Environmental Engineering with Software Environment Modelling Based on Threat Model

Qiuying Li¹,²,* and Ruize Sun¹,²
¹School of Reliability & Systems Engineering, Beihang University, China
²Science & Technology on Reliability & Environmental Engineering Laboratory, Beijing, China

*Corresponding author email: li_qiuying@buaa.edu.cn

Abstract. The influence of software environment and its changes on software safety are becoming more and more important. This paper analyzes the relationship between software environment and software safety requirement, and builds software environment model related to safety requirement based on threat model from a conceptual hierarchy. Finally, an application is given for case study.

Keywords: Software environment; Environment modelling; Threat model; Software safety.

1. Introduction
Software safety refers to the attribute of the software whose functions can avoid casualties and property losses, i.e. whose essence is failure safety [1]. Software safety threats refer to behaviors that may destroy software safety policies and safety attributes [2]. The threat model builds software safety threats.

Software environment describes the external condition that affect functions and performance of the software. Only when it is clear what tasks the software should complete and what external conditions will affect the completion of the tasks, can developers completely determine the functional requirements and performance requirements of the software [3-5]. These information and functional requirements constitute a complete description of the software requirements. Therefore, from this viewpoint, the software environment plays an important role in improving software requirements.

On the other hand, due to the diversity and complexity of the software environment, it is almost impossible for developers to determine all environmental information that may affect software functions and performance based on their understanding of software business requirements and the environment itself. In this case, developers need to find the environmental information that affects the expression of requirements based on the existing software requirements.

Then, the software environment is an important factor affecting software operation. In recent years, with the vigorous development of the internet, the application scenarios of software continue to expand. People hope that the software can not only work normally in a certain environment, but also that when the environment changes, the software can perceive such changes and adjust itself to ensure functions and performance. The software with this kind of characteristic is called self-adaptive software. Self-adaptive software refers to software that can monitor changes in the environment and the system itself, and can dynamically modify its behavior and structure in response to changes. No matter what kind of self-adaptive software, its adaptation is based on cybernetics with “collection-decision-control” as its core. It is essential for the perception and expression of environmental information and its changes. And the display effect also determines the level of adaptive ability [6].
Furthermore, the software environment is open, dynamic, uncontrollable and unpredictable. Recent research content of software environment is mainly divided into the description of the state information of software environment and the changing rules of software environment. The description of the state information helps to further improve the requirements and generate more comprehensive and effective test cases in software development, in order to ensure the normal expression of functions when the software is running, and quickly and accurately locate the source of faults. In addition, the environment itself is not dependent on the existence of software. Therefore, the environment can be reused on different software, that is, the description of the environment should be reusable, and modeling the environment can enhance the replication of the environment. However, the existing methods of modelling software environment mostly focus on describing the state of the environment at a certain moment, reflecting the current environment information, such as feature models, ontology models, etc. Based on the above mentioned, this paper studies the software environment modelling methods related to software safety based on threat model.

2. Modelling Software Environment Based on Threat Model

2.1. Overview
The environmental threat model refers to a model that is based on software safety goals to find threats in the environment and organize them based on the relationship between threats.[7]

2.2. Modeling Process
The flowchart of this method is shown in figure 1.

![Flowchart of environmental threat model method.](image)

Step 1: Identify top-level goal
The top-level goal refers to a direct manifestation of software safety requirement. Top-level means that failure to complete this goal will lead to the overall breakdown of software safety rather than the failure of a certain function of a module. The top-level target is not unique, but a threat model will correspond to a top-level target when it is determined.[8]

Step 2: Build target model
The target model is based on the functional module or logical sequence of the software. For example, in order to ensure the realization of the top-level goals, it is necessary to clarify the behaviors that each module of the software needs to perform. The behavior is positive, i.e., the module should normally achieve a certain function or should achieve certain performance. In this way, dividing the top-level goal into sub-goals of each module is called goal refinement. The end of target refinement is that each underlying target has its own entity undertaker. That is to say, the target cannot continue to be decomposed, and each underlying target has a certain physical entity or virtual entity to complete.

Step 3: Establish conceptual domain model
The conceptual domain model is the sorting of uncertain factors (UML class diagram) which identify the key elements of the system and their relationships (e.g. sensors, user interface). These factors cause environmental conditions and the system must be dealt with (whether it will cause major problems) uncertainty. The domain model is used to determine the uncertainty range of the system, that is, the elements in the domain model are either sources of uncertainty or used to monitor environmental conditions that cause uncertainty. The elements in the domain model mainly include two categories: one is the physical environment elements perceived by the software, including physical entities (such as sensors) and information describing the entities, the other is the virtual entities that interact with the software and the outside world (such as user interfaces).

Step 4: Build threat model
Start from the bottom layer of the target model, at each layer, the conceptual domain model is used to find threat elements that may affect the realization of the target. The possible impact of threat elements propagates upwards along the target model, and finally a model composed of threats is formed. The environmental threat model is essentially anti-model of the target model, which comes from the negation of the target model.

Step 5: Output environmental threats
Start from the bottom layer of the target model, in each layer, we look for threats that may affect the realization of the target from the conceptual domain model. The model can output possible environmental threats, which can be single or multiple. The structure predicts how these environmental threats will affect the goals of the software at all levels.

3. Case Study
Here take an adaptive software as an example, which is an environment-assisted living system, working in the following environment: there is a middle-aged woman (called A), who lives alone with high blood pressure and cholesterol, and there is a smart refrigerator in her room. It has 4 temperature and 2 humidity sensors, which can read, store and transmit RFID information on food packaging. The refrigerator environment assisted living system (AAL) communicates and integrates itself. In particular, it can detect the presence of spoiled food, find and receive the diet plan to be monitored based on the food that A eats. An important part of A’s diet is to ensure a minimum fluid intake. In order to improve the calculation accuracy, there are special sensor-enabled cups in the room: some cups have sensors that will beep when water is needed and have a certain liquid level to monitor the consumed fluid. They coordinate seamlessly to estimate the amount of liquid absorbed: the latter informs the former of the leak so that the water level can be updated. Sensors in faucets and toilets also provide a way to monitor this measurement.

The process of establishing an environmental threat model is shown as follows:
1) Establish the top-level goal: make A’s water intake as much as possible to reach the ideal value.
2) Establish a target model: the target model is obtained by refining the target, which is shown in figure 2.
Water intake is as ideal as possible

A normally drinks
Reminder device normally prompts
Ensure adequate water supply

A
Remind drinking water in time
Correctly calculate water shortage
Refrigerator

AAL
Utensils are special water cups
Calculate water consumption correctly
Correct calculation of theoretical intake

Special water cup
Ensure that all the water in the cup is taken in

Special water cup
Refrigerator
Faucet sensor

Figure 2. Target model.

3) Establish a conceptual domain model: a conceptual domain model is obtained by analyzing various entities in the room and their relationships, which is shown in figure 3.

Figure 3. Conceptual domain model.
4) Build threat model: start from the bottom of the target model, at each layer, look for threat elements that may affect the realization of the target from the viewpoint of the conceptual domain model, which is shown in figure 4.

Insufficient water intake

- A forgets to drink water
- AAL doesn't remind normally
- Insufficient water supply to refrigerator

- AAL doesn't promptly remind
- Wrong calculation of water shortage

- There are other water glasses involved
- Wrong calculation of water consumption

- Water pouring
- Water consumed for other reasons

Figure 4. Threat model.

5) Output environmental threats
Threat behaviors in the environment that affect the realization of the top-level goals include: water is consumed for other reasons but its consumption is not counted, and the water in the cup is poured before drinking. When there are other water cups in the room, it will cause the water shortage calculation to be wrong, and further cause AAL to fail to remind normally. In addition, A’s forgetting to drink water and the lack of water supply in the refrigerator and AAL’s failure to remind them together logically will lead to insufficient A’s water intake.

4. Conclusions
Threat models are mainly used in self-adaptive software that requires high safety to model threat behavior in the environment. Although the independent modelling of the environment is theoretically realized, its independence is mainly reflected in the abstract concept of the model. The definition of the resource by the model is very superficial, and the specific meaning of the resource and the extraction method are not pointed out, which makes the model can be used in the application. Software interaction is still inseparable from the required input.

This paper explains the background knowledge of the model, then proposes the definition and characteristics of the model in the software environment field, sorts out the model flow chart, explains the key steps in detail, and instantiates the model according to the process. There are shortcomings in the following aspects: only a conceptual hierarchical threat model is stressed, whose modelling process is still not detailed enough, and no quantitative or comparative results are presented in the example demonstration. These shortcomings need to be further refined and strengthened in the future research.

Acknowledgments
This work was supported in part by the Defense Key Laboratories Support Program of China (Grant No. WDZC2019601A303).
References

[1] GJB5236-2004 Military software quality metrics Chinese People’s Liberation Army General Armament Department

[2] Lu XY, Jiang SL, Sun L et al 2013 Research on Risk Assessment Methods of Bank Information System Development Journal of Yangze University (Nat. Sci. Edit) vol 10 no 19 pp 68-70 (in Chinese)

[3] Xu C, He YX, Chen Y et al 2012 Embedded system oriented threat modelling and risk evaluation Application Research of Computers vol 29 no 3 pp 826-828+846 (in Chinese)

[4] Bai WT, Liu ZJ 2004 Requirement analysis and design principle for user interface Journal of Dalian Maritime University vol 30 no 4 pp 86-88 (in Chinese)

[5] Wang XC, Gao Z 2002 Research on Software Requirements Analysis Computer Engineering and Design vol 8 pp 18-21 (in Chinese)

[6] Cheng BHC, Sawyer P, Bencomo N et al 2009 A Goal-Based Modeling Approach to Develop Requirements of an Adaptive System with Environmental Uncertainty Proceedings of the 12th International Conference on Model Driven Engineering Languages and Systems DBLP

[7] Salehie M, Pasquale L, Omoronyia I et al 2012 Requirements-driven adaptive security: Protecting variable assets at runtime 20th IEEE International Requirements Engineering Conference (RE) Chicago pp 111-120

[8] Tsigkanos C, Pasquale L, Ghezzi C et al 2018 On the Interplay Between Cyber and Physical Spaces for Adaptive Security IEEE Transactions on Dependable & Secure Computing vol 15 no 3 pp 466-480