Int J Syst Assur Eng Manag (December 2023) 14(6):2485–2496
https://doi.org/10.1007/s13198-023-02098-6

ORIGINAL ARTICLE

Formally modeling and verifying a software component retrieval system using mCRL2

Nisha Pal[1] · **Dharmendra Kumar Yadav**[1]

Received: 21 October 2021 / Revised: 21 May 2023 / Accepted: 8 August 2023 / Published online: 27 August 2023

Abstract Software reuse is the process of building a new software application by using existing software components. Component-based software engineering is one of the approaches that is based on reusability concepts. It helps to improve the software quality and performance. Effective storage and retrieval scheme are two of the essential factors in the software reuse process. It reduces maintenance costs and easily upgrades a large and complex software system. The selection of appropriate components becomes more complex due to unexpected requirements. To overcome these problems, many researchers have developed different types of software component storage and retrieval techniques. However, all the techniques do not give the proper satisfaction of the developer requirements. In this paper, we proposed a software component retrieval schema which is built on different group of repositories namely metadata repository, description repository, component repository, and ontology repository. These repositories give semantic information related to the component. For the correctness of the proposed system, we proposed a formal model of that system which verifies the correct flow of a sequence of these repositories for finding the desired component. Formal specification and verification technique helps to determine the correct matching component from the repository. This new system improves searching results for the developer to develop the software project. We used mCRL2 process algebra for describing the behavior of storing and retrieving system. The requirement of the proposed system has written in a modal mu-calculus. It has been verified by using the mCRL2 toolset.

Keywords CBSE · Metadata repository · mCRL2 tool · Formal verification · Ontology repository · Formal methods

1 Introduction

Client's expectations have been increasing continuously for the recent years. This situation affects the quality and productivity of the software, which increases the complexity of the software system. Therefore, the developer needs an effective technique to achieve the desired result. CBSE is one of the reuse techniques that help to achieve the above goal. It reduces development cost and effort, and increases the quality and productivity of the software system. In this approach, the software is developed by selecting appropriate components and then assembling these components with well-defined architecture. The selection of the desired component becomes more complex due to the rapidly changing end user requirements. For any software applications to be successful it is very important that the appropriate component is selected which minimizes the developer's effort and development time (Yahlali 2022). A software repository is one of the essential elements for successfully developing the software product. It is used for storing, retrieving and managing a large number of software components. It supports the effective classification schema, standard component framework, and allocates the desired software resource (Guo et al. 2000; Xu et al. 2020). Repositories have designed to meet the rapidly changing demand of the software development organization. The storage and retrieval mechanism of the repository in CBSE's are the main key research area.

✉ Nisha Pal
2016rcs05@mnnit.ac.in

Dharmendra Kumar Yadav
dky@mnnit.ac.in

[1] Motilal Nehru National Institute of Technology Allahabad, Prayagraj, Uttar Pradesh 211004, India

Researchers have proposed several storage and retrieval approaches for acquiring desired components from repository (Aman et al. 2014). However, these approaches retrieve a limited set of components from the repository that do not meet the user's requirement. Because, these techniques do not understand the user query properly and also do not give proper meaning of the software component. Due to this, the important information related to the software component is omitted. Hence, software repositories need a technique that provides the semantic interrelation among the components for effectively storing and retrieving. This technique helps in locating and comparing different components in repository.

In this paper, we proposed a software component storage and retrieval system with the help of four types of repository such as metadata, description, component, and ontology. These repositories provide exact behavior and domain knowledge of the component and help to store, search and reuse these components on demand. These repositories help to retrieve the target software component from the repository. Hence, users get the generic or acceptable software component from the repository. For the correctness of the proposed system, we have proposed a formal model of proposed system that verifies the correct flow of a sequence of these repositories for finding the desired component (Almeida et al. 2011). Formal specification and verification techniques help to determine the correct matching component from the repository. We represent the flow of a sequence of the proper selection of software components and reduce the mismatch problem for selecting the appropriate component from the repository through the formal method. It helps to automate the search and retrieval mechanism. To describe the proposed system's behavior, we have used mCRL2 formal language, and through the mu-calculus we have represented the requirement of the proposed model. In this paper, we design a system that supports automatically identifying, comparing and retrieving the desired reusable software component from the repository. This proposed model has been verified by using the mCRL2 toolsets (Groote et al. 2008; Bunte et al. 2019).

The rest of the paper is organized as follows: Sect. 2 presents the motivation of the work. Section 3 explores the related work. Section 4 provides the overview of mCRL2 tool sets. Section 5 discusses the software component storing and retrieving system. Section 6 describes the modeling and verification of software component retrieval system. Finally some concluding and future work remarks are drawn in Sect. 7.

2 Motivation

Component-Based Software Development (CBSD) is an approach in which every software component is an independent executable unit that is tested and deployed, and stored in a component repository. Hence, these components can be assembled together to form a new software. The main objective of CBSE is to reuse the components for building more complex and large systems with no or minor modification. In the CBSE technique, the developer spends more time and effort for selecting relevant reusable components from a large repository. If search and retrieval techniques are not effective, users may not be able to get the desired component because many components have almost the same name and functionality. They get multiple and ambiguous components. To the success of any software application an effective scheme is needed for organizing and describing the repository properly at the different levels of detail, and also provided mechanism for retrieving and storing these components to the repository. Hence, the software industry needs for an effective storage and retrieval mechanism that provides the exact software components and delivers the quality software for the end user.

3 Related work

Effective storing and retrieving schema are the essential for locating and comparing any desired services from the commercial repository. Since in many fields such as software engineering, IoT, image processing, soft computing, healthcare sector, etc., exact information retrieval according to the end user is difficult task (Kavitha and Vidhya Saraswathi 2021; Gavrilović and Mishra 2021; Shi et al. 2021; Vasanthi et al. 2021; El-Ansari et al. 2021). Because due to a large number of available services, to choose the appropriate services that satisfying the end user requirement is time consuming process. In the case of component base software engineering selecting the appropriate components from the repository is a main process for the success or failure of building a software system. Since the repository is very large and selection of components is difficult due to continuously change the component's demand. Therefore, to understand the customer demands is an important task. There are many techniques exist that help to search and store a component in a repository effectively (Yahlali 2022; Lucredio et al. 2004).

In Aman et al. (2014), Bawa and Kaur (2016), the authors provided a systematic review of different kinds of storing and retrieving techniques. In this paper, the authors discussed various algorithms for storing and retrieving the exact components with well-structured repositories. In Chatterjee and Rathi (2014), the author discussed different storage and retrieval techniques, but the main focus is on the concept of associated with signature matching and keywords base technique. In Gajala (2013), the author described another integrated technique based on attribute value and faceted classification schema. This technique improves the

retrieving technique, reduces cost and investment on the user requirement. Lucredio et al. (2004) presented an effective retrieving technique that supports metric indexing structure and similarity measures between components. This technique overcomes the problem of component retrieval and offer reductions in search time, and increasing the reuse and saving effort.

Dixit and Saxena (2009) presented another component retrieval technique that using the genetic algorithm. In this paper, the authors discussed the major issue related to the component selection from the repository. Using the genetic algorithm, they tried to minimize the gap between components' needs and availability. In Nie and Zhong (2009), the authors discussed the component retrieval technique based on ontology. In this paper, the authors presented a component description model based on ontology features and the retrieval method of user interest. This retrieval method provides a clear definition of component that easily understands by the user. This can be processed by computer automatically. In Desouky and El-Khouly (2015), the authors have proposed an overview of the key research on software component clustering and explores the need for an effective search mechanism in order to enable software component as a service in the cloud. In Zhang (2007), Chythanya and Reddy (2021), the authors presented a survey of different retrieval techniques and introduced the concepts of some commercial repositories. According to the authors, systematic reuse activities help reduce the duplication of effort, imposed standardizations, and ensure the correct retrieving component from the repository on user demand. In Guo et al. (2000), Bakshi and Bawa (2013), the authors presented a survey of effective search and retrieval methods and techniques and various software repositories. According to the authors, an effective retrieval schema is essential for locating, storing, and comparing the component in a repository. The authors discussed different repository requirements based on different domain-specific communities.

In Chapman et al. (2009), the authors discussed the approaches and challenges of the metadata repository. In this paper, the authors discussed the repository efforts at three major universities. They discussed the challenges of creation, management, and access to the repository. The authors also discussed local strategies for metadata creation, user behavior, and the aggregation of heterogeneous metadata. In Bibi et al. (2022), the authors have build a conceptual framework that retrieves code snippets and implements a targeted code retrieval method. In order to obtain more precise and pertinent computer programming code for a query (i.e., search results based on query interpretation), this research will use ontologies and construct a web application. At the end of this paper data has been extracted from GitHub repositories in order to rate the results of a Natural Language Processing (NLP) search for source code. The correctness of the component obtained from the repository is evaluated using performance metrics like F-measure, precision, or recall.

In Singh (2013), the authors discussed software component retrieval based on metadata and ontology repository. This paper presents a meta-data model and effective storage and retrieval system of software components that considers semantic domain information based on ontology. This system provides an effective storage and retrieval schema but does not guaranty to give the exact appropriate component from the software repository. It needs to verify the correctness of the system. In Chang et al. (1997), the authors proposed an approach to reuse-based software development using formal methods. In this paper, authors formalized each component with the help of a set of predicates. The user retrieves the component from the repository using either keywords or predicates and integrates the components with the designed system. Using this approach user try to find out the best component from the repository that meets the requirements. In Guo et al. (1999), the authors discussed an automated signature matching retrieval technique for searching and storing the component from the software component repository. The signature matching technique helps to roll out the candidate component automatically and reduces the effort for searching the large amounts of candidate components during development time.

In Lewczuk (2021), the authors discussed the reliability and dependability of Automated Storage and Retrieval Systems (ASRS), which are viewed as solutions with high technical reliability. This literature study is offered along with a definition of the terms reliability and dependability as they relate to logistics systems like ASRS. Based on this, the elements influencing the dependability of ASRS are discussed in a way that hasn't been covered in the conversation so far. The main purpose of the study is to ascertain how the aforementioned variables affect the reliability and performance of ASRS as well as the importance of the OTIFEF (on-time, in-full, error-free parameter), which includes logistics time metrics and timely warehouse task execution.

4 Overview of mCRL2 tool set

The analysis of the distributed and parallel system is a complex task that requires some tools. These types of systems are difficult to design correctly. The mCRL2 language and

toolsets are formal methods based approach that supports the analysis of distributed applications. mCRL2 stands for micro Common Representation Language 2. It is a specification language that can be used to specify and analyze the behaviour of distributed and concurrent systems. It is based on a mathematical approach. It is allowing automatic analysis and verification of these systems. Hence, it provides the correct specification and verification of the system (Cranen et al. 2013). The mCRL2 is extended of the algebra of communicating processes (ACP) with abstract data types (Hojjat et al. 2011). This mCRL2 formal language is the collection of a rich set of abstract data types. It contains a toolset for analyzing and verifies the specification of the system. The researchers at the Eindhoven University of Technology was developed this mCRL2 language and toolset in 2006 (Bunte et al. 2019). The basic syntax of the mCRL2 specification language is given below.

$$\mathrm{p}::=\mathrm{a}(\mathrm{d}1,...,\mathrm{dn})|\tau|\delta|\mathrm{p}+\mathrm{p}|\mathrm{p}.\mathrm{p}|\mathrm{p}||\mathrm{p}|\tau_I(\mathrm{p})|\partial_H(\mathrm{p})|\nabla_V(p)|$$
$$\Gamma_C(\mathrm{p})|\sum_{d:D}\mathrm{p}|\mathrm{c}\rightarrow\mathrm{p}0\diamond\mathrm{p}1$$

Here, “a" is a basic action of a process with some number of arguments d1,..., dn. “τ" action is the internal action with no any parameter. “δ" represents the deadlock process where no further transition is possible. “+” is a choice operator that may follow different interactions patterns in a non-deterministic environment. “.” is a sequential operator composed process sequentially. “||” is a parallel composition operator in which two processes communicate concurrently. “τ_I” is an abstraction operator that renames actions in I. “∂_H” is encapsulation operator that specifies the set of actions in H that are not allowed to occur. “∇_V" is a restriction operator that allows only the multi-action in set V to occur in p. “Γ_C” is a communication operator that depicts possible communications in a system and the resulting actions. “$\sum_{d:D}$” is the summation operator that provides sum of the non-deterministic variable D. “c $\rightarrow$ p0 $\diamond$ p1” is a conditional operator where process p0 will happen if condition c evaluates to true otherwise, p1 will take over. In mCRL2 languages, a number of built-in data types exist, such as integers, real, sets, lists, sort, cons, struct, maps, equ, and functions that support implementation. A new process is defined by proc and initialize by the keyword init.

The demonstration and analysis procedures of mCRL2 tool as a model-checker has been shown in Fig. 1. The mCRL2 toolset allows verifying the specification of complex system designs that are formally described in the mCRL2 language (Groote et al. 2007; Man and van der Wulp 2008). It transforms a mCRL2 specification of the system into the corresponding linear representation. Then simulator can simulate the behavior of a linear representation interactively. After this, a space generator can be generating a state space from a linear representation. Furthermore, the generated state space of a linear representation can be read or visualization with the help of backend tools (CADP). Finally, the requirement for system behavior is verified (that is specified by the mCRL2 specification) by using a parameterized Boolean equation system (PBES) that contains a symbolic specification of the system’s behavior. The requirements or properties of the system are formulated with the help of modal mu-calculus.

5 Software component storing and retrieving system

Component-based software engineering has offered many facilities to develop a large and complex software system. However, these advancements produce many challenges, one of them being to develop an effective storage and retrieval mechanism. Hence, the software industry searches for an effective storage and retrieval schema that provides the exact software components and delivers the best results. Storage and retrieval schema is essential for the software component reuse process. Many authors have proposed different storage and retrieval techniques, but none of the techniques have fulfilled the desired demand (Gupta and Kumar 2013). Because these techniques provide a limited set of components. Every technique has its own advantages and disadvantages. The proposed system have four repositories such as metadata, description, component, and ontology repository. This new approach helps to locate the appropriate component from the repository and develop the software application. The workflow diagram of the proposed system has shown in Fig. 2.

All these repositories help to identify the function, static and dynamic behavior, domain knowledge, and working environment of the software component. It provides the component’s accurate behavior that helps store, reuse, and search the component on demand. The metadata repository stores the component’s domain knowledge and helps provide the accurate description of the component. The description repository provides a detailed description of the components such as function, interface, language, applied

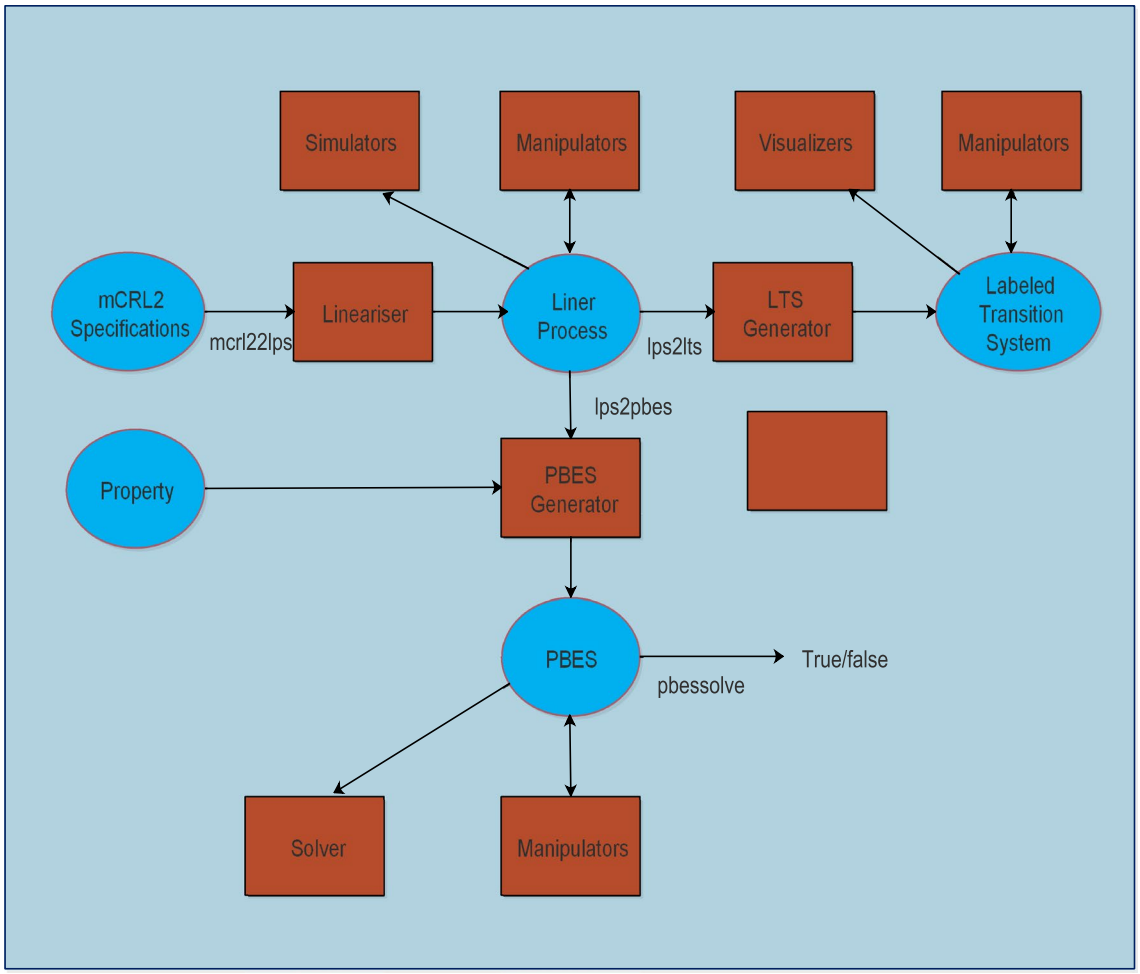

Fig. 1 The basic work-flows of mCRL2 toolset

environment and so on. The component repository helps to provide appropriate formulated accurate query terms for the desired component. The ontology repository captures the semantic information related to the component and provides the suitable characteristic that helps to retrieve the desired components. Therefore, the effective storage and retrieval schema captures the correct domain information related to the component, which helps analyze the desired component's functionality and behavior. It also gives semantic information related to the desired component.

6 Modeling and verification of software component retrieval system

Model-based formal approaches are used for specifying the behavior of a retrieval system application. Modeling is especially helpful for all aspects of systems development, including the need to understand and organize the specifications, and supports the automation for implementation of system. We have created the modeling system for specifying the characteristics of the proposed system through a modeling language such as mCRL2 languages. We have written the property of proposed system in modal mu-calculus. The proposed model solves the challenge for the retrieving and storing component. It provides the correct flow of the sequence for selecting the appropriate component in a repository.

In proposed system, a user provides the input query with the help of the query interface. The query interface firstly gives the general query terms related to the desired component through the original query module and this module matches these terms in the metadata repository. If match is successfully, then the metadata repository sends the accurate

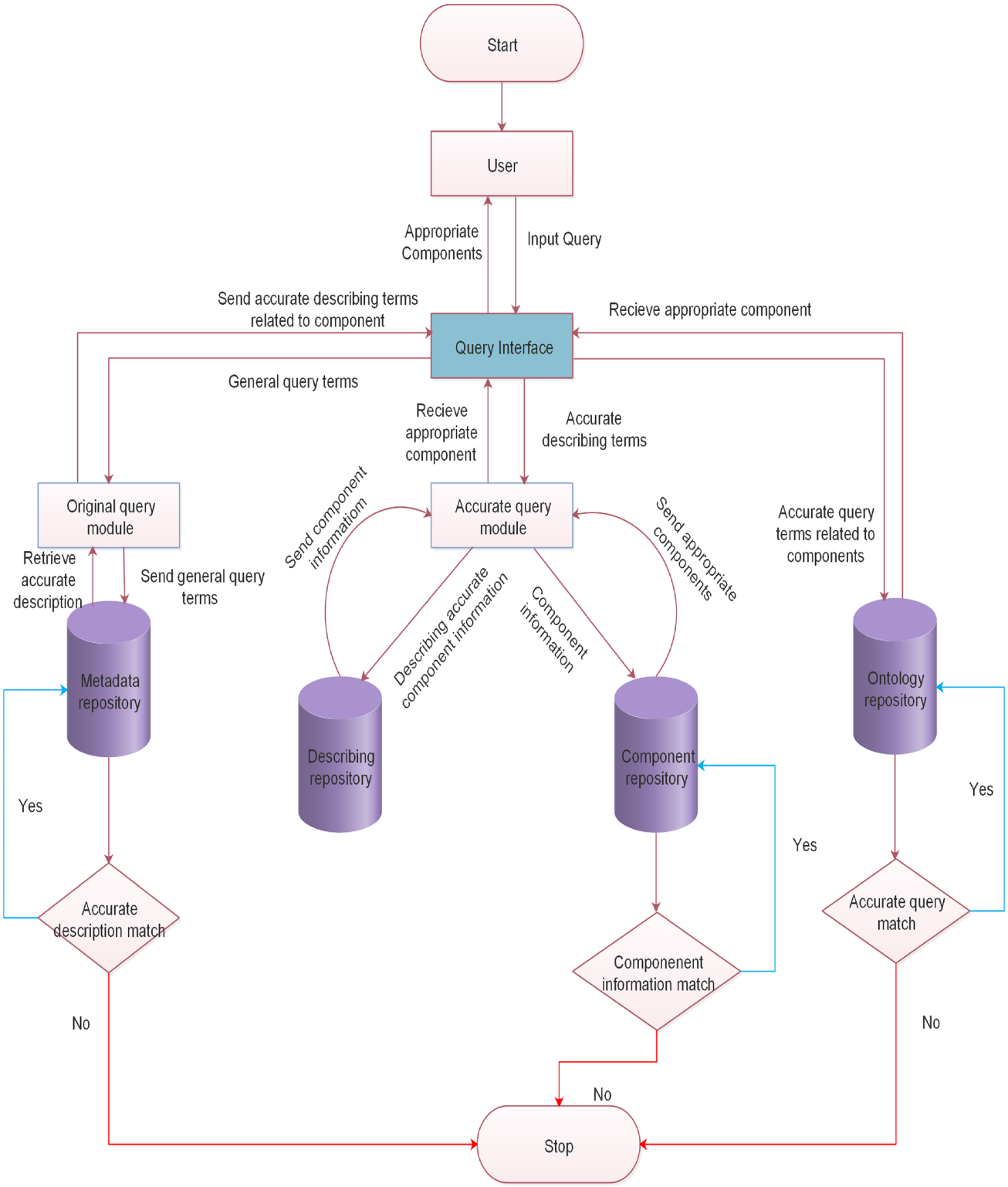

Fig. 2 Software Component retrieval system

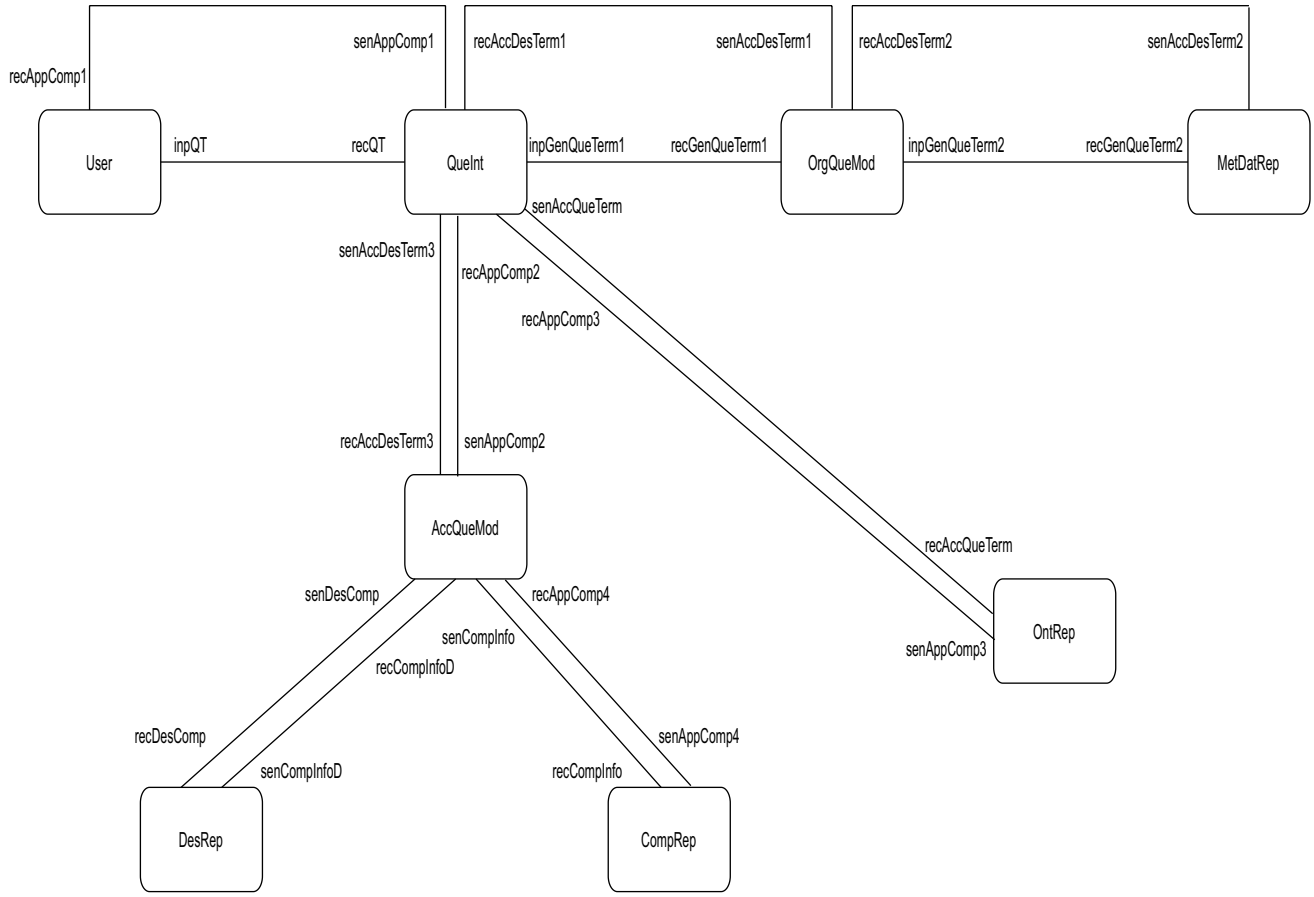

Fig. 3 Architecture of proposed retrieval system

description terms for refining the user query. After this, query interface sends the accurate description terms related to the components to the accurate query module. The accurate query module sends these descriptions in the description repository. Description repository checks the functionality and detailed description of the component and sends the detailed information related to the component to the accurate query module. The accurate query module sends the accurate component information to the component repository. This repository sends the information related to filter components to the accurate query module. Then, accurate query module provides appropriate software components to the query interface and the user downloads these components from the repository.

If the accurate description of the component does not match in the component repository then these components are searched in the expert description repository which is called as ontology repository. The query interface searches the desired component by the accurate query term in the ontology repository. These accurate query terms are based on certain rules or formulas. If components match according to accurate query terms then the user gets the appropriate component and can downloads it from the repository. If the query does not match then it needs to create the component from the scratches and stores it in the component repository. This proposed model (see Fig. 3) has been verified by using the mCRL2 toolsets. The mCRL2 toolsets supported LTSGraph tool that generate the state space diagram of the proposed model as shown in Fig. 4. LTS tool provides the node-link graph that helps to visualized all possible state of proposed system and their transition relations.

The architecture specification of proposed system using mCRL2 is given below:

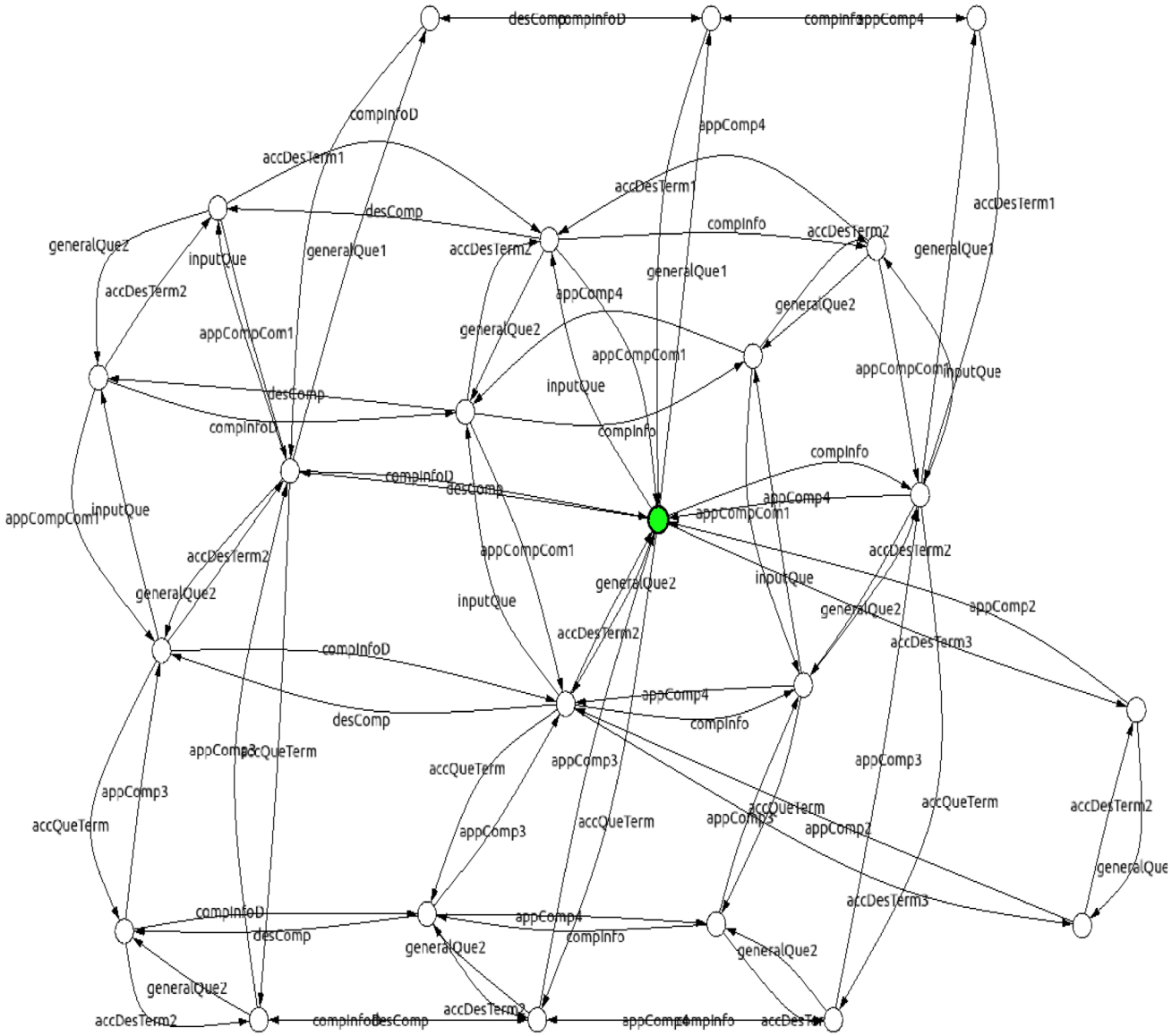

Fig. 4 State space diagram generated by mCRL2 for software component retrieval system

```
act
    inpQT,recAppComp1,recQT,senAppComp1,inpGenQueTerm1,recAccDesTerm1,senAccDes
    Term3,recAppComp2,senAccQueTerm,recAppComp3,recGenQueTerm1,senAccDesTerm1,
    inpGenQueTerm2,recAccDesTerm2,recGenQueTerm2,senAccDesTerm2,senDesComp,
    recCompInfoD,senCompInfo,recAppComp4,recAccDesTerm3,senAppComp2,recDesComp,
    senCompInfoD,recCompInfo,senAppComp4,recAccQueTerm,senAppComp3,inputQue,
    appCompCom1,generalQue1,generalQue2,accDesTerm1,accDesTerm2,accDesTerm3,
    accQueTerm,appComp2,appComp3,desComp,compInfoD,compInfo,appComp4;

proc
     User = inpQT.recAppComp1.User;
     QueInt = recQT.senAppComp1.QueInt + inpGenQueTerm1.recAccDesTerm1.QueInt +
              senAccDesTerm3.recAppComp2.QueInt + senAccQueTerm.recAppComp3.
              QueInt;
     OrgQueMod = recGenQueTerm1.senAccDesTerm1.OrgQueMod +
                 inpGenQueTerm2.recAccDesTerm2.OrgQueMod;
     MetDatRep = recGenQueTerm2.senAccDesTerm2.MetDatRep;
     AccQueMod = senDesComp.recCompInfoD.AccQueMod + senCompInfo.recAppComp4.
                 AccQueMod + recAccDesTerm3.senAppComp2.AccQueMod;
     DesRep = recDesComp.senCompInfoD.DesRep;
     CompRep = recCompInfo.senAppComp4.CompRep;
     OntRep = recAccQueTerm.senAppComp3.OntRep;
init
     allow({appCompCom1, generalQue1, generalQue2, accDesTerm1, accDesTerm2,
            accDesTerm3,accQueTerm, appComp2, appComp3, desComp, compInfoD,
            compInfo, appComp4, inpQT, recQT},
     comm({senAppComp1 | recAppComp1 -> appCompCom1, inpGenQueTerm1 |
           recGenQueTerm1 -> generalQue1, inpGenQueTerm2 | recGenQueTerm2 ->
           generalQue2, recAccDesTerm1 | senAccDesTerm1 -> accDesTerm1,
           recAccDesTerm2 | senAccDesTerm2 -> accDesTerm2,recAccDesTerm3 |
           senAccDesTerm3 -> accDesTerm3, senA ccQueTerm | recAccQueTerm ->
           accQueTerm, senAppComp2 | recAppComp2 -> appComp2, senAppComp3 |
           recAppComp3->appComp3, senDesComp|recDesComp -> desComp, recCompInfoD
           | senCompInfoD -> compInfoD, recCompInfo|senCompInfo-> compInfo,
           senAppComp4|recAppComp4 -> appComp4},
     User||QueInt||OrgQueMod||AccQueMod||MetDatRep||DesRep||CompRep||OntRep
   )
);
```

Some Essential Properties of the Proposed System is given below:

1. There can be an independent input query forward from the user (P1).
2. After a request forward through the input query to the query interface, eventually there will be get a appropriate component (P2).
3. After an input query request has been made from the user, systems' response will be either get the appropriate component or request to the origin query module (P3).
4. After an input query request has been made from the user, systems' response will be either get the appropri-

Table 1 Abbreviations and descriptions of proposed model

S.No.	Abbreviations	Descriptions
1	QueInt	Query interface
2	OrgQueMod	Original query module
3	AccQueMod	Accurate query module
4	MetDatRep	Metadata repository
5	DesRep	Description repository
6	CompRep	Component repository
7	OntRep	Ontology repository
8	inpQT	Input query term
9	recQT	Receive query term
10	senAppComp	Send appropriate component
11	recAppComp	Receive appropriate component
12	inpGenQueTerm	Input general query term
13	recGenQueTerm	Receive general query term
14	recAccDesTerm	Receive accurate description term
15	senAccDesTerm	Send accurate description term
16	senAccQueTerm	Send accurate query term
17	recAccQueTerm	Receive accurate query term
18	senDesComp	Send description of component
19	recDesComp	Receive description of component
20	senComInfoD	Send component information in detail
21	recCompInfoD	Receive component information in detail
22	senCompInfo	Send component information
23	recCompInfo	Receive component information

ate component or request to the ontology repository (P4).

5. It is not possible to have both response and request to the original query module after request from the user (P5).
6. Immediately after inserting the input query from user, query interface either send the appropriate component or send the general query term to the original query module (P6).
7. Immediately after request from query interface, original query module either send the accurate describing term of component or input the accurate description of the component to the describing repository (P7).
8. There is a possibility of a request from user, followed by request generated from the query interface, followed by response from the ontology repository, finally response from the query interface. This sequence of action may repeat infinitely (P8).
9. There is a possibility of a request from user, followed by request generated from the query interface, followed by request generated from the original query module, followed by response from the metadata repository, followed by response from the original query module, finally response from the query interface. This sequence of action may repeat infinitely (P9).
10. There is a possibility of a request from query interface, followed by request generated from the accurate query module, followed by response from the describing repository, followed by request generated from the accurate query module, followed by response from the component repository, finally response from the accurate query module. This sequence of action may repeat infinitely (P10).
11. The proposed system is deadlock free (P11).

$$prop\ P_1 = \langle\ \text{true}\ *\ .\ \text{inpQT}\ \rangle true$$

$$prop\ P_2 = [\ \text{true}\ *\ ][\ \text{inpQT}\ ]\langle\ \text{true}\ *\ .\ \text{appCompCom1}\ \rangle true$$

$$prop\ P_3 = [\ \text{true}\ *\ ][\ \text{recQT}\ ]\langle\ \text{true}\ *\ .\ \text{appCompCom1}\ \rangle true\ \ ||\ \ [\ \text{recQT}\ ]\langle\ \text{true}\ *\ .\ \text{generalQue1}\ \rangle true$$

$$prop\ P_4 = [\ \text{true}\ *\ ][\ \text{recQT}\ ]\langle\ \text{true}\ *\ .\ \text{appCompCom1}\ \rangle true\ \ ||\ \ [\ \text{recQT}\ ]\langle\ \text{true}\ *\ .\ \text{accQueTerm}\ \rangle true$$

$$prop\ P_5 = [\ \text{true}\ *\ ][\ \text{recQT}\ ]\langle\ \text{true}\ *\ .\ \text{appCompCom1}\ \rangle true\ \ \&\&\ \ [\ \text{recQT}\ ]\langle\ \text{true}\ *\ .\ \text{generalQue1}\ \rangle true$$

$$prop\ P_6 = (\langle\ \text{true}\ *\ .!\text{generalQue1}\ \rangle true\ \ \&\&\ \ \langle\ \text{true}\ *\ .\ \text{appCompCom1}\ \rangle true)\ \ ||\ \ (\langle\ \text{true}\ *\ .\ \text{generalQue1}\ \rangle true\ \ \&\&\ \ \langle\ \text{true}\ *\ .!\text{appCompCom1}\ \rangle true)$$

$$prop\ P_7 = (\langle\ \text{true}\ *\ .!\text{accDesTerm1}\ \rangle true\ \ \&\&\ \ \langle\ \text{true}\ *\ .\ \text{generalQue2}\ \rangle true)\ \ ||\ \ (\langle\ \text{true}\ *\ .\ \text{accDesTerm1}\ \rangle true\ \ \&\&\ \ \langle\ \text{true}\ *\ .!\text{generalQue2}\ \rangle true)$$

$$prop\ P_8 = \langle\ \text{true}\ *\rangle\ \ mu\ \ X\ \ .\ \ \langle\ \text{recQT}\ \rangle\langle\ \text{accQueTerm}\ \rangle\langle\ \text{accComp3}\ \rangle\langle\ \text{appCompCom1}\ \rangle X$$

$$propP_9 = \langle\ \text{true}\ *\rangle\ \ mu\ \ Y\ \ .\ \ \langle\ \text{recQT}\ \rangle\langle\ \text{generalQue1}\ \rangle\langle\ \text{generalQue2}\ \rangle\langle\ \text{accDesTerm2}\ \rangle\langle\ \text{accDesTerm1}\ \rangle\langle\ \text{appCompCom1}\ \rangle Y$$

$$prop\ P_{10} = \langle\ \text{true}\ *\rangle\ \ mu\ \ Z\ \ .\ \ \langle\ \text{recAccDesTerm3}\ \rangle\langle\ \text{desComp}\ \rangle\langle\ \text{compInfoD}\ \rangle\langle\ \text{compInfo}\ \rangle\langle\ \text{appComp4}\ \rangle\langle\ \text{appComp2}\ \rangle Z$$

$$prop\ P_{11} = [\ \text{true}\ *\ ]\langle\ \text{true}\ \rangle true$$

Abbreviations Table 1 represents the abbreviations and descriptions of proposed model

7 Conclusion and future scope

The component-based software system provides more functionality to develop a complex and large software system. The essential step of this approach is to reuse the software components already developed in the repository to build new software systems. A software repository is not only a collection of reusable resources, but also helps to classify, store and retrieve reusable software components on demand. The proposed work aims to provide an effective software component retrieval system for locating and storing required components from the

repository. The proposed system is based on metadata and ontology repositories. The combination of these repositories proposes a new type of system for processing software development. It provides the semantic information of the component that helps to search for a more relevant component on demand. Formal methods help to prove the correctness of the proposed system. It verifies the correct flow of a sequence of these repositories for finding the appropriate component. The proposed system helps to minimize the development cost and effort at the design time of the software system.

In the future work, we will be implemented a web-based reusable component repository with several components using component-based software engineering approach and try to resolve application design issues using CBSE and elaborated some design guidelines for CBSE.

Funding This research received no specific grant from any funding agency.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Human or animals rights Research does not involve human participants and/or animals.

Informed consent Research does not involve humans.

References

Almeida JB, Frade MJ, Pinto JS, de Sousa SM (2011) An overview of formal methods tools and techniques. In: Rigorous software development, pp 15–44

Aman N et al (2014) Component retrieval techniques-a systematic review. Int J Sci Eng Res 5(1):1699–1706

Bakshi A, Bawa S (2013) A survey for effective search and retrieval of components from software repositories. IJERT Int J Eng Res Technol

Bawa R, Kaur I (2016) Algorithmic approach for efficient retrieval of component repositories in component based software engineering. Indian J Sci Technol 9(48):27–70

Bibi N, Rana T, Maqbool A, Alkhalifah T, Khan WZ, Bashir AK, Zikria YB (2022) Reusable component retrieval: a semantic search approach for low resource languages. In: Transactions on Asian and low-resource language information processing

Bunte O, Groote JF, Keiren JJ, Laveaux M, Neele T, de Vink EP, Wesselink W, Wijs A, Willemse TA (2019) The mCRL2 toolset for analysing concurrent systems: improvements in expressivity and usability. In: Tools and algorithms for the construction and analysis of systems: 25th international conference, TACAS 2019, Held as part of the european joint conferences on theory and practice of software, ETAPS 2019, Prague, Czech Republic, April 6–11, 2019, Proceedings, Part II 25, Springer, pp 21–39

Chang C-T, Chu WC, Liu C-S, Yang H (1997) A formal approach to software components classification and retrieval. In: Proceedings twenty-first annual international computer software and applications conference (COMPSAC'97). IEEE, pp. 264–269

Chapman JW, Reynolds D, Shreeves SA (2009) Repository metadata: approaches and challenges. Cat Classif Q 47(3–4):309–325

Chatterjee R, Rathi H (2014) A prolific approach towards automating component repository search. In: 2014 Seventh international conference on contemporary computing (IC3). IEEE, pp 547–552

Chythanya NK, Reddy C (2021) A survey on mechanisms of reusable code component retrieval from component repository. In: 2021 2nd international conference on smart electronics and communication (ICOSEC). IEEE, pp 764–769

Cranen S, Groote JF, Keiren JJ, Stappers FP, De Vink EP, Wesselink W, Willemse TA (2013) An overview of the mCRL2 toolset and its recent advances. In: International conference on tools and algorithms for the construction and analysis of systems. Springer, pp. 199–213

Desouky E, El-Khouly M (2015) A survey on clustering software components for efficient component retrieval. J Inf Soc, ISSN 2356–9328

Dixit A, Saxena P (2009) Software component retrieval using genetic algorithms. In: 2009 International conference on computer and automation engineering. IEEE, pp 151–155

El-Ansari A, Beni-Hssane A, Saadi M, El Fissaoui M (2021) Papir: privacy-aware personalized information retrieval. J Ambient Intell Humaniz Comput 12:9891–9907

Gajala G (2013) Implementation of attribute value & faceted value classification scheme for constructing reuse repository. Int J Comput Trends Technoly (IJCTT), vol 4(1). ISSN 2231–2803

Gavrilović N, Mishra A (2021) Software architecture of the internet of things (IoT) for smart city, healthcare and agriculture: analysis and improvement directions. J Ambient Intell Humaniz Comput 12(1):1315–1336

Groote JF, Keiren J, Mathijssen A, Ploeger B, Stappers F, Tankink C, Usenko Y, van Weerdenburg M, Wesselink W, Willemse T et al. (2008) The mCRL2 toolset. In: Proceedings of the international workshop on advanced software development tools and techniques (WASDeTT 2008), p 53

Groote JF, Mathijssen A, Reniers M, Usenko Y, Van Weerdenburg M (2007) The formal specification language mCRL2. In: Dagstuhl Seminar Proceedings, Schloss Dagstuhl-Leibniz-Zentrum für Informatik

Guo J et al. (1999) Toward automated retrieval for a software component repository. In: Proceedings ECBS'99. IEEE conference and workshop on engineering of computer-based systems. IEEE, pp. 99–105

Guo J. et al. (2000) A survey of software reuse repositories. In: Proceedings seventh IEEE international conference and workshop on the engineering of computer-based systems (ECBS 2000). IEEE, pp 92–100

Gupta S, Kumar A (2013) Reusable software component retrieval system. Int J Appl Innov Eng Manag 2(1):187–94

Hojjat H, Mousavi MR, Sirjani M (2011) Formal analysis of SystemC designs in process algebra. Fund Inform 107(1):19–42

Kavitha P, Vidhya Saraswathi P (2021) Content based satellite image retrieval system using fuzzy clustering. J Ambient Intell Humaniz Comput 12:5541–5552

Lewczuk K (2021) The study on the automated storage and retrieval system dependability. Eksploatacja i Niezawodność 23(4):709–718

Lucredio D, Gavioli A, do Prado AF, Biajiz M (2004) Component retrieval using metric indexing. In: Proceedings of the 2004 IEEE international conference on information reuse and integration, 2004. IRI 2004. IEEE, pp 79–84

Lucredio D, Prado AFd, de Almeida ES (2004) A survey on software components search and retrieval. In: Proceedings 30th Euromicro Conference, 2004. IEEE, pp 152–159

Man K, van der Wulp J (2008) Specification and analysis of hardware designs using mcrl2. In: 2008 Canadian conference on electrical and computer engineering. IEEE, pp 000211–000214

Nie L, Zhong L (2009) Component retrieval based on domain ontology and user interest. In: 2009 International conference on e-business and information system security. IEEE, pp 1–4

Shi H, Chen Y, Hu J-Y (2021) Deep learning on information retrieval using agent flow e-mail reply system for IoT enterprise customer service. J Ambient Intell Human Comput. https://doi.org/10.1007/s12652-021-02991-7

Singh S (2013) An experiment in software component retrieval based on metadata and ontology repository. Int J Comput Appl 61(14):1–8

Vasanthi R, Jayavadivel R, Prasadh K, Vellingiri J, Akilarasu G, Sudhakar S, Balasubramaniam P (2021) A novel user interaction middleware component system for ubiquitous soft computing environment by using fuzzy agent computing system. J Ambient Intell Humaniz Comput 12:4827–4840

Xu B, An L, Thung F, Khomh F, Lo D (2020) Why reinventing the wheels? an empirical study on library reuse and re-implementation. Empir Softw Eng 25:755–789

Yahlali M (2022) Software components selection process: comparative study

Zhang L (2007) Software component repositories. In: Wiley encyclopedia of computer science and engineering