TRUSTED REQUIREMENT MODEL AND REQUIREMENT ACQUISITION PROCESS BASED ON META-REQUIREMENT

Xiangjun KONG[1], Xuejun YU[2]*

With the development of computer technology, the social demand for software is increasing at a very high speed, especially for trusted software. In the development life cycle of trusted software, the acquisition and analysis of trusted requirements become the core. Trusted software requirements should have the characteristics of accuracy, reliability, security, correctness, timeliness, maintainability. Trusted software development environment has expanded from traditional closed static environment to open, dynamic and changeable Internet environment. Trusted requirements acquisition is an important part of trusted software development, which obtains clear and accurate trusted requirements through research and communication with users. The purpose of this paper is to improve a series of problems, such as software quality problems, budget overruns and delivery delays, which are caused by poor efficiency and difficulty in acquiring trusted requirements in the process of acquiring trusted requirements. This paper introduces the concept of meta-requirement into trusted requirements, puts forward the concept of trusted meta-requirement, studies its basic elements and characteristics, constructs a trusted meta-requirement model based on it, and puts forward a trusted requirement acquisition process based on this model, so as to improve the efficiency, accuracy and comprehensiveness of trusted requirement acquisition and analysis.

Keywords: Trustworthy requirement; Meta-requirement; Requirement model

1. Introduction

The word "trusted" comes from trusted computing, which is a trusted computing platform supported by hardware security modules widely used in computing and communication systems [1]. TCG defines software credibility as: If the execution process of an entity meets the specified conditions and can originate in the specified direction, subsequently, the entity is assumed to be trusted [2]. Currently, constructing trusted software has become an important trend and inevitable choice in the development and application of modern software technology, among which requirement engineering is the key. The statistical data presented in a study conducted in 2011 show that most defects

[1] Faculty of Information Technology, Beijing University of Technology, Beijing, China, e-mail: 827696951@qq.com
[2] Faculty of Information Technology, Beijing University of Technology, Beijing, China, *e-mail: yuxuejun@bjut.edu.cn

come from the requirements engineering stage, accounting for 56% in total, while the design stage, coding stage and other stages only account for 27%, 7% and 10% [3]. Requirements engineering is a human-centered software engineering activity, which tends to make mistakes [4].

This paper combines the concept of meta-requirement with the process of obtaining trusted requirements. Meta-requirement is the most elementary characteristic and the minimum component unit of requirements, and it is the main workpiece describing the most basic functions that the system must have [5].

In accordance with research, Gerwinn [6] put forward the main concepts of the meta-requirements model under the support of domain-specific scenarios. Saoud Z [7] and others built a software credibility evaluation model based on fuzzy evaluation and studied the credibility attributes of the software. Bawdry [8] and others proposed a meta-model of requirements and how to use it under the top-level natural linguistics constraints of requirements definition. Natalie, Thomas Lee and others [9] put forward a model of non-functional requirements of trusted software, and formally expressed it and analyzed its satisfiability, notwithstanding, they did not study the trusted requirements in detail. Sun [10] proposed an extended soft target interdependence graph model to describe and represent nonfunctional requirements for trusted software. Xiao [11] and others analyzed the quality index system of software requirements and trusted operation and proposed a formal modeling method of software requirements defects on the basis of the unified modeling language with the usability and maintainability of trusted attributes as indicators. Through this method, software requirements defects can be predicted, and the usability and maintainability of software can be enhanced through this method. Rouland [12] et. al. used modeling and formal methods to decrease security vulnerabilities in software design and improve system security and reliability and studied the usage of formal methods to accurately standardize and analyze security architecture threats. Ramos H [13] and others centered on user perception, through questionnaire data collection, empathic map modeling and map grouping to generate roles, then modeled a nonfunctional requirement in software - interpretable requirement, which improved the acquisition efficiency of interpretable requirement. Ong m [14] and others proposed a formal initial model of user requirements standard and meta-requirements and mapped the user requirements standard to the meta-requirements model. Additionally, Luo et al. [15] proposed meta-requirement model and suite of meta-requirement sets based on ontology. It can be seen that the researchers did not have a thorough discussion on the application of meta-equid in reliable requirements.

These research studies have made their own contributions to some aspects of trusted software or software requirements, but on the one hand, no one has applied meta requirements to trusted requirements; On the other hand, some

studies are related to a trusted attribute of trusted requirements, but there is a lack of overall research on trusted requirements in trusted software.

Therefore, this research analyzes some commonalities between meta-requirements and trusted requirements, and afterward combines with their basic elements. The definition and components of trusted meta-requirements are given. Meta-requirements cannot be split, can clearly express client requirements, and has the characteristics of accuracy, completeness, and measurability, so it can enhance the accuracy of credible requirement description. On this basis, a trusted meta-requirement model is proposed, and the visual representation diagram of the model is given. A trusted requirement acquisition process is proposed according to the model, this process obtains the final trusted requirements of the whole system through screening and merging. It can improve the efficiency of obtaining trusted requirements.an experiment demonstrates the effectiveness of this model in the analysis of trusted requirements acquisition.

2. Trusted meta-requirement model

Trusted meta-requirement combines the concepts of trusted requirement and meta-requirement and incorporates the atomicity of meta-requirement in trusted requirement. It embodies the most basic characteristics of trusted requirements and affects the whole software in various trusted attributes respectively. Meta-requirements solve a class of problems rather than a single problem [16], their collection is also the trusted requirement of the whole software. The analysis of all aspects of the trusted meta-requirements is the basis of constructing the trusted meta-requirements model.

2.1. Trusted meta-requirement semantic model

2.1.1. Trustworthy Indicators

In searching for software credibility, how to describe and measure software credibility is important content. The concept of trusted attribute came into being. The trustworthiness indicators in the trustworthiness meta-requirement involved in this paper are as follows:

1. Availability (AT): The degree to which the software is effective, effortless to learn, efficient, easy to remember, error-less and satisfactory to the user, in other words, whether the user can adapt the software to complete his tasks in time and effectively.

2. Security (ST): This attribute supplies a systematic method to identify, analyze and track software measures to mitigate and control system hazards and functions that may cause hazards, so as to ensure safer software operation in the system.

3. Reliability (RBT): the capacity of software to run effectively in a

specified environment and within a specified timeframe.

4. Instantaneity (IT): The ability of software to respond or deliver the output within a specified time.

5. Survivability (SVT): The ability of the software to continuously provide services and restore all services within a specified timeframe when it is attacked or defaulted.

6. Maintainability (MT): The difficulty of maintenance personnel to maintain the software, including the difficulty of understanding, correcting, changing and improving the software.

The above-mentioned trustworthiness indicators will be used as the standard of trustworthiness-related elements in the trustworthiness meta-requirement model and delimit the scope of trustworthiness tendency elements in the model and trustworthiness elements in the trusted meta-requirement.

2.1.2 Basic elements

Based on the literature and the analysis of some commonness between meta-requirements and trusted requirements, combined with some basic elements of meta-requirements and trusted requirements, the following elements are selected as fundamental elements of trusted meta-requirements in this study.

1. ID, the unique identification of the trusted meta-requirement.

2. Type (T), the category of trusted meta-requirements, functional requirements and non-functional requirements of software jointly affect the credibility of software [17], accordingly, the two largest categories are divided into functional requirements and non-functional requirements.

3. Description (D), the description information of the requirements, is in text format.

4. Priority (PR), the importance of the requirement, given by the requirement proposer or by the expert, is divided into five degrees, which are unimportant (0), less important (0.25), medium (0.5), and more important (0.75) and important (1).

5. Credibility (CD), the influence degree of requirements on each trustworthiness attribute of the whole software, is divided in accordance with the trustworthiness indicators, including availability (AD), safety (SD), reliability (RD), real-time (ITD), survivability (SVD) and maintainability (MD), which are positive or negative, expressed by positive numbers and negative numbers respectively, and the range is [-1, 1].

6. Status (S), the state of requirements in the whole software process, generally includes several states: proposal, analysis, reception, rejection and replacement.

7. Particle size (PS), which represents the level of trusted requirements for the whole software, is divided into 1-6 levels, among which 6 level and 5 level requirements that cannot be divided called trusted meta-requirements.

Level Six: Trusted function point elements differentiated by trusted attributes in a single function point, such as whether duplicate names are supported when usernames are modified or registered, which involves maintainability, usability, and other attributes. Each button or link should give the user meaningful feedback when touched or clicked, which is a simple need to improve availability.

Level Five: Trusted requirements in the form of a single function point, such as avatar modification, avatar viewing, username modification and other functions.

Level Four: Multiple function points composite trusted requirements, such as username-related functions.

Level Three: It is composed of four-level granularity trusted requirements with similar aspects, such as personal information-related requirements combined with username function and gender function.

Level Two: A collection of all trusted requirements on a page, and this page should have multiple trusted requirements with tertiary granularity, for instance, the personal center pages with personal information, member-related, order-related and other functions.

Level One: the collection of all trusted requirements of the entire project.

8. Keywords (KW), refer to the functions and attributes of trusted meta-requirements. Keywords in a meta-variable should be expressed as an array, i.e., a meta-variable can have more than one classification. Such as verification code requirements, its keywords can be: security, login, registration, etc. This attribute and granularity attribute are used to merge, and reuse trusted meta-requirements.

9. Generality (G), which is used to determine whether the trusted meta-requirement should be attached to the requirements database to be reused, is expressed by Boolean type, and the trusted meta-requirement generality should be set to false only for the particularity of the current project.

10. Cost (C), which represents the estimated time for the trusted meta-requirements to be realized and measures by day.

2.1.3 Definition of relationships between trusted meta requirements

Trusted meta-requirements relationship elements refer to the relationship between trusted meta-requirements and other trusted meta-requirements or other untrusted requirements. It is the index of the relation between requirements, and it is the basis of knowing if the requirements can coexist and combine with each other in a software.

1. Conflicts (CF), which means that the trusted meta-requirement R1 is in conflict with the requirement R2, means that the implementation of R1 hinders the implementation of R2, and vice versa.

2. Negative impact (NI) means that the implementation of trusted meta-requirements will affect or even damage other requirements, which will lead to the

decline of software quality.

3. Repress (RP) refers to the conflict between the requirements of trustworthy elements on the influence of trustworthy attributes. If there is only repress relationship but no promotion relationship between them, it is called a complete repress relationship.

4. Promotion (PM) refers to the positive influence between the requirements of trustworthy items on trustworthy attributes. If there is only a relation of promotion but not repress between them, it is a relation of complete promotion between them.

5. Same particle size (SPS) means that trusted requirements have the same granularity and can be merged, while trusted requirements without the same granularity cannot be merged.

2.1.4 Other definitions

Other elements refer to some elements which will have an impact on trusted meta-requirements in the trusted software development process.

1. Stakeholder (SH), the direct or indirect proponent of requirements, will have an impact on project decision-making according to their roles in the development process [18], and can give the weight value of requirements when proposing requirements, and coordinate the conflicts between them through the weight value.

2. Credibility tendency (CT), which indicates the tendency of the whole project to trustworthiness attributes, determines the effect of the choice of trustworthiness meta-requirements of the whole project, which is determined by stakeholders. The classification of trustworthiness tendency is based on trustworthiness indicators: availability tendency (ATT), security tendency (STT), reliability tendency (RBTT), instantaneity tendency (ITT), survivability tendency (SVTT) and maintainability tendency (MTT).

3. Other requirements (OR), requirements that be independent of trusted attributes.

4. Trusted budget (B), the trusted budget of the whole project, determines the choice of trusted requirements.

5. Meta Requirement Database (MRD), which regards object classes as reusable sub-items and organizes a class database [19]. The reusable meta-requirements are stored in the database, and the weighting attribute for meta-requirements in the database is 0.

2.2 Visual Representation of Trusted Meta-Requirement Model

Based on the above elements of trusted meta-requirements, the trusted meta-requirements model is visually represented graphically, as presented in the Fig. 1:

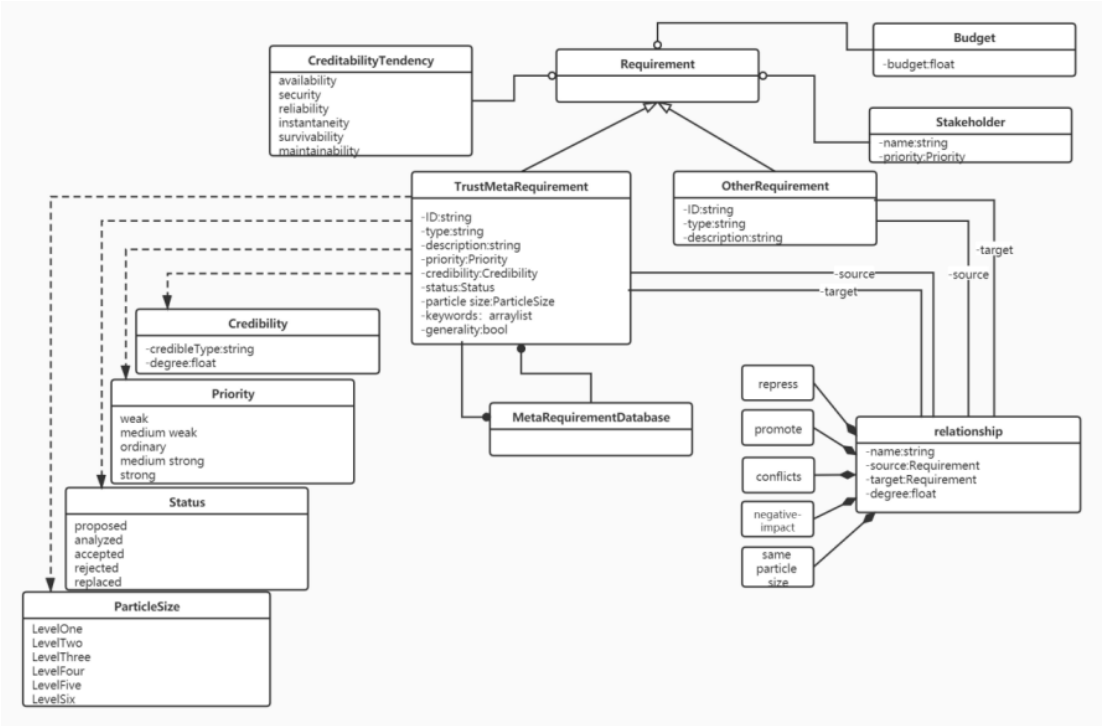

Fig.1. Trusted Meta Requirements Model

As demonstrated in Fig. 1, this figure is based on the Class diagram of UML, and I added two symbols to better represent the relationships between elements. The meta-requirement database class in association with the meta-requirement class by a black solid circle, which represents the contribution relationship, that is, a part of the requirements stored in the meta-requirement database will contribute to the trusted meta-requirement of the whole software, which is represented by extracting reusable trusted meta-requirements from the meta-requirement database. Conversely, trusted meta-requirements in the software will also contribute to the meta-requirements database. It proves that some of the newly proposed trusted meta-requirements in software development ought to be stored in the meta-requirements database.

The Budget class, the Stakeholder class, and the Credibility Tendency class are linked with the Requirement class by hollow circles, indicating that they have an influence on the Requirement class, which is mainly manifested in the screening of requirements by the model according to them, so they will have an impact on the final requirements of the whole software system.

3. Model-based Trusted Requirements Acquisition Process

3.1 Trusted requirements acquisition process

According to the Trusted Meta Requirements model, this paper proposes a

method and process of obtaining trusted requirements on the basis of the trusted meta-requirements under this model, addressing to simplifying the process of obtaining trusted requirements, as shown in Fig. 2:

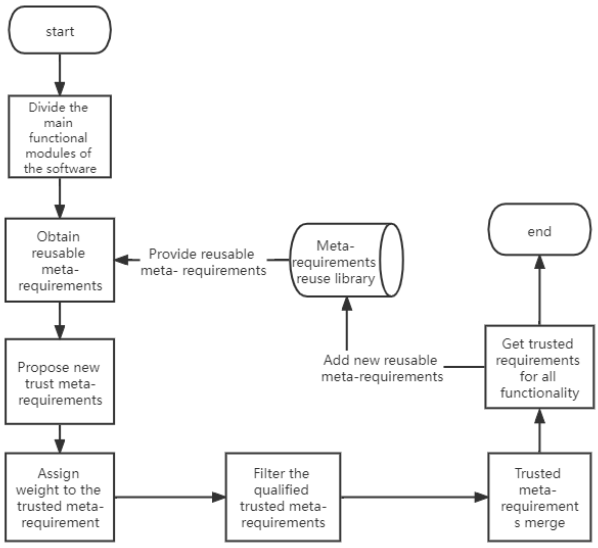

Fig.2 Trusted Requirements Acquisition Process

3.2 Screening and merging rules of trusted meta-requirements

This module will summarize the rules and algorithms of screening and merging trusted meta-requirements in the process of obtaining trusted requirements based on trusted meta-requirements model proposed above.

3.2. 1 Trusted Meta Requirements Screening Rules

After screening, the trusted meta-requirements should satisfy that the overall credibility of the meta-requirements set should be greater than or equal to the trustworthy tendency of software. After screening, the trusted meta-requirements should meet that the overall cost of meta-requirements set is less than the software budget. The screened trusted meta-requirements should respect the non-conflict relationship between requirements.

At the time that there is a conflict between the trusted requirement set and the current trusted requirement set and the trusted requirement set does not meet the credibility tendency, the scheme that maximizes the weight of the whole trusted requirement set should be selected. Eventually, the trusted meta-requirement set is the one with the largest weight value in the meta-requirement set that satisfies the used situation.

3.2. 2 Trusted Meta-requirements Merge Rules

The merging operation of trusted meta-requirements is to merge the trusted

meta-requirements in the finally screened trusted meta-requirements set TMR, and the merging rules are as follows.

(1) Only trusted requirements with same particle size can be merged.

(2) Keywords are also classified by particle size, such as secondary particle size keywords: mall, personal information, members, and five granularity keywords: viewing orders.

(3) Merge according to granularity from large to small. Start from the second level.

3.2.3 Reusable Trusted Meta-requirements Receipt Rules

After the merge process, the trusted meta-requirements that meet the following conditions should be put into database, and the weight of those trusted meta-requirements should be set to 0.

(1) The requirement is a trusted meta-requirement that does not exist in the trusted meta-requirement database.

(2) The generality value of the incoming trusted meta-requirement should be true.

4. Experimental simulation and result analysis

In order to validate the availability of the trusted meta-requirements model and embody the validity of trusted requirements acquisition under this model, a small program requirement of a company is used for simulation.

4.1. Brief introduction of experimental cases

A company operates a creative park. Its main business is to organize vehicle exhibitions and run restaurants and vehicle parts stores. The company's software requirements are : mall system: display merchant information and online sales of goods; Member system: responsible for member handling, member coupons and points functions; Activity function: publishing daily activities of the park and preferential activities launched by merchants. Software developer add personal information and login function.

4.2. Confirm the partition of software modules

According to the communication between the two sides of the project and the demand documents, the extracted software modules are divided as follows: mall, activities, member, personal information and login.

4.3. Extracting Reusable Trusted Meta-requirements from Meta-requirements database

Reuse is a fundamental activity that improves the quality and productivity of software products [20]. In this paper, part of the requirements is extracted from

the meta-requirements database, which avoids some repetitive work and improves the efficiency of acquisition. The existing reusable trusted meta-requirements in the meta-requirements database are as follows:

Change password: {
id:5001, tpye:'Functional requirements', Description: 'Modify the user's password', priority:null,credibility:[availability:0.6,security:0.2,reliability:0,instantaneity:0,survivability:0,maintainability:0],status:waited,particle size:LevelFive,
keywords: ["Personal Information", "Settings", "Users", "Passwords", "Availability", "Security"], generality: true, cost: 1};

Modify username:{
id:5002, tpye:‘'Functional requirements', description: 'Modify the user's username',priority:null,credibility:[availability:0.6,security:0.2,reliability:0,instantaneity:0,survivability:0,maintainability:0], status:waited,particle size: LevelFive,keywords: ["Personal Information", "Settings", "User", "User Name", "Availability", "Security"],generlity: true, cost: 1};

User login: {
id:5003,tpye:'Functional requirements', description:'Login function',priority:null, credibility:[availability:0,security:0.8,reliability:0,instantaneity:0,survivability:0, maintainability:0], status:waited,particle size: LevelFive,keywords: ['Login', 'User', 'Password', 'User Name', 'Security'],generality: true, cost: 3};

Login verification code:{
id:6001,tpye: 'Non-functional requirements', description: 'Verification code verification function at login',priority:null,credibility:[availability:0,security:0.4, reliability:0,instantaneity:0,survivability:0.3,maintainability:0],status:waited,particle size: LevelSix,keywords:["Login", "User", "Verification Code", "Security", "Survivability"],generality: true, cost: 1};

Quick response:{
id:6002,tpye: 'Non-functional requirements', description: 'When the user sends a request, the program should give timely feedback',priority:null,credibility:[availability:0.4,security:0,reliability:0,instantaneity:0.8,survivability:0,maintainability:0], status:waited,particle size: LevelSix,keywords:["System", "Availability", "Real Time"],generality: true, cost: 2};

Click feedback:{
id:6003,tpye: 'Non-functional requirements', description:'When the user clicks a button, he should give an obvious feedback'.priority:null,credibility:[availability :0.5,security:0,reliability:0,instantaneity:0,survivability:0,maintainability:0],status :waited,particle size: LevelSix,keywords:["System", "Button", "Click", "Availability"],generality: true, cost: 1};

Data security: {
id:6004,tpye:'Functional requirements', description: 'Take measures to ensure security when obtaining data from or sending data to

clients',priority:null,credibility:[availability:0,security:0.8,reliability:0,instantaneity:0,survivability:0,maintainability:0.5],status:waited,particle size:LevelSix, keywords: ["Data", "System", "Security", "Maintainability"],generality: true, cost: 4};

Location information:{
id:6005,tpye: 'Non-functional requirements', description:'Get the user's location information',priority:null,credibility:[availability:0.4,security:0,reliability:0,instantaneity:0,survivability:0,maintainability:0], status:waited,particle size: LevelSix,keywords:["Personal Information", "Availability"],generality: true, cost: 1};

Message push: {
id:6006,tpye: 'Non-functional requirements', description:'Push relevant information to users',priority:null,credibility:[availability:0.5,security:0,reliability:0,instantaneity:0,survivability:0,maintainability:0],status:waited,particle size: LevelSix,keywords:["Events", "Members", "Mall", "Availability"],generality: true, cost: 1};

Payment function: {id:5004,tpye:'Functional requirements', description:'Enable users to make payments',priority:null,credibility:[availability:0.8,security:0, reliability:0,instantaneity:0,survivability:0,maintainability:0],status:waited,particle size: LevelFive,keywords:["Membership", "Mall", "Availability"],generality: true, cost: 2};

User registration: {
id:5005,tpye:'Functional requirements', description: 'User registration account function',priority:null,credibility:[availability:0.8,security:0,reliability:0,instantaneity:0,survivability:0,maintainability:0], status:waited,particle size: LevelFive,keywords:["Login", "User", "Availability"], generality: true, cost: 2};

Accessibility:{
id:6007,tpye: 'Non-functional requirements', description:‘'Providing barrier-free functions to users', priority:null,credibility:[availability:0.3,security:0, reliability:0,instantaneity:0,survivability:0,maintainability:0],status:waited,particle size: LevelSix,keywords:["Users", "Availability"],generality: true, cost: 1};

Report a bug function: {
id:6008,tpye: 'Non-functional requirements', description:'Users can submit bugs to the background', priority:null,credibility:[availability:0,security:0,reliability:0, instantaneity:0,survivability:0.4,maintainability:0.5], status:waited,particle size: LevelSix,keywords: ["Settings", "Maintainability", "Survivability"],generality: true, cost: 1};

According to the module division, the trusted meta-requirements extracted are as follows: modify password, modify user name, user login, login verification code, quick response, data security, click feedback, information push, payment

function, user registration, barrier-free function, report a bug function and location information.

4.4 Propose new trusted meta-requirements

Propose requirements based on the model, the requirements are more standardized and the accuracy of the requirements will enhance. According to the communication between the two parties of the project and the requirements documents, the extracted new trusted meta-requirements are as follows:

Coupon reminder:{
id:6009,tpye: 'Non-functional requirements', description: 'Remind users when the coupon is about to expire', priority:null,credibility:[availability:0.4,security:0, reliability:0,instantaneity:0,survivability:0,maintainability:0],status:waited,particle size: LevelSix,keywords: ["Availability", "Membership", "Mall"],generality: false, cost: 1};

Message token:{
id:6010,tpye: 'Non-functional requirements', description:'Mark messages with circle numeric subscripts',priority:null,credibility:[availability:0.4,security:0, reliability:0,instantaneity:0,survivability:0,maintainability:0],status:waited,particle size: LevelSix,keywords:["Personal Information", "Availability", "Messages"],generality: true, cost: 1};

Activity display: {
id:6011,tpye: 'Non-functional requirements', description:'Activities related to goods or shops are displayed on the page of goods shops',priority:null,credibility:[availability:0.5,security:0,reliability:0,instantaneity:0,survivability:0,maintainability:0], status:waited,particle size: LevelSix,keywords: ["Activity", "Mall", "Availability"],generality: false, cost: 1};

Data cache:{
id:6012,tpye: 'Non-functional requirements', description:'Cache part of high-frequency access data',priority:null,credibility:[availability:0,security:0, reliability:0.5,instantaneity:0.7,survivability:0,maintainability:0],status:waited,particle size: LevelSix,keywords:["Activity", "Mall", "Data", "Reliability", "Real-time"],generality: true, cost: 1};

4.5 Giving the Trustworthy Element Demand Weight and Determining Trustworthy

Because stakeholders and experts give weight to requirements, software demanders can finally get more satisfactory software. According to the determination of stakeholders and experts, the final demand weight is as Table 1:

Table 1

Trusted Demand Weight Table

Change password: strong	**Modify username:** medium strong	**User login:** strong
Quick response: strong	**Login verification code:** medium strong	**Data security:** strong
Payment function: strong	**Click feedback:** ordinary	**Message push:** ordinary
User registration: strong	**Coupon reminder:** ordinary	**Message token:** ordinary
Activity display: medium weak	**Accessibility:** weak	**Data cache:** strong
Report a bug function: medium weak	**Location information:** medium weak	

The overall trustworthiness tendency of the software is determined as follows: CT= {5.5, 1.5, 0.3, 0.8, 0.4, 0}, software trusted budget is: budget=22.

4.6 Trusted meta-requirement screening

4.6.1 Screening by Trustworthy Propensity and budget

In this paper, the credibility of the whole software is calculated by the way of credibility addition.

From 4.5, we can see that the whole software trustworthiness tendency is CT= {5, 1.5, 0.3, 0.7, 0.3, 0}, and the trusted budget of software is: budget=22. According to this, the combination of satisfying conditions is as follows (expressed by id + name):

Scheme 1: {5001 change password, 5002 modify username, 5003 user login, 6004 data security, 5004 payment function, 5005 user registration, 6012 data cache, 6001 login verification code, 6002 quick response, 6003 click feedback, 6006 information push, 6008 report a bug function, 6009 coupon reminder, 6010 message mark, 6011 activity display}, cost=22

Scheme 2: {5001 change Password, 5002 modify username, 5003 user login, 6004 data security, 5004 payment function, 5005 user registration, 6012 data cache, 6002 quick response, 6003 click feedback, 6005 location information, 6006 information push, 6008 report a bug function, 6009 coupon reminder, 6010 message mark, 6011 activity display} , cost=22

Scheme 3: {5001 change password, 5002 modify username, 5003 user login, 6004 data security, 5004 payment function, 5005 user registration, 6012 data cache, 6002 quick response, 6003 click feedback, 6006 information push, 6008 report a bug function, 6009 coupon reminder, 6010 message mark, 6011 activity display}, cost=21

4.6.3 Determined by weight

According to Table1, the weights of the three schemes are as follows:

Scheme 1:11 , Scheme 2: 10.75 , Scheme 3:10. 5

Therefore, the final choice is Scheme 1.

4.7 Trusted meta-requirements merging results

Merge by keyword and particle size, merge from level one to level six. First, level one: System, which includes the trusted meta-requirements of the whole system, then merge the second level. For example, the trusted meta-requirements with Login keywords include user login, login verification code and user registration，so merge them together. Merge in this way until the merging is completed. For example, user login and login verification code in the Login keyword have the same five-level keyword Username, so the final consolidated result of the Login module is Login: {User: {Username: {user login, login verification code}, user registration}.

So, the final trusted requirements are as follows:

System: {

6004 Data Security,6002 quick response,6003 Click Feedback,

Mall: {Availability: {6006 information push, 5004 payment function, 6009 coupon reminder, 6011 activity display},6012 Data Cache},

Activities: {User: 6012 data cache, Availability: {6011 activity display, 6006 information push}},

Member: {Availability: {6006 information push, 5004 payment function, 6009 coupon reminder}},

Personal information: {Settings: {Security: {5001 change password, 5002 modify username}}, 6008 report a bug function,6010 message token},

Login: {User: {Username: {5003 user login, 6001 login verification code},5005 user registration}}

}

Compared with other methods of obtaining trusted requirements, using trusted mata-requirement model to obtain trusted requirements offers following advantages：

1. In comparison to the survey method, the use of meta-requirements database improves efficiency. In the beginning of the requirements communication, developers can first show the trusted meta-requirements in the meta-requirements database to the software demanders, in such a way that the software demanders can extract the requirements required by the project from the database, to improve the efficiency of requirements acquisition, and additionally, enable developers to reuse the codes related to these requirements to improve the efficiency of software development.

2. In comparison with scenario-based method, the expression of trusted meta-requirement permits the software demanders to propose their own ideas in a more detailed manner according to the different kinds of elements of trusted meta-requirement defined in this paper, in place of proposing an extensive scene for software developers to refine later. Because software developers in the process of refining requirements may appear and proposes the idea of different requirements,

makes the requirements and that demanders participant really need quite dissimilar. So, using the trusted meta-requirement model to obtain trusted requirements will more precise, and can reduce the cost led by the rework.

3. Using keywords to merge trusted meta-requirements can help software developers accurately understand what trusted requirements need to be developed in each module, enhance development efficiency, and reduce development errors.

4. Compared with the knowledge-based requirements acquisition method, the learning cost of trusted meta-requirements is lower. As long as the relevant elements of trusted meta-requirements are introduced to the software demander and the concept of meta-requirements is understood by demander. The software demander can communicate with the software developer in accordance with the trusted meta-requirement.

5. Conclusion

In order to come to terms with the increasingly complex situation of trusted software requirements acquisition, this paper integrates the concept of meta-requirements into trusted requirements, puts forward the concept of trusted meta-requirements and establishes a trusted meta-requirements model, and on this basis proposed a trusted requirements acquisition process based on the trusted meta-requirements model, which can be summarized as follows:

(1) This research combines trusted requirements with meta-requirements and study the basic elements of trusted meta-requirements.

(2) Based on elements, the trusted meta-demand model is established.

(3) According to the trusted meta-requirement model, the trusted requirement acquisition process is proposed.

(4) The feasibility of the model and acquisition process is verified by experimental cases.

In this paper, the calculation algorithm of trustworthy tendency and weight used in the selection of trusted requirements is simple and rough, so the accuracy of obtaining trusted requirements is insufficient. In the future, optimization algorithms can be introduced to screen trusted requirements to improve the accuracy, and to determine the best scheme among all feasible schemes.

R E F E R E N C E S

[1] Chenshijie Security design of embedded system based on trusted computing technology [d] Chengdu: University of Electronic Science and technology, 2018.

[2] *Jacquin L, Shaw A L, Dalton C.,* Towards trusted software-defined networks using a hardware-based Integrity Measurement Architecture. //Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft), 2015.1-6.

[3] *Ong MIU, Ameedeen MA, Kamarudin IE.,* Approaches in Creating Meta-requirement: A Systematic Literature Review. IOP Conference Series: Materials Science and Engineering,

2020, 769(1):012049 (8pp).
[4] *Anu V, Hu W, Carver J.C., et al.* Development of a Human Error Taxonomy for Software Requirements: A Systematic Literature Review. Information & Software Technology, 2018, 103(NOV.):112-124.
[5] *Ong MIU, Ameeden MA.* Meta-Requirement Mapping Model. IOP Conference Series Materials Science and Engineering, 2020, 769:012051.
[6] *Gerwinn S.,* A Compositional Safety Specification Using a Contract-Based Design Methodology. 2015.
[7] *Saoud Z, Faci N, Maamar Z, et al.* A Fuzzy-based Credibility Model to Assess Web Services Trust under Uncertainty. Journal of Systems & Software, 2016, 122(2016): 496-506.
[8] *BAUDRY B, NEBUT C, Le TRAON Y.* Model-driven engineering for requirements analysis//Proc of the 11th IEEE International Enterprise Distributed Object Computing Conference. [S.l.]: IEEE, 2007:459-466.
[9] *Natalie, Thomas Lee, Wang Xu, Yu Qian, Yu Yong, Jolie.* Formal representation and satisfiability analysis of non-functional requirements of trusted software. Acta Software Sinica, 2015, 26 (10): 2545-2566.
[10] *Sun Jinwen, Kang Yan, Yang Xiaodong*. Non functional requirements evaluation method of trusted software based on FAHP [J] Computer engineering and application, 2017,53 (11): 223-228 + 270.
[11] *Xiao zhetao, Liu Zhenyu.* Formal modeling and analysis of software requirement defects based on UML [J] Instrumentation and automation: 2019.12-83 DOI:10.14016/j.cnki. 1001-9227.2019.12.083.
[12] *Rouland Q, Hamid B, Jaskolka J.* Specification, detection, and treatment of STRIDE threats for software components: Modeling, formal methods, and tool support[J]. Journal of Systems Architecture, 2021, 117(5):102073.
[13] *Ramos H, Fonseca M, Ponciano L. 2021.* Modeling and Evaluating Personas with Software Explainability Requirements, in: Artificial Intelligence Research. Artificial Intelligence Research, pp. 136–149. doi:10.1007/978-3-030-92325-9_11.
[14] *Ong M , Ameeden M A . Meta-Requirement Mapping Model[J]. IOP Conference Series Materials Science and Engineering, 2020, 769:012051*.
[15] *LUO Xin-xing, WANG Wei-fang.* Research on meta-requirement model and meta-requirement set based on ontology. Application Research of Computers, 2012, 29(04): 1391-1394.
[16] *Ong MIU, Ameedeen MA, Kamarudin IE.* Meta-requirement Method Towards Analyzing Completeness of Requirements Specification// Proceedings of the Future Technologies Conference. Springer, Cham, 2018.
[17] *Shirakawa, Natalie, Wang Xu, Kang Yanni.* Economic method analysis of non-functional requirements satisfiability of trusted software. Computer Engineering and Application, 2017, 53 (22): 249-257.
[18] *Adanma Cecilia Eberendu, Edem OkonPeter Akpan, Emmanuel C. Ubani, et al.* Analysis of Stakeholder Influence: A Perception of Software Engineering Projectsin Nigeria. 2017, 11(2):1-8.
[19] *Zhou Miao.* Research on software requirements reuse and implementation of its tools. Huazhong University of Science and Technology, 2007.
[20] *Ya'UBI, Nordin A, Salleh N.* Meta-Modeling Constructs for Requirements Reuse (RR): Software Requirements Patterns, Variability and Traceability. Malaysian Journal of Computing, 2018, 3(2): 119-137.