Construction and application of knowledge graph of domestic operating system testing

Dongsheng Jin
No.15 Research Institute of China Electronics Technology Group Corporation
jinds1225@foxmail.com

Zhi Wang
No.15 Research Institute of China Electronics Technology Group Corporation
wangzhi_nci@163.com

Mingyang Li
Institute of Software, Chinese Academy of Sciences
mingyang2017@iscas.ac.cn

Xinjie Zhu
No.15 Research Institute of China Electronics Technology Group Corporation
zhuxinjie1234@126.com

ABSTRACT

Aiming at the problems of poor reusability of domestic operating system test cases and insufficient sharing of test case design experience at this stage, a method for constructing knowledge graphs in the field of domestic operating system testing is proposed, and ontology construction and natural language processing technologies are applied to the field of software testing. Use the strong correlation of the knowledge graph to mine the experience knowledge in the design of historical test cases, select and reuse test cases that meet the test requirements for testers, and help them design test cases more efficiently. Through empirical research, this method gives full play to the advantages of knowledge graphs in relational network analysis and retrieval, and the coverage rate of reused test cases reaches 71%, which can greatly save test costs and improve test efficiency, and has strong engineering application value.

CCS CONCEPTS

• **Information systems**; • **Information systems applications**;

KEYWORDS

Domestic operating system, Software testing, Knowledge graph, Ontology construction, Reuse of test cases

ACM Reference Format:
Dongsheng Jin, Zhi Wang, Mingyang Li, and Xinjie Zhu. 2021. Construction and application of knowledge graph of domestic operating system testing. In *2021 4th International Conference on Computer Science and Software Engineering (CSSE 2021) (CSSE 2021), October 22–24, 2021, Singapore, Singapore.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3494885.3494933

1 INTRODUCTION

In recent years, the localization of the party, government and military has been gradually carried out, and the ability of independent control has been gradually strengthened. The domestic operating systems represented by the winning Kirin, Galaxy Kirin, Puhua, Deepin, etc. have initially possessed the ability to compete with similar foreign products. In order to ensure the quality of the domestic operating system, a large number of tests have been carried out in the military and civilian fields, and test cases related to the domestic operating system have been accumulated. However, most of the domestic operating system test schemes are customized around each project and are written based on experience and requirements. A considerable part of the use cases can only be used in specific projects, and the reusability is poor; moreover, the method of manually designing test cases is not only efficient. Low, resulting in a waste of historical test experience, and other people cannot learn relevant experience and knowledge from the designed test plan.

This paper constructs the knowledge graph ontology in the field of domestic operating system testing, and uses knowledge graph technology to model and represent the knowledge in the test data to help testers analyze historical data more efficiently, and can reuse from massive test cases to meet test requirements Test cases. Finally, the verification of examples shows that the knowledge graph technology can effectively improve the reuse quality of test cases in the field of domestic operating systems, thereby providing a new solution for the further research of knowledge graph construction in the field of software testing.

2 RELATED RESEARCH STATUS

With the improvement of domestic software product quality awareness, the proportion of software testing costs is getting higher and higher. In order to save test costs and improve test efficiency, the research of test case reuse has attracted the attention of many scholars at home and abroad in recent years. Literature [1] designs and optimizes the test case database, uses the database query function to query the current use case reuse library, and directly reuses the returned test cases. Literature [2] proposes to classify test cases, using the idea of test case index tree and tree matching model to propose a test case retrieval method for different data patterns. Literature [3] combined the collaborative filtering recommendation method and proposed a test case recommendation reuse technology based on radar software defect library.

These studies have improved the test case retrieval algorithms by building test case databases, index trees, and collaborative filtering, but most of them start from the test case itself for reuse and retrieval, and do not fully consider the test case as part of the

CSSE 2021, October 22–24, 2021, Singapore, Singapore

ACM ISBN 978-1-4503-9067-5/21/10…$15.00
https://doi.org/10.1145/3494885.3494933

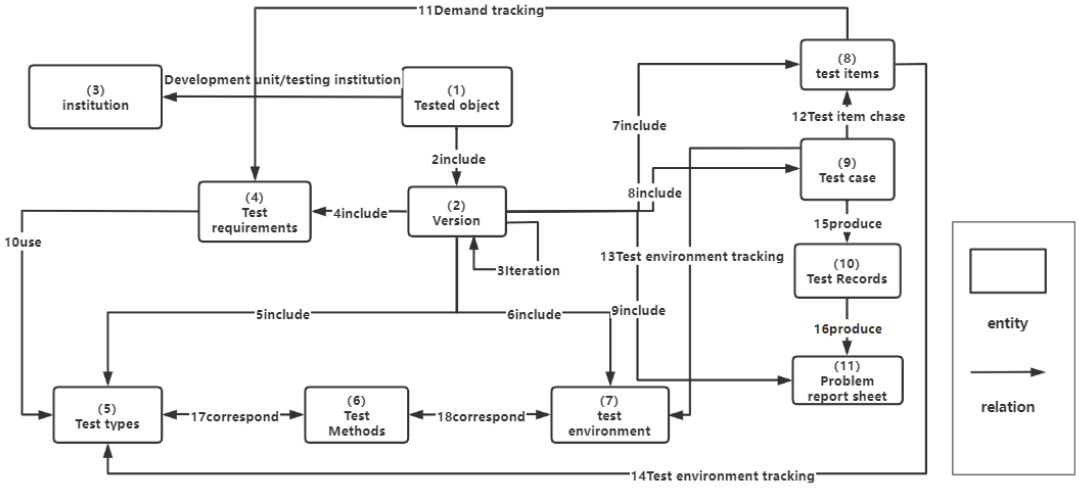

Figure 1: Ontology Model of Domestic Operating System Testing Domain.

test plan and other tests. The attribute of the relationship between knowledge. In the face of specific projects with changes in test knowledge such as test objects and test requirements, the actual application effect often fails to meet expectations. At the same time, since most software test case libraries at this stage use relational databases to store the underlying data, with the rapid growth of the number of test items, the limitations of the traditional test case reuse library design architecture have become more obvious, and it has gradually become unclear. Express the test knowledge relationship between different items. Therefore, in the face of more complex test knowledge relationship query requirements, higher requirements are put forward for the data retrieval efficiency of the entire test history data. In order to make up for the shortcomings of traditional software testing knowledge management tools and further optimize the ability to reuse the test cases of domestic operating systems, this paper uses the ontology construction and knowledge extraction technology of the domain knowledge graph to store the historical test data of the domestic operating system in the knowledge graph. Atlas realizes the reuse of test cases.

3 TEST CASE REUSE BASED ON KNOWLEDGE GRAPH

3.1 Domain ontology construction

The knowledge content of the domestic operating system testing field is relatively clear, and the relationship is relatively clear. Under the current immature Chinese ontology automatic construction technology, combined with the characteristics of the software testing field, the historical test outline and test plan are obtained through methods such as ontology learning and statistical learning. Based on the ontology knowledge, summarize expert opinions, and use the top-down model to develop the ontology design of the map. Fully consider the system and software quality requirements and the actual application requirements of test case reuse design. According to the characteristics of the domestic operating system test outline and test plan, the entity, relationship, attribute selection and structure design are carried out. The entity-relationship model of its ontology is shown in the Figure 1 shown.

Ontology construction first needs to summarize the core concepts of the test domain. The core concepts of the domain correspond to the entity(label) and their attributes in the ontology. Each core concept corresponds to many instances, for example: "By As an entity, the “test object” corresponds to instances such as “Galaxy Kylin Server Operating System”, “Galaxy Kylin Desktop Operating System”, and “Puhua Server Operating System”. The design of the entity should adhere to the principles of independence and sharing. The former refers to the entity that can exist independently and does not depend on a specific field; the latter refers to the entity can be shared, that is, there is the possibility and necessity of reuse. Moreover, the number of entities contained in the ontology should follow the principle of minimization, and eliminate redundancy as much as possible, and finally determine the core concept of the domestic operating system testing field, that is, the 11 entities and their attributes in Figure 1

(1) Tested object: refers to the domestic operating system tested in the test task, including domestic operating systems such as the winning Kylin, Galaxy Kylin, Puhua, Deepin and so on.

(2) Version: Each test task has a unique version number for this test object. When the regression test is performed, the tested object remains unchanged, and only the version number is changed.

(3) Institutions: Institutions are divided into development units and testing institutions. The development unit is generally the sender and the testing institution is the tester, including attributes such as address, contact person, and telephone number.

(4) Test requirements: In the actual test process, the test outline requires the test items to strictly cover the test requirements, and the test items are decomposed into multiple test cases. Testers need to write test cases according to the requirements specification. According to the "Chinese People The National Standard of the Republic GB/T 25000.51-2016 and related standards, for the quality

requirements of the system and software, test requirements can be divided into document requirements, functional requirements, interface requirements, performance and efficiency requirements, compatibility requirements, ease-of-use requirements, Reliability requirements, information security requirements, maintainability requirements, portability requirements, effectiveness requirements, efficiency requirements, satisfaction requirements, risk resistance requirements, peripheral coverage requirements, and other requirements.

(5) Test types: According to different test requirements, use corresponding test types.

(6) Test methods: There are three main test methods, including manual testing, automated testing and other tests.

(7) Test environment: For each test task, testers will specially deploy the test environment, which mainly includes hardware items, software items, software test tools, data sources, etc.

(8) Test items: Test items should cover all requirement categories, and test items should not contain specific test data. The data is reflected in use cases, including test item identification, test item description, test process description, sufficiency requirements, and constraints Conditions, termination requirements, passing criteria, priority, etc.

(9) Test case: The test item is decomposed into multiple test cases. The attributes of the test case include use case identification, test item identification, use case initialization, premises and constraints, termination conditions, passing criteria, designers, test steps, expected results and evaluation Standards, etc.

(10) Test records: Test records are generated by test cases. The attributes of test records include testers, test time, execution results, passing status, and problem identification.

(11) Problem report sheet: The test record will correspond to the problems that occurred in the problem report sheet. The attributes of the problem report sheet include reporter, report date, problem type, problem level/severe problem registration, problem description, regression test result, Repair man, repair plan.

The second step needs to define the domain relationship (relation) and its constraints. The relationship is the core basic element of the ontology. It is the description of the interaction between the concepts and examples in the test domain. The relationship directly determines the knowledge richness of the ontology knowledge graph. And the functional scope of other application systems based on the knowledge graph. Through the study of historical data and taking into account specific application scenarios, 18 relationships between entities are determined. The specific connection form is shown in Figure 1. The "iterative" relationship between versions means that the test field generally needs to consider regression testing. When the developer modifies the old code and re-tests, it only needs to iterate the version number, and does not need to change the object under test. Setting up the four relations of "Requirement Tracking", "Test Item Tracking", "Testing Environment Tracking" and "Testing Type Tracking" mainly considers the reuse of test cases based on the knowledge graph.

At present, scholars in the field of knowledge graphs recognize that domain experts are required to participate and collaborate in the process of constructing domain knowledge graph ontology. Therefore, after completing the two steps of building the ontology, experts in the field of testing and first-line testers with rich test experience and test plan design capabilities are invited to guide the inspection and evaluation of the ontology. According to the guidance of experts, through multiple rounds of iterations , After modification and improvement, the final ontology of the domestic operating system test domain is obtained.

3.2 Domain knowledge extraction

The purpose of knowledge extraction is to complete the extraction of entities, attributes, and relationships according to different data sources and different data formats [8]. This is a very critical part of the knowledge map construction process, and the quality of knowledge extraction determines the quality of the knowledge map. The relationship between entities and the attribute values of entities can be represented by triples (subject, predicate, object), for example (test case, generation, test record) is a triple, so knowledge extraction can also be called three Tuple extraction. The test data of the domestic operating system is divided into structured and semi-structured data. Structured data generally refers to the standardized data existing in the database. Test cases are generally stored in the excel table of the test management database, which can be regarded as structured data. The method of semantic mapping can be transformed into the form of knowledge triples. Correspondingly, for semi-structured text, the method of formulating templates for rule matching is adopted to obtain knowledge triples from historical test files.

The historical test data of domestic operating systems are mostly semi-structured texts, mainly from .doc/.docx test files such as test outlines, test requirements, and test plans. The manual extraction method for these large numbers of documents provided by multiple organizations is extremely inefficient. In order to improve the efficiency of constructing the test knowledge graph of the domestic operating system, the automatic extraction of test knowledge was realized with the help of the python-docx tool of the python library according to the characteristics of the test data. python-docx is a Python library that is used to create and modify Microsoft Word files, and use regular expressions to match tables and paragraphs of historical test data. Paragraph matching generally refers to paragraphs such as the operating system overview. Table matching includes Related organizations, software environment, test cases, test records, problem reports and other forms of data.

3.3 Knowledge Graph Application

The knowledge graph in the field of domestic operating system testing is mainly used in test case reuse. In order to make full use of the relationship between domestic operating system test cases and other test knowledge, and to further optimize the effect of test case reuse, a domestic operation based on knowledge graph is proposed. System test case reuse method.

Taking into account the scale and characteristics of the existing domestic operating system test knowledge graph, a CBOW (Continuous Bag-of-Words) continuous bag-of-words model based on the feature attribute text of the test requirement entity was selected. Natural Language Processing (NLP) Algorithm to quickly carry out text similarity comparison of test requirements entity attributes. After the CBOW model is fully trained, it can use the specified current word and its context as input to predict the probability of

the word's appearance. When using Python3.7 for algorithm programming, the Word2Vec toolkit provided by the Gensim library is selected.

For the input semantic text, after semantic keyword extraction and English abbreviation translation, the Chinese keyword sequence of the sentence is obtained. The training output file of the CBOW model can be used to obtain the word vector value of each keyword, and then all the sentences in the sentence After the word vector is averaged, the sentence vector vec(K) of the attribute text is obtained. Given two test requirement entities x and y to be matched, both have the same set of text attributes, by calculating x, y the cosine value between the sentence vectors K_x and K_y of the test requirement attribute description text, Get the text similarity sim (x, y) of entities x and y, As shown in equation 1):

$$sim_{attr}(x,y) = \frac{vec(K_x)\, vec(K_y)}{|vec(K_x)|\, |vec(K_y)} \quad (1)$$

In software testing, the test requirements are decomposed into multiple test items, and these test items are decomposed into multiple test cases. Therefore, the test cases corresponding to similar test requirements have a certain correlation, and the current software test can be calculated For historical software requirements with similar requirements, the corresponding test cases in the knowledge graph are used as reuse cases. With the continuous accumulation of historical data, there are also a large number of similar domestic operating system testing requirements in the knowledge base, and more optimal processing is required. Take the knowledge of test requirements as the core, construct a knowledge sub-graph including the test requirements of the project to be tested and its related entities, and collectively match the similar sub-graphs in the domestic operating system test knowledge graph network, so as to obtain a better retrieval output result. The specific algorithm is as follows.

Input: The test requirement entity Er of the project to be tested and the knowledge entities related to it, such as the tested object, test method, test type and test environment, form a software requirement knowledge sub-graph Gr.

Output: Reusable test case set index list.

(1) According to the attribute key values of E_r 's "object under test" and "test environment", quickly partition the test knowledge graph of the domestic operating system, and extract the subgraph G_r'of the calculated knowledge to be matched.

(2) Set the judgment threshold TH, calculate the attribute similarity sim_{attr} (E_r, E_r') of each software requirement E_r' in Er and Gr' according to equation 1), and retrieve *Top-n* and The list of test requirement entities larger than TH is L_r={$E_r'_1$, $E_r'_2$, . . ., $E_r'_n$}, Er'_k $(1 \le k \le n) \in L_r$.

(3) If L_r is not empty, select $E_r'_k$ with the largest similarity value as the software requirement matching result, and filter its associated test cases as the output based on the test type and test method.

(4) If L_r is empty, it is judged that the demand E_r has no matching demand in the knowledge graph, and the system has no corresponding reuse case output.

(5) After instantiating and modifying the output test case, the reused test case of the project is obtained.

4 APPLICATION RESULTS AND ANALYSIS

4.1 The results of domain knowledge map construction

This article chooses Neo4j graph database to store the knowledge map of the domestic operating system testing field, and uses Cypher data query language for data access. The application system of the knowledge graph is designed as a B/S architecture and implemented using Python programming. The back-end Web service is built using the Flask framework, and the front-end uses ECharts and HTML5 related technologies for visual display. The display effect is shown in Figure 2

4.2 Test case reuse results based on knowledge graph

In order to prove the effectiveness of the method proposed in this paper, the test data of 10 typical domestic operating systems are used as the experimental input, based on the Windows 10 operating system, the hardware platform is i5 CPU+8 GB memory, and the knowledge graph construction and application development platform is Python3.7+ py2neo4 .3.0, to compare the coverage rate with the current laboratory software test management platform.

The software test management platform currently used in the laboratory uses a relational database as the underlying data storage, enters the same test requirements, and compares the method in this paper with the coverage of test cases reused by the current test management platform. The calculation formula for coverage rate is coverage rate = number of eligible sampling points/total number of sampling points*100%, where the total number of sampling points is the number of test cases in the historical test case set prepared by test experts. The number of eligible sampling points refers to this article Method reused test cases or the current test management platform retrieves the number of test cases that are basically the same as the test cases in the historical test case set.

The experimental results show that the current test management platform uses a relational database to store the underlying data, and retrieves related test cases through keyword retrieval. There are big defects in performance, and the test cases cannot be well combined with other test knowledge. Once the keywords fail Accurate matching cannot achieve a good reuse effect, and the search matching results are not stable enough. Reuse test cases based on the knowledge map of the domestic operating system testing domain. By comparing the attribute text similarity of the test requirements, the advantages of the knowledge map relationship network analysis are fully utilized, and the reuse performance and stability have been significantly improved. The average coverage of test cases reached 71%.

5 SUMMARIZE

In order to solve the problem that the historical test knowledge of domestic operating systems is not well used, a knowledge map of the domestic operating system testing domain is constructed through domain ontology construction and knowledge extraction technology, and the knowledge map is strengthened for the reuse of test cases, and example applications are carried out. The results of this research show that by constructing a knowledge graph in the

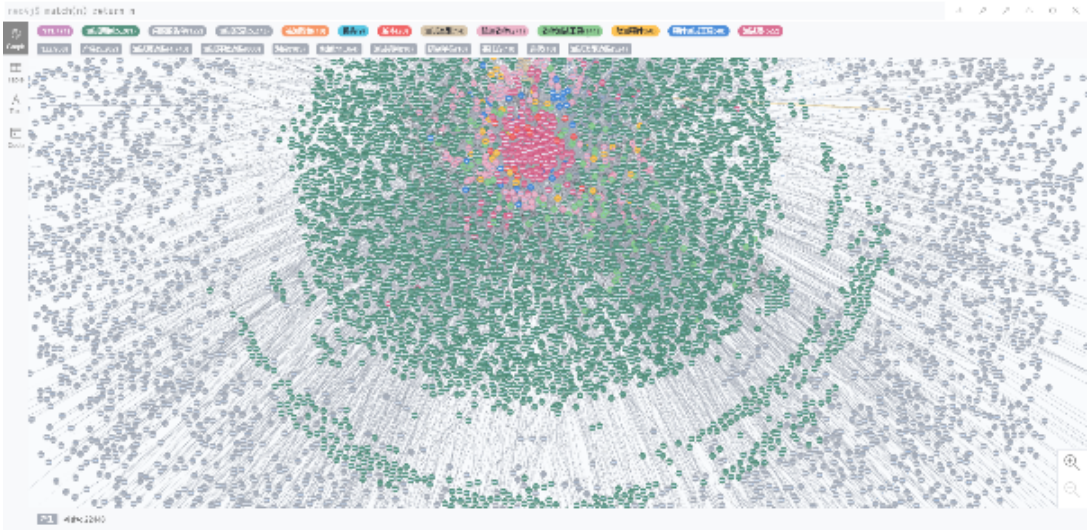

Figure 2: Knowledge graph display in the field of domestic operating system testing.

Table 1: Project verification data

Project Number	Test requirements (number)	Test cases (number)
P1	189	1692
P2	180	1256
P3	94	986
P4	169	1668
P5	171	1997
P6	174	1894
P7	386	2097
P8	491	2489
P9	272	1964
P10	201	1773

Table 2: Experimental data results

Project Number	Method of this article		Software test management platform	
	Recommended test cases(Number)	Coverage	Recommended test cases(Number)	Coverage
P1	1143	67.58%	507	29.97%
P2	865	68.85%	361	28.76%
P3	744	75.43%	353	35.79%
P4	1165	69.87%	454	27.24%
P5	1323	66.24%	539	26.98%
P6	1380	72.88%	311	16.42%
P7	1503	71.68%	12	0.56%
P8	1823	73.24%	119	4.78%
P9	1425	72.57%	566	28.84%
P10	1271	71.69%	493	27.78%

field of domestic operating system testing, it can effectively improve the quality of test case reuse, improve test efficiency, and help testers learn and design test cases. Compared with the traditional test management platform that uses relational databases as the underlying storage, the test case coverage rate reaches 71%. The reusability statistics of 34 test items based on the domestic operating system test knowledge graph to carry out the reuse design of use

cases showed that the average reuse rate reached 76%, which saved about 69% of the time spent on test case design.

Based on the knowledge map of the domestic operating system testing domain, more extended applications can be carried out, such as: rapid deployment of the test environment based on the knowledge map, recommendation of the test plan based on the knowledge map, etc. There is still a lot of work to continue research.

REFERENCES

[1] W. Li and M. Duan. A study on the software test case reuse model of feature oriented.2014 IEEE 3rd International Conference on Cloud Computing and Intelligence Systems. IEEE, 2014, pp. 241–246.

[2] Z. Nan, C. Haiyan, and H. Xinyu. Research on the reuse of test case for warship equipment software.2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C). IEEE,2018, pp. 64–69..

[3] S. Guo, J. Zhang, W. Tong, and Z. Liu. An application of ontology to test case reuse.2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC). IEEE, 2011, pp.775–778..

[4] Liu X ,Yang G, Cai L , *et al.* Ontology description and retrieval of test case based on reuse behavior [J]. Computer Applications and Software, 2011.

[5] H.-H. Song and Z.-X. Zhang. Study on approach of software mainte-nance based on ontology evolution.2008 International Conference on Computer Science and Software Engineering, vol. 5. IEEE, 2008,pp. 585–588..

[6] A. J. Wiebe and C. W. Chan. Ontology driven software engineering. 2012 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE). IEEE, 2012, pp. 1–4..

[7] S. Popereshnyak and A. Vecherkovskaya. Modeling ontologies in soft-ware testing.2019 IEEE 14th International Conference on Computer Sciences and Information Technologies (CSIT), vol. 3. IEEE, 2019, pp.236–239..

[8] Gupta A, Tyagi S, Panwar N, *et al.* NoSQL databases: Critical analysis and comparison[C]//2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN). IEEE, 2017: 293-299.