Creating Sound and Reversible Configurable Process Models Using CoSeNets*

Dennis M.M. Schunselaar, Eric Verbeek,
Wil M.P. van der Aalst, and Hajo A. Reijers

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

Abstract. All Dutch municipalities offer the same range of services, and the processes delivering these services are quite similar. Therefore, these municipalities can benefit from configurable process models. This requires the merging of existing process variants into configurable models. Unfortunately, existing merging techniques (1) allow for configurable process models which can be instantiated to unsound process models, and (2) are not always reversible, which means that not all original models can be obtained by instantiation of the configurable process model. In this paper, we propose to capture the control-flow of a process by a CoSeNet: a configurable, tree-like representation of the process model, which is sound by construction, and we describe how to merge two CoSeNets into another CoSeNet such that the merge is reversible. Initial experiments show that this approach does not influence complexity significantly, i.e. it results in similar complexities for the configurable process model compared to existing techniques, while it guarantees soundness and reversibility.

1 Introduction

Within the CoSeLoG project, we have 10 municipalities involved offering essentially the same set of services. The process models supporting these services are very similar, due to legislation and standardisation, but are different, due to *couleur locale* and local decision making. The goal of the project is to support the different process models via configurable process models.

Configurable process models are process models with configuration options. The user has the possibility to configure the configurable process model by making configuration choices for options. These configurations are used to deduce the process models from the configurable process model (instantiation), by taking the different choices for the configuration options into account. Obtaining a configurable process model can be done via the merger of process models. Merging a set of process models should be such that the behaviour of a configurable process model is (an over-approximation of) the union of allowed behaviour from the different process models.

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

W. Abramowicz et al. (Eds.): BIS 2012, LNBIP 117, pp. 24–35, 2012.

We require that every instantiation of the configurable process model yields a sound process model [1]. Furthermore, to support the different municipalities, we want that the configurable process models are reversible [2], i.e. the models used for obtaining the configurable model should be instantiations of the configurable model. Both requirements should not impact the complexity of the resulting configurable process model significantly in comparison to the state-of-the-art. Applying existing techniques to obtain configurable process models from the CoSeLoG process models resulted in configurable process models which can be instantiated to unsound process models, and some techniques are not reversible.

To counter the aforementioned problems with existing techniques, and adhering to the requirements, we propose to capture the process models in tree-like representations of block-structured process models, which are sound by construction (see [3] for a comparison between block-structured and graph-structured process models). Since there is a straightforward transformation from this tree-like representation to, for instance, Petri nets, we can use the classical notion of soundness.

The configurable variant of these process models is captured in CoSeNets, a tree-like representation of configurable process models. Instantiating a CoSeNet always yields sound process models. Furthermore, the merger of two CoSeNets is always reversible. In order to show that the complexity is not significantly influenced, an empirical comparison is presented between our approach and existing techniques. This empirical evaluation shows that the complexity is comparable to, or lower than existing techniques, using a subset of the processes used in [4].

This paper is structured as follows: Sect. 2 lists the relevant related work. In Sect. 3, the CoSeNets are explained. We explain the merger of CoSeNets in Sect. 4. In Sect. 5, we show a comparison between our approach and existing approaches. Finally, Sect. 6 contains the conclusions and future work. For an extended version of this paper with more technical details, we refer the reader to [5].

2 Related Work

A number of configurable process modelling languages has been proposed. These modelling languages can be subdivided into two categories, i.e. *imperative* and *declarative*. Declarative languages constrain the allowed behaviour, i.e. everything is allowed unless stated otherwise. An example of a configurable declarative process modelling language is *Configurable Declare* [6]. Imperative languages are the opposite of declarative languages; imperative languages specify the allowed behaviour, i.e. nothing is allowed unless stated otherwise. *C-SAP WebFlow*, *C-BPEL*, *C-YAWL* [7], and *C-EPC* [8] are examples of imperative configurable process modelling languages. These configurable modelling languages do not always have to yield sound process models when being instantiated [9,10].

A number of merging techniques have been defined in literature. Gottschalk [7] elaborates on the merger of process models into a single configurable process

Table 1. Comparing the different merging techniques, note that the soundness property of Gottschalk [7] is after some postprocessing

Approach	Gottschalk [7]	Li et al. [11]	Mendling et al. [12]	La Rosa et al. [2]	Sun et al. [14]
Soundness	✓	✗	✗	✗	✗
Reversibility	✓	✗	✗	✓	✗

model (e.g. EPCs). All requirements are met, however, one has to perform a postprocessing step to transform the process model into a sound process model. Furthermore, Gottschalk allows for all possible different instantiations of the configurable process model, i.e. certain parts can be blocked even if it was not observed in any of the input models. Although this does not limit the reversibility, it is noteworthy from an application point of view since it will become harder in this way to identify which parts are commonalities and which parts are variabilities.

Li et al. [11] present an approach for creating a new reference model based on models mined from a log. These models represent the different variations of a reference model. Li et al. only consider "AND" and "XOR" operators. Furthermore, they seek to minimise a distance measure between the reference model and the mined models, where distance is defined as the number of insertions, deletions, and moving activities within the process model. By allowing the insertion of activities, the reference model cannot be configured to obtain the input models. Note that in our approach it is not allowed to insert activities.

In the paper by Mendling et al. [12], an approach is presented to merge the different views on a process model. This approach does not yield configurable models, i.e. the output is an EPC and not a C-EPC. Furthermore, soundness is not guaranteed by this particular approach.

La Rosa et al. [2] present an approach for merging a set of process models into a single configurable process model. With the use of AProMoRe [13] they are even able to merge different formalisms into a single configurable process model. Although this approach allows for the deduction of the input models, it does not guarantee the deduction of sound process models.

Sun et al. [14] focus on merging block-structured process models. They, however, provide a merge for disjoint process fragments, while in this paper we focus on the merger of variants of the *same* process. Due to the merger of disjoint process fragments, some activities are marked as redundant and removed from the resulting model, which we consider undesirable as it does not allow for the deduction of the input models. Finally, their approach does not allow for configurations, i.e. the resulting model is a non-configurable process model.

Table 1 lists the different merging approaches and their adherence to our requirements.

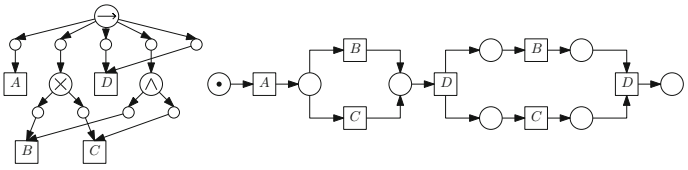

Fig. 1. A process model with the corresponding Petri net

3 CoSeNet and Metrics

Here, we introduce our new representations of process models and configurable process models, i.e. process models and CoSeNets. Furthermore, the metrics used for the experimental evaluation are elaborated on.

3.1 Process Model

For our purposes, a process model is captured as a tree-like representation of a block-structured process model. However, we allow for sharing subprocesses, i.e. some subtrees might have multiple incoming edges to support reuse. Therefore, we use Directed Acyclic Graphs (DAGs) to represent process models.

Fig. 1 depicts a process model with its corresponding Petri net. The top node in the process model denotes the root and is a SEQ node (sequence, $\rightarrow$). The SEQ node has 5 children, i.e. three activity nodes (A and twice D) and two operator nodes: XOR ($\times$) and AND ($\wedge$). Although the context in which D is executed might be different for both D's, from a control-flow perspective we assume they are the same D. A SEQ node executes its children in the order in which they occur, thus A is the first and the second D is the last. The XOR node denotes an exclusive choice between any of its children, and the AND node denotes the parallel execution of its children. Apart from the SEQ, XOR, and AND, we currently support the OR node (one can execute any number of children but at least one) and the DEF node (deferred choice, i.e. the choice based on events instead of data). Furthermore, we support LOOP nodes, which have three children: a *do* child, a *redo* child, and an *exit* child. The do child is the root of the subgraph representing the body of the loop. After having executed the do child, there is a choice to either execute the redo child and, afterwards, execute the do child again, or to execute the exit child and exit the loop construct. The choice between the redo child and the exit child can be either exclusive (LOOPXOR) or deferred (LOOPDEF). For the sake of brevity, we use LOOP as a shorthand for both LOOPXOR and LOOPDEF.

3.2 CoSeNet

A CoSeNet is an extension of the process model we discussed hitherto. Fig. 2 depicts (a) the process model from Fig. 1 extended with some extra annotations and nodes, and (b) a second CoSeNet which we want to merge with (a). The stop

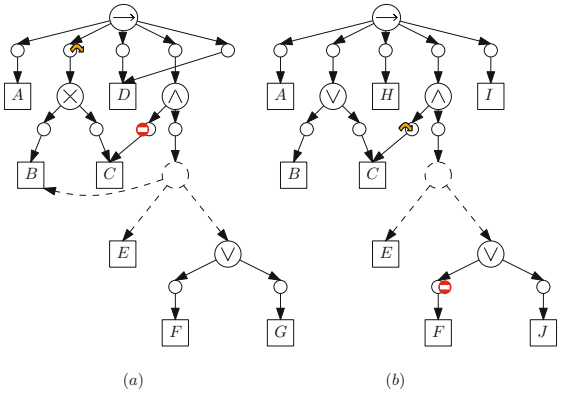

Fig. 2. Two CoSeNets we want to merge

sign denotes that this branch can be *blocked*, i.e. the execution of this subgraph can be prevented. *Hiding* a particular branch, i.e. substituting the branch by a silent transition (τ), is denoted with the orange arrow. Finally, we have added a *placeholder* node (dashed circle) to offer the user to select a subgraph to replace this placeholder node. In this example, the user can select to substitute the placeholder node by the activity node B, E, or by the subgraph rooted at the OR node. A placeholder node is used instead of an exclusive choice to select a subgraph. This to shift the moment of choice from run-time to configuration-time. Fig. 1 can be obtained from (a) by not blocking the blockable branch, not hiding the hideable branch, and replacing the placeholder node by activity node B.

3.3 Metrics

In order to compare our approach with existing approaches w.r.t. complexity (specifically, La Rosa et al. [2] and Gottschalk [7]), we use the complexity metrics used in [4] since this is a continuation of that work. We elaborate briefly on the used metrics. For a more complete discussion, we refer the reader to [4,15,16,17].

Control-Flow Complexity (CFC) [16] computes for every operator a weight based on the number of outgoing edges. The CFC for a process model is the summation of the weights for the individual operators in that process model.

The *density* of a process model [15] is defined as the amount of edges in the model divided by the total amount of edges possible in that model.

With the *Cross-Connectivity (CC)* metric [17], one first computes the weight of the different nodes (connectors and tasks) in the process model (based on the amount of outgoing edges). Afterwards, the weight of the edge between two nodes is deduced from the weight of the nodes the edge is connected to. From this, the maximal weight for the paths between two nodes u and v is computed, where a path is a sequence of edges. Finally, the summation of paths with the

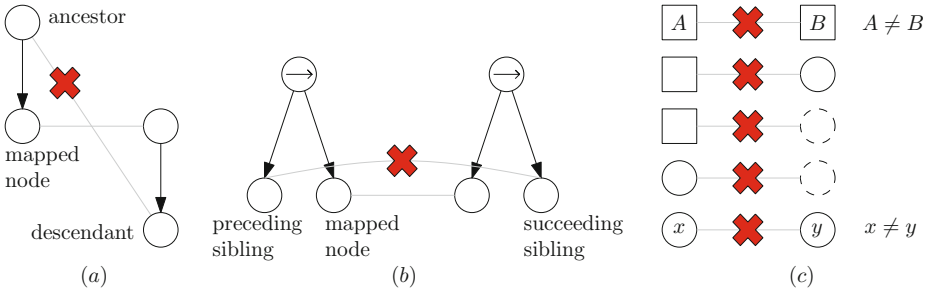

Fig. 3. Restrictions on a proper CoSeMap

largest weight between all pairs of nodes, is divided by the total amount of edges possible in a directed graph with N nodes (i.e. $N \cdot (N-1)$).

4 Merge

When merging two CoSeNets into a single CoSeNet it is important to know which nodes from the original CoSeNets may be merged into a single node. For this reason, we introduce the concept of a *node mapping* between both original CoSeNets, called a *CoSeMap*: Only if a node from one CoSeNet is mapped onto a node of the other CoSeNet, then these nodes may be merged into a single node. For the sake of simplicity, we assume a one-to-one correspondence between nodes, and a CoSeMap corresponds to an injective function to and from the nodes in both CoSeNets (for the sake of convenience, we assume a CoSeMap to be symmetrical).

As a CoSeNet corresponds to a DAG, the CoSeNet that results from a merge may not contain any cycles. Cycles may appear in the resulting CoSeNet if an ancestor node of a mapped node in one CoSeNet is mapped onto a descendant node of the node that the mapped node is mapped onto in the other CoSeNet (see Fig. 3(a)). Second, for a resulting sequence node a correct ordering of its children should be feasible, which is impossible if any preceding sibling of a mapped child in one CoSeNet is mapped onto a succeeding sibling of the node the mapped child is mapped onto in the other CoSeNet (see Fig. 3(b)). It makes no sense to map a node from one type (activity, operator, placeholder) to a node of another type, or to map an activity node to another activity node with a different label (see Fig. 3(c)). Finally, we only allow the mapping of LOOP nodes if and only if all nodes related to the LOOP nodes are mapped. I.e. the root of the *do* subgraph, *redo* subgraph, and the root of the *exit* subgraph. For these reasons, we require a CoSeMap to be *proper*:

1. No ancestor node is mapped onto a descendant node;
2. No preceding child is mapped onto a succeeding child for any sequence node;

3. All nodes are mapped onto nodes from the same type and with the same label (in case they are activity nodes);
4. Loop nodes are mapped onto each other if all children are mapped in order.

Given a proper CoSeMap between them, two CoSeNets can be merged in a straightforward way, which is the *CoSeMerge*:

1. All nodes from the first CoSeNet are added to the resulting CoSeNet;
2. All unmapped nodes from the second CoSeNet are added;
3. All edges from the first CoSeNet are added;
4. All edges that involve some unmapped node from the second CoSeNet are added, but only after any mapped node is replaced by the node it is mapped onto from the first CoSeNet;
5. Configuration options from the second CoSeNet that involve only mapped nodes are added (if needed) to a corresponding branch from the first CoSeNet. Furthermore, some extra configuration options have to be added (see [5], for the exact details);
6. A new root node is added, which is a placeholder node with both root nodes as children. If both root nodes are mapped onto each other, the new root node will only have an edge to the root of the first CoSeNet.

Please note that edges from a SEQ/LOOP node are added in such a way that the order in which the children occur of both original nodes are taken into account. The CoSeMerge yields a CoSeNet containing the union of behaviour of the input CoSeNets.

When applying our CoSeMerge on any two CoSeNets (say, N_1 and N_2), it is easy to see that N_1 can be instantiated from the resulting CoSeNet as all nodes and edges from N_1 are present: After having removed all other nodes and edges, and after having removed the new root placeholder node and configuration options that were only present in N_2, the resulting graph is identical to N_1. For N_2 this is a bit harder to see, as some nodes and edges from this net are not present. However, it is clear that only those nodes and edges are left out for which an alternative exists in N_1. Thus, after having removed all other nodes and edges, and the new root placeholder node and configuration options only present in N_1, the CoSeNet is identical to N_2. Hence, both N_1 and N_2 can be instantiated from the resulting CoSeNet, which means that the CoSeMerge is reversible.

In the remainder of this paper, we will describe two different ways to construct a proper CoSeMap, called an *activity* CoSeMap and an *extended* CoSeMap. An activity CoSeMap maps only activity nodes; it is computed by taking all pairs of activity nodes with the same label. Recall that we assume that a CoSeNet does not contain two activity nodes with the same label. If a CoSeNet contains two activity nodes with the same label, we can combine these into a single activity node. Fig. 4 shows the resulting CoSeNet after having merged both CoSeNets from Fig. 2 using an activity CoSeMap. An extended CoSeMap takes an activity

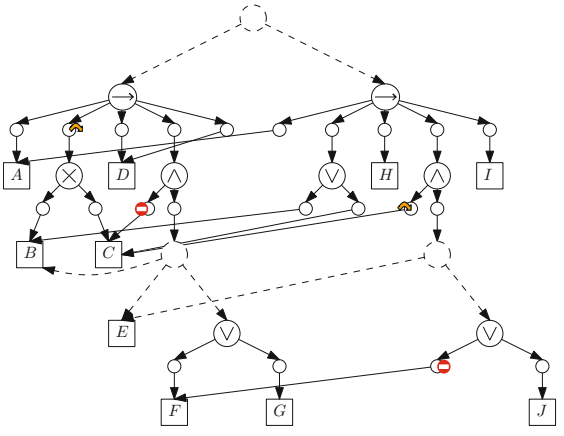

Fig. 4. Merging the models from Fig. 2 using an activity CoSeMap

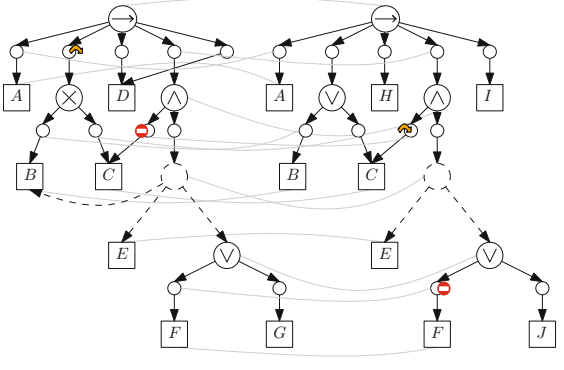

Fig. 5. An extended CoSeMap (horizontal gray lines) for merging the CoSeNets from Fig. 2

CoSeMap as a starting point, but extends this CoSeMap with operator nodes and placeholder nodes (if possible) in such a way that the number of mapped nodes is maximised. The basic strategy for constructing the extended CoSeMap is to map one node onto another node (of the same type) if some child of the first node is mapped onto some child of the second node. As a result, the construction of this CoSeMap works in a bottom-up way, as initially we only have activity nodes that are mapped onto each other, which reside at the bottom of a CoSeNet. Note that it is possible that we have to choose between alternative ways to extend a current partial extended CoSeMap. Hence, given an activity map, multiple extended CoSeMaps may exist. If this is the case, we arbitrarily choose one of the alternatives. Fig. 5 shows a possible extended CoSeMap for merging both CoSeNets from Fig. 2, Fig. 6 shows the result after having merged these CoSeNets using this CoSeMap.

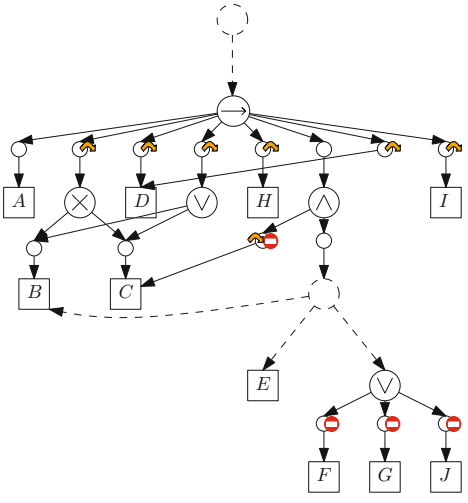

Fig. 6. Merging the models from Fig. 2 using the extended CoSeMap from Fig. 5

5 Experimental Evaluation

In the previous sections, we have argued that CoSeNets can only be instantiated to sound process models. Furthermore, the CoSeMerge ensures that the merged CoSeNet is reversible. In this section, we want to show that these attractive properties of soundness and reversibility do not incur a penalty on the complexity of the configurable process model.

The implementation of the construction of an activity CoSeMap, the construction of an extended CoSeMap, and the CoSeMerge have been implemented as ProM 6 plug-ins[1]. The construction of an activity CoSeMap is a straightforward implementation of string comparison on the activity labels. The construction of an extended CoSeMap is computed via linear programming. Linear programming is used since we have an maximisation problem, i.e. we desire that our extended CoSeMap is maximal. If our extended CoSeMap is maximal, it allows us to share as much subgraphs as possible, which reduces the duplication of subgraphs and hence the complexity of the configurable process model. After the merger, the CoSeNet is converted into a YAWL [18] model, which can be analysed in a repaired version of ProM 5.2[2]. Noteworthy facts of this conversion are that it treats a placeholder node as an XOR operator node, that it will reuse the YAWL fragment that corresponds to an operator node if possible, but that it will not reuse the YAWL fragment that corresponds to an activity node.

[1] `http://www.promtools.org/prom6/`

[2] `http://win.tue.nl/coselog/files/ProM-CoSeLoG-20110802.zip`, which corresponds to ProM 5.2 with some bugs in the algorithms to compute the various metrics have been fixed.

Table 2. The complexities of the models using different merging techniques

Municipality	GBA 1	GBA 2	GBA 3	MOR	WOZ
Mun_A	5	21	11	42	12
Mun_B	3	29	11	23	8
Mun_C	2	38	28	29	14
Mun_D	3	35	18	24	11
Mun_E	6	25	26	25	25
Mun_F	3	21	9	44	15
Mun_G	5	21	9	20	15
Mun_H	5	29	11	18	11
Mun_I	3	41	9	28	11
Mun_J	3	29	8	25	26
Act. CoSeMap	56	435	209	397	223
Ext. CoSeMap	39.8 $\pm$4.3	126.3 $\pm$10.6	172.9 $\pm$23.2	262.4 $\pm$18.9	134.1 $\pm$13.6
La Rosa et al.	146.6 $\pm$12.9	781.3 $\pm$42.7	412.7 $\pm$16.5	937.8 $\pm$34.3	707.1 $\pm$34.9
Gottschalk	80	317	210	-	335

We compare our approach to the approaches from Gottschalk [7] and La Rosa et al. [2]. As the process models used in [4] were not well-structured, we could not use these process models directly and had to modify them for our analysis.

Table 2 lists the values for the various complexity measures for the individual process models as well as for the merged process model. In case an approach yields a non-deterministic result, the average μ and standard deviation σ of 10 results are listed as "$\mu \pm \sigma$". "Act. CoSeMap" and "Ext. CoSeMap" represent the merger with an activity CoSeMap and with an extended CoSeMap. The Synergia tool-set[3], implements the merge by La Rosa et al. [2]. Finally, "Gottschalk" is the implementation of the EPC-merge by Gottschalk in ProM 5.2[4] [7].

From Table 2, one can see that the complexities of the different approaches vary significantly, i.e. ranging from roughly a factor 2 (GBA 1) up to roughly a factor 6 (GBA 2). Complexity metrics allow us to compare processes in an quantitative manner. In general, a higher complexity for an approach means a worse approach than an approach which yields models with a lower complexity (Occam's Razor). See [15], for an extensive evaluation of the different metrics.

Placing the 10 models next to each other can be seen as a base case (which corresponds to the activity CoSeMap), as our conversion to YAWL will not reuse YAWL fragments that correspond to activity nodes, as mentioned earlier). We wish to avoid that the complexity of the configurable process model is significantly greater than the sum of the complexities of the individual models. In case of La Rosa et al. [2], they merge different connectors into a single connector which increases the CFC metric, especially with OR-operators which are exponential in the out-degree.

[3] `http://www.processconfiguration.com/`
[4] `http://promtools.org/prom5/`

The approach of Gottschalk [7] is in some cases comparable (w.r.t. complexity) to our approach (e.g. GBA 3), but yields in some cases significantly higher levels of complexity (e.g. GBA 2) or is not computable (e.g. MOR). In general, Gottschalk performs similar to the activity CoSeMap.

All in all, it can be concluded that, at least with this set of models, we achieve a comparable (or even lower) complexity w.r.t. existing techniques. Hence, it seems possible to obtain sound and reversible process models without paying a too expensive price for this, or actually no price at all.

6 Conclusion

Existing techniques allow for the instantiation of unsound process model from a configurable process model. Furthermore, some techniques are not reversible. Our solution successfully addresses both problems. Soundness is addressed by considering tree-like representations of process models and CoSeNets. The merge as we defined it in this paper takes care of the reversibility property. When merging CoSeNets into a single CoSeNet, the original CoSeNets are instantiations of the resulting CoSeNet.

Apart from defining our solution, we also applied our solution on the process models from the CoSeLoG project. This evaluation supports the view that the complexity of our approach is similar to/lower than the approach by Gottschalk and La Rosa et al. Thus, the guarantee of soundness and reversibility does not incur a penalty on the complexity of the configurable process models.

However, there is still room for improvement. This paper is, therefore, to be considered as a starting point. There are numerous ways in which we want to continue the development of this new approach. Amongst other in the following directions: this research has been started to support the processes of the municipalities. Therefore, we plan to extend our CoSeMerge and CoSeMaps, e.g. different process models have different granularity (see the work of Weidlich et al. [19]). Furthermore, as noted in the experimental evaluation, some quality dimensions have not (yet) been addressed. We intend to define these quality dimensions on our CoSeNets.

Finally, we would like to extend CoSeNets with resources and data, in order to fully support the processes of the municipalities.

Acknowledgements. We would like to thank Marcello La Rosa for his comments, which improved this paper significantly.

References

1. van der Aalst, W.M.P.: Verification of Workflow Nets. In: Azéma, P., Balbo, G. (eds.) ICATPN 1997. LNCS, vol. 1248, pp. 407–426. Springer, Heidelberg (1997)
2. La Rosa, M., Dumas, M., Uba, R., Dijkman, R.M.: Business Process Model Merging: An Approach to Business Process Consolidation (in press, 2012)

3. Kopp, O., Martin, D., Wutke, D., Leymann, F.: The Difference Between Graph-Based and Block-Structured Business Process Modelling Languages. Enterprise Modelling and Information Systems 4(1), 3–13 (2009)
4. Vogelaar, J.J.C.L., Verbeek, H.M.W., Luka, B., van der Aalst, W.M.P.: Comparing Business Processes to Determine the Feasibility of Configurable Models: A Case Study. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM Workshops 2011, Part II. LNBIP, vol. 100, pp. 50–61. Springer, Heidelberg (2012)
5. Schunselaar, D.M.M., Verbeek, H.M.W., van der Aalst, W.M.P., Reijers, H.A.: Creating Sound and Reversible Configurable Processes Models using CoSeNets. Technical Report BPM Center Report BPM-11-21, BPMcenter.org (2011)
6. Schunselaar, D.M.M.: Configurable Declare. Master's thesis, Eindhoven University of Technology, The Netherlands (2011)
7. Gottschalk, F.: Configurable Process Models. PhD thesis, Eindhoven University of Technology, The Netherlands (2009)
8. Rosemann, M., van der Aalst, W.M.P.: A Configurable Reference Modelling Language. Information Systems 32(1), 1–23 (2007)
9. van der Aalst, W.M.P., Lohmann, N., La Rosa, M.: Ensuring Correctness During Process Configuration Via Partner Synthesis. Information Systems 37(6), 574–592 (2012)
10. van der Aalst, W., Lohmann, N., La Rosa, M., Xu, J.: Correctness Ensuring Process Configuration: An Approach Based on Partner Synthesis. In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 95–111. Springer, Heidelberg (2010)
11. Li, C., Reichert, M., Wombacher, A.: Discovering Reference Models by Mining Process Variants Using a Heuristic Approach. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 344–362. Springer, Heidelberg (2009)
12. Mendling, J., Simon, C.: Business Process Design by View Integration. In: Eder, J., Dustdar, S. (eds.) BPM Workshops 2006. LNCS, vol. 4103, pp. 55–64. Springer, Heidelberg (2006)
13. 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, 7029–7040 (2011)
14. Sun, S., Kumar, A., Yen, J.: Merging Workflows: A New Perspective on Connecting Business Processes. Decision Support Systems 42(2), 844–858 (2006)
15. Mendling, J.: Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness. Springer (2008)
16. Cardoso, J.: Control-flow Complexity Measurement of Processes and Weyuker's Properties. In: 6th International Enformatika Conference, Transactions on Enformatika, Systems Sciences and Engineering, vol. 8, pp. 213–218 (2005)
17. Vanderfeesten, I.T.P., Reijers, H.A., Mendling, J., van der Aalst, W.M.P., Cardoso, J.: On a Quest for Good Process Models: The Cross-Connectivity Metric. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 480–494. Springer, Heidelberg (2008)
18. ter Hofstede, A.H.M., van der Aalst, W.M.P., Adams, M., Russell, N. (eds.): Modern Business Process Automation: YAWL and its Support Environment. Springer (2010)
19. Weidlich, M., Dijkman, R., Mendling, J.: The ICoP Framework: Identification of Correspondences between Process Models. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 483–498. Springer, Heidelberg (2010)