Hierarchical Structuring of a Workflow Model in Petri-Net

Xinlei Zhao, Yongsun Choi

Dept. of Systems Management & Engineering, Inje University, Kimhae, Korea
xinleizhao@yahoo.com, yschoi@inje.ac.kr

ABSTRACT

In this paper, we introduce the way of deriving hierarchical structure of a workflow model represented in classical Petri-net, even for the cases with cycles, which allows handling a workflow model efficiently. More specifically, our method identifies any block structures as candidates for the subprocesses and represents them as a single block node in the upper layer of the hierarchical model. The proposed method can make workflow analysis and design more accurate and efficient and further lead to a better design on a collaborate environment.

Keywords: Workflow Model, Hierarchical Structuring, Process Abstraction, Hierarchical Petri-Net

1. INTRODUCTION

Recently, workflow management technology has shown to be one of the driving tools in accelerating process-oriented applications ([8], [11], [14]). There are typically two stages of workflow management, workflow specification and workflow execution. The former defines a workflow model, and the latter generates workflow instances guided by the workflow model [15]. The objective of workflow modeling is to provide high-level specification of processes that are independent of the implementation intricacies of the target workflow management system [16]. The recent surges of e-business process automation efforts in the corporations worldwide places workflow analysis and design as a fundamental task of more importance and asks for a model capable of easy interpretation and various analyses with efficiency [4].

Petri-nets, a directed bipartite graph with two node types called places and transitions, have been utilized as an effective methodology in system modeling ([6], [13]). The Petri-net models provide clear graphical representation and profound expressiveness for modeling concurrent, qualitative and quantitative properties. Moreover, the availability of various analysis utilities with Petri-nets provides a means to verify and validate a system, thus has made Petri-nets widely applied for analyzing systems in many areas, including workflow models ([3], [4], [7]). However, introducing the states of the system in the model, called places, the process model represented in classical Petri-nets almost doubles the complexity of representation, in number of nodes and arcs, than the activity-based direct graph model representation. This makes it hard to interpret the business processes represented in Petri-nets for the human designers and rare to be employed by commercial workflow management systems.

Hierarchical Petri-nets, as a type of high-level Petri-nets, provide a mean to model complex system in a more manageable way. When to model complex processes, a classical Petri-net is structured hierarchically by introducing subprocesses, using either a top-down or a bottom-up approach. This divide-and-conquer strategy of dividing a complex process into smaller subprocesses allows overcoming the complexity for further analysis. Furthermore, the identification of subprocesses provides the way of reusing previously defined processes and often makes it possible to model a complex process more quickly ([1], [2]). Although hierarchical decomposition of a complex workflow process is a useful step in workflow analysis and design, until now there has been no reported work of its automated support in the literature. When the process model is complex, it is quite perplexing for a human designer to recognize substructures of potential subprocesses [5].

In this paper, we introduce the way of deriving hierarchical structure of a workflow model represented in classical Petri-nets, even for the cases with cycles, which allows handling workflow model efficiently. More specifically, our method identifies any block structures as candidates for the subprocesses and represents them as a single block node in the upper layer of the hierarchical model. In case of a model with cycles, our method partitions the given model into several acyclic subsets according to feedback structures, and utilizes the resulting partitioned structures when to compose a block. This paper is organized as follows. In section 2, we present the preliminary concepts such as a workflow-net and its hierarchical extension. Section 3 describes the steps of hierarchical structuring for the workflow models represented in classical Petri-nets, with an illustrative example. Section 4 concludes the paper.

2. A WORKFLOW-NET AND ITS HIERARCHICAL EXTENSION

2.1 Workflow-nets

Petri-nets, represented by a directed bipartite graph in which nodes are either places or transitions, are widely studied and successfully applied in many discrete-event dynamic systems ([6], [13]). Places, represented by circles, describe states or conditions of the system and

transitions, represented in rectangles, describe the events or transactions. The relationships between them are represented by a set of arcs in either direction.

The strong mathematical foundation of Petri-nets and the availability of a wide range of supporting tools have made them popular including workflow domain ([3], [4], [7]). Petri-net-based workflow models represent business logics by a formal but also graphical language. Workflow procedures are specified using a technique with formal semantics of the classical Petri-net and several extensions (color, time, or hierarchy). Availability of abundant analysis techniques [12] is another driving force for the Petri-net-based workflow models. In general, these methodologies can be used to prove properties, like safety, invariance, deadlock, etc., and to calculate performance measures, like response times, waiting times, occupation rates, etc.

A *workflow-net* is a Petri-net-based representation of a workflow process with some syntactical requirements [2]. A workflow process defined in terms of a Petri-net has a single input place *start* and a single output place *end*. And each transition or place should lie on a directed path from *start* to *end*. In other words, there should be no "loose" nodes. Thanks to this requirement, each node can be reached from the place *start* and the place *end* is always reachable from each node, by following a number of arcs. Of all workflow perspectives, e.g., control-flow, data, organization, task, and operation, the control-flow perspective is the most prominent one because it defines the backbone of the workflow on which other perspectives can be specified [4]. Workflow-nets focused on control-flow perspective in modeling a workflow process definition are utilized in this paper.

2.2 Hierarchical workflow-nets

Although the Petri-net has many features to model concurrent, qualitative and quantitative properties, the strict representation of a complex business process is hard to read and understand for the human designers. When facing more complicated situations, the classical Petri-nets become too large and inaccessible, or it is not possible to model a particular activity. The high-level Petri-net, extended with color, time, or hierarchy, etc., helps for more close representation of the problem situation, specific with the perspective considered. The high-level Petri-net, inherits all the advantages of the classical Petri-net, such as the graphical and precise nature, the firm mathematical foundation, and abundance of analysis methods [1].

The workflow-nets encountered in practice have many tasks with very complex interaction structures. For the designer of such a workflow the complexity is overwhelming and communication with end-users using one huge diagram is difficult [5]. Hierarchical Petri-net, as a type of high-level Petri-net can help to model complex situations in more structured and accessible way [1]. A complex workflow can be decomposed into smaller subflows until the desired level of detail is reached. In addition, this mechanism can be utilized for the reuse of existing workflows [5]. Although hierarchical decomposition of a complex workflow process is a useful step in workflow analysis and design, until now there has been no reported work of its automated support in the literature. Next, we explain our approach of identifying and abstracting the blocks of the workflow model as potential subprocesses to make the process more simplified and manageable.

3. HIERARCHICAL STRUCTURING OF A WORKFLOW-NET

Hierarchical structuring of a workflow-net by abstraction is used to identify the potential sub processes and to make the original process more manageable. Our workflow abstraction method utilizes the concept of inline blocks. An *inline block* is a subset of nodes and arcs among those nodes that satisfies the blocked transition property [15]. According to the Workflow Management Coalition, the *blocked transition property* states that any *inward transition* to the inline block can only occur to the start node of the block and that any *outward transition* from the inline block can only occur at the end node of the block. An inline block is *reducible* to a *block node* or may be modeled as a sub-process of the original process definition. This helps in managing a large-scale model, including verification of structural conflicts [5], being represented as a hierarchy of simple smaller models. Identifying inline blocks manually from a complex workflow [5] is a difficult task even for an experienced process designer.

3.1 Steps of hierarchical structuring

Fig. 1 shows the block diagram of the suggested method for the hierarchical structuring of a workflow model represented in Petri-net. As shown in Fig. 1, our method is comprised of three main steps: 1) Partition a cyclic workflow-net into a set of acyclic substructures; 2) Compose next available candidate blocks; 3) Abstract any block satisfying blocked transition property into a block node.

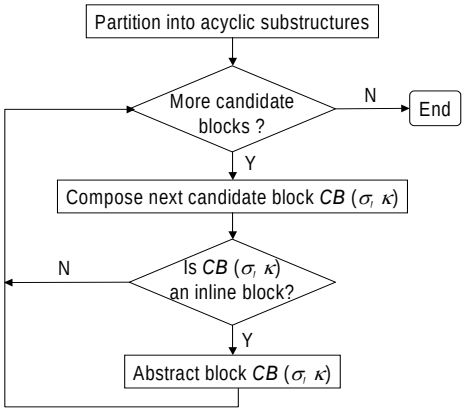


Fig.1. Illustration of the algorithm

Partitioning a cyclic workflow model into acyclic substructures

Step 1 identifies feedback structures and partitions the given cyclic model into a set of acyclic substructures iteratively, even for the models of nested feedback structures. The rank of each node, utilized in Step 2 when to select the next candidate block, is computed with "back edges" [10] temporarily removed. According to the *order of feedback*, nodes are classified into the so-called n^{th}*-order Feedback Nodes*, denoted by $\boldsymbol{FN}^n$. The *n-th* iteration of Step 1 identifies $\boldsymbol{FN}^n$, by classifying the set of nodes $\boldsymbol{N}$. The detailed explanation of Step 1 with illustrative examples is given in [9], with a directed graph representation.

Candidates of inline blocks with cycles

Step 2 first configures potential inline blocks, substructures of the given model, with one of the *Split nodes* (except Feedback Splits) or *Feedback Joins* as the block start node, called the *source*, and one of the *Join nodes* (except Feedback Joins) or *Feedback Splits* as the block end node, called the *sink* [9]. Those potential inline blocks are referred to as *candidate blocks* because they may or may not satisfy the blocked transition property. This initial candidate block, composed with split and join nodes as the border nodes, can be easily extended without further verification effort by adding sequential nodes at the borders. This way of composing candidate blocks can reduce computational cost significantly by focusing on the core candidate blocks.

A candidate block, composed with a node σ as source and another node κ as sink, will be denoted by $\boldsymbol{CB}$ (σ, κ). For convenience, the set of nodes that spans the candidate block $\boldsymbol{CB}$ (σ, κ) will be also denoted as $\boldsymbol{CB}$ (σ, κ), without confusion. The algorithm starts with the simplest candidate block and extends to larger ones, iteratively. More specifically, the algorithm starts with $\boldsymbol{CB}$ (σ, κ), with σ as one of the candidate sources of maximum rank and κ as one of the candidate sinks, reachable from σ, of minimum rank. At next iteration, new candidate block is selected by fetching new sink κ' of the next higher rank with the same source σ of the current candidate block; when all candidate blocks with σ as source are evaluated or excluded, The algorithm fetches new source σ' of the next lower rank from the stack of candidate sources and proceeds forward.

Blocked Transition Property for Cyclic Workflows

Step 3 checks the *blocked transition property* [15], i.e., no disallowed inward and outward arcs should exist for the newly composed candidate block. If the candidate block $\boldsymbol{CB}$ (σ, κ) satisfies the blocked transition property, it is abstracted into a new block node, otherwise the proposed method fetches next available candidate block.

In case violations of the blocked transition property happen only at the source or the sink, we can compose an inline block by splitting the source or the sink. Fig. 2 illustrates an example of composing an inline block with $\boldsymbol{CB}$ (σ'', κ'') by splitting both the source σ and the sink κ of $\boldsymbol{CB}$ (σ, κ). Note that if only one of these two violations occur, we can compose an inline block with $\boldsymbol{CB}$ (σ'', κ) by splitting the source σ, $\boldsymbol{CB}$ (σ, κ'') by splitting the sink κ. The arc from node A or the arc to node B, where A, B $\notin$ $\boldsymbol{CB}$ (σ'', κ''), does not violate the blocked transition property for the resulting inline block of $\boldsymbol{CB}$ (σ'', κ''). The newly added nodes σ'' and κ'' in Fig. 2 are null transition of no tasks to perform. The null state nodes σ' and κ' are added to meet the requirements of the Petri-net model, that is nodes should be connected with others of distinct types. Note that it is not necessary the source σ and sink κ are of same node type and is worth for splitting only at the source or at the sink.

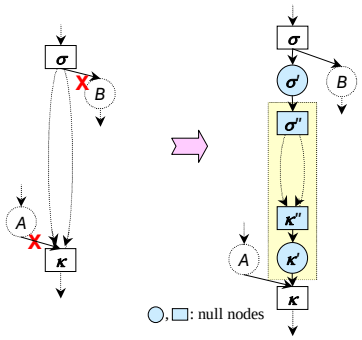


Fig. 2. Composing an inline block by splitting the source or the sink

Types of block structures

According to the node type of the source or the sink, each block structure can be classified into one of the four types that can be reviewed as follows from the Petri-net model perspective:

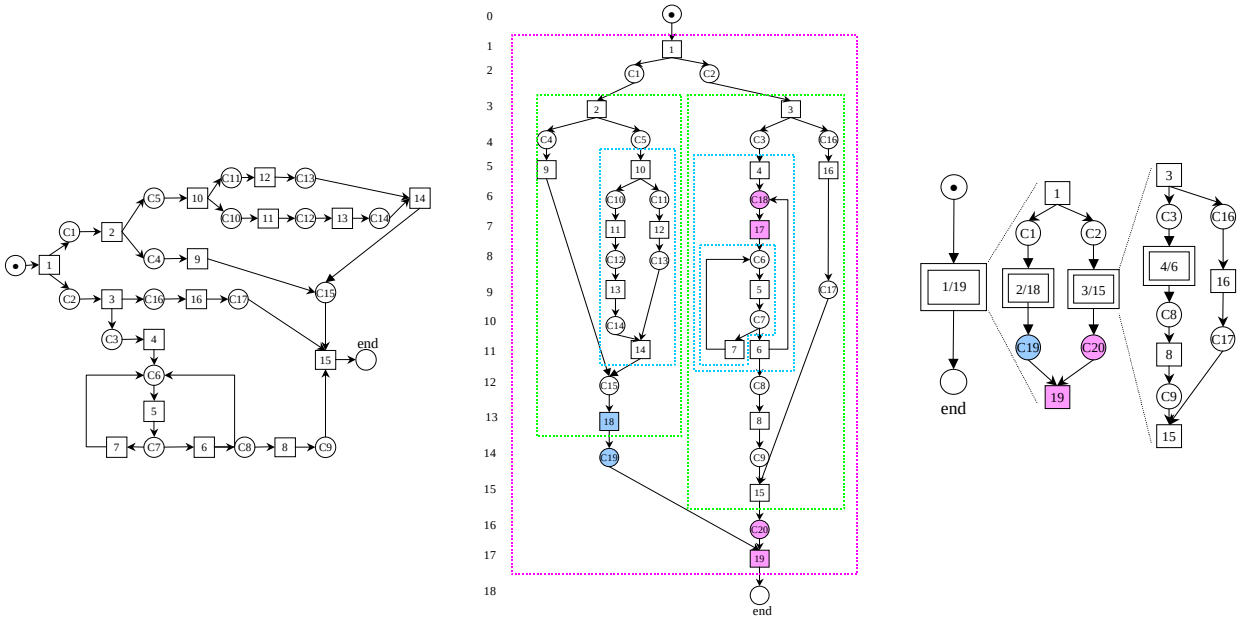
Transition-to-Transition block (T-T block) or Place-to-Place block (P-P block): The abstracted block node of a *subnet*, either type of a transition or a state, is connected with distinct type of nodes, and the resulting abstracted model meets the requirement of Petri-net representation. The block structure of any of these types can be represented as a separate subprocess, making the workflow model simpler. When a particular block structure, represented as a subprocess recurs several times in distinct workflow processes, the reuse of this subprocess often makes it possible to model a complex process more quickly. $\boldsymbol{CB}$ (σ'', κ'') in Fig. 2 is a *T-T block*.

Transition-to-Place block (T-P block) and Place-to-Transition block (P-T block): These types of block structures will make it ambiguous to assess the types of the abstracted block nodes. Moreover, whatever node type is assessed for the abstracted node, the resulting abstracted model, with the abstracted node connected with a node of same type, will not meet the requirements

of the Petri-net representation. To handle these mismatching source-sink block structures, we basically extend these structures by adding a sequential node prior to the source or after the sink. When there exists no such sequential node, our method inserts a pair of new null nodes either before or after these structures and then extends the structure. Our method gives priority to compose the extended block into a *T-T block*, when to insert new pair of nodes, if necessary, and to select a sequential node to add.

3.2 An illustrative example

Fig. 3(a) shows an example workflow-net model, of 16 transitions and 17 places, for a process of organizing a party modified from [2]. Figure 3(b) shows the normalized model with ranks of nodes indicated to the left. Figure 3(b) also shows 6 substructures, indicated by dotted boxes, composing each corresponding block satisfying blocked transition property. Note that two pairs of nodes (17, C18) and (19, C20) are inserted to split nodes C6 and 15 and to compose the block structures of ***CB*** (C6, C7) and ***CB*** (3, 15), respectively. Another pair of nodes (18, C19) is inserted to make the mismatching block of ***CB*** (2, C15) into a *T-T block* of ***CB*** (2, 18) extended by adding node 18. Note also that another mismatching block of ***CB*** (C18, 6) is extended into a *T-T block* of ***CB*** (4, 6) by adding an existing node 4 before ***CB*** (C18, 6).



(a) (b) (c)

Fig. 3 (a) The original workflow-net model modified from [2], (b) the normalized model with blocks indicated, and (c) Part of top 3 layers of the resulting hierarchical model of total 5 layers

4. CONCLUDING REMARKS

The business processes tend to be more complicated and have more functions according to the needs of the internal and external request. Thus the resulting process models with many transactions and activities are hard to manage. Moreover, enterprises try to make their business processes more extensible to inter-organizational workflow models for the cross-organizational process, requiring several distinct process design teams should work cooperatively.

Though hierarchical decomposition of a business process can help to manage complex situations in more structured and accessible way [1], until now there has been no reported work of its automated support in the literature. In this paper, we introduced the way of hierarchical structuring of a workflow model by detecting and abstracting the block structures as potential subprocesses to make the given large-scale workflow model more simplified and manageable. Our method is comprised of three main steps: 1) Partition a cyclic workflow-net into a set of acyclic substructures; 2) Compose next available candidate blocks; 3) Abstract any block structures, as candidates for the subprocesses, into a block node in the upper layer of the hierarchical model. Automated support of hierarchical structuring of large-scale enterprise processes will make them more manageable and further lead to a better process design on a collaborate environment.

REFERENCES

[1] Aalst, W.M.P. van der, K.M. van Hee, and G.J. Houben, "Modelling workflow management systems with high-level Petri nets." In G. De Michelis, C. Ellis, and G. Memmi, editors, *Proceedings of the second Workshop on Computer-Supported Cooperative Work, Petri nets and related formalisms*, pp. 31-50, 1994.

[2] Aalst, W.M.P. van der, K. M. van Hee, *Workflow Management: Models, Methods, and Systems*, MIT Press, 2000.
[3] Aalst, W. M. P. van der, K. van Hee, "Business Process Redesign: A Petri-net-based approach", *Computers in Industry*, 29(1-2), pp. 15-26, 1996.
[4] Aalst, W. M. P. van der, "The Application of Petri Nets to Workflow Management", *The Journal of Circuits, Systems and Computers*, vol. 8, no. 1, pp. 21-66, 1998.
[5] Aalst, W. M. P. van der, "Workflow Verification: Finding Control-Flow Errors using Petri-net-based Techniques", In W.M.P. van der Aalst, J. Desel, and A. Oberweis, Editors, *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pp. 161-183, Springer-Verlag, Berlin, 2000.
[6] Aalst, W.M.P. van der. "Putting Petri nets to work in industry" *Computers in Industry*, vol. 25, no. 1, pp. 45-54, 1994.
[7] Adam, N. R., V. Atluri, and W. Huang. "Modeling and Analysis of Workflows using Petri Nets," *Journal of Intelligent Information Systems*, vol. 10, pp. 131-158, 1998.
[8] Basu, A. and A. Kumar, "Research commentary: Workflow management issues in e-Business," *Information Systems Research*, vol. 13, no. 1, pp. 1-14, 2002.
[9] Choi, Y. and J. L. Zhao**,** "Partitioning into acyclic flows and 2-phased verification of structural conflicts for a cyclic workflow graph", *The Fourth International Conference on Electronic Business (ICEB2004)*, Dec. 2004, *forthcoming*.
[10] Cormen, T. H., C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms* (2nd Ed.), MIT Press, 2001.
[11] Delphi Group, *BPM2002: Market Milestone Report*, available at http://www.delphigroup.com/.
[12] Georgakopoulos, D., M. Hornick, and A. Sheth, "An overview of workflow management: from process modeling to workflow automation infrastructure", *Distributed and Parallel Databases*, vol. 3, pp.119-153, 1995.
[13] Murata, T., "Petri nets: Properties, analysis, and applications", *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541-580, 1989.
[14] Sheth, A. P., W. M. P. van der Aalst, and I. B. Arpinar, "Processes driving the networked economy," *IEEE Concurrency*, vol. 7, no. 3, pp. 18–31, 1999.
[15] Workflow Management Coalition, *Glossary. Document Number WfMC-TC-1011*, 1999.
[16] Workflow Management Coalition, *Interface 1: Process Definition Interchange Process Model. Document Number WfMC TC-1016-P*, 1999.