A Formal Engineering Approach to Service-based Software Modeling and Integration Testing

Weikai Miao
Faculty of Computer and Information Sciences
Hosei University
weikai.miao.x1@stu.hosei.ac.jp

***Abstract*—With the increasing popularity of service-based software in recent years, engineering methods for developing high quality service-based systems is highly demanded which are expected to support the essential engineering processes of *system modeling*, *web service selection* and *system testing*. However, few systematic methods that unify the above three essential activities are available, and the present supporting technologies of these three activities are still not satisfactory. In order to tackle this challenge, this paper proposes a formal engineering approach that integrates precise system modeling, accurate service selection and rigorous system integration testing. It includes a unified three-step formal engineering framework for interactive service-based system modeling and existing service adoption, a service selection method that combines both static matching and specification-based conformance testing and a formal specification-based integration testing method. We have also developed a prototype tool that supports the proposed engineering framework. An empirical case study and corresponding experiments are conducted to show the feasibility of the proposed approach.**

I. Introduction

In recent years, large-scale service-based software systems have been successfully deployed in many enterprises and organizations ranging from transportation, healthcare, financial companies, governments, and other social sectors [1][2]. Everyday or even every minute, service-based software systems are continuously providing the business services, which has substantially improved our daily life. Inspired by such great achievements and bright future, the engineering method for constructing high-quality service-based software systems is one of the major research areas, which attracts growing attention of both the industry and research communities.

Service-based software system is the software system in which appropriate existing web services are reused as software components to perform the required business functions. Its development, to a large extent, inherits the traditional software engineering principles, though specific engineering processes need to be considered for dealing with the web services that participate in the construction of service-based software. System modeling and testing serve as the most critical stages for constructing high-quality service-based software systems.

Unfortunately, effective service-based software modeling and testing are still facing great challenges. Specifically, one major challenge is how to carry out the service-based software modeling for constructing the precise requirements and design specifications so that existing services can be efficiently and accurately discovered, selected, and reused in the target system. The second challenge is how to effectively utilize the system models to carry out service-based software testing to ensure the software quality.

Formal methods are regarded as potential solutions to tackle the above challenges which embrace two techniques: *formal specification* for precise description of system requirements and formal verification for proving the correctness of software against formal specifications. It is well recognized that formal software modeling, including both formal requirements and design specifications, based on correct understanding of requirements contributes significantly to software quality [3][4]. Unfortunately, the formal modeling is in fact not widely used in industry, partly because the mathematically-based notation is hard for ordinary practitioners to accept and master, partly because practical constraints, such as time, expertise, or cost, prevent companies from providing an environment suitable for application of formal methods [5][6], and partly because there is still lack of systematic engineering approaches to formal specification construction. Moreover, appropriate services need to be discovered, selected and integrated into the system architecture for developing service-based software, which cannot be handled by directly applying the traditional formal methods.

To this end, this paper put forward a formal engineering approach to service-based software modeling and integration testing. It inherits the basic idea of the SOFL (Structured Object-oriented Formal Language) formal engineering method [7][8] that supports the application of formal methods in developing large-scale software systems, providing systematic engineering processes to support the formal modeling and formal specification-based integration testing of service-based software systems.

Specifically, the formal engineering approach consists of a formal engineering framework (FEFSSM) and a specification-based integration testing method. The FEFSSM offers a three-step modeling framework for specification construction and integrates web service discovery and selection into the entire modeling phase, aiming at providing an engineering procedure for constructing precise, comprehensible, and satisfactory specifications of service-based software. The constructed system models are represented by both the formal specifications that precisely define the expected functions and the corresponding rigorous condition data flow diagrams (CDFDs) that precisely describes the interactions among the system modules.

The textual and the diagrams are adopted as the foundation of the rigorous integration testing.

To promote the effectiveness and efficiency of service-based software modeling and service selection, we have developed a prototype tool that can support the three-step specification construction and the service discovery and selection activities.

The remainder of this paper is organized as follows. Section II summarizes the research work related to our approach from various aspects and illustrates the novelty of our research through the comparison with these related work. Section III presents the main principle underlying our formal engineering approach to service-based software modeling and integration testing. Sections IV, V, and VI describe detailed techniques involved in different specification construction stages. Section VII describes the formal specification-based integration testing method. Section VIII shows the prototype tool that supports the FEFSSM. Section IX reports the case study and the corresponding experiments conducted for validating the feasibility of the approach. Section X presents the conclusion of our work.

II. Related Work

Wide ranges of methods have been devoted to service-based system modeling and construction. One representative approach is the SOMA (Service-Oriented Modeling and Architecture) method proposed by IBM [9]. SOMA incorporates key aspects of service-oriented software modeling (e.g., service identification, business modeling and transformation and etc.) and offers a unified framework for developing service-oriented software. Inspired by the basic principle underlying the SOMA method, various MDA-based approaches to service-based software modeling have been put forward. These approaches mainly adopt main stream modeling notations such as the UML and BPMN for modeling the expected requirements and business processes [10][11][12][13]. In most of the MDA modeling approaches, the high-level system models specified by UML or BPMN are gradually transformed into final implementations. Although the BPMN language and UML-based approaches have contributed to the modeling of service-based software, precise specifications of service-based software cannot be achieved by following these techniques alone. In order to facilitate the construction of precise specifications of service-based systems, one representative trend is the formalism of BPMN or BPEL notations. The integration of Petri-Net and BPMN are reported in works [14][15][16]. Similarly, the authors of work [17] suggest using YAWL to enhance the formal semantics of BPMN notations. In work [18], authors extend current BPEL language for modeling service choreographies. To some extent, enhancement of modeling languages contributes to service-based software modeling from various aspects. However, these improvements focus on the language level rather than the perspective of methodology for service-based software modeling. The modeling of large-scale systems are still not easy since few engineering methods can guide practitioners construct the formal specifications.

Existing service discovery and selection techniques can be summarized into two categories: the *match-making* approaches and the *conformance testing* approaches. As a main stream trend, *match-making* methodology is widely adopted as the foundation of many service discovery and selection approaches. Signature matching is the simplest way of web service match-making, which can be traced back to resources discovery from software libraries [19]. One typical scenario of signature matching is that requirements specifications are abstracted by some keywords that are used to match the informal descriptions of candidate web services. Since most services are published with WSDL files which specify their interfaces syntax, approaches based on interface structure match-making have been proposed [20][21]. Due to the lack of analysis on the functional behaviors of services, these approaches cannot achieve an accurate service selection. With the development of service description technologies, the behavior match-making approaches to service discovery and selection come into existence. In particular, matching approaches based on semantic web services are widely used by many projects [22][23][24][25][26][27][28]. In spite of the advantage brought by semantic services, two challenges that obstacle wide application of semantic services still need to be tackled. One major problem is the insufficient available semantic services under real web environment. The other concern is the reliability of semantic descriptions. Therefore, Service conformance testing is regarded as an alternative solution to accurate service selection. A number of service conformance testing approaches are presented in work [29][30][31][32][33][34][35][36][37][38][39][40].. Despite of their contributions to service conformance testing, a majority of current service testing methods focus on checking the conformance of service functions with respect to pre-defined specifications before service publication or service description documents. To solve this problem, user-end conformance testing approaches are proposed. Test data of the user-end conformance testing is generated from the users' (i.e., the service requestors') requirements specifications [38][41].

Various integration testing approaches adopt the control flow diagrams, especially the UML sequence diagram for test case generation. In work [42], the authors transform the UML sequence diagram into the Sequence Dependency Diagram for test case generation. In the work [43] and [44], state charts are used as the foundation of test case generation for integration testing. Similarly, the authors of work [45] and [46] combine the state-machine and UML sequence diagram for test data generation of integration testing. The work [47] also adopts the UML sequence diagram as the foundation for the integration testing of object-oriented programs. However, the control flow diagrams and state machines can still satisfy the demanding of integration testing. As pointed out by the authors of [45], state charts are more powerful in unit testing rather than in integration testing. Formal specification-based testing is regarded as a promising technique for rigorous software faults detection [48][49]. However, current formal specification-based testing methods face a challenge that most of them are powerful in unit testing but relatively difficult to be applied in integrating testing.

III. Overview of the Approach

The proposed formal engineering approach consists of a formal engineering framework for service-based software modeling, and a method for service-based software integration testing based on the formal specifications. The main principle of our this approach is shown in Figure 1.

As shown in Figure 1, the approach primarily consists of a three-step modeling and a specification-based integration testing processes. The main idea of the modeling process is to provide an evolutionary modeling framework that offers a three-step specification construction procedure in which appropriate services are also discovered, selected and integrated into the system design architecture. Specifically, the three stages of the modeling approach are listed below:

- Informal Specification Construction

  The goal is to acquire requirements as completely as possible and discover sufficient candidate services based on the informal requirements for further conformance checking at later stages. Since requirements are imprecise at this stage, candidate services are preliminarily explored and filtered using keywords that abstract the corresponding informal requirements. For example, in Figure 1, requirements are documented as the informal specification accompanied with a corresponding decomposition diagram that described the relations between the expected functions. Potential services are discovered based on the informal requirements. For instance, function a_2_1 is decomposed from the sub-function a_2 of function A, which is associated to candidate services $S1$ and $S3$ through the keyword-based service discovery.
- Semi-Formal Specification Construction

  This stage aims at carrying out accurate service selection and further clarifying the requirements. All service-associated functions are transformed into formal processes specifying the expected functions rigorously while other functions are transformed into semi-formal processes that consist of formal data structures and informal operational semantics. Services are accurately selected through static behavior analysis and specification-based conformance testing; rest part of the informal specification is then clarified into semi-formal specification. In Figure 1, the informal specification is evolved into semi-formal specification. Service $S1$ is identified as the most relevant service of function a_2_1. Assume that service $S1$ is selected for the service-associated process a_2_1 through the testing, rest part of the specification can be semi-formalized.
- Formal Specification Construction

  In this stage, all the pre- and post-conditions of all processes are formalized and all the processes are organized into a hierarchical structure of system modules. Service-associated processes can be used as the foundation for gradually formalizing the entire specification since they have been determined in the previous modeling stage. The formal specification in Figure 1 shows that all processes and data structures are formally specified.

Integration testing is a quality assurance engineering activity to be carried out after the system construction and unit testing. Since unit testing approaches are relatively mature, our method focuses on the integration testing techniques in the testing stage. The first step of integration testing approach is to derive all potential data flow paths from the CDFDs. Each data flow path is a sequence of data flows and the involved processes of the path. The pre- and post-conditions of each process are then further transformed into a functional scenario form. Concrete test cases are then generated to cover all the functional scenario sequences of each path.

Based on the framework, we will explain the detail techniques and methods of each stage of the three-step modeling framework and the integration testing method, respectively.

IV. Informal Specification Construction Stage

Constructing an informal specification is a mean to help the initial requirements acquisition and service discovery. That is, in addition to capturing client's requirements through writing the informal specification, the content of the specification can also help the developer consider whether any existing services can be utilized to realize some of the required functions.

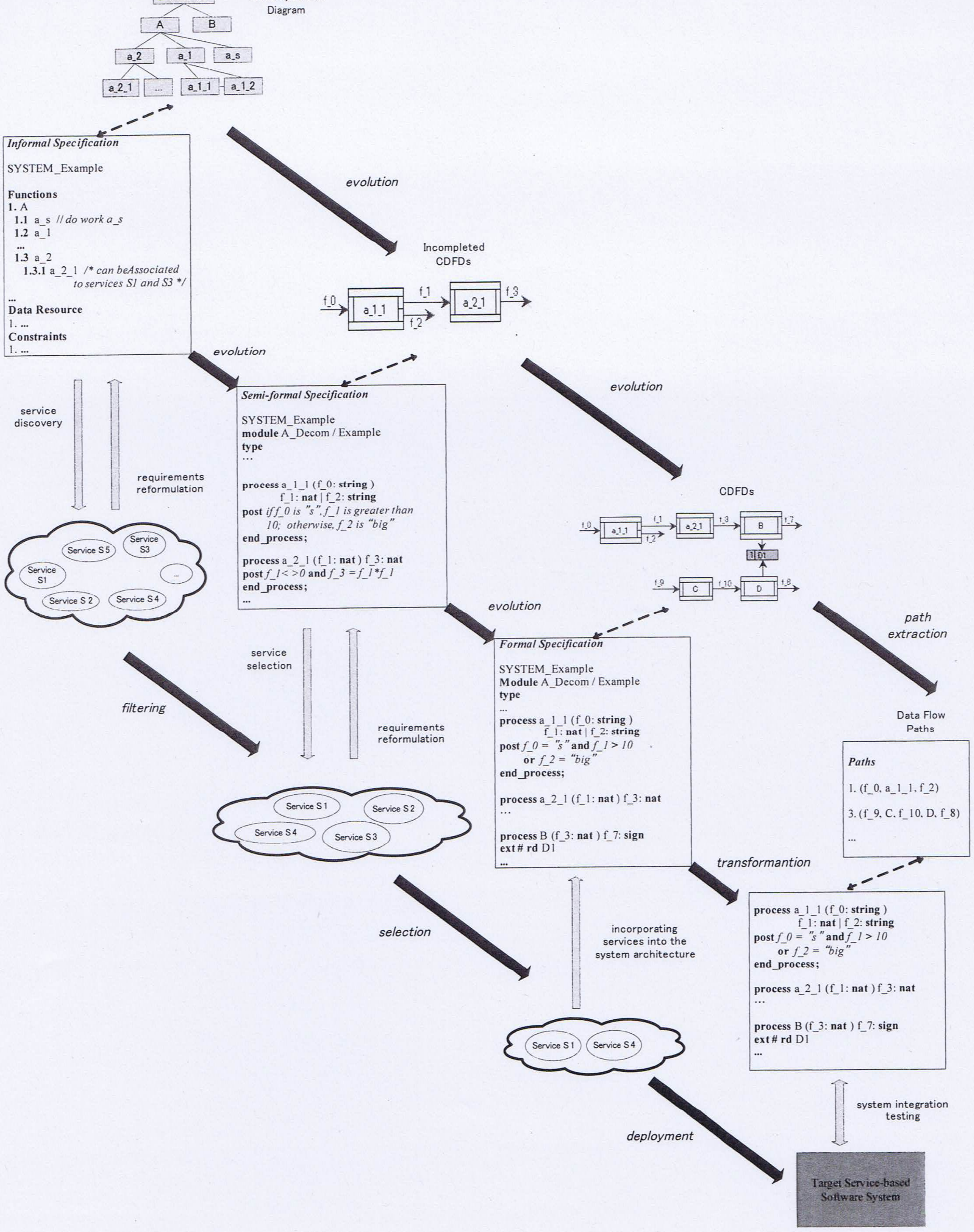
A. Requirements Acquisition

Requirements acquisition is an activity to discover, understand and document the customer requirements, which is aimed at achieving a complete coverage of desired requirements and finding clues for services discovery. Specifically, requirements are acquired via the discussion between the client and the system developer based on the understanding of the business goals, the documents of the legacy systems, the domain knowledge and etc. A top-down strategy is adopted for analyzing the business processes. Specifically, the requirements analysis starts from the top-level business processes. Through the analysis of the business processes, including the decomposition of functions, expected functions can be extracted and documented in a hierarchical structure to clearly describe the client's requirements.

B. Service Discovery

During the construction of the informal specification, services are discovered and associated with corresponding functions. Since requirements are informally described, the developer only needs to discover candidate services for more accurate selection at a later stage.

The technique for service discovery of our approach is *keyword-based searching*, where a keyword in our context is an English word. The keywords are derived from the informal specification and reflect the major features of the desired functions that the potential candidate services are expected to offer. Specifically, the derived keywords are used to either partially or completely match with the service names or informal descriptions stored in the service repositories.



```
Informal Specification

SYSTEM_Example

Functions
1. A
  1.1 a_s // do work a_s
  1.2 a_1
  ...
  1.3 a_2
    1.3.1 a_2_1 /* can beAssociated
                   to services S1 and S3 */
...
Data Resource
1. ...
Constraints
1. ...
```

```
Semi-formal Specification

SYSTEM_Example
module A_Decom / Example
type
...

process a_1_1 (f_0: string )
        f_1: nat | f_2: string
post if f_0 is "s", f_1 is greater than
     10; otherwise, f_2 is "big"
end_process;

process a_2_1 (f_1: nat) f_3: nat
post f_1< >0 and f_3 = f_1*f_1
end_process;
...
```

```
Formal Specification

SYSTEM_Example
Module A_Decom / Example
type
...
process a_1_1 (f_0: string )
          f_1: nat | f_2: string
post f_0 = "s" and f_1 > 10
   or f_2 = "big"
end_process;

process a_2_1 (f_1: nat ) f_3: nat
...

process B (f_3: nat ) f_7: sign
ext # rd D1
...
```

```
Paths

1. (f_0, a_1_1, f_2)

3. (f_9, C, f_10, D, f_8)

...
```

```
process a_1_1 (f_0: string )
          f_1: nat | f_2: string
post f_0 = "s" and f_1 > 10
   or f_2 = "big"
end_process;

process a_2_1 (f_1: nat ) f_3: nat
...

process B (f_3: nat ) f_7: sign
ext # rd D1
...
```

Fig. 1. Main Principle Underlying the Formal Engineering Approach

C. Dynamic Informal Specification Construction

In FEFSSM, when the business processes are analyzed, for each single function involved in each business process , the developer needs to consider whether the function can be implemented by certain services, decomposed into sub-functions for further analysis, or just left for coding from the scratch. An algorithm is proposed showing how the service searching is accompanied with the handling of a single function in the specification.

V. Semi-formal Specification Construction

Our formal engineering method inherits the semi-formal stage of specification construction from the traditional SOFL formal engineering method and extends it for handling service-based software modeling. In order to effectively exploit existing services into the system architecture, services need to be first accurately selected and the grouping of related functions, data resources, and constraints into modules is then carried out. To this end, our formal engineering method extends the SOFL method and suggest the following steps in the stage of semi-formal specification construction:

1) filtering candidate services via static behavior analysis
   Large number of candidate services may be discovered at the informal specification construction stage. To facilitate effective service selection, services are ranked based on their relevance to the associated functions. The relevance is judged by the developer through static behavior analysis on the descriptions of candidate services, since most services are provided with some description files, especially WSDL files specifying their interfaces. Some irrelevant services can be directly eliminated according to the developer's judgements. Such a ranking and filtering mechanism is achieved based on the analysis of the services descriptions in a static manner rather than dynamically checking the functional behaviors of the services. As the result of static behavior analysis, each service-associated function is finally associated with its most relevant service. The most relevant services will be used as the basis for formalizing the associated functions and also be tested prior to the less relevant services for accurate service selection.
2) formalizing the service-associated functions
   To carry out accurate service selection through specification-based conformance testing, service-associated functions need to be completely formalized as the foundation for generating proper test cases to run the corresponding candidate services (i.e., the most relevant services). The formalization includes two steps. The first step is to modularize the service-associated functions into proper SOFL processes. The second step is to fully formalize the pre- and post-conditions of these processes to precisely express the expected operational semantics upon their associated services.
3) determining services using specification-based conformance testing
   To check the conformance of the candidate services with respect to the expected functions, services are tested by running the test cases generated from the formal service-associated processes [50]. The final decisions on service selection are made by the developer via the analysis upon testing results and the service-associated processes. If a service under test does not satisfy the required functions, other relevant services can be picked up for testing according to their relevance ranking established via static behavior analysis. In this case, specification may have to be reformulated.
4) semi-formalizing the specification based on the determined service-associated processes
   Appropriate services are determined after the specification-based conformance testing. The determined formal service-associated processes can be used as foundation for semi-formalizing rest part of the specification. This involves formalizing all the related data structures, transforming all the functions into processes and grouping them into independent system modules.

VI. Formal Specification Construction

The major task of the formal specification construction is twofold. One goal is to design the system architecture by coupling the independent system modules; the other one is to formally define all the components (e.g., processes and invariants) that are not completely formalized in the specification. A formal specification represents the architecture of the entire system and functional definitions of its components.

To effectively support the formal specification construction, we adopt the CDFD assisted middle-out strategy. Specifically, the following steps are taken for exploiting the strategy:

1) select the crucial modules that represent the important functions
2) construct the CDFDs of the crucial modules
   a) if a crucial module is decomposed, formalize the processes of this module and then construct the CDFDs of the lower-level modules and formalize the involved processes
   b) if a crucial module is not decomposed, formalize all the processes of this module
3) organize the modules that are fully formalized according to the design decisions
4) complete the formal specification based on the corresponding CDFDs

By following the middle-out strategy, we can gradually construct all the CDFDs of the modules and processes at different levels; meanwhile, the textual formal specifications of each module and process are completed guided by the organization procedure of the CDFDs.

Based on the formal specification, the developer can implement the target service-based software using appropriate program languages. Since the SOFL formal language is not executable, the formal specification cannot be automatically transformed into programs. The developer needs to manually map the specification to the programs by following the basic guidelines proposed in our previous work of the SOFL [8].

VII. The Formal Specification-based Integration Testing Method

The goal of this method is to detect software faults that are concealed in the interactions between system operations. The foundation of this formal specification-based integration testing is the textual formal specification and the affiliated hierachical CDFDs of the system architecture. Since the hierarchy of the system modules are determined, the bottom-up strategy for integration testing can be adopted.

Specifically, the integration testing starts from the bottom level system modules. When a lower-level module is thoroughly tested, it is nested as a process of its higher level system module for integration testing. This iterative procedure continues until the top-level system module is finally tested. Therefore, the fundamental problem to be resolved is the integration testing of each individual module described by a CDFD. The following steps can be taken for the integration testing of each module:

1) extracting all independent paths from the CDFD;
2) transforming the pre- and post-conditions of each process involved in each path into functional scenario form;
3) constructing the functional scenario sequences of the involved processes of each path, and then testing the system using the test cases generated from the constructed functional scenario sequences;

VIII. The Prototype Tool of FEFSSM

To effectively support the FEFSSM and the specification-based integration testing approach, we also devote to the development supporting tools. Currently, a prototype tool that supports the FEFSSM is constructed.

A. Tool Design

The tool is designed as a three-layered system which provides the major functions supporting the application of the FEFSSM for modeling service-based software systems. The architecture of this supporting tool is described in Figure 2.

The infrastructure layer mainly consists of the service repositories, the specification files and the files of functional scenario matrixes. In our case study, the service repository is only the AXIS2 service server. Specifications are documented in the XML files. We use the XML files to store the specifications since the XML format files are machine-readable, platform-independent and can easily represent the hierarchical structures of the processes and other SOFL components (also including the CDFD).

The function layer consists of three modules: the module of service discovery and analysis, the module of specification construction and the module of functional scenario matrix establishment. The module of service discovery supports the keyword-based web service discovery and the static behavior

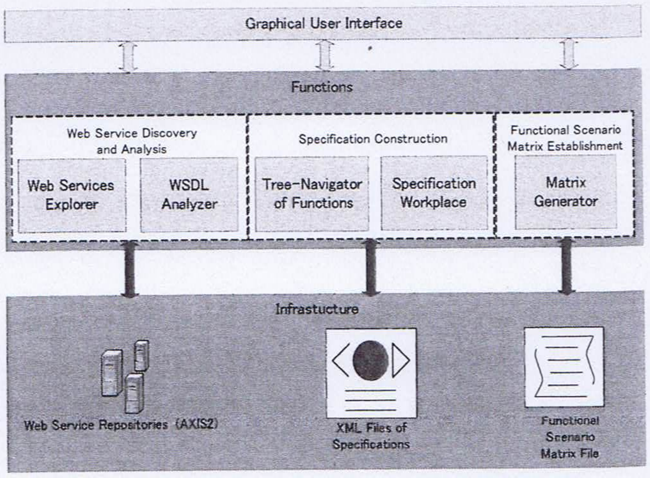

Graphical User Interface
Functions
Web Service Discovery and Analysis
Web Services Explorer
WSDL Analyzer
Specification Construction
Tree-Navigator of Functions
Specification Workplace
Functional Scenario Matrix Establishment
Matrix Generator
Infrastucture
Web Service Repositories (AXIS2)
XML Files of Specifications
Functional Scenario Matrix File

Fig. 2. Architecture of the FEFSSM Supporting Tool

analysis in which services are ranked. Specifically, the service discovery is achieved by the web service explore, and the static behavior analysis is mainly carried out by the WSDL Analyzer component.

B. Implementation of the Prototype Tool

Figure 3 gives a snapshot of the main interface of the tool.

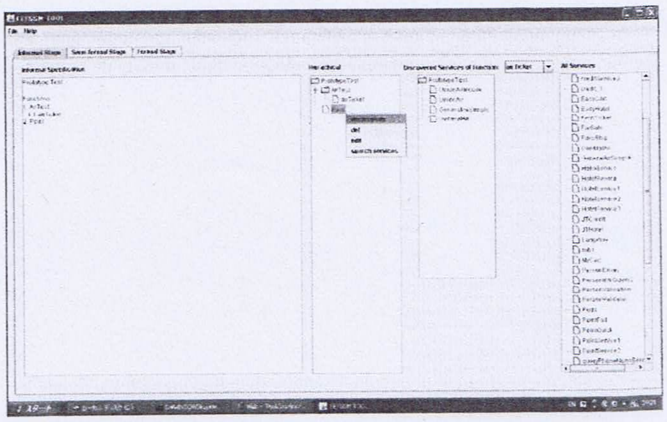


Fig. 3. The Snapshot of the Main Interface of the Prototype Tool

The text edition area located in the left-side of the interface is the workplace for textual specification construction. The tree-navigator is in the middle part. A tree structure of expected functions is described by current navigator in which each node is an expected function. By right-clicking the node, the user can decide to decompose, delete, edit the function or search candidate services for the function. Discovered services can be listed after the keyword-based searching. The right-side of this interface is the area of service repositories. All candidate services are listed in this area. For the sake of space limitation, other functions of the tool are omitted for brevity.

IX. Case Study and Experiments

We have carried out an empirical case study of developing a Travel Agency System (TAS) for validating the feasibility of the FEFSSM and two experiments for evaluating the service conformance testing method of service selection and the integration testing method.

The case study shows that the number of candidate services gradually decreases along with the specification evolution and the number of required functions increases simultaneously. At the stage of informal specification construction, functions were gradually recorded along with service discovery. During the stages of semi-formal and formal specification construction, the number of functions increased rapidly. Stimulated by the detailed information of service operations extracted from the WSDL files, the developer and the client clarified the specification extensively. As a result, sub-functions were added for their high-level functions that were roughly associated to the corresponding services and some functions were reformulated.

The experiment for service conformance testing method of service selection shows that the overall fault detection rate is approximately 90%. This result demonstrates that our conformance testing method achieve a relative high detection rate of software faults. Such a testing approach can effectively check the conformance of services to the expected functions and also facilitates the reliability of the target service-based software.

The experiment for integration testing method shows that the overall fault detection rate is 85%. This fault detection rate is higher than the one reported in work [51] in which test data generation is achieved based on the paths coverage but regardless of the functional scenarios.

X. Conclusion

With the increasing demands of efficient and value-added software systems for enabling complex business requirements, and the continuous development of computer sciences and technologies, the service-based software systems have seized great interests of the industry and research communities. The development methods for constructing high-quality service-based software are one of the most crucial topics in the area of software engineering.

Targeting on the most critical engineering tasks of service-based software system development, the modeling and the testing phases, we propose a formal engineering approach to service-based software modeling and integration testing. Specifically, the formal engineering framework for service-based software modeling (FEFSSM) supports the evolutionary formal specification construction and the accurate web service selection. On the basis of the formal specifications and the affiliated condition data flow diagrams (CDFDs), the formal specification-based integration testing approach contributes to the systems quality via a rigorous testing procedure. An empirical case study of developing a *Travel Agency System* (TAS) is carried out for validating the feasibility of the FEFSSM, and the related experiments are also conducted for evaluating the service conformance testing method of service selection and the integration testing method. A prototype tool is developed to support the FEFSSM.

Our research and its novelty are summarized as the follows:

1) The Formal Engineering Framework for Service-based Software Modeling
   We have presented a novel framework FEFSSM for service-based software modeling. The main contribution of the FEFSSM is a systematic solution to service-based software modeling through an evolutionary process for specification construction and accurate service selection.
2) The Formal Specification-based Integration Testing Method
   To facilitate the quality assurance of service-based software systems, we have proposed a integration testing method based on the formal specification and the CDFDs acquired form the modeling phase. The most distinguished merit of the integration testing method is the utilization of both the SOFL textual formal specification and the condition data flow diagrams as the foundation of rigorous and intuitive test data generation and test result analysis.
3) The Case Study and Experiments
   An empirical case study of developing a Travel Agency System (TAS) and the related experiments are carried out for validating and evaluating the feasibility of the proposed approach.
4) The Prototype Supporting Tool
   A prototype tool that supports the FEFSSM is established.

References

[1] 2012. [Online]. Available: "http://www.tibco.com/multimedia/ss-delta-tcm8-754.pdf"

[2] 2012. [Online]. Available: http://www-03.ibm.com/press/us/en/pressrelease/26995.wss

[3] A.Hall, "Using Formal Methods to Develop an ATC Information System," *IEEE Software*, vol. 13, pp. 66–76, March 1996.

[4] G. Booch, *Object-Oriented Analysis and Design with Applications*. Addison Wesley Longman, 1994.

[5] J. C. Knight, C. L. DeJong, M. S. Gibble, and L. G. Nakano, "Why Are Formal Methods Not Used More Widely?" in *Fourth NASA Langley Formal Methods Workshop*, C. M. Holloway and K. J. Hayhurst, Eds., no. 3356, Hampton, Viginia, 1997, pp. 1–12.

[6] J. Woodcock, P. G. Larsen, J. Bicarregui, and J. Fitzgerald, "Formal Methods: Practice and Experience," *ACM Computing Surveys*, vol. 41, no. 4, 2009.

[7] S.Liu, A.J.Offutt, C.Ho-Stuart, Y.Sun, and M.Ohba, "SOFL: A Formal Engineering Methodology for Industrial Applications," *IEEE Transactions on Software Engineering*, no. 1, pp. 24–45, 1998.

[8] S.Liu, *Formal Engineering for Industrial Software Development Using the SOFL Method*. Springer-Verlag, 2004.

[9] A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy, and K. Holley, "SOMA: A method for Developing Service-Oriented Solutions," *IBM Systems Journal*, vol. 47, no. 3, pp. 377 –396, 2008.

[10] H. Gao, J. Zhang, R. Povalej, and W. Stucky, "Service-Oriented Modeling Method for the Development of an E-Commerce Platform," in *Int'l Conf. on E-Business and Information System Security*, May 2009, pp. 1 –5.

[11] A. Ruokonen, L. Pajunen, and T. Systa, "Scenario-Driven Approach for Business Process Modeling," in *IEEE Int'l Conf. on Web Services (ICWS)*, July 2009, pp. 123 –130.

[12] H. D. Kim, "BPMN-Based Modeling of B2B Business Processes from the Neutral Perspective of UMM/BPSS," in *IEEE Int'l Conf. on e-Business Engineering (ICEBE)*, Oct. 2008, pp. 417 –422.

[13] A. Sadovykh, P. Desfray, B. Elvesaeter, A.-J. Berre, and E. Landre, "Enterprise architecture modeling with SoaML using BMM and BPMN - MDA approach in practice," in *6th Central and Eastern European Software Engineering Conference*, Oct. 2010, pp. 79–85.

[14] P. Y. H. Wong and J. Gibbonse, “Verifying Business Process Compatibility,” in *Proceedings of The Eighth International Conference on Quality Software*. IEEE Computer Society, June 2008.

[15] T. Takemura, “Formal Semantics and Verification of BPMN Transaction and Compensation,” in *IEEE Asia-Pacific Services Computing Conference (APSCC)*, Dec. 2008, pp. 284 –290.

[16] B.-H. Schlingloff, A. Martens, and K. Schmidt, “Modeling and Model Checking Web Services,” *Electronic Notes in Theoretical Computer Science*, pp. 3–26, March 2005.

[17] J. Ye, S. Sun, W. Song, and L. Wen, “Formal Semantics of BPMN Process Models Using YAWL,” in *Second International Symposium on Intelligent Information Technology Application*, vol. 2, Dec. 2008, pp. 70 –74.

[18] G. Decker, O. Kopp, F. Leymann, and M. Weske, “BPEL4Chor: Extending BPEL for Modeling Choreographies,” in *IEEE Int'l Conf. on Web Services (ICWS)*, July 2007, pp. 296–303.

[19] A.M.Zaremski and J.M.Wang, “Signature Matching: A Tool for Using Software Libraries,” *ACM Transactions on Software Engineering and Methodology*, vol. 4, no. 2, pp. 146 –170, 1995.

[20] Y.Wang and E.Stroulia, “Semantic Structure Matchng for Assessing Web-Service Similarity,” in *1st Int'l Conf. on Service Oriented Computing (ICSOC03)*. Springer-Verlag, Dec. 2003, pp. 194–207.

[21] W.Hoschek, “The Web Service Discovery Architecture,” in *1st IEEE/ACM Supercomputing Conference*. IEEE Computer Society Press, Nov. 2002, pp. 1–15.

[22] B. F. M. Klusch and K. Sycara, “Automated Semantic Web Service Discovery with OWLS-MX,” in *Fifth Int'l Joint Conf. on Autonomous Agents and MultiAgent Systems (AAMAS)*, Hakodate, Japan, May 2006, pp. 915–922.

[23] G. Meditskos and N. Bassiliades, “Structural and Role-Oriented Web Service Discovery with Taxonomies in OWL-S,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 2, pp. 278 –290, Feb. 2010.

[24] K. Sycara, M. Paolucci, J. Soudry, and N. Srinivasan, “Dynamic discovery and coordination of agent-based semantic Web services,” *Internet Computing, IEEE*, vol. 8, no. 3, pp. 66 – 73, may-jun 2004.

[25] I.Mecar, A.Devlic, and K.Trzec, “Agent-Oriented Semantic Discovery and Match-Making of Web Wervices,” in *8th Int'l Conf. on Telecommunications (ConTEL),*, June 2005, pp. 603–607.

[26] 2011. [Online]. Available: http://www.soa4all.eu/,

[27] R. Aggarwal, K. Verma, J. Miller, and W. Milnor, “Constraint Driven Web Service Composition in METEOR-S,” in *IEEE Int'l Conf. on Services Computing (SCC)*, Sept. 2004, pp. 23 – 30.

[28] 2011. [Online]. Available: http://www.shape-project.eu/,

[29] M. Jokhio, G. Dobbie, and J. Sun, “Towards Specification Based Testing for Semantic Web Services,” in *Software Engineering Conference, 2009. ASWEC '09. Australian*, April 2009, pp. 54 –63.

[30] M. Shaban, G. Dobbie, and J. Sun, “A Framework for Testing Semantic Web Services Using Model Checking,” in *Fourth South-East European Workshop on Formal Methods (SEEFM)*, Dec. 2009, pp. 17 –24.

[31] R.Heckel and L.Mariani, “Automatic Conformance Testing of Web Services,” in *8th Int'l Conf. on Fundamental Approaches to Software Engineering (FASE)*, Edinburgh, UK, April 2005, pp. 34–48.

[32] M.Paradkar, A.Sinha, and et al, “Automated Functional Conformance Test Generation for Semantic Web Services,” in *IEEE Int'l Conf. on Web Services (ICWS)*, Utah, USA, July 2007, pp. 110–117.

[33] J. Z.Li and et al, “Towards a Practical and Effective Method for Web Services Test Case Generation,” in *ICSE Workshop on Automation of Software Test*, Vancouver, Canada, May 2009, pp. 106–114.

[34] C.Ma, C.Du, and et al, “WSDL-Based Automated Test Data Generation for Web Service,” in *IEEE Int'l Conf. on Computer Science and Software Engineering*, Wuhan,China, December 2008, pp. 731–737.

[35] S.Hanna and M.Munro, “An Approach for Specification-based Test Case Generation for Web Services,” in *IEEE/ACS Int'l Conf. on Computer Systems and Applications*, Amman, Jordan, May 2007, pp. 16–23.

[36] S.Noikajana and T.Suwannasart, “Web Service Test Case Generation Based on Decision Table,” in *The Eighth Int'l Conf. on Quality Software (QSIC)*, Oxford, UK, August 2008, pp. 321–326.

[37] M.Lohmann, L.Mariani, and R.Heckel, *A Model-driven Approach to Discovery,Testing, and Monitoring of Web Services*. Heidelberg, Germany: Springer, 2007.

[38] Y.Park, W.Jung, B.Lee, and C.Wu, “Automatic Discovery of Web Services Based on Dynamic Black-Box Testing,” in *33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC)*, Seattle, USA, July 2009, pp. 107–114.

[39] L.Li and C.Wu, “An Abstract GFSM Model for Optimal and Incremental Conformance Testing of Web Services,” in *IEEE Int'l Conf. on Web Services (ICWS)*.

[40] C.Ma, J.Wu, and et al, “Web Services Testing Based on Stream X-machine,” in *IEEE Int'l Conf. on Quality Software (QSIC)*, Zhangjiajie, China, July 2010, pp. 232–239.

[41] A. Flores, A. Cechich, A. Zunino, and M. Usaola, “Testing-Based Selection Method for Integrability on Service-Oriented Applications,” in *Fifth Int'l Conf. on Software Engineering Advances (ICSEA)*, Aug. 2010, pp. 373 –379.

[42] P. Samuel and A. Joseph, “Test Sequence Generation from UML Sequence Diagrams,” in *Ninth ACIS Int'l Conf. on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, SNPD '08.*, Aug. 2008, pp. 879–887.

[43] S.Kansomkeat and W.Rivepiboon, “Automated-Generating Test Case using UML Statechart Diagrams,” in *2003 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on Enablement through Technology*, ser. SAICSIT'03. South African Institute for Computer Scientists and Information Technologists, 2003, pp. 296–300.

[44] L. Castro, M. Francisco, and V. Gulias, “A Practical Methodology for Integration Testing,” in *Computer Aided Systems Theory-EUROCAST 2009*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, vol. 5717, pp. 881–888.

[45] S. Kansomkeat, J. Offutt, A. Abdurazik, and A. Baldini, “A Comparative Evaluation of Tests Generated from Different UML Diagrams,” in *Ninth ACIS Int'l Conf. on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, Aug. 2008, pp. 867–872.

[46] A. Bandyopadhyay and S. Ghosh, “Test Input Generation Using UML Sequence and State Machines Models,” in *Int'l Conf. on Software Testing Verification and Validation, ICST '09.* , April 2009, pp. 121–130.

[47] Z. Li and T. Maibaum, “An Approach to Integration Testing of Object-Oriented Programs,” in *Seventh Int'l Conf. on Quality Software*, Oct. 2007, pp. 268–273.

[48] G. Bernot, M. Gaudel, and B. Marre, “Software Testing based on Formal Specifications: a Theory and a Tool,” *Software Engineering Journal*, vol. 6, no. 6, pp. 387–405, 1991.

[49] I.K.El-Far and J.A.Whittaker, *Model-based Software Testing*. Wiley, 2001.

[50] W.Miao and S.Liu, “A Formal Specification-based Testing Approach to Accurate Web Service Selection,” in *IEEE Int'l Conf. on Asia-Pacific Services Computing (APSCC)*, Jeju, Korea, Dec. 2011, pp. 259–266.

[51] Y. Chen, S. Liu, and F. Nagoya, “An Approach to Integration Testing Based on Data Flow Specifications,” in *Int'l Conf. on Theoretical Aspects on Computing*, vol. 3407, 2004, pp. 235–249.