Validation of the Formal Model of Web Services Applications for Digital Reference Service of Library Information System

Zainab M. Musa, Nordin M. A. Rahman, Julaily A. Jusoh

Abstract—The web services applications for digital reference service (WSDRS) of LIS model is an informal model that claims to reduce the problems of digital reference services in libraries. It uses web services technology to provide efficient way of satisfying users’ needs in the reference section of libraries. The formal WSDRS model consists of the Z specifications of all the informal specifications of the model. This paper discusses the formal validation of the Z specifications of WSDRS model. The authors formally verify and thus validate the properties of the model using Z/EVES theorem prover.

Keywords—Validation, verification, formal, theorem proving.

I. INTRODUCTION

FORMAL proving is the act of showing the correctness of a system with respect to a certain formal specification or property using mathematical methods. After the formal model of a program is built, a variety of properties can be validated over it. The formal specification of a system can also be verified to ensure its correctness and to prove its consistency and completeness using formal verification techniques before system design and implementation [1], [3]. Formal proving is a complete argument of mathematical representation and is used to validate statements about system description. Usually, formal proving can be done manually or automatically using formal method tools such as theorem provers. Developers usually cover a long time while performing the theorem proving process, so there might be a great possibility of mistakes. The proofs are efficient when presented in a user-friendly approach and not in an unreasonable large size. However, a lot of the proofs that are involved in software validation are naturally detail, low-level and repetitious. So we can briefly state that it is unsuitable for human checking. Thus, formal proving supported by tool, do not only reduce the possibility of mistakes but also removes it totally. Hence, the use of support tool is a main factor that can affect the acceptance of formal method practically [1].

The Z specification language is a way of decomposing a specification into small pieces called schemas. Each piece can be linked with comments that give informal explanation about the importance of the formal mathematics. A schema is essentially the formal specification analogous to programming language subroutines that are used to structure a system, where the schemas are used to structure a formal specification. The Z is physically powerful on sets and functions. Generally, Z notation is use for sequential situation and model-based specification. It combines formal and informal description and uses graphical highlighting when presenting specifications [2], [7].

In this paper, we validate the Z specifications of WSDRS model by using theorem proving technique based upon two aspects: the initial state and the pre-conditions. Validation of the initial state is to show that the Z specifications developed were consistent. While the validation of the pre-conditions is to show that the Z specifications developed were complete, consistent and were applied in the right domain.

In order to implement the validation process, a few theorems will be developed for both aspects. Each theorem will be checked using Z/EVES theorem prover tool. The tool will help in reducing time, energy and mistakes compared to manual theorem proving which can be error full and tedious.

II. THEOREM PROVING (DEDUCTIVE VERIFICATION)

Theorem-proving means that systems satisfy their specification, which given by temporal Logic formulas, using deductive (i.e. theorem proving) methods. It involves generating a collection of mathematical proof obligations from a system and its specifications, the truth of which imply conformance of the system to its specification, and discharging these obligations using theorem provers such as interactive theorem provers, automatic theorem provers, satisfiability modulo theories (SMT) solvers. This approach requires the user to understand in detail why the system works correctly, and to convey this information to the verification system, either in the form of a sequence of theorems to be proved or in the form of specifications of system components (e.g. functions or procedures) and subcomponents (such as loops or data structures) [4]. These techniques are not fully automatic and require user interaction and the effective guidance of a theorem-proving tool. It is the most powerful and least restricted verification technique because it can prove anything [5].

Theorem-provers are softwares that help in solving problems and answering questions that involve reasoning. The assistance can either be interactive, where one instructs the program to draw some conclusions, present them to the user, and then to ask for a new set of instructions; or fully automatic, where the program is assigned an entire reasoning task [5].

Zainab Musa Magaji is with the Universiti Sultan Zainal Abidin, Malaysia (e-mail: zeemusa5@gmail.com).
There are several automated theorem-proving tools with different properties used in theorem proving. A few examples are Z/EVES, E, Otter, SETHEO, Vampire, SPASS.

In this research work, the formal verification technique that would be used is theorem proving because it can be implemented on any logic in mathematics notations in Z language. Formal specification is complete and consistent if it can be proven as true. Completeness and consistency for Z specification can be articulated by proving the following aspects: initial state and pre-condition. First, each aspect is presented in a theorem form to be proved and then theorem-proving process will be carried out. Z/EVES theorem prover will be used because it is suitable for formal specification developed in Z language.

III. THEOREM PROVING (INITIAL STATE OF THE WSDRS Z SPECIFICATIONS)

Initial state theorem proving begins with the development of an initial state theorem. The skeleton for an initial state theorem is as:

\[
\text{Theorem TheoremName} \\
\quad \text{E State, InitState}
\]

where Theorem Name: name of the theorem, State: state schema, InitState: Initial state schema [6].

Validation process of the initial state theorem begins with the development of the initial state schema. Next is the development of the initial state theorem depicted in Fig. 1 (a).

Fig. 1 Theorem Initial state for digital reference service web services of LIS

Fig. 1 (b) indicates that the theorem proves to be “true”. This theorem proving process shows that the Z specifications that have been developed and discussed in our previous paper “Formal Specification of Web Services Applications for Digital Reference Services of Library Information System” are consistent and reliable.

IV. THEOREM PROVING PRE-CONDITION

The process of proving the pre-condition theorems of all the operation schemas that were discussed in the previous chapter is presented in this section. Theorem proving is done after all theorems that have been developed are free from any syntax errors.

Pre-condition theorem proving is to show that each operation is applied in the right domain. It begins with the development of pre-condition theorem using the following format.

Theorem TheoremName
A S; in?: N | P. pre Op

where TheoremName: name of the theorem, S: other schemas that are involved in the operation schema, In?: input variable in declaration part of the operation schema, N: type of the input variable in declaration part of the operation schema, P: pre-condition of the operation, Op: operation schema that will be proved to be with or without domain error.

A correct pre-condition theorem, of the above form can be a useful form of documentation of a specification [6]. It is a state before an operation and it relates to a state, which might happen after an operation. State before or after an operation must comply with a specified property.

If the result of a pre-condition theorem proving is true, then the pre-condition operation schema relates accurately to a state after an operation. Otherwise, if the theorem proving is not true then the pre-condition of the operation schema is not consistent and therefore the operation schema or the pre-condition theorem of the schema must be corrected due to unidentified problem in the domain.

There are eighteen pre-condition theorems that have been developed for web services of digital reference service for LIS. These theorems were developed based on the eighteen operation schemas of the Z specifications discussed in our previous paper “Formal Specification of Web Services Applications for Digital Reference Services of Library Information System”. Below are the pre-condition theorems developed for the operation schemas. Each theorem is followed by a screenshot that displays its proof script as validated by the Z/EVES theorem prover using prove by reduce command. All the theorems are proved ‘True’. This result showed that the operations in Z specifications for WSDRS of LIS were applied in the right domain. Moreover, it means that the pre-condition operation schemas of the specifications are related accurately to a state after an operation and have comply with the property been specified in the state schema. So, it can be concluded that the Z specifications that are discussed in the said previous paper are consistent.

1. Theorem select_info_pre

A DRS, patron?: PERSON, active_system_ws!: SYSTEMWebService; action!: ACTION |
\[ \text{patron?} \in \text{users} \]
\[ f \text{select} \text{patron?} = \text{infor_service_ws} \]
\[ f \text{action!} = \text{invoke_infor_service_ws} \]
\[ f \text{active_system_ws!} = \text{infor_service_ws} \approx \text{pre Select_info} \]
2. Theorem  Select_guidpre
A DRS; patron?: PERSON; active_system_ws!: SYSTEM_WEBSERVICES;
action!: ACTION
| patron? ∈ users
f selection patron? = guidance_ws
f action! = invoke_guidance_ws
f active_system_ws! = guidance_ws ∞ pre Select_guid

3. Theorem  Select_directcommpre
A DRS; patron?: PERSON; active_system_ws!: SYSTEM_WEBSERVICES;
action!: ACTION
| patron? ∈ users
f selection patron? = direct_comm_ws
f action! = invoke_direct_com_ws
f active_system_ws! = direct_comm_ws ∞ pre Select_directcomm

4. Theorem  Question_Operationpre
A DRS; patron?: PERSON; patronquestion?: MESSAGE
| patron? ∈ users
f (selection patron? = infor_service_ws
∈ selection patron? = guidance_ws) ∞ pre
Question_Operation

5. Theorem  Answer_simple_conditionpre
A DRS; patron?: PERSON; patronquestion?: MESSAGE;
correctanswer!: MESSAGE;
clearity!: MESSAGE_STATUS
| patron? ∈ users
f clarity! = questionClear
f patronquestion? ∈ questions
f correctanswer! = answerbase patronquestion?
∞ pre Answer_simple_condition
6. Theorem  
Answer_complex_conditionpre
A DRS; patron?: PERSON; patronquestion?: MESSAGE;
correctanswer!: MESSAGE; 
action!: ACTION; clarity!: MESSAGE_STATUS; 
outside ws?: WEBSERVICES; 
outsideAnswerbase?: MESSAGE =&gt; MESSAGE
| patron? ∈ users 
| f clarity! = questionClear 
| f patronquestion? ∈ questions 
| f action! = invoke_other_ws 
| f outside ws? ∈ outsideWebServices 
| f patronquestion? ∈ dom outsideAnswerbase? 
| f correctanswer! = outsideAnswerbase? patronquestion? 
= pre Answer_complex_condition

Fig. 7 Proof script for Answer_complex_conditionpre theorem

7. Theorem  
Request_Clarification_Simplepre
A DRS; patron?: PERSON; requestClarificationMessage?: MESSAGE; 
clearity!: MESSAGE_STATUS; 
outside ws?: WEBSERVICES 
| patron? ∈ users 
| f clarity! = questionNotClear 
| f call_out! = 0 
| f outside ws? ∈ outsideWebServices 
| f outside ws? ∈ ran call_out = pre Request_Clarification_Simple

Fig. 8 Proof script for Request_Clarification_Simplepre theorem

8. Theorem  
Request_Clarification_Complexpre
A DRS; patron?: PERSON; requestClarificationMessage?: MESSAGE; 
outside ws?: WEBSERVICES 
| patron? ∈ users 
| f call_out! = 0 
| f outside ws? ∈ outsideWebServices 
| f outside ws? ∈ ran call_out = pre Request_Clarification_Complex

Fig. 9 Proof script for Request_Clarification_Complexpre theorem

9. Theorem  
Clarification_Simplepre
A DRS; patron?: PERSON; ClarificationMessage?: MESSAGE; 
clearity!: MESSAGE_STATUS; 
outside ws?: WEBSERVICES 
| patron? ∈ users 
| f clarity! = questionNotClear 
| f call_out! = 0 
| f outside ws? ∈ outsideWebServices 
| f outside ws? ∈ ran call_out = pre Clarification_Simple

Fig. 10 Proof script for Clarification_Simplepre theorem

10. Theorem  
Clarification_Complexpre
A DRS; patron?: PERSON; ClarificationMessage?: MESSAGE; 
outside ws?: WEBSERVICES 
| patron? ∈ users 
| f call_out! = 0 
| f outside ws? ∈ outsideWebServices 
| f outside ws? ∈ ran call_out = pre Clarification_Complex
11. Theorem  Constraint_By_Userpre
A DRS; patron?: PERSON; constraintMessage?: MESSAGE  | patron? ε users
∞ pre Constraint_By_User

12. Theorem  Constraint_By_Webservicepre
A DRS; patron?: PERSON; constraintMessage?: MESSAGE  | patron? ε users
∞ pre Constraint_By_Webservice

13. Theorem  Constraint_Reply_By_Userpre
A DRS; patron?: PERSON; constraintReplyMessage?: MESSAGE; constraintStatus?: MESSAGE_STATUS; acceptanceFlag?: RESPONSE  | patron? ε users
f (if acceptanceFlag? = True
then constraintStatus? = Accepted
else constraintStatus? = NotAccepted) ∞ pre Constraint_Reply_By_User

14. Theorem  Constraint_Reply_By_Webservicepre
A DRS; patron?: PERSON; constraintReplyMessage?: MESSAGE; constraintStatus?: MESSAGE_STATUS; acceptanceFlag?: RESPONSE  | patron? ε users
f (if acceptanceFlag? = True
then constraintStatus? = Accepted
else constraintStatus? = NotAccepted) ∞ pre Constraint_Reply_By_Webservice

15. Theorem  Action_Request_Operationpre
A DRS; patron?: PERSON; actionRequestMessage?: MESSAGE; actionRequest?: ACTION  | patron? ε users
f (actionRequest? = suspend_process_until_another_time
  actionRequest? = resume_process
  actionRequest? = reset_activity_time
  actionRequest? = close_transaction
  actionRequest? = request_for_status_report)
∞ pre Action_Request_Operation

16. Theorem  Status Reply Operation
A DRS; patron?: PERSON; statusReplyMessage?: MESSAGE;
actionRequestStatus?: MESSAGE_STATUS;
statusReply?: RESPONSE
| patron? ∈ users
  f (if statusReply? = Success
    then actionRequestStatus? = Accepted
    else actionRequestStatus? = NotAccepted)
∞ pre Status_Reply_Operation

17. Theorem  Connection Operation
A DRS; patron?: PERSON; Select_Comm?: PERSON CON\ COMM MEDIUM; action?: ACTION
| patron? ∈ users
  f selection patron? = direct_comm_ws
  f patron? ∈ dom Select_Comm?
  f (if Select_Comm? patron? = Call
    then action? = connect_video_and_voice_call_service
    else action? = connect_text_chat_service) ∞ pre Connection_Operation

18. Theorem  Error Operation
A DRS; patron?: PERSON; report!: RESPONSE; cleartiy?: MESSAGE_STATUS
| patron? ∈ users f cleartiy? = questionEmpty f report! = questionInvalid
∞ pre Error_Operation

Figs. 2-19, it is clear that the proving of all the precondition theorems returns true. This means that the specifications we have developed which are defined in our previous paper titled “Formal Specification of Web Services Applications for Digital Reference Services of Library Information System” are correct, consistent and are applied in the right domain. This means that the model, which we develop for web services application for digital reference services of the library information system, is valid. Thus, it can be adopted and adapted by any library to provide a networked asynchronous digital reference service across the globe.

V. CONCLUSION

This paper demonstrated that all the theorems developed are proved to be true and are found to be error free, consistent and in the correct domain. Therefore, the Z specifications for web
service applications for digital reference service of LIS are reliable as such they can be transformed into programming. At the same time, the ambiguity problem of informal specifications that may occur to the developers is overcome. Therefore, program development time and errors occurrence will decrease.

REFERENCES


