Research on User Knowledge Acquisition and Application in Software Ecology

Ding Qiuyua, Shang Fuhua

College of Computer and Information Technology, Northeast Petroleum University

a1246035670@qq.com, 0086-457-19845924336
bShangfh@163.com, 0086-457-13904597865

ABSTRACT. In the process of software development, users play a crucial role. User knowledge plays a role of energy driver in the software ecology, and user knowledge plays a guiding and reference role in software development. From the perspective of knowledge chain, improving the acquisition rate and utilization rate of user knowledge can better complete the upgrading work of large software platform. Based on this starting point, this paper puts forward the idea of actively acquiring user behavior information, mining association rules of software platform components, and finally applying knowledge mapping technology to the knowledge representation of user knowledge and inherent knowledge of software platform, so as to strengthen the extensibility and continuity of knowledge. Experiments show that above methods can improve the acquisition rate and utilization rate of user knowledge, quickly locate relevant entities in the software to improve the speed and quality of large software platforms.

1. INTRODUCTION

As an independent discipline, software has experienced challenges such as software crisis in its development process, and has derived related researches such as software engineering and software ecosystem. One of the problems that is aimed at solving is the growing user base of software systems and the personalized development of user needs [1-3]. In the software ecosystem, energy is the driving force for its normal functioning, and this energy comes from the use of knowledge [4, 5]. Since 2011, the project team has participated in the national key science and technology project “Integrated Logging Platform CIFLog” [6] and has been committed to the research of industry application software ecosystem, and has actively responded to the national call and is committed to the in-depth exploration of smart energy. This paper focuses on improving the ecological environment of large-scale software platforms in the logging field and improving the utilization of knowledge within the ecosystem.

The software ecosystem in the context of this paper refers to the ecological environment of the domain software system. The existing research finds that there is a relationship between developers, projects and users. This relationship is called the knowledge chain [5, 8], the relationship between the developer and the user in the knowledge chain flows as follows: the user uses the software platform and feeds back to the developer the corresponding experience information, and the experience information may be related to any aspect of the software, such as the adaptability and growth of the software. And bugs and suggestions for improvement. Software development is a phased process [9]. In the early days of software products, especially large software platforms, it was carefully designed and has a good architecture. However, with the development of time, the needs of users have changed, and maybe even the audience has changed. At this time, the software platform needs to make corresponding changes.
Therefore, software reconstruction is often used to modify and maintain the code in the software platform development process [10-12], and the user feedback information is developed. User feedback is an important reference for developers to improve and upgrade the software platform. In the process of users using the software platform, new knowledge is continuously generated, which plays an important role in guiding and learning from the flow of users to developers [13]. In this article, all knowledge generated by a user using a published software platform is referred to as user knowledge, such as the experience of use, expectations of the software platform, the process of using the software, and the like.

In summary, improving the accessibility and utilization of user knowledge is one of the means of improving software quality and development efficiency in current software engineering. Although there are researches on software development and knowledge organization, how to develop software development knowledge of Keqing et al. Describe the framework SOKM [14], Lu Yi and Jin Zhi's "Knowledge Project" [15], Li Weipeng and other people's project knowledge graph construction [16], etc., but many studies have largely ignored users generated during the user's use of software Formal representation of knowledge., while the research on user behavior information acquisition ignores the important role of knowledge extraction and knowledge representation in the software development process [17-20]. Knowledge acquisition in the user use phase is mainly faced with the following challenges:

1. User may not provide feedback information for software developer software use;
2. The entire process of bugs in the use of the software platform cannot be accurately obtained;
3. Even if the process of using the software platform by the user and related information (except for the user's own data) can be obtained, knowledge with greater value density cannot be extracted from it.

In order to effectively organize the knowledge generated by the user in the process of using the software platform, and better enable the software developer to reconstruct the software, this paper aims to optimize the knowledge chain between users and developers by using artificial intelligence technology and knowledge organization technology. Landing experiments were carried out in the “Integrated Logging Platform CIFLog”, a large software platform for logging. Firstly, the user collects the behavior information of the software platform, such as behavior data, bugs, user feedback, etc., and then refines the knowledge, including entity extraction and association rule mining, and finally, the knowledge gained from the mining and the knowledge inherent in the software platform are formally represented.

2. User knowledge acquisition

2.1 User knowledge flow

As mentioned above, the focus of this article is to enable software developers to maximize user knowledge. Traditionally, the transfer of information between users and developers is one-way, that is, if the user does not actively send feedback to the developer, the developer cannot obtain any information. This article advocates that developers actively acquire user knowledge, which requires certain permissions to be retained during the software development process to obtain user behavior information. When the user installs the software, the user is prompted to participate in the user experience plan. What the developer needs to obtain is the function of the user to use the software and the specific operation steps, so the user participating in the user experience plan locally generates relevant documents for recording the user usage information. At the same time, the document will be sent to the server of the software development office at regular intervals.

In the record document, each button will have its own id, which has the same name as the code in the software development, which saves a lot of work for subsequent document preparation and code reconstruction. The feedback information flow process at this time is as follows:
2.2 Component Association Rules

After obtaining the user's behavior information, knowledge extraction is required. The goal of knowledge extraction in this paper is to dig out the relationship between components in the software, including the relationship between modules, the relationship between buttons and the relationship between modules.

First, the behavior information of different users obtained is processed. Each user has a table in the database, the table id is the time when the user uses the software, the table content is the behavior information of the user using the software, and the use ends to exit the software as a mark.

After pre-processing, the association rules are mined based on the obtained information. Let $P = \{p_1, p_2, ..., p_n\}$ be a collection of all component items in the software platform, data set $D$ is a collection of database transactions storing user behavior information, and any transaction $T$ is a collection of some component items in the software platform, $T \subseteq P$. Let $A$ be a component item set that makes up the software platform, then transaction $T$ contains $A$ as $A \subseteq T$. The association rules are expressed in the form of implications. For example, in the software platform, button $A$ and button $B$ are related to the implied expression of $A \Rightarrow B$, where $A \subseteq P$, $B \subseteq P$, and $A \cap B = \phi$.

Two important concepts in relation mining are support and confidence. The support degree $s$ is the ratio of the union of the software platform component item set $A$ and the other software platform component item set $B$ in the database transaction set $D$ storing the user behavior information:

$$s = \frac{\text{count}(A \cup B)}{|D|} \quad (1)$$

Confidence $c$ is the ratio of transactions that also contain $B$ in transactions that contain both $A$ in $A$ and transactions that contain only $A$:

$$c = \frac{\text{count}(A \cup B)}{|D|} \quad (2)$$

In relation mining, a set of items whose support degree is greater than or equal to the minimum support level threshold is called a frequent item set or a large item set. Association rule mining is to find the top-K term in the frequent itemsets with confidence greater than the minimum confidence threshold. This process is also the mining process of strong association rules. The mining association rules will be stored and represented in the form of knowledge for use by developers.

2.3 Component Association Rule Category

At present, the association rules are mainly divided into two categories in different application environments: Boolean and numerical. The association rules mined in this paper are Boolean. Boolean association rules are used to indicate whether there is an association between associated objects, and the associated objects are usually discrete objects. For example, in the "Integrated Logging Platform CIFLog", the user first imports data before using the horizontal well interpretation module to construct...
the geological body. The import data function belongs to the data management module, which is
Horizontal well interpretation⇒ Data management module, what is embodied here is the existence
relationship.

2.4 Association rule mining algorithm selection
The most important step in the mining of association rules in this paper is to find frequent itemsets with
confidence greater than the minimum confidence threshold. First, all the frequent items that satisfy the
minimum support threshold from the transaction database, and then find the frequent items found. A set
of items with a confidence greater than the minimum confidence threshold. Complete the above two
steps to find a strong association between components.

Among the many association rule mining algorithms, Apriori algorithm is considered to be the most
classic rule mining algorithm [21], and it is also a relatively mature association rule mining algorithm.
And because of its good performance in mining the association rules between sparse data, and with good
scalability, this paper uses this algorithm to mine the association rules between software platform
components. The Apriori algorithm uses a breadth-first search strategy to iteratively mine Boolean
association rules from the database layer by layer.

In this paper, the Aoriori algorithm is used to scan the user behavior information in the form of arrays
stored in the database, and then the support degree of each item set is calculated to obtain the frequent
itemsets \( \{L_1, L_2, L_3, \ldots, L_k\} \), overlay from 1 until you can't find a new frequent item set. When
calculating frequent k item sets \( \mathcal{L}_k(k = 1, 2, \ldots) \), firstly, the candidate set \( \mathcal{C}_k \) is generated by \( \mathcal{L}_{k-1} \) self-
joining, and then \( \mathcal{C}_k \) is reduced by using a certain trimming frequent item set strategy. Finally, the
transaction database is scanned to obtain the frequency of the candidate set, and the infrequent item set
is cut off, thereby obtaining the frequent item set \( \mathcal{L}_k \).

2.5 Component Association Rule Mining Based on User Behavior Information
First, the obtained user behavior information is preprocessed, and the meaningless and non-resulting
operations are deleted: for example, repeatedly clicking a button or clicking on a blank space without
caus ing any feedback information of the software.

Secondly, the information is classified from the user behavior information.
1) The functional module that the user clicks for the first time is classified according to the
classification, and the next functional module is finished, and the operation process of the period is
extracted and defined as User_Module_time; the process ends and the search is continued from the last
time. Extract
2) Extract the function module button clicked by the user each time the software is used from the
information document, and name it User_Module;
3) The function module that the user clicks for the first time is classified according to the
classification, and the next function module is finished, and the usage information of all the buttons in
the toolbar is extracted from the information document, and is defined as
User_Module_ToolbarButton_Time;
4) (4) Extract all the abnormal exit software and the first ten buttons of the previous steps from the
information document, and all the insufficient ones are extracted, defined as User_Bug_Time;

In this paper, each serial of the user's operating software is treated as a separate individual, and then
these independent individuals are clustered according to the module of the software function in which
the first click is. The information obtained after clustering is stored in the database and marked.

Finally, the Apriori algorithm is used to mine the association rules in the above four types of user
behavior information, and then the associated association rules are applied to construct the user behavior
information knowledge graph, so as to more clearly analyze and apply the knowledge obtained for
software analysis and upgrade.
3. User Behavior Information Knowledge Graph

This section combines the experimental process and results in the "Integrated Logging Platform CIFLog" to describe the construction and application process of user behavior information knowledge graph.

Before the user behavior information graph is constructed, the user information behavior information needs to be preprocessed, deduplicated and invalidated, and then the obtained user behavior information is clustered according to the above classification manner, and then the Apriori algorithm is used to perform association rule mining. The knowledge obtained from the final application is used to construct the user behavior information knowledge graph.

![Diagram of user knowledge processing]

The construction method of user behavior information knowledge graph mainly draws on the semi-automatic construction method of pet knowledge graph of Yuan Qi et al. [22]. The construction process of user behavior information knowledge graph is as follows:
3.1 Construction Schema Layer

As can be seen from the above figure, the primary task of constructing the user behavior information graph is the construction of the Schema layer. In this paper, the concept layer of the user behavior information knowledge graph is constructed in a top-down manner. Since building a Schema layer is equivalent to modeling the entire software platform knowledge framework, the Schema layer needs to accurately define and constrain the entity classes and relationships, which is equivalent to mapping the definitions of relationships between classes and classes to the concepts in the graph. The definition and constraints of the semantic relationship with the concept, so we extract the main classes and relationships from the software platform.

In this article, the basic classes defined are modules, functions, interfaces, and buttons.

Second is the definition of the attribute:

a) The properties of the module include: name, function, interface, button, bug, pre-module, post-module;

b) The attributes of the function include: name, module, interface, button, bug, pre-function, post-function;

c) The attributes of the interface include: name, function, module, button, bug, front interface, and rear interface;

d) The properties of the button include: name, module, interface, function, bug, front button, and rear button.

According to the four categories in software development defined above, we define three semantic relationships:

a) $\text{E\_HasFunction}$ (function): refers to the relationship between modules and functions;

b) $\text{E\_HasInterface}$ (with interface): refers to the relationship between the function and the interface;

c) $\text{E\_HasButton}$ (with button): refers to the relationship between the interface and the button.
The constructed Schema layer is as above. According to the programming specification, the function module can be obtained by identifying the name or the annotation, and the mapping of the computer interface to the underlying code can be realized through this feature.

3.2 Information Extraction and Entity Filling
After constructing the Schema layer of the user behavior information knowledge graph, entity extraction and entity filling are required, and the mined association rules are filled in. Taking the horizontal well interpretation module in the “Integrated Logging Platform CIFLog” as an example, some of the results obtained by mining the association rules based on user behavior information are as follows:

Table 1 A part of association rules table of horizontal well interpretation module

| Software knowledge component entity set = {module, function, interface, button} |
| Strong association = \{data management module, horizontal well interpretation module\}, \{multi-well evaluation module, horizontal well interpretation module\} |
| In-module functional relationship = \{horizontal well data import, three-dimensional visualization of work area\}, \{three-dimensional visualization of work area, three-dimensional geological model of horizontal well\}, \{three-dimensional geological model of horizontal well, optimization of stratigraphic model\}, \{three-dimensional geological model of horizontal well, stratum model optimization\} |
| Association between Button = \{[open attribute body model card], [select target well and front well], [introduction level model], [select formation to construct geological body]], ([select formation to construct geological body], [measurement Distance between two wells]], ([selecting strata to build geological bodies], [grid lines], [rulers]]\} |

This paper mainly extracts the entity from the user behavior information. Due to the limitation of the programming specification, the process of entity extraction is relatively simple. Only by iteratively reading the data stream and combining the regular expression can get a good effect. The extracted information is filled according to the top-level framework defined by the Schema layer, so that the attributes of the components in the software platform are represented, which are no longer only a named symbol, but are richly described when the user pairs. The entity can obtain a large amount of information of the entity while searching, or the system can quickly and accurately lock to the corresponding entity.
when inputting the description information, and formalize the related entities and their relationships according to the association rules.

3.3 Knowledge Representation

The knowledge graph can be regarded as a graph network structure. Therefore, the knowledge graph model can be represented by the RDF or attribute graph proposed by the W3C. In this paper, the OrientDB graph database is used to store the obtained user knowledge and software inherent knowledge, so the attribute is selected. A graph model to formally represent user knowledge. The nodes and edge elements in the property diagram are as follows:

a) A set of nodes. Each node has a unique identifier @rid, which corresponds to the defined entity type @class, which represents the concept class to which the entity belongs. Each vertex has a key-value pair to represent the attribute. Nodes are connected by edges with nodes, so vertices have edges and edges, but not all.

b) A set of edges. Each edge has a unique identifier @rid, which corresponds to the defined entity type @class, indicating which relationship type the edge belongs to. At the same time, each edge must have a head node and a tail node and each edge has a key-value pair to define the attribute combination.

The following figure depicts an attribute graph model of OrientDB. The relationship between the module class "Horizontal_well_interpretation_module" in the software platform and the module class "Data_management_module" is "E_Has_Strong_correlation". Where @rid is the unique identifier, @class is the entity type, which is the corresponding concept class, out corresponds to the head node, in corresponds to the tail node, and the key value corresponding to name is the description of the corresponding node attribute.

3.4 Knowledge Storage

The database used in this study is OrientDB. The reason for choosing it is that it supports multiple modes, which can store and form representations of graphics, documents, relationships, etc., and also provide a bridge for the management of graph databases. Support us to master more proficient SQL statements and SQL-like statements. When applying OrientDB to knowledge integration and storage of instance layer data obtained from software development code, it is necessary to create a pattern according to the definition of Schema layer, and then load node information and node relationship. In order to prevent duplicate node information and duplicate relationships when importing data information, use SQL-like query statements to determine whether or not to repeat.

3.5 Knowledge Update

User knowledge is constantly being generated with the user's use, and the user's demand for the software platform is constantly changing. Therefore, our user behavior information knowledge graph also needs to constantly update its knowledge nodes and rule policies. However, because the software framework
is relatively stable and less subject to change, this article only needs to consider the update of the data level.

The data update of the user behavior information knowledge graph is to update the knowledge nodes and edges according to the changes of the software platform and the association rules excavated according to the user behavior information. When the above data layer changes occur, the knowledge graph is updated based on the detected changes in the database.

4. Applications

4.1 Application Display

In the "Integrated Logging Platform CIFLog", taking the horizontal well interpretation module as an example, the statistics of the extracted software platform knowledge entities and associations are shown in the following table.

| Software resource type | Software knowledge entity | Number of entities |
|------------------------|---------------------------|-------------------|
| Features               |                           | 34                |
| Button                 |                           | 125               |
| Source code            | Interface                 | 39                |
|                        | Bug                       | 3                 |
|                        | Front module              | 8                 |
|                        | Rear module               | 9                 |

Table 3 Software platform entity association statistics table

| Relation type          | Number of relationships |
|------------------------|-------------------------|
| E_HasFunction          | 40                      |
| E_HasInterface         | 61                      |
| E_HasButton            | 156                     |

Table 4 Software platform entity association relationship mining results statistics table

| Relationship type          | Number of relationships |
|---------------------------|-------------------------|
| Intra-module functional relationship | 13                     |
| Inter-interface associationInter-module relationship | 9                     |
| Button relationship        | 42                      |

The graphical visualization search results are shown in the following figure. The following figure shows the modules with related relationships obtained by the horizontal well interpretation module through the steps of knowledge extraction and association rule mining and related functions.
Fig.6 Schematic diagram of horizontal well module association rules

It can be seen that the graph structure supported by OrientDB can clearly express the relationship between the mining and the extracted knowledge in the form of edges, and at the same time, the color of the different knowledge categories is displayed, and the relationship and nodes are support for the lookup of SQL statements.

5. Conclusion
Through the method of this paper, knowledge acquisition and association rule mining of user knowledge can achieve greater improvement of the utilization of user knowledge in software ecology, improve the scalability and continuity of knowledge, and realize the quality and efficiency of software platform development. Improve and ensure the sustainability of the software, so that the software can better meet the needs of users, enhance the user experience, and enable developers to follow the software platform upgrade and improvement process. From the perspective of software engineering, improving the utilization of user knowledge is also a process that must be experienced and overcome in the development of software platforms. For the extremely powerful software platform, such as the experimental platform “CIFLog” in this paper, this is also the intelligent research in the intelligent energy direction, especially the intelligent software platform in the logging field. A key step.

There are still some work in this paper that need to continue to advance, such as knowledge self-learning mechanism, knowledge mining algorithm, knowledge update method and intelligent analysis algorithm of knowledge software platform.

Acknowledgements
This effort is sponsored by the National Science and Technology Major Project.

REFERENCES
[1] Randell Brian. Software engineering: Concepts and techniques: proceedings of the NATO conferences [J]. 1976.
[2] Yang FQ. Thinking on the development of software engineering technology. Journal of Software, 2005,16(1): 1–7.
[3] Messerschmitt D, Szyperski C. Software Ecosystem: Understanding An Indispensable Technology and Industry[M]. Cambridge, USA: MIT Press, 2005.
[4] N. V. Bala Iyer, Chi-Hyon Lee. Managing in a "small world ecosystem": Lessons from the software sector. Harvard Business Review, (5), May 2006.
[5] Zhang Deguang, Li Bing, He Peng, et al. Characteristic Study of Open-source Community Based on Software Ecosystem[J]. Computer Engineering, 2015, 41(11): 106-113.
[6] Li Ning, Wang Caizhi, Liu Yingming, et al. CIFLog: the 3rd generation logging software based on Java-NetBeans[J]. Acta Petrolei Sinica, 2013, 34(01): 192-200.

[7] National Development and Reform Commission, National Energy Administration, Ministry of Industry and Information Technology. Guidance on promoting the development of “Internet +” smart energy [J]. City Gas, 2016(4):4-9.

[8] Boucharas V, Jansen S, Brinkkemper S. Formalizing software ecosystem modeling[C]// International Workshop on Open Component Ecosystems. ACM, 2009.

[9] Yang FQ. Software Reuse and Its Correlated Techniques [J]. Computer Science, 1999, 05): 1-4.

[10] Yang Fuqing, Mei Hong, Li Keqin. Software Reuse and Software Component Technology [J]. Acta Electronica Sinica, 1999, 27(2): 68-75.

[11] Zhan Haomin, Yin Guisheng, Wang Shuyin, et al. Requirements optimization based on combination of technologies [J]. Systems Engineering and Electronics, 2013,35(07):1467-1471.

[12] Xu Wenbo, Hua Qibing, Fei Na. Object-oriented Software Refactoring [J]. Computer Engineering, 2005(05):82-84.

[13] Chen G, Lu RQ, Jin Z. Constructing virtual domain ontologies based on domain knowledge reuse. Journal of Software, 2003,14(3):350–355.

[14] He Keqing. Representation and utilization model of software development knowledge SOKM [J]. Computer Science, 1989, 16(6):18-26.

[15] Lu Ruqian, Jin Zhi. From knowledge-based software engineering to knowledge-based software engineering [J]. Chinese Science (Series E: Information Science), 2008, (06): 843-63.

[16] Li Wenpeng, Wang Jianbin, Lin Zeqi, et al. Software Knowledge Graph Building Method for Open Source Project [J]. Journal of Frontiers of Computer Science and Technology, 2017, 11(06): 851-62.

[17] Farha R, Leongarcia A. Blueprint for an Autonomic Service Architecture[C]// International Conference on Autonomic & Autonomous Systems. 2006.

[18] Yang Liu. Research on Proactive Self-adaptive Software based on User Behavior [D]. Central South University,2013.

[19] Vavilov D, Melezhik A, Platonov I. Reference Model for Smart Home User Behavior Analysis Software Module[C]// IEEE Fourth International Conference on Consumer Electronics – Berlin. IEEE, 2015.

[20] Yi Ma, Crook P. A, Sarikaya R, et al. Knowledge Graph Inference for spoken dialog systems; proceedings of the IEEE International Conference on Acoustics, F, 2015 [C].

[21] Jiexun Li, Guoqing Chen. A SAR-based Interesting Rule Mining Algorithm[C]. NAFIPS, 2002, 178-183.

[22] Yuan Qi, Liu Yuan, Xie Zhenping, Lu Jing, et al. Semi-automated construction method of pet knowledge graph [J]. Application Research of Computers, 2018, 1-7.