

QUT Digital Repository:  
<http://eprints.qut.edu.au/>



Gottschalk, Florian and Wagemakers, Teun A.C. and Jansen-Vullers, Monique H. and van der Aalst, Wil M.P. and La Rosa, Marcello (2009) ***Configurable process models : experiences from a municipality case study***. In: 21th International Conference on Advanced Information Systems Engineering, 10-12 June 2009, Amsterdam, the Netherlands

© Copyright 2009 Springer

# Configurable Process Models: Experiences from a Municipality Case Study

Florian Gottschalk<sup>1</sup>, Teun A.C. Wagemakers<sup>1</sup>, Monique H. Jansen-Vullers<sup>1</sup>,  
Wil M.P. van der Aalst<sup>1,2</sup>, Marcello La Rosa<sup>2</sup>

<sup>1</sup> Eindhoven University of Technology, The Netherlands.  
{f.gottschalk, m.h.jansen-vullers, w.m.p.v.d.aalst}@tue.nl,  
teun.wagemakers@pallas-athena.com

<sup>2</sup> Queensland University of Technology, Brisbane, Australia.  
m.larosa@qut.edu.au

**Abstract.** Configurable process models integrate different variants of a business process into a single model. Through configuration users of such models can then combine the variants to derive a process model optimally fitting their individual needs. While techniques for such models were suggested in previous research, this paper presents a case study in which these techniques were extensively tested on a real-world scenario. We gathered information from four Dutch municipalities on registration processes executed on a daily basis. For each process we identified variations among municipalities and integrated them into a single, configurable process model, which can be executed in the YAWL workflow environment. We then evaluated the approach through interviews with organizations that support municipalities in organizing and executing their processes. The paper reports on both the feedback of the interviewed partners and our own observations during the model creation.

**Keywords:** Business Process Models, Configuration, YAWL, Registration Process, Questionnaires, Case Study

## 1 Introduction

Many processes in public administration are driven by legislation, e.g. the process of renewing a drivers licence is constrained by law. Therefore, the processes executed in the administration of municipalities are extensively regulated. Although legislation is establishing the important steps, some freedom is left regarding the concrete implementation of such processes. Hence, municipalities can still adapt their processes to local needs and preferences, e.g. depending on the size of the municipality, or on the services provided along with these processes.

*Configurable process models* were developed to align the variation options of widely standardized processes with small variations like the ones executed by municipalities. Further, they enable software providers to support the execution of these process variations through their service-oriented software. For this, configurable process models integrate several process variants into a single process model. To adapt this integrated model to individual needs, a configurable

process model can be configured allowing an organization to disable all the unnecessary process parts. In this way, an organization can derive a process model that fits its individual needs without actually performing any process modeling activities. If the configurable model is defined as a workflow specification, the resulting models can then be executed through a workflow engine [5].

As disabling unnecessary process parts of the workflow definitions still requires insights into the process modeling notation, this can be steered through a questionnaire with domain-related questions. In this way, domain experts without such skills can configure and derive executable workflow specifications fitting their particular needs [7,8].

To evaluate the concept of configurable process models in practice, we performed a case study with four municipalities of different sizes in the Netherlands. Using the YAWL workflow notation and its configuration extension [5], we created configurable process models for four registration processes that are executed within each of the selected municipalities on a daily basis. The four configurable models incorporate all the variations in the execution of these processes among the municipalities as well as the suggestions of a reference model for these processes, i.e.  $5 \times 4 = 20$  processes were used as input for creating these models. Afterwards, we evaluated the practical usefulness of the resulting models through focus group interviews with software providers, consultants, and the municipalities themselves. During these interviews the stakeholders could derive individual process models for these four processes. To test whether the resulting models conform to what they intended by answering the questionnaires, they could execute the resulting process definitions using the YAWL system<sup>3</sup>.

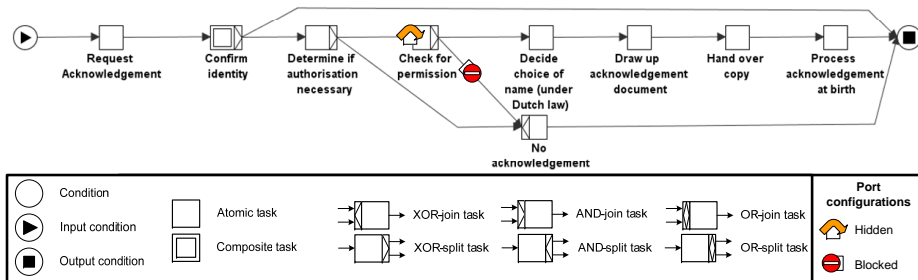
The results of the case study are presented in this paper which is structured as follows. Section 2 will provide some background information on configurable process models and configurable YAWL, as well as on how these models can be configured through questionnaires. Next, Section 3 first depicts how we created the configurable models for the municipality processes before summarizing our practical experiences during the model creation phase. Section 4 provides details on the interviews we performed with the stakeholders, as well as the conclusions from these interviews. The paper ends with a brief overview on similar case studies in Section 5, and draws overall conclusions in Section 6.

## 2 Background

Configurable extensions have been suggested for several process modeling notations. In this case study we used configurable YAWL, i.e. an extension of the YAWL notation aiming at configuration [5,7]. Using a workflow notation for implementing the configurable model comes with the advantage that we can execute the configured process models in the corresponding workflow engine and thus test and demonstrate the practical feasibility of the suggested methodology even to users unfamiliar with process modeling. In addition, the steering

---

<sup>3</sup> <http://www.yawlfoundation.org>



**Fig. 1.** A YAWL process model for acknowledging an unborn child. The input port of *check permission* is configured as hidden and one output port is blocked.

of the configuration through a questionnaire aims at providing such domain experts with the ability to derive configurations for process models. In this way, the adaptation of a business process model for later execution can be achieved without extensive modeling skills.

## 2.1 Configurable YAWL

YAWL is a process modeling notation and workflow environment based on Petri nets but extended with powerful features for cancelation, OR-joins, etc. It has been developed with the aim to provide a notation with formal semantics that supports all desired workflow patterns [1]. The YAWL system is open-source and supports the execution and work distribution of workflows depicted in such models even in production environments. Thus, although originally developed as a proof of concept, the YAWL system can be used for practical applications.

Figure 1 depicts a simple YAWL model for the process executed by municipalities when a man registers that he will become the father of a not-yet-born child although he is not married to the mother.<sup>4</sup> In this model tasks are depicted as rectangles while circles represent conditions like the initial and final condition in this example. Conditions mark the states between tasks but can be omitted for simplicity (like in the example). Composite tasks enable the hierarchical specification of sub-processes while split and join types of tasks allow the specification of how the process should proceed in case a task splits or joins the process’s control flow. For this, YAWL distinguishes an XOR-split (as in the example in Figure 1) allowing the triggering of only one of the subsequent paths, an AND-split requiring the triggering of all subsequent paths, and an OR-split requiring the triggering of at least one subsequent path but allowing also for path combinations. Similarly, a task with an XOR-join can be executed as soon as one of its incoming paths is triggered, an AND-join requires that all incoming arcs are triggered, and a task with an OR-join allows for the execution of the task as soon as no further incoming paths can potentially be triggered at any future point in time (see [1] for further details).

<sup>4</sup> Note that this process is specific for the Netherlands and constrained by Dutch law.

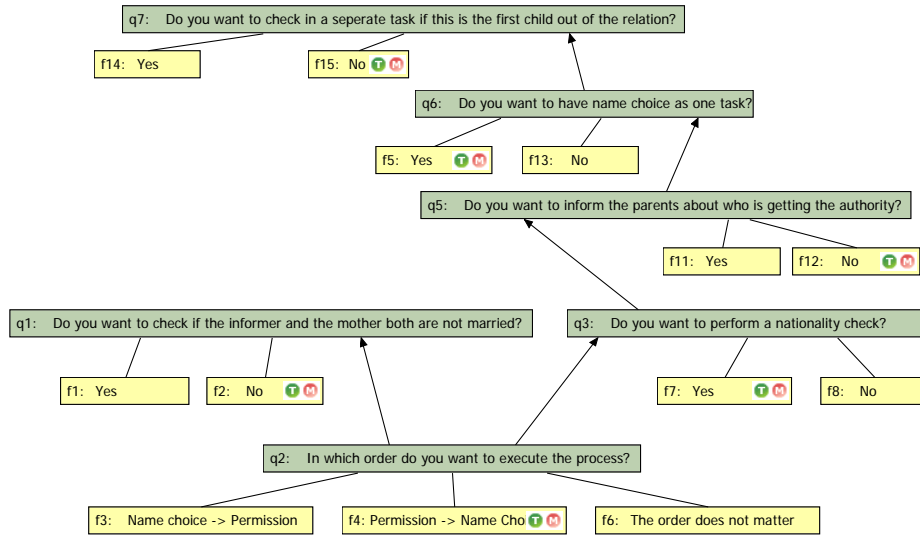
This routing behavior can be restricted by *process configuration*. For this purpose, *input ports* are assigned to each task depicting how the task can be triggered and *output ports* are assigned to depict which paths can be triggered after the completion of the task. A task with an XOR-join can be triggered via each of its incoming paths. Thus, it has a dedicated input port for each of these paths. Tasks with AND-joins and OR-joins can only be executed if all paths (that can potentially be triggered) are triggered, i.e. there is only one way these tasks can be triggered and thus only one input port. A task with an XOR-split has an output port for each subsequent path as each of these paths can be triggered individually while a task with an AND-split has only one output port as all subsequent tasks must always be triggered. A task with an OR-split can trigger a subset of the outgoing paths, i.e. in this case a separate output port exists for each of these combinations.

The process flow can be restricted at these ports. A *blocked* port prevents the process flows through it, i.e. a blocked input port prevents the triggering of the task through the port while a blocked output port prevents that the corresponding output paths can be triggered. In the model in Figure 1, we blocked the output port from *Check for permission* to *No acknowledgement*. Thus, the task *Check for permission* must always be followed by the task *Decide choice of name (under Dutch law)* as the path to the task *No acknowledgement* can no longer be triggered. Input ports can not only be blocked but also be configured as *hidden*. Similarly, the subsequent task can then not be triggered through this port anymore. However, in this case the process flow is not completely blocked, but only the execution of the corresponding task is skipped. The process execution continues afterwards. In Figure 1 the input port of the task *Check for permission* is hidden. Thus, the execution of this task is skipped which also explains why we blocked one of the task’s output ports: the configuration results in skipping the check. Hence, it can no longer fail and the process must continue normally. Further details on configurable YAWL can be found in [5].

As we can observe from this example, the configurations of ports are often not independent from each other but rather driven by more general domain-related aspects. It is therefore suggested to steer the configuration through questions on these domain-related aspects as is discussed next.

## 2.2 Steering Process Configuration through Questionnaires

In principle, the variability of the domain can be depicted independently of the process flow by means of a set of domain *facts* that form the space of possible answers to a set of *questions*. A domain fact is a boolean variable representing a feature of the domain, e.g. *Perform a check of the nationality*, that can be enabled or disabled. Questions can group domain facts according to their content, so that all the facts of the same question can be set at once by answering the question. Interdependencies between questions can specify a partial order in which the questions should be posed to the user. Figure 2 depicts such a questionnaire model for the various options in the process of acknowledging an unborn child.



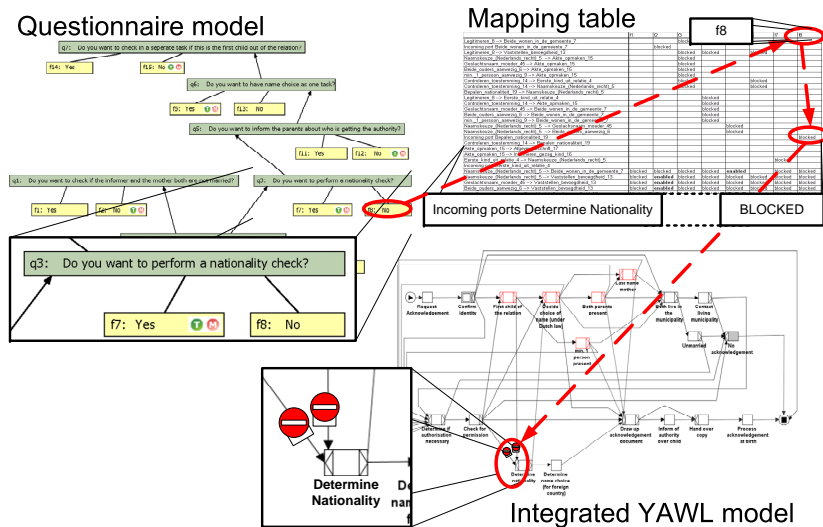
**Fig. 2.** The questionnaire model addressing the various options in performing the process of acknowledging an unborn child.

Each configuration of a port in the process model can then depend on such domain facts. For example, the input ports of the task in which the nationality check is performed must be set to allowed when the corresponding domain fact is set to *true* while it must be hidden or blocked when the domain fact is set to *false*. Such a port configuration might also be dependent on a combination of answers, i.e. domain facts. For this, the facts can be combined in propositional logic expressions that capture their interplay. It is then important to make sure that a single port will never have two configuration values at the same time (e.g. blocked and hidden) which can be achieved through corresponding, additional constraints. These can then also imply that certain answers given in the questionnaire automatically define the answers to further questions.

Figure 3 depicts and summarizes the steering of process configuration through questionnaires. Further details on the approach can be found in [7,8].

### 3 Creating Configurable Process Models

In the first project phase we created configurable process models for four registration processes. These processes are executed on a daily basis in each of the municipalities. In this way, we evaluated the feasibility of configurable process models in public administration. The steps taken during the creation of the models are explained in the first part of this section. Afterwards, we summarize the challenges we were confronted with during the creation of such models and how we addressed them.



**Fig. 3.** The setting of a domain fact through answering a question leads to the selection of a particular port configuration via the so-called mapping table.

### 3.1 Building the Models

In total we created configurable process models for the following four registration processes:

- *Acknowledging an unborn child:* This process is executed when a man wants to register that he will be the father of a child still to be born while he is not married to his pregnant partner.
- *Registering a newborn:* This process is executed by the municipality to register a newborn child and handing out a birth certificate.
- *Marriage:* This process includes all steps necessary before a couple can get married in a Dutch municipality.
- *Decease:* This process is executed when a person deceases to provide the relatives with the documentation necessary to bury the deceased.

Reference process models for these processes are available from the Nederlandse Vereniging Voor Burgerzaken (NVVB<sup>5</sup>), i.e. the Dutch association for services to the public. These models describe a single “best-practice” version of how the particular process should be executed. While these models are available in several notations, we used the notation of the business process modeling tool Protos. Protos is very popular among Dutch municipalities: these reference models are used by over 100 Dutch municipalities (mainly for auditing purposes).

To detect the variations of the processes in the daily practice we visited four municipalities in the Netherlands. We selected the municipalities such that they

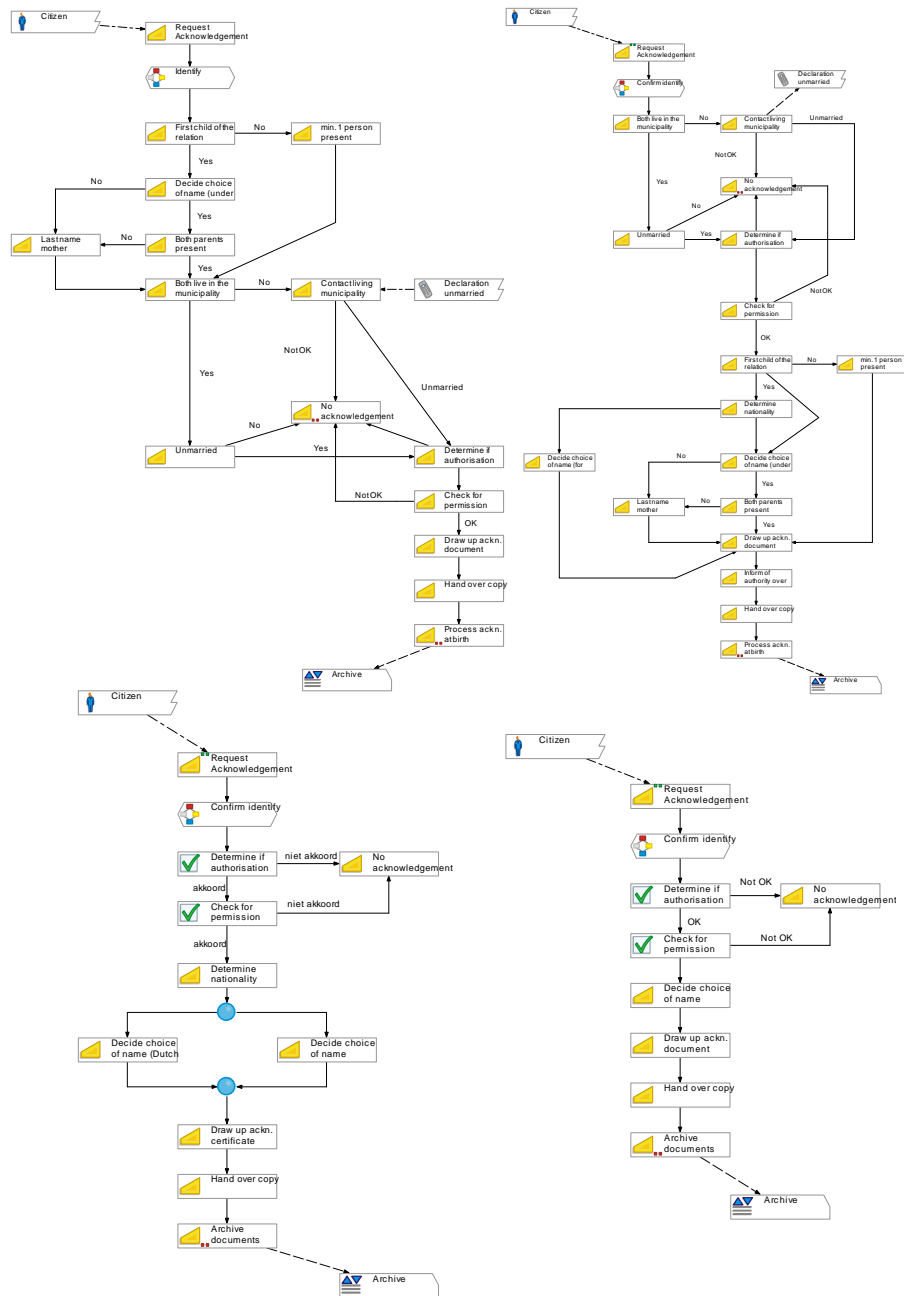
<sup>5</sup> <http://www.nvvb.nl>

vary in the size of their population (between 26.000 and 201.000 inhabitants) and such that they use software from different providers to support the process execution. Without confronting the process owners of the selected municipalities with the reference models, we asked them to explain how they execute the various processes. We then used again Protos to create a separate process model for each process in each municipality. During this phase, some of the municipalities provided us with process models which they created to document their processes. In these cases, we based our models on the models which were provided by them. We only made modifications where it became clear from our visits to the particular municipality that a process model did not reflect what was actually happening. To make sure that we correctly depicted the processes, we asked the process owners at the end of this phase to validate the models. Figure 4 shows the four Protos models we derived from the four municipalities for the process of acknowledging an unborn child. While the control flow of these four processes is similar, the number of steps taken as well as the concrete order of executing tasks varies among municipalities.

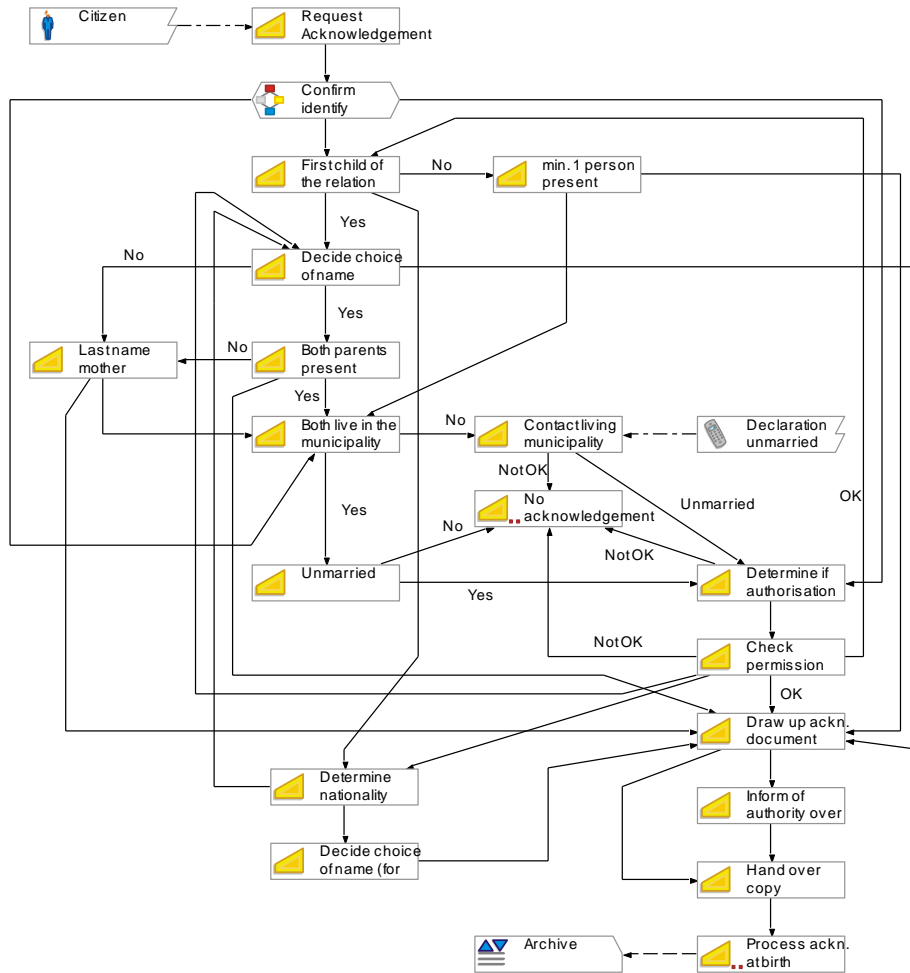
For each of the four selected business processes we then identified all the differences among the five process variants (the reference model plus the models of the four municipalities) by comparing them with each other. Based on this information, we created for each business process a single Protos model that incorporates all the variations from the five input models as ordinary runtime choices. The integrated model derived from the reference model and the four process variants for acknowledging an unborn child shown in Figure 4 is shown in Figure 5. Note that out of the four business processes we analyzed in this case study, the process of acknowledging an unborn child is the simplest, i.e. the three other combined process models include both more tasks and more arcs.

To be able to configure and execute these models, we then switched to a workflow environment that supports both the configuration and execution of Protos models. In particular we chose YAWL here as our ideas on process configuration [5,8] are implemented in this environment. The translation from Protos to YAWL was done manually as we not only translated the pure control flow from the Protos models, but also implemented the data upon which the process relies and which is only available in a descriptive way in the Protos models. In this way, we created the basis to route cases through the process model according to the data collected during the process execution. That means, the resulting YAWL models are fully executable in the YAWL workflow engine. The YAWL model for the acknowledgement of an unborn child is shown in Figure 6a.

The resulting four YAWL models integrating all the variations of the processes were of course far too complex to be used and configured by the stakeholders of the municipalities. Thus, we also created a questionnaire for each of the four business processes as explained in Section 2.2. In the questionnaires we addressed each variation possibility at a particular stage of the process by at least one question. The questionnaire model for the process of acknowledging an unborn child was already shown in Figure 2. The answers to the questions were then mapped to allowing, hiding, or blocking the process flow through various

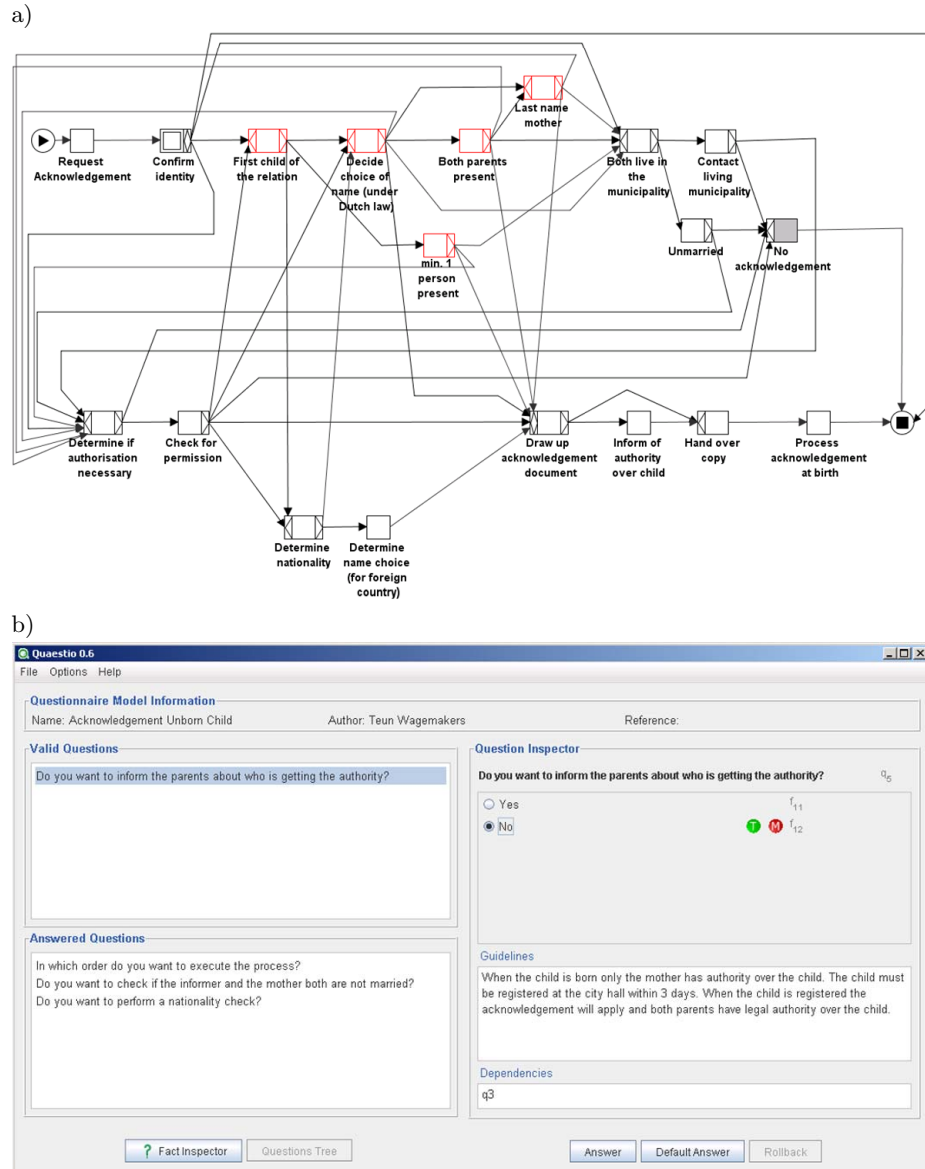


**Fig. 4.** The different process variants of how municipalities perform the acknowledgement of an unborn child.



**Fig. 5.** All variants for acknowledging an unborn child integrated into a single Protos model.

ports. In this way, the configuration of the process model integrating the five process variants can be done by the stakeholders through simply answering the questionnaire (see Figure 6b). There is no need for the stakeholders to understand the implications of blocking or hiding certain ports. In fact, they do not even need to be confronted with the integrated process models as the configuration decisions resulting from the answers to the questionnaire can be applied automatically to this model. For example, the process model we showed in Figure 1 is in fact derived from the integrated model in Figure 6a using the answers given by one of the involved municipalities. In our approach, the stakeholders only receive these individually relevant models which are also directly executable



**Fig. 6.** The model from Figure 5 translated into YAWL (a) and the corresponding questionnaire in the Questio tool (b) which allows user to configure the model through the given answers.

using the YAWL workflow engine. Then, the users of the model will be filling out forms generated based on the information of the configured process model. Thus, these users do not see but benefit from the configured model.

## 3.2 Observations

First of all, it should be noted that for all four business processes we were able to create integrated process models and questionnaires that allow users to derive an individual model. For each process and each municipality we were able to generate a model equivalent to the original Protos model by answering the questionnaire and applying the resulting configuration to the YAWL model. This illustrates that it is possible to integrate several process variants such that all desired individual variants can be derived from it.

Still, we had to master several challenges during the creation of the configurable models. While deriving the individual process variants was straight forward, the first challenges arose when integrating the different variants into a single process model as matching identical tasks among the variants was often only possible after comparing the exact task descriptions. Moreover, during the manual compilation of the integrated model some paths, i.e. process flows, of the individual models were easily overlooked, and thus not incorporated into the integrated model. Only by carefully “re-playing” the processes of the individual models in the combined models these forgotten arcs were discovered.

Due to the extensive support of control flow patterns, translating the control flow from the Protos models to YAWL models was easy. Tricky was however the implementation of the data perspective for determining the precise runtime routing of cases through the integrated process model. This was especially the case when a choice between various options was introduced in the integrated model while in fact there is no such run-time decision in any of the municipalities. The variation is thus a pure configuration decision based on the difference between the municipalities. For example, this applies for the task *Confirm identity* in Figure 6 which uses an OR-split to branch into four outgoing paths. The decision, which combination of paths should be triggered after the completion, is partly a run-time decision and partly a configuration decision. During run-time it is decided if the identification was successful or not. If not, the process completes immediately. However, the decision which combinations of the remaining three arcs are triggered in case the identification was successful is already a configuration decision (it might be desired to transform this into a run-time decision, but this was not the case in any of the involved municipalities). A correct definition of the process flow details in such situations requires the implementation of a “default” decision as well as a very good anticipation of the implications when this default decision has to change due to a configuration decision.

Questions in the questionnaire abstract from the control flow of the process and usually address larger process parts. Thus, the interdependencies between the answers that can be given in the questionnaire are not always obvious or immediately derivable from the process’s control flow. Hence, ordering of questions and the definition of constraints between the answers turned out to be challenging and required a good anticipation of the desired impact of the configuration decisions which becomes more difficult the more complex the model is.

This phase was mainly performed by one core project member and took, including his familiarization with the used techniques, approximately six months.

## 4 Evaluation of the Approach

To get insights into the practical applicability of the models we derived, we performed an additional analysis using three approximately two-hour-long focus group interviews with one to three employees of the following three organizations:

- Pallas Athena as the supplier of Protos which is actively used by over 250 of the in total 441 Dutch municipalities,
- PinkRocade Local Government who provides a software to execute municipality processes used by more than 50% of the Dutch municipalities, and
- a world-wide operating consultancy firm who adapts their own reference process models during process implementations for their clients.

All interview partners were first given a presentation on the techniques we used during this project as well as on details of how we created the four configurable models and their questionnaires. Afterwards, the interview partners had the opportunity to derive their own executable process models through answering the questionnaires. The models resulting from the answers given in the questionnaire during the interview were immediately presented to them. Not all the interview partners were domain experts for the given processes. Thus, it was possible for them to ask questions on the implications of the various possible configuration decisions in the questionnaire.

Subsequently, we triggered a discussion with the interview partners focussing on potential practical needs for adaptable process models, on the feasibility of creating such configurable models in real-life environments, and on the practical usefulness of applying such configurable models. Key results from these interviews are summarized in Figure 7. In general, we can summarize that all interview partners immediately saw a potential value of the technique of configurable process models for past or current projects. The steering of the actual process configuration is seen as a useful tool to assist end users, but even without this support, direct process configuration might prove to be beneficial in various projects where process adaptation is necessary. The main concerns raised by the interview partners were the efforts necessary to create questionnaires and establishing the links between the potential answers and all the ports as well as the incorporation of resource assignments to tasks during the configuration process.

## 5 Related work

The case study reported on in this paper uses our earlier work on process configuration [5,7,8]. Similar techniques for adapting process models were suggested by Becker et al. [3,4]. Their approach links adaptation parameters and their possible values to model elements to indicate which sections of the model are relevant or not to a specific application scenario. Thus, a user can configure a process model by setting parameters, i.e. the process model does not need to be consulted. Compared to the approach used here, the approach of Becker et al. is applied to Event-driven Process Chains instead of YAWL, i.e. to a notation

Interview partner	Potential applications and advantages (+) as well as concerns (-)
Pallas Athena	(+) Configurable Process models would have been useful for the development of a “one point of contact” workflow product for municipalities developed based on a new law that requires municipalities to re-structure the customer interaction of their business processes (+) Potential applications in highly regulated, publicly documented and accessible, or non-core business processes like HR processes. (-) The integrated model must be complete. Is this possible and how can this information be derived from existing processes?
PinkRocade Local Government	(+) Questionnaire answers can be linked to other configurable elements, like the configuration of software screens and windows as well as data fields. (+) Configuration through questionnaires enables software providers to create applications that prevent that users can fail during the process configuration. (+) A user sees in the questionnaire the configuration freedom she has rather than the limitations the configuration is subject to. (+) Clients often ask for software adaptation and modifications for a better support of their desired business processes which is currently expensive due to the need for external consultants. Currently, this often results into workarounds. (-) A configuration of the resources that are involved in a process is not possible.
Consultancy Firm	(+) Best-practice reference models are often not sufficient: there is no single best-practice. (+) It would have been useful in a world-wide role-out of new business processes where it was a headquarter policy that 80 % of the processes needed to remain conform to the global process while it was allowed to deviate by 20 % to make the process compliant to local regulations. (+) In some industries production processes are so standardized that the technique might here even be applicable to core processes. (-) The creation of such models seems to require big efforts, sponsoring for this might be difficult to find. (-) The identification of variations between processes is difficult, i.e. tools are necessary for this.

**Fig. 7.** The main comments of the interviewed stakeholders

that is mostly used for process visualization (like the Protos models we created) and not for the enactment of these processes. Moreover, it lacks steering of the parameter setting through an interactive questionnaire.

The use of questionnaires to guide the configuration of process models is inspired by similar configuration processes for software applications. For example, the CML2 language, designed to capture configuration processes for the Linux kernel, guides the configuration process through a structured set of questions that lead to a given symbol being given a value [10]. Also, in CML2 the validity of these values can be ensured by constraints. More generally, variability of

large software systems has been studied in the field of Software Product Line Engineering (SPLE) [9].

Algermissen et al. performed a case study with municipalities to identify best-practice in public administration [2]. Similar to our approach, they initially visited a number of municipalities to observe and depict their business processes. Different from our approach, they do not focus on providing a model with various configuration options, but rather aim at deriving a single, “ideal” process model from these variants. Thus, their approach is similar to the one taken by the NVVB, whose best-practice recommendation we incorporated in our models.

Karow et al. provide guidelines specifically for the construction of reference models in public administration [6]. While our goal here was to test the feasibility and identify the opportunities of using configurable process models in a reference modeling context, we would need to address such guidelines more rigorous if we want to extend our work to providing a complete reference model in the future.

Best-practice reference models have been investigated in several other case studies. For example, Thomas et al. developed a reference model for event management [13], and Scheer designed a reference model for industrial enterprises [11]. A case study on developing a business process reference model for the screen business was performed by Seidel et al. [12]. Also template repositories as provided by vendors of BPM solutions like the ones of SAP and IBM can be considered as such best-practice reference models.

## 6 Conclusions

In this case study, we developed configurable process models for four business processes of municipalities based on information from four different municipalities and a corresponding reference model. Afterwards, we performed expert interviews with various stakeholders about the potential use of these models and the underlying techniques.

During the case study, the suggested techniques proved to be suitable for the intended purposes: we achieved our goal to be able to derive all the initial, individual models of the various municipalities as well as further model variants from the integrated models by answering simple questionnaires. Despite that, the creation of the configurable models required significant efforts, modeling experience, and domain knowledge. Thus, the simplified adaptation of process models is at the expense of a complex creation of the configurable model.

It was obvious during the case study that many issues that arose during the model creation could be improved or even avoided by further tool support, e.g. ensuring consistent identifiers or automatically identifying and integrating process variations. Thus, all interview partners were also interested in techniques that can help here. In general, they all saw potential value for themselves in the technique, which they stressed by mentioning current or past projects where configurable process models could have provided additional benefits. But the interviewees also made clear that process configuration should not be restricted to the control flow perspective of business processes, but should also be inte-

grated with the resource and data perspectives to provide a strong and universal configuration tool.

**Acknowledgements.** We would like to thank the NVVB, Pallas Athena, and PinkRoccade Local Government as well as the municipalities, consultants, and software developers involved in this project for their input and feedback.

## References

1. W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245–275, 2005.
2. L. Algermissen, P. Delfmann, and B. Niehaves. Experiences in Process-oriented Reorganisation through Reference Modelling in Public Administrations — The Case Study Regio@KomM. In *Proceedings of the 13th European Conference on Information Systems (ECIS)*, Regensburg, 2005.
3. J. Becker, P. Delfmann, A. Dreiling, R. Knackstedt, and D. Kuroпка. Configurative Process Modeling – Outlining an Approach to increased Business Process Model Usability. In *Proceedings of the 15th IRMA International Conference*, New Orleans, 2004. Gabler.
4. J. Becker, P. Delfmann, and R. Knackstedt. Adaptive Reference Modelling: Integrating Configurative and Generic Adaptation Techniques for Information Models. In J. Becker and P. Delfmann, editors, *Reference Modeling. Efficient Information Systems Design Through Reuse of Information Models*, pages 27–58. Springer, 2007.
5. F. Gottschalk, W.M.P. van der Aalst, M.H. Jansen-Vullers, and M. La Rosa. Configurable Workflow Models. *International Journal of Cooperative Information Systems (IJCIS)*, 17(2):177–221, June 2008.
6. M. Karow, D. Pfeiffer, and M. Räckers. Empirical-Based Construction of Reference Models in Public Administrations. In *Proceedings of the Multikonferenz Wirtschaftsinformatik 2008. Referenzmodellierung*, pages 1613–1624, 2008.
7. M. La Rosa, F. Gottschalk, M. Dumas, and W.M.P. van der Aalst. Linking Domain Models and Process Models for Reference Model Configuration. In *Proceedings of the BPM 2007 Workshops*, volume 4928 of *Lecture Notes in Computer Science*, pages 417–430, 2008.
8. M. La Rosa, W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede. Questionnaire-based Variability Modeling for System Configuration. *Software and Systems Modeling*, 2008. (forthcoming).
9. K. Pohl, G. Böckle, and F. van der Linden. *Software Product-line Engineering – Foundations, Principles and Techniques*. Springer, Berlin, 2005.
10. E. S. Raymond. The CML2 Language, 2000. <http://catb.org/esr/cml2/cml2-paper.html>.
11. A.-W. Scheer. *Business Process Engineering, Reference Models for Industrial Enterprises*. Springer-Verlag, Berlin, 1994.
12. S. Seidel, M. Rosemann, A.H.M. ter Hofstede, and L. Bradford. Developing a Business Process Reference Model for the Screen Business - A Design Science Research Case Study. In *Proceedings of the 17th Australasian Conference on Information Systems (ACIS 2006)*, Adelaide, Australia, 2006.
13. O. Thomas, B. Hermes, and P. Loos. Towards a Reference Process Model for Event Management. In *Business Process Management Workshops*, volume 4928 of *Lecture Notes in Computer Science*, pages 443–454, 2008.