should take its place.	No
14	Batches can be signed	The machine must be able to have batches accepted, which releases the batch, so that more can be produced at that location.	Yes

Table 1: Requirements specification

System overview

The system is built as two independent components. There is the Tracker component, which handles the physical aspects of running the machine and Grading component, which handles orders and piece evaluation. The two components in that manner represent the factory floor and back-office respectively. The two communicate with function calls. In the illustrations below the two components can be seen as they were conceived. The purpose of the component design is that in most ways the two are fully independent, and would in a full implementation be deployed as such.

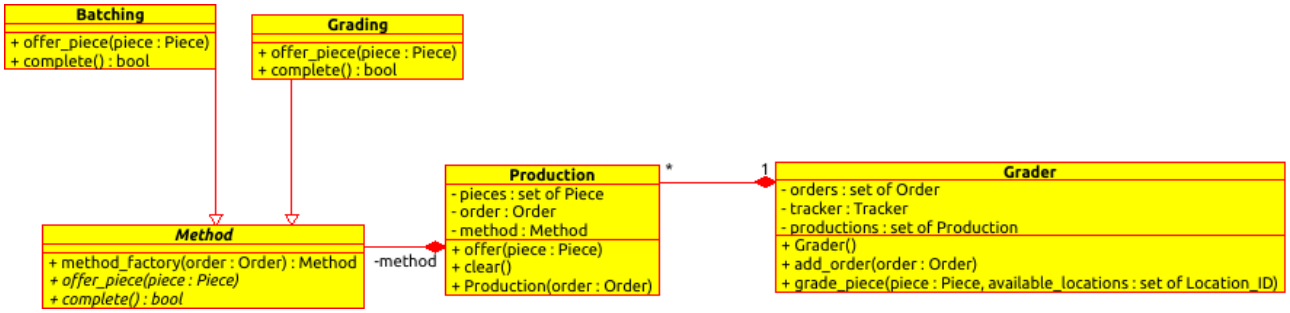

Illustration 2: Grading component

At the core of the Grader component is the Grader class. The Grader class receives new orders from the Environment, creating Productions based on the criteria given in the order, it also receives pieces for grading and returns which location the Tracker should route the piece to. Production initializes the method using the Method factory available in Method, which is the base class for the production methods Grading and Batching. At this point it should be mentioned that the Production class as such is not needed as it is written. But instead of removing it, it should be re-purposed into an aggregate for producing orders instead of batches. That change will allow for the creation of more intelligent methods, which require interaction between the batches being created at different locations. This flaw in the design was discovered too late to fix in time for the delivery deadline.

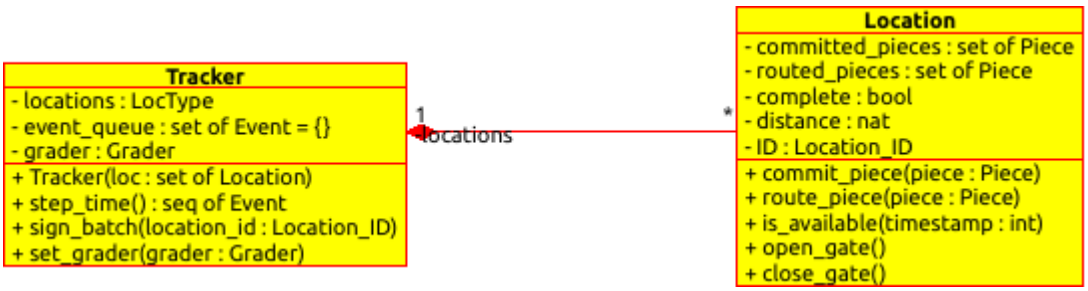

Illustration 3: Tracker component

The tracker component is represented by the Tracker class, which initializes a set of locations. It is the Tracker class which is time sensitive, where pieces are tracked forward using a simple event structure. The physical constraints on the machine are implemented in the Tracker class, while Location handles the state machine of each location. The state machine inside Location has four states, which are mainly concerned with determining if the Location is available for new pieces, which it is not when in *waiting_last_piece* and *full*. The state machine can be seen in Illustration 4.

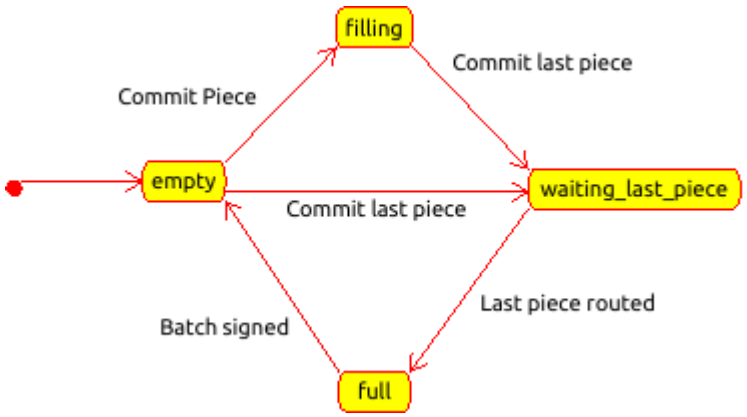

Illustration 4: Location State machine

Results

The model despite its limitations is fully capable of running through the test set, using both types of production methods. For testing the system I generated a set of 100 pieces with normal distributed weights and lengths, and two production orders, one for each type. This is the dataset I've used during development of the model and while expanding the pre/post conditions in the system. Sadly the model is very data driven, and as such combinatorial testing is unlikely to provide much of interest. Having reviewed the Test obligations, there are no obvious contradictions. The best method for verifying the model would be to build a visual simulation, where a visual inspection of the many moving parts in the system could be easily observed in concert with each other.

Conclusion

As mentioned previously some parts of the system have not been developed as well as they could have, and several features had to be cut. But on the whole I would say that the project is a success as the actual purpose of the project was to learn how to use modeling to develop systems at the proof of concept stage. In the model I have had opportunity to try out multiple different methods and aspects of VDM. Although it is still missing a handful of non-functional requirements and I have not had a chance at preparing a proper simulation using VDM Tools.

At multiple points during development I came to a stop and had to consider and accept that an additional feature or constraint really was needed to get the model to operate as I wanted it. The ease by which additional features can be added to a model, raises a serious risk of feature creep. As a consequence not all of the features of the system are covered by the requirements. This is mostly due to a lack of discipline on my behalf.