Design of meat processing systems with agent-based production control

Citation for published version (APA):

Document license:
CC BY-NC-ND

DOI:
10.1016/j.ifacol.2021.08.205

Document status and date:
Published: 01/01/2021

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

Download date: 08. Jun. 2022
Design of meat processing systems with agent-based production control

N. Paape * J.A.W.M. van Eekelen * M.A. Reniers *
* Eindhoven University of Technology, Eindhoven, The Netherlands
E-mails: {n.paape, j.a.w.m.v.eekelen, m.a.reniers}@tue.nl

Abstract:
Designing meat processing systems can be a complex process since this type of cyber-physical production system has two distinguishing properties: divergent product routings due to the cutting up of meat (known as co-production), and variability in the outcome of production processes (known as random yield). In this paper, we present a model-based design framework for meat processing systems, which can be used for performance prediction and system validation of a designed system. We show how an agent-based production control layer can be integrated into a model-based design architecture for meat processing systems. Finally, this framework describes how to route products through a system with co-production and random yield.

Copyright © 2021 The Authors. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0)

Keywords: Meat processing, cyber-physical production systems, model-based design, agent-based production control

1 INTRODUCTION

The transition in how we as a society consume meat is one of the driving factors for innovation in the meat processing industry: more sustainable production, higher food safety standards, better traceability, and an ever-increasing variety of ready-to-cook products, are just some of the trends seen in this industry. It is becoming increasingly complex to design meat processing systems that accommodate these developments, and to keep successfully doing so requires new design methods.

A meat processing system can be categorized as a type of cyber-physical production system (CPPS) with two notable characteristics. The first property is that these systems have diverging production processes due to the cutting up of products: most CPPSs only deal with converging production processes such as batching or assembly. The second property is that there is a natural variation in the animals which are processed; each animal has a different shape, size, quality, etc. Due to this natural variation, the properties of products resulting from a production process cannot be (fully) predicted. This makes it difficult to plan ahead which end products can be obtained from an animal, as this depends on what yield is obtained during the production processes. Systems with these properties can be classified as systems with co-production and random yield [Gerchak et al., 1996]. Other examples of systems with these properties can be found in a wide range of industries such as float glass manufacturing [Taşkin and Ünal, 2009], semiconductor manufacturing [Bitran and Gilbert, 1994], and lumber processing [Cid Yáñez et al., 2009].

Currently, the meat industry is in the process of moving away from dedicated processing lines in which every product type has its own processing line (flow-shop type systems), and is instead moving towards more flexible processing systems (job-shop type systems), in which different types of products are processed simultaneously, and in which each product has a unique path through the system [Tufano et al., 2019]. However, this transition also brings about a rising complexity in designing such systems. One area of research which can help in improving the design process of CPPSs is that of model-based design, in which mathematical models are used to aid in the design process [Biffi et al., 2017]. The model-based design (MBD) paradigm has many advantages. For example, it allows for performance analysis and validation of the proposed system early on in the design process. Compared to using prototypes, MBD reduces time-to-market, is less expensive, and it allows for performance prediction of scenarios which are not feasible to replicate (e.g., growth scenarios, or destructive scenarios). By building these models in a modular fashion, they can be easily reused in other projects. Examples of how MBD can be used in the food industry are shown in Penazzi et al. [2017] and in Pla-Aragónes et al. [2017], in which simulation models are used to aid system design in, respectively, the catering industry and the pig meat industry.

A big challenge in designing flexible meat processing systems lies in the production control layer; not only in material flow, but also in information flow. The variability in production yield causes uncertainty in product properties. This is less of a problem in dedicated systems, as all products have roughly the same route through the system, and if the system is designed correctly, then there should be enough information about that product to make control decisions on the flow of material (e.g. all products are first weighed and are routed through the system according to their weight). However, in a more flexible system, every product can have a different path through the system, which also means that similar products can have different levels of information availability. To give an example: three products at the same point in the production cycle
might have similar properties, but the first product has been weighed accurately, the second product is estimated to be in a certain weight range, and the weight of the last product is completely unknown. These three products might require a completely different path through the system, based on the difference in availability of information on the product.

One area of research which can help in designing the production control layer is that of agent-based production control, which is discussed in Shen et al. [2006], Leitão [2009] and Leitão et al. [2016]. As shown in Luder et al. [2013], agent-based production control fits very well into the model-based design paradigm; its distributed method of control offers high flexibility, modularity, and scalability.

However, the agent-based production control approaches in the aforementioned papers are almost exclusively aimed at the manufacturing domain. In manufacturing systems, product routing is largely decided by which end product is being produced, and agent-based production control approaches for manufacturing systems were developed with the preconception that the end product is known. In meat processing systems, an unfinished product is not allocated to a specific end product, as the processing yield determines which end product(s) can be obtained. For example, meat with a high-quality yield will be used for high-value end products, while lower quality meat will be used for lower-value end products such as ground meat. As a consequence, product routing in a meat processing system is generally only planned for the next few production steps, and not for the entire production cycle.

In this paper, a model-based design framework is introduced for meat processing systems with an agent-based production control layer. This framework can aid in:

- How to design a meat processing system in a scalable and modular fashion using model-based design.
- How an agent-based production control layer for these types of systems can be designed.
- How such an agent-based production control layer can be used to route products through the system.

This framework describes how meat processing systems can be designed using models. These models allow for evaluation of the system design in early phases of the design processes, both in prediction of performance measures (e.g. throughput), and in validation of the system design (e.g. are all products weighed accurately). Although intended for meat processing systems, the framework potentially extends to other types of systems with co-production and random yield, such as float glass manufacturing, semiconductor manufacturing, or lumber processing.

The paper is structured as follows. First, in Section 2, a small example of a poultry processing system is introduced. Next, in Section 3 the design process of a meat processing system is discussed. Section 4 defines how an agent-based production control layer can be designed, and Section 5 shows how it can be used to route products through the system. How the framework can be applied is explained in Section 6 using the example of Section 2. Finally, concluding remarks are given in Section 7.

2 EXAMPLE INTRODUCTION

In this paper, a fictive design for a small part of a poultry processing system — a fillet processing line — will be used to showcase the advantages of this framework. A product flow diagram of the proposed design is shown in Figure 1. The system takes chicken fillets and produces trays with batches of chicken fillets to be sold at supermarkets. At the system entry, poultry fillets flow into the system with a randomly distributed weight. These are transported and distributed through the system using conveyors, a merger, and a diverter. While being distributed through the system, the fillets are weighed using a weighing machine. The system then features two processing machines: a batching machine and a trimming machine, each with a buffer in front of them. The batching machine batches multiple fillets together in a bin until a target bin weight is reached, after which the bin is sent to the exit. Supermarkets often only pay for the target bin weight; every gram above the target bin weight is given away for free. Reducing this giveaway is essential for increasing profits; this problem is known as the bin covering problem [Peeters et al., 2017]. To help reduce giveaway, a fillet can first be sent to the trimming machine, which cuts a small piece off a fillet. This piece can be used for other purposes such as chicken nuggets. The difficulties in this example lie in designing the production control layer, and in performance prediction and validation of the proposed system design.

3 DESIGN OF A MEAT PROCESSING SYSTEM

The design process of a meat processing system starts with the requirements which the system must fulfill. With these requirements in mind, a design is proposed for the system. This design can be split into two layers: the plant layer, and the production control layer. The plant layer consists of the resources of the system (machines, buffers, etc.) and the connections between these resources. The fillet processing line of Figure 1 is such a plant layer design. The production control layer, as the name suggests, controls the plant layer to supervise production.

Designing a meat processing system can be a complex process, and model-based design can help in this process. The idea behind model-based design is that predictions can be made on the performance of the design early in the design process, by capturing how the design will function in its environment. To capture the behavior of the plant layer in a model, the products which are processed by these resources must also be modeled, as they are part of the environment of the plant. How the components of
the plant layer are modelled is explained below, and how the production control layer is built up using agents is described in the next section.

**Product:** A product has only properties and no behavior. A product’s model should describe these properties, such as product type or weight. An example of a product model in the meat processing industry would be a product with type “fillet” and weight “100 grams”. It is important that the model captures all the characteristics of the product which are relevant in describing the behavior of the system.

**Resource:** A resource is a component of the system which processes products. In meat processing systems, resources are typically inline machines with a number of inflow and outflow lanes, with these resources functioning mostly independently from one another. Examples of resources are machines, conveyors, buffers, and system in-/outfeeds. The model of a resource should describe its properties and its production behavior under control of its local controller. The model of the meat processing system should describe how these resources are connected in terms of product and information flows. The system designer decides on which abstraction level a resource is modeled; a resource model can represent a small part of a machine or even an entire subsystem. As an example, the resource model of the Trimmer in Figure 1 would describe its one inflow lane and two outflow lanes (for fillets and nuggets respectively), it would describe how long it takes to process a fillet, and it would describe how products are processed (e.g. fillets are trimmed into trimmed fillets and nuggets with a weight ratio in the neighborhood of 80%/20%, depending on some variance).

## 4 Agent-Based Production Control

The production control layer of a meat processing system supervises the flow of products through the system, which it does by controlling the resources in the system. The goals of the production control layer are to guarantee the fulfilment of orders, to maximize the value obtained during production, and to balance the utilization of the system’s resources. It accomplishes these goals by instructing the resources in what production mode they should be, and what processing steps they must perform on products.

Ideally, designing the production control layer happens in a manner that is modular, scalable, and in which components are re-usable. Agent-based production control satisfies all of these properties and has the added benefit that the agent models used in this methodology can be integrated seamlessly into the model-based design paradigm. As discussed in Leitão [2009], many different definitions for agents can be found in literature. In this paper, an agent is defined as a virtual representation of a (not necessarily physical) component of the system, which contains information on the component, and its behavior and decision making. In this framework three types of agents are proposed: the Product Agent (PA), the Resource Agent (RA), and the Production Control Agent (PCA). Every product and resource in the system is represented by exactly one agent, although one product or resource could comprise multiple “objects”, e.g. multiple pieces of chicken, or a group of machines.

This architecture bears resemblance to those used in holonic manufacturing systems; in holonic architectures such as PROSA and ADACOR, a component can have both a physical and decisional part [Leitão, 2009]. The goal of these holonic architecture is to bring benefits of holonic organization such as stability and adaptability to manufacturing through the use of autonomous building blocks. However, unlike in holonic manufacturing systems the components in our proposed architecture cannot be (fully) autonomous. For example, a PA cannot have autonomy over its own state, as the knowledge and information required to keep its state up-to-date is located in the RAs.

**Product Agents:** A Product Agent (PA) represents a product in the system and it describes its observed properties, which resource it is currently located at, and its routing through the system. The properties of a PA do not have to have a fixed value, but can also be a range, a distribution, or can be completely unknown if there is uncertainty on what the product’s properties are. For example, the PA might state that a products properties are between 90 ~ 110 grams. The routing of a PA is the sequence of operations by specific resources which are scheduled on the product, with an operation representing a production step by a resource in the system (more on routings can be found in Section 5). In a meat processing system an operation by a resource could be “Batcher.batch” or “Trimmer.trim”. The PA has no behaviour of itself. Other agents in the system can look up the properties and planned routing of the product.

**Resource Agents:** A Resource Agent (RA) represents one resource in the system and its main purposes are ex-
executing the required production steps and administering the information flow between resource and PA or PCA. When a product enters a resource, the RA obtains information on the product and the routing of that product from the corresponding PA. It then instructs the resource based on which operations are required on the product. In response it receives feedback on the execution of the operation, which the RA processes, and in turn creates, destroys, or modifies product agents to reflect the current state of the products.

A resource agent has one “receive/provide PA” port for every “product inflow/outflow” port of the resource. Whenever a product is about to exit a resource, the RA hands over the PA of that product to the next RA in the production line. This framework does not specify how such a handover must be executed, as this can vary depending on the requirements of the system. The RA can have multiple production modes, with a specific set of possible operations for each production mode. An example of production modes can be found in the Chapter of Figure 1, which has two modes in which it produces fillet batches of either 400 or 1000 gram. Lastly, the RA can communicate triggers for scheduling/planning to a PCA. Examples of such triggers are requesting the PCA to schedule the routing for a PA which the resource has finished processing or notifying that a failure has occurred in the resource, upon which the PCA must act.

Production Control Agents: A Production Control Agent (PCA) resembles the mechanisms through which production is controlled. It controls production through two types of actions: it can set the routing of product agents, and it can communicate to resource agents which production mode is required. These actions can be initiated by the PCA itself, or they can be triggered by an RA (as explained in the previous paragraph). One PCA can be connected to multiple RAs, and multiple PCAs can potentially coordinate with each other to achieve a system-wide goal (e.g., balancing of production lines). However, such coordination is not a trivial task. One solution might be to divide the system into groups of resources, with each group controlled by a singular PCA which schedules the routing for PAs in that group, while one or more higher-level PCAs coordinate the production targets for these lower-level PCAs.

Product agent uncoupling

One aspect that is unique to systems with co-production is that information about one product could potentially be derived from the properties of another product. Suppose there is a product with a known weight. If this product is cut in half, and one of the halves is later weighed, then the weight of the other half can be inferred. However, it is best practice to uncouple product agents from each other. If one were to design a system in which a coupling between PAs is possible, then a knowledge base with extensive bookkeeping would be required, which tracks not only what the correlation between PAs is, but also if these correlations are still relevant. For example, if either one of the halves is further processed before either is weighed, then the correlation might be lost. Such a knowledge base would be comparable to the temporal database with data dependencies as shown in Dori et al. [1996].

![Figure 3. The component interactions in a meat processing system with agent-based production control.](image)

## 5 PRODUCT ROUTING

A product’s routing is the sequence of operations by specific resources which are scheduled to be executed on the product. In other types of production systems, a product’s routing can generally be defined as a linear sequence of operations by resources [Lin and Solberg, 1991]. An example of such a linear sequence would be routing $r = \text{Conveyor1.move, Merger.move, Weigher.weigh}$. However, due to the characteristics of meat processing systems, a routing cannot always be described as a linear sequence. In this section the required extensions are discussed.

### Product routing and co-production

Firstly, in a meat processing system, a production process can yield multiple co-products from one product. One of the contributions in this framework is the ability to schedule the required operations for both the product, and the co-products which will be derived from this product in the future. This is accomplished by defining a product’s routing as a sequence that branches out whenever the product is cut up. Identifiers are used to indicate which branch belongs to which (co-)products. It is not necessary that an identifier relates to a single product; it could be that multiple (co-)products share an identifier. The branching of a routing due to co-production is indicated by square brackets, with an identifier preceding the subsequent routing for each co-product:

$$r = r_{\text{pre-cutup}}, \begin{cases} \text{identifier}_A \colon r_{\text{identifier}_A} \\ \vdots \\ \text{identifier}_Z \colon r_{\text{identifier}_Z} \end{cases}$$

### Product routing and batching

Product routings can converge when products are batched together. In these types of systems it is best practice to schedule a routing for a batch upon its creation, instead of trying to schedule ahead. The reason is that in meat processing systems it generally cannot be predicted which products will be batched together. Nonetheless, some use cases might require scheduling ahead: e.g. if very specific products must be batched together for a custom order. In these instances the appropriate solution is best chosen on a case-by-case basis.
Product routing and random yield

Due to random yield in production, there is unpredictability in how many products will be produced, and what the properties of those products will be. In this paper we assume that the creation of products during processing must be either deterministic and/or observable, meaning that an RA can either predict when and how many products are created, or can derive this information through feedback from the resource. The reason for this is that a resource agent must always know when and how many product agents must be created.

In some cases, the required routing for a product might depend on what yield is realized for that product. As proposed in Lipset et al. [2001], the subsequent routing of a product can be decided based on evaluation of one or more conditions, which is defined as conditional choice. Because of the earlier mentioned uncoupling between product agents, a condition can only depend on the properties of the product in question, or properties of a resource, and not on the properties of other products in the system. When describing a product’s routing, conditional choice is indicated by braces and a set of conditions, each condition followed by the product’s subsequent routing.

\[ r = \begin{cases} \text{condition}_A : r_{\text{choice}_A} \\ \vdots \\ \text{condition}_Z : r_{\text{choice}_Z} \end{cases} \]

As mentioned before, a PA property can have uncertainty. For this reason, a condition does not always evaluate to either true or false, it can also evaluate to unknown. Suppose that the weight of a chicken fillet is estimated to be between 90 ~ 110 gram, then the condition \( weight > 100 \) cannot be evaluated. A solution can be found in database-management languages such as SQL, where three-valued logic is used, in which a condition evaluates to either true, false, or unknown [Libkin, 2016]. A special \( ? \) operator is introduced, with the function \( ?(P) \) evaluating to true if predicate \( P \) cannot be evaluated to either true or false. For example, \( ?(weight > 100) \) would evaluate to true.

Routing example

To showcase how routings are formulated, a use case is considered for the fillet processing line of Figure 1. The partial routings of this system are as following:

\[ r_{\text{entry}} = \text{Conveyor1.move, Merger.move, Weigher.weigh} \]
\[ r_{\text{batching}} = \text{Diverter.to_Batcher, Buffer1.forward, Batcher.batch} \]
\[ r_{\text{trim}} = \text{Diverter.to_trimmer, Buffer2.forward, Trimmer.trim} \]
\[ r_{\text{trimmedfillet}} = \text{Conveyor2.move, Merger.move, Weigher.weigh} \]
\[ r_{\text{filletbatch}} = \text{BatchExit.store} \]
\[ r_{\text{nugget}} = \text{NuggetExit.store} \]

Suppose there is a use case in which fillets are trimmed before being batched if their weight \( w \) is over 100 grams. If their weight is under 100 grams, or if it cannot be determined (e.g. if the Weigher malfunctions), then the fillet is batched straight away. Upon entry, the PCA determines the fillet’s routing \( r \) for the entire production cycle. As can be seen below, determining the routing for a fillet’s entire production cycle can become convoluted and is not advisable. Batches of fillets created at the Batcher are scheduled to exit the system with \( r = r_{\text{filletbatch}} \).

\[ r = r_{\text{entry}} \begin{cases} w > 100 : r_{\text{trim}} : \begin{cases} \text{nugget} : r_{\text{nugget}} \\ \text{fillet} : r_{\text{trimmedfillet}} : r_{\text{batching}} \end{cases} \\ w \leq 100 : r_{\text{batching}} \\ ?(w > 100) : r_{\text{batching}} \end{cases} \]

6 APPLICATION OF THE FRAMEWORK

This section explains how the framework can be used for the model-based design of meat processing systems. The design steps for a meat processing system are as follows.

1. Define requirements of the system
2. Design plant layout (resources & their connections)
3. Model plant layer (resources & products they process)
4. Design production control layer through agent models
5. Validate the design with respect to the requirements

Iterate this process till the design is correct

Example of application of the framework

The proposed design method is showcased on the fillet processing line example of Figure 1 as introduced in Section 2. The design steps can be applied to the use case as follows:

1. The main requirement of the system is that it can produce batches of fillets and nuggets, within a specified production target and giveaway.
2. Figure 1 shows a flow diagram of the proposed design.
3. The exact resource models are not shown for brevity, but their general behavior is described in Section 2.
4. In this step the agents and their interactions are modeled. The resources and products in the system are represented by their respective agents, and one single PCA is required, which after weighing, decides if a fillet should be batched or trimmed, and sets the routing \( r \) of the fillet’s PA respectively as either \( r = r_{\text{batching}} \) or \( r = r_{\text{trim}} \).
5. Figure 4 gives an insight into the interactions between the RA Weigher and other system components, when the weigher processes a fillet. Figure 4a shows a sequence diagram of the interactions between the RA Weigher and other system components, when the weigher processes a fillet. Figure 4b shows a sequence diagram of the interactions (sequence diagrams are a UML formalism which can be used to capture the interactions between components [Fowler and Scott, 1999]). Finally, Figure 4c shows how the model of PA Fillet is modified as it is processed. The step-by-step scenario shown in the figure is as follows:

(a) RA Merger hands over PA Fillet to RA Weigher
(b) RA Weigher requests properties and routing from PA Fillet
(c) RA Weigher instructs Resource Weigher to weigh the next product (as per PA Fillet’s routing). The RA in turn receives feedback from the resource on the execution of the operation.
(d) RA Weigher updates information on PA Fillet
(e) RA Weigher triggers the scheduling of PA Fillet by the PCA, which sets the routing of the PA.
(f) RA Weigher hands over PA Fillet to RA Diverter


The framework described in this paper can be used to help in the design process of meat processing systems. The contributions of this framework are a modeling architecture which is effective for model-based design of both the plant and agent-based production control layer in a modular, scalable, and reusable fashion, and a method for product routing in meat processing systems.

The next step in our research will be the application and validation of the framework with a case study supplied by industry. The main challenge we predict to encounter in such an industrial case study is modeling the resources and agents of such a system. We wish to develop a methodological approach on how these models can be derived from formal system design specifications.

REFERENCES