Configuring Configurable Process Models Made Easier: An Automated Approach

D.M.M. Schunselaar[1⋆], H. Leopold[2], H.M.W. Verbeek[1⋆], W.M.P. van der Aalst[1⋆], and H.A. Reijers[1,3⋆]

[1] Eindhoven University of Technology, P.O. Box 513, 5600 MB, Eindhoven, The Netherlands
{d.m.m.schunselaar,h.m.w.verbeek,w.m.p.v.d.aalst,h.a.reijers}@tue.nl

[2] WU Vienna, Welthandelsplatz 1, 1020 Vienna, Austria
henrik.leopold@wu.ac.at

[3] Perceptive Software, Piet Joubertstraat 4, 7315 AV Apeldoorn, the Netherlands

Summary. Configurable process models have shown their usefulness for capturing the commonalities and variability within business processes. However, an end user will require an abstraction from the configurable process model, which is a highly technical artifact, to select a suitable configuration. Currently, the creation of such an abstraction requires considerable steps and technical knowledge. We provide an approach to construct such an abstraction automatically on the basis of an understanding of common concepts underlying process models on the one hand and automated analysis techniques on the other. Our approach also guarantees the consistency between the configuration choices of the end user. A positive yet preliminary evaluation with business users has been carried out to test the usability of our approach.

Key words: Automatic, Configuring, Configurable Process Model, Concepts

1 Introduction

Configurable process models form a well-studied and highly evolved formalism for capturing the commonalities and variability between (similar) process models [1]. However, to obtain the exact configuration from a configurable process model that best suits a particular context is in many respects still a challenge.

Early approaches have mostly focused on the formalism to specify a configurable model and to keep configuration choices consistent, but did not guide an end user through the entire configuration process [2]. Later work incorporated so-called "auto-complete" features to automatically set configuration points which otherwise would lead to incorrect models [3]. More recent work has incorporated guidance by making end-users go through an electronic questionnaire: Its questions relate to the configuration options, while the answers to such questions can be mapped to configuration choices [4]. At this point, the creation and maintenance of such questionnaires to a large extent relies on manual work. For instance, in the questionnaire-based approach

⋆ This research has been carried out as part of the Configurable Services for Local Governments (CoSeLoG) project (http://www.win.tue.nl/coselog/).

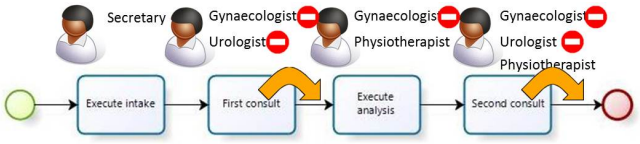

Fig. 1: Example configurable model consisting of 4 steps and 4 resources.

an expert has to define the questions to be posed to the end-user and must map the configuration settings to answer options. This is an intricate and laborious activity, particularly if the size of the configurable process model is large. Furthermore, changes to the configurable model may require elaborate inspection of the manual work that has already been conducted. Finally, questionnaire-based approaches rely on the help of experts to use the instrument and are thus not directly applicable off-the-shelf.

In this work, we provide support for the configuration process that is both automatically generated and universally applicable, i.e. independent of the model domain. To this end, we exploit the notion of general concepts that are at the core of many process modelling techniques. In this paper, we rely on the meta-model of APROMORE [5] to identify instances of these concepts, i.e. *resource* instances, *variable* instances, and *activity* instances. But our approach can be extended with other concepts. Our approach allows for an automatic identification of the concepts and their inter-relations in a configurable process model. The application-domain expert can then use these insights to make high-level decisions that tune the configurable model towards its intended use without the need to go through each and every element of the configurable model. For instance, if we take the model from Fig. 1 as our input, our approach would automatically deduce, amongst others, *gynaecologist* and *consult* as instances of the resource concept.

To optimally guide the user in setting their preferences, we developed a so-called *consistency graph*. This consistency graph is automatically constructed from the configurable model and signals users when they specify contradictory requirements. For instance, the user wants to include an activity but does not have the resources capable of executing it. Based on the requirements of the user, we infer the model(s) that best fit these. To show the business value of our approach, we have applied our approach on a real-life case study and evaluated its use with experts from a consultancy firm which is active in de healthcare ICT. In this evaluation, we used a configurable process model that captures the variety of process set-ups within Pelvic Floor Examination units in hospitals. From this configurable process model, we automatically distilled the concepts present. With the concepts, we automatically constructed the consistency graph without the need of any domain expert. Next, by presenting a view on the consistency graph and using this view to set their preferences, we demonstrated to the end-users how a model can be derived that best fits their needs. This paper discusses their feedback on this approach.

The structure of this paper is as follows: Sect. 2 lists the related work. In Sect. 3, we present the consistency graph, how we construct the consistency graph, and how we can deduce the configuration(s) using the consistency graph. Section 4 contains the

implementation details of our approach. The evaluation of our approach is presented in Sect. 5. Finally, the conclusions and future work are presented in Sect. 6.

2 Related work

Various approaches exist to provide an abstraction of the configurable process model as to simplify the configuration process [1]. In the approach that is described in [6], facts are set by posing questions to the user. For instance, "Shipping via DHL" is a question which is used to deduce the fact on the carrier for a package. Within the questionnaire-based approach, one can define constraints over the possible facts, e.g., at least one shipping company has to be selected. When the various facts have been set, these facts are used to set configuration options for the various configuration points. Within this approach, the questions about the facts have to be created by hand. Our approach can aid in this since the facts are related to instances of concepts, e.g., *Shipping* and *DHL* would both be concept instances.

In [7], the `Provop` approach is presented to use contextual information in the configuration of the configurable process model. Based on facts, various configuration options are selected, for instance, whether the "Quality Relevance" is high. Next to setting the facts directly, `Provop` also offer the possibility to reason over the facts, e.g., setting a particular value for a fact can have a cascading effect on another fact. Like the questionnaire-based approach, also `Provop` has been used in real-life case studies and its applicability and usability have been clearly demonstrated. Again, our approach can act as an intermediate approach to ease the manual work of distilling which facts to ask the user.

Various approaches exploit feature models to abstract from the configurable process model at hand. Feature models [8] are a way to capture aspects as well as the interdependencies between features. Feature models allow for a hierarchical decomposition of features making it possible to define one's preference at different levels of granularity. Various papers have brought the feature models to the area of configurable process models, e.g. [9], [10], and [11]. As with other approaches, the construction of the feature model is a manual task. Our approach can indicate which features play a role in the process model.

Finally, in [12], the authors present an approach for querying a repository of models. In order to query this repository, the user has to design parts of a process model which are matched to process models in the repository and all process models containing these parts are returned. This approach is applicable to our setting, i.e., instead of querying a repository of models, the various models obtainable from the configurable process model are queried. However, this requires modelling skills from the end user, not necessarily present, and, the query being declarative in nature, requires the end user to inspect the returned process models to learn what is (not) possible in them.

In summary, the main limitation of existing approaches is the manual link between the configurable process model and the abstraction presented to the user, and requiring skills not necessarily present with the end user. In the next section, we present our way to improve on the state of the art by automatically deducing that abstraction without requiring particular skills of the end user.

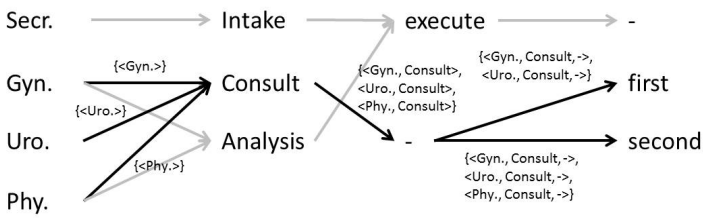

Fig. 2: The concept graph belonging to the model in Fig. 1. Some relations between the concepts have been grayed out to better indicate the contexts belonging to other relations.

3 Configuration Space Pruning

In this section, we introduce our configuration approach. We start by giving a general overview. Then, we discuss how we derive the concepts from the process model and how we use a consistency graph for the configuration.

3.1 Overview

As mentioned, we use the meta-model of APROMORE for the identification of instances of concepts. If we, for instance, consider the model in Fig. 1, we have 4 activities being executable by the resources as indicated, e.g., the activity *First consult* can be performed by the *Gynaecologist* and the *Urologist*. Taking a more detailed perspective, we can decompose activities into so-called business objects, actions, and business object modifiers. Hence, we associate the model in Fig. 1 with the resource instances *Secretary*, *Gynaecologist*, *Urologist*, and *Physiotherapist*, the business object instances *intake*, *consult*, and *analysis*, the object modifier instances *first* and *second*, and the action instance *execute*.

All these instances are combined into a single consistency graph such that the user can reason about these concept instances without going into each and every activity. We define the consistency graph on the entire configurable process model to present the user with a complete overview of the concept instances in the potential instantiation. By presenting a complete overview, the user can also indicate their preference on the non-configurable parts. If the non-configurable part is incompatible with the user's organisation, this is notified at the earliest possible moment.

As not all the concepts have to be equally important, the user has the option to define a partial order on the concepts. In this way, we are able to present a view on the consistency graph starting from the most important concept. Let's assume the user has the following preferences with respect to the importance of concepts: (1) resource (the most important), (2) business object, (3) action, and (4) business object modifier (the least important). In that case, we would obtain the concept graph as depicted in Fig. 2. Note that we introduced a dummy "-" when a particular instance of the respective concept does not exist.

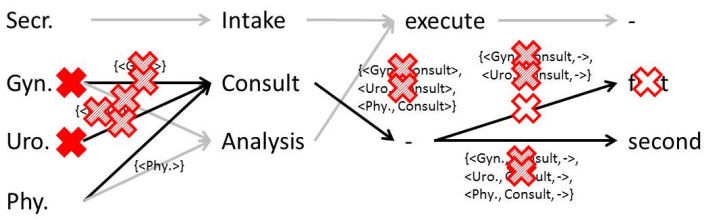

Fig. 3: The concept graph from Fig. 2 enriched with the information that the user does not have gynaecologists and urologists in their organisation. The completely filled crosses are the user's choice, the crosses filled with diagonal lines are directly deduced from the user's choice, and the crosses without a fill are deduced from the combination of the user's choices.

Apart from the concept instances, the consistency graph also contains relations (edges) between instances based on the ordering of concepts, e.g., the *consult* is performed by the *gynaecologist*. The relations are decorated with contexts. These contexts indicate when two concept instances are related, e.g., the relation between - and *first* only exists if the resource is either the *Gynaecologist* or the *Urologist*, the business object is a *consult*, and the action is -.

After having deduced the concept graph from the process model, the user can select which concepts, relations, and contexts are to be taken into account in the configuration process. For instance, the user might want to express that a gynaecologist and urologist are not present in their organisation and should therefore not play any part in the configuration. The concept graph from Fig. 2 is then annotated as depicted in Fig. 3. By storing this information in the concept graph, we do not return instantiations of the configurable process model which has a resource gynaecologist or urologist.

Next to indicating that certain concept instances are not present, the user also has the option to indicate an element (concept instance/relation/context) is highly relevant in a configuration context and should therefore be present in the configured model. This can be indicated in two ways; `all` and `some`. `All` indicates that an element is present and all elements related to it are all present. For instance, if the organisation would have indicated that the gynaecologist and urologist are present in `all` cases, then every cross in Fig. 3 would have been substituted by a checkmark indicating they are present in `All` cases. `Some` is in between `all` and `none`. For instance, taking Fig. 3 as an example, if the user indicates that the physiotherapist is present in their organisation, then this means *Consult* is set to `some` as it is present in some contexts but not in all.

In the concept graph in Fig. 3, we have crossed out *first* (set it to `none`) although this was not explicitly encoded by the user. Rather, this is a result of the user's action to eliminate the urologist and gynaecologist from the configuration process. By setting *first* to `none`, we make the user aware of this result. The user could still opt to try and set *first* to `some` or even `all`. However, if the user decides to do this, the graph becomes inconsistent. After all, there is no qualified resource available anymore to execute a first consult. In order to compute the transitive effect of choices in the configuration process, we defined rules to transfer the selected answer to answers for other concepts,

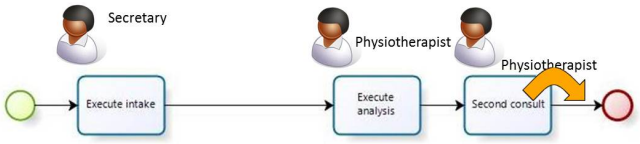

Fig. 4: The model obtained after applying the consistency graph of Fig. 3 to the model in Fig. 1. All elements with a cross have been removed.

relations, and contexts. Applying the consistency graph from Fig. 3 onto the model of Fig. 1 results in the model depicted in Fig. 4.

In the following, we elaborate how we obtain the concept instances from the process model, we provide a formal definition of the consistency graph and rules to transfer an answer, and how we use the consistency graph for configuration.

3.2 Obtaining the concepts in the consistency graph

The consistency graph is obtained on basis of the process model. The resources and data instances can be directly deduced from the process model. The decomposition of the activity labels is accomplished by using the language-based analysis technique from [13]. This technique builds on the insight that activity labels follow regular structures, so-called labelling styles. By automatically recognising the varying labelling styles, it automatically decomposes every activity into the underlying action, business object, and additional fragments. As an example, consider the activity "Notify customer via e-mail". As a result from the label style recognition and the subsequent decomposition, the analysis technique returns the action "notify", the business object "customer", and the additional fragment "via e-mail". Note that this technique can be effectively adapted to languages other than English [14].

3.3 The Consistency Graph

The consistency graph consists of two elements: a concept graph and a set of rules to ensure the consistency of that graph. In the concept graph, we have `dontCare` next to `all`, `some`, and `none`. `DontCare` is used as a default value and it can be used to indicate that the user is not interested in that particular element.

Definition 1 (Concept graph). *A concept graph G is a 5-tuple (V, E, C, QA, R) where:*

- *V is a set of vertices representing the different concept instances,*
- *$E \subseteq V \times V$ is a set of directed edges, denoting the relations*
- *$C : E \rightarrow 2^{V^*}$ are the contexts for each edge, denoting when a particular relation holds,*
- *$QA : V^* \cup E \cup V \rightarrow \{\texttt{All}, \texttt{Some}, \texttt{None}, \texttt{DontCare}\}$, the options selected for the contexts, vertices, and edges,*
- *(V, E) forms a DAG,*

Table 1: The different requirements for transferring the selected answer.

$\forall_{e \in E}(QA(e) = \texttt{all} \Leftrightarrow \forall_{o \in C(e)} QA(o) = \texttt{all})$
$\forall_{e \in E}(QA(e) = \texttt{some} \Leftrightarrow \exists_{o \in C(e)} QA(o) \in \{\texttt{all}, \texttt{some}\} \wedge \exists_{o \in C(e)} QA(o) \neq \texttt{all})$
$\forall_{e \in E}(QA(e) = \texttt{none} \Leftrightarrow \forall_{o \in C(e)} QA(o) = \texttt{none})$
$\forall_{v \in V}(QA(v) = \texttt{all} \Leftrightarrow \forall_{(v,v') \in E} QA\big((v, v')\big) = \texttt{all})$ $\forall_{v \in V}(QA(v) = \texttt{all} \Leftrightarrow \forall_{(v',v) \in E} QA\big((v', v)\big) = \texttt{all})$
$\forall_{v \in V}(QA(v) = \texttt{some} \Leftrightarrow \exists_{(v,v') \in E} QA\big((v, v')\big) \in \{\texttt{all}, \texttt{some}\} \wedge \exists_{(v,v') \in E} QA\big((v, v')\big) \neq \texttt{all})$ $\forall_{v \in V}(QA(v) = \texttt{some} \Leftrightarrow \exists_{(v',v) \in E} QA\big((v', v)\big) \in \{\texttt{all}, \texttt{some}\} \wedge \exists_{(v',v) \in E} QA\big((v', v)\big) \neq \texttt{all})$
$\forall_{v \in V}(QA(v) = \texttt{none} \Leftrightarrow \forall_{(v,v') \in E} QA\big((v, v')\big) = \texttt{none})$ $\forall_{v \in V}(QA(v) = \texttt{none} \Leftrightarrow \forall_{(v',v) \in E} QA\big((v', v)\big) = \texttt{none})$
$\forall_{(v,v') \in E, lo \in C((v,v'))}(QA(lo) = \texttt{all} \Leftrightarrow \forall_{(v',v'') \in E, lo' \in C((v',v''))} lo \sqsubseteq lo' \Rightarrow QA\big(lo'\big) = \texttt{all})$
$\forall_{(v',v) \in E, lo \in C((v',v))}(QA(lo) = \texttt{all} \Rightarrow \exists_{(v'',v') \in E, lo' \in C((v'',v'))} lo' \sqsubseteq lo \Rightarrow QA\big(lo'\big) \in \{\texttt{all}, \texttt{some}\})$
$\forall_{(v,v') \in E, lo \in C((v,v'))}(QA(lo) = \texttt{some} \Leftrightarrow \exists_{(v',v'') \in E, lo' \in C((v',v''))} lo \sqsubseteq lo' \Rightarrow QA\big(lo'\big) \in \{\texttt{all}, \texttt{some}\} \wedge$ $\exists_{(v',v'') \in E, lo' \in C((v',v''))} lo \sqsubseteq lo' \Rightarrow QA\big(lo'\big) \neq \texttt{all})$
$\forall_{(v',v) \in E, lo \in C((v',v))}(QA(lo) = \texttt{some} \Rightarrow \exists_{(v'',v') \in E, lo' \in C((v'',v'))} lo' \sqsubseteq lo \Rightarrow QA\big(lo'\big) = \texttt{some})$
$\forall_{(v,v') \in E, lo \in C((v,v'))}(QA(lo) = \texttt{none} \Leftrightarrow \forall_{(v',v'') \in E, lo' \in C((v',v''))} lo \sqsubseteq lo' \Rightarrow QA\big(lo'\big) = \texttt{none})$ $\forall_{(v',v) \in E, lo \in C((v',v))}(QA(lo) = \texttt{none} \Rightarrow \exists_{(v'',v') \in E, lo' \in C((v'',v'))} lo' \sqsubseteq lo \Rightarrow QA\big(lo'\big) \in \{\texttt{some}, \texttt{none}\})$

– $R \subseteq V$ *are the roots (the most important concept instances).*

Prior to defining the rules for consistency, we first need a notion of subsumption on the contexts.

Definition 2 (Subsumption). *Let $lo, lo' \in V^*$ be two lists of concept instances, then we say lo' subsumes lo, denoted by $lo \sqsubseteq lo'$, if and only if: all concept instances in lo occur in lo' and they occur in the same order.*

In Table 1, the requirements to which the concept graph has to adhere to be a consistency graph are listed. The requirements reflect the intuitive meaning of each of the options a user has (`all`, `some`, `none`, `dontCare`).

The first three lines show the connection between the edges and the contexts on those edges. For instance, if an edge should be present in `some` cases, then there is at least one context which is either present for `all` or `some` but there should also be a context which is not `all`.

The next six lines connect the vertices with the incoming and outgoing edges. For instance, if a vertex is set to `none`, then all the incoming and outgoing edges are set to `none`. The last six lines are connecting the different contexts with each other, e.g., if a particular context is set to `all`, then all subsuming contexts are also set to `all`.

Definition 3 (Consistency graph). *Let $G = (V, E, C, QA, R)$ be a concept graph, we say G is a consistency graph if it adheres to the requirements in Table 1.*

To aid the user in creating the consistency graph, we apply the requirements from Table 1 after each choice of the user. In the application of the requirements, we take the user's choices done so far into account. For instance, if the requirements state that a concept instance has to be set to `all` but the user already set if to `none`, then the affected elements are denoted as inconsistent.

Next to verifying the consistency of the concept graph, we also verify that there is a possible instantiation from the configurable process model. This is a two-step approach,

first based on the consistency graph, we can already configure certain parts, e.g., from the consistency graph, we can deduce which resources are allowed to execute which activities. In the second step, we iterate through possible instantiations of the configurable process model and verify if this instantiation adheres to the consistency graph.

In order to verify whether an instantiation adheres to the consistency graph based on the user's choices (G), we build the consistency graph of the instantiation (G_i). If an element in G is set to `all` or `some`, then we expect that this element is present in G_i. Note that, we know that by applying the requirements the user's choices have propagated and hence we can verify the presence locally. If an element in G is set to `none`, we expect that element is not present in G_i.

4 Implementation

The consistency graph has been implemented as part of *Petra* [15] (Process model based Extensible Toolset for Redesign and Analysis), a ProM plug-in and takes as input a configurable process model. It can be downloaded from `www.processmining.org`. *Petra* is a framework designed for the analysis of the configuration space of a configurable process model. Within *Petra*, different Key Performance Indicators (KPIs) and different tools for computing these KPIs can be used. The work presented here is used to prune the configuration space and thus reducing the computing power needed to analyse the configuration space. However, the ideas presented here transcend the use within *Petra* and can easily be made available to other formalisms, e.g., C-YAWL.

We start with a configurable process model. Afterwards, prior to eliciting the requirements with the plug-in, the relative importance of the different concepts is to be elicited by the user (see Fig. 5). The screenshot shows the concepts present in a particular example. Next to the concepts, examples of instances of these concepts based on the configurable process model are provided to the domain expert. Using the sliders, the user can indicate their relative importance.

Based on the relative importance of the various concepts as indicated by the user, the concept instances are ordered. In the next screen (Fig. 6), the user can indicate which concept instances are to be preserved and whether there is a relationship between concept instances. The user has full freedom to explore the various concept instances and tailor these and their relationship to their preference. In Fig. 6, the user has indicated that *Bureau opname* (intake office) is related to `all`, which means that *Bureau opname* is related to *Afspraak* (appointment) and *Vragenlijst* (Questionnaire).

Next to this, the user can select, on a more fine-grained level, whether there is a relation between concept instances in a particular context (Fig. 7). In Fig. 7, the user has indicated that *multi disciplinair na* (*multi-disciplinary post*) is `all` in the context that the resource instance is a *Verpleegkundige* (*Nurse*) working on the Business Object *overleg* (*deliberation*), and the action *uitvoeren* (*execute*). By setting the use of *multi disciplinair na* to `all`, it can be automatically deduced on the basis of the consistency graph and the introduced rules that the other concept instances in this context have to be set to `some`. However, the user has set the *overleg* to `none`, resulting in an inconsistency as shown by the red parts. Note that, the inconsistency moves from context to edge to concept instances, resulting in signals of inconsistencies on a larger scale. By

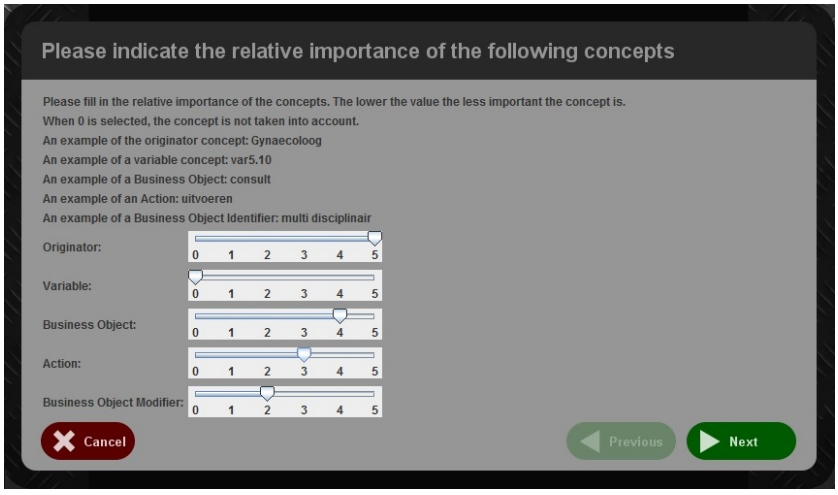

Fig. 5: The GUI presented to the user where they can indicate the relative importance of the various concepts.

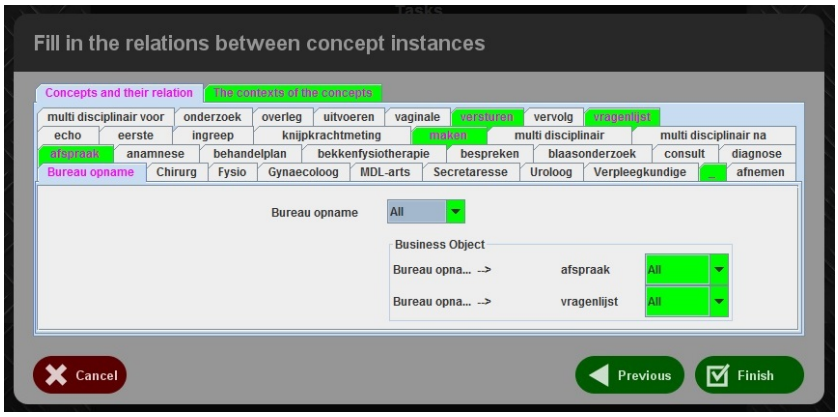

Fig. 6: The GUI presented to the user where they can indicate the relation between the concept instances, in this case the relations between *Bureau opname*, and *Afspraak* and *Vragenlijst*.

showing these inconsistencies, the user is notified of an impossibility. The impossibility can indicate for instance that the user's requirements are not according to certain rules, but it can also indicate that the configurable model is not suitable for their organisation.

After pressing *Finish*, the tool automatically sets configuration points based on the settings by the user. This results in a (configurable) process model which is used in *Petra*. *Petra* checks for each instantiation if it adheres to the consistency graph.

Fig. 7: The GUI presented to the user where they can indicate the relation between the concept instances, with a particular context. The context is build-up similar to the folder structure on a PC, e.g., *multi disciplinair na* is used in the context that the resource instance *Verpleegkundige* is working on Business Object *overleg*, and action *uitvoeren*. In this case, there is an inconsistency between *overleg* and *multi disciplinair na*.

5 Evaluation

For the evaluation of our approach, we cooperated with the consultancy agency iCON healthcare[2]. This agency advises hospitals, clinics, and other medical institutes on the improvement of their healthcare operations. We cooperated specifically with a consultant with deep knowledge on the way Pelvic Floor Examinations are carried out and a coach specialised in process modelling. Pelvic Floor Examinations are typically set up in an outpatient clinical setting, where various fields of medical expertise are brought together such as gynaecology, urology, physiotherapy, and surgery. Within the setting of Pelvic Floor examinations, there is considerable freedom to organise the diagnostic and treatment processes. Therefore, the consultants are seeking ways to guide their clients to a process set-up that best fits the local requirements but exploiting the options present in process set-ups within other hospitals. As a first step towards providing such services, the consultants held structured interviews with stakeholders within various hospitals to obtain an insight into these processes. This knowledge was codified in the form of process models. Using an extension to the techniques presented in [16], we were able to construct a configurable process model from these models.

In the evaluation of the approach we described in this paper, we went through 3 scenarios to expose the consultants to its characteristics. In the first scenario, we showed how the tool could be used to enable/disable a single concept, in casu a particular type of medical expert. In the second scenario, we went one step further by showing how the relations between concepts could be enabled/disabled. In particular, we showed the way

[2] www.iconhc.nl

to specify whether a surgeon would be allocatable to an activity involving a questionnaire. Finally, in the third scenario, we used a configurable process model obtained after merging a number of process models. Using this configurable process model, we were able to configure a specific process model which was applicable to one of the hospitals in their network. The presentation and discussion of the scenarios lasted approximately 80 minutes; the reflection on the approach 40 minutes.

The first outcome of our evaluation concerns the perceived usefulness of the tool. Both consultants looked highly favourable into this aspect. One of them stated that "the business case for this tool is that the information analysis phase can be strongly reduced", hinting at how the number of process set-ups to be evaluated could be highly reduced. As a precondition, it was noted that there should be an upfront investment in a database with configurable process models. The second point of reflection was the usability of the tool. According to one of the consultants "the interface is easy and intuitive". The notion of concepts in particular was well understood, as well as the meaning of the configuration choices. Finally, the third point of discussion related to potential improvements. The suggestions mostly involved the direct context of the toolkit: To improve the way to reach a sufficient amount of high-quality models from the hospital context and to provide support for bringing more detail to a model once it is configured, e.g., the inclusion of work instructions and detailed resource constraints. One of the consultants even stated that: "The more we can fill in, the more valuable the tool becomes".

6 Conclusion

In this work, we showed an approach to automatically generate support to simplify and speed up the configuration of a configurable process model. We zoomed in on the use of general concepts, which can be recognised in most if not all process models, to specify those elements and relations that are relevant to the particular context of the model to be configured. We paid attention to safeguarding the consistency of the various choices an end user can make as well.

One of the main limitations of our approach at this moment is the lack of support for configuration constraints. Contrary to the configuration constraints present in related work, we intend configuration constraints from the end-user and between concept instances. For instance, for a hospital it does not matter which of two concept instances are present as long as at least one is present. In our current implementation, the user is not able to state conditional requirements between concept instances. We plan to add support for this using constraints in the spirit of a declarative modelling language. Next to supporting configuration constraints, we also plan to develop a *constructive* approach for deducing if there is an instantiation adhering to the consistency graph as defined by the user. When working with large amounts of variation points, the verification whether there is an instantiation adhering to the requirements of the end users can become a computational bottleneck.

In conclusion, we hope to contribute with this and future work to the further uptake and application of configurable process models.

Acknowledgements: The authors would like to thank Aukje Houben and Bram de Kort from iCON Healthcare for their cooperation in our case study.

References

1. La Rosa, M., Aalst, W.M.P. van der, Dumas, M., Milani, F.P.: Business process variability modeling : A survey (2013) ACM Computing Surveys.
2. Rosemann, M., Aalst, W.M.P. van der: A Configurable Reference Modelling Language. Information Systems **32**(1) (2007) 1–23
3. Aalst, W.M.P. van der, Lohmann, N., La Rosa, M.: Ensuring correctness during process configuration via partner synthesis. Inf. Syst. **37**(6) (2012) 574–592
4. La Rosa, M., Lux, J., Seidel, S., Dumas, M., Hofstede, A.H.M. ter: Questionnaire-driven Configuration of Reference Process Models. Advanced Information Systems Engineering **4495** (2007) 424–438
5. La Rosa, M., Reijers, H.A., van der Aalst, W.M.P., Dijkman, R.M., Mendling, J., Dumas, M., García-Bañuelos, L.: APROMORE: An advanced process model repository. Expert Systems with Applications **38** (2011) 7029–7040
6. La Rosa, M., Aalst, W. M. P. van der, Dumas, M., Hofstede, A. H. M. ter: Questionnaire-based variability modeling for system configuration. Software and System Modeling **8**(2) (2009) 251–274
7. Hallerbach, A., Bauer, T., Reichert, M.: Capturing Variability in Business Process Models: The Provop Approach. Journal of Software Maintenance and Evolution: Research and Practice **22**(6-7) (November 2010) 519–546
8. Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S.: Feature-oriented domain analysis (foda) feasibility study. technical report cmu/sei-90-tr-21 esd-90-tr-222. Technical report, Software Engineering Institute, Carnegie Mellon University (1990)
9. Schnieders, A., Puhlmann, F.: Variability mechanisms in e-business process families. In Abramowicz, W., Mayr, H.C., eds.: BIS. Volume 85 of LNI., GI (2006) 583–601
10. Acher, M., Collet, P., Lahire, P., France, R.B.: Managing variability in workflow with feature model composition operators. In Baudry, B., Wohlstadter, E., eds.: Software Composition. Volume 6144 of Lecture Notes in Computer Science., Springer (2010) 17–33
11. Czarnecki, K., Antkiewicz, M.: Mapping features to models: A template approach based on superimposed variants. In Glück, R., Lowry, M.R., eds.: GPCE. Volume 3676 of Lecture Notes in Computer Science., Springer (2005) 422–437
12. Awad, A., Sakr, S., Kunze, M., Weske, M.: Design by selection: A reuse-based approach for business process modeling. In Jeusfeld, M.A., Delcambre, L.M.L., Ling, T.W., eds.: ER. Volume 6998 of Lecture Notes in Computer Science., Springer (2011) 332–345
13. Leopold, H., Smirnov, S., Mendling, J.: On the refactoring of activity labels in business process models. Information Systems **37**(5) (2012) 443–459
14. Leopold, H., Eid-Sabbagh, R.H., Mendling, J., Azevedo, L.G., Baião, F.A.: Detection of naming convention violations in process models for different languages. Decision Support Systems **56** (2013) 310–325
15. Schunselaar, D.M.M., Verbeek, H.M.W., van der Aalst, W.M.P., Reijers, H.A.: Petra: A tool for analysing a process family. In Moldt, D., Rölke, H., eds.: International Workshop on Petri Nets and Software Engineering (PNSE'14). Number 1160 in CEUR Workshop Proceedings, Aachen, CEUR-WS.org (2014) 269–288 http://ceur-ws.org/Vol-1160/.
16. Schunselaar, D.M.M., Verbeek, H.M.W., Aalst, W.M.P. van der, Reijers, H.A.: Creating Sound and Reversible Configurable Process Models Using CoSeNets. In Abramowicz, W., Kriksciuniene, D., Sakalauskas, V., eds.: BIS. Volume 117 of Lecture Notes in Business Information Processing., Springer (2012) 24–35