# An evolutionary model for dynamic and adaptative service composition in distributed environment

Jiawei Lu, Huan Zhou, Jun Xu, Gang Xiao
*School of Computer Science and Technology*
*Zhejiang University of Technology, Hangzhou, China*
*Email: {viivan, zhouhuan, xujun, xg}@zjut.edu.cn*

Haibo Pan
*Whzhen avenue Technology Co., Ltd*
*Hangzhou, China*
*Email: 345383630@qq.com*

***Abstract*—Service composition is an important mean for integrating the individual Web services to create new value-added systems that satisfy complex requirements. Therefore, how to effectively analyze different types of services and find out the matching similarity between services to efficiently substitute failed services in a distributed and dynamic environment becomes crucial to service composition. In this paper, we propose a novel approach based on a data cell evolution model (DCEM) to support the dynamic adaptation of service compositions. The model combines data service information and biological cell behavior analysis to encapsulate data services into data cells. In order to reach optimum adaptations, we analyzed the static and dynamic structure of data cells based on bigraph theory to guarantee the consistency of service evolution. To evaluate the proposed approach, a series of simulation experiments and comparisons are conducted to demonstrate the effectiveness of service composition.**

***Keywords- Service Composition; Data Service; Bigraph; Data Cell; Service Evolution.***

## I. INTRODUCTION

Data as a Service (DaaS) is a new cloud computing service model that provides consumers on demand with data through different protocols on the Internet in a timely and low cost manner. However, as the function of a single Web service is simple and limited, it is difficult to meet the various requirements in a complex network environment. How to effectively model evolution of service composition and analyze its service behavior has become an issue that existing research must deal with.

Due to Web service with multi-source, heterogeneous, autonomous and dynamic characteristics, the evolution of service composition is different from traditional software evolution and faces more serious challenges. Many scholars have researched in this field through formal methods [1], semantic [2] or combinatorial models [3]. However, the above studies mainly focus on abstract behaviors and semantic analysis of services in specific field, without considering the dynamic contexts. Consequently, they easily lead to performance degradation and composition failure when the environment changes.

Biological cell is a precise structural, functional, and evolutionary unit, which structure can change dynamically with the environment during growth, differentiation and physiological process. Comparing the dynamic behavior of service composition with biological cell, they have similarities in some aspects. It is possible to combine data service with biological cell to analyze the evolution of service composition [4,5].

In this paper, we encapsulate data services into data cells and analyze the dynamic behavior at the cell level. The data cell, like biological cell has a strong hierarchical structure. Thus, a formal method is needed to effectively explain the static and dynamic information of data cells and reflect the important characteristics such as functions and location interconnectivity of services. Bigraph [6] is a graphical formalization theory tool. It has more extensive applications in formal modeling and consistent evolutionary analysis [7].

In order to reason about the evolution of service composition better, we propose a data cell evolution model (DCEM) to increase the flexibility of service in Web system. The main contributions of this paper can be summarized as follows:

- We use bigraph theory to encapsulate data services into data cells, and construct evolution model for data cell to describe different information of service such as service name, service quality, service context, and so on.
- Because the composition processes may be canceled, or services may be moved or withdrawn, it is necessary to recombine it to provide a more powerful service, so that the service dynamic behavior based on bigraphical reactive system is analyzed. Meanwhile, we propose a self-healing algorithm which is used to dynamically adjust available service to substitute the failed ones.

## II. DATA CELL MODELING BASED ON BIGRAPH THEORY

### A. The Bigraph Theory

Bigraph is proposed by Milner and other scholars in 2001 to emphasize the location and connection of computing (physical or virtual) [6]. Bigraph is a 2-tuple $B = <B^P, B^L>$. $B^P$ is the place graph and $B^L$ is the link graph. The place graph is used to represent the location of nodes, which are nested with each other in bigraph. The link graph ignores the nested relationship and only indicates the connection between nodes. Fig. 1 shows a bigraphical structure.

DOI reference number: 10.18293/SEKE2019-120

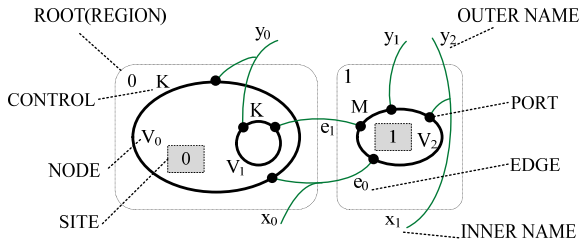



Fig. 1. Elements of bigraph

## B. Data Cell Modeling

Based on the bigraph theory, we constructed DCEM that maps the structures and message interaction of services to bigraphs, so as to formalize the services and their compositions by process calculus. DCEM mainly consists of two layers: data cell and data cell cluster. The form is defined as follows:

**Definition 1 (Data Cell).** A bigraph definition of data cell is a 5-tuple $DC=<S, E, Ctrl, C^P, C^L>: <m, X> \rightarrow <n, Y>$, where:

- $S$ is a limited set of services in a data cell, $\forall s \in S$ is called a data service;
- $E$ is a set of finite edges, $\forall e \in E$ is called a connecting edge;
- $Ctrl: S \rightarrow C$ is a mapping relation between services and service controls;
- $C^P$ is the place graph to represent the location of services and $C^L$ is the link graph to represents service dependencies;
- The inner interface $<m, X>$ indicates that the bigraph has $m$ sites and a set of inner names $X$. The outer interface $<n, Y>$ indicates that the bigraph has $n$ regions and a set of outer names $Y$.

**Definition 2 (Service Control).** A service control is a 5-tuple $C=<CN, CT, P, CL, U>$, where:

- $CN$ is a control name of service. $CT$ is used to specify the type of this service, whether atomic or composite;
- $P$ is a limited set of ports, which describes the inputs and outputs of service, $\forall p \in P$ is called a service port;
- $CL=<DL, CN>$ is the dependency status of current service, including $DL$ which is the dependent level with $CN$ from other service;
- U is a probability value which represent the service reliability.

**Definition 3 (Bigraphical Reactive System).** A bigraphical reactive system for data cell is a 3-tuple $BS=<BC, R, BC'>: BC \rightarrow BC'$, where:

- $BC$ is the reactants and $BC'$ is the products, which are corresponding to data cells with the bigraphical structures;
- $R$ is a set of reaction rules and specifies the reaction process from $BC$ to $BC'$.

As examples depicted in Fig. 2, the bigraph definition of data cell aims to represent the structural relationship and data characteristics in services. The core element correspondence between data cell and bigraphical structure is also shown in Table I.

TABLE I. DATA CELL STRUCTURE DEFINITION

| Element in data cell | Element in bigraph | The example |
|---|---|---|
| *DC* | Root | *0, 1…* |
| *m* | Site | 1 2 … |
| *S* | Node | $s_1, s_2, s_3$… |
| *CL* | Edge | $e_0, e_1, e_2$ |
| *C* | Node control | $C_1, C_2, C_3$… |
| *pC* | Node Ports | |

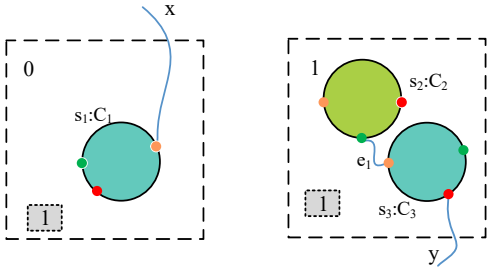



Fig. 2. Bigraph form of data cells

In practice, in order to meet the increasingly complicated requirements of users, it is necessary to select appropriate services from the network and combine them according to certain business rules to construct a scalable, loosely coupled combination. The data cell cluster is based on four kinds of structures (sequence, conditional, parallel, and loop) in service composition and combines a plurality of data cells with bigraph operations. The relevant forms of data cell cluster proposed in this paper are as follows:

**Definition 4 (Data Cell Cluster).** A bigraph definition of data cell cluster is a 3-tuple $DCC=<DCS, CS, LinkS>$, where:

- $DCS$ is a limited set of data cells;
- $CS$ is a limited set of composite structures in data cells;
- $LinkS$ is a limited set of link ports in the cluster, $\forall Link \in LinkS$ is called a connection between two ports.

The term language [13] is the basis for the formal specification and verification of dynamic evolution in bigraph. Fig. 3 shows the structure of $DCC$ based on different workflows. Taking the parallel structure (Case 3) as an example, the cluster has three data cells $DC_0$, $DC_1$, and $DC_2$. The place graph indicates the positional relationship of services and other information (e.g., the number and distributivity of the cells). The link graph shows the dependency relationship of the services.

To adapt the dynamic environments to complex requirements, the data interaction between cells is constantly changing, forming new cell clusters or modifying the original cell cluster structure. Ensuring the structural integrity of data cells during this interactive process, while increasing the effectiveness of service composition, requires serious consideration. A checking technique for verifying the data flow of the process model and re-adjusting the model

according to the feedback has been proposed [14]. However, this process model has a large detection granularity and may easily detect distortion. We present the bigraph matching algorithm (Algorithm 1) to evolve data cells according to a bigraphical reactive system (see definition 3). During matching, the constraints of *R* in the bigraph are dynamically determined by the context and requirements. A reaction rule in *R* specifies the reaction process, and can take any number of parameters. Finally, a new bigraph is generated when the data cells match successfully.

The algorithm contains two phases and takes into account time, QoS, and service context information constraints. In the first phase, we take an initial bigraph *BC* and a set of reaction rules *R*. For each reaction rule *r* in *R*, the method *isMatch(BC,r)* is called to determine whether the elements in the bigraph can be matched. In the second phase, if *r* is matched and the constraint is satisfied, the matching part in the bigraph will be replaced by products in *r*. *isMatch(BC,r)* is a recursive method that is iteratively executed until the last node in the bigraph has been checked.

**Algorithm1** BigraphMatch

**Input:** bigraph *BC*(an initial bigraph), a set of reaction rules *R*
**Output:** a new bigraph *BC′*

```
1: if R == Null then
2:   return BC
3: else
4:   for each reaction rule r in R do
5:     flag = isMatch(BC, r)
6:     if(flag == TRUE&&timeConstraints == true) then
     // If the match is successful and satisfies the time constraint, the
reaction proceeded
7:     BC′ = BC′∪ {BC|the matching part in BC with reaction in r}
8:     end if
9:   end for
10:   return BC′
11: end if
Procedure isMatch(BC, r)
Input: bigraph BC, term r in R
Output: a flag to indicate whether the match is found
1: for each service s in BC do
2:   if(r contains s) then
3:     if(s.C.CN != r.C.CN || s.C.CT != r.C.CT || s.C.P != r.C.P)  then
     // Service control matching
4:       continue
5:     else
6:       for each port p in C do   // Service port matching
7:         if(s.p.pI != r.p.pI || s.p.pN != r.p.pN || s.p.pT != r.p.pT ||
s.p.pC != r.p.pC) then
8:           continue
9:         else
10:          return True
11:        end if
12:      end for
13:    end if
14:  else
15:    continue
16:  end if
17: end for
18: return False
```

## III. EVOLUTIONARY ANALYSIS OF SERVICE DYNAMIC BEHAVIOR

In actual use, service composition may face situations such as service failure and service composition disorder. These cause data cell variation, thus losing the original functional properties and structural stability.

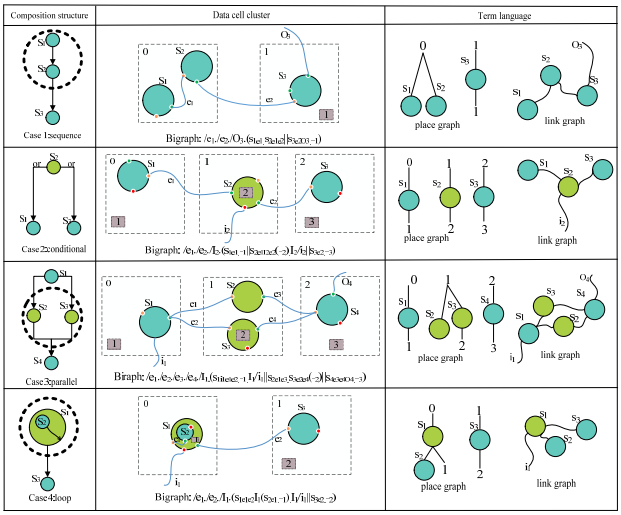



Fig. 3. The basic structures of data cell cluster

In our work, DCEM periodically tests the service availability through a data cell self-healing algorithm (Algorithm 2), allowing the structural variation of the cells to be fixed. This enables the service composition to restore the expected functions, and ultimately achieves the effect of self-repair to improve the service adaptability. The basic process of Algorithm 2 is as follows:

**Algorithm2** Data Cell Self-Healing

**Input:** bigraph *BC*(an initial bigraph)
**Output:** a new bigraph *BC′*

```
1: for each service s in BC do
2:   flag = isInvalid(s) //indicate whether the services is invalid
3:     if (flag == true) then
4:       a broken bigraph BC* = BigraphReplace(BC, s)
          //Choose the most reliable service MDCC
5:       MDCC = Max(CR(DCC))
6:       generate new reaction rules R = createR(s, MDCC)
        // use algorithm 1 to ensure the structural integrity of DC
7:       BC′ = BigraphMatch(BC*, R)
8:     end if
9: end for
```

- Call method *isInvalid(s)* at a specified time interval to check whether the service in data cell is fail.
- If the service fails then use Algorithm 3 to adjust the bigraphical structure, such as delete nodes and control belong to the failed service. Otherwise adjust the time interval to continue testing.
- Select the most reliable service from the similar service clustering to replace the fail one. There are many clustering algorithms; we mainly use the tag clustering algorithm in reference [15] and execute the aggregation process to construct the similar service clustering. Then the chosen service is considered as a reaction in rules *R*.
- Based on the previous steps and the bigraphical reactive system, generate the appropriate reaction rules *R*, and call Algorithm 1 to verify the rationality of reaction. Finally, a new bigraph is generated with a new reliable composition.

When service is detected as failed, Algorithm 3 needs to find out the failed service in the data cell. For given bigraphical information and a failed service *s*, we traverse each node in bigraph. If there is a surjection relationship

between the node and s, the corresponding structure is deleted and a broken bigraph is given. The specific algorithm is described as follows:

**Algorithm3** BigraphReplace

**Input:** bigraph *BC*(an initial bigraph), *s*(a failed service)
**Output:** a broken bigraph *BC**

```
1: for each service s′ in BC do
     //Indicate whether the service is match
2:   flag = node_conMatch(s′, s)
3:   if (flag == true) then
       //Delete structures belong to s′
4:     BC* = deleteBigraph(BC, s′)
5:     if (s′.C.CL.DL != single ) then
         // Delete dependency belong to s′
6:       BC* = deleteDependent(BC*, s′)
7:     end if
8:   end if
9: end for
```

## IV. CASE STUDIES AND VERIFICATION

To illustrate the effectiveness of DCEM in service composition, we introduce a composite service that supports online book shopping at Orange Country Bookstore (OCB) [16] (depicted in Fig. 4).

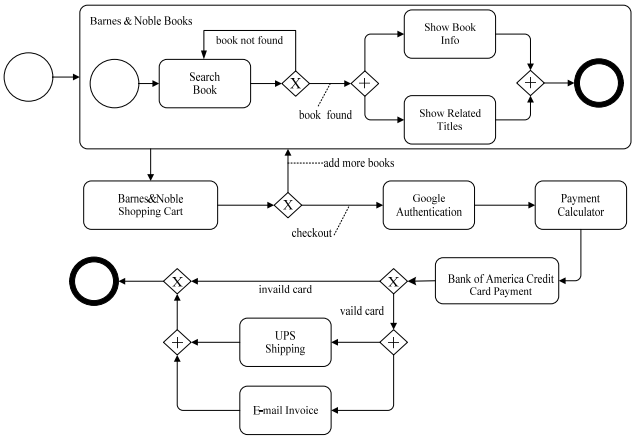



Fig. 4. Composite service for online book shopping

The business process in OCB includes (1) Look for books, including *Search Book*, *Show Book Info*, *Show Related Titles*, which are all part of the *Barnes & Noble Books* composite service. (2) Add books to shopping cart in a loop, e.g., *Barnes & Noble Shopping Cart.* (3) Authentication and payment at checkout, such as *Google Authentication* or *Payment Calculator*. (4) Email and invoice service, such as *invalid Card*, *UPS Shipping Web*, and *E-mail Invoice*.

The service composition created by this instance may change the contextual events during the actual operation, resulting in service failure. In addition, the system requirements state that the service composition must maintain a main workflow after the evolutionary adjustment (for example, always ensuring that books are first searched and then added to the cart).

Here, we fully consider the evolution possibility of each service by analyzing its context. First, we construct the data cells based on the different business processes, as listed in Table II, and then further evolve the data cells into cell clusters according to functional attributes and requirements. Table III lists the data cell clusters related to the main workflow of the shopping cart, searching for books, and payment in the OCB website.

When *Barnes & Noble Books* is unavailable, causing the functionality of $s_4$ to go missing (see Fig. 5), the system detects the failed service and traverses the data cell cluster according to Algorithm 3 to alter the structure. It then finds the substitute service *Amazon Books* ($DCC_{AB}$) from the similar service clustering via Algorithm 2 to repair the structure.

TABLE II. DATA CELL MODELING

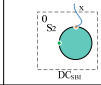
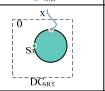
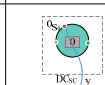
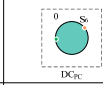
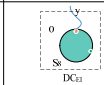
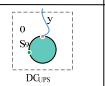
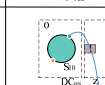
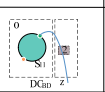
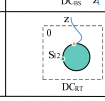
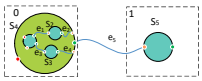
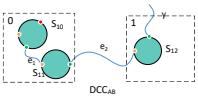
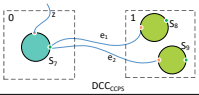
| Data Service | Data Cells | Term Language | Data Service | Data Cells | Term Language |
|---|---|---|---|---|---|
| *Search Book* |  | $/n.(S_{1n}\|\neg_1 n'x)$ | *Show Book Info* |  | $S_{2x}$ |
| *Show Related Titles* |  | $S_{3x}$ | *Shopping Cart* |  | $/n.(S_{4nt}\|\neg_1 n'y)$ |
| *Google Authentication* |  | $S_5$ | *Payment Calculator* |  | $S_6$ |
| *Bank of America Credit Card Payment* |  | $/n.(S_{7n}\|\neg_1 n'y)$ | *E-mail Invoice* |  | $S_{8y}$ |
| *UPS shipping* |  | $S_{9y}$ | *Book Searching* |  | $/k.(S_{10k}\|\neg_1 k'z)$ |
| *Book Description* |  | $/k.(S_{11k}\|\neg_2 k'z)$ | *Related Titles* |  | $S_{12z}$ |

TABLE III. DATA CELL CLUSTER MODELING

| Data Services | Data Cell Clusters | Term Language |
|---|---|---|
| *Barnes & Noble Books Shopping Cart* |  | $/e_1./e_2./e_3./e_4./e_5.(s_{4e2e4e5}(s_{1e1e2}\|s_{2e1e3}\|s_{3e2e4})\|\|s_{5e5})$ |
| *Amazon Books* |  | $/e_1./e_2.(s_{10e1}\|s_{11e1e2}\|\|s_{12e2y})$ |
| *Credit Card Payment shopping* |  | $/e_1./e_2.(s_{7e1e2z}\|\|s_{8e1}\|s_{9e2})$ |

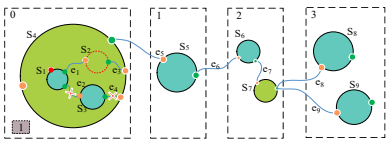


Fig. 5. Data cell cluster on OCB when *Barnes & Noble Books* has failed

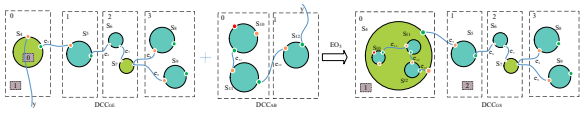


Fig. 6. Self-healing process in data cell cluster with *Amazon Books*

Finally, as shown in Fig. 6, $DCC_{AB}$ is absorbed by the data cell cluster, retaining an unchanged workflow order such as a looping structure with *Book Searching*, *Book Description*, *Related Titles*, and *Shopping Cart*. $s_{10}$, $s_{11}$, and $s_{12}$ are added in the site of $s_4$. After that, the output port of $s_{12}$ is connected to the input port of $s_4$ by $e_{13}$. Tables IV summarize the term language operational sets as $EO_1$ to formalize this process of evolution.

TABLE IV. TERM LANGUAGE OPERATIONAL SET FROM $EO_1$

| ***DC*** | ***EO₁*** |
|---|---|
| *DCC GE* | $EL_{rule}$: $/n. \rightarrow /e_{13}$. |
| | *Site*: $\text{-}_0 \rightarrow s_{10e_{11}}\|s_{11e_{11}e_{12}}\|s_{12e_{12}e_{13}}$ |
| | $x/y$: $n/y \rightarrow \Phi$ |
| | $\prod_{i=1}^{4} \text{DC}_{i\ ar(f)}$: $(s_{4e_5}n(\text{-}_0)\|\text{-}_1\|n/y\|\|s_{5e_5e_6}\|\|s_{6e_6e_7}\|s_{7e_7e_8e_9}\|\|s_{8e_8}\|s_{9e_9}) \rightarrow (s_{4e_{13}e_5}(s_{10e_{11}}\|s_{11e_{11}e_{12}}\|s_{12e_{12}e_{13}})\|\text{-}_1\|\| s_{5e_5e_6}\|\text{-}_2\|\|s_{6e_6e_7}\|s_{7e_7e_8e_9}\|\|s_{8e_8}\|s_{9e_9})$ |
| *DCC AB* | $EL_{rule}$: $/y. \rightarrow /e_{13}$. |
| | $U\|\|V$: $s_{10e_{11}}\|s_{11e_{11}e_{12}}\|\|s_{12e_{12}y} \rightarrow s_{4e_{13}e_5}(s_{10e_{11}}\|s_{11e_{11}e_{12}}\|s_{12e_{12}e_{13}})$ |

These actions express how to reorganize elements in the composition model to re-select suitable data cells from the same cluster when the *Barnes & Noble Books* composite service fails. The failed units are replaced by *Amazon Books* and *Related Titles* (according to the business process from Fig. 4).

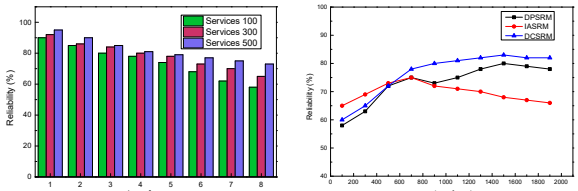
The self-healing process in Fig. 6 is formalized by the following term language:

$/e_5./e_6./e_7./e_8./e_9./n.\,(s_{4e_5n}(\text{-}_0)\|\text{-}_1\|n/y\|\|s_{5e_5e_6}\|\|s_{6e_6e_7}\|s_{7e_7e_8e_9}\|\|s_{8e_8}\|s_{9e_9})$
$\rightarrow /e_{11}./e_{12}./e_{13}./e_5./e_6./e_7./e_8./$
$e_9.\,(s_{4e_{13}e_5}(s_{10e_{11}}\|s_{11e_{11}e_{12}}\|s_{12e_{12}e_{13}})\|\text{-}_1\|\|s_{5e_5e_6}\|\text{-}_2\|\|s_{6e_6e_7}\|s_{7e_7e_8e_9}\|\|s_{8e_8}\|s_{9e_9})$

## V. EXPERIMENT ANALYSIS

To illustrate the effectiveness of the algorithm described in this paper, we conducted a series of experiments in different settings. The PC configuration was as follows: Intel Core i5-8250U CPU (1.6 GHz), Windows 8 and 6 GB RAM. We used all 12 of the atomic services discussed in Section 4. We then extracted their parameters and used them as a seed to randomly generate an extended dataset with 500-2000 services. Each service contained basic information such as the service name, input, output, and success probability. The experiments compared three kinds of algorithm: (1) The algorithm (DPSRM) based on the Dynamic Software Product Line approach [17]; (2) The algorithm (IASRM) based on the process ontology and multiple recovery [18]; (3) The data cell self-healing algorithm (DCSRM) proposed in this paper. In addition, we randomly set the effective service that failed as a variant $v$ during the composition, which automatically triggers the service substitution. To ensure unbiased statistical results, all algorithms were executed independently 20 times.

Fig. 7a presents the results obtained using only DCSRM. The service number varies from 100 to 500 and the number of variants increases gradually from 1 to 8. We found that when the number of variants is small, an increase in the service number produces a steady increase in the reliability. However, as the number of variants increases, the reliability becomes relatively low, especially when the number of services is small. This is because there are fewer similar services in the small service set, resulting in no suitable service being found when the number of variants grows. Fig. 7b shows results for $v$ = 9 and the service number varying from 100 to 2000. Initially, IASRM gives the highest reliability, but this obviously decreases as the number of services increases. Furthermore, DCSRM always outperforms DPSRM.




(a) (b)

Fig. 7. Comparison of reliability with different variants and services

From Fig. 8, we can see that the response time grows with the number of services and variants. Moreover, no matter how $v$ is allocated, DCSRM performs much better and faster than IASRM and DPSRM. Thus, the experimental results illustrate that our algorithm significantly improves reliability and reduces the time cost of service composition.

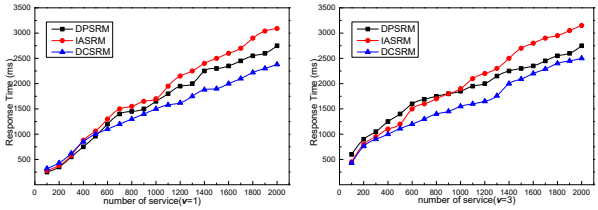



Fig. 8. Comparison of response time with different variants and services

## VI. RELATED WORK

### A. Bigraphs and their application

Bigraphs were proposed by Milner and other scholars in 2001 to emphasize the location and connection of computing units (physical or virtual) [6], and has now become the tool of choice for many service-adaptive and software-reconfigurable systems because of its complete formal theory and dynamic mobility. For example, Lian et al. [7] simulated modeling using a bigraphical reactive system to analyze the mobile cloud. Calder et al. [8] proposed a model based on checking predicates from user-initiated and network events by extension to a bigraphical reactive system

### B. Evolution of service composition

Ensuring the rationality of business process structures after evolution is an important problem in the composition of services. If the evolution operation is not implemented properly, it may cause problems such as a logical deadlock or component service unreachability [20]. This section analyzes the evolution of service composition in terms of the rationality of service evolution operations and the current solutions for failure recovery.

#### 1) Rationality of service evolution operations

The rationality of the service composition process and the correctness of the data flow are usually guaranteed by defining the evolution criterion in the operation process, thereby avoiding complex verification processes after evolution. To ensure the rationality of the service composition process, Zeng et al. [9] proposed a set of basic evolutionary operations for adjusting the service workflow structure based on workflow network modeling, which guarantees the rationality of the internal process logic during the business process. Urbieta et al. [3] propose an adaptive service composition framework that supports the dynamic reasoning of user requests and service behaviors in the smart city. Khanfir et al. [2] propose a framework for automatic generation and publishing of service descriptions by using OWL-S semantic annotations, the purpose of it is to analyze the process modeling and the choreography of service composite. In service behavior analysis, the research is mainly used by formal methods such as Petri net, process algebra, and π calculus, etc.

*2) Current solutions to failure recovery*

Self-adaptation is the ability of a system to adapt to changes in its environment to maintain the original functionality, and is used in different problem domains. Many recent studies have focused on enabling adaptation for BPEL processes. For instance, the monitoring mechanism embedded in the BPEL engine can be used to capture fault messages [10], allowing existing processes to be directly deployed without any modifications. Some scholars perform fault monitoring and recovery through transaction attributes of object states. Ettazi et al. [11] fulfilled user requirements in mobile environments by focusing on transactional aspects of context-aware services. In addition, there has been some research based on security monitoring and self-healing systems [12,19]. Asim [12] presented a framework that automates the monitoring of business processes and reports the compliance violations at runtime.

## VII. CONCLUSION

Service composition is an important technology for integrating information to create new value in systems that satisfy complex requirements. In this paper we propose a novel approach for service composition with data cell modeling, which is inspired by biological cell and guided by the bigraph theory. This enables users to efficiently analyze the services in a dynamic distributed environment.

However, the preliminary data cells and clusters from model still need to be manually configured by analyzing and extracting the important characteristics from services. So, for future work, some tools may be applied to extract service features automatically. Another problem for future research is that more constraints from specific customer requirements, including service rating, service price and so on, need be considered to optimize the model in different scenes.

## ACKNOWLEDGMENT

This work is supported by the Science and Technology Key Research Planning Project of Zhejiang Province, China (NO.2018C01064), and Zhejiang Natural Science Foundation, China (No. LY19F020034).

## REFERENCES

[1] J. Cheng, C. Liu, M. Zhou, Q. Zeng, and A. Ylä-Jääski, "Automatic composition of semantic web services based on fuzzy predicate petri nets," IEEE Transactions on Automation Science and Engineering, 12(2), pp. 680-689. 2015.

[2] E. Khanfir, R B. Djemaa and I. Amous, "Automatic Adaptable Intentional Service Generating and Publishing Framework using OWL-S Annotation," International Journal of Web Services Research (IJWSR), 15(1), pp. 1-26. 2018.

[3] A. Urbieta, A. González-Beltrán, S. B. Mokhtar, M. A. Hossain, and L. Capra, "Adaptive and context-aware service composition for IoT-based smart cities," Future Generation Computer Systems, 76, pp. 262-274. 2017.

[4] Z. Xiong, W. Luo, L. Chen, and L. M. Ni, "Data vitalization: a new paradigm for large-scale dataset analysis," In 2010 IEEE 16th International Conference on Parallel and Distributed Systems. IEEE. 2010, pp. 251-258.

[5] W. Zhou, L. Liu, C. Pu, et al, "An Experimental Study of a Biosequence Big Data Analysis Service," 2017 IEEE International Conference on Web Services (ICWS). IEEE, 2017, pp. 237-244.

[6] R. Milner, "Bigraphical reactive systems," International Conference on Concurrency Theory. Springer, Berlin, Heidelberg, 2001, pp. 16-35.

[7] L. Yu, W T. Tsai, X. Wei, et al. "Modeling and analysis of mobile cloud computing based on bigraph theory," 2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering. IEEE, 2014, pp. 67-76.

[8] M. Calder, A. Koliousis, M. Sevegnani and J. Sventek, "Real-time verification of wireless home networks using bigraphs with sharing," Science of Computer Programming, 80, pp. 288-310. 2014.

[9] J. Zeng, HL. Sun, XD. Liu, T. Deng and JP. Huai, "Dynamic evolution mechanism for trustworthy software based on service composition," Journal of Software, 21(2), pp. 261−276. 2010.

[10] H. Huang, X. Chen and Z. Wang, "Failure recovery in distributed model composition with intelligent assistance," Information Systems Frontiers, 17(3), pp. 673-689. 2015.

[11] W. Ettazi, H. Hafiddi, M. Nassar, and S. Ebersold, "Micats: Middleware for context-aware transactional services," In International Conference on Enterprise Information Systems, Springer. 2015, pp. 496-512.

[12] M. Asim, A. Yautsiukhin, A.D. Brucker, et al, "Security policy monitoring of BPMN‐based service compositions," Journal of Software: Evolution and Process, , 30(9), e1944. 2018.

[13] R. Milner, "Axioms for bigraphical structure," Mathematical Structures in Computer Science, 15(6), pp. 1005-1032. 2005.

[14] N. Trčka, WMP. Van der Aalst and N, "Sidorova Data-flow anti-patterns: Discovering data-flow errors in workflows," International Conference on Advanced Information Systems Engineering. Springer, Berlin, Heidelberg, 2009, pp. 425-439.

[15] X. Liu, Y. Ma and G. Huang et al, "Data-driven composition for service-oriented situational web applications," IEEE Transactions on Services Computing, 8(1), pp. 2-16. 2015.

[16] G. H. Alférez, V. Pelechano, R. Mazo, C. Salinesi, and D. Diaz, "Dynamic adaptation of service compositions with variability models," Journal of Systems and Software, 91, pp. 24-47. 2014.

[17] M. Bashari, E. Bagheri, W. Du, "Self-healing in service mashups through feature adaptation," Proceedings of the 21st International Systems and Software Product Line Conference, ACM.2017, pp. 94-103.

[18] H. Huang, X. Chen, Z.Wang, "Failure recovery in distributed model composition with intelligent assistance," Information Systems Frontiers, 17(3), pp.673-689. 2015.

[19] S. Subramanian, P. Thiran, N C. Narendra, et al. "On the enhancement of bpel engines for self-healing composite web services," International Symposium on Applications and the Internet. IEEE, 2008, pp. 33-39.

[20] S. Rinderle, M. Reichert and P. Dadam, "Correctness criteria for dynamic changes in workflow systems—a survey," Data & Knowledge Engineering, 50(1), pp. 9-34. 2004,