Workflow Partitioning for Offline Distributed Execution on Mobile Devices

Peter Wakholi and Weiqin Chen

University of Bergen, POB 7802, N-5020, Bergen, Norway
peterokhisa@gmail.com

Abstract. Traditionally, workflow systems are built on the client/server architecture, in which a single workflow server takes the responsibility for the operation of the whole process, thereby requiring connections each time a task is completed. In cases where connection between client and server is not readily available - like in mobile environments, such an approach proves infeasible. Enabling the execution of a group of tasks by mobile clients in distributed and disconnected environments has been proposed as a possible solution. However, the partitioning of a workflow into groups of tasks for offline execution has not been adequately explored. This paper proposes an approach for workflow partitioning and an algorithm that enables automatic discovery of such partitions from a process model as a vital step in assigning grouped tasks. We have implemented the algorithm, evaluated and validated it and proposed ways in which it could be implemented in a real workflow environment.

Keywords: workflow models, partitioning, offline behaviour

1 Introduction

Traditionally, workflow systems are built on the client/server architecture, in which a single workflow server takes the responsibility for the operation of the whole process, thereby requiring connections each time a task is completed. In environments where connections cannot be sustained e.g. devices using mobile networks in rural areas, such an approach would be infeasible. The practice is to allow for offline data collection and require the client device to make connections once network is stable [1]. Solutions that use distributed Workflow Systems have been proposed[2], making it necessary to partition a process model into a group of tasks that can be executed offline.

A number of methods for partitioning workflows for distributed execution have been proposed [3, 4]. However many of the approaches while addressing the need for distributed offline execution of work, do not cater for the need to provide an option that enables the server to maintain control. The partitions created should be simple enough to be executed by a light-weight mobile client and should not in any way alter the underlying logic of the original model maintained on the server. This enables distributed execution to be just an option and not a requirement. In this paper we have

proposed a Petri-net based approach to partitioning workflows, based on structural and behavioural aspects of the original process model. We present a method for partitioning that enables work to be dynamically assigned at different stages of the execution process as long as they can be carried out independent of the original model.

This paper proceeds as follows. In section 2 we explore the requirements and provide rules for partitioning. Section 3 provides an algorithm for automatic discovery of such Partitions. Section 4 provides a theoretical and experimental evaluation of the rules and propositions provided. Related work and the contribution of the paper are discussed in section 5. Finally, section 6 provides future work and further discussions.

2 Workflow Partitioning

2.1 Application Example

Before partitioning a workflow model, one needs to consider the goals and operating environment of the system. We therefore provide a case study of the Promise-Pep Clinical Trial[5] that is considering adding mobile phones as one of the platforms used to collect information. The aim of using workflows is to automate the process of data collection and ensure adherence to pre-defined procedures in the trial protocol. The use of mobile phones for data collection would be when community-based visits are undertaken. We take an instance of a field activity (which is part of larger process) that involves a doctor conducting clinics for HIV+ mothers. First, he looks up the client information (loaded on the mobile device) and records additional information about the visit. Then he does two lab tests (Plasma HIV-1 RNA and Stored plasma) and records the information. Finally, he provides a prescription and records this on his mobile device. At the end of the day, when network connection is established, he uploads this information to the server and the process continues. This portion of work assigned to the doctor is a sub-workflow process that is an independent and complete workflow activity, and does not depend on any outside intervention in order to complete execution.

Conceptually, one can therefore argue that partitioning of workflows needs to ensure that the resulting partitions form some logical workflow process and that there are no data, resource and control flow dependencies outside the partition. [2] provides a set of conditions that workflow partition needs to address, which include that, it must be possible to partition a process model and to allocate the resulting fragments on mobile devices as well as stationary computers; the soundness of the process needs to be ensured; both the overall process model as well as its fragments might have to be adapted during runtime, e.g. to deal with exceptional situations. Based on this case study and related literature, the following are necessary for partitioning process models:

1. A partition should not alter the underlying logic of a process model. Any combination of tasks should preserve the order of execution as defined by the process modeller.

2. All tasks from the partition can be assigned to one resource and can be executed by the service(s) on the client device.
3. All data dependencies for the tasks to be executed are contained in a partition and hence there is no need to connect to the server during the lifetime of its execution.
4. User can define the optimum number of tasks to be assigned or select from a set of possible partitions.

2.2 Workflow Partitioning Rules

We use an example of a workflow net in figure 1, and the unfolded net in figure 2. Unfolding the model as proposed by [6] enables us to determine the dependencies of the tasks in an execution sequence. In the unfolded model, *T9* and *P9* refer to the transition $T8$ and place $P10$. It can be observed that the possible partitions for this model consists of the transitions; $(T4, T5)$, $(T2, T4, T5, T6, T7)$, $(T3, T8)$, $(T2, T4, T5, T6, T7, T8)$. The following do not form valid partitions $(T2, T4)$, $(T5, T7)$, $(T2, T6)$, $(T6, T7)$, $(T1, T3)$, $(T1, T2)$. These only serve as examples since the number of possible combinations are much higher. The following observations can be made about the partitions generated.

1. For all partitions there is one incoming arc one outgoing arc.
2. For all partitions it is possible to move from one transition to another without requiring input or output to the rest of the process model.
3. The groupings that do not form valid partitions either depend on outside conditions or provide output before fully executing.
4. None of the partitions modifies the execution sequence provided in the in figure 1.

Based on these observations, we generate the following rules for partitioning workflow models: Given an unfolding of a workflow net $W = \langle \mathrm{S};\ \mathrm{T};\ \alpha; \beta, \mathrm{is} \rangle$ is, such that $\mathrm{S} = \{ \mathrm{s}_0, \mathrm{s}_1, \ldots\ldots, \mathrm{s}_\mathrm{m} \}$ is a set of states and $\mathrm{T} = \{ \mathrm{t}_0, \mathrm{t}_1, \ldots\ldots, \mathrm{t}_\mathrm{n} \}$ is a set of transitions; the relation $\alpha: \mathrm{T} \rightarrow \mathrm{S}$ associates to each transition its source state; $\beta: \mathrm{T} \rightarrow \mathrm{S}$ associates to each transition its target state. A partition $\mathrm{W}' = \langle \mathrm{S}', \mathrm{T}' \alpha, \beta, \mathrm{is} \rangle$ where $T' = \{t_i, t_{i+1}, \ldots, t_{i+m}\} \subset T$ and $S' = \{s_i, s_{i+1}, \ldots, s_{i+n}\} \subset S$ is possible if:

1. Rule 1: $\forall W' \exists \{\alpha(t_i) = is, \beta(t_i) = s_i, \ldots \beta(t_n) = s_{i+n}\}$ (causal and connected)
2. Rule 2: $\forall W' \neq \{W \notin W'\}$ (No contradiction)

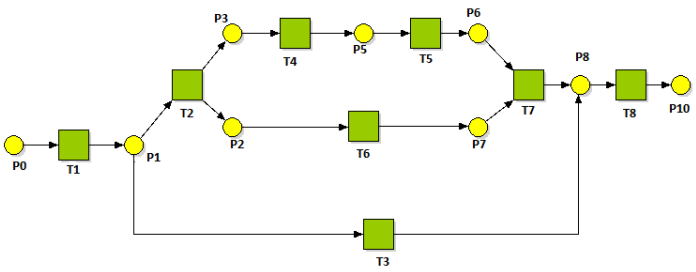

Fig. 1. Example workflow net

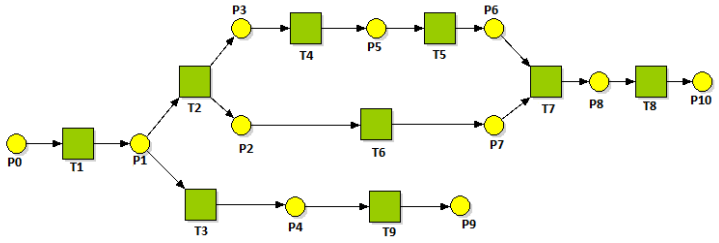

Fig. 2. Workflow net unfolded

3 Automatic Partitioning Algorithm

The relations in the unfolded net can be represented as an incidence matrix that defines the causal relations between the transitions and places as shown in table 1. For a transition column a value of 1 represents an arc entering from a place in the corresponding row, while -1 is for an outgoing arc. The incidence matrix of a directed graph $M = (P, T)$ is a $P \times T$ matrix $B = (b_{ij})$ such that

$$b_{ij} = f(x) = \begin{cases} -1, & \text{if edge } j \text{ leaves transition } j \ (the\ relation\ \beta(t_j)) \\ 1, & \text{if edge } j \text{ enters transition } j \ (the\ relation\ \alpha(t_j)) \\ 0, & \text{otherwise} \end{cases}$$

Table 1. Incidence Matrix for unfolded net

	T1	T2	T3	T4	T5	T6	T7	T8	T9	total
P0	-1	0	0	0	0	0	0	0	0	-
P1	1	-1	-1	0	0	0	0	0	0	-
P2	0	1	0	0	0	-1	0	0	0	0
P3	0	1	0	-1	0	0	0	0	0	-
P4	0	0	1	0	0	0	0	0	-1	0
P5	0	0	0	1	-1	0	0	0	0	0
P6	0	0	0	0	1	0	-1	0	0	0
P7	0	0	0	0	0	1	-1	0	0	0
P8	0	0	0	0	0	0	1	-1	0	-
P9	0	0	0	0	0	0	0	0	1	-
P10	0	0	0	0	0	0	0	1	0	-
total	-	1	-	0	0	0	-1	-	-	0

First we apply rule 2: since there must be only one incoming and one outgoing arc, the sum of all arcs within the Partition must be zero. Therefore any arc that is created within the Partition must eventually come to a close. It can be observed that a node (place or transition) with one incoming arc and one outgoing arc will always have Transition column and Place row sum as 0. Rule 2 therefore proposes that for a Partition, the cumulative sum of all the places and transitions must be zero. Considering one of the Partitions $W_1 = (\mathrm{T2}, \mathrm{T6}, \mathrm{T4}, \mathrm{T5}, \mathrm{T7})$ the columns (transitions) and the rows (places) as shaded in table 1 are summed. It can be observed that the cumulative sum of rows and columns is zero. Rule 1 requires that Partitions should be composed of connected and causally related transitions. We therefore institute a search algorithm that starts with the first transition of the intended Partition and adds adjustment transitions (connectedness) while checking for causality. For every transition added, rule 2 is applied and if the cumulative sum of columns and rows is 0, then a Partition is discovered.

The algorithm for Partitions is given below. A mathematical evaluation proves its validity. Since the net is unfolded, the direction of an arc implies that the place/transition can only be visited once in any execution sequence. A column sum in the incidence matrix represents difference in the number of incoming and outgoing arcs while a row represents those of a place. If a Partition has one input place and one output place, then every new arc leaving a Transition creates an execution path that must always close, thus giving a net sum of zero.

```
Require: A = ⟨S; T; α; β; is⟩          <Unfolded Workflow net>
    M = (P, T)                                  <Incidence Matrix>
    T_j = (b_ij)                       <Initial transition column>
    n = 1
    While (n < Number of transitions) do
      If T_j ⇒ T_j+n then                                  <Rule 1>
        add T_j+n                          <Adjacent transition>
        SumTransitions ∑_j^(j+n) T
        sumplaces ∑_i^(i+n) P
        If (sumTransitions+SumPlaces=0) then               <Rule 2>
          Partition = (T_j, T_j+1, ..... T_j+n
        End If
      End If
    End While
```

4 Evaluation

This algorithm was implemented by creating a module in the PIPE framework[7]. Based on the unfolded model, Partitions for the first Transition were searched, then the next, until all transitions in the sequence had been visited. The result of running the algorithm gave possible Partitions as, $(\mathrm{T2}, \mathrm{T6}, \mathrm{T4}, \mathrm{T5}, \mathrm{T7})$, $(\mathrm{T2}, \mathrm{T6}, \mathrm{T4}, \mathrm{T5}, \mathrm{T7}, \mathrm{T8})$, $(\mathrm{T3}, \mathrm{T9})$ and $(\mathrm{T4}, \mathrm{T5})$. These match with the observation

that were made in section 2.2. The algorithm was evaluated by experimenting on a number of process models whose choice was based on complexity [8] of the underlying constructs by considering the number of Workflow patterns and tasks. Table 2 shows the results of the experiments. A total of five process models were tested, one of which is based on the YAWL for Film [9] process model. For each model, we observed the number of Workflow patterns and the number of tasks. After running the Partition algorithm using the software developed, we counted the number of Partitions discovered and evaluated their correctness and possible omissions.

The hypothesis was that there should be no incorrect or omitted Partitions in order for the rules generated to be valid. This hypothesis was proved to hold for sound Workflow nets, except for state-based patterns like Interleaved Parallel Routing, Milestone, Critical Section, and Interleaved Routing. One common characteristic of all these patterns is that they take a token and return to a place thus giving a net cancellation effect on the Partition. In order to address this problem, the unfolding considered a unidirectional arrow to the transition - based on the fact that our interest is only aimed at ascertaining the state. In addition, Advanced Synchronisation and Multiple instance patterns like the Multi-choice and Multi-merge create large numbers of unfolding, which become complex for analysis.

5 Related Work

There have been attempts to develop a framework that delivers workflow definitions to mobile devices in disconnected environments[10]. The IBM FlowMark [11] is an example of a meta-model that seeks to address the constraints of deploying mobile workflows. Work is loaded to a mobile with the hope that a user is committed to do them. Exotica is an example of a distributed workflow platform where processes are transferred to sites thereby eliminating the need of a centralised server [11].

[12] uses workflow partitioning in BPEL to structurally provide rules based on graph transformations. Transformations are based on rules that ensure that the system exposes the functional behaviour and the flow of the original workflow is observed. The partitioned BPEL processes is executed onto a network of mobile phones. Through this approach, they are able to produce an overall execution model that is equivalent to a centralized one, implemented using disconnected components and independent workflow engines. [2] presents the MARPLE architecture that enables the execution of processes on mobile devices. Through their system, they realize generic process management. The architecture meets the performance requirements of mobile scenarios to cope with specific requirements like broken connections and limited GUIs. They provide a set of requirements for the architecture to work which form part of the basis for the partitioning algorithms proposed. In their work, conceptual issues regarding the partitioning of processes are not provided.

[4] provides a Petri-net based approach for fragmenting workflows for distributed execution. The fragments created can migrate to servers where tasks are performed and new fragments are created. Through this approach a case can be executed on several servers in succession thus enabling the outsourcing of business functionalities.

[3] presents an approach for the distributed execution of workflows based on the fragmentation of high-level Petri-nets. The Petri-nets are fragmented horizontally, vertically and diagonally, and fulfil the necessary requirements for formal workflow behaviour like completeness, minimality and disjointedness. Conceptually, formal methods presented in [3, 4] for distributed workflow differ from our approach due to the problem addressed and deployment environment. Our approach seeks to move work to a client with a light-weight workflow engine for offline execution by combining two or more tasks while maintaining semantics of the original model.

Table 2. Evaluation

Inherent Patterns	**No. of tasks**	**No. of Partitions**	**Incorrect**	**Omissions**
Synchronization, Choice, Sequence, Simple merge, Parallel split	8	4	0	0
Synchronization (nested), Choice, Sequence(x2), Simple merge, Parallel split	9	3	0	0
Synchronization(x3), Simple choice(x6), sequence(x1), Simple merge(x7), Parallel split(2), iterations (x7), synchronising merge (x1)	20	13	0	0
Synchronization, Sequence(x2), Parallel split, Milestone	6	3	2	0
Multichoice, Multimerge	5	0	0	0

6 Discussion and Future Work

In this paper we present an approach to address the problem of partitioning workflows for offline execution in mobile environments and automatically discovering of groups of tasks (Partitions). We present a scenario for such a need and provide a set of requirements for partitioning workflows. Additionally an algorithm for discovering such partitions based on model unfolding and checking is presented. It is important to note that model unfolding represents the full reachability graph using partial orders that preserve the relations between transition occurrences [6]. All reachable markings are therefore represented in a Petri-net unfolding. This enables us to determine the relationships between occurrences thereby making the algorithm proposed sound. The approach proposed therefore considers the static and dynamic behaviour of workflow models when developing partitions.

The algorithm provided in this paper does function correctly for a range of patterns and therefore provides a viable approach to addressing the problem identified. Because an unfolding of a net produces the structural and behavioural aspects of the net, it can be extended to other process modelling languages to enable the discovery of Partitions. Pragmatism in unfolding is necessary to provide workable solutions. So rather than unfolding models to their basic Petri-net based representations, it would be prudent to extend the principles enshrined in this paper in implementing the algorithms on real process modelling environments. Future work will involve implementing this approach in a real workflow environment, developing a light weight workflow engine that is able to execute Partitions on a mobile phone and devising mechanisms for synchronising Partitions with the entire workflow.

7 References

1. Wakholi, P., Chen, W., Klungsøyr, J.: Workflow Support for Mobile Data Collection. Enterprise, Business-Process and Information Systems Modeling 299-313 (2011)
2. Pryss, R., Tiedeken, J., Kreher, U., Reichert, M.: Towards flexible process support on mobile devices. Information Systems Evolution 150-165 (2011)
3. Guth, V., Lenz, K., Oberweis, A.: Distributed workflow execution based on fragmentation of petri nets. pp. 114-125. Citeseer, (Year)
4. Tan, W., Fan, Y.: Dynamic workflow model fragmentation for distributed execution. Computers in Industry 58, 381-391 (2007)
5. http://clinicaltrials.gov/ct2/show/NCT00640263
6. Esparza, J., Heljanko, K.: Implementing LTL model checking with net unfoldings. Model Checking Software 37-56 (2001)
7. Akharware, N., Miee, M.: Pipe2: Platform independent petri net editor. (2005)
8. Lassen, K.B., van der Aalst, W.M.P.: Complexity metrics for Workflow nets. Information and Software Technology 51, 610-626 (2009)
9. Ouyang, C., La Rosa, M., ter Hofstede, A.H.M., Dumas, M., Shortland, K.: Toward web-scale workflows for film production. Internet Computing, IEEE 12, 53-61 (2008)
10. Bahrami, A., Wang, C., Yuan, J., Hunt, A.: The workflow based architecture for mobile information access in occasionally connected computing. pp. 406-413. IEEE, (Year)
11. Alonso, G., Gunthor, R., Kamath, M., Agwaral, D., Mohan, C.: Exotica/FMDC: A Workflow Management System for Mobile and Disconnected Clients. Distributed and Paralell databases 4, 229-247 (1996)
12. Baresi, L., Maurino, A., Modafferi, S.: Workflow partitioning in mobile information systems. Mobile information systems 93-106 (2005)