2012 IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops

# Extending the Four-Variable Model for Cyber-Physical Systems

Xiaohong Chen*+ Mingsong Chen*

*Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, 20062, CHINA
+Key Laboratory of High Confidence Software Technologies (Peking University), Ministry of Education, Beijing, 100871, CHINA

Email: xhchen@sei.ecnu.edu.cn Email: mschen@sei.ecnu.edu.cn

***Abstract*—As Cyber-physical System (CPS) is gaining more and more concerns recently, the requirements modeling becomes a critical issue in CPS. In this paper we propose a requirements model for CPS by extending the Four-Variable Model. Through extension, the requirements model deals with complex network environment that CPS are facing, and views timing and position requirements as functional requirements. Besides the framework of this model, we also give a guiding process on how to select corresponding environment variables in this model. An example on one-way traffic light is illustrated to show the effectiveness of our approach.**

***Keywords- requirements model; Cyber-Physical Systems; environment; Four-Variable Model; scenario***

## I. INTRODUCTION

Cyber-Physical System (CPS) is recognized as an important trend of information technology which will deeply affect our daily life [7]. It is widely adopted in various critical areas such as distributed energy systems, transportation systems, healthcare monitoring systems and etc. These areas are often extensively related and of large scale. Therefore requirements modeling for CPS becomes a key challenge during the CPS development.

To model the requirements of CPS, the first step is to find out what the CPS is, what its requirements are and how they differ from the traditional requirements. The CPS is a controllable, dependable, extendable networked physical device system deeply integrated with computation, communication and control capabilities based on the environment awareness [1]. It adds or extends new functions through real-time interactions, monitors and controls physical entities in a safe, dependable, efficient, and real-time way [1]. In fact, CPS is an embedded system. In order to model embedded system requirements, Heiteryer et al. proposed a Software Cost Reduction (SCR) approach [4, 5, 11]. To provide a formal semantics for the SCR notation, a formal requirements model called the Four-Variable Model is introduced. This model represents a software system as a finite-state automaton, which produces externally visible outputs in response to changes in monitored environmental quantities [11]. The Four-Variable Model introduces four kinds of environment variables and a set of mathematical relations on these variables although it does not explain how to get these environment variables.

However, CPS is more than an embedded system in many aspects. In this paper, we focus on the following two aspects:

- It is facing a more complex network environment which is dynamic and open while the environment of embedded systems is closed.
- Timing and position requirements become functional requirements. If these requirements cannot be performed properly in a correct manner, the system to be developed may be of nonsense.

These bring new challenges for the Four-Variable Model. Existing model cannot meet the challenges. To address these two challenges, this paper presents an extend the Four-variable model to specify the CPS requirements. In addition to the four variables in the Four-variable model, we introduce a referred model for representing network environment for the open knowledge used in deciding the controlled variables. Based on the Four-Variable Model, three steps are needed to finish this requirement model: i) to select the corresponding environment variables; ii) to specify the relations between these variables; and iii) to derive requirements from these relations. Due to the limit of space, this paper mainly deals with step i). A process to obtain these environment variables, which is not fulfilled in the Four-Variable Model, is proposed in this paper.

The rest of this paper is organized as follows. Section II reviews the Four-Variable Model. Section III presents the extended model for CPS. Section IV builds the environment variable capture model for the extended model. A typically example is illustrated in Section V to show how to capture environment variables using our method. Finally, section VI concludes the paper.

## II. REVIEW OF THE FOUR-VARIABLE MODEL

The Four-Variable Model [11], illustrated in Figure 1, describes the required system behavior, including the required timing and accuracy, as a set of mathematical relations on four sets of variables—*monitored* and *controlled* variables and *input* and *output* data items. A monitored variable represents an environmental quantity that influences

978-0-7695-4669-8/12 $26.00 © 2012 IEEE
DOI 10.1109/ISORCW.2012.16



IEEE computer society

system behaviors. A controlled variable represents an environmental quantity which the system controls. In the Four-Variable Model, input devices (e.g., sensors) measure the monitored quantities, and output devices set the controlled quantities. Input and output data items represent the values that the devices read and write.

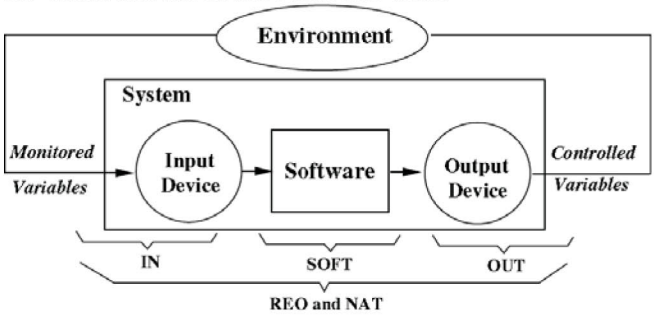


Figure 1. The Four-Variable Model [11]

There are five relations on the four kinds of variables: IN, SOFT, OUT, REQ and NAT. From the monitored quantities to the controlled quantities, REQ and NAT represent a black-box specification of required behaviors. REQ defines additional constraints on the system to be built as relations that the system must maintain between the monitored and the controlled quantities. NAT, which defines the set of possible values, describes the natural constraints on the system behavior, such as constraints imposed by physical laws and the system environment. The relation IN defines the mapping from the monitored quantities to the input data items. The relation OUT defines the mapping from the output data items (e.g., actuators) to the controlled quantities. The use of monitored and controlled quantities to define the required behavior (rather than input and output data items) keeps the specification in the problem domain and allows a simpler specification. The relation SOFT defines software requirements by using a mapping from the input data items to the output data items.

## III. EXTENSION OF FOUR-VARIABLE MODEL FOR CPS

As described in section I, CPS is a system that will interact with its environment quite often. Therefore, our model consists of two parts: the system and its environment. Different from the environment in the original Four Variable Model, the environment for CPS is very complex. It includes the physical world as well as other computational systems that will interact with system.

Within CPS, the physical environment is continuously monitored, and accordingly be controlled based on the knowledge from the other computational system. Thus, the environment of the CPS can be classified into the *monitored environment*, the *referred environment*, and the *controlled environment*. The *monitored environment* is the environment to be monitored, the *referred environment* includes knowledge to be based on, and the *controlled environment* is the environment to be controlled. Among the three environments, the monitored environment may overlap with the controlled environment.

From the monitored environment, we can obtain the monitored variables. Besides the monitored variables, sometimes the relations among several monitored variables are required. For example, the speed of the train will directly be affected by the time and position information. In this situation, the monitored constraints are introduced. Similarly, we get the controlled variables from the controlled environment. Since control involves both time and position information, the controlled constraints are derived from controlled variables as well as time/position data. For the referred environment, it provides knowledge (i.e., speed curve) for computation. Generally, it is represented by functions or tables rather than variables only. We call them referred models. These models will be used by the software.

For the system part in CPS, the software will interact with the physical world via sensors and actuators. The sensors will be used to achieve the input monitored values, and actuators will output the controlled values. In other word, the sensors are the input devices in the Four-Variable Model, and the actuators are the output devices. The sensors give input data items to the software, and the actuators generate output data items from the software. The referred model can be read and written by the software through network. The software interacts with the physical environment by network via sensors and actuators.

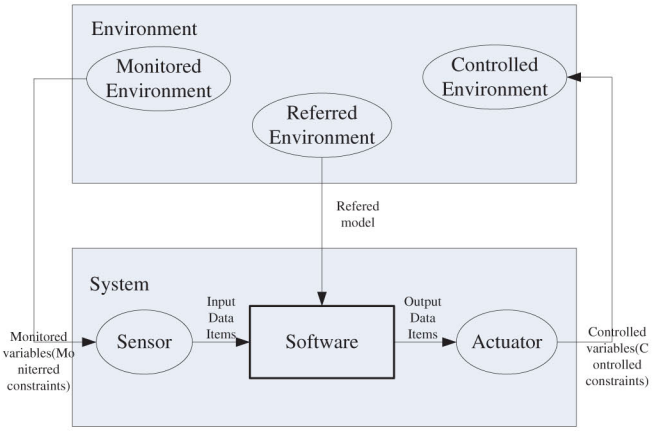


Figure 2. A requirement model for CPS

According to descriptions above, we get a model for CPS as shown in Figure 2. It is extended from the Four-Variable Model. The differences lie in:

- The environment is classified. In addition to the monitored environment and controlled environment, the referred environment is introduced which represents the open world knowledge.
- The information got from the monitored environment includes two parts: the monitored variables and the monitored constraints which allow time and position constraints.

- The information posed on the controlled environment also includes two parts: monitored variables and the controlled constraints which allow constraints for time and position requirements.
- The information obtained from the referred environment is represented by referred model.
- Specific input devices and output devices (i.e., sensors and actuators) are used for CPS.

## IV. ENVIRONMENT VARIABLE CAPTURE MODEL

Section III presents a requirements model framework for CPS. To get a real model from beginning, it is necessary to figure out how to get these environment variables (including referred models) first. Before we introducing a guide process, a few concepts are needed for capturing these environment variables.

### A. *Environment variables*

From environment to environment variables, we introduce a new concept – **environment entity**. In our previous work, the environment of the system is assumed to be composed by a set of environment entities [2, 8]. Each environment entity has attributes and behaviors. Each attribute has values. Attributes can be monitored variables while behaviors can be controlled variables.

It is important to know how to get the environment entities from the requirements description. When capturing requirements, we usually use the scenario-based approaches [9]. As scenarios support ground argument and reasoning in specific detail or examples and focus on reality that forces us to address the "devil in the detail" during requirements specification and validation, scenarios have attracted considerable attentions in Requirements Engineering [9]. In the scenario based requirements approaches, requirements are often embodied in various scenarios. In each scenario, many environment entities are involved. Scenarios are located in environment.

Accordingly, we get the conceptual model of requirements as shown in Figure 3 with the OMT notation [6]. Table 1 gives the meaning of each concept.

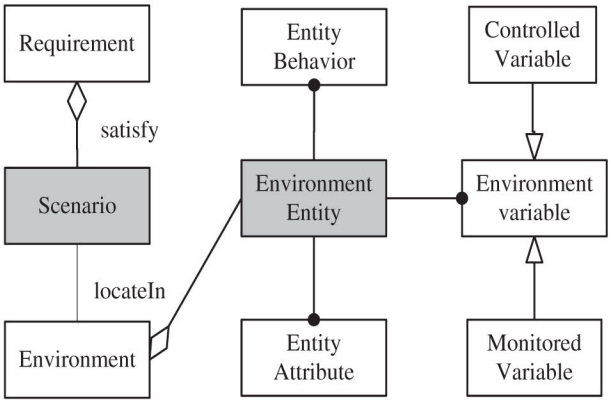


Figure 3. Structure of requirement

TABLE I. CONCEPT MEANING OF REQUIREMENT

| Concept | Meaning |
|---|---|
| Requirement | function of the system |
| Scenario | details of scenes of future events, used to reveal the choices available and their potential consequences |
| Environment | composed of a set of environment entities |
| Environment entity | a relevant real world entity that will interact with the system |
| Attribute | the stable value property of the environment entity |
| Behavior | operations of the environment entity |
| Value | an intangible individual that is not subject to change |
| EnvironmentVariable | variables or models of the environment |
| Monitored Variable | variables to be monitored |
| ControlledVariable | variables to be controlled |

### B. *Referred models*

According to the definition of environment in Section IV.A, the referred environment is also composed of a set of environment entities. In fact, the referred model indicates the relations among attributes of these entities rather than relations among environment entities. These models can be functions of variables, and such functions can be continuous on time and position features. They can also be obtained from scenarios.

## V. ENVIRONMENT VARIABLE CAPTURE PROCESS: AN EXAMPLE

In section IV, an environment variable capture model is given. Based on this model, we define a 5-step process shown in Figure 4 for guiding environment variable capture. The input of this process is the problem statement, and the output is the environment variable description.

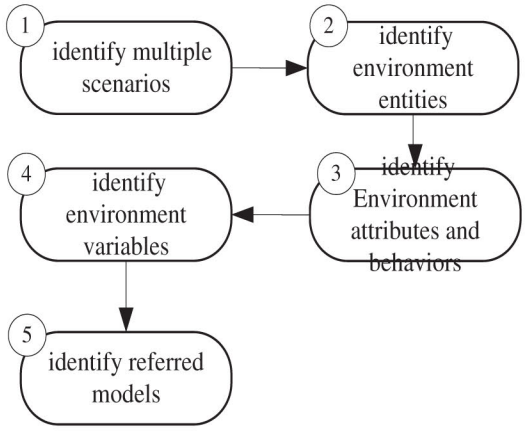


Figure 4. The capture process of the environment variables

In the following, we will use a one-way traffic lights problem to illustrate our approach. The problem statement is as follows [7].

*When a section of road is being repaired, it is often necessary to enforce one-way traffic. Half of the road width is used for traffic and half for the repair work. The traffic is controlled by a pair of simple portable traffic light units. There are two lights: left light and right light. At each side of the road, there are sensors to detect pedestrians and vehicles. The light controller can access all kinds of vehicle parameters though Internet.*

**Step 1: identify scenarios under specific requirement**

Firstly, it needs to specify the requirements. The requirements can be objectives of stakeholders. A piece of requirement may associate to many stakeholders which may conflict with each other. To make it fair, we often use the third party as the main stakeholder. For details, please refer to [10].

The to-be software will be running under different scenarios in order to satisfy this requirement. We propose to elicit scenarios using the goal-scenario coupling approach [10]. In this approach, after requirement discovery, scenario authoring is initiated. Here we use the natural language to express the scenario. The basic syntax would be "When…do…".

**Example 1**

The stakeholders involve government, pedestrians and vehicles. The main actor is the government who is a third party. The requirement of the one-way traffic light problem from the perspective of the government is to *go through the one-way road safely and efficiently*.

There are many scenarios that the traffic would encounter. From the sentence, we firstly distinguish two kinds of situations: *rush time* and *leisure time*. Rush time is the time such as 7:00-9:00 am, 11:00 am-1:00 pm, 5:00-7:00 pm. They are the time when the traffic is very heavy. There are many pedestrians and vehicles on both sides. In contrast, the leisure time is the time when there are not many pedestrians and vehicles on both sides. Importantly, there are emergent vehicles such as ambulances. Certainly, they are of the highest priority. In this situation, green light will be always open for them. The following is a list of classical scenarios.

(sce1) When it is at rush time, there are many pedestrians and vehicles. In this situation, just make the light of each side turn red and green at a particular rate, that is, make them wait for the same time.

(sce2) When it is at leisure time, there are only pedestrians on one side. In this situation, usually make this side pedestrian go.

(sce3) When it is at leisure time, there are few pedestrians on both sides. In this situation, make the less side go first, and don't let both sides waiting for too long. In order to realize this, the opposite lights turn green.

(sce4) When it is at leisure time, there are only vehicles on one side. In this situation, usually make this side vehicle go.

(sce5) When it is at leisure time, there are not many vehicles on both sides. In this situation, make the less side go first, and don't let the other side wait for too long.

(sce6) When it is at leisure time, there is an ambulance from one side. Make the ambulance go first.

(sce7) When it is at leisure time, there are pedestrians and vehicles on both sides. In this situation, make the less side go first, and don't let the other side wait for too long.

(sce8) When a crazy car comes hurtling by, search its speed curve, estimate its coming time, and let it go at appropriate time.

Among all these scenarios, $sce_1$ and $sce_7$ are the common scenarios. The other scenarios are extremes.

**Step 2: identify environment entities**

The environment entities can be achieved from the scenario descriptions. We could use the noun phrase identification method to find them. They can be obtained in 3 sub-steps:

(1) identify the nouns from scenario statement;
(2) organize these words;
(3) distinguish the environment entities.

**Example 2**

Taking $sce_1$ for example, we could find the following nouns :

- *time*
- *many pedestrians*
- *many vehicles*
- *side*
- *light turn red and green*
- *wait time*

In these words, *time* can be an entity. *Pedestrian*, *vehicle*, and *light* can also be *entities*.

Similarly, we could get the other entities as shown in Table II. These entities recognized and the software form a context diagram in the Problem Frames approach[7] notation shown in Figure 5.

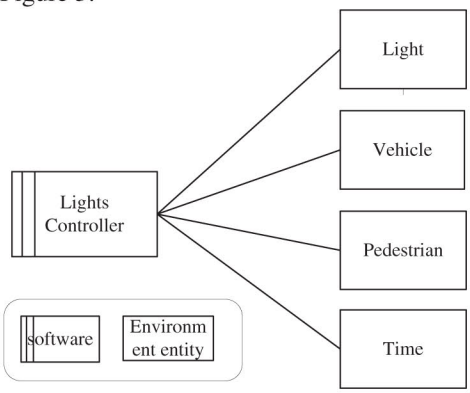


Figure 5. Context diagram of lights controller

TABLE II. ENVIRONMENT ENTITIES, ATTRIBUTES AND BEHAVIORS

| Scenario | Environment entities | Entity attributes/behaviors |
|---|---|---|
| $sce_1$ | Time<br>Pedestrian<br>Vehicle<br>Light | Pedestrian.num<br>Vehicle.num<br>Pedestrian.side<br>Light.waitime<br>Vehicle.waittime<br>Pedestrain.waittime<br>Pedestrian.side<br>Light.turn_red()<br>Light.turn_green() |
| $sce_2$ | Time<br>Pedestrian<br>Light | Pedestrian.side<br>Pedestrian.num<br>Light.side<br>Light.turn_red()<br>Light.turn_green() |
| $sce_3$ | Time<br>Pedestrian<br>Light | Vehicle.num<br>Pedestrian.side<br>Light.side<br>Light.turn_red()<br>Light.turn_green() |
| $sce_4$ | Time<br>Vehicle<br>Light | Vehicle.side<br>Light.side<br>Light.turn_red()<br>Light.turn_green() |
| $sce_5$ | Time<br>Vehicle<br>Light | Vehicle.side<br>Vehicle.num<br>Light.side<br>Light.turn_red()<br>Light.turn_green() |
| $sce_6$ | Time<br>Ambulance<br>Light | Vehicle.type<br>Light.side<br>Light.turn_red()<br>Light.turn_green() |
| $sce_7$ | Time<br>Pedestrian<br>Vehicle<br>Light | Pedestrian.num<br>Vehicle.num<br>Pedestrian.side<br>Light.side<br>Light.turn_red()<br>Light.turn_green() |
| $sce_8$ | Time<br>Vehicle<br>Light | Vehicle.side<br>Light.turn_red()<br>Light.turn_green() |

**Step 3: identify entity attributes and entity behaviors**

The environment entity attributes, and entity behaviors can be achieved from the scenario descriptions. We could use the quantified adjectives and verbs identification method to find them. They can be obtained in 3 sub-steps:

(1) identify the nouns, quantified adjectives and verbs of the environment entities obtained in step 2;
(2) organize these words;
(3) distinguish the environment entity attributes, and behaviors.

**Example 3**

Take $sce_1$ as an example, we could find the nouns, quantified adjectives, and verbs as follows:

- *many pedestrians*
- *many vehicles*
- *side*
- *light turn red and green*
- *wait time*

In these words, `*Many pedestrians*' indicates the number of *pedestrian*. Therefore, *pedestrian.num* is an attribute of *pedestrian*. Similarly, *vehicle.num* is an attribute of *vehicle*. *Pedestrian.side*, *vehicle.side, Pedestrian.waittime, and vehicle.waittime* are also attributes. *Time.time* is an attribute of time. The `*light turn red and green*' leads to two behaviors of light. They are '*light.turn_red()'* and '*light.turn_green()'*.

Similarly, we could get the other entity attributes and behaviors as shown in Table II.

**Step 4: identify environment variables**

Environment variables are recognized by the environment attributes and behaviors. Two different kinds of variables need to be distinguished: monitored variables and controlled variables. This step includes four sub-steps:

(1) find all the environment entities;
(2) for each entity, identify the environment variables from the attributes and behaviors;
(3) find the monitored variables and the value ranges of them;
(4) find the controlled variables, and the value ranges of them.

**Example 4**

From Table II, all the recognized environment entities are: *Time, Pedestrian, Vehicle,* and *Light*. All these environment entities have attributes and behaviors. Among them, the problem is to monitor the *time, pedestrian* and *vehicle*. According to the monitor, the controller which is used to control the light is built. Therefore, the attributes of time, pedestrian and vehicle are potential monitored variables. The attributes or behaviors of light are potential controlled variables.

For example, *Pedestrian* has three attributes, *Pedestrian.num*, *Pedestrian.waittime*, and *Pedestrian.side*. Each attribute plays a great role in determining the control behaviors of light. Therefore, they are all monitored variables. The following will determine their ranges. For instance, *Pedestrian.num* ranges from integer (i.e., *int* for short). All the monitored variables and their ranges are given in Table III.

*Light* is the entity to be controlled. Light's behavior is to be controlled. Therefore, *Light.behavior* is the control variable. It has two values: *turn_red()* and *turn_green()*. Likewise, the light's waiting time can also be controlled. Therefore, *light.waittime* is a control variable. All the controlled variables and their ranges are also given in Table III.

TABLE III. ENVIRONMENT VARIABLES OBTAINED

| Entities | Monitored variable/ranges | | Controlled variable/range | |
|---|---|---|---|---|
| Time | Time.time | [0,24h] | | |
| Pedestrian | Pedestrian.num | int | | |
| | Pedestrian.side | left,right | | |
| | Pedestrian.waittime | [1,300s] | | |
| Vehicle | Vehicle.type | Car,ambulence, trunk,… | | |
| | Vehicle.num | left,right | | |
| | Vehicle.side | left,right | | |
| | Vehicle.waittime | [0,300s] | | |
| Light | Light.side | left,right | Light.Behavior | turn_red(), turn_green() |
| | | | Light.waittime | [0,300s] |

**Step 5: identify referred models**

The referred models are also recognized from the statement of scenarios. We could use the specific phrase identification to find them. These phrases can be "search for…" or "find…" or "read …". The content of searching or finding or reading is a description of referred model.

**Example 5**

Through search, we could find the following sentence in $sce_8$: *When a crazy car comes hurtling by, search its speed curve, estimate its coming time, and let it go at appropriate time.* The phrase "search its speed curve" establishes that the **speed curve** is the referred model. The speed curve can be obtained from Internet by inputting the type of the car. It may be a quadratic equation or a function of exponential. It depends on the type of the car.

## VI. CONCLUSION

Requirements modeling for CPS is becoming an important issue during the CPS development. In this paper, we presented a requirement model for CPS by extending the Four-Variable Model. At the same time, we also provided guidance for obtaining environment variables in the extended model. The contributions of this paper are as follows:

- By extending the Four-Variable Model, a requirements model for CPS is proposed. The requirement model tentatively tries to address the following two problems: one is the modeling of the complex network environment of CPS, and the other is the timing and position requirements for CPS.
- A model for capturing environment variables is presented. This lays solid foundation for guiding environment variable capture.
- A guiding process for obtaining environment variables is given, which is not handled by the Four-Variable Model.

More work needs to be done to complete the requirements model proposed in this paper. (1) Sensors and actuators and their input/out data items are not considered in this paper. (2) The relations among environment variables need specifying. And (3) requirements need to be derived from these relations. Moreover, applying our approach to more concrete real cases is also an important future work.

## ACKNOWLEDGMENT

This work was supported by the National Basic Research and Development 973 Program of China (Grant No.2009CB320702), the National Natural Science Foundation of China (Grant No.90718014, No.61170084), the Key Project of National Natural Science Foundation of China(Grant No.90818026), the National 863 High-tech Project of China (Grant No.2011AA010101), and the Science Fund for Creative Research Groups of the National Natural Science Foundation of China(Grant No.61021004).

## REFERENCES

[1] J.F. He. Cyber Physical Systems. Communication of China Computer Federation, 2010, 6(1):25-29.

[2] Z. Jin, X.H. Chen, and D. Zowghi. Performing Projection in Problem Frames Using Scenarios. Proceedings of 16th Asia-Pacific software Engineering Conference, APSEC 2009. Pp.249-256, 2009.

[3] M. Jackson. Problem Frames. Analyzing and structuring software development problems. Addison-Wesley, 2001.

[4] J. Kirby, M. Archer and C. Heitmeyer. SCR: A Practical Approach to Building a High Assurance COMSEC System, Proceedings of 15th Annual Computer Security Applications Conference (ACSAC '99), Dec 1999.

[5] C. Heitmeyer. Applying 'Practical' Formal Methods to the Specification and Analysis of Security Properties, Proceeding of Information Assurance in Computer Networks (MMMACNS 2001), Russia, pp.21-23 May 2001.

[6] J. Rumbaugh, M. Blaha, W.Premerlani, F.Eddy, and W. Lorensen. Object-Oriented Modeling and Design. Prentice Hall, 1990.

[7] Edward A. Lee. Cyber Physical Systems: Design Challenges. In proc. Of 11th IEEE Symposium on Object Oriented Real-Time Distributed Compputing (ISORC), PP. 363-369, 2008.

[8] X. Chen, Z. Jin, and B. Yin, "An approach for capturing software requirements from interactive scenarios," Chinese Journal of Computers, 34(2):329-341, 2011, (in Chinese).

[9] A. Sutcliffe. Scenario-based requirements engineering. 11th IEEE International Requirements Engineering Conference (RE'03), 2003.

[10] C.Rolland, C.Souveyet, and C.B.Achour, "Guiding goal modeling using scenarios," IEEE Transactions on Software Engineering, 1998, 24( 12):1055–1071.

[11] C.L.Heitmeyer, R.D.Jeffords, and B.G.Labaw. Automated Consistency Checking of Requirements Specification. The ACM Transactions on Software Engineering and Methodology, 1996, 5(3):231-261.