Automated evolutionary restructuring of workflows to minimise errors via stochastic model checking

Luke Thomas Herbert [a]*, Zaza Nadja Lee Hansen [b] and Peter Jacobsen [b]

[a]DTU Compute, Lyngby, Denmark
[b]DTU Management, Lyngby, Denmark

Abstract: This paper presents a framework for the automated restructuring of workflows that allows one to minimise the impact of errors on a production workflow. The framework allows for the modelling of workflows by means of a formalised subset of the Business Process Modelling and Notation (BPMN) language, a well-established visual language for modelling workflows in a business context. The framework's modelling language is extended to include the tracking of real-valued quantities associated with the process (such as time, cost, temperature). In addition, this language also allows for an intention preserving stochastic semantics able to model both probabilistic- or non-deterministic branching behaviour. We further extend this formalism to allow for the introduction of error states which allow for both fail-stop behaviour and continued system execution. We explore the practical utility of this approach by means of a case study from the food industry. Through this case study we explore the extent to which the risk of production faults can be reduced and the impact of these can be minimised, primarily through restructuring of the production workflows. This approach is fully automated and only the modelling of the production workflows and the expression of the goals require manual input.

Keywords: Consequence Modeling and Management, Enterprise Risk Management, Industrial Safety and Accident Analysis, Reliability Analysis and Risk Assessment Methods, Safety Assessment Software Tools, Safety Management and Decision Making

* Corresponding author, lthhe@dtu.dk

1. INTRODUCTION

It is vital for all industries, for example the food industry, that workflows are generated which are safe and fulfil specific parameters and criteria like hygiene, cost, efficiency and speed (due to the perishable nature of the products in the food industry). A production workflow is for example cutting, forming or moulding a product or conducting quality control measures. Developing production and business processes is today predominantly an activity in which software tools are used to draw the process maps. The processes are analysed by hand and improved configurations are found by a process of trial and error, often taking too much time to arrive at an optimal practice due to the learning experience involved. There is therefore a need for a more efficient approach.

In this paper we present a framework for the automated restructuring of workflows that allows one to minimise the impact of errors on a production workflow. This framework allows for the modelling of workflows by means of a formalised subset of the Business Process Modelling and Notation (BPMN) language, a well-established visual language for modelling workflows in a business context. The frameworks modelling language is extended to include the tracking of real-valued quantities associated with the process (such as time, cost, temperature). In addition, this language also allows for an intention preserving stochastic semantics able to model both probabilistic- or non-deterministic branching behaviour. We further extend this formalism to allow for the introduction of error states which allow for both fail-stop behaviour and continued system execution. We employ stochastic model checking to efficiently explore the entire statespace of a workflow. The temporal logic PCTL is used to encode properties of interest for model checking (e.g. the probability that the next error-free product will come off the production line in less than 60 seconds).

We present an algorithm that allows for the weighted generation of PCTL queries that may be used to express a desired balance between the occurrence of errors and data quantities associated with the workflow. The expected mean values at points of failure can be calculated in turn, allowing for the expression of queries such as identifying the errors in operations which has the largest impact on production cost and/or time. These queries along with a model of an existing workflow, are used as inputs to an evolutionary algorithm which iteratively, through a process of mutation and cross-over, generates candidate improved workflows. The model checking of a weighted set of queries is used as a fitness function for determining the degree of improvement of candidate workflows. Being an evolutionary algorithm when a candidate workflow shows improvement it is used as the basis for the next round of mutation and cross-over. The software tool SBOAT is presented which implement our approach.

We explore the practical utility of this approach by means of a case study from the food industry. The case company is one of the largest Danish producers of baked goods. Their goal is to reduce the cost of waste in their production workflows. Through this case study we explore the extent to which the risk of production faults can be reduced and the impact of these can be minimised, primarily through restructuring of the production workflows. We discuss both the degree of improvement achieved by use of this evolutionary approach and the limitations of the approach and the additional work needed, compared to other process optimisation techniques, to make use of the approach.

2. RELATED WORK

In general by performing model checking to determine quantitative and qualitative properties of BPMN models, this work draws a comparison with a number of other BPMN analysis techniques outlined below. The selection of analyses discussed is not exhaustive, but covers the main approaches which have been widely referenced in the literature. They can broadly be defined as functional analyses, non-functional analyses and stochastic analyses.

In terms of the analysis of functional qualitative properties a wide range of approaches have been developed for BPMN. These are predominantly focused on the analysis limited sets of functional properties, such as proving the absence of deadlocks. The work of Ouyang et al. [8], [9] is the closest match to the type of translation approach taken in this paper. Here, translation of BPMN models is done directly into the web-services orientated BPEL [12] by means of an algorithm similar to what is presented in this thesis. However, the approach by Ouyang et al. is intended to support simulation through execution of the BPEL services with all the limitations that a simulation approach entails. Limitations are for example that statistical simulation can explore some situations, but cannot observe all behaviours. Safety properties which guarantee that specific behaviour will always, or, can never, occur need to be evaluated under all possible situations which simply cannot be achieved by the simulation method. The other key limitation is that simulations need to be executed for a certain amount of time.

A number of different approaches have been developed to analyse non-functional properties of BPMN models. In particular there has been a focus on determining timing properties of BPMN models. General quantitative analysis has only been identified as being explored by Prandi et. al. [10].

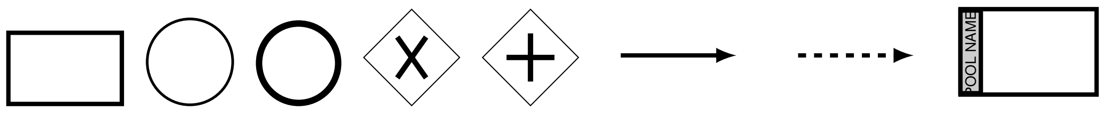
Analysis of BPMN models extended with stochastic properties has seen limited development with only two approaches identified of dealing with general models which exhibit both probabilistic and non-deterministic transitions. Prandi et. al. [10] have identified PRISM as ideally suited to the analysis of stochastic PRISM business processes. This effort involves conversion of PRISM models into a model expressed in the COWS [11], which in turn is converted into a model that can be analysed using PRISM [5]. This approach adds unnecessary complexity in that it is possible to convert the notation of BPMN directly into the PRISM modelling language, and then allow PRISM to impose a semantic interpretation without the additional semantic restrictions of going via COWS. Further, the translations PRISM to COWS and in particular from COWS to PRISM is loosely defined and, in the form described by the authors, not amenable to algorithmic translation. This approach, however, does allow the use of rewards. Consequently, the PRISM model checker is potentially able to perform analysis of both quantitative and stochastic properties of a business process. However, details of how such properties will be included in the original BPMN models is not described.

3. FRAMEWORK

We present a framework which uses an extended version of the BPMN language to automatically restructure of workflows so as to minimise the impact of errors on a production workflow. The software tool which implements this framework is called the Stochastic BPMN Optimisation and Analysis Tool (SBOAT).

3.1. Core BPMN

For the purposes of this paper, only a small subset of BPMN that is sufficient to illustrate the principles of this work will be used. Often called the *core* subset of BPMN, it consists of the eight elements found to be most commonly used in a large survey of real-world BPMN usage [6]; indeed more than 70% of models surveyed consisted only of these elements. The graphical elements of core BPMN are shown in fig. 1 and described below.



(a) Task (b) Start (c) End (d) Decision Gateway (e) Parallel Gateway (f) Sequence Flow (g) Message Flow (h) Pool

Figure 1: Core BPMN elements.

The process of modelling a workflow in BPMN involves composing a number of BPMN elements into a BPD. The intention is that a business process diagram captures the complete workflow of a business process, with separate sub-components of a workflow organised into separate pools. A BPD is formally defined here as simple partitions of a process graph with synchronisation arcs as defined in **??**, with the straightforward addition of a function to capture pools.

Definition 1 (Core BPD). *A core BPD is an extended process graph tuple BPD* $= (\mathbf{N}, \mathcal{F}, \mathbf{P}, \mathsf{pool}, \mathbf{L}, \mathsf{lab})$ *where* $\mathbf{N} \subseteq \mathbf{T} \cup \mathbf{E} \cup \mathbf{G}$*, is a set of nodes composed of the following disjoint sets:*

- *Tasks* $\mathbf{T}$*, are the basic actions performed as part of a business process.*
- *Events* $\mathbf{E} \subseteq \mathbf{E^S} \cup \mathbf{E^E}$*, where the disjoint sets* $\mathbf{E^S}$ *and* $\mathbf{E^E}$ *respectively represent start and end events.*
- *Gateways* $\mathbf{G} \subseteq \mathbf{G^D} \cup \mathbf{G^F} \cup \mathbf{G^M}$*, where the disjoint sets* $\mathbf{G^D}$*,* $\mathbf{G^F}$ *and* $\mathbf{G^M}$ *respectively represent exclusive decision gateways, parallel fork gateways, and parallel merge gateways.*

$\mathcal{F} \subseteq \mathcal{S} \cup \mathcal{M}$ *is a set of flow relations, where sequence flows* $\mathcal{S} \subseteq \mathbf{N} \times \mathbf{N}$ *relate nodes to each other and message passing* $\mathcal{M} \subseteq \mathbf{T} \times \mathbf{G^M}$ *is a relation between tasks and parallel merge gateways.* $\mathbf{P} \subset \mathfrak{P}(\mathbf{N})$ *is a set of disjoint pools and* $\mathsf{pool} : \mathbf{N} \to \mathbf{P}$ *maps nodes to a pool* $p \in \mathbf{P}$*.* $\mathbf{L}$ *is a set of unique labels and* $\mathsf{lab} : \mathcal{F} \to \mathbf{L}$ *is a labelling function which assigns labels to flows.*

The definition of a *BPD* given in definition 1 models business processes by using elements of $\mathcal{F}$ to define a directed graph with nodes which are elements of **N**. However, definition 1 allows for graphs which are unconnected, do not have start or end elements, and are free from various other properties which place them outside what is implied to be permitted in standard BPMN models. To ensure that a *BPD* describes a meaningful business process we have developed a set of structural semantics (well-formedness) rules [4] which enforce restrictions on connecting elements, pool boundaries, and message passing.

When performing quantitative analysis of a graph based process language we choose well-formedness conditions such that they impose the minimum semantic interpretation necessary to determine the control flow of a model. In the case of a BPMN *BPD*, it adds no more semantic interpretation than implied by the standard [7]. In the case of BPMN, when we have imposed restrictions they have been made only for simplicity and are discussed at length in previous work [4].

It should be noted that by combining several Core BPMN elements any element of the entire BPMN language can be simulated. Although a few constructs pose more complex issues. Of particular note are inclusive gateways which pose a challenge as the specification of their semantics include a non-trivial and non-local backwards search of the flow graph of the *BPD*. However, these can be addressed through the work of Christiansen, Carbone and Hildebrandt [1] who present a method to translate this construct into other BPMN processes with some restructuring of the overall *BPD*. In general, all elements of the full BPMN language can be incorporated by simply including a preprocessing step in the analysis of BPMN which translates non-core BPMN elements to core BPMN elements.

3.2. Extending Core BPMN

BPMN makes use of external conditions on decision gateways to select the outgoing flow from a decision point. These decisions are modelled by the set **L** and assigned to specific flows by the function lab introduced in definition 1. In practice, decision points in a business process will have outcomes which depend on some inherent property of the task or on outside factors. The idea is that at a decision point an active choice is made, and then that choice results in a number of different possible outcomes.

This behaviour, which is similar to a Markov decision process [13], where we wish to preserve the intention of actors in a process and still enable probabilistic behaviour, can be effectively captured by annotating the possible outcomes of specific decisions with pairs of labels and probabilities (l, p). We employ the following function to ensure meaningful assignment of these intention preserving probabilistic annotations.

Definition 2 (BPD Gateway Flow Probability Function)**.** *Given a BPD, a decision gateway probability function is a partial function* $\mathcal{P} : \mathcal{S} \times \mathbf{L} \to [0,1]$ *which for a node* $g \in \mathbf{G^D}$ *and label* $l \in \mathbf{L}$*, assigns probabilities to all outgoing sequence flows* (g,x)*, such that for a given* l*:*

$$\sum_{\forall x \in \mathsf{out}(g)} \mathcal{P}((g,x),l) = 1$$

Definition 2 ensures that all decision gateways have an associated probability and that the sum of all probabilities for a given label l is 1.

To enable quantitative analysis of business processes we add numerical data to our models by using the following function which associates positive real numbers with tasks in a *BPD*.

Definition 3 (BPD Task Reward Function)**.** *For a BPD a reward function for a task* $t \in \mathbf{T}$ *is a partial function* $\mathcal{R} : \mathbf{T} \to \mathbb{R}_{\geq 0}$.

This function captures the notion that certain nodes have some reward or cost associated with the task. The term *reward* is somewhat misleading: there is no practical distinction between costs and rewards, and we can use these annotated values to keep track of whichever quantities may be of interest in a process. We may associate as many reward structures as we wish with a given *BPD*, so that a single task may have multiple different numerical properties which are incremented when the task is performed.

Further, we allow some reward structures to be parametrised, allowing us to model limited resources involved in a process. In this case a reward has an associated upper and lower bound.

Note that the reward structures presented here are simple additions to the BPMN language, or other graph based process languages, and do not alter the implied semantics of the language.

3.3. Probabilistic CTL Property Specification

The main goal with this work is to be able to perform stochastic model checking of BPMN models. We will employ the property specification language called PCTL [5] based on classical continuous stochastic logic [3] extended to probabilistic quantification of described properties. This logic allows us to reason about properties relating to: timing, occurrence and ordering of events, reward values, transient and steady-state probabilities, and best- and worst-case scenarios. An implementation of the PCTL logic is employed by the PRISM model checker in its property specification language and has the following basic elements.

The key constructs in the PRISM property specification language, as it applies to Markov decision processes, are the P and R operators. The P operator refers to the probability of an event occurring, more precisely, the probability that the observed execution of the model satisfies a given specification. The R operator is used to express properties that relate to rewards (more precisely, the expected value of a random variable, associated with particular reward structure) and since a model will often be decorated with multiple reward structures, we augment the R operator with a label. For example, to determine the mean time to exhaust the supply of cake filler we would specify the following property:

$$\mathsf{R}_{time} =_{?} [F^{[0,\infty]}\, \mathit{filler\_empty}]$$

In this property we employ the time-bounded operator F^I, where $I \in [t_1, t_2] \in \mathbb{R}^2$, to describe if a subsequent boolean variable becomes true, it remains true. Further the operator U^I allow for specification of execution paths where one boolean variable holds until another is true, and G^I, which can be seen as the dual of F^I: expresses the fact that a condition remains, rather than becomes, true. Note that these operators can be combined in arithmetic expressions, allowing more complex measures to be expressed.

3.4. SBOAT

SBOAT is a *Stochastic BPMN Optimisation and Analysis Tool*, which allows a user to model, and annotate with rewards and stochastic branching, a business processes as a BPMN *BPD*. Analysis is specified using a PRISM style PCTL query and depending on the nature of the query one or number of results are calculated. At the core of SBOAT is the PRISM model checker [5] which performs analysis of individual models generated by the SBAT. An overview of the design of SBOAT is shown in fig. 2.

Designing a BPMN model and adding annotations to a model is done in the modeller, shown in fig. 3. Within SBOAT we employ an *adjacency list* [2] data structure to store the modelled *BPD* as a directed graph.

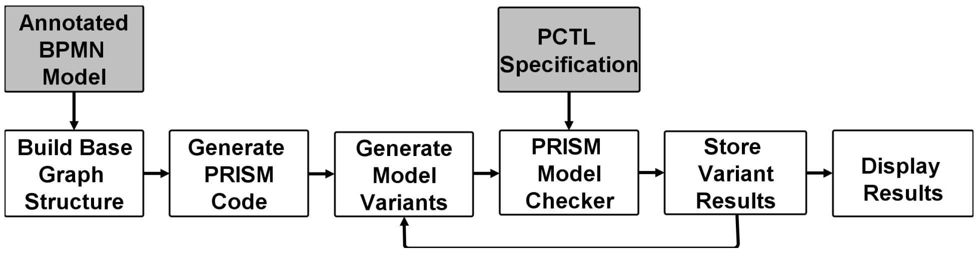
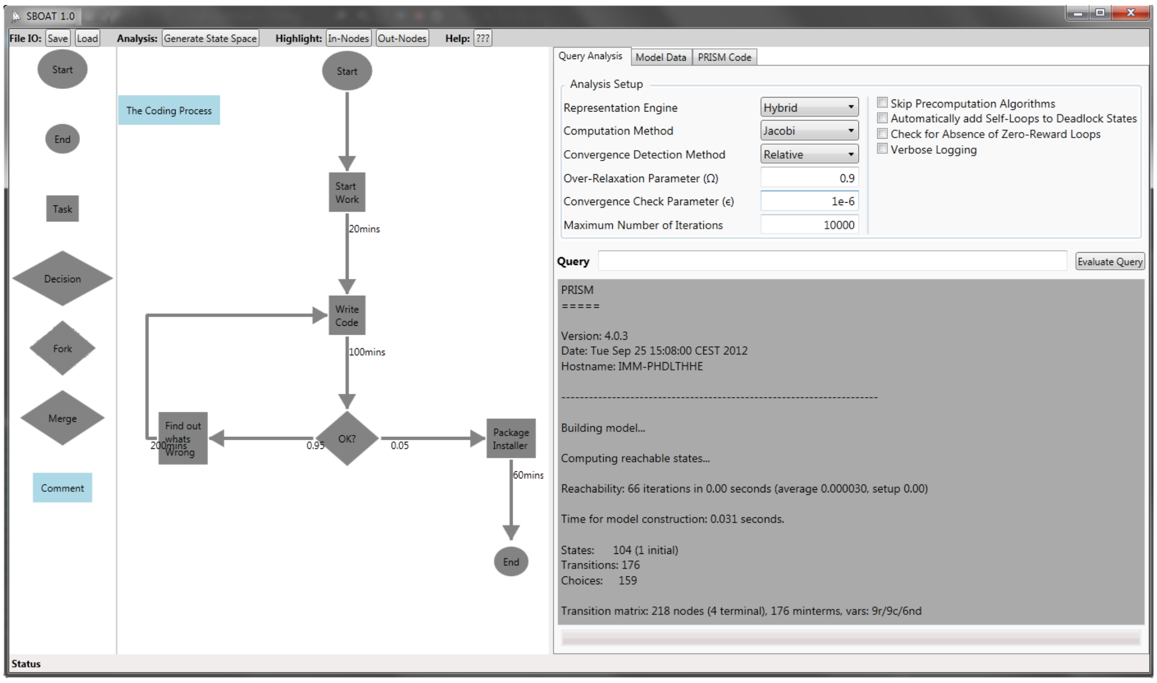


Figure 2: Overall design of SBOAT



Analysis Setup

Representation Engine	Hybrid
Computation Method	Jacobi
Convergence Detection Method	Relative
Over-Relaxation Parameter (Ω)	0.9
Convergence Check Parameter (ϵ)	1e-6
Maximum Number of Iterations	10000

- [ ] Skip Precomputation Algorithms
- [ ] Automatically add Self-Loops to Deadlock States
- [ ] Check for Absence of Zero-Reward Loops
- [ ] Verbose Logging

Query

```
PRISM
=====

Version: 4.0.3
Date: Tue Sep 25 15:08:00 CEST 2012
Hostname: IMM-PHDLTHHE

-------------------------------------------------------------------

Building model...

Computing reachable states...

Reachability: 66 iterations in 0.00 seconds (average 0.000030, setup 0.00)

Time for model construction: 0.031 seconds.

States:      104 (1 initial)
Transitions: 176
Choices:     159

Transition matrix: 218 nodes (4 terminal), 176 minterms, vars: 9r/9c/6nd
```

Figure 3: SBAT 1.0 User Interface

This implementation approach allows nodes and edges to be of any data type, where edges implement a simple interface that defines connections to nodes. This flexibility also supports almost any type of reward annotation to be stored in nodes or edges. Further, the data structure is mutable and cloneable making it possible to build a suitable data structure for a wide range of possible graph based process languages. Internally, the data structure keeps a dictionary from nodes to a unordered list of edge elements.

The data structure used is serializable allowing for generation of XML files to store models. In the case of BPMN models, we emit files that comply with the XML based BPMN file format [7], using the annotation features inherent in the BPMN file format to store rewards and stochastic branching probabilities. When importing data files we parse the BPMN XML and at that stage checks for the well-formedness of the model are made as new nodes and edges are added.

3.5. Evolutionary algorithm

The evolutionary algorithm we have developed to optimise business workflow processes employs a genotype-style representation of the optimization problem, variation and selection operators, and a fitness function. However, many details are quite different from typical evolutionary approaches. The algorithm performs optimisation of a BPD by performing modifications directly on the BPD ensuring that the final improved process is also a BPD and requires no special interpretation by end users. The description of the algorithm in this section is intended only to explain the principle of this method and not detail the mathematical proofs behind it.

First, an initial population of BPD variants is generated. Due to the computational expense of performing quantitative model checking of a BPD we filter these variants using well-formedness (structural semantics) criteria and functional requirements, before evaluating their quantitative properties.

The evolution of the initial population takes place for a number of generations determined by limit. For each generation, a population of a given size is generated. This population is produced by selecting pairs from the previous generation, in a fashion that is proportional to their fitness score. This pair is used to generate a new variant BPD using a crossover operator. Mutation is achieved by performing a number of alterations of a BPD dictated by the mutation rates. If a variant proves to be well-formed and meets functional requirements it becomes part of the next generation.

Finally, once the generation limit has been reached, the highest scoring member of the final generation with regard to the optimisation goals, becomes the optimized BPD. Before returning this optimised BPD, any redundant components are removed.

To enable the combination of multiple weighted objectives, and to have sets of optimization goals in which both rewards and event probabilities can be expressed, we employ a set of optimisation goal tuples to define an individual optimisation goal using PCTL formulae. This is because in practice a production process is frequently optimised with regard to multiple quantitative properties. We use a set of optimisation goal tuples for this purpose. For a set of optimization goal tuples, we evaluate the relative improvement of a new production process BPD compared to an existing BPD using an optimisation goals scoring function.

Functional requirements allow the expression of properties which must hold for any future production process BPD derived from a BPD. Like optimization goals, functional requirements will be defined using PCTL formulae, however, in this case we will require that probabilities or reward values within the query are explicitly defined, such that the return value of the query is a Boolean variable. This is to ensure the functional requirements for each individual can be quickly evaluated as either being true or false.

A key step in our algorithm is the selection of members of a current generation used to derive the next generation. Here we employ stochastic sampling with limited replacement. In essence, each member of a current generation is mapped to a contiguous segment of a line, such that each individual's segment is proportional in size to its fitness. A random number is generated and an individual A whose segment spans the random number is selected. The process is repeated to obtain a partner with the restriction that if A is selected a new sample is chosen.

When generating variants we employ the traditional evolutionary algorithm approach of constructing a separate genotype representation upon which to perform modification of a BPD. Our approach allows the genome structure to closely reflect the phoneme structure. Encodings with this property are believed to make the evolutionary algorithm more robust (i.e. reduce the probability of fatal mutations), and also improve the capacity of a system for adaptive evolution.

We employ an adjacency matrix style representation of the underlying graph structure of the BPD for our genotype, where each matrix element is a vector which stores the reward structures associated with the given node of a BPD. The phenotype is simply the BPD that is derived from this matrix representation.

Crossover follows naturally from the structure of the genotype representation. Instead of creating a child by swapping information from two parents based upon one or more points in a linear structure as is commonly done, we use a rectangular section of the matrix structure selected at random. An offspring is then created by using information from inside the rectangle of one parent, and outside the rectangle of the other parent.

Mutation is also defined as a mathematical operator, which is applied to specific elements of the matrix representation of a BPD. This compliments our crossover operator by injecting small local changes to a BPD. We define mutation to allow for considerable variation of a source BPD. This definition allows mutations to have two effects on a BPD:

- **Re-sequencing:** This modification alters the BPD element which defines sequence flows. Specifically, it alters the relation between two nodes in the sequence Low, replacing the destination node with a different node, and reconnecting any excluded nodes to follow after the re sequencing. In effect, randomly reordering the sequencing of a number of tasks in the BPD.
- **Parallelization:** This modification functions by injecting pairs of parallel merge and fork gateways. These can be injected at any point other than at start and end elements, and the nodes between the injected gateways are initially all assigned to one of the parallel paths. Note that when this is combined with the resequencing operator both parallel branches will eventually contain nodes.

4. CASE STUDY

Baked Goods A/S has two production lines. Line 1 develops cakes and pastries and line 2 develops baked goods like sausage rolls and pizzas. The company has an approximate overall waste of 20% for all products during the whole production cycle. The waste is calculated at different places on the production line, but currently it is only possible to get the overall waste percentage of a product. The production process is shown in Figure 2. The dough is prepared, then the product, for example a cake, is prepared and made, then baked or frozen, followed by packaging and then shipment. Dough leftover during production is reused in the preparation of new dough. Quality checks take place during and after production as well as after freezing where defective products are discarded.

Depending on the line and product, the packaging is either automatic or manual and either controlled for number of pieces and weight or just number of pieces.

The processing steps are as follows;

1. Dough mixing. The ingredients for the dough are mixed in vats with a screw-hook mixer. Water and flour are delivered automatically through transport tubes, all other ingredients are manually added including ice.
2. Lamination. If specified the dough is laminated, meaning folded in on itself and rolled out. This takes place continually. Dough is supplied into a dispensing system. The dough is split into two sheets of continues length, and if specified, a layer of fat is supplied in between these layers. The lamination step can be skipped by using the second dispensing machine situated further up the production line.
3. Make-up and filling. Make-up refers to cutting the shapes out from the dough carpet. The product can also be filled with creams etc.
4. Leavening. This is done to induce a faster rising of the product in order to achieve the wanted volume of the product. This is done by raising the temperature and having a high humidity. This will promote the growth of yeast producing gas and the humidity will counteract surface drying allow for the dough to expand without generating stress cracks etc.
5. Baking. A baking step can be used after leavening.
6. Freezing.
7. Packing.

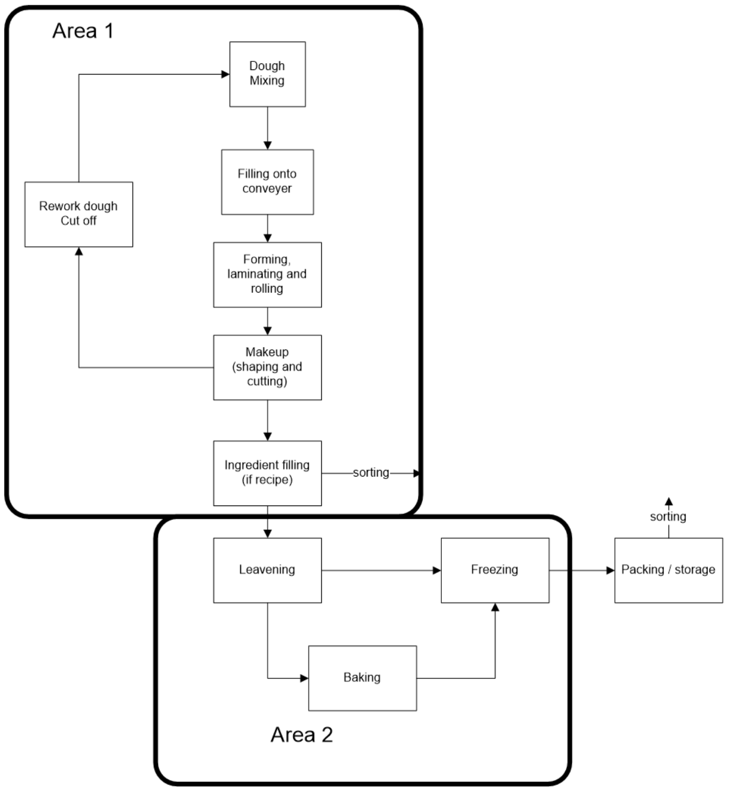


Figure 4: The production process at Baked Goods A/S.

4.1. Analysis of the case study

To illustrate an application of this method we will consider a specific example of a simple production process inspired from the baked goods case study involving bread production. To employ the method, one begins by building a BPD model of an existing production process. fig. 5 is an example of such a process which is annotated with rewards and information about its stochastic behaviour. This naively-designed production process consists of two processes, Conveyor Belt modelling the actions of the machines on a conveyor belt, and Filling Robot which models the actions of robot which fills the dough with cream etc. when needed.

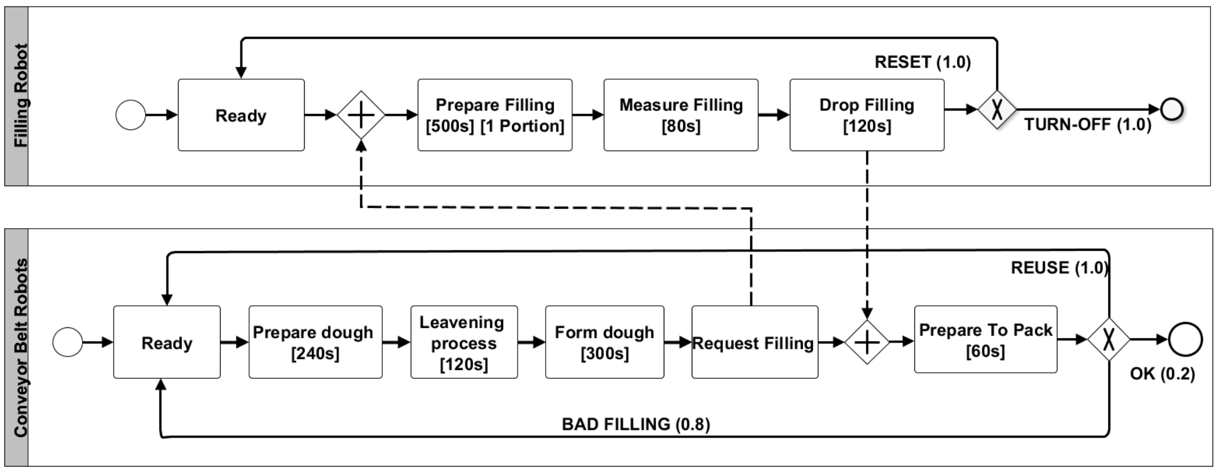


Figure 5: BPD model of a baking workflow process before optimization.

For the production process described in fig. 5 it would be desirable to see an improvement in the time taken for a conveyer belt machine to complete baking a cake. This is in other words the optimisation goal. Further, it would also be desirable that the rate of filling consumption and the consequent probability, given a specific filling stock size, of running out of the filling is kept as low as possible. This is the second optimisation goal.

In addition to the optimisation goals, a number of functional requirements exist for this process (formally expressed using the temporal logic PCTL in SBOAT). These requirements describe the sequences the steps the process have to be in:

1. The baking of the dough should take place before the leaving process of the dough.
2. All dough making, leaving, cut and filling, must take place before a cake is packed:
3. The conveyer belt machine cannot pack the cake before it has received filling.
4. The Filling Robot must ensure that a filling has been prepared and measured before it is sent to the conveyor belt.
5. When the conveyer belt determines that a bad dose of filling has been received it must immediately request a new dose.

Note that the final functional requirement (item 5) is not currently satisfied by the initial BPD shown in fig. 5.

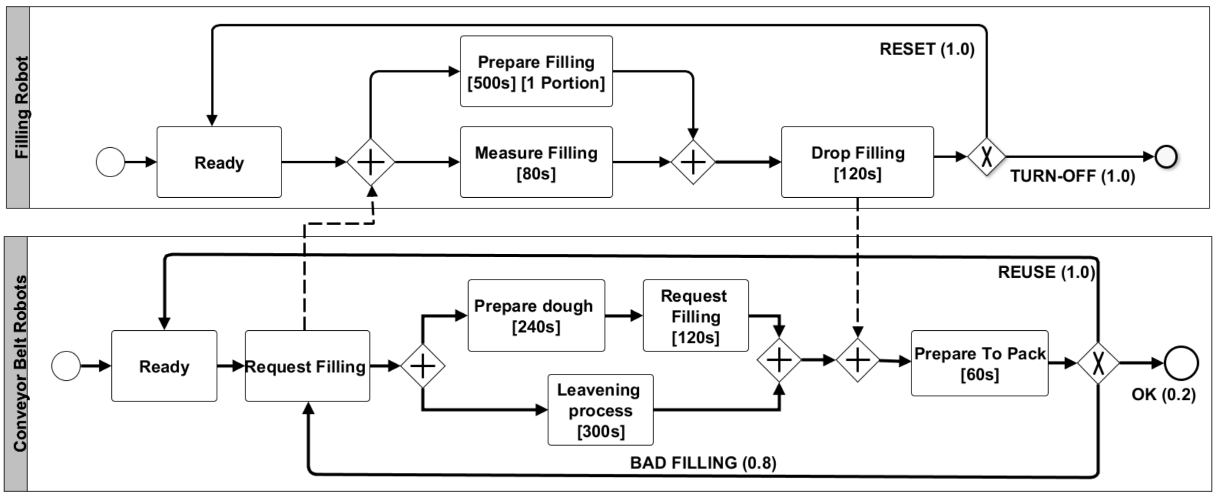


Figure 6: BPD model of a baking system after 28 generations of population size 500 optimisations.

fig. 6 illustrates one possible outcome of applying our optimisation methods to the BPD shown in fig. 5. Specifically, this is the outcome of 28 generational improvements of population size 500 of the process. Note that the new functional requirement (item 5) is now satisfied. In the case of this example the rates for sequencing and parallelizing are set so that the Mutate function ensures that considerably more re-sequencing modifications are performed than parallelization modifications.

In this run of our optimisation method we have identified two opportunities to parallelize actions. Within the Filling Robot process, the filling can be prepared and measured at the same time. In the conveyer belt process, it is possible to prepare dough and cut it while the dough simultaneously leaves. In both cases this saves time, as when performing actions in parallel only the path with the slowest behaviour is counted towards the parallel sections contribution to the reward value.

We have also determined that, in this simplified example, the conveyer belt process always orders filling to finish the cakes but the conveyer belt machines must wait while the Filling Robot performs its operations and then returns the filling. As the filling will inevitably be needed, and only the packing of the cake needs to be done after the dough has been made, it is within the functional requirements for the process, and results in a considerable time saving to order the filling immediately before even preparing the dough. This ensures that there will be no delay imposed on the Conveyer Belt process by the actions of the Filling Robot process. This simplified optimisation example does not violate the functional requirements and results in a significant reduction of the time taken for the execution of the production process.

Existing languages for the modelling of business processes such as BPMN, UML activity diagrams or YAWL lack a formalised semantic basis which would enable formal analysis and subsequent automated scheduling. Further, these languages do not allow for modelling stochastic behaviour or provide mechanisms to effectively track the consumption of resources during execution. These aspects are therefore the key strengths of this optimisation method as no other method, to our knowledge, has all these features. Further, it should be noted that our method by employing the PRISM tool calculates exact values. However, this need for precision also means that a disadvantage of our approach is that it requires detailed knowledge of the workflow processes being optimised. Another disadvantage of our method is that to use the optimisation schedule in practice great computing power is needed which can be both expensive and time-consuming. However, our method allows for automatic optimal scheduling with mathematical precision and within specific parameters which can help organisations limit waste of, for example, energy or material as well as optimise production with regard to parameters such as time, human resources and cost.

5. CONCLUSION

In this paper we have presented a framework for the automated restructuring of workflows that allows one to minimise the impact of errors on a production workflow. We did this by means of a formalised subset of the Business Process Modelling and Notation (BPMN) language. The frameworks modelling language is extended to include the tracking of real-valued quantities associated with the process (such as time, cost, temperature). In addition, this language also allows for an intention preserving stochastic semantics able to model both probabilistic- or non-deterministic branching behaviour. We further extend this formalism to allow for the introduction of error states which allow for both fail-stop behaviour and continued system execution. We employed stochastic model checking to efficiently explore the entire statespace of a workflow. The temporal logic PCTL is used to encode properties of interest for model checking (e.g. the probability that the next error-free product will come off the production line in less than 60 seconds). We presented an algorithm that allows for the weighted generation of PCTL queries that may be used to express a desired balance between the occurrence of errors and data quantities associated with the workflow.

We explored the practical utility of this approach by means of a case study from the food industry. The case showed that the framework could be used to reduce the risk of production faults and that the impact of these can be minimised by restructuring the production workflow.

The key strength of this approach is fully automated and only the modelling of the production workflows and the expression of the goals require manual input.

Further research will focus on refining this framework and the software tool, SBOAT. We hope to release this tool in late 2014 and use it in several case studies. This will allow for a more extensive formalised exploration of the scope and parameters of this method.

ACKNOWLEDGEMENTS

We would like to thank the Inspire project for funding.

REFERENCES

[1] D. R. Christiansen, M. Carbone, and T. Hildebrandt, “Formal semantics and implementation of BPMN 2.0 inclusive gateways”, in *Proceedings of the 7th international conference on Web services and formal methods*, M. Bravetti and T. Bultan, Eds., ser. Lecture Notes in Computer Science, vol. 6551, Berlin, Heidelberg: Springer-Verlag, 2011, pp. 146–160, ISBN: 978-3-642-19588-4. DOI: `10.1007/978-3-642-19589-1_10`.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd. The MIT Press, 2009, ISBN: 0262033844.

[3] H. Hansson and B. Jonsson, “A logic for reasoning about time and reliability”, *Formal Aspects of Computing*, vol. 6, no. 5, pp. 512–535, 1994, ISSN: 0934-5043. DOI: `10.1007/BF01211866`.

[4] L. Herbert and R. Sharp, “Quantitative analysis of probabilistic BPMN workflows”, in *ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE2012))*, ser. ASME Conference Proceedings, (Awaiting Publication), Jul. 2012.

[5] M. Z. Kwiatkowska, G. Norman, and D. Parker, “PRISM 4.0: verification of probabilistic real-time systems”, in *Proceedings of the 23rd International Conference on Computer Aided Verification (CAV’11)*, G. Gopalakrishnan and S. Qadeer, Eds., ser. Lecture Notes in Computer Science, vol. 6806, London, UK: Springer-Verlag, 2011, pp. 585–591, ISBN: 978-3-642-22109-5. DOI: `10.1007/978-3-642-22110-1_47`.

[6] M. Z. Muehlen and J. Recker, “How much language is enough? theoretical and practical use of the business process modeling notation”, in *Proceedings of the 20th international conference on Advanced Information Systems Engineering*, ser. Conference on Advanced Information Systems Engineering 2008, Berlin, Heidelberg: Springer-Verlag, 2008, pp. 465–479, ISBN: 978-3-540-69533-2. DOI: `10.1007/978-3-540-69534-9_35`.

[7] Object Management Group, “Business process model and notation (BPMN) 2.0”, Object Management Group, Needham MA, USA, Standards Document formal/2011-01-03, Jan. 2011. [Online]. Available: `http://www.omg.org/spec/BPMN/2.0/`.

[8] C. Ouyang, M. Dumas, and A. H. M. T. Hofstede, “Pattern-based translation of BPMN process models to BPEL web services”, *International Journal of Web Services Research (JWSR)*, vol. 5, no. 1, pp. 42–62, 2007. DOI: `10.1.1.143.3118`.

[9] C. Ouyang, M. Dumas, A. H. M. ter Hofstede, and W. M. P. van der Aalst, “From BPMN process models to BPEL web services”, in *Proceedings of IEEE International Conference on Web Services*, Washington, DC, USA: IEEE Computer Society, 2006, pp. 285–292, ISBN: 0-7695-2669-1. DOI: `10.1109/ICWS.2006.67`.

[10] D. Prandi, P. Quaglia, and N. Zannone, “Formal analysis of BPMN via a translation into COWS”, in *Proc. of the 10th international conf. on Coordination models and languages*, ser. COORDINATION 2008, Berlin, Heidelberg: Springer-Verlag, 2008, pp. 249–263, ISBN: 3-540-68264-3, 978-3-540-68264-6. DOI: `10.1007/978-3-540-68265-3_16`.

[11] R. Pugliese and F. Tiezzi, “A calculus for orchestration of web services”, *Journal of Applied Logic*, vol. 10, no. 1, pp. 2 –31, 2012, ISSN: 1570-8683. DOI: `10.1016/j.jal.2011.11.002`.

[12] O. for the Advancement of Structured Information Standards (OASIS), “Web services business process execution language (WS-BPEL) version 2.0”, Standards Document WSBPEL-v2.0-OS, Apr. 2007. [Online]. Available: `http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html`.

[13] D. J. White, *Markov decision processes*. New Jersey, USA: John Wiley & Sons, 1993, ISBN: 9780471936275.