Research on well-formed business process modelling mechanism

Chen Kai[1], Xie Yi[2*]

[1] *College of technology, Lishui University, Lishui, China*

[2] *College of computer science & information engineering, Zhejiang Gongshang University, Hangzhou, China*

Received 1 March 2014, www.tsi.lv

Abstract

It is very important to ensure that the logic structure of business process model is correct before the model is implemented. Because traditional graphical process modelling methods lack efficiency mechanisms or rules to ensure correctness of the logical structure during business process modelling, they need additional methods to verify its correctness of the logic structure after the business process model is established. Therefore, the well-formed business process modelling mechanism is researched. The business process logic structure model is built firstly. Then the semantic and syntactic rules are presents for the correctness of business process logic structure model, and the algorithm is proposed to detect whether the model meets the rules. The modelling mechanism has been applied in our business process scheduling optimization system with integration of modelling and simulation, which shows its feasibility and effectiveness.

Keywords: business process modelling, modelling mechanism, well-formed model, model verification

1 Introduction

Process structure is the most important and primary aspect of a process model [1]. After the business process or workflow is deployed, detecting and repairing errors of process structure at run time is very expensive and time consuming. Therefore, a critical challenge in business process modelling lies in the design-time verification of business process models [2].

In order to adaptive to the varying needs of the organizations, business processes or workflows are often modelled as graphs [2, 3], in which individual activities within the process and their temporal constraints are shown as a series of nodes and edge (e.g. rectangles and arrows). Process modelling methods based on graphs (e.g., BPMN, activity diagrams of UML) are easy to use and master, the built process models using these methods are simple and intuitively understandable at a glance. However, work of general process modelling based on graphs includes arbitrariness and lacks strictness [4]. Moreover, process modelling methods based on graphs themselves do not provide some effective methods, rules or mechanism against errors in logical structure of process model. Thus, before utilizing the process model, it must be verified by following methods or techniques: Petri-net reduction techniques [5-8], graph reduction techniques [9-10], integer programming [2], simulation techniques [11], process logic [12], π calculus [13], semantic deduction [14], matrix calculus [15].

Petri-net reduction techniques, integer programming, simulation techniques, process logic, π calculus, semantic deduction, matrix calculus are indirect verification methods, and model transformations are required for verifying process model. After model transformations, process models lose their natural structures and are not easy to be understood at a glance [16]. Moreover, the performance of these methods drops off precipitously when the process model becomes more and more complicated with the node increasing, thus it is difficult to complete the verification of the large complicated business process models in distributed real-time interactive environment. Graph reduction techniques can directly detect the structural conflicts of graphical process model through a reduction process based on reduction rules. However, they can detect a limited set of process anomalies because the set of the developed reduction rules is not complete [12, 17]. They give no details about the causes of these conflicts, and, therefore, provide no help for further improvement [16].

In fact, there are two approaches to ensuring the correctness of process model. Besides to check it completely based on these approaches above, another is to build it correctly, which relies on strict grammatical rules that govern the composition of the various elements in the process model [18]. In reference [19] the concept of well-formed business process are proposed based on the idea of structured programming. In reference [19-22] the performance of well-formed business process is analysed and optimized. However, modelling mechanism of well-formed business process is not studied well yet, and there are lacks of formal, systematic description and supporting algorithm.

**Corresponding author* e-mail: xieyi@mail.zjgsu.edu.cn

To fill the gap, this study aims to formalize modelling mechanism of well-formed business process and design the effective algorithms of rule verifications, which can satisfy the requirements to verify the correctness of the large complicated business process in the real-time distributed interactive modelling environment. The proposed approaches have been applied to our developed Business Process Scheduling & Optimization System.

2 Business process logical structure model

2.1 FORMALIZING THE BUSINESS PROCESS LOGICAL STRUCTURE MODEL

The business process logical structure model (BPLSM) represents the control dependency or temporal constraint relationship between activities in business process, and can be formally defined below.

Definition 1 (Business process logical structure model): A BPLSM is defined by a 3-tuple $BPLSM = (A, C, L)$, which is characterized as follows:

(1) A is the set of activities.

(2) C is the set of connectors. $N = A \bigcup C$ is the set of nodes in the business process.

(3) $L \subseteq N \times N$ is the set of links. $l \in L$, $l = \langle n_1, n_2 \rangle$ represents the link from n_1 to n_2.

Apparently, a BPLSM is a directed graph $G_L = (N, L)$ which is composed of a set of nodes $N = A \bigcup C = \{n_i\}$ and a set of links $L = \{l_k\} \subseteq N \times N$.

Definition 2 (τ, η). $\tau : C \to \{And, Or\}$ is a function, which maps each connector onto this connector's logical type. $\eta : L \to (0,1]$ is a function, which maps each link onto the execution probability of this link.

Definition 4 (Directed path, elementary path). A directed path p from a node n_1 to a node n_k is a sequence $\langle n_1, n_2, \cdots, n_k \rangle$, so that $\langle n_i, n_{i+1} \rangle \in L$ for $1 \le i \le k-1$. p is elementary path iff for any two nodes n_i and n_j on p, $i \ne j \Rightarrow n_i \ne n_j$.

Definition 5 ($^\bullet$, $|\ |$). $|S|$ is the number of elements in the set S. For $n \in N$, $^\bullet n = \{m | (m,n) \in L\}$ is the set of input nodes, and $|^\bullet n|$ is the number of input nodes. $n^\bullet = \{m | (n,m) \in L\}$ is the set of output nodes, and $|n^\bullet|$ is the number of output nodes.

2.2 CONTROL DEPENDENCY RELATIONSHIPS BETWEEN ACTIVITIES IN BPLSM

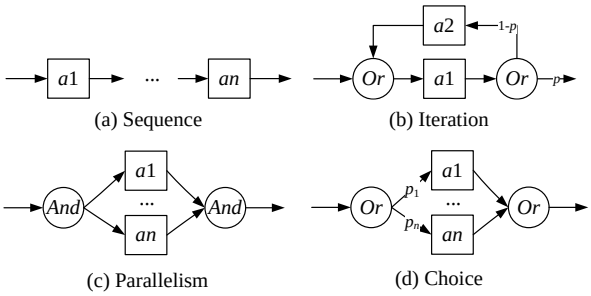
A common relationship between activities in business process model is logistic order that denotes a kind of partial order, called control dependency. The control dependencies such as sequence, And-Split, And-Join, Or-Split, Or-Join, and iteration, compose the model structures of business process. Four kinds of basic model structures shown as Figure 1 are sequence (SEQ), iteration (LOOP), parallelism (AND), and choice (OR). p is the probability of exit loop in Figure 1(b). p_i is the probability of selection branch in Figure 1(d), where $\sum_{i=1}^{n} p_i = 1$. The LOOP, AND, and OR basic model structures are called the non-sequential basic model structure. The semantics of these basic model structures excerpted from WfMC (1999) are listed below.

(1) Sequence (SEQ): Activities are executed in order under a single thread of execution, which means that the succeeding activity cannot start until the preceding activity is completed.

(2) Iteration (LOOP): A business process cycle involves the repetitive execution of one (or more) business process activities until a certain condition is satisfied.

(3) Parallelism (AND): A single thread of control splits into two or more threads that are executed in parallel within the business process, allowing multiple activities to be executed simultaneously. Once these parallel executing threads are all completed, they converge into a single common thread of control.

(4) Choice (OR): A single thread of control makes a decision upon which branch to take when encountered with multiple alternative business process branches. No synchronization is required because of no parallel activity execution.



(a) Sequence (b) Iteration (c) Parallelism (d) Choice

FIGURE 1 Four kinds of basic model structures

3 Modelling rules and its verification algorithm for well-defined BPLSM

The correctness of BPLSM includes two aspects: syntax and semantics.

3.1 SYNTAX RULES

A correct BPLSM must satisfy the following syntax rules:

(1) There is only one node $n_s \in N$, $|^\bullet n_s| = 0$, called as starting node. And there is only one node $n_e \in N$, $|n_e^\bullet| = 0$, called as ending node.

(2) $\forall n \in N$, $\exists p = \langle n_s, \cdots, n, \cdots n_e \rangle$. The node n must locate in a directed path from the starting node n_s to the

ending node n_e, namely, isolated nodes are not allowed.

(3) $\forall a \in A$, $\left|{}^{\bullet}a\right| \le 1 \wedge \left|a^{\bullet}\right| \le 1$; $\forall c \in C$, $(\left|{}^{\bullet}c\right| > 1 \wedge \left|c^{\bullet}\right| \le 1) \vee (\left|{}^{\bullet}c\right| \le 1 \wedge \left|c^{\bullet}\right| > 1)$. For every activity, the number of its preceding and succeeding node are respectively less than 2, namely, every activity cannot have multiple input links and multiple output links. Whereas, a connector is either a join node $(\left|{}^{\bullet}c\right| > 1 \wedge \left|c^{\bullet}\right| \le 1)$ or a split node $(\left|{}^{\bullet}c\right| \le 1 \wedge \left|c^{\bullet}\right| > 1)$.

(4) $\forall l \in L$, if $l \in \left\{\langle c,n\rangle \middle| \tau(c) = OR \wedge \left|c^{\bullet}\right| > 1\right\}$ then $\sum_{i=1}^{|c^{\bullet}|} \eta(l_i) = 1$, else $\eta(l) = 1$. Namely, for every split connector whose logical type is "*OR*", the sum of execution probabilities of its output links is equal to 1.

3.2 SEMANTIC RULES

A BPLSM, which satisfies the syntax rules may still have some semantic errors which lead to appear abnormality in execution of business process. To identify the semantic errors in BPLSM, the instance sub-graph and correctness criterion for BPLSM are defined firstly as follows:

Definition 6 (Instance sub-graph).An instance sub-graph represents a subset of activities that may be executed for a particular instance of a business process.

Definition 7 (Correctness criterion for BPLSM). A BPLSM is correct if all instance sub-graph can be executed without exception from starting node to ending node, namely for any one execution of a business process the ending node can be executed only once.

According to the correctness criterion for BPLSM, there are two semantic errors in BPLSM: deadlock and lack of synchronization. A deadlock conflict will be introduced if exclusive choice paths are joined with a synchronizer (a join node whose logical type is "*AND*"). A deadlock at synchronizer blocks the continuation of a business process path since one or more of the preceding transitions of the synchronizer are not triggered. A lack of synchronization will be introduced if concurrent paths are joined with a merge node (a join node whose logical type is "*OR*"). A lack of synchronization at a merge node results into unintentional multiple activation of nodes that follow the merge node.

It is very important to ensure that a business process model has correct logical structure before it is deployed. Because process modelling methods based on graphs themselves do not provide some effective methods, rules or mechanism against errors in logical structure of built BPM, based on the Jin Hyun Son's research [19], modelling rules are introduced and formalized for well-defined business process as follows:

Rule 1: An AND-Split control dependency should have its matching AND-Join control dependency, which forms a correct *AND* model structure.

The rule 1 is formalized as follows: for any $c_i \in C$, $\tau(c_i) = And \wedge \left|c_i^{\bullet}\right| > 1$, there is a $c_j \in C$, $\tau(c_j) = And \wedge \left|{}^{\bullet}c_j\right| = \left|c_i^{\bullet}\right|$. The connectors c_i and c_j satisfy with ① $\forall p = \langle c_i, \cdots, n_e\rangle, c_j \in p$; $\forall p = \langle n_s, \cdots, c_j\rangle$, $c_i \in p$; ② $\neg n \in N$, $\forall p = \langle c_i, \cdots, c_j\rangle$, $n \in p$. All nodes and links located in $p = \langle c_i, \cdots, c_j\rangle$ form a parallel model structure $PMS\{c_i, c_j\}$.

Rule 2: An OR-Split control dependency should have its matching OR-Join control dependency, which forms a correct *OR* or *LOOP* model structure.

The rule 2 is formalized as follows: for any $c_i \in C$, $\tau(c_i) = Or \wedge \left|c_i^{\bullet}\right| > 1$, there is a $c_j \in C$, $\tau(c_j) = Or \wedge \left|{}^{\bullet}c_j\right| = \left|c_i^{\bullet}\right|$. The connectors c_i and c_j must satisfy one of the following two cases:

Case 1: ① $\forall p = \langle c_i, \cdots, n_e\rangle, c_j \in p$; $\forall p = \langle n_s, \cdots, c_j\rangle$, $c_i \in p$; ② $\neg n \in N$, $\forall p = \langle c_i, \cdots, c_j\rangle$, $n \in p$. In this case, all nodes and links located in $p = \langle c_i, \cdots, c_j\rangle$ form a *OR* model structure $SMS\{c_i, c_j\}$.

Case 2: ① $\left|{}^{\bullet}c_j\right| = \left|c_i^{\bullet}\right| = 2$, $\exists p = \langle c_i, \cdots, c_j\rangle \wedge \exists p' = \langle c_j, \cdots, c_i\rangle$; ② $\forall n \in p$, $n \ne ci \wedge n \ne cj$, $\vec{n} \in p$, $\vec{n} \in p$ Where $\vec{n} \in {}^{\bullet}n$, $\vec{n} \in n^{\bullet}$, $p = \langle c_i, \cdots, c_j\rangle$ or $p = \langle c_j, \cdots, c_i\rangle$. In this case, all nodes and links located in $p = \langle c_i, \cdots, c_j\rangle$ and $p = \langle c_j, \cdots, c_i\rangle$ form a *LOOP* model structure $LMS\{c_i, c_j\}$.

Rule 3: an activity in the model structure can be replaced by a basic model structure to form a new model structure.

Rules 1 and 2 are match rules. Rule 3 is replacing or nesting rule.

Definition 8 (nested model structure)**.** A nested model structure is a model structure that contains non-sequential model structures.

Definition 9 (basic model structure)**.** A basic model structure is a model structure that does not contain any non-sequential model structures.

The simplest BPLSM which has only one activity node is shown as Figure 2. Based on the simplest model, a complex BPLSM can be usually formed by nesting four kinds of basic model structures (*SEQ*, *LOOP*, *AND*), and *OR*) according to the rules 1-3. Figure 3 shows an example to form a complex BPLSM.

Definition 10 (well-defined BPLSM)**.** A well-defined BPLSM is a BPLSM that is formed by nesting four kinds of basic model structures (*SEQ*, *LOOP*, *AND*), and *OR*) based on a single activity according to the rules 1-3.

Theorem 1. A well-defined BPLSM must be correct in logical structure.

Proof: firstly, there are no errors in syntax. Secondly, because the process of forming well-defined BPLSM is reversible, any well-defined BPLSM can be simplified to

a single active by the inverse process. This single active is both stating node and ending node, thus when business process can be executed once the ending node will be executed only once. According to definition 7 this BPLSM is correct.

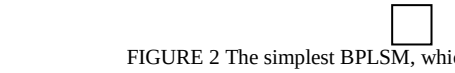


FIGURE 2 The simplest BPLSM, which has only one activity node

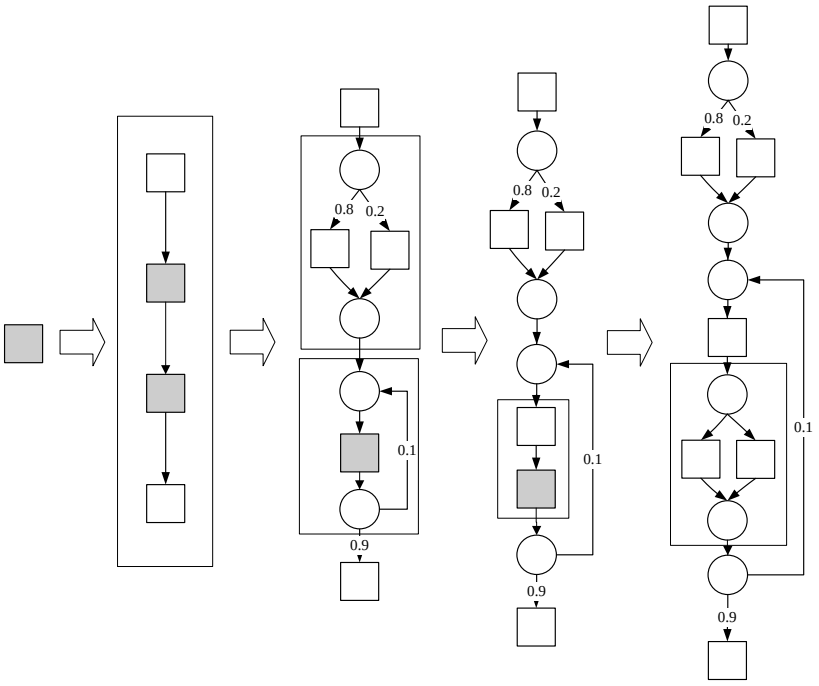


FIGURE 3 A process of forming a complex BPLSM from the simplest BPLSM

3.3 CHECKING ALGORITHM

It is not necessary to design the special algorithm for checking whether the BPLSM meet these simple syntax rules. However, the semantic rules are more complex than the syntax rules, the algorithm for checking whether the BPLSM meet the match and nesting rules are developed as follows:

Algorithm 1:

Step1: If the BPLSM is a basic sequence model structure then proceed to Step5, else proceed to Step2.

Step2: If $|JN| = |SN| > 0$, where JN is the set of join node and SN is the set of split node, then proceed to Step3, else proceed to Step6.

Step3: if the set of join node JN is not null then find out a join node jn_i and proceed to Step4, else proceed to Step5.

Step4: According to rules 1-3 find the split node sn_j from SN which matches with the join node jn_i. if the split node sn_j exist then jn_i, sn_j, and their matching type are recorded, and they are deleted from JN and SN respectively, proceed to Step3, else proceed to Step6.

Step5: The BPLSM meet the match and nesting rules, and output matching connectors. The algorithm terminates.

Step6: The BPLSM does not meet the match and nesting rules. The algorithm terminates.

To estimate the efficiency of this algorithm, "Big-O" is used to estimate its time complexity. In particular, the time complexity of the proposed algorithm will be $O(\frac{\lfloor |C|/2 \rfloor^2}{2})$ in the worst case. Thus, the developed algorithm can complete verification of well-defined BPLSM in very short time, and can be applied to verifying the large complicated business process models in distributed real-time interactive environment.

4 Conclusion

Because traditional graphical process modelling methods lack efficiency mechanisms or rules to ensure correctness of the logical structure during business process modelling, they need additional methods to verify its correctness of the logic structure after the business process model is established. Therefore, the well-formed business process modelling mechanism is researched. The semantic and syntactic rules are presents for the correctness of business process logic structure model, and the algorithm for checking if the model meets these rules is proposed. The

modelling mechanism has been applied in our business process scheduling optimization system with integration of modelling and simulation. Compared with conventional verification methods, for example, Petri-net reduction techniques, integer programming, process logic, π calculus, they have the following characteristics:

- The matching and nesting rules are simple and effective, and they are easy to use and master. And the correctness of the logical structure can be ensured by employing these rules.
- Model transformations is not required for checking if business process models satisfy the matching and nesting rules to verify their correctness.
- The algorithm for check rules is simple and its time complexity is small. It can avoid the sharp decline in the performance of algorithm with the growing process model and increasing node. Thus, it can be applied to verify the correctness of the large complicated business process in distributed real-time interactive environment.

Future studies can be done to enhance flexibility of modelling and presentation power of model, by incorporating some unstructured model structure into the well-formed business process modelling mechanism.

Acknowledgments

The authors would like to Ministry of Education of China, Zhejiang Provincial Natural Science Foundation of China, and Zhejiang Provincial Science Technology Department of china for financially supporting this research under the Grants Nos.11YJA630161, Y1111039 and 2014C31031.

References

[1] Touré F, Baïna K, Benali K 2008 An efficient algorithm for workflow graph structural verification. Proc. Of the Int. Conf. on Cooperative Information Systems *Lecture Notes in Computer Science* **5331** 392-408

[2] Eshuis Rik, Kumar Akhil 2010 An integer programming based approach for verification and diagnosis of workflows *Data & Knowledge Engineering* **69**(2010) 816-35

[3] Giouris A, Wallace M 2007 Graph Based Workflow Validation *Artificial Intelligence and Innovations* **247** 45-53

[4] Morimoto Shoichi 2008 A Survey of Formal Verification for Business Process Modeling. ICCS2008 *Lecture Notes in Computer Science* **5102** 514-22

[5] Van der Aalst W M P, Ter Hofstedev A H M 2000 Verification Of Workflow Task Structures: A Petri-net-based Approach *Information Systems* **25**(1) 43-69

[6] Naijing Hu, Liang Zhao, Hu Jinhua 2007 Verification of Evolution Rules on Workflow Net Based on Petri-Net *Journal of Chinese Computer Systems* **28**(6) 1076-9

[7] Jiantao Zhou, Meilin Shi, Xinming Ye 2005 A Method for Semantic Verification of Workflow Processes Based on Petri Net Reduction Technique *Journal of Software* **16**(7) 1242-51

[8] Liang Zhang, Shuzhen Yao 2007 Research on Correctness of Workflow Model Based on Petri Nets Reduction Techniques *Computer Engineering* **3**(9) 60-1

[9] Zhengjun Dang, Zhongjun Du 2011 Improved Algorithm combining graph-reduction and graph-search for workflow verification *Computer Engineering and Applications* **47**(4) 226-8

[10] Sadiq W, Orlwska M E 2000 Analysing process models using graph reduction techniques *Information Systems* **25**(2) 117-34

[11] Ke Ning Qing Li, Yuliu Chen 2005 Verification of IDEF3 process models *J Tsinghua Univ (Sci&Tech)* **45**(4) 540-4

[12] Bi H H, Zhao J L 2004 Process logic for verifying the correctness of business process models *Proc. of the 25th International Conference on Information Systems (2004ICIS)*.Washington, D.C., USA 91-100

[13] Abouzaid Faisal, Mullins John 2008 A Calculus for Generation, Verification and Refinement of BPEL Specifications *Electronic Notes in Theoretical Computer Science* **200**(2008) 43-65

[14] Hong Ling, Jiangbo Zhou, Zhengchuan Xu 2006 Semantic deduction-based workflow structure verification method *Computer Integrated Manufacturing Systems* **12**(6) 893-8

[15] Choi Y, Zhao J 2002 Matrix-based abstraction and verification for E-business processes *Proc. of the 1st Workshop on e-Business* Barcelona, Spain 154-65

[16] Liu R, Kumar A 2005 An analysis and taxonomy of unstructured workflows *Proc. of the 3rd Conference on Business Process Management (BPM 2005)* Lecture Notes in Computer Science **3649** 268-84

[17] Van der Aalst W M P, Hirnschall A, Verbeek H M W 2002 An Alternative Way to Analyse Workflow Graphs *Proc. of the 14th Int. Conf. on Advanced Information Systems Engineering (CAiSE'02)* Lecture Notes in Computer Science **2348** 535-52

[18] Sivaraman E, Kamath M 2002 On the use of Petri nets for business process modelling *11th Annual Industrial Engineering Research Conference*, Orlando, Florida

[19] Hyun Son Jin, Sun Kim Jung, Ho Kim Myoung 2005 Extracting the workflow critical path form the extended well-formed workflow schema *Journal of Computer and System Sciences* **70**(2005) 86-106

[20] Sheng Liu, Yushun Fan, Huiping Lin 2009 Dwelling time probability density distribution of instances in a workflow model. *Computer & Industrial Engineering* **57**(2009) 874-9

[21] Zuoxian Nie, Xinhua Jiang, Jiancheng Liu, Haiyan Yang 2009 Performance analysis for instances of generalized well-formed workflow *Computer Integrated Manufacturing Systems* **15**(12) 2424-31

[22] Liqin Tian, Chuang Lin 2003 Method for computing performance of a kind of workflow models nested by basic model *Acta Elect ronica Sinaca* **31**(12A) 2167-70

Authors

Kai Chen, born on April 26, 1968, Zhejiang China

Current position, grades: senior engineer at Lishui University, Lishui, China
University studies: M.S. degrees in Technology of Computer Application from Hangzhou Dianzi University in 2007
Scientific interest: system modelling and optimization, mechanical and electrical engineering and related control techniques

Yi Xie, born on July 23, 1975, Zhejiang China

Current position, grades: professor of Zhejiang Gongshang University, Hangzhou, China
University studies: M.S. and Ph.D. degrees in Mechanical Engineering from Zhejiang University in 2003, and 2011, respectively
Scientific interest: workflow, business process management, scheduling and optimization