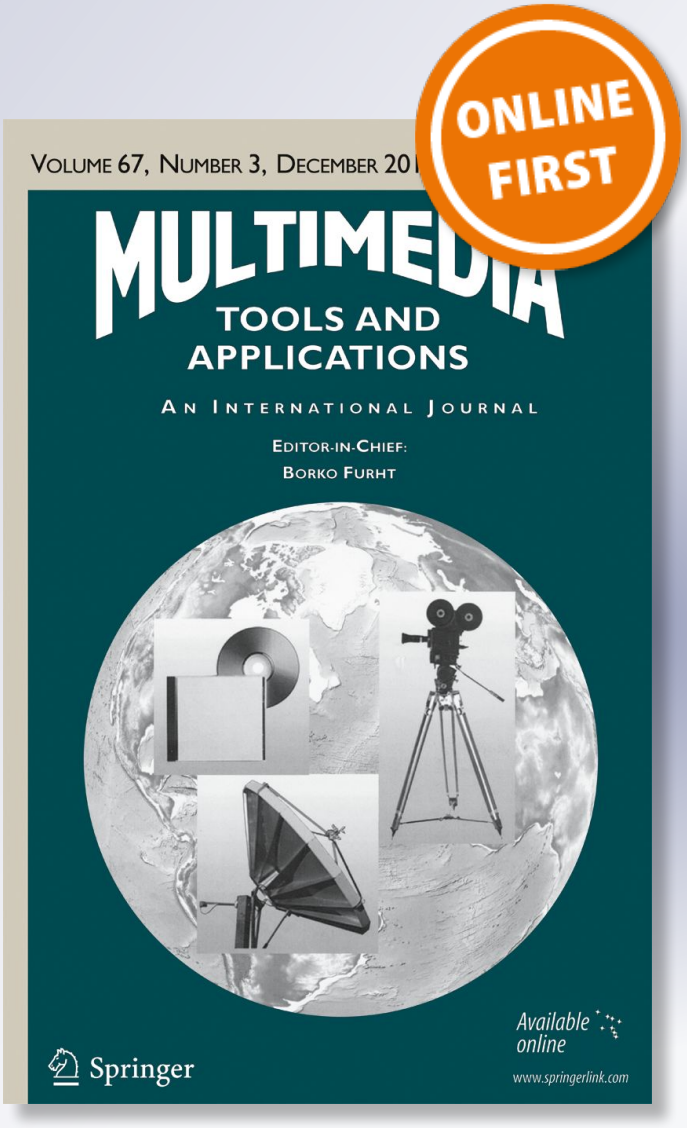
Automatic test case generation for validating the dynamic behaviors of the complete solar power monitoring system

Woo Sung Jang, Byung Kook Jeon, Hyun Seung Son & R. Young Chul Kim

Multimedia Tools and Applications
An International Journal

ISSN 1380-7501

Multimed Tools Appl
DOI 10.1007/s11042-017-5599-4



Springer

Multimed Tools Appl
https://doi.org/10.1007/s11042-017-5599-4

Automatic test case generation for validating the dynamic behaviors of the complete solar power monitoring system

Woo Sung Jang[1] · **Byung Kook Jeon**[2] · **Hyun Seung Son**[1] · **R. Young Chul Kim**[1]

Received: 1 July 2017 / Revised: 12 December 2017 / Accepted: 28 December 2017

Abstract Recently, a photovoltaic energy integrated monitoring system may happen a serious problem with a little error. After constructing the complete solar power monitoring system, this is, a kind of safety-critical system, it should consider how to completely validate it based on requirements of this system. For even code based test coverage in safety critical area, it must achieve 100% of Path and MC/DC coverage on dynamic analysis. To do this, we suggest to automatic test generation on Model Driven Approach to ensure 100% functional requirement coverage using minimal test cases. More important idea is to generate an image (of test case) with an image (of sequence diagram) based on automatic multimedia image translation processing, but not use an algorithmic method in the old fashion. Our approach is focusing on validating the dynamic behaviors of the safety system, which generate test cases based on sequence diagram, that is, use case scenario to design the dynamic behaviors of the solar power monitoring system. As a result, it can validate the safety critical system with minimal test cases.

Keywords Automatic test case generation · Message-sequence diagram · Integrated monitoring system · Automatic multimedia image translation processing

✉ R. Young Chul Kim
bob@selab.hongik.ac.kr

Woo Sung Jang
jang@selab.hongik.ac.kr

Byung Kook Jeon
jeonbk@gwnu.ac.kr

Hyun Seung Son
son@selab.hongik.ac.kr

[1] SE Laboratory, Sejong Campus, Hongik University, Sejong 30016, South Korea

[2] Department of Software, Gangneung-Wonju National University, Wonju 26403, South Korea

Published online: 10 January 2018

1 Introduction

In these days, most of systems are operated through software. Unfortunately, the more software complexity in safe critical fields is increased, the more risks in the systems are occurred. The potential risks of the system are closely associated with errors and vulnerabilities of software. Especially a functional failure in industrial fields such as nuclear power, aviation and railway industries might lead to a serious accident; therefore, it is urgently required to find a way to improve stability and reliability of software. Most global corporations spend more budgets on efforts of improving the stability and reliability with diverse tools. Even they choose software development methodology in order to minimize potential risks.

Approximately 60% of software errors occur in the pre-design stages as well as the design stage, while only 40% of them in the post-design stages [9]. Furthermore, because the requirement specifications have ambiguous, they are not easily detected with test cases. Also, the requirement is misinterpreted, which causes another issue of maintenance of the system. That is, a main reason to cause software errors may be a test case based on incomplete demands [1]. Model based testing with multimedia processing generally decreases the number of incomplete test cases generated by demands. If demand-based test cases are generated in the design stage, the modules being developed based on misinterpreted demands would be identified more quickly. In addition, readability is high, and easy to understand.

The existing methods are model-based test case generations such as State Machines [3, 7], Use Case Diagram [5], etc. In this study, Model based testing is based on Message-Sequence Diagram, which is an effective model to show some message flows of a parallel system [6, 11]. The approach is satisfied with 100% of test coverage, which generates test case through transforming the case effect diagram based on extended message sequence diagram [12]. This paper represents how to generate test case with decision table via cause-effect diagram based on our extended message sequence diagram for validating the safety against potential errors of the solar power monitoring system. Our approach has more important idea in this paper, which is to use an image (of sequence diagram) for generating an image (of test case) based on automatic multimedia image translation processing [2], but not use an algorithmic method in the old fashion.

This paper is introduced below. Chapter 2 describes related studies. Chapter 3 shows a safety design of a solar power monitoring system. Chapter 4 describes automatically to generate test cases for validating the dynamic behaviors of the solar power monitoring system, followed by conclusion.

2 Related work

2.1 Model-based testing

Model-based testing [1] would turn incomplete demands into adequate test cases. Figure 1 illustrates that a test case is generated from the test models based on demand documents. Test

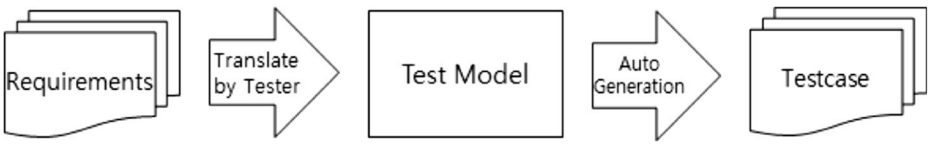


Fig. 1 Model-based testing process

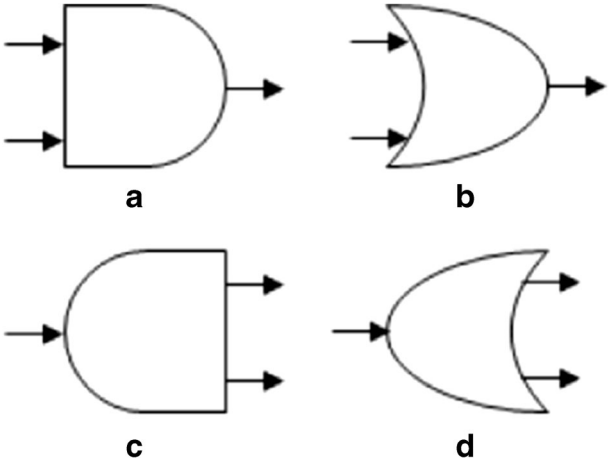


Fig. 2 AND/OR gates in extended message-sequence diagram

models have one standard notation such as JML; therefore actions of the system could accurately be expressed. Also, a verification tool is used on the standard notation which leads to greater accuracy and consistency of the test model.

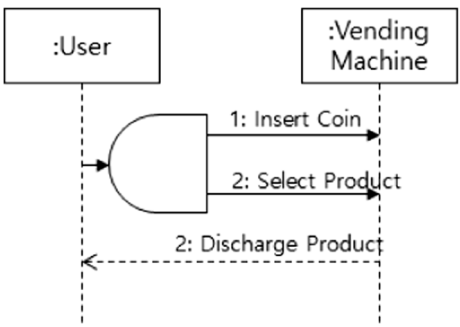
The existing ways manually design a demand test case. Test model, however, is supposed to design a test case, which be able to generate more test cases compared to the existing ones.

Furthermore, only designing with a model-based approach would promote a quality of the test case in the design stage. We can improve the quality of test cases based on model based testing.

2.2 Extended message-sequence diagram

We extended message sequence diagram with notations of AND/OR gates for message communication between objects [11]. Figure 2a shows to come in the number of in-messages >2 at the same time, and out a single out-message on an object. Figure 2b shows to come in at least one in-message, and out a single out-message on an object. Figure 2c shows to come in a single in-message, and out the number of out-messages >2 at the same time on an object. Figure 2d shows to come in one in-message, and out more than one out-message on an object.

Fig. 3 AND gate usage exmaple



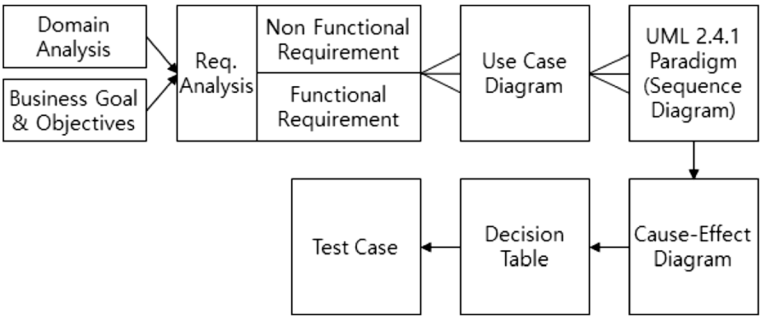


Fig. 4 Automatic test case design mechanism

Figure 3 shows an example of extended diagram with AND gate. Simultaneously need to send two out-messages to other object(s) with the diagram.

2.3 Test case extraction

Gary suggests a method of designing a test case using cause-effect diagram. In this way, minimal test cases would be generated that assures 100% of functional test coverage [8]. Our approach focuses on UML 2.1.4 message-sequence diagram, which is transformed to cause-effect diagram. We also convert the cause-effect diagram into decision table, and then generates test cases with this table. These test cases are satisfied with 100% of functional test coverage based on Gary approach. Figure 4 shows automatic test case design mechanism. This mechanism starts that Message-Sequence Diagram is designed by using each use case in the Use Case Diagram; Message-Sequence Diagram is transformed to Cause-Effect Diagram; Cause-Effect Diagram is transformed to decision table; and decision table turns into a test case.

Such translation process is automated through the model translation technique. Figure 5 represents the automated model translation process, which uses the model translation technique for designing each meta model in each stage; Model translation rules are defined by applying ATL (Atlas Translation Language) to the meta model designed; and Message-Sequence Diagram is automatically transformed to a test case using the model translation rules.

2.4 New & renewable energy integrated monitoring system

New & renewable energy total monitoring system needs to have a standard interface, which interprets different types of data to be delivered from various kinds of new &

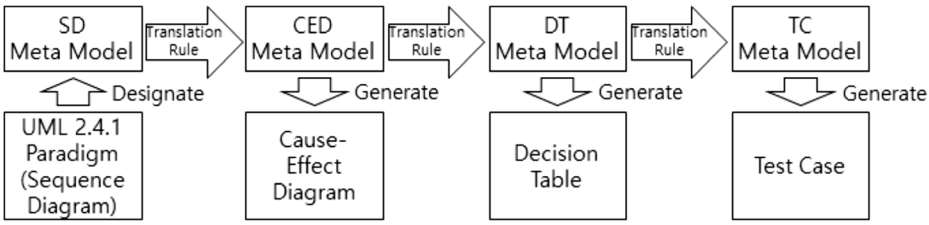


Fig. 5 Model translation process for automatic test case generation

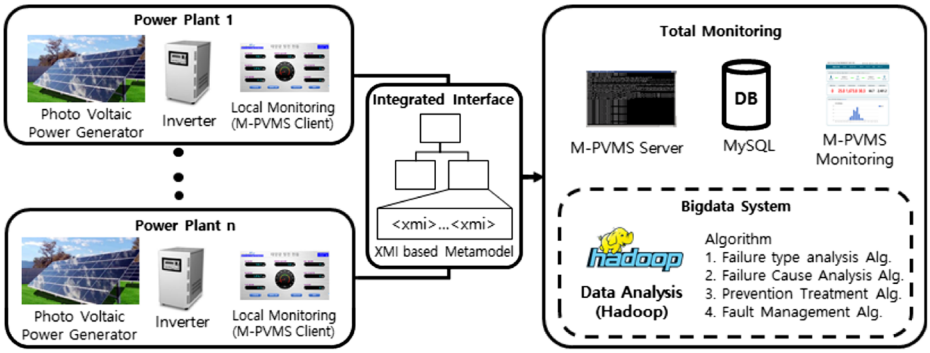


Fig. 6 The solar nergy integrated monitoring system

renewable energy plants. Because the standard interface should be designed based on meta model, a new data type needs to be easily added into it, that is, plug-play mechanism. Moreover, when it should provide total monitoring services based on the web server, users can easily track the current power via each web browser. It also needs to provide some related predictions of the power with statistical or machine learning methods on a big data [4].

Figure 6 illustrates the detail structure of the solar power monitoring syste. M-PVMS client is installed in each plant. From the inverter, M-PVMS client transforms the data to M-PVMS server. The data is meta model-based XMI files which be delivered to M-PVMS server. M-PVMS server then interprets the SMI, and stores on the database. The web server-based M-PVMS monitoring records a data as the form of web page into database [10].

Figure 7 shows an implemented M-PVMS Server. It shows to display a data and some alarms delivered from M-PVMS client.

Figure 8 shows an implemented M-PVMS client. It shows to display the power output of each plant on Java based multimedia UI.

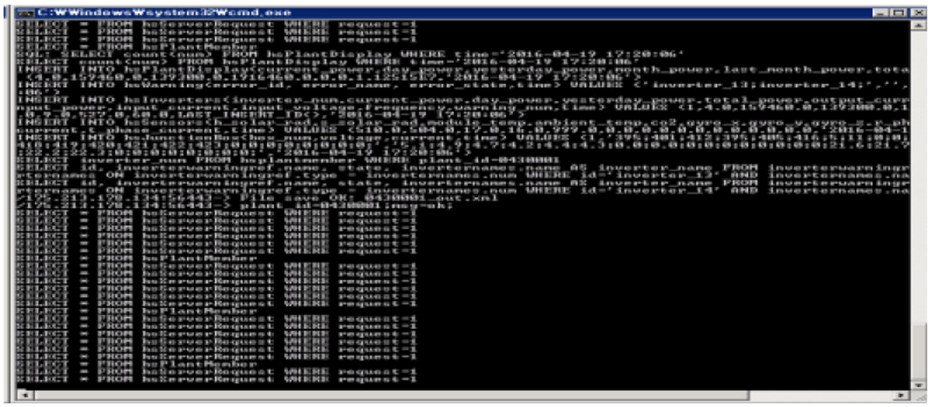


Fig. 7 The implemented M-PVMS Server

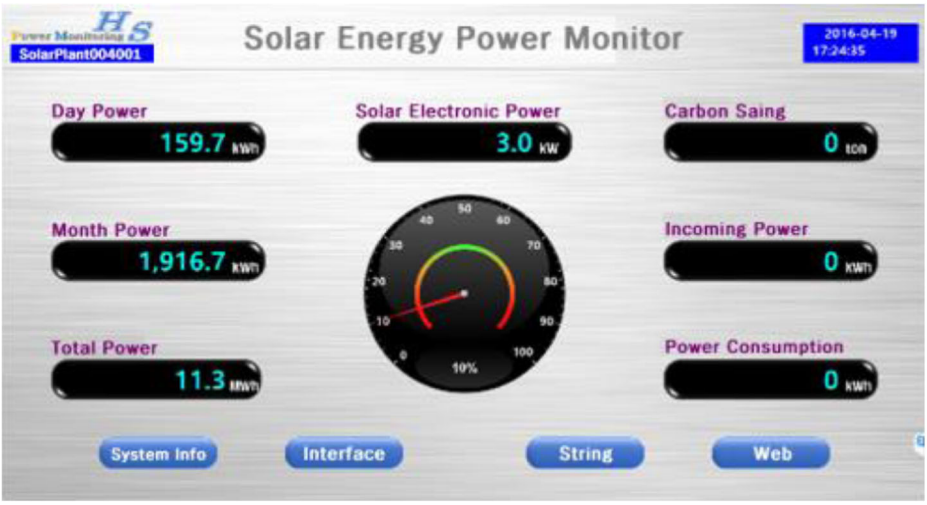


Fig. 8 The implemented M-PVMS client

Figure 9 shows an implemented M-PVMS Monitor. It shows to display a number and graph of power output transfered from each plant.

Fig. 9 The implemented M-PVMS monitor

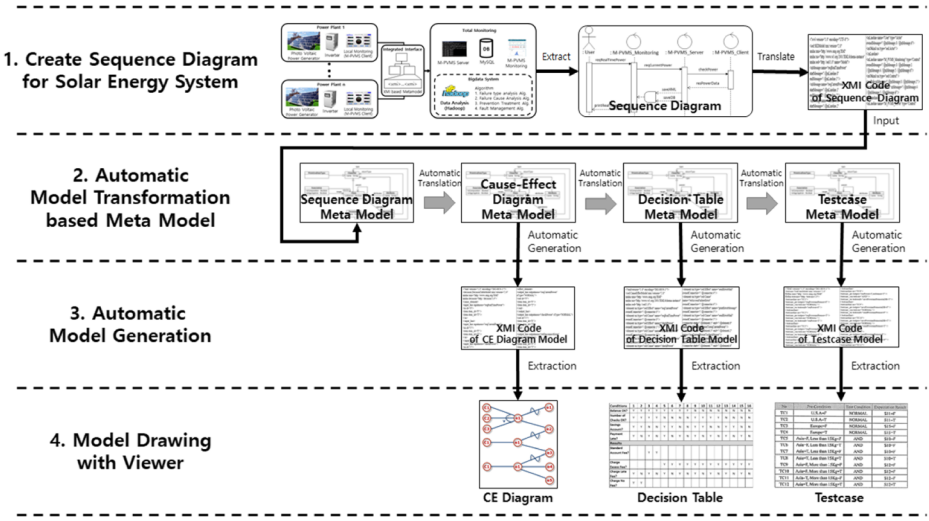


Fig. 10 Multimedia translation process for generating test case from decision table via cause-effect diagram from sequence diagram

3 The dynamic behavioral modeling of a solar power monitoring system using multimedia processing

This approach generates automatically test case from one image of Sequence Diagram using automatically multimedia translation processing. So developers just design the dynamic behaviors of the system, then can automatically get test cases.

In Fig. 10, we develop each meta model per each diagram, and make each transformation rules to covert one diagram to other diagram. We show how to generate test case with decision table via cause-effect diagram based on our extended message sequence diagram for validating the safety against potential errors of the solar power monitoring system. That is, first converts sequence diagram to XMI code,

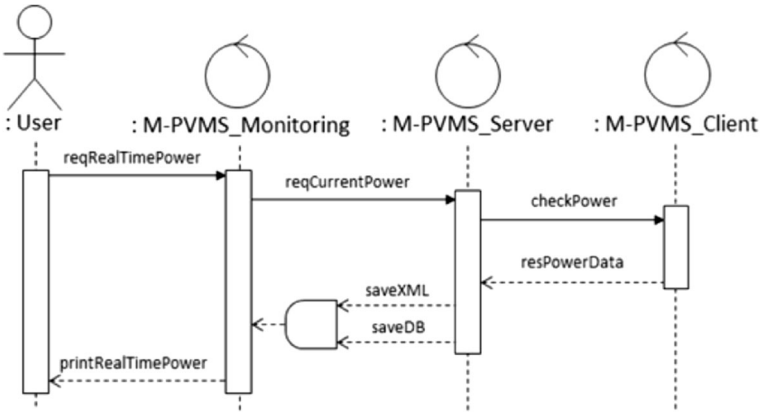


Fig. 11 The flow of real time power generation at the nomal connection

Table 1 Messages in message sequnce diagram for real time power generation at normal connection

Message	Description
reqRealTimePower	Request real-time power
reqCurrentPower	Request current power
checkPower	Check power
resPowerData	Response power/sensor data
saveXML	Save data in XML file
saveDB	Save data in database
printRealTimePower	Output real-time power

and input XMI code of sequence diagram into meta model translation engine. The translation engine automatically translated XMI code into a meta model of the sequence diagram. The meta model of sequence diagram is automatically translated into meta model of cause-effect diagram. Meta model of the cause-effect diagram is automatically translated into meta model of the decision table. Meta model of the decision table is automatically translated into a meta model of testcase. XMI code is automatically generated from the each generated metamodel. The graph viewer draws the automatically generated XMI code into a graph. Figure 10 shows the transformation process for generating test case from Decision Table via Cause-Effect Diagram from Sequence Diagram.

We are carefully considering of modeling the dynamic behaviors to enhance the safety of this system, which extracts test cases via one image (modeling the dynamic behaviors), that is, *extended message sequence diagram* based on potential errors/problems/malfunction.

This paper describes some cases of '*Real time Display of Power Output*', '*Backup of Solar Power Genration*' in potential errors/problems/malfunction of the system.

3.1 Real time display of power output

The real time display of power output is a function of checking each power generation for a user to watch M-PVMS monitor. When a user asks real time power generation of a particular plant, M-PVMS Monitoring asks current power generation of the plant to M-PVMS server. The M-PVMS server asks current power generation to M-PVMS client in which plant a user

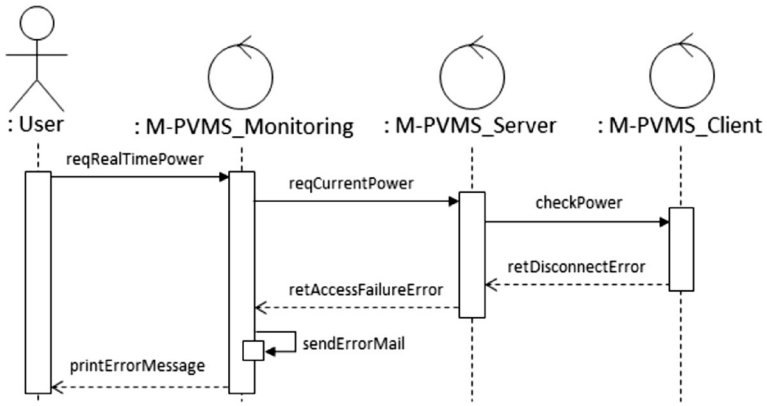


Fig. 12 The flow of real time power generation at disconnection of M-PVMS client

Table 2 Messages in message sequnce diagram for real time power generation at disconnection

Message	Description
reqRealTimePower	Request real-time power
reqCurrentPower	Request current power
checkPower	Check power
retDisconnectError	Client connect failed
retAccessFailureError	Return connect failure error
sendErrorMail	Send error message to administrator
printErrorMessage	Output error message

selects. The M-PVMS client sends the current volume of power generation to the M-PVMS server. The M-PVMS server stores the current volume as XMI file into Database. M-PVMS monitor shows real time power volume to the user by reading XMI file. Figure 11 shows how to display general power generation. In Table 1, we describe all meanings of each messages in the extended message sequence diagram in Fig. 11.

Figure 12 shows the flow of real time power generation at disconnection of M-PVMS client. When disconnect with M-PVMS client, automatically send a message to administrator. Then display Disconnection Error on M-PVMS monitor.

Table 2 describe all meanings of each message in the extended message sequence diagram for *'real time power generation at disconnection'*.

3.2 Backup of solar power generation

The backup of power generation is a store function of transmitted power generation at M-PVMS server. M-PVMS client refers the power generation of the inverter, and sends this data to M-PVSM server. Figure 13 shows the general backup flow of power generation. When the M-PVMS client asks the current power generation and sensor's data to the inverter, the inverter sends them to the server. After sending the data, M-PVMS client saves this data into the local file database of a plant. As soon as the server receives this data, he sends a complete message

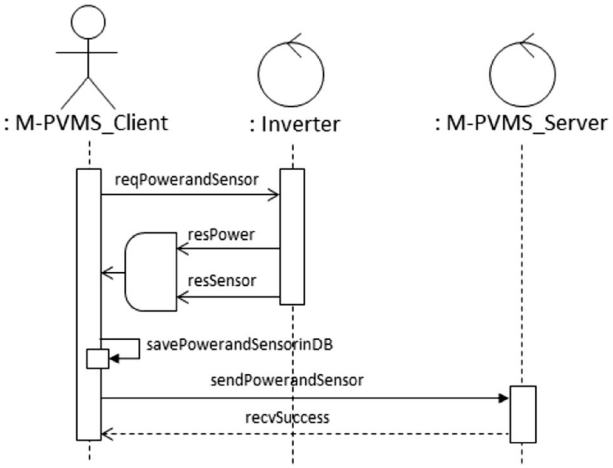


Fig. 13 The general backup of power generation

Table 3 Messages in message sequnce diagram for backup of real time power generation

Message	Description
reqPowerandSensor	Request power/sensor data
resPower	Response power data
resSensor	Reponse sensor data
savePowerandSensorinDB	Save power/sensor data in database
sendPowerandSensor	Send power/sensor data
recvSuccess	Received power data

to the client. Table 3 describes Messages in Message sequnce diagram for *'backup of real time power generation'*.

Figure 14 shows the backup flow of real time power generation at disconnection of M-PVMS server. When disconnect with M-PVMS server, it shows how to save real time power generation. Until being re-connected with M-PVMS server, the client saves all data into the local database. After re-connecting the server, the client sends the client all data which not transmitted. Table 4 describes messages of message sequence diagram at disconnection with M-PVMS server.

Figure 15 shows the backup flow of real time power generation at occurring an error on the inverter. When occurring an error on the inverter, it sends an error message to M-PVMS client. M-PVMS client transmits an error code to M-PVMS server which sends the transmitted error code to administrator. Table 5 describes Messages of Message sequence diagram for occuring an error on the inverter.

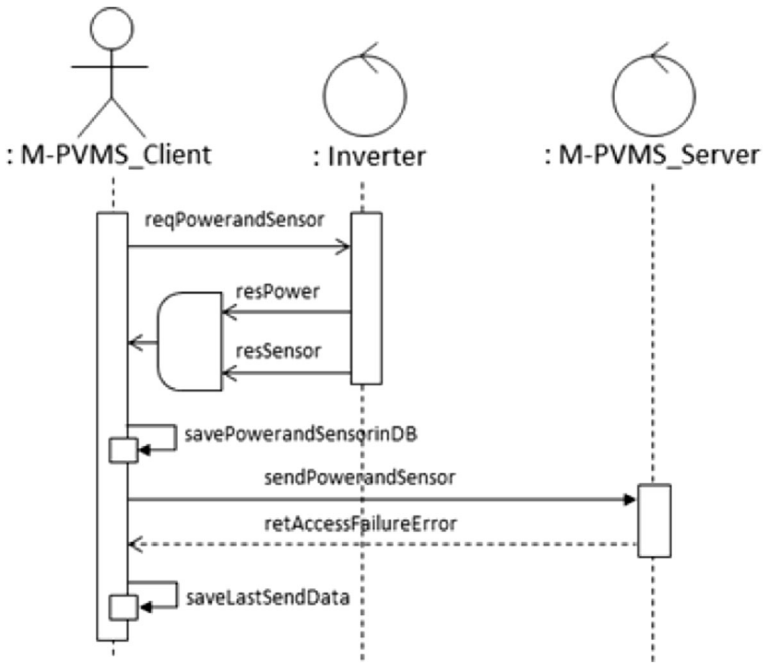


Fig. 14 The backup flow of power generation at disconnection of M-PVMS server

Table 4 Messages of message sequence diagram for power generation at disconnection of M-PVMS server

Message	Description
reqPowerandSensor	Request power/sensor data
resPower	Response power data
resSensor	Reponse sensor data
savePowerandSensorinDB	Save power/sensor data in database
sendPowerandSensor	Send power/sensor data
retAccessFailureError	Server connect failed
saveLastSendData	Save last sent data(to send again when server is connected)

4 Test case generation for the solar power monitoring system

To validating the dynamic behaviors of the system, we propose to generate test cases based on extended message sequence diagram. Message sequence diagram is transformed into XMI code. Our proposed meta model tool are implemented with ATL based model transformation rule [12]. This tool can extract test cases with this XMI code.

4.1 XMI code transformation with extended sequence diagram

First, Message sequence diagram should be transformed into XMI code. Table 6 shows to transform XMI code of message sequence diagram for real time power generation.

At disconnection with M-PVMS client, it shows to transform XMI code of message sequence diagram for real time power generation. Table 7 shows XMI code transformation of Message Sequence Diagram for real time power generation at disconnectoin with M-PVMS client.

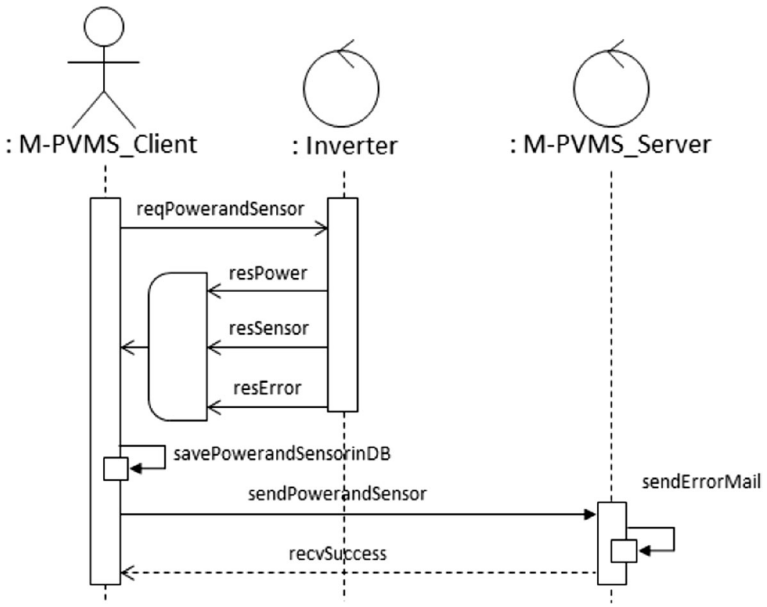


Fig. 15 The flow of message sequence diagram for an error on the inverter

Table 5 Messages of message sequence diagram for occuring an error on the inverter

Message	Description
reqPowerandSensor	Request power/sensor data
resPower	Response power data
resSensor	Reponse sensor data
resError	Response inverter error
savePowerandSensorinDB	Save power/sensor data in database
sendPowerandSensor	Send power/sensor data
sendErrorMail	Send error message to administrator
recvSuccess	Received power data

Table 8 shows to transform XMI code of message sequence diagram for general back up of real time power generation.

At disconnection with M-PVMS server, it shows to transform XMI code of message sequence diagram for real time power generation. Table 9 shows XMI code transformation of message sequence diagram for real time power generation at disconnection with M-PVMS server.

Table 6 XMI code transformation of message sequence diagram for real time power generation

```
<?xml version = "1.0" encoding = "UTF-8"?>
<sed:SEDModel xmi:version = "2.0" xmlns:xmi = "
   http://www.omg.org/XMI"
   xmlns:xsi = "
   http://www.w3.org/2001/XMLSchema-instance"
   xmlns:sed = "http://sed/1.0″ name = "Model">
<mMessage name = "reqRealTimePower"
   startMessage = "//@mLineline.0"
   endMessage = "//@mLineline.1"/>
<mMessage name = "reqCurrentPower"
   startMessage = "//@mLineline.1"
   endMessage = "//@mLineline.2"/>
<mMessage name = "checkPower"
   startMessage = "//@mLineline.2"
   endMessage = "//@mLineline.3"/>
<mMessage name = "resPowerData"
   startMessage = "//@mLineline.3"
   endMessage = "//@mLineline.2"/>
<mMessage name = "saveXMI"
   startMessage = "//@mLineline.2"
   endMessage = "//@mLineline.1"/>
<mMessage name = "saveDB"
   startMessage = "//@mLineline.2"
   endMessage = "//@mLineline.1"/>
<mMessage name = "printRealTimePower"
   startMessage = "//@mLineline.1"
   endMessage = "//@mLineline.0"/>
<mLineline name = "User" type = "Actor"
   ownedMessage = "//@mMessage.0
   //@mMessage.6">
<mObkind xsi:type = "sed:Actor"/>
</mLineline>
<mLineline name = "M_PVMS_Monitoring" type = "
   Control" ownedMessage = "//@mMessage.0
   //@mMessage.1 //@mMessage.4 //@mMessage.5
   //@mMessage.6">
<mObkind xsi:type = "sed:Control"/>
<ecaRule mMessage = "//@mMessage.0
   //@mMessage.1"/>
<ecaRule ecaRule = "AND"
   mMessage = "//@mMessage.4 //@mMessage.5
   //@mMessage.6"/>
</mLineline>
<mLineline name = "M_PVMS_Server" type = "
   Control" ownedMessage = "//@mMessage.1
   //@mMessage.2 //@mMessage.3 //@mMessage.4
   //@mMessage.5">
<mObkind xsi:type = "sed:Service"/>
<ecaRule mMessage = "//@mMessage.1
   //@mMessage.2"/>
<ecaRule mMessage = "//@mMessage.3
   //@mMessage.4"/>
<ecaRule mMessage = "//@mMessage.3
   //@mMessage.5"/>
</mLineline>
<mLineline name = "M_PVMS_Client" type = "
   Control" ownedMessage = "//@mMessage.2
   //@mMessage.3">
<mObkind xsi:type = "sed:Control"/>
<ecaRule mMessage = "//@mMessage.2
   //@mMessage.3"/>
</mLineline>
</sed:SEDModel>
```

Table 7 XMI code transformation of message sequence diagram for real time power generation at disconnectoin with M-PVMS client

```
<?xml version = "1.0" encoding = "UTF-8"?>
<sed:SEDModel xmi:version = "2.0"
    xmlns:xmi = "http://www.omg.org/XMI"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xmlns:sed = "http://sed/1.0" name = "Model">
<mMessage name = "reqRealTimePower"
    startMessage = "//@mLineline.0"
    endMessage = "//@mLineline.1"/>
<mMessage name = "reqCurrentPower"
    startMessage = "//@mLineline.1"
    endMessage = "//@mLineline.2"/>
<mMessage name = "checkPower"
    startMessage = "//@mLineline.2"
    endMessage = "//@mLineline.3"/>
<mMessage name = "retDisconnectError"
    startMessage = "//@mLineline.3"
    endMessage = "//@mLineline.2"/>
<mMessage name = "retAccessFailureError"
    startMessage = "//@mLineline.2"
    endMessage = "//@mLineline.1"/>
<mMessage name = "sendErrorMail"
    startMessage = "//@mLineline.1"
    endMessage = "//@mLineline.1"/>
<mMessage name = "printErrorMessage"
    startMessage = "//@mLineline.1"
    endMessage = "//@mLineline.0"/>
<mLineline name = "User" type = "Actor"
    ownedMessage = "//@mMessage.0 //@mMessage.6">
<mObkind xsi:type = "sed:Actor"/>
</mLineline>
<mLineline name = "M_PVMS_Monitoring" type = "Control"
    ownedMessage = "//@mMessage.0 //@mMessage.1
    //@mMessage.4 //@mMessage.5 //@mMessage.6">
<mObkind xsi:type = "sed:Control"/>
<ecaRule mMessage = "//@mMessage.0 //@mMessage.1"/>
<ecaRule mMessage = "//@mMessage.4 //@mMessage.5"/>
<ecaRule mMessage = "//@mMessage.5 //@mMessage.6"/>
</mLineline>
<mLineline name = "M_PVMS_Server" type = "Control"
    ownedMessage = "//@mMessage.1 //@mMessage.2
    //@mMessage.3 //@mMessage.4">
<mObkind xsi:type = "sed:Service"/>
<ecaRule mMessage = "//@mMessage.1 //@mMessage.2"/>
<ecaRule mMessage = "//@mMessage.3 //@mMessage.4"/>
</mLineline>
<mLineline name = "M_PVMS_Client" type = "Control"
    ownedMessage = "//@mMessage.2 //@mMessage.3">
<mObkind xsi:type = "sed:Control"/>
<ecaRule mMessage = "//@mMessage.2 //@mMessage.3"/>
</mLineline>
</sed:SEDModel>
```

Table 8 XMI code transformation of message sequence diagram for backup of real time power generation

```
<?xml version = "1.0" encoding = "UTF-8"?>
<sed:SEDModel xmi:version = "2.0"
    xmlns:xmi = "http://www.omg.org/XMI"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xmlns:sed = "http://sed/1.0" name = "Model">
<mMessage name = "reqPowerandSensor"
    startMessage = "//@mLineline.0"
    endMessage = "//@mLineline.1"/>
<mMessage name = "resPower"
    startMessage = "//@mLineline.1"
    endMessage = "//@mLineline.0"/>
<mMessage name = "resSensor"
    startMessage = "//@mLineline.1"
    endMessage = "//@mLineline.0"/>
<mMessage name = "savePowerandSensorinDB"
    startMessage = "//@mLineline.0"
    endMessage = "//@mLineline.0"/>
<mMessage name = "sendPowerandSensor"
    startMessage = "//@mLineline.0"
    endMessage = "//@mLineline.2"/>
<mMessage name = "recvSuccess"
    startMessage = "//@mLineline.2"
    endMessage = "//@mLineline.0"/>
<mLineline name = "M_PVMS_Client" type = "Actor"
    ownedMessage = "//@mMessage.0 //@mMessage.1
    //@mMessage.2 //@mMessage.3 //@mMessage.4
    //@mMessage.5">
<mObkind xsi:type = "sed:Actor"/>
<ecaRule ecaRule = "AND"
    mMessage = "//@mMessage.1 //@mMessage.2
    //@mMessage.3"/>
<ecaRule mMessage = "//@mMessage.3
    //@mMessage.4"/>
</mLineline>
<mLineline name = "Inverter" type = "Control"
    ownedMessage = "//@mMessage.0
    //@mMessage.1 //@mMessage.2">
<mObkind xsi:type = "sed:Control"/>
<ecaRule mMessage = "//@mMessage.0
    //@mMessage.1"/>
<ecaRule mMessage = "//@mMessage.0
    //@mMessage.2"/>
</mLineline>
<mLineline name = "M_PVMS_Server"
    type = "Control"
    ownedMessage = "//@mMessage.4
    //@mMessage.5">
<mObkind xsi:type = "sed:Service"/>
<ecaRule mMessage = "//@mMessage.4
    //@mMessage.5"/>
</mLineline>
</sed:SEDModel>
```

Table 9 XMI code transformation of message sequence diagram for real time power generation at disconnectoin with M-PVMS server

```
<?xml version = "1.0" encoding = "UTF-8"?>
<sed:SEDModel xmi:version = "2.0"
    xmlns:xmi = "http://www.omg.org/XMI"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xmlns:sed = "http://sed/1.0″ name = "Model">
<mMessage name = "reqPowerandSensor"
    startMessage = "//@mLineline.0"
    endMessage = "//@mLineline.1"/>
<mMessage name = "resPower"
    startMessage = "//@mLineline.1"
    endMessage = "//@mLineline.0"/>
<mMessage name = "resSensor"
    startMessage = "//@mLineline.1"
    endMessage = "//@mLineline.0"/>
<mMessage name = "savePowerandSensorinDB"
    startMessage = "//@mLineline.0"
    endMessage = "//@mLineline.0"/>
<mMessage name = "sendPowerandSensor"
    startMessage = "//@mLineline.0"
    endMessage = "//@mLineline.2"/>
<mMessage name = "recvAccessFailureError"
    startMessage = "//@mLineline.2"
    endMessage = "//@mLineline.0"/>
<mMessage name = "saveLastSendData"
    startMessage = "//@mLineline.0"
endMessage = "//@mLineline.0"/>
<mLineline name = "M_PVMS_Client" type = "Actor"
    ownedMessage = "//@mMessage.0 //@mMessage.1
    //@mMessage.2 //@mMessage.3 //@mMessage.4
    //@mMessage.5 //@mMessage.6″>
<mObkind xsi:type = "sed:Actor"/>
<ecaRule ecaRule = "AND" mMessage = "//@mMessage.1
    //@mMessage.2 //@mMessage.3"/>
<ecaRule mMessage = "//@mMessage.3 //@mMessage.4"/>
<ecaRule mMessage = "//@mMessage.5 //@mMessage.6"/>
</mLineline>
<mLineline name = "Inverter" type = "Control"
    ownedMessage = "//@mMessage.0 //@mMessage.1
    //@mMessage.2">
<mObkind xsi:type = "sed:Control"/>
<ecaRule mMessage = "//@mMessage.0 //@mMessage.1"/>
<ecaRule mMessage = "//@mMessage.0 //@mMessage.2"/>
</mLineline>
<mLineline name = "M_PVMS_Server" type = "Control"
    ownedMessage = "//@mMessage.4 //@mMessage.5">
<mObkind xsi:type = "sed:Service"/>
<ecaRule mMessage = "//@mMessage.4 //@mMessage.5"/>
</mLineline>
</sed:SEDModel>
```

Table 10 XMI code transformation of message sequence diagram at an error on the inverter

```
<?xml version = "1.0" encoding = "UTF-8"?>
<sed:SEDModel xmi:version = "2.0"
    xmlns:xmi = "http://www.omg.org/XMI"
    xmlns:xsi = "
    http://www.w3.org/2001/XMLSchema-instance"
    xmlns:sed = "http://sed/1.0″ name = "Model">
<mMessage name = "reqPowerandSensor"
    startMessage = "//@mLineline.0"
    endMessage = "//@mLineline.1"/>
<mMessage name = "resPower"
    startMessage = "//@mLineline.1"
    endMessage = "//@mLineline.0"/>
<mMessage name = "resSensor"
    startMessage = "//@mLineline.1"
    endMessage = "//@mLineline.0"/>
<mMessage name = "resError" startMessage = "//@mLineline.1"
    endMessage = "//@mLineline.0"/>
<mMessage name = "savePowerandSensorinDB"
    startMessage = "//@mLineline.0"
    endMessage = "//@mLineline.0"/>
<mMessage name = "sendPowerandSensor"
    startMessage = "//@mLineline.0"
    endMessage = "//@mLineline.2"/>
<mMessage name = "sendErrorMail"
    startMessage = "//@mLineline.2"
    endMessage = "//@mLineline.2"/>
<mMessage name = "recvSuccess"
    startMessage = "//@mLineline.2"
    endMessage = "//@mLineline.0"/>
<mLineline name = "M_PVMS_Client" type = "Actor"
ownedMessage = "//@mMessage.0 //@mMessage.1
    //@mMessage.2 //@mMessage.3 //@mMessage.4
    //@mMessage.5 //@mMessage.7″>
<mObkind xsi:type = "sed:Actor"/>
<ecaRule ecaRule = "AND" mMessage = "//@mMessage.1
    //@mMessage.2 //@mMessage.3 //@mMessage.4"/>
<ecaRule mMessage = "//@mMessage.4
    //@mMessage.5"/>
</mLineline>
<mLineline name = "Inverter" type = "Control"
    ownedMessage = "//@mMessage.0 //@mMessage.1
    //@mMessage.2">
<mObkind xsi:type = "sed:Control"/>
<ecaRule mMessage = "//@mMessage.0
    //@mMessage.1"/>
<ecaRule mMessage = "//@mMessage.0
    //@mMessage.2"/>
<ecaRule mMessage = "//@mMessage.0
    //@mMessage.3"/>
</mLineline>
<mLineline name = "M_PVMS_Server" type = "Control"
    ownedMessage = "//@mMessage.4 //@mMessage.5">
<mObkind xsi:type = "sed:Service"/>
<ecaRule mMessage = "//@mMessage.5
    //@mMessage.6"/>
<ecaRule mMessage = "//@mMessage.6
    //@mMessage.7"/>
</mLineline>
</sed:SEDModel>
```

Table 11 XMI code transformation of cause-effect graph for real time power generation

```
<?xml version = »1.0» encoding = »ISO-8859-1»?>
<ced:CauseEffectModel xmi:version = »2.0»
    xmlns:xmi = »http://www.omg.org/XMI»
    xmlns:xsi = »http://www.w3.org/2001/XMLSchema-instance»
    xmlns:ced = »http://ced/1.0»>
<element xsi:type = »ced:Effect» name = »reqCurrentPower»
    ownedConnector = »//@connector.0»/>
<element xsi:type = »ced:Cause» name = »reqRealTimePower»
    ownedConnector = »//@connector.0»/>
<element xsi:type = »ced:Effect» name = »checkPower»
    ownedConnector = »//@connector.1»/>
<element xsi:type = »ced:Cause» name = »reqCurrentPower»
    ownedConnector = »//@connector.1»/>
<element xsi:type = »ced:Effect» name = »resPowerData»
    ownedConnector = »//@connector.2»/>
<element xsi:type = »ced:Cause» name = »checkPower»
    ownedConnector = »//@connector.2»/>
<element xsi:type = »ced:Effect» name = »saveXMI»
    ownedConnector = »//@connector.3»/>
<element xsi:type = »ced:Cause» name = »resPowerData»
    ownedConnector = »//@connector.3»/>
<element xsi:type = »ced:Effect» name = »saveDB»
    ownedConnector = »//@connector.4»/>
<element xsi:type = »ced:Cause» name = »resPowerData»
    ownedConnector = »//@connector.4»/>
<element xsi:type = "ced:Effect" name = "printRealTimePower"
    ownedConnector = "//@connector.5 //@connector.6"
    eType = "AND"/>
<element xsi:type = "ced:Cause" name = "saveXMI"
    ownedConnector = "//@connector.5"/>
<element xsi:type = "ced:Cause" name = "saveDB"
    ownedConnector = "//@connector.6"/>
<connector start = "//@element.1" end = "//@element.0"
    name = "reqRealTimePowerToreqCurrentPower"/>
<connector start = "//@element.3" end = "//@element.2"
    name = "reqCurrentPowerTocheckPower"/>
<connector start = "//@element.5" end = "//@element.4"
    name = "checkPowerToresPowerData"/>
<connector start = "//@element.7" end = "//@element.6"
    name = "resPowerDataTosaveXMI"/>
<connector start = "//@element.9" end = "//@element.8"
    name = "resPowerDataTosaveDB"/>
<connector start = "//@element.11" end = "//@element.10"
    name = "saveXMIToprintRealTimePower"/>
<connector start = "//@element.12" end = "//@element.10"
    name = "saveDBToprintRealTimePower"/>
</ced:CauseEffectModel>
```

At occurring an error on the inverter, it shows to transform XMI code of message sequence diagram for backup of power generation. Table 10 shows XMI code transformation of Message sequence diagram at an error on the inverter.

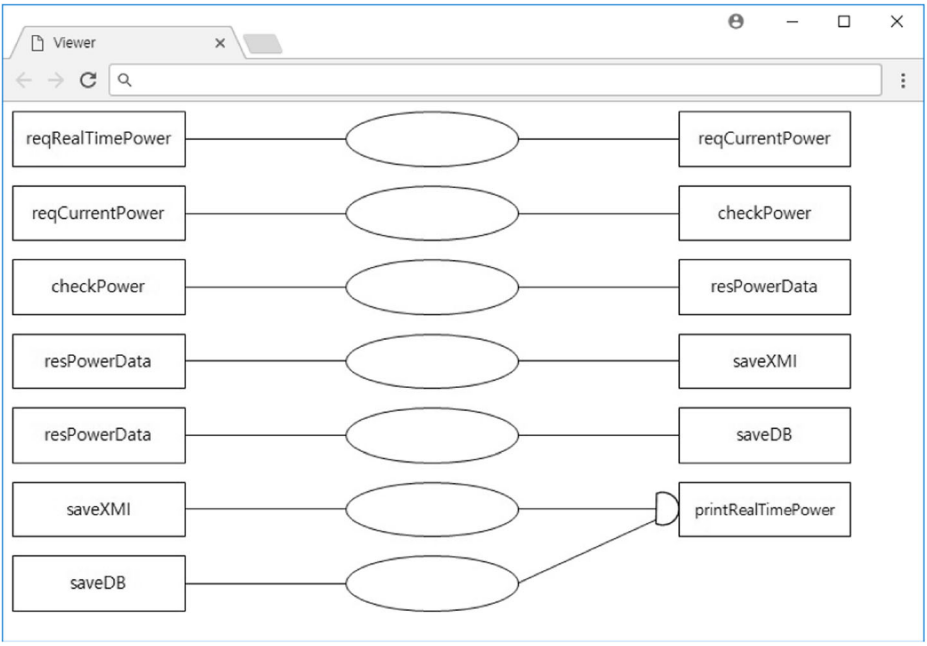


Fig. 16 Cause-effect diagram of Table 11

Table 12 XMI code transformation of cause-effect graph at disconnection with M-PVMS client

```
<?xmlversion = "1.0" encoding = "ISO-8859-1"?>
<ced:CauseEffectModel xmi:version = "2.0"
    xmlns:xmi = "http://www.omg.org/XMI"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ced = "http://ced/1.0">
<element xsi:type = "ced:Effect" name = "reqCurrentPower"
    ownedConnector = "//@connector.0"/>
<element xsi:type = "ced:Cause" name = "reqRealTimePower"
    ownedConnector = "//@connector.0"/>
<element xsi:type = "ced:Effect" name = "checkPower"
    ownedConnector = "//@connector.1"/>
<element xsi:type = "ced:Cause" name = "reqCurrentPower"
    ownedConnector = "//@connector.1"/>
<element xsi:type = "ced:Effect" name = "retDisconnectError"
    ownedConnector = "//@connector.2"/>
<element xsi:type = "ced:Cause" name = "checkPower"
    ownedConnector = "//@connector.2"/>
<element xsi:type = "ced:Effect" name = "retAccessFailureError"
    ownedConnector = "//@connector.3"/>
<element xsi:type = "ced:Cause" name = "retDisconnectError"
    ownedConnector = "//@connector.3"/>
<element xsi:type = "ced:Effect"
    name = "sendErrorMail"
ownedConnector = "//@connector.4"/>
<element xsi:type = "ced:Cause"
    name = "retAccessFailureError"
    ownedConnector = "//@connector.4"/>
<element xsi:type = "ced:Effect"
    name = "printErrorMessage"
    ownedConnector = "//@connector.5"/>
<element xsi:type = "ced:Cause"
    name = "sendErrorMail"
    ownedConnector = "//@connector.5"/>
<connector start = "//@element.1"
    end = "//@element.0"
    name = "reqRealTimePowerToreqCurrentPower"/>
<connector start = "//@element.3"
    end = "//@element.2"
    name = "reqCurrentPowerTocheckPower"/>
<connector start = "//@element.5"
    end = "//@element.4"
    name = "checkPowerToretDisconnectError"/>
<connector start = "//@element.7"
    end = "//@element.6"
    name = "
    retDisconnectErrorToretAccessFailureError"/>
<connector start = "//@element.9"
    end = "//@element.8"
    name = "retAccessFailureErrorTosendErrorMail"/>
<connector start = "//@element.11"
    end = "//@element.10"
    name = "sendErrorMailToprintErrorMessage"/>
</ced:CauseEffectModel>
```

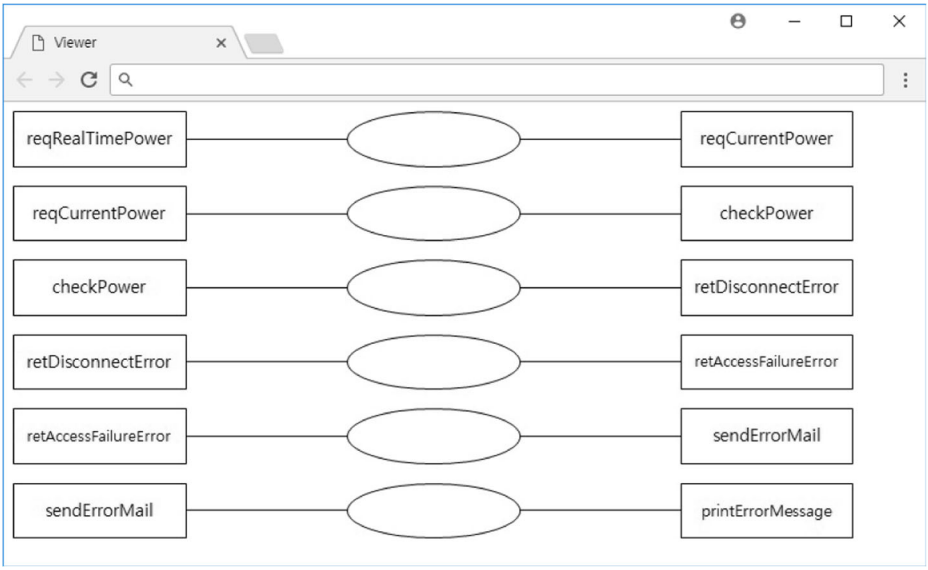


Fig. 17 Cause-effect diagram of Table 12

Table 13 XMI code transformation of cause-effect gragh for real time generaiton

```
<?xml version = "1.0" encoding = "ISO-8859-1"?>
<ced:CauseEffectModel xmi:version = "2.0"
    xmlns:xmi = "http://www.omg.org/XMI"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ced = "http://ced/1.0">
<element xsi:type = "ced:Effect" name = "resPower"
    ownedConnector = "//@connector.0"/>
<element xsi:type = "ced:Cause" name = "
    reqPowerandSensor"
    ownedConnector = "//@connector.0"/>
<element xsi:type = "ced:Effect" name = "resSensor"
    ownedConnector = "//@connector.1"/>
<element xsi:type = "ced:Cause" name = "reqPowerandSensor"
    ownedConnector = "//@connector.1"/>
<element xsi:type = "ced:Effect"
    name = "savePowerandSensorinDB"
    ownedConnector = "//@connector.2 //@connector.3"
    eType = "AND"/>
<element xsi:type = "ced:Cause" name = "resPower"
    ownedConnector = "//@connector.2"/>
<element xsi:type = "ced:Cause" name = "resSensor"
    ownedConnector = "//@connector.3"/>
<element xsi:type = "ced:Effect"
    name = "sendPowerandSensor"
ownedConnector = "//@connector.4"/>
<element xsi:type = "ced:Cause"
    name = "savePowerandSensorinDB"
    ownedConnector = "//@connector.4"/>
<element xsi:type = "ced:Effect" name = "recvSuccess"
    ownedConnector = "//@connector.5"/>
<element xsi:type = "ced:Cause"
    name = "sendPowerandSensor"
    ownedConnector = "//@connector.5"/>
<connector start = "//@element.1" end = "//@element.0"
    name = "reqPowerandSensorToresPower"/>
<connector start = "//@element.3" end = "//@element.2"
    name = "reqPowerandSensorToresSensor"/>
<connector start = "//@element.5" end = "//@element.4"
    name = "resPowerTosavePowerandSensorinDB"/>
<connector start = "//@element.6" end = "//@element.4"
    name = "resSensorTosavePowerandSensorinDB"/>
<connector start = "//@element.8" end = "//@element.7"
    name = "
    savePowerandSensorinDBTosendPowerandSensor"/>
<connector start = "//@element.10" end = "//@element.9"
    name = "sendPowerandSensorTorecvSuccess"/>
</ced:CauseEffectModel>
```

4.2 XMI code transformation of cause-effect graph

Our proposal mechanism is transformed XMI code of cause-effect graph with XMI code of message sequence diagram [1]. Table 11 shows to transform XMI code transformation of

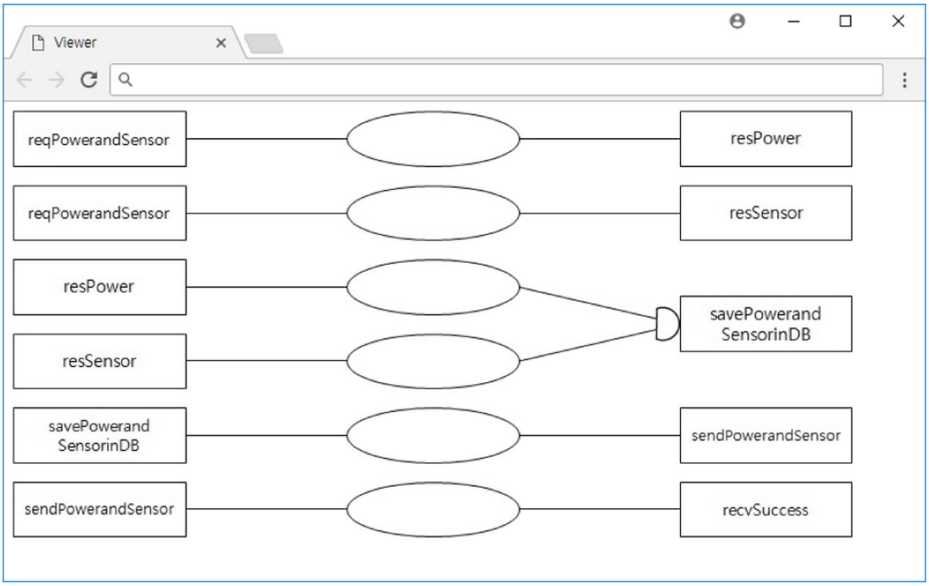


Fig. 18 Cause-effect diagram of Table 13

Table 14 XMI code transformation of cause-effect graph at disconnection of M-PVMS server

```
<?xml version = "1.0" encoding = "ISO-8859-1"?>
<ced:CauseEffectModel xmi:version = "2.0"
    xmlns:xmi = "http://www.omg.org/XMI"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ced = "http://ced/1.0">
<element xsi:type = "ced:Effect" name = "resPower"
    ownedConnector = "//@connector.0"/>
<element xsi:type = "ced:Cause" name = "reqPowerandSensor"
    ownedConnector = "//@connector.0"/>
<element xsi:type = "ced:Effect" name = "resSensor"
    ownedConnector = "//@connector.1"/>
<element xsi:type = "ced:Cause" name = "reqPowerandSensor"
    ownedConnector = "//@connector.1"/>
<element xsi:type = "ced:Effect"
    name = "savePowerandSensorinDB"
    ownedConnector = "//@connector.2 //@connector.3"
    eType = "AND"/>
<element xsi:type = "ced:Cause" name = "resPower"
    ownedConnector = "//@connector.2"/>
<element xsi:type = "ced:Cause" name = "resSensor"
    ownedConnector = "//@connector.3"/>
<element xsi:type = "ced:Effect" name = "sendPowerandSensor"
    ownedConnector = "//@connector.4"/>
<element xsi:type = "ced:Cause" name = "
    savePowerandSensorinDB"
    ownedConnector = "//@connector.4"/>
<element xsi:type = "ced:Effect" name = "retAccessFailureError"
    ownedConnector = "//@connector.5"/>
<element xsi:type = "ced:Cause" name = "sendPowerandSensor"
    ownedConnector = "//@connector.5"/>
<element xsi:type = "ced:Effect" name = "saveLastSendData"
    ownedConnector = "//@connector.6"/>
<element xsi:type = "ced:Cause" name = "retAccessFailureError"
    ownedConnector = "//@connector.6"/>
<connector start = "//@element.1" end = "//@element.0"
    name = "reqPowerandSensorToresPower"/>
<connector start = "//@element.3" end = "//@element.2"
    name = "reqPowerandSensorToresSensor"/>
<connector start = "//@element.5" end = "//@element.4"
    name = "resPowerTosavePowerandSensorinDB"/>
<connector start = "//@element.6" end = "//@element.4"
    name = "resSensorTosavePowerandSensorinDB"/>
<connector start = "//@element.8" end = "//@element.7"
    name = "savePowerandSensorinDBTosendPowerandSensor"/>
<connector start = "//@element.10" end = "//@element.9"
    name = "sendPowerandSensorToretAccessFailureError"/>
<connector start = "//@element.12" end = "//@element.11"
    name = "retAccessFailureErrorTosaveLastSendData"/>
</ced:CauseEffectModel>
```

cause-effect diagram for real time power generation. Figure 16 shows to convert a graph image from XMI code transformation of cause-effect graph for *'real time power generation'*.

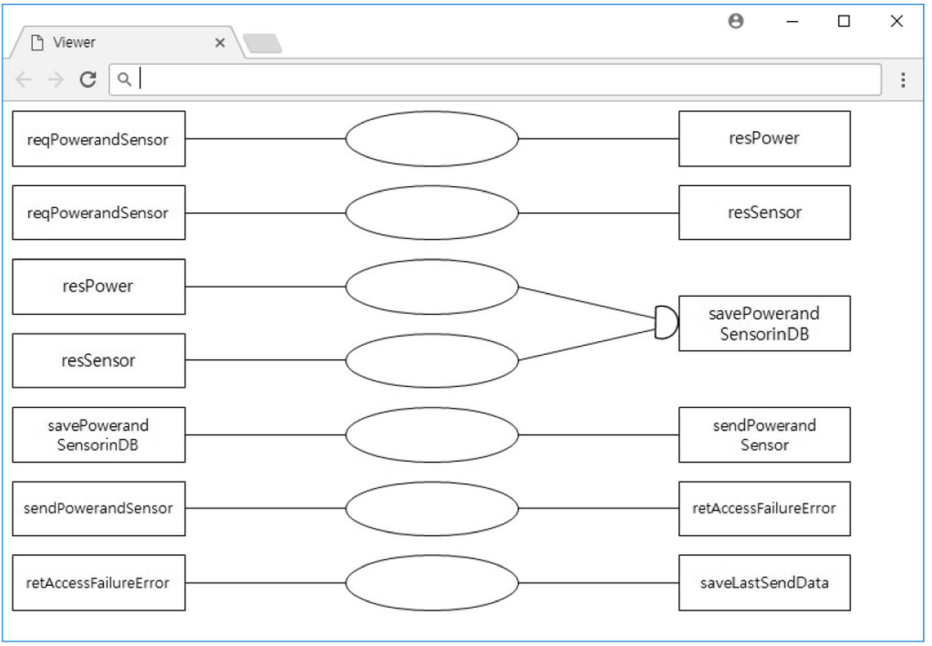


Fig. 19 Cause-effect diagram of Table 14

Table 15 XMI code transformation of cause-effect graph for backup of power generation at an error on the inverter

```
<?XMI version = "1.0" encoding = "ISO-8859-1"?>
<ced:CauseEffectModel xmi:version = "2.0"
    XMIns:xmi = "http://www.omg.org/XMI"
    XMIns:xsi = "http://www.w3.org/2001/XMISchema-instance"
    XMIns:ced = "http://ced/1.0">
<element xsi:type = "ced:Effect" name = "resPower"
    ownedConnector = "//@connector.0"/>
<element xsi:type = "ced:Cause" name = "reqPowerandSensor"
    ownedConnector = "//@connector.0"/>
<element xsi:type = "ced:Effect" name = "resSensor"
    ownedConnector = "//@connector.1"/>
<element xsi:type = "ced:Cause" name = "reqPowerandSensor"
    ownedConnector = "//@connector.1"/>
<element xsi:type = "ced:Effect" name = "resError"
    ownedConnector = "//@connector.2"/>
<element xsi:type = "ced:Cause" name = "reqPowerandSensor"
    ownedConnector = "//@connector.2"/>
<element xsi:type = "ced:Effect" name = "
    savePowerandSensorinDB"
    ownedConnector = "//@connector.3 //@connector.4
    //@connector.5" eType = "AND"/>
<element xsi:type = "ced:Cause" name = "resPower"
    ownedConnector = "//@connector.3"/>
<element xsi:type = "ced:Cause" name = "resSensor"
    ownedConnector = "//@connector.4"/>
<element xsi:type = "ced:Cause" name = "resError"
    ownedConnector = "//@connector.5"/>
<element xsi:type = "ced:Effect" name = "sendPowerandSensor"
ownedConnector = "//@connector.6"/>
<element xsi:type = "ced:Cause"
    name = "savePowerandSensorinDB"
ownedConnector = "//@connector.6"/>
<element xsi:type = "ced:Effect" name = "sendErrorMail"
    ownedConnector = "//@connector.7"/>
<element xsi:type = "ced:Cause" name = "sendPowerandSensor"
    ownedConnector = "//@connector.7"/>
<element xsi:type = "ced:Effect" name = "recvSuccess"
    ownedConnector = "//@connector.8"/>
<element xsi:type = "ced:Cause" name = "sendErrorMail"
    ownedConnector = "//@connector.8"/>
<connector start = "//@element.1" end = "//@element.0"
    name = "reqPowerandSensorToresPower"/>
<connector start = "//@element.3" end = "//@element.2"
    name = "reqPowerandSensorToresSensor"/>
<connector start = "//@element.5" end = "//@element.4"
    name = "reqPowerandSensorToresError"/>
<connector start = "//@element.7" end = "//@element.6"
    name = "resPowerTosavePowerandSensorinDB"/>
<connector start = "//@element.8" end = "//@element.6"
    name = "resSensorTosavePowerandSensorinDB"/>
<connector start = "//@element.9" end = "//@element.6"
    name = "resErrorTosavePowerandSensorinDB"/>
<connector start = "//@element.11" end = "//@element.10"
    name = "
    savePowerandSensorinDBTosendPowerandSensor"/>
<connector start = "//@element.13" end = "//@element.12"
    name = "sendPowerandSensorTosendErrorMail"/>
<connector start = "//@element.15" end = "//@element.14"
    name = "sendErrorMailTorecvSuccess"/>
</ced:CauseEffectModel>
```

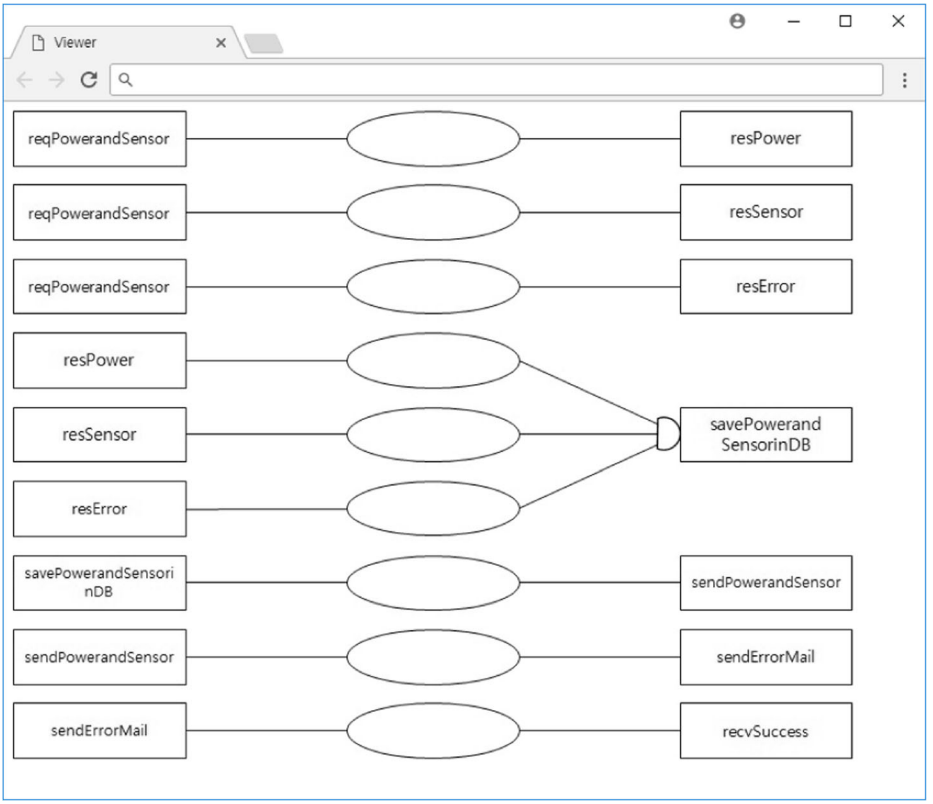


Fig. 20 Cause-effect diagram of Table 15

Table 12 shows XMI code Transformation of Cause-effect graph at disconnection with M-PVMS client. Figure 17 shows a graph image from XMI code transformation of cause-effect graph for *'disconnection with M-PVMS client'*.

Table 16 XMI code transformation of decision table for real time power generation

```
<?xml version = "1.0"
   encoding = "ISO-8859-1"?>
<decision:DecisionTableModel
   xmi:version = "2.0"
   xmlns:xmi = "http://www.omg.org/XMI"
   xmlns:decision = "http://decision/1.0">
<cause_element>
<input_line
   inputmess = "reqRealTimePower">
<in id = "0">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "reqCurrentPower">
<in id = "1">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "checkPower">
<in id = "2">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "resPowerData">
<in id = "3">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "resPowerData">
<in id = "4">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "saveXMI">
<in id = "5">
<item item_id = "F"/>
<item item_id = "F"/>
<item item_id = "T"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "saveDB">
<in id = "6">
<item item_id = "F"/>
<item item_id = "T"/>
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
</cause_element>
<effect_element>
<output_line outputmess = "reqCurrentPower"
   eType = "NORMAL">
<out id = "0">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
<output_line outputmess = "checkPower"
   eType = "NORMAL">
<out id = "1">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
<output_line outputmess = "resPowerData"
   eType = "NORMAL">
<out id = "2">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
<output_line outputmess = "saveXMI" eType = "NORMAL">
<out id = "3">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
<output_line outputmess = "saveDB" eType = "NORMAL">
<out id = "4">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
<output_line outputmess = "printRealTimePower"
   eType = "AND">
<out id = "5">
<item item_id = "F"/>
<item item_id = "F"/>
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
</effect_element>
</decision:DecisionTableModel>
```

Table 13 shows to transform XMI code of cause-effect graph with XMI message sequence diagram at normal situation for real time power generation. Figure 18 shows a graph image from XMI code transformation for *'real time generation'*.

Table 14 shows to transform XMI code of cause-effect graph with XMI code of message sequence diagram at disconnection of M-PVMS server. Figure 19 shows a graph image from XMI code transformation of cause-effect graph for *'disconnection of M-PVMS server'*.

Table 15 shows to transform XMI code of cause-effect graph with XMI code of message sequence diagram for backup of power generation at an error on the inverter. Figure 20 shows a graph image from XMI code transformation for *'power generation at an error in the inverter'*.

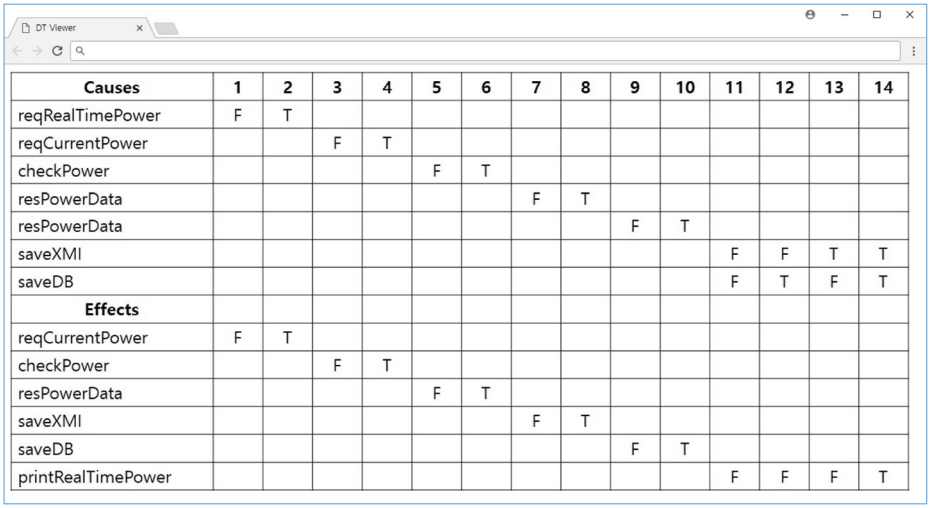
4.3 XMI code transformation of decision table with cause-effect graph

Our mechanism is also transformed XMI code of decision table with XMI code of cause-effect graph. Table 16 shows to transform XMI code of decision table with cause-effect graph for real time power generation. Figure 21 shows a graph image from XMI code transformation of decision table for *'real time generation'*.

Table 17 shows to transform XMI code of decision table with XMI code of message sequence diagram for power generation at disconnection of M-PVMS client. Figure 22 shows a graph image from XMI code transformation for '*power generation at disconnection of M-PVMS client*'.

Table 18 shows to transform XMI code of decision table with XMI code of message sequence diagram for power generation at normal situation. Figure 23 shows a graph image from XMI code transformation of decision table for *'normal power generation'*.

Table 19 shows to transform XMI code of decision table with XMI code of message sequence diagram for power generation at disconnection of M-PVMS server.

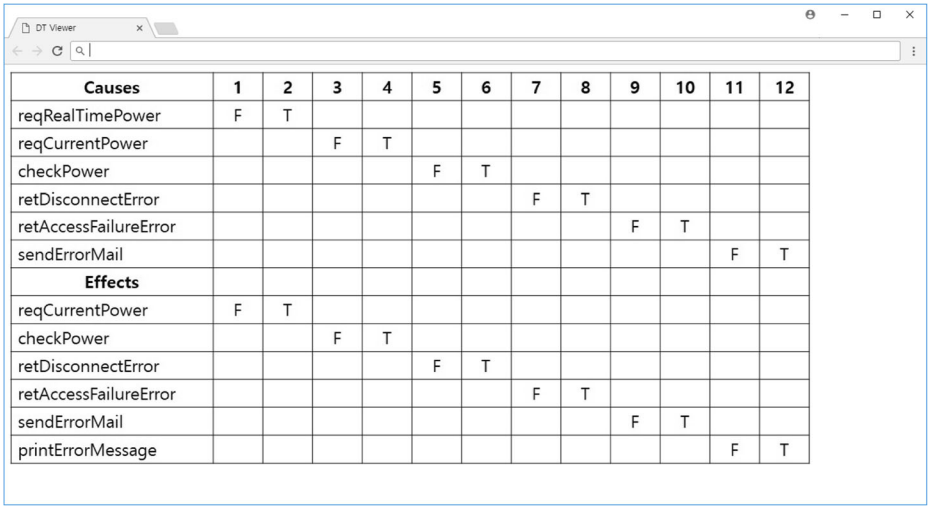


Causes	1	2	3	4	5	6	7	8	9	10	11	12	13	14
reqRealTimePower	F	T												
reqCurrentPower			F	T										
checkPower					F	T								
resPowerData							F	T						
resPowerData									F	T				
saveXMI											F	F	T	T
saveDB											F	T	F	T
Effects														
reqCurrentPower	F	T												
checkPower			F	T										
resPowerData					F	T								
saveXMI							F	T						
saveDB									F	T				
printRealTimePower											F	F	F	T

Fig. 21 Decision table of Table 16

Table 17 XMI Code transformation of decision table at disconnection of M-PVMS client

```
<?xml version = "1.0" encoding = "ISO-8859-1"?>
<decision:DecisionTableModel xmi:version = "2.0"
    xmlns:xmi = "http://www.omg.org/XMI"
    xmlns:decision = "http://decision/1.0">
<cause_element>
<input_line inputmess = "reqRealTimePower">
<in id = "0">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "reqCurrentPower">
<in id = "1">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "checkPower">
<in id = "2">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "resPowerData">
<in id = "3">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "resPowerData">
<in id = "4">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "saveXMI">
<in id = "5">
<item item_id = "F"/>
<item item_id = "F"/>
<item item_id = "T"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "saveDB">
<in id = "6">
<item item_id = "F"/>
<item item_id = "T"/>
<item item_id = "F"/>
<?xml version = "1.0" encoding = "ISO-8859-1"?>
<decision:DecisionTableModel xmi:version = "2.0"
    xmlns:xmi = "http://www.omg.org/XMI"
    xmlns:decision = "http://decision/1.0">
<cause_element>
<input_line inputmess = "reqRealTimePower">
<in id = "0">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "reqCurrentPower">
<in id = "1">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "checkPower">
<in id = "2">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "retDisconnectError">
<in id = "3">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "retAccessFailureError">
<in id = "4">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "sendErrorMail">
<in id = "5">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
</cause_element>
<effect_element>
<output_line outputmess = "reqCurrentPower" eType = "NORMAL">
<out id = "0">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
<output_line outputmess = "checkPower" eType = "NORMAL">
<out id = "1">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
<output_line outputmess = "retDisconnectError" eType = "NORMAL">
<out id = "2">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
<output_line outputmess = "retAccessFailureError" eType = "NORMAL">
<out id = "3">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
<output_line outputmess = "sendErrorMail" eType = "NORMAL">
<out id = "4">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
<output_line outputmess = "printErrorMessage" eType = "NORMAL">
<out id = "5">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
</effect_element>
</decision:DecisionTableModel>
```

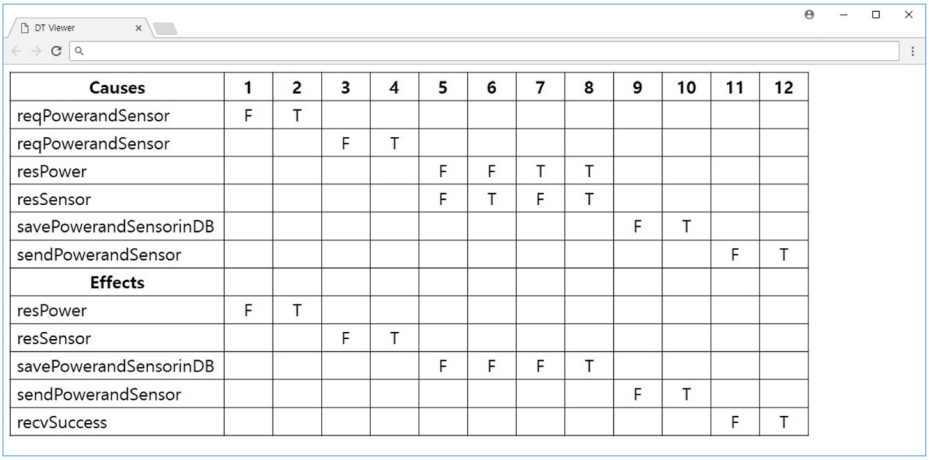


Causes	1	2	3	4	5	6	7	8	9	10	11	12
reqRealTimePower	F	T										
reqCurrentPower			F	T								
checkPower					F	T						
retDisconnectError							F	T				
retAccessFailureError									F	T		
sendErrorMail											F	T
Effects												
reqCurrentPower	F	T										
checkPower			F	T								
retDisconnectError					F	T						
retAccessFailureError							F	T				
sendErrorMail									F	T		
printErrorMessage											F	T

Fig. 22 Decision table of Table 17

Table 18 XMI code transformation of decision table for normal power generation

```
<?xml version = "1.0" encoding = "ISO-8859-1"?>
<decision:DecisionTableModel xmi:version = "2.0"
    xmlns:xmi = "http://www.omg.org/XMI"
    xmlns:decision = "http://decision/1.0">
<cause_element>
<input_line inputmess = "reqPowerandSensor">
<in id = "0">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "reqPowerandSensor">
<in id = "1">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "resPower">
<in id = "2">
<item item_id = "F"/>
<item item_id = "F"/>
<item item_id = "T"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "resSensor">
<in id = "3">
<item item_id = "F"/>
<item item_id = "T"/>
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "savePowerandSensorinDB">
<in id = "4">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "sendPowerandSensor">
<in id = "5">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
</cause_element>
<effect_element>
<output_line outputmess = "resPower" eType = "NORMAL">
<out id = "0">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
<output_line outputmess = "resSensor" eType = "NORMAL">
<out id = "1">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
<output_line outputmess = "savePowerandSensorinDB" eType = "AND">
<out id = "2">
<item item_id = "F"/>
<item item_id = "F"/>
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
<output_line outputmess = "sendPowerandSensor" eType = "NORMAL">
<out id = "3">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
<output_line outputmess = "recvSuccess" eType = "NORMAL">
<out id = "4">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
</effect_element>
</decision:DecisionTableModel>
```



Causes	1	2	3	4	5	6	7	8	9	10	11	12
reqPowerandSensor	F	T										
reqPowerandSensor			F	T								
resPower					F	F	T	T				
resSensor					F	T	F	T				
savePowerandSensorinDB									F	T		
sendPowerandSensor											F	T
Effects												
resPower	F	T										
resSensor			F	T								
savePowerandSensorinDB					F	F	F	T				
sendPowerandSensor									F	T		
recvSuccess											F	T

Fig. 23 Decision Table of Table 18

Table 19 XMI code transformation of decision table for power generation at disconnection of M-PVMS server

```
<?xml version = "1.0" encoding = "ISO-8859-1"?>
<decision:DecisionTableModel xmi:version = "2.0"
    xmlns:xmi = "http://www.omg.org/XMI"
    xmlns:decision = "http://decision/1.0">
<cause_element>
<input_line inputmess = "reqPowerandSensor">
<in id = "0">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "reqPowerandSensor">
<in id = "1">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "resPower">
<in id = "2">
<item item_id = "F"/>
<item item_id = "F"/>
<item item_id = "T"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "resSensor">
<in id = "2">
<item item_id = "F"/>
<item item_id = "T"/>
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "savePowerandSensorinDB">
<in id = "3">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "sendPowerandSensor">
<in id = "4">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "retAccessFailureError">
<in id = "5">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
</cause_element>
<effect_element>
<output_line outputmess = "resPower" eType = "NORMAL">
<out id = "0">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
<output_line outputmess = "resSensor" eType = "NORMAL">
<out id = "1">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
<output_line outputmess = "savePowerandSensorinDB" eType = "AND">
<out id = "2">
<item item_id = "F"/>
<item item_id = "F"/>
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
<output_line outputmess = "sendPowerandSensor" eType = "NORMAL">
<out id = "3">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
<output_line outputmess = "retAccessFailureError" eType = "NORMAL">
<out id = "4">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
<output_line outputmess = "saveLastSendData" eType = "NORMAL">
<out id = "5">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
</effect_element>
</decision:DecisionTableModel>
```

Figure 24 shows a graph image from XMI code transformation for *'power generation at disconnection of M-PVMS server'*.

Table 20 shows to transform XMI code of decision table with XMI code of cause-effect graph for power generation at an error of the inverter. Figure 25 shows a graph image from XMI code transformation of decision table for *'power generation at an error of the inverter'*.

4.4 XMI code transformation of test case with decision table

Our mechanism does also transform XMI code of test case with XMI code of decision table. Figure 26 shows to transform XMI code of decision table with cause-effect graph for real time power generation. To visualize XMI code of test case, we should save XMI code and load this file into Microsoft excel program.

Table 21 shows to transform XMI code of test case with XMI code of decision table for *'Display function of power generation'*.

Figure 27 shows test case from Microsoft Excel file which are saved with XMI code of test case for *'display function of power generation'*.

Table 22 shows to transform XMI code of test case with XMI code of decision table for *'a diaplay function of power generation at disconnection of M-PVMS client'*.

Figure 28 shows to transform XMI code of Microsoft excel file with XMI code of test case for *'a display function of power generation at disconnection of M-PVMS client'*.

Table 23 shows to transform XMI code of Microsoft Excel file with XMI code of test case for a backup function of power generation.

Figure 29 shows to transform XMI code of Microsoft excel file with XMI code of test case for *'a backup function of a normal power generation'*.

Table 24 shows to transform XMI code of Microsoft Excel file with XMI code of test case for a backup function of power generation at disconnection of M-PVMS server.

DT Viewer

Causes	1	2	3	4	5	6	7	8	9	10	11	12	13	14
reqPowerandSensor	F	T												
reqPowerandSensor			F	T										
resPower					F	F	T	T						
resSensor					F	T	F	T						
savePowerandSensorinDB									F	T				
sendPowerandSensor											F	T		
retAccessFailureError													F	T
Effects														
resPower	F	T												
resSensor			F	T										
savePowerandSensorinDB					F	F	F	T						
sendPowerandSensor									F	T				
retAccessFailureError											F	T		
saveLastSendData													F	T

Fig. 24 Decision table of Table 19

Table 20 XMI code transformation of decision table for power generation at an error of the inverter

```
<?xml version = "1.0" encoding = "ISO-8859-1"?>
<decision:DecisionTableModel xmi:version = "2.0"
    xmlns:xmi = "http://www.omg.org/XMI"
    xmlns:decision = "http://decision/1.0">
<cause_element>
<input_line inputmess = "reqPowerandSensor">
<in id = "0">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "reqPowerandSensor">
<in id = "1">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "reqPowerandSensor">
<in id = "1">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "reqPowerandSensor">
<in id = "2">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "resPower">
<in id = "3">
<item item_id = "F"/>
<item item_id = "F"/>
<item item_id = "F"/>
<item item_id = "F"/>
<item item_id = "T"/>
<item item_id = "T"/>
<item item_id = "T"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "resSensor">
<in id = "4">
<item item_id = "F"/>
<item item_id = "F"/>
<item item_id = "T"/>
<item item_id = "T"/>
<item item_id = "F"/>
<item item_id = "F"/>
<item item_id = "T"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "resError">
<in id = "5">
<item item_id = "F"/>
<item item_id = "T"/>
<item item_id = "F"/>
<item item_id = "T"/>
<item item_id = "F"/>
<item item_id = "T"/>
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "savePowerandSensorinDB">
<in id = "6">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "sendPowerandSensor">
<in id = "7">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
<input_line inputmess = "sendErrorMail">
<in id = "8">
<item item_id = "F"/>
<item item_id = "T"/>
</in>
</input_line>
</cause_element>
<effect_element>
<output_line outputmess = "resPower" eType = "NORMAL">
<out id = "0">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
<output_line outputmess = "resSensor" eType = "NORMAL">
<out id = "1">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
<output_line outputmess = "resError" eType = "NORMAL">
<out id = "2">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
<output_line outputmess = "savePowerandSensorinDB" eType = "AND">
<out id = "3">
<item item_id = "F"/>
<item item_id = "F"/>
<item item_id = "F"/>
<item item_id = "F"/>
<item item_id = "F"/>
<item item_id = "F"/>
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
<output_line outputmess = "sendPowerandSensor" eType = "NORMAL">
<out id = "4">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
<output_line outputmess = "sendErrorMail" eType = "NORMAL">
<out id = "5">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
<output_line outputmess = "recvSuccess" eType = "NORMAL">
<out id = "6">
<item item_id = "F"/>
<item item_id = "T"/>
</out>
</output_line>
</effect_element>
</decision:DecisionTableModel>
```

Causes	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
reqPowerandSensor	F	T																		
reqPowerandSensor			F	T																
reqPowerandSensor					F	T														
resPower							F	F	F	F	T	T	T	T						
resSensor							F	F	T	T	F	F	T	T						
resError							F	T	F	T	F	T	F	T						
savePowerandSensorinDB															F	T				
sendPowerandSensor																	F	T		
sendErrorMail																			F	T
Effects																				
resPower	F	T																		
resSensor			F	T																
resError					F	T														
savePowerandSensorinDB							F	F	F	F	F	F	F	T						
sendPowerandSensor															F	T				
sendErrorMail																	F	T		
rescSuccess																			F	T

Fig. 25 Decision table of Table 20

Figure 30 shows to transform XMI code of Microsoft excel file with XMI code of test case for *'a backup function of a power generation at disconnection of M-PVMS server'*.

Table 25 shows to transform XMI code of Microsoft excel file with XMI code of test case for a backup function of power generation at an error of the inverter.

Figure 31 shows to transform Microsoft excel file with XMI code of test case for *'a power generation at an error of the inverter'*.

4.5 System validation with the generated test cases

Our approach is focusing on validating the dynamic behaviors of the safety system, which generate test cases based on message sequence diagram, that is, use case scenario to design the dynamic behaviors of the solar power monitoring system. As a result, it can validate the safety critical system with minimal test cases. Table 26 shows test result for 'the display function of power generation'.

Table 27 shows test result of the display function of power generation at disconnection of M-PVMS client. TC10 and TC11 failed because the mail server was turned off.

Table 28 shows test result of a backup function for a normal power generation.

Table 29 shows test result of a backup function for a power generation at disconnection of M-PVMS server.

Table 30 shows test result of power generation at an error of the inverter. TC18 and TC20 failed because the mail server was turned off.

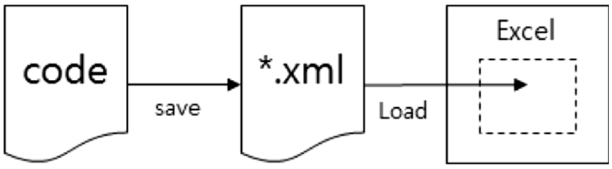


Fig. 26 Visualization of XMI code of Test Case

Table 21 XMI code transformation of test case for real time power generation

```
<?xml version = "1.0" encoding = "ISO-8859-1"?>
<testcase:TestCaseModel xmi:version = "2.0"
    xmlns:xmi = "http://www.omg.org/XMI"
    xmlns:testcase = "http://testcase/1.0">
<testcaseline no = "TC1">
<testcase_pre testpre = "reqRealTimePower = F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "reqCurrentPower = F"/>
</testcaseline>
<testcaseline no = "TC2">
<testcase_pre testpre = "reqRealTimePower = T"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "reqCurrentPower = T"/>
</testcaseline>
<testcaseline no = "TC3">
<testcase_pre testpre = "reqCurrentPower = F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "checkPower = F"/>
</testcaseline>
<testcaseline no = "TC4">
<testcase_pre testpre = "reqCurrentPower = T"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "checkPower = T"/>
</testcaseline>
<testcaseline no = "TC5">
<testcase_pre testpre = "checkPower = F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "resPowerData = F"/>
</testcaseline>
<testcaseline no = "TC6">
<testcase_pre testpre = "checkPower = T"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "resPowerData = T"/>
</testcaseline>
<testcaseline no = "TC7">
<testcase_pre testpre = "resPowerData = F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "saveXMI=F"/>
</testcaseline>
<testcaseline no = "TC8">
<testcase_pre testpre = "resPowerData = T"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "saveXMI = T"/>
</testcaseline>
<testcaseline no = "TC9">
<testcase_pre testpre = "resPowerData = F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "saveDB=F"/>
</testcaseline>
<testcaseline no = "TC10">
<testcase_pre testpre = "resPowerData = T"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "saveDB = T"/>
</testcaseline>
<testcaseline no = "TC11">
<testcase_pre testpre = "saveXMI=F,saveDB=F"/>
<testcase_con testcon = "AND"/>
<testcase_res testresult = "printRealTimePower = F"/>
</testcaseline>
<testcaseline no = "TC12">
<testcase_pre testpre = "saveXMI=F,saveDB = T"/>
<testcase_con testcon = "AND"/>
<testcase_res testresult = "printRealTimePower = F"/>
</testcaseline>
<testcaseline no = "TC13">
<testcase_pre testpre = "saveXMI = T,saveDB=F"/>
<testcase_con testcon = "AND"/>
<testcase_res testresult = "printRealTimePower = F"/>
</testcaseline>
<testcaseline no = "TC14">
<testcase_pre testpre = "saveXMI = T,saveDB = T"/>
<testcase_con testcon = "AND"/>
<testcase_res testresult = "printRealTimePower = T"/>
</testcaseline>
</testcase:TestCaseModel>
```

no	testpre	testcon	testresult
TC1	reqRealTimePower=F	NORMAL	reqCurrentPower=F
TC2	reqRealTimePower=T	NORMAL	reqCurrentPower=T
TC3	reqCurrentPower=F	NORMAL	checkPower=F
TC4	reqCurrentPower=T	NORMAL	checkPower=T
TC5	checkPower=F	NORMAL	resPowerData=F
TC6	checkPower=T	NORMAL	resPowerData=T
TC7	resPowerData=F	NORMAL	saveXML=F
TC8	resPowerData=T	NORMAL	saveXML=T
TC9	resPowerData=F	NORMAL	saveDB=F
TC10	resPowerData=T	NORMAL	saveDB=T
TC11	saveXML=F,saveDB=F	AND	printRealTimePower=F
TC12	saveXML=F,saveDB=T	AND	printRealTimePower=F
TC13	saveXML=T,saveDB=F	AND	printRealTimePower=F
TC14	saveXML=T,saveDB=T	AND	printRealTimePower=T

Fig. 27 An image of test case based on microsoft excel file for power generation

Table 22 XMI code transformation of test case for a display function of power geberation at disconnection of M-PVMS client

```
<?XMI version = "1.0" encoding = "ISO-8859-1"?>
<testcase:TestCaseModel xmi:version = "2.0"
   XMIns:xmi = "http://www.omg.org/XMI"
   XMIns:testcase = "http://testcase/1.0″>
<testcaseline no = "TC1">
<testcase_pre testpre = "reqRealTimePower = F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "reqCurrentPower = F"/>
</testcaseline>
<testcaseline no = "TC2">
<testcase_pre testpre = "reqRealTimePower = T"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "reqCurrentPower = T"/>
</testcaseline>
<testcaseline no = "TC3">
<testcase_pre testpre = "reqCurrentPower = F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "checkPower = F"/>
</testcaseline>
<testcaseline no = "TC4">
<testcase_pre testpre = "reqCurrentPower = T"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "checkPower = T"/>
</testcaseline>
<testcaseline no = "TC5">
<testcase_pre testpre = "checkPower = F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "retDisconnectError = F"/>
</testcaseline>
<testcaseline no = "TC6">
<testcase_pre testpre = "checkPower = T"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "retDisconnectError = T"/>
</testcaseline>
<testcaseline no = "TC7">
<testcase_pre testpre = "retDisconnectError = F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "retAccessFailureError = F"/>
</testcaseline>
<testcaseline no = "TC8">
<testcase_pre testpre = "retDisconnectError = T"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "retAccessFailureError = T"/>
</testcaseline>
<testcaseline no = "TC9">
<testcase_pre testpre = "retAccessFailureError = F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "sendErrorMail = F"/>
</testcaseline>
<testcaseline no = "TC10">
<testcase_pre testpre = "retAccessFailureError = T"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "sendErrorMail = T"/>
</testcaseline>
<testcaseline no = "TC11">
<testcase_pre testpre = "sendErrorMail = F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "printErrorMessage = F"/>
</testcaseline>
<testcaseline no = "TC12">
<testcase_pre testpre = "sendErrorMail = T"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "printErrorMessage = T"/>
</testcaseline>
</testcase:TestCaseModel>
```

no	testpre	testcon	testresult
TC1	reqRealTimePower=F	NORMAL	reqCurrentPower=F
TC2	reqRealTimePower=T	NORMAL	reqCurrentPower=T
TC3	reqCurrentPower=F	NORMAL	checkPower=F
TC4	reqCurrentPower=T	NORMAL	checkPower=T
TC5	checkPower=F	NORMAL	retDisconnectError=F
TC6	checkPower=T	NORMAL	retDisconnectError=T
TC7	retDisconnectError=F	NORMAL	retAccessFailureError=F
TC8	retDisconnectError=T	NORMAL	retAccessFailureError=T
TC9	retAccessFailureError=F	NORMAL	sendErrorMail=F
TC10	retAccessFailureError=T	NORMAL	sendErrorMail=T
TC11	sendErrorMail=F	NORMAL	printErrorMessage=F
TC12	sendErrorMail=T	NORMAL	printErrorMessage=T

Fig. 28 An image of test case based on microsoft excel file for power generation at disconnection of M-PVMS client

Table 23 XMI code transformation of microsoft excel for a normal power geberation

```
<?XMI version = "1.0" encoding = "ISO-8859-1"?>
<testcase:TestCaseModel xmi:version = "2.0"
   XMIns:xmi = "http://www.omg.org/XMI"
   XMIns:testcase = "http://testcase/1.0″>
<testcaseline no = "TC1">
<testcase_pre testpre = "reqPowerandSensor = F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "resPower = F"/>
</testcaseline>
<testcaseline no = "TC2">
<testcase_pre testpre = "reqPowerandSensor = T"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "resPower = T"/>
</testcaseline>
<testcaseline no = "TC3">
<testcase_pre testpre = "reqPowerandSensor = F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "resSensor = F"/>
</testcaseline>
<testcaseline no = "TC4">
<testcase_pre testpre = "reqPowerandSensor = T"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "resSensor = T"/>
</testcaseline>
<testcaseline no = "TC5">
<testcase_pre testpre = "resPower = F,resSensor = F"/>
<testcase_con testcon = "AND"/>
<testcase_res
   testresult = "savePowerandSensorinDB=F"/>
</testcaseline>
<testcaseline no = "TC6">
<testcase_pre testpre = "resPower = F,resSensor = T"/>
<testcase_con testcon = "AND"/>
<testcase_res
   testresult = "savePowerandSensorinDB=F"/>
</testcaseline>
<testcaseline no = "TC7">
<testcase_pre testpre = "resPower = T,resSensor = F"/>
<testcase_con testcon = "AND"/>
<testcase_res testresult = "savePowerandSensorinDB=F"/>
</testcaseline>
<testcaseline no = "TC8">
<testcase_pre testpre = "resPower = T,resSensor = T"/>
<testcase_con testcon = "AND"/>
<testcase_res
   testresult = "savePowerandSensorinDB = T"/>
</testcaseline>
<testcaseline no = "TC9">
<testcase_pre testpre = "savePowerandSensorinDB=F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "sendPowerandSensor = F"/>
</testcaseline>
<testcaseline no = "TC10">
<testcase_pre testpre = "savePowerandSensorinDB = T"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "sendPowerandSensor = T"/>
</testcaseline>
<testcaseline no = "TC11">
<testcase_pre testpre = "sendPowerandSensor = F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "recvSuccess = F"/>
</testcaseline>
<testcaseline no = "TC12">
<testcase_pre testpre = "sendPowerandSensor = T"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "recvSuccess = T"/>
</testcaseline>
</testcase:TestCaseModel>
```

5 Conclusion

In software industrial fields, it will need to increase reliability of software. Specially, the photovoltaic energy integrated monitoring system is very complex one, which may

no	testpre	testcon	testresult
TC1	reqPowerandSensor=F	NORMAL	resPower=F
TC2	reqPowerandSensor=T	NORMAL	resPower=T
TC3	reqPowerandSensor=F	NORMAL	resSensor=F
TC4	reqPowerandSensor=T	NORMAL	resSensor=T
TC5	resPower=F,resSensor=F	AND	savePowerandSensorinDB=F
TC6	resPower=F,resSensor=T	AND	savePowerandSensorinDB=F
TC7	resPower=T,resSensor=F	AND	savePowerandSensorinDB=F
TC8	resPower=T,resSensor=T	AND	savePowerandSensorinDB=T
TC9	savePowerandSensorinDB=F	NORMAL	sendPowerandSensor=F
TC10	savePowerandSensorinDB=T	NORMAL	sendPowerandSensor=T
TC11	sendPowerandSensor=F	NORMAL	recvSuccess=F
TC12	sendPowerandSensor=T	NORMAL	recvSuccess=T

Fig. 29 An image of test case based on microsoft excel file for a normal power generation

Table 24 XMI code transformation of microsoft excel file for a backup function of power generation at disconnection of M-PVMS server

```
<?XMI version = "1.0" encoding = "ISO-8859-1"?>
<testcase:TestCaseModel xmi:version = "2.0"
   XMIns:xmi = "http://www.omg.org/XMI"
   XMIns:testcase = "http://testcase/1.0″>
<testcaseline no = "TC1">
<testcase_pre testpre = "reqPowerandSensor = F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "resPower = F"/>
</testcaseline>
<testcaseline no = "TC2">
<testcase_pre testpre = "reqPowerandSensor = T"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "resPower = T"/>
</testcaseline>
<testcaseline no = "TC3">
<testcase_pre testpre = "reqPowerandSensor = F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "resSensor = F"/>
</testcaseline>
<testcaseline no = "TC4">
<testcase_pre testpre = "reqPowerandSensor = T"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "resSensor = T"/>
</testcaseline>
<testcaseline no = "TC5">
<testcase_pre testpre = "resPower = F,resSensor = F"/>
<testcase_con testcon = "AND"/>
<testcase_res
   testresult = "savePowerandSensorinDB=F"/>
</testcaseline>
<testcaseline no = "TC6">
<testcase_pre testpre = "resPower = F,resSensor = T"/>
<testcase_con testcon = "AND"/>
<testcase_res
   testresult = "savePowerandSensorinDB=F"/>
</testcaseline>
<testcaseline no = "TC7">
<testcase_pre testpre = "resPower = T,resSensor = F"/>
<testcase_con testcon = "AND"/>
<testcase_res
   testresult = "savePowerandSensorinDB=F"/>
</testcaseline>
<testcaseline no = "TC8">
<testcase_pre testpre = "resPower = T,resSensor = T"/>
<testcase_con testcon = "AND"/>
<testcase_res
   testresult = "savePowerandSensorinDB = T"/>
</testcaseline>
<testcaseline no = "TC9">
<testcase_pre testpre = "savePowerandSensorinDB=F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "sendPowerandSensor = F"/>
</testcaseline>
<testcaseline no = "TC10">
<testcase_pre testpre = "savePowerandSensorinDB = T"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "sendPowerandSensor = T"/>
</testcaseline>
<testcaseline no = "TC11">
<testcase_pre testpre = "sendPowerandSensor = F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "retAccessFailureError = F"/>
</testcaseline>
<testcaseline no = "TC12">
<testcase_pre testpre = "sendPowerandSensor = T"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "retAccessFailureError = T"/>
</testcaseline>
<testcaseline no = "TC13">
<testcase_pre testpre = "retAccessFailureError = F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "saveLastSendData = F"/>
</testcaseline>
<testcaseline no = "TC14">
<testcase_pre testpre = "retAccessFailureError = T"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "saveLastSendData = T"/>
</testcaseline>
</testcase:TestCaseModel>
```

no	testpre	testcon	testresult
TC1	reqPowerandSensor=F	NORMAL	resPower=F
TC2	reqPowerandSensor=T	NORMAL	resPower=T
TC3	reqPowerandSensor=F	NORMAL	resSensor=F
TC4	reqPowerandSensor=T	NORMAL	resSensor=T
TC5	resPower=F,resSensor=F	AND	savePowerandSensorinDB=F
TC6	resPower=F,resSensor=T	AND	savePowerandSensorinDB=F
TC7	resPower=T,resSensor=F	AND	savePowerandSensorinDB=F
TC8	resPower=T,resSensor=T	AND	savePowerandSensorinDB=T
TC9	savePowerandSensorinDB=F	NORMAL	sendPowerandSensor=F
TC10	savePowerandSensorinDB=T	NORMAL	sendPowerandSensor=T
TC11	sendPowerandSensor=F	NORMAL	retAccessFailureError=F
TC12	sendPowerandSensor=T	NORMAL	retAccessFailureError=T
TC13	retAccessFailureError=F	NORMAL	saveLastSendData=F
TC14	retAccessFailureError=T	NORMAL	saveLastSendData=T

Fig. 30 An image of test case based on microsoft excel file for a backup function of a power generation at disconnection of M-PVMS server

Table 25 XMI code transformation of microsoft excel file for a backup function of power generation at an error of the inverter

```
<?XMI version = "1.0" encoding = "ISO-8859-1"?>
<testcase:TestCaseModel xmi:version = "2.0"
    XMIns:xmi = "http://www.omg.org/XMI"
    XMIns:testcase = "http://testcase/1.0">
<testcaseline no = "TC1">
<testcase_pre testpre = "reqPowerandSensor = F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "resPower = F"/>
</testcaseline>
<testcaseline no = "TC2">
<testcase_pre testpre = "reqPowerandSensor = T"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "resPower = T"/>
</testcaseline>
<testcaseline no = "TC3">
<testcase_pre testpre = "reqPowerandSensor = F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "resSensor = F"/>
</testcaseline>
<testcaseline no = "TC4">
<testcase_pre testpre = "reqPowerandSensor = T"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "resSensor = T"/>
</testcaseline>
<testcaseline no = "TC5">
<testcase_pre testpre = "reqPowerandSensor = F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "resError = F"/>
</testcaseline>
<testcaseline no = "TC6">
<testcase_pre testpre = "reqPowerandSensor = T"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "resError = T"/>
</testcaseline>
<testcaseline no = "TC7">
<testcase_pre
    testpre = "resPower = F,resSensor = F,resError = F"/>
<testcase_con testcon = "AND"/>
<testcase_res testresult = "savePowerandSensorinDB=F"/>
</testcaseline>
<testcaseline no = "TC8">
<testcase_pre
    testpre = "resPower = F,resSensor = F,resError = T"/>
<testcase_con testcon = "AND"/>
<testcase_res testresult = "savePowerandSensorinDB=F"/>
</testcaseline>
<testcaseline no = "TC9">
<testcase_pre
    testpre = "resPower = F,resSensor = T,resError = F"/>
<testcase_con testcon = "AND"/>
<testcase_res testresult = "savePowerandSensorinDB=F"/>
</testcaseline>
<testcaseline no = "TC10">
<testcase_pre
    testpre = "resPower = F,resSensor = T,resError = T"/>
<testcase_con testcon = "AND"/>
<testcase_res testresult = "savePowerandSensorinDB=F"/>
</testcaseline>
<testcaseline no = "TC11">
<testcase_pre testpre = "resPower = T,resSensor = F,resError = F"/>
<testcase_con testcon = "AND"/>
<testcase_res testresult = "savePowerandSensorinDB=F"/>
</testcaseline>
<testcaseline no = "TC12">
<testcase_pre testpre = "resPower = T,resSensor = F,resError = T"/>
<testcase_con testcon = "AND"/>
<testcase_res testresult = "savePowerandSensorinDB=F"/>
</testcaseline>
<testcaseline no = "TC13">
<testcase_pre testpre = "resPower = T,resSensor = T,resError = F"/>
<testcase_con testcon = "AND"/>
<testcase_res testresult = "savePowerandSensorinDB=F"/>
</testcaseline>
<testcaseline no = "TC14">
<testcase_pre testpre = "resPower = T,resSensor = T,resError = T"/>
<testcase_con testcon = "AND"/>
<testcase_res testresult = "savePowerandSensorinDB = T"/>
</testcaseline>
<testcaseline no = "TC15">
<testcase_pre testpre = "savePowerandSensorinDB=F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "sendPowerandSensor = F"/>
</testcaseline>
<testcaseline no = "TC16">
<testcase_pre testpre = "savePowerandSensorinDB = T"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "sendPowerandSensor = T"/>
</testcaseline>
<testcaseline no = "TC17">
<testcase_pre testpre = "sendPowerandSensor = F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "sendErrorMail = F"/>
</testcaseline>
<testcaseline no = "TC18">
<testcase_pre testpre = "sendPowerandSensor = T"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "sendErrorMail = T"/>
</testcaseline>
<testcaseline no = "TC19">
<testcase_pre testpre = "sendErrorMail = F"/>
<testcase_con testcon = "NORMAL"/>
<testcase_res testresult = "recvSuccess = F"/>
</testcaseline>
<testcaseline no = "TC20">
<testcase_pre testpre = "sendErrorMail = T"/><testcase_con
    testcon = "NORMAL"/>
<testcase_res testresult = "recvSuccess = T"/>
</testcaseline>
</testcase:TestCaseModel>
```

no	testpre	testcon	testresult
TC1	reqPowerandSensor=F	NORMAL	resPower=F
TC2	reqPowerandSensor=T	NORMAL	resPower=T
TC3	reqPowerandSensor=F	NORMAL	resSensor=F
TC4	reqPowerandSensor=T	NORMAL	resSensor=T
TC5	reqPowerandSensor=F	NORMAL	resError=F
TC6	reqPowerandSensor=T	NORMAL	resError=T
TC7	resPower=F,resSensor=F,resError=F	AND	savePowerandSensorinDB=F
TC8	resPower=F,resSensor=F,resError=T	AND	savePowerandSensorinDB=F
TC9	resPower=F,resSensor=T,resError=F	AND	savePowerandSensorinDB=F
TC10	resPower=F,resSensor=T,resError=T	AND	savePowerandSensorinDB=F
TC11	resPower=T,resSensor=F,resError=F	AND	savePowerandSensorinDB=F
TC12	resPower=T,resSensor=F,resError=T	AND	savePowerandSensorinDB=F
TC13	resPower=T,resSensor=T,resError=F	AND	savePowerandSensorinDB=F
TC14	resPower=T,resSensor=T,resError=T	AND	savePowerandSensorinDB=T
TC15	savePowerandSensorinDB=F	NORMAL	sendPowerandSensor=F
TC16	savePowerandSensorinDB=T	NORMAL	sendPowerandSensor=T
TC17	sendPowerandSensor=F	NORMAL	sendErrorMail=F
TC18	sendPowerandSensor=T	NORMAL	sendErrorMail=T
TC19	sendErrorMail=F	NORMAL	recvSuccess=F
TC20	sendErrorMail=T	NORMAL	recvSuccess=T

Fig. 31 An image of test case based on microsoft excel file for a power generation at an error of the inverter

Table 26 Test result of a display function for power generation

No	Input	Condition	Output	Test result
TC1	reqRealtimePower = F	Normal	reqCurrentPower = F	Pass
TC2	reqRealtimePower = T	Normal	reqCurrentPower = T	Pass
TC3	reqCurrentPower = F	Normal	checkPower = F	Pass
TC4	reqCurrentPower = T	Normal	checkPower = T	Pass
TC5	checkPower = F	Normal	resPowerData = F	Pass
TC6	checkPower = T	Normal	resPowerData = T	Pass
TC7	resPowerData = F	Normal	saveXML = F	Pass
TC8	resPowerData = T	Normal	saveXML = T	Pass
TC9	resPowerData = F	Normal	saveDB=F	Pass
TC10	resPowerData = T	Normal	saveDB = T	Pass
TC11	saveXML = F,saveDB=F	And	printRealTimePower = F	Pass
TC12	saveXML = F,saveDB = T	And	printRealTimePower = F	Pass
TC13	saveXML = T,saveDB=F	And	printRealTimePower = F	Pass
TC14	saveXML = T,saveDB = T	And	printRealTimePower = T	Pass

Table 27 Test result of a display function for power generation at disconnection of M-PVMS client

No	Input	Condition	Output	Test result
TC1	reqRealTimePower = F	Normal	reqCurrentPower = F	Pass
TC2	reqRealTimePower = T	Normal	reqCurrentPower = T	Pass
TC3	reqCurrentPower = F	Normal	checkPower = F	Pass
TC4	reqCurrentPower = T	Normal	checkPower = T	Pass
TC5	checkPower = F	Normal	retDisconnectionError = F	Pass
TC6	checkPower = T	Normal	retDisconnectionError = T	Pass
TC7	retDisconnectionError = F	Normal	retAccessFailureError = F	Pass
TC8	retDisconnectionError = T	Normal	retAccessFailureError = T	Pass
TC9	retAccessFailureError = F	Normal	sendErrorMail = F	Pass
TC10	retAccessFailureError = T	Normal	sendErrorMail = T	Failure
TC11	sendErrorMail = F	Normal	printErrorMessage = F	Pass
TC12	sendErrorMail = T	Normal	printErrorMessage = T	Failure

Table 28 Test result of a backup function for a normal power generation

No	Input	Condition	Output	Test result
TC1	reqPowerandSensor = F	Normal	resPower = F	Pass
TC2	reqPowerandSensor = T	Normal	resPower = T	Pass
TC3	reqPowerandSensor = F	Normal	resSensor = F	Pass
TC4	reqPowerandSensor = T	Normal	resSensor = T	Pass
TC5	resPower = F,resSensor = F	AND	savePowerandSensorinDB=F	Pass
TC6	resPower = F,resSensor = T	AND	savePowerandSensorinDB=F	Pass
TC7	resPower = T,resSensor = F	AND	savePowerandSensorinDB=F	Pass
TC8	resPower = T,resSensor = T	AND	savePowerandSensorinDB = T	Pass
TC9	savePowerandSensorinDB=F	Normal	sendPowerandSensor = F	Pass
TC10	savePowerandSensorinDB = T	Normal	sendPowerandSensor = T	Pass
TC11	sendPowerandSensor = F	Normal	recvSuccess = F	Pass
TC12	sendPowerandSensor = T	Normal	recvSuccess = T	Pass

Table 29 Test result of backup function for a power generation at disconnection of M-PVMS server

No	Input	Condition	Output	Test result
TC1	reqPowerandSensor = F	Normal	resPower = F	Pass
TC2	reqPowerandSensor = T	Normal	resPower = T	Pass
TC3	reqPowerandSensor = F	Normal	resSensor = F	Pass
TC4	reqPowerandSensor = T	Normal	resSensor = T	Pass
TC5	resPower = F,resSensor = F	AND	savePowerandSensorinDB=F	Pass
TC6	resPower = F,resSensor = T	AND	savePowerandSensorinDB=F	Pass
TC7	resPower = F,resSensor = F	AND	savePowerandSensorinDB=F	Pass
TC8	resPower = F,resSensor = T	AND	savePowerandSensorinDB = T	Pass
TC9	savePowerandSensorinDB=F	Normal	sendPowerandSensor = F	Pass
TC10	savePowerandSensorinDB = T	Normal	sendPowerandSensor = T	Pass
TC11	sendPowerandSensor = F	Normal	retAccessFailureError = F	Pass
TC12	sendPowerandSensor = T	Normal	retAccessFailureError = T	Pass
TC13	retAccessFailureError = F	Normal	saveLastSendData = F	Pass
TC14	retAccessFailureError = T	Normal	saveLastSendData = T	Pass

Table 30 Test result of power generation at an error of the inverter

No	Input	Condition	Output	Test result
TC1	reqPowerandSensor = F	Normal	resPower = F	Pass
TC2	reqPowerandSensor = T	Normal	resPower = T	Pass
TC3	reqPowerandSensor = F	Normal	resSensor = F	Pass
TC4	reqPowerandSensor = T	Normal	resSensor = T	Pass
TC5	reqPowerandSensor = F	Normal	resError = F	Pass
TC6	reqPowerandSensor = T	Normal	resError = T	Pass
TC7	resPower = F,resSensor = F,resError = F	AND	savePowerandSensorinDB=F	Pass
TC8	resPower = F,resSensor = F,resError = T	AND	savePowerandSensorinDB=F	Pass
TC9	resPower = F,resSensor = T,resError = F	AND	savePowerandSensorinDB=F	Pass
TC10	resPower = F,resSensor = T,resError = T	AND	savePowerandSensorinDB=F	Pass
TC11	resPower = T,resSensor = F,resError = F	AND	savePowerandSensorinDB=F	Pass
TC12	resPower = T,resSensor = F,resError = T	AND	savePowerandSensorinDB=F	Pass
TC13	resPower = T,resSensor = T,resError = F	AND	savePowerandSensorinDB=F	Pass
TC14	resPower = T,resSensor = T,resError = T	AND	savePowerandSensorinDB = T	Pass
TC15	savePowerandSensorinDB=F	Normal	sendPowerandSeosnr = F	Pass
TC16	savePowerandSensorinDB = T	Normal	sendPowerandSeosnr = T	Pass
TC17	sendPowerandSensor = F	Normal	sendErrorMail = F	Pass
TC18	sendPowerandSensor = T	Normal	sendErrorMail = T	Failure
TC19	sendErrorMail = F	Normal	recvSuccess = F	Pass
TC20	sendErrorMail = T	Normal	recvSuccess = T	Failure

have a serious problem on occurring a little error. After constructing the complete solar power monitoring system, this is, a kind of safety-critical system, it should consider how to completely validate it based on requirements of this system. For code based test coverage of the system in safety critical area, there are very difficult to achieve 100% of Path and MC/DC coverage on dynamic analysis. We propose to validate a safety of the system with an automatically extracted test cases based on model driven test case generation with multimedia processing. More important idea is to generate easy-to-read, easy-to-understand images (of test case, decision table, cause-effect diagram) with an image (of sequence diagram) based on automatic multimedia image translation processing, but not use an algorithmic method in the old fashion. To validate the dynamic behaviors of the safety system, we can automatically generate test cases through designing the system behaviors of the solar power monitoring system. With these test cases, we can expect to validate the safety of the system.

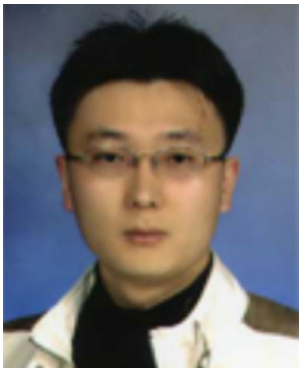
Acknowledgements This work was supported by the Human Resource Training Program for Regional Innovation and Creativity through the Ministry of Education and National Research Foundation of Korea (NRF-2015H1C1A1035548), and this research was supported by a grant (17CTAP-C133299-01) from Technology Advancement Research Program funded by Ministry of Land, Infrastructure and Transport of Korean government.

References

1. Chae HS (2014) Model-based test-concepts and issues. Korean Institute of Information Scientists and Engineers 32(4):59–71
2. Duygulu P, Barnard K, de Freitas JFG, Forsyth DA (2002) Object recognition as machine translation: learning a lexicon for a fixed image vocabulary. Computer Vision — ECCV 2353: 97–112
3. Fekih H, Ayed LJB, Merz S (2006) Transformation of B specifications into UML class diagram and state machines. 21th Annual ACM Symposium on Applied Computing - SAC 2:1840–1844
4. Jang WS, Son HS, BoKyung P, Kim RYC (2016) Implementation of effective web integrated monitoring system for small renewable energy business industries. Korean Institute of Smart Media 5(1):303–305
5. Kim RYC, Joo B-G, Kim K-C, Joen B-K (2003) Scenario Based Testing & Test Plan Metrics Based on A User Case Approach for Real Time UPS. The 2003 International Conference on Computational Science and Its Applications. pp 646–655
6. Kim Y, Son H, Kim W, Seo J, Kim D, Seo Y, Dongkuk R, Kim RY (2007) A study on modeling the obstacle avoidance UGV using extended executable UML. Joint Workshop on Software Engineering Technology 5(1):108–116
7. Kim WY, Son HS, RYC Kim (2011) A Study on Test Case Generation Based on State Diagram in Modeling and Simulation Environment. Advanced Communication and Networking Springer, CCIS199. pp 298–305
8. Mogyorodi GE (2005) B. Math. Requirements-Based Testing-Cause-Effect Graphing. Software Testing Services
9. NIPA (2014) Software Engineering White Book. National IT Industry Promotion Agency, 2014
10. Son HS, Kim RYC (2016) Modeling a Photovoltaic Monitoring System based on Maintenance Perspective for New & Renewable Energy. The 2nd International Joint Conference on Convergence, AACL 07. pp 144–147
11. Woo S, Hyunseung S, Kim RY (2012) A study on extending message-sequence diagram for mapping cause-effect diagram. Korea Information Processing Society 19(1):1251–1254
12. Woo S, Son H, Kim W, Jaeseung K, Kim RY (2012) A study testcase extraction based M&S for pre-testing. Korea Conference on Software Engineering 14(1):181–183

Woo Sung Jang received the M.S. degrees Computer Software Engineering from Hongik University, Sejong, Korea, in 2011. He is currently a Ph. D. candidate in Hongik University. His current research interests include embedded software testing and metamodel modeling and common criteria.

Byung Kook Jeon received his B.S., M.S. and Ph.D. degrees in Computer Science from Kwangwoon University in 1985, 1991, and 2000, respectively. He is a professor in Gangneung-Wonju Nat'l University in Korea. His research interests include Geofence, LBS, BigData, Cloud computing, Smart agents, Sensor networks.



Hyun Seung Son received the B.S., M.S., and Ph.D degree in Software Engineering from Hongik University, Korea in 2015. His research interests are in the areas of Automation Tool Development in Embedded Software, Testing, Metamodel design, and Model Transformation, Model Verification & Validation Method.

R. Young Chul Kim received the B.S. degree in Computer Science from Hongik University, Korea in 1985, and the Ph.D. degree in Software Engineering from the department of Computer Science, Illinois Institute of Technology (IIT), USA in 2000. He is currently a professor in Hongik University. His research interests are in the areas of Simplified Test Maturity Model, Embedded Software Development Methodology, Model Based Testing, Metamodel, Business Process Model and User Behavior Analysis Methodology.