International Journal of u- and e- Service, Science and Technology
Vol.9, No. 10 (2016), pp.387-398
http://dx.doi.org/10.14257/ijunesst.2016.9.10.35

The Functional Similarity Computing for Service Workflow using Temporal Behaviors Verification

Xiaoxian Yang[1,3], Tao Yu[1] and Huahu Xu[2,3]

[1]*School of Management, Shanghai University, Shanghai 200444, China*
[2]*School of Computer Engineering and Science, Shanghai University, 200444, China*
[3]*Shanghai Shang Da Hai Run Information System Co., Ltd, Shanghai,200444,China*
yangxiaoxian@shu.edu.cn

Abstract

Service workflow is the new solution to applications integration, which aims to use Web service as activity to make the business logic executable. It adopts abstract business process with complex control structures to describe interactive behaviors among e-commerce systems. In general, the new workflow should be verified before deploying it to running environment or even updating it into the workflow repository. To compare new workflow with selected workflows, the functional similarity is requested to compute the similarity degree which has been an important research in business process management. However, existing works focus on business process modeling and structural similarity computing, which lacks of the functional similarity computing that the application failure may be reoccurred due to incorrect temporal operations. In this paper, we concern on the functional similarity issue of service workflow in order to provide functions with newly-added values. The consistency verification is employed to check the functional similarity between service workflows. First, two workflows are selected as verification model and requirement model respectively. Then, it proposes automaton model to formalize behaviors and interactions of new workflow. Third, the verification property in the form of temporal logic is generated from requirement model, which is used to automatically check the functional consistency in a qualitative way. Fourth, the functional similarity degree is computed according to verification results for the quantitative evaluation. Furthermore, the functional similarity aggregation is introduced to workflow recommendation based on Pearson formula. Finally, the architecture of our method is discussed to guide engineering practices.

Keywords*: Workflow Modeling, The Functional Similarity Computing, Temporal Behaviors, Property Verification*

1. Introduction

With the development of workflow technologies, various applications and systems can be seamlessly integrated and interoperated to support complex business logics, accelerating their response to the frequent changes of custom demands and environment [1-3]. The aim of service workflow is to specify a loosely-coupled design and make business process executable [4]. It contributes to business agility, flexibility and availability, which uses Web services as business activities to enable heterogeneous system or even legacy system to be interacted with each other. Thus, more and more enterprises have adopted service workflows to develop e-commerce software system. It is considered as a promising approach to the next generation of service computing for intelligent business integration and cross-domain communication.

ISSN: 2005-4246 IJUNESST

Due to the limitation of communication protocols and running platforms, Web service is an effective solution to resources sharing and applications integration linking the boundary of enterprise over Internet. Enterprise can reduce cost and save time through using services to quickly respond to market's demands. In general, the service workflow first defines target processes in the form of workflow to describe business logics. Then, each activity is mapped with a Web service, which is a flexible functional entity since it can be dynamically allocated and replaced. However, the service workflow will display different functions and performances, as well as under different structures and service mapping strategies. To this problem, the workflow comparison is important that the new process should be compared with the existing process in order to discover which parts are different [5]. In adverse, it helps to confirm which scopes are overlapping portions between two workflows.

The functional similarity computing is the core problem of workflow comparison to business process management. At present, there are researches providing significant contributions to the workflow behavior analysis. But, little works were related to the quantitative evaluation of functional similarity [16-25]. In this paper, the functional similarity computing for service workflow mainly considers the consistency verification of temporal behaviors, which requests business logics of one workflow should be appeared in the other one. The temporal behavior of workflow is the feature of critical importance in business process verification. To this end, the temporal logic formula is employed to describe the functional behavior of service workflow. The other workflow is transformed into automaton model for automatic verification against a set of temporal logics formulae. Each verification result shows the assertion *yes*/*no* to judge whether temporal logic is satisfied or not. After performing formal verifications, the similarity degree is computed based on verification results. If two workflows are similar, it means that at least one temporal behavior is guaranteed in both workflows.

The rest of this paper is organized as follows. In Section 2, the motivation scenario is introduced to make the problem clearer. In section 3, it describes the verification process of the functional similarity computing, and then proposes methods to compute the similarity degree between two service workflows. In Section 4, it presents a brief introduction to architecture. In Section 5, it reviews related works. Finally, it concludes this research and discusses future works in Section 6.

2. Motivation

In this section, we discuss an example about the test paper generation system to show our motivation scenario. Suppose that test papers should be automatically generated for the final examination. The process refer to 8 services to handle corresponding events, including login action (LA), homework checking (HC), paper type checking (PTC), teaching affairs office audit (TAOA), supervisor audit (SA), undergraduate paper generation (UPG), graduate paper generation (GPG), and paper printing (PP). In order to get better classification, the test paper generation refers to two types, mainly the undergraduate test paper and the graduate test paper. As Figure 1(a), shown, the first event is login action, which is checked by Information Office. The follow-up event is homework checking, which is supported by the FTP based system. Then, the paper type checking is worked to assign different test paper to student. Teaching Affairs Office audits the test paper for undergraduate students, and Supervisor audits the test paper for graduate students. Finally, all test papers are printed by the paper printing service.

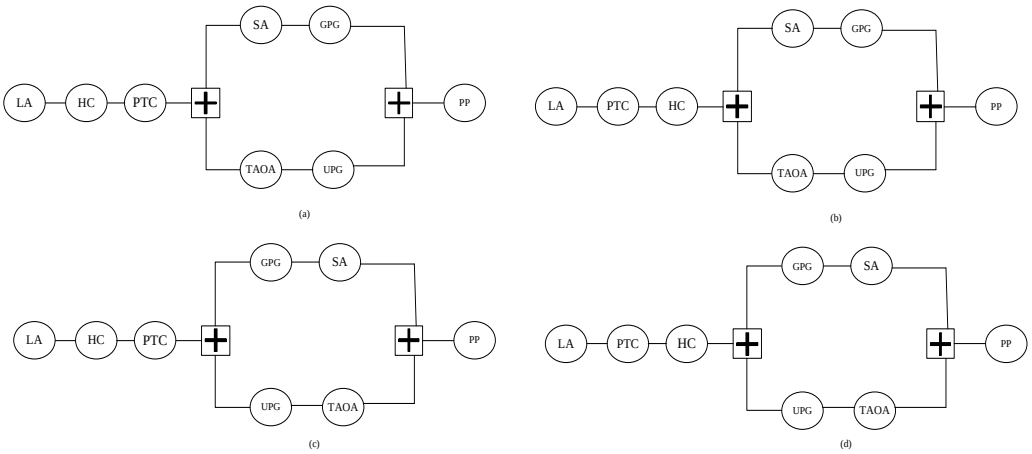

Figure 1. The Process of Test Paper Generation

Generally, the structure similarity is used to distinguish between two workflows using the graph structure. In Figure 1, four workflows can be transformed into an isomorphic graph structure with the same number of nodes and branches. However, the graph structure lacks of semantic information, especially temporal behaviors. In practices, these four workflows are different from each other. To illustrate the situation, we use ~ to denote the connection relation for temporal behavior, and ⊕ to denote the selection branch. Then, temporal behaviors of Figure 1 are as follows

1) For Figure 1(a), it is LA ~ HC ~ PTC ~ (SA ~ GPG ⊕ TAOA ~ UPG) ~ PP
2) For Figure 1(b), it is LA ~ PTC ~ HC ~ (SA ~ GPG ⊕ TAOA ~ UPG) ~ PP
3) For Figure 1(c), it is LA ~ HC ~ PTC ~ (GPG ~ SA ⊕ UPG ~ TAOA) ~ PP
4) For Figure 1(d), it is LA ~ PTC ~ HC ~ (GPG ~ SA ⊕ UPG ~ TAOA) ~ PP

Our goal is to compute the similarity degree for all feasible workflows. Thus, to compute the functional similarity of service workflow, the temporal behavior considers the relationship between business activities. Suppose Figure 1(a), is selected as source workflow. In Table 1, it shows the different temporal behaviors between workflows, comparing Figure 1 (b), Figure 1 (c), Figure 1 (d) with Figure 1 (a), respectively.

Table 1. The Different Between Workflows

ID	Target Workflow	Different Temporal Behaviors
1	LA ~ HC ~ PTC ~ (SA ~ GPG ⊕ TAOA ~ UPG) ~ PP	-
2	LA ~ PTC ~ HC ~ (SA ~ GPG ⊕ TAOA ~ UPG) ~ PP	LA ~ PTC; PTC ~ HC; HC ~ SA; HC ~ TAOA
3	LA ~ HC ~ PTC ~ (GPG ~ SA ⊕ UPG ~ TAOA) ~ PP	GPG ~ SA ; PTC ~ GPG; PTC ~ UPG; UPG ~ TAOA; SA~PP; TAOA~PP
4	LA ~ PTC ~ HC ~ (GPG ~ SA ⊕ UPG ~ TAOA) ~ PP	LA ~ PTC; PTC ~ HC; HC ~ GPG ; HC ~ UPG; GPG ~ SA ; UPG ~ TAOA; SA~PP; TAOA~PP

After calculating different temporal behaviors, the functional similarity degree can be figured out by the same behavior ratio for each workflow. From Table 1, the similarity degree for Figure 1(b), Figure 1(c), Figure 1(d), are 4/8, 2/8, and 0 respectively. In the worst case, the similarity degree between Figure 1(a), and Figure 1(d), is 0, even if they have same graph structure. Therefore, the structural similarity is independent of the functional similarity.

Then, we compute all similarity degrees for four workflows of Figure 1. According to Table 2, Figure 1(a), and Figure 1(b), are similar, while Figure 1(c), and Figure 1(d), are similar. In this case, the functional similarity based on temporal behaviors is more useful since it can discover more different between service workflows.

Table 2. The Similarity Degree Result

	1)	2)	3)	4)
1)	1	4/8	2/8	0
2)	4/8	1	0	2/8
3)	2/8	0	1	4/8
4)	0	2/8	4/8	1

The above motivation scenario shows that computing the functional similarity rather than the structural similarity can bring more benefits for workflow compaction. But it is not easy to solve the problem. First, it is infeasible to compare workflows in manual, especially when the searching space becomes large. It should consider the automatic mechanism to verify workflows. Second, the current approaches for the structural similarity computing cannot be extended to the functional similarity degree.

Although two workflows are similar at the structure level with the same graph, the functional similarity will return different result since that they may be dissimilarity without any overlapping temporal behaviors. To this point, we are motivated to consider the functional similarity computing of two service workflows, which can be used the service workflow selection and recommendation.

3. The Functional Similarity Computing

In this section, we define key concepts used in service workflow and discuss the functional similarity computing using the model checking technology.

3.1. Modeling Service Workflow

The formal verification is adopted and extended for computing the functional similarity degree of service workflow. The property verification is divided into three steps, including model, property and verification [6-7]. In general, the verification process is as follows.

1) It formalizes the target workflow into formal model M based on the model description language, such as Kripke [13], DTMC, and CTMC [8].

2) It generates the verification property of source workflow selected from workflow repository using temporal logic ϕ, such as CTL [12], LTL, TCTL, PCTL [9] and PTCTL[10].

3) It performs model checking, which aims to verify the satisfaction relation between model and formula that $M \models \phi$. This process is run automatically by the supporting tool.

The workflow provides process designers with support for constructing the new business process. In our work, the service workflow is formalized into Kripke structure which is one of automaton models.

Definition 1(Modeling Service Workflow). The service workflow is transformed into Kripke structure [13]. It is denoted as $WM=(S, I, AP, L, T)$, where,

1) S is a finite set of states, in which each state is mapped to an activity of workflow.

2) $I \in S$ is the starting state.

3) $T \subseteq S \times S$ is a finite set of transitions, by which each transition is mapped to business process of workflow.

4) AP is a set of atomic propositions, specifying the activity and its requirements.

5) $L : S \rightarrow 2^{AP}$ is a function that labels each state with the set of atomic propositions *true* in that state.

The workflow model shows an abstract automaton model where each transition can be executed since Web service supports the remote invocation and execution. The atomic proposition *AP* is built by the property of Web service. A transition in Kripke structure denotes change in the value of one or more states. For a state *s*, there is a set of succeeding states that $\{s' | \exists s \in S \bullet (s,s') \in T\}$. A path $\pi = <s_0, s_1, \ldots s_n>$ of an workflow model is a finite sequence that $1 \leq i \leq n$ and $(s_i, s_{i+1}) \in T$. Each path is called business process where each state has sequential relation. If there is no transition from *s* that $\{s' | \exists s \in S \bullet (s,s') \in T\} = \varnothing$, then a deadlock occurs. In technology, the self- transition will be added to the ending state that $(s,s) \in T$.

Different from the general workflow, *WM* model does not have control structures, such as the sequence branch, the concurrent branch, and the selection branch [11]. Thus, we consider structure mapping rules for the service workflow transformation.

Definition 2(Sequence Transformation). Given a sequence of activity $a_1 \sim a_2$, the sequence transformation mapping rule is to change sequence workflow into a single transition.

1) Generating two new states *s* and *s'* corresponds to a_1 and a_2 respectively.
2) Setting activity as atomic proposition of each state in the form of *truth*-value.
3) Adding a new transition (*s*,*s'*) to *T*.

Definition 3(Concurrent Transformation). The concurrent branch calls for multiple branches to execute in parallel and converge into a common point. We consider the AND-spilt and AND-join type, using symbol $\oplus$ to denote concurrent behaviors. Given a concurrent of activity $a_1 \sim (a_2 \oplus a_3) \sim a_4$, the concurrent transformation mapping rule is to change concurrent workflow into four transitions.

1) Generating four new states s_1, s_2, s_3 and s_4 corresponds to a_1, a_2, a_3, and a_4 respectively.
2) Setting activity as atomic proposition of each state in the form of *truth*-value.
3) Adding a new transition (s_1, s_2) to *T*, a new transition (s_1, s_3) to *T*, a new transition (s_2, s_3) to T, a new transition (s_3, s_4) to *T* respectively.

Definition 4(Selection Transformation). The selection branch calls for only one of branches can be executed. It is a mutual exclusion. We consider the XOR-spilt and XOR-join type, using symbol $\otimes$ to denote concurrent behaviors. Given a selection of activity $a_1 \sim (a_2 \otimes a_3) \sim a_4$, the selection transformation mapping rule is to change selection workflow into four transitions.

1) Generating four new states s_1, s_2, s_3 and s_4 corresponds to a_1, a_2, a_3, and a_4 respectively.
2) Setting activity as atomic proposition of each state in the form of *truth*-value.
3) Adding a new transition (s_1, s_2) to *T*, a new transition (s_1, s_3) to *T*, a new transition (s_2, s_3) to T, a new transition (s_3, s_4) to *T* respectively.

After obtaining the formal model of service workflow, the next step is to generate verification properties. The Computation Tree Logic (CTL) is introduced to describe temporal behaviors, which is a prominent branching temporal logic for specifying verification properties [12].

Definition 5(Computation Tree Logic). The state formulae of CTL are defined as follows. It is in the form of BNF (Backus-Naur Form), that is,

$$\Phi ::= true \mid p \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \Phi \rightarrow \Phi \mid \mathbf{E}\Psi \mid \mathbf{A}\Psi \quad (1)$$

where $p \in$AP, the quantifier **E** means 'along at least (there **E**xists) one path from the current state', the quantifier **A** means 'along **A**ll paths from the current state'.

The path formulae Ψ of CTL are defined as follows,

$$\Psi ::= \mathbf{X}\Phi \mid \Phi_1 \cup \Phi_2 \mid \mathbf{G}\Phi \mid \mathbf{F}\Phi \quad (2)$$

where Φ, $\Phi_1$and Φ_2 are state formulae, **X** is the *neXt-time* operator, **U** is the *Until* operator, **G** is the *Globally* operator and **F** is the *Future* operator.

The CTL formulae are used to describe the temporal behaviors, by which we can generate corresponding verification property for checking service workflow. The following formulae are presented to show the semantic description example of temporal behaviors.

1) $M \models \mathbf{AF}\varphi$ means that $\mathbf{AF}\varphi$ is valid in stats *s* if and only if φ holds globally along all path that start from *s*.

2) $M \models \mathbf{EG}\varphi$ means that $\mathbf{EG}\varphi$ is valid in stats *s* if and only if there exists some paths starting at *s* such as that for each state on this path the formula φ holds;

3) $M \models \mathbf{E}[\phi \mathbf{U} \varphi]$ means that $\mathbf{E}[\phi \mathbf{U} \varphi]$ is valid in stats *s* if and only if at least one path starting at *s* such as that for some states on this path the formula φ holds ***U**ntil* a state holds φ.

3.2. The Consistency Verification

The consistency verification of two workflows is to check whether they are consistent or not. We consider the new workflow as target model and the selected workflow as source of requirement model. It is worked under following principles.

Definition 6(Consistency Principles). Given two workflows *WM_1 and WM_2*, the consistency principles consider states and transitions.

1) All states of WM_1 are requested to be appeared in WM_2

2) All transitions of WM_1 are requested to be appeared in WM_2

3) WM_2 does not contain any transition which does not belong to WM_1.

Definition 6 indicates that WM_1 is similar to WM_2 when first two principles are satisfied. The last principle is a necessary condition which prevents WM_2 from implementing irrelevant business logics that are not requested by WM_1. According to Definition 6, the coverage criteria are employed to generate temporal logic formulae for the formal verification, mainly state coverage and transition coverage.

Definition 7(State Coverage). Suppose WM_1 is a formal model of the selected service workflow, which will be considered as requirement model to check the new service workflow. State coverage requires that each state node in WM_1 should be visited at least once. We use $\mathbf{EF}\varphi$ to generate temporal logic formulae.

$$\bigcup_{s \in S, a \in L(s)} EF(s \wedge a) \quad (3)$$

Definition 7 indicates that each state in WM_1 should have a corresponding state in WM_2. Thus, all states of WM_1 are extracted to generate temporal logic formulae.

Definition 8(Transition Coverage). Suppose WM_1 is a formal model of the selected service workflow, which will be considered as requirement model to check the new service workflow. Transition coverage requires that each transition relation in WM_1 should be visited at least once. We use $\mathbf{AG}\varphi$ to generate temporal logic formulae.

$$\bigcup_{s\in S,(s.s')\in T} AG(s \to EX(s')) \tag{4}$$

$$\bigcup_{\substack{s\in S,(s.s')\in T \\ a\in L(s),a'\in L(s')}} AG(a \to EX(a')) \tag{5}$$

Definition 8 indicates that each transition in WM_1 should have corresponding transition in WM_2. Thus, all transitions of WM_1 are extracted to generate temporal logic formulae.

3.3. Computing Similarity Degree

Given a set of temporal logics $TL=\{\phi_1, \phi_2, \ldots, \phi_n\}$, the functional similarity computing includes following three steps.

1) The first step is to check whether $TL=\varnothing$ or not. If it is empty, then the new service workflow is not similar to the selected service workflow. Otherwise, at least one property is satisfied by which the new workflow is similar to the selected workflow.

2) The second step is to compute the similarity degree. The satisfied property set is picked out from *TL* as $SP=\{\phi_i|\exists\phi_i\in TL\bullet WM|=\phi_i\}$. The similarity degree is $\gamma=\frac{|SP|}{|TL|}\in[0,1]$.

3) The third step is to make a decision by comparing the similarity degree with the user requirement.

Definition 9(The Results Vector for Similarity Verification). The verification results for temporal logics $TL=\{\phi_1, \phi_2, \ldots, \phi_n\}$ is in the form of $VR=\{vr_1, vr_2, \ldots, vr_n\}$. Each element vr_i is assigned to 0 or 1 according to the verification result. If the property ϕ_1 is verified as *true*, then vr_1 is marked as 1; Otherwise vr_1 is marked as 0.

$$vr_i=\begin{cases}1 & WM\models\phi_i \\ 0 & otherwise\end{cases} \tag{6}$$

Let {1,0,0,1,0,1} be verification results. It means that the 1st, 4th , and 6th of properties are satisfied. Let {0,0,0,0,0,0} be verification results. They are not similar to each other.

Definition 10(The Similarity Degree). Given verification results $VR=\{vr_1, vr_2, \ldots, vr_n\}$, the similarity degree is computed as follows

$$\gamma=\frac{\sum_{vr_i\in VR} vr_i}{|VR|} \tag{7}$$

Definition 11(The Average Similarity Degree). Due to workflow repository will have a lot of similar workflows, the average similarity degree γ_{avg} is worked as follow,

$$\gamma_{avg}=\frac{\sum_{WM_i\in WMS}\gamma_{WM_i}}{|WMS|} \tag{8}$$

In Definition 11, $WMS=\{\gamma_1,\gamma_2,\ldots,\gamma_n\}$ is a set of similarity degrees for current service workflow comparing with workflows selected from the workflow repository. Based on the average similarity degree, the specified threshold value τ requested by user is used to filter and recommend workflow. If $\gamma>\tau$, the new workflow can be stored into the workflow repository and mapped to the same functional domain. After that, similar workflows can be recommended to users when they retrieve workflows.

Definition 12(The Functional Similarity Aggregation). In order to aggregate similar workflows, the Pearson formula is used to the new workflow aggregation according to the similarity degree. Suppose the verification results matrix is as follow,

$$\begin{array}{c} \\ w_1 \\ w_2 \\ .... \\ w_{n-1} \\ w_n \end{array} \begin{array}{c} \begin{array}{cccccc} y_1 & y_2 & & y_{m-1} & y_m \end{array} \\ \left\{ \begin{array}{cccccc} vr_{1,1} & vr_{1,2} & . & vr_{1,m-1} & vr_{1,m} \\ vr_{2,1} & vr_{2,2} & . & vr_{2,m-1} & vr_{2,m} \\ . & . & . & . & . \\ vr_{n-1,1} & vr_{n-1,2} & & vr_{n-1,m-1} & vr_{n-1,m} \\ vr_{n,1} & vr_{n,2} & & vr_{n,m-1} & vr_{n,m} \end{array} \right\} \end{array}$$

The row w_i means the new workflow, and the column y_i means the selected workflow. The matrix contains all similarity degrees computed from the model checking process. Then, we use the following Pearson formula to get the correlation between service workflows.

$$c_{w,w'} = \frac{\sum_{i \in n}(vr_{w,i} - \overline{VR}_w)(vr_{w',i} - \overline{VR}_{w'})}{\sqrt{\sum_{i \in n}(vr_{w,i} - \overline{VR}_w)^2 \sum_{i \in n}(vr_{w',i} - \overline{VR}_{w'})^2}} \quad (9)$$

The $c_{w,w'}$ is a correlation value between w^{th} new workflow and w'^{th} new workflow. The element $vr_{w,i}$ is the similarity degree between w^{th} new workflow and i^{th} selected workflow. The $\overline{VR}_w$ is the average similarity degree in the verification results matrix. After calculation, the Pearson matrix shows a new relationship between service workflows. Only the most relevant workflows can be considered as similar workflows.

4. Architecture Introduction

The architecture to the functional similarity computing includes four modules, mainly the functional similarity verification (FSV), the workflow repository management (WRM), the basic configuration (BC), and the functional similarity computing (FSC). Its goal is to provide a convenient and efficient way to model and execute the collaborative service based on abstract workflow across different enterprises.

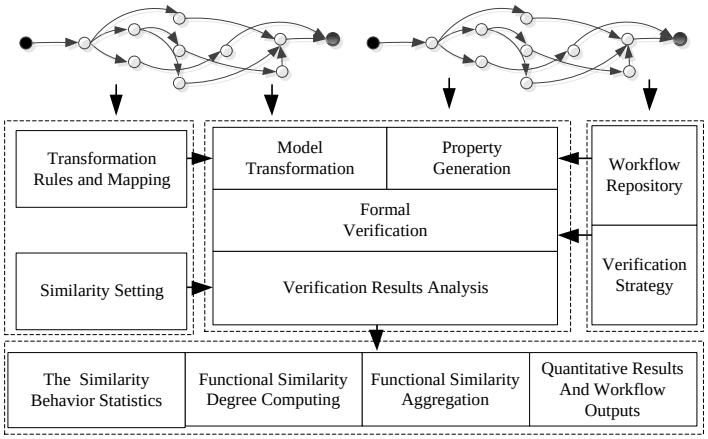

Figure 2. The Architecture of Functional Similarity Computing

The functional similarity verification (FSV) is to verify the functional similarity for service workflow. In our study, the high-level description tool BPEL4WS is used as workflow modeling tool, which is transformed into Kripke model, an automaton of formal model. When receiving a workflow, it begins to fetch workflow from the workflow repository, which is considered as requirement model. Then, coverage criteria are employed to generate the verification property. After that, the formal verification is

automatically performed by model checker NuSMV [13] to check each verification property. All these verification results will be analyzed for the functional similarity computing (FSC).

The workflow repository management (WRM) is in charge of workflow management. It includes two parts that the workflow repository and the verification strategies. The workflow repository recommends workflows to the functional similarity verification (FSV). The verification strategies support the on-the-fly strategy and the abstraction refinement strategy for improving the efficiency of formal verification.

The basic configuration (BC) is responsible for storing transformation rules and mapping methods since the control structure of workflow should be considered. Moreover, the similarity setting is used to filter unsatisfied workflow when the functional similarity degree is computed according to verification results.

The functional similarity computing (FSC) is the focus of this paper. First, it carries out the statistical result of similar behaviors based on verification results. Second, the functional similarity degree is computed in a quantitative way. Third, the functional similarity aggregation is to cluster analysis, by which the similarity setting is used as threshold value to aggregate similar workflows. Finally, quantitative results and workflows are returned as outputs to users.

5. Related Works

The similarity computing is an important research topic in workflow management, which has been widely used to the version update and the workflow recommendation. We give a review on the major techniques and researches that are most closely related to our works.

Some works focus on the similarity metric. Dong [14] studied the similarity metric algorithm based on the principal transition sequences (PTS), and proposed an improved scheme by defining the complete firing sequences to express model behavior. Dijkmana [15] presented three similarity metrics, mainly node matching similarity, structural similarity, and behavioral similarity. Kunze [16] presented an indexing approach based on metric trees to save comparison operations during searching, which is a hierarchical search structure.

Some works focus on the ontology description. Chen [17] proposed a two ontology using similarity computing algorithm, which considered both the taxonomy similarity and the feature similarity. The similarity between a detected object and a predefined object was calculated by combining the two similarities together. Krzywucki [18] proposed a similar workflow search algorithm based on the semantic type comparison, which involved reasoning based on myGrid ontologies. Schumacher [19] presented an approach towards a trace index based workflow similarity function, which speedups the calculation of comparison.

Some works focus on the behavioral similarity. WANG [20] proposed the behavioral similarity algorithm based on SSDT (Shortest Succession Distance between Tasks) matrix. Grigori [21] reduced the problem of behavioral matching to a graph matching problem. They developed a BPEL ranking platform that allowed to find in a service repository, a set of service candidates satisfying user requirements, and then, to rank these candidates using a behavioral-based similarity measure.

Some works focus on the structures similarity. Zha [22] considered process similarity measure focusing on the control flow structures. They proposed a label-free similarity measure between process models based on transition adjacent relations (TARs) in the context of workflow nets (WF-nets). Wombacher [23] researched the problem of service discovery. Different similarity measures facilitating structured workflows and higher level change operations were presented and evaluated based on a pilot of an empirical study. Wang [24] proposed a novel method to determine the degree of similarity between

process models. It was designed for labeled Petri nets which was a foundational theory for process modeling. Wang [25] proposed a grid workflow process design method using event-condition-action (ECA) rule, and proposed a new process similarity measure approach by revising existing clustering algorithm.

Different from above exiting woks, this paper focuses on the functional similarity computing considering the temporal logic property verification. The model checking is employed to perform the functional similarity verification. Then, the functional similarity degree based on verification results is computed in a quantitative way.

6. Conclusion

In this paper, we propose a novel approach to the functional similarity computing for service workflow using temporal logic property verification. First, the model checking is used to functional similarity verification where the new workflow is compared with the selected workflow. The core research is the temporal behavior at the functional similarity level. Then, the functional similarity degree is computed according to verification results. Third, considering the workflow number in the workflow repository, a set of the functional similarity degrees is analyzed to the functional similarity aggregation for the similar workflows recommendation. Furthermore, we introduce architecture of the proposed method to guide the system implementation. Our method contributes to assign appropriate services to workflow as an optimal solution.

However, this paper only considers the functional similarity focusing on the temporal logics of behavior. We will research non-functional behavior about how to handle data-ware workflow and user preferences to support the personalized workflow recommendation in the future.

Acknowledgment

This paper is supported by National Natural Science Foundation of China (NFSC) under Grant No. 61572306 and 61502294, Natural Science Foundation of Shanghai under Grant No.15ZR1415200, and CERNET Innovation Project under Grant No. NGII20150609.

References

[1] Y. Li, B. Cao, L. Xu, J. Yin, S. Deng, Y. Yin and Z. Wu, "An Efficient Recommendation Method for Improving Business Process Modeling", IEEE Transactions on Industrial Informatics, vol. 10, no. 1, **(2014)**, pp. 502-513.
[2] B. CAO, J. X. WANG, J. FAN and T.Y. DONG, "Mapping elements between process models based on Petri net", Ruan Jian Xue Bao/Journal of Software, (in Chinese), vol. 26, no. 3, **(2015)**, pp.474-490.
[3] J. Ma, K. Wang and L. Xu, "Modelling and analysis of workflow for lean supply chains", Enterprise Information Systems, vol. 5, no. 4, **(2011)**, pp. 423-447.
[4] S. Stein, T. R. Payne and N. R. Jennings, "Flexible provisioning of web service workflows", ACM Transactions on Internet Technology, vol. 9, no.1, (**2009)**, pp. 295-299.
[5] R. Dijkman, M. Dumas and L. G. Banuelos, "Graph matching algorithms for business process model similarity search", International Conference on the Business Process Management, **(2009)**, pp. 48-63.
[6] E. M. Clarke, E. A. Emerson and A. P. Sistla, "Automatic verification of Finite state concurrent systems using temporal logic specfications", The 10th Annual ACM Symposium on Principles of Programming Languages (POPL 83), **(1983)**, pp. 117-126.
[7] C. Baier and J. P. Katoen, "Principles of Model Checking. The MIT Press", Cambridge, Massachusetts,London, England, **(2007)**.
[8] M. Kwiatkowska, G. Norman, and D. Parker, "Advances and Challenges of Probabilistic Model Checking", The 48th Annual Allerton Conference on Communication, Control and Computing, (**2010)**, pp. 1691-1698.
[9] J. Berendsen, D. N. Jansen and F. W. Vaandrager, "Fortuna: Model checking priced probabilistic timed automata", International Conference on the Seventh International Conference on the Quantitative Evaluation of Systems (QEST'10), **(2010)**, pp. 273-281.
[10] H. Hansson and B. Jonsson, "A logic for reasoning about time and reliability. Formal aspects of computing", vol. 6, no. 5, **(1994)**, pp. 512-535.

[11] W. M. P. V. Der Aalst, A. H. Ter Hofstede, B. Kiepuszewski and A. P. Barros, "Workflow patterns", Distributed and Parallel Databases, vol. 14, no. 1, **(2003)**, pp. 5-51.
[12] E. M. Clarke and E. A. Emerson, "Design and synthesis of synchronization skeletons using branching time temporal logic", Workshop on Logic of Programs, **(1981)**, pp. 52-71.
[13] K. L. McMillan, "The SMV System for SMV version 2.5.4", http://www.cs.cmu.edu/~modelcheck/smv/-smvmanual.ps, **(2006)** October.
[14] Z. H. DONG, L. J. WE and H. W. HUANG, "Behavioral similarity algorithm for process models based on firing sequence collection", Ruan Jian Xue Bao/Journal of Software, (in Chinese), vol. 26, no. 3, **(2015)**, pp. 449-459.
[15] R. Dijkmana, M. Dumasb, B. van Dongenc, R. Käärikb and J. Mendlingd, "Cover image Similarity of business process models: Metrics and evaluation", Information Systems, vol. 36, no. 2, **(2011),** pp. 498-516.
[16] M. Kunze and M. Weske, "Metric Trees for Efficient Similarity Search in Large Process Model Repositories", Workshop on Business Process Management, **(2011)**, pp. 535-546.
[17] Y. Chen, J. Wang, Z. Cheng, L. Jing and Y. Zhou, "An algorithm to compute similarity between danger objects based on ontology for danger-aware systems", 2010 2nd International Symposium on Aware Computing (ISAC), **(2010)**, pp. 128-135.
[18] M. Krzywucki and S. Polak, "WORKFLOW SIMILARITY ANALYSIS" Computing & Informatics, vol. 30, no. 4, **(2011)**, pp. 773-791.
[19] P. Schumacher and M. Minor, "Towards a Trace Index Based Workflow Similarity Function. KI 2014: Advances in Artificial Intelligence, **(2014)**, pp. 225-230.
[20] S. H. WANG, L. J. WEN, D. S. WEI, J. M. WANG and Z. Q. YAN, "SSDT matrix-based behavioral similarity algorithm for process models", Computer Integrated Manufacturing Systems, vol. 19, no. 8, **(2013)**, pp. 1822-1831.
[21] D. Grigori, J. C. Corrales, M. Bouzeghoub and A. Gater, "Ranking BPEL Processes for Service Discovery", IEEE Transactions on Services Computing, vol. 3, no. 3, **(2010)**, pp. 178-192.
[22] H. Zha, J. Wang, L. Wen, C.Wang, "A label-free similarity measure between workflow nets", 2009 IEEE Asia-Pacific Services Computing Conference APSCC, **(2009)**, pp. 463-469.
[23] A. Wombacher and C. Li, "Alternative Approaches for Workflow Similarity", 2010 IEEE International Conference on Services Computing (SCC), **(2010)**, pp. 337-345.
[24] J. M. Wang, T. F He, L. J. Wen, N. H. Wu, A. H. M. ter Hofstede and J. W. Su, "A behavioral similarity measure between labeled Petri nets based on principal transition sequences", International Conference on the Move to Meaningful Internet Systems (OTM 2010), **(2010)**, pp. 394-401.
[25] Y. Wang, M. Li, J. Cao, X. Lin and F. Tang, "Workflow Similarity Measure for Process Clustering in Grid", Fourth International Conference on Fuzzy Systems and Knowledge Discovery, vol. 4, **(2007)**, pp. 629-635.

Authors

Xiaoxian Yang, She is a Ph.D. candidate in Management Science and Engineering, School of Management, Shanghai University, China. Her research interests include Service Computing, Business Process Management and Formal Methods.

Tao Yu, He is a full professor and Ph.D. supervisor of the School of Management, Shanghai University, China. His research interests include Computer Integrated Manufacturing System and Business Process Management.

Huahu Xu, He is a full professor and Ph.D. supervisor of the School of Computer Engineering and Science, Shanghai University, China. His research interests include Multimedia Technology, Computer Integrated Manufacturing System, and Software Engineering.