Generating Requirements Documents for Embedded Systems: A Device Knowledge-Guided Approach

Chunhui Wang[1], Jiaqi Zhao[1], Xiaohong Chen[2], Zhi Jin[3,4]
[1] College of Computer Science and Technology, Inner Mongolia Normal University, Hohhot, China
[2] Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China
[3] School of Computer Science, Peking University, China
[4] Key Lab of High-Confidence of Software Technologies (PKU), Ministry of Education, China
ciecwch@imnu.edu.cn,zhaojiaqi30@163.com,xhchen@sei.ecnu.edu.cn, zhijin@pku.edu.cn

***Abstract*—Requirements documentation is an important activity in requirements development. Embedded systems consists of software and hardware, as well as multiple stakeholders. As systems grow in size and complexity, writing requirements documents becomes more time-consuming and laborious. Based on the strong correlation between embedded system and physical device, this paper proposes to use device knowledge base to support requirements documentation. Stakeholders only need to fill in the intention and device selection, and then the device and system requirements are supplemented by device knowledge. Templates and environment modeling and analysis methods are proposed to further guide requirements acquisition. This leads to considerably simplify and accelerate the requirements document development for embedded systems.**

***Index Terms*—requirements documentation, embedded system, device knowledge-guided, requirements document generation, requirements template**

I. Introduction

Requirements documentation is an important process in software development [1]. Requirements documents need to be written based on the needs of the business domain, and the documents must be easily understood by domain users/customers and software engineers. However, as software systems grow in size and the domain-dependent nature of much software increases, writing requirements documents that can be clearly understood by multiple stakeholders can be costly in manpower and time.

Some studies have proposed automatic requirements document generation methods. These methods first need to acquire some kind of requirements model, such as a UML model [2] or scenario graph [3], or ontology [4], and then define rules to implement the generation from the model to the document. Other studies have focused on the design of requirement templates to guide and standardize the writing of requirements document, such as defining requirements description templates for embedded systems [5], [6].

Embedded system consists of software and hardware. In addition to describing the requirements information such as purpose, scope, definition, product functions and constraints, the requirements document also needs to express the characteristics of embedded system requirements such as hardware interfaces and device constraints (e.g., availability, reliability, security). Requirements documentation is more challenging because it involves many stakeholders in the business domain, software domain, and hardware domain [7].

For an embedded system, after stakeholders describe the intention of the target system, it is usually necessary to select suitable devices to achieve the intention. Normally, the device is defined in advance before developing the embedded software, and the system requirements (i.e., the ability of the system to interact with the device) can be cleared through the device knowledge. In addition, device knowledge will provide constraint information, such as hardware limitation, parallel operation, protocols, safety and security considerations. In fact, the device knowledge is the environment knowledge that is required for deriving specifications from requirements [8].

In this paper, we first defines 4 requirements levels of embedded systems (i.e., intention, system requirements, device requirements and software requirements), and then proposes a framework for generating requirements documents containing these 4 requirements levels by means of device knowledge-guided method. Specifically, the framework includes three phases, which are gathering requirements information, automatic supplement of requirements information and automatic generation of requirements document. In the first phase, we use templates to guide stakeholders to fill in the embedded software development intention and select which devices to use. In the second phases, we supplement and derive the necessary requirements information (e.g., device requirements, system requirements, software requirements, device constraints, etc.) with the help of the device knowledge base and environment modeling methods. Finally, we have detailed rules for automating the generation of requirements documents by synthesizing the gathered requirements information and models. This way will minimize the burden of requirements documentation for embedded system.

The rest of this paper is organized as follows. Section II introduces requirements levels of embedded systems to clarify the elements of requirements documentation. Section III presents the proposed generation process for embedded system requirements documents. Section IV shows a case study for demonstrating the document generation process. Section V compares our work with related work. Finally, section VI concludes this paper.

Identify applicable funding agency here. If none, delete this.

II. Requirements levels of embedded systems

Embedded system consists of hardware and software with specific functions. Among them, the software component is used as the controller to coordinate the system devices to complete the design intention of the embedded software system, and the hardware component includes various sensors and actuators and other devices that can be managed and scheduled. Sensors are responsible for obtaining the external environment properties (such as people, physical environment, natural environment or software), which are the devices in the system and the external interface of the system. Actuator are responsible for executing the instructions issued by the controller, exerting operations on the external environment and producing effects. The composition of the embedded system is shown in the Fig. 1.

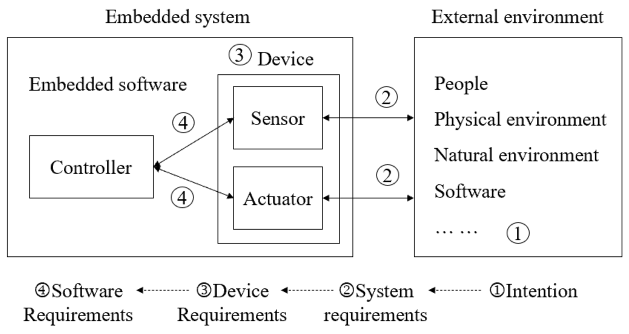


Fig. 1. Requirements level model for embedded systems

According to the embedded system composition architecture shown in Fig. 1, the requirements of embedded software systems are divided into *intention, system requirements, device requirements and software requirements*. Among them, the intention expresses the goal of the system, follows the requirements engineering concept of environmental modeling [9], and concretely expresses the expected effect of the software system on the external environment, that is, the expected changes and conditions on the entity of the external environment. For example, for the "intelligent elderly care system", the intention is that "Through a robot to sense the location of the elderly and autonomously follow the elderly, when the system finds abnormal conditions (such as the elderly fall), it will notify the elderly's family and doctors in time." Here, the elder (location), doctors, family members, etc. are all entities in the external environment.

System requirements describe the capability of an embedded system to interact with the external environment in order to achieve its intention. For example, in order to achieve the elderly care system, the system must be able to measure the location of the elder, be able to tell if the elder has fallen, and be able to send messages to family members and doctors.

In embedded systems, device requirements involve a variety of sensors and actuators. It reflects the ability of the device, such as the robot in the intelligent care system could track the location of an elderly person, that is, the robot has the ability to follow the elder. Software requirements refer to the interaction between the controller and the device and its related constraints, which are divided into functional requirements and non-functional requirements. For example, "The software controls the robot to track the elderly" is a function requirement, while "maintaining a safe distance when the robot is tracking the elderly" is a constraint on the robot and is a non-functional requirement.

III. Proposed Generation Process

A. Framework

To reduce the burden of embedded system requirements documentation, we proposed a device knowledge-guided embedded requirements document generation framework as shown in Fig. 2. This approach is based on human-computer interaction, i.e., stakeholders provide the necessary requirement information and supplement or confirm the requirement information, and the automated method provides templates to be filled, supplements the requirement information, creates the models and generates the requirements document.

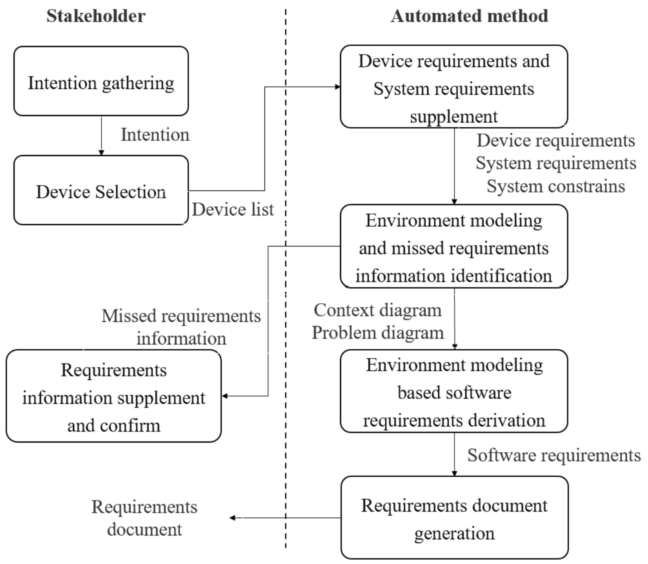

Stakeholder
Automated method
Intention gathering
Intention
Device Selection
Device list
Device requirements and System requirements supplement
Device requirements
System requirements
System constrains
Environment modeling and missed requirements information identification
Missed requirements information
Requirements information supplement and confirm
Context diagram
Problem diagram
Environment modeling based software requirements derivation
Software requirements
Requirements document
Requirements document generation

Fig. 2. Device knowledge-guided embedded system requirement document generation framework

In this process, device knowledge plays an important guiding role. Specifically, stakeholders first provide information about the *intention* and select the applicable *devices* from the list of devices provided by the device knowledge base. Then, the information of *device requirements* and *system requirements* are supplied with the support of the device knowledge base. Then, based on the above collected requirements information, environment models are constructed and the completeness of the requirements information is checked based on the extracted models. When missing requirements information is found, stakeholders are prompted to complete it. The completed information can be used to update the device knowledge base. Based on the built environment models, the required system behavior is automatically derived by state machine in device base to obtain software requirements [10]. After the *intention, device requirements, system requirements, software requirements and requirements models* are integrated, the requirements document will be generated. Further, the requirements document will be fed back to the stakeholders to support continued modification. The interactive process in Fig. 2 mainly includes following 5 key activities.

B. Template-guided requirements gathering

In this paper, we provide using requirements sentence template and requirements document template to facilitate the generation of embedded system requirements documents. The sentence template is used to prompt stakeholders to fill in or complete the necessary requirements information. The document template is used to describe the content, organization, and constraint for embedded system.

A requirements sentence template include requirements elements and fixed syntax elements. Among them, the requirements element is the requirements information to be supplemented, and the fixed grammar element is the invariant that plays the role of the prompt word. For example, "⟨System⟩ need ⟨Do What⟩, In order to ⟨Purpose⟩" is a requirements statement template that describes the *intention*. Where, the part with the brackets is the requirements element. This part will be filled in according to the instance. And "need, in order to" is the fixed grammar elements.

In this paper, we define 6 types of sentence templates (shown in Table I) for *intention, device selection, device requirements, system requirements, software requirements*, and *system constraints*, which are illustrated in Fig. 1. Where, the interactions in system requirements represent information such as physical quantities and control instructions that the device interacts with external entities. In the software requirements, the interactions represent information such as control signal and device states that the device interacts with controller.

TABLE I
REQUIREMENTS SENTENCE TEMPLATE FOR EMBEDDED SYSTEM

ID	Type	Sentence
01	Intention	⟨System⟩ need ⟨Do What⟩,In order to ⟨Purpose⟩
02	Device Selection	⟨System⟩ need ⟨Device List⟩
03	Device Requirements	⟨System⟩ need ⟨Device⟩, In order to ⟨Purpose⟩
04	System Requirements	[when ⟨Event|state⟩] ⟨System⟩ need ⟨Interaction⟩ with ⟨External entity⟩ to achieve ⟨system requirements⟩
05	Software Requirements	[when ⟨Event|state⟩] ⟨Controller⟩ need ⟨Interaction⟩ with ⟨Device⟩ to achieve ⟨software requirements⟩
06	System Constraints	⟨Constraint Quantity⟩ [equal,less,more,. . .] ⟨Value⟩

For the embedded system requirements document template, we refer to the IEEE standard specification [11]. It mainly includes 4 chapters. The first is the introduction of the system, including the intention and tasks of target system and the system composition. The second is the device requirements, including device list, device interface and device requirements. The third is the description of system requirements, including system requirements list and the corresponding requirements model describing its system requirements and device interaction. The fourth is the description of software requirements, including each software requirements details, requirements constrain (e.g., space requirements and time limitation) and corresponding environment models.

Template-guided requirements gathering process designs templates according to the categories of embedded system requirements shown in Fig. 1 and collect requirements information. Specifically, starting from the *intention*, it is required to fill in the intention of developing the embedded system and the main tasks and goals. Based on the intentions and tasks, the template of *device selection* is generated, that is, the device that achieves the intention or task is required to be selected in the device knowledge base. Then, according to the device selection, the template of the interaction between the device and the system to achieve the intention/task is generated to clarify the *system requirements*. Finally, according to the model derivation, the software requirements description template is generated, and the states and conditions of the software requirements are filled.

C. Device Knowledge-based Requirements Supplement

The device knowledge base provides detailed information such as the device name, device category, device usage, and interface description. With knowledge of devices, it can guide the requirements development team in deciding which devices to select to achieve the user/customer's intention. At the same time, after the device is identified, some requirements information (such as the interface of the device, the capability of the device, the data of the device interaction or the control signal of the device) can be clarified. These knowledge are filled into the requirements elements of the requirements sentence templates. Further, this requirements information and device knowledge base is used to assist in the generation of environment models.

In this paper, the device knowledge base is designed according to the attributes of embedded system device. The metamodel is shown in Fig. 3. The device knowledge base contains 5 kinds of attributes, i.e., name, type, application list, interface, and state machine. Among them, the types are divided into two categories: controller and sensor. The application list indicates the usage of the device and can include a variety of applications. In order to explain its usage, it is necessary to describe which physical quantities (indicators or parameters) to perform what action (gather data or control other device) to achieve what purpose (intention). The interface includes device's ports, data protocols, and constraints. A state machine shows several device states and transitions between states. The state machine expresses the state change of the device and reflects the behavior requirements.

Building a high quality device knowledge base is the basis to support this phase. In order to build the device knowledge

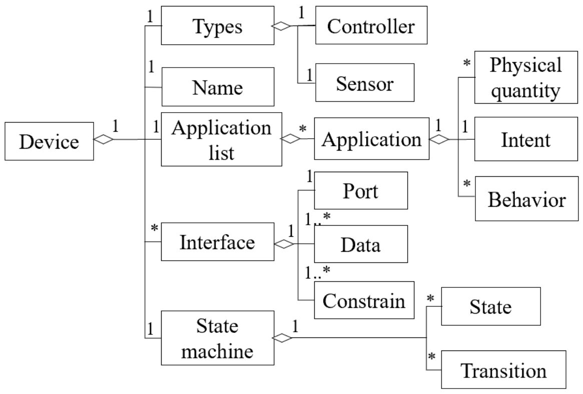


Fig. 3. Device knowledge base meta-model

base, we plan to use a large language model and carry out prompt engineering to extract the device knowledge base information from collected embedded requirements documents, and then generate the device knowledge base through manual confirmation and proofreading. Specifically, we firstly collect the embedded system requirements documents related with a specific domain (such as aeronautical control system). The large language model (e.g., GPT) is selected to carry out the prompt engineering, and the entity in the device knowledge conceptual model is extracted. In order to ensure the extraction quality, the results are fed back to the manual for confirmation after de-duplication and merging.

D. Environment modeling and missed requirements information identification

For embedded system, in order to fully express the system composition, device capability, system capability and other semantic information. We propose a problem framework approach to model embedded systems [12]. In the construction of the models, we combine rule-based and machine learning-based methods to extract the model elements and build the models. During the model building process, the missed model information will be checked and fed back to the stakeholders for supplement. For example, missing requirement information may be a lack of a certain type of entities in the system or a lack of necessary relationships between concepts.

To check the missed requirements information with the help of the models, we build the corresponding identification rules. The rules mainly include the following 3 situations:

- Entity type: The problem diagram model should include several concept categories such as intentions, external entities, sensors, controllers, and actuators. No one type should be missing from the model.
- Relationship: Each entity has a corresponding category of relationship with some other entity. If there is no a relationship on an entity or there is a wrong category of relationship on an entity, a prompt is fed back.
- Interaction information: There is interaction information between the *intention* or the *external entity*, and between the *external entity* and the *device*. The interaction information cannot be missing.

E. Environment modeling based software requirements derivation

This step is to derive software requirements from system requirements. It can be automated based on [10]. The basic idea is to use the environment model as a basis to provide expected software behaviors. Each device in the model is represented by a state machine that provides the control signal instigating state changes. System requirements impose constraints on devices, expressed via expected states. Consequently, the control signal can be identified and designated as a software behavior responsible for sending such a control signal.

For instance, Fig. 4 illustrates the derivation process of the software behavior required to turn the robot camera on. The system requirement necessitates the camera to remain in the ‘On’ state. From the state machine, we identify that a control signal ‘onPulse’ is needed. Consequently, we can derive a software behavior, represented as M!{onPulse}.

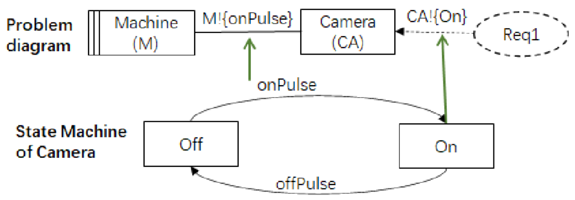


Fig. 4. An example of software requirements derivation

F. Template-Based Requirements Documents Generation

In order to generate the requirements documents, according to the defined format of the requirements document template, this paper combines the instantiated requirements sentence templates and requirements models obtained from the previous steps. Specifically, we will define requirements document generation rules, organize the content by chapter, and feed the generated documents back to the requirements development team. The requirements development team can edit the requirements document and finally obtain a high-quality requirements document. To support this process, an interactive requirements document generation tool will be developed.

Fig. 5 shows the mapping of the requirements information obtained from above steps, including *intention, device selection, device requirements, system requirements* and *models*, to the requirements document in four chapters.

IV. Case Study

In this paper, a sun search system is used to examine the requirements document generation process based on device knowledge. The main function of the sun search control software is to capture the sun, determine the current attitude of the satellite by collecting the measurement data of the gyros and the sun sensor, and then control the rotation of the satellite

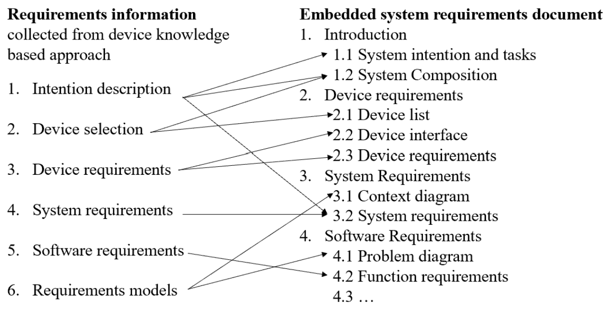


Fig. 5. The mapping from the requirements information to requirements document

so that the sun sensor can find the sun, and maintain the orientation of the satellite to the sun. The sun search system includes hardware devices (i.e., gyros, sun sensor, thruster, data management computer) and sun search control software to achieve the purpose of controlling satellite to find and track the sun.

When the device knowledge base related to the aeronautical field is built, device knowledge such as gyros, thrusters and sun sensors can be obtained beforehand. Fig. 6 shows an example of building a requirements document for the sun search system. Firstly, the system development intentions of the sun search system are filled in the intention template and the devices are selected. Then, based on the device knowledge, the templates of the device requirements and system requirements are automatically filled and the related requirements description is generated. Based on the requirements description supplied by device knowledge, the requirements models (i.e., context diagram and problem diagrams) are extracted and constructed. Then, the missed requirements information in the requirements models are identified and filled in. Next, the software requirements are derived based on the device state machine, problem diagram, and system requirements. Finally, these requirements information will be synthesized to form the embedded system requirements document.

V. Related work

There are some research efforts focused on (semi-) automatic generation of requirements documents to reduce the burden of requirements documentation. Some researchers put forward the method of generating requirements document from existing requirements models. For example, Bao et al. [2] proposed a tool to automatically generate IEEE requirements documents from UML models. They used four UML models and established rules for mapping from models to document. Yanuarifiani et al. [4] defined ontologies to express requirement information and provided mapping rules from ontologies to documents. These methods rely on existing models to automate document generation by defining corresponding rules. However, due to the strong dependence on existing models, it is difficult to apply this method to complex systems with unclear requirements.

In order to improve the quality of requirements documentation, some researchers pay attention to the definition of requirements templates. For example, Denger et al. [5] designed a metamodel for the requirement description statements of embedded systems, paying attention to the language patterns of sentences, specifying the requirements elements, and completing the requirement description through pattern combination. Mavin et al. [6] proposed the EARS template, which includes five pattern sentence structures (ubiquitous, event-driven, unwanted behavior, state-driven, and optional features). For example, "While ⟨optional precondition⟩; when ⟨optional trigger⟩; ⟨system name⟩; shall ⟨system response⟩;" The sentence template shows that under what conditions and when what happens, how should the software system respond.

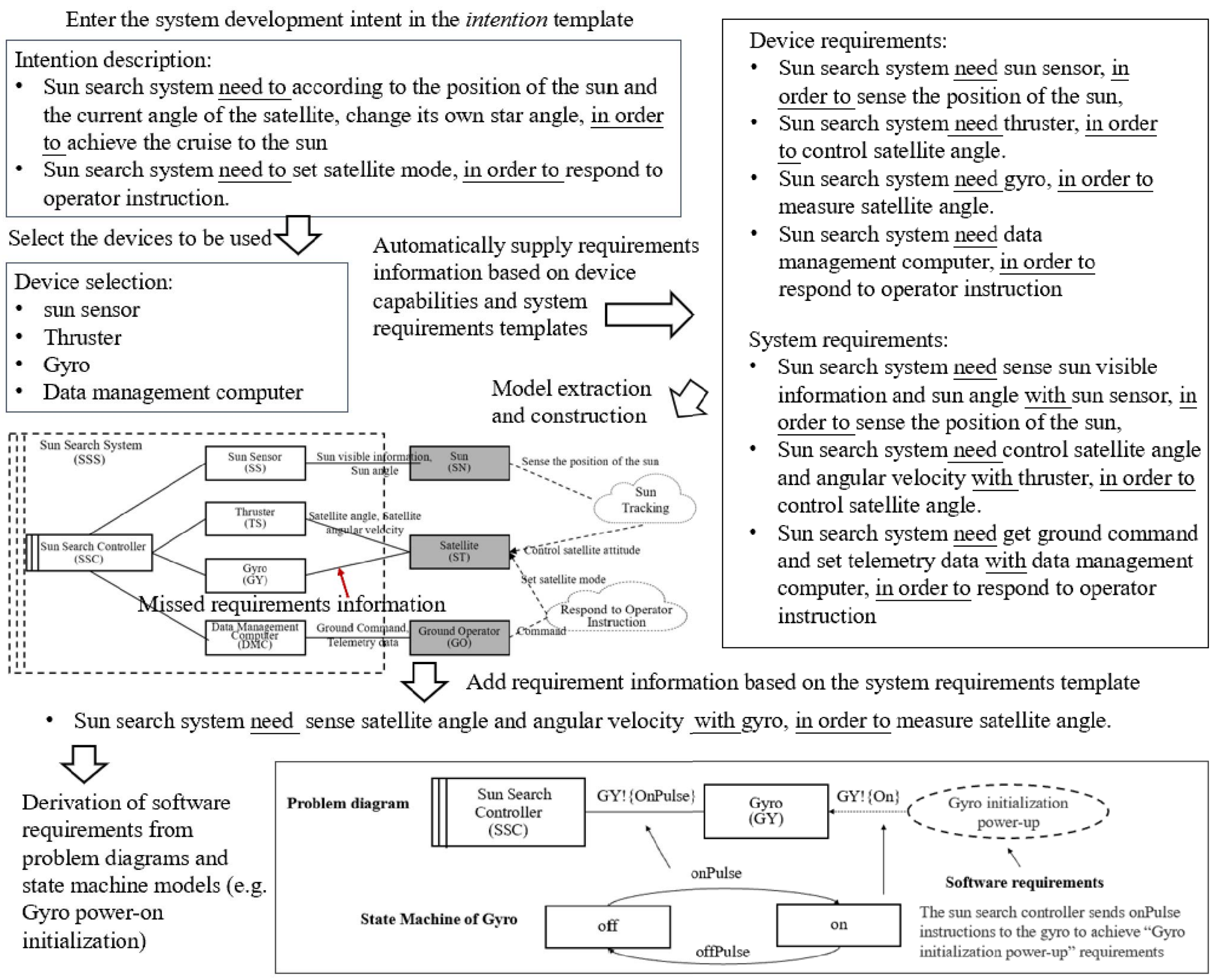
Other researchers believe that the specification of templates limits the requirements elicitation. They divided requirements documentation into three parts [13] (i.e., requirement discovery, requirement analysis and requirement specification), and the automatic generation of requirements documents is completed through tool support.

In our work, based on the characteristics of embedded system hardware devices, we propose to drive the generation of requirements documents with the help of device knowledge. On the one hand, this method provides guidance for the writing of requirements documents, which pays attention to the whole process of requirements acquisition. At the same time, with the help of the existing device knowledge, the writing time of documents is greatly reduced. On the other hand, with the help of templates and automated model construction and analysis methods, the incomplete requirements information can be effectively standardized and identified, and the writing efficiency of requirements documents can be further improved. Finally, through the interaction and confirmation of stakeholders, the writing of the requirements document is completed.

VI. Conclusion

Requirements documentation is an important activity in requirements engineering. Device knowledge provides the necessary information to clear the requirements of embedded system. In this paper, the device knowledge base is used to guide the writing of requirements document, reduce the writing time of requirements document and accelerate the requirements development. This paper puts forward the applicable requirements statement template and requirements document template, obtains the requirement information through the guide of the template, checks the completeness of the requirement information with the help of model extraction and analysis, combines the requirement information and model by rules, and then generates the requirements document. A case study is explored to demonstrate the process of the approach provided in this paper.

As for our next step work, we plan to develop a prototype tool that implements our proposed framework. In addition, we will conduct large scale experiments on real data sets to verify the effectiveness of the proposed approach.


Enter the system development intent in the *intention* template
Select the devices to be used
Automatically supply requirements information based on device capabilities and system requirements templates
Model extraction and construction
Sun Search System (SSS)
Sun Sensor (SS)
Thruster (TS)
Sun Search Controller (SSC)
Gyro (GY)
Data Management Computer (DMC)
Sun visible information, Sun angle
Satellite angle, Satellite angular velocity
Ground Command, Telemetry data
Sun (SN)
Satellite (ST)
Ground Operator (GO)
Sense the position of the sun
Sun Tracking
Control satellite attitude
Set satellite mode
Respond to Operator Instruction
Command
Missed requirements information
Add requirement information based on the system requirements template
Derivation of software requirements from problem diagrams and state machine models (e.g. Gyro power-on initialization)
Problem diagram
Sun Search Controller (SSC)
GY!{OnPulse}
Gyro (GY)
GY!{On}
Gyro initialization power-up
onPulse
Software requirements
State Machine of Gyro
off
on
offPulse

Intention description:
- Sun search system need to according to the position of the sun and the current angle of the satellite, change its own star angle, in order to achieve the cruise to the sun
- Sun search system need to set satellite mode, in order to respond to operator instruction.

Device selection:
- sun sensor
- Thruster
- Gyro
- Data management computer

Device requirements:
- Sun search system need sun sensor, in order to sense the position of the sun,
- Sun search system need thruster, in order to control satellite angle.
- Sun search system need gyro, in order to measure satellite angle.
- Sun search system need data management computer, in order to respond to operator instruction

System requirements:
- Sun search system need sense sun visible information and sun angle with sun sensor, in order to sense the position of the sun,
- Sun search system need control satellite angle and angular velocity with thruster, in order to control satellite angle.
- Sun search system need get ground command and set telemetry data with data management computer, in order to respond to operator instruction

- Sun search system need sense satellite angle and angular velocity with gyro, in order to measure satellite angle.

The sun search controller sends onPulse instructions to the gyro to achieve "Gyro initialization power-up" requirements

Fig. 6. Sun search system requirements eliciting based on device knowledge

ACKNOWLEDGMENT

This research was supported by the National Natural Science Foundation of China(62162051,62192731,62192730).

REFERENCES

[1] S.L. Pfleeger, and J.M. Atlee, Software Engineering: Theory and Practice. Upper Saddle River, New Jersey, USA, Prentice Hall, 2006.

[2] T. Bao, J. Yang, Y. Yang and Y. Yin, "RM2Doc: A Tool for Automatic Generation of Requirements Documents from Requirements Models," 2022 IEEE/ACM 44th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), Pittsburgh, PA, USA, 2022, pp. 188-192

[3] Xiaohong Chen, Zhi Jin: Capturing Requirements from Expected Interactions Between Software and Its Interactive Environment: An Ontology Based Approach. Int. J. Softw. Eng. Knowl. Eng. 26(1): 15-40 (2016)

[4] A. P. Yanuarifiani, F. -F. Chua and G. -Y. Chan, "Feasibility Analysis of a Rule-Based Ontology Framework (ROF) for Auto-Generation of Requirements Specification," 2020 IEEE 2nd International Conference on Artificial Intelligence in Engineering and Technology (IICAIET), Kota Kinabalu, Malaysia, 2020, pp. 1-6.

[5] C. Denger, D. M. Berry and E. Kamsties, "Higher quality requirements specifications through natural language patterns," Proceedings 2003 Symposium on Security and Privacy, Herzlia, Israel, 2003, pp. 80-9.

[6] A. Mavin, P. Wilkinson, A. Harwood and M. Novak. Easy Approach to Requirements Syntax (EARS). In proceedings of the 17th IEEE International. Requirements Engineering Conference. 2009. p. 317-322

[7] Y. Mengfei, G. Bin, D. Zhenhua, and J. zhi, "Intelligent program synthesis framework and key scientific problems for embedded software," Chinese Space Science and Technology, vol. 42, no. 4, pp. 1–6, 2022.

[8] Michael Jackson: The Meaning of Requirements. Ann. Softw. Eng. 3: 5-21 (1997)

[9] Z. Jin, Environment Modeling-Based Requirements Engineering for Software Intensive Systems. Massachusetts, USA: Morgan Kaufmann, 2018.

[10] Bian Han, Xiaohong Chen, Zhi Jin, Min Zhang: Approach to Generating TAP Rules in IoT Systems Based on Environment Modeling. Int. J. Softw. Informatics 11(3): 263-286 (2021)

[11] IEEE Std 830-1998, Recommended Practice for Software Requirements Specifications, IEEE Xplore, 1998.

[12] Chunhui Wang, Lu Hou, Xiaohong Chen: Extracting Requirements Models from Natural-Language Document for Embedded Systems. RE Workshops 2022, pp. 18-21.

[13] M. G. Georgiades and A. S. Andreou, "Automatic generation of a Software Requirements Specification (SRS) document," 2010 10th International Conference on Intelligent Systems Design and Applications, Cairo, Egypt, 2010, pp. 1095-1100.