Design and Implementation of a Framework for Automatically Generating Control and Monitoring Software

Dae-Sung Yoo, Min-Suck Sim, Sung-Ghue Park, Jong-Hwan Kim, Myeong-Jae Yi

School of Computer Engineering and Information Technology, University of Ulsan, Ulsan, Korea
(Tel: +82-52-2592218; E-mail: ymj@mail.ulsan.ac.kr)

Abstract: In this paper, we present a framework that is easy to develop, modify, maintain and extend a control and monitoring software for any kinds of instruments. The presented framework is composed of three XML documents (IID, MAP, and CMIML) and two tools (Virtual Instrument Wizard, Generator). Interface information about behaviors and states of instruments is written on IID. Mapping information between the interface information in IID and API of a real instrument driver is written on MAP. Finally information about control and monitoring software is written on CMIML. IID, MAP and CMIML are written with XML format to provide common usage and platform independence of the suggested framework. VI Wizard generates CMIML (platform independent intermediate document) using IID and existing CMIML, and Generator generates source code of a control and monitoring software (platform dependent code) automatically using CMIML and MAP. The suggested framework that automatically generates control and monitoring software based on GUI provides easy development and maintenance. Also, reusability can be increased by reusing platform independent software description documents.

Keywords: Control and Monitoring Software, Automatically Generating, XML, IID, MAP, CMIML, VI Wizard, Generator

1. INTRODUCTION

Instruments are controlled by PC based systems to control complex automation systems. So importance of control and monitoring software for automation instruments is increasing.

There are some problems in software development for controlling and monitoring many automation instruments that are connected through various networks. Because driver APIs offered by different manufacturers are different, many software should be developed to control instruments that form complex automation systems. In addition, software should be redeveloped whenever driver API is changed or new requirements are occurred. And various platforms and communication methods were used in automatic instruments. Therefore different software should be developed according to the various platforms and communication methods

Due to these problems, additional effort and cost is required on the development and maintenance of control and monitoring software.

The suggested framework minimizes dependency between instrument and software by writing XML documents that describe instrument as well as software.

Besides, reusability can be increased by reusing platform independent software description document and not reusing platform dependent software code. Also, development and maintenance cost of software can be reduced.

2. RELATED WORK

Instruments which were used in field of production were expensive and maintenance cost is very high. Therefore industry companies were interest in automation of instruments for cost-cutting and various studies about implementation of automation instruments or protocol which connect instruments have been done.

A software such as labVIEW[1] and rsVIEW[2] was developed already. These tools are used in control and monitoring for instruments and offer a VI (Virtual Instrument) editing environment based on GUI. Therefore easy development and maintenance are available by using these tools.

Other studies such as EDDL (Electronic Device Description Language)[3] and GSD (Generic Station Description)[4] of Profibus and EDS (Electronic Data Sheet)[5] of CANOpen about describing instruments and communication protocol has been done.

Also, studies such as CoML(CANOpen Markup Language)[6] of CANOpen and IML(Instrument Markup Langauge)[7] of NASA about describing instruments by using XML for easy and common usage have been done.

The suggested framework supports easy and fast development and maintenance by providing a VI editing environment such as labVIEW and rsVIEW. And three XML based documents (IID, MAP, and CMIML) are written to solve dependency problem about instruments or API in suggested framework.

3. FRAMEWORK FOR AUTOMATICALLY GENERATING CONTROL AND MONITORING SOFTWARE

Fig. 1 shows the suggested framework architecture which consists of three XML DTDs (IID, MAP, CMIML), VI Wizard and Generator.

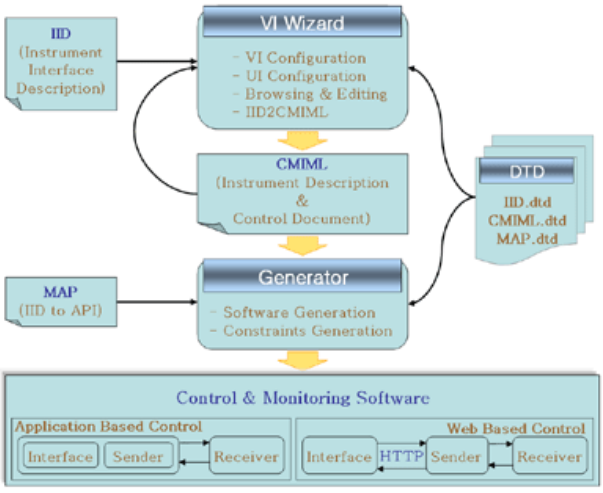

Fig. 1 CMIML Framework Architecture

The VI Wizard includes two XML DTD (IID, CMIML) and constructs VI (Virtual Instrument) and UI(User Interface) based on GUI using IID and existing CMIML. Using new VI and UI, VI Wizard generates a CMIML (Control & Monitoring Instrument Markup Language). Generator automatically generates control and monitoring software sing CMIML and MAP.

4. THREE XML DOCUMENTS – IID, MAP, CMIML

Three XML DTDs (IID, MAP, and CMIML) which are used in automatically generation of control and monitoring software were showed in our past study [8]. In this paper, we will show examples of three xml documents.

4.1 IID (Instrument Interface Description)

Instrument developer defines behavior and state information of instrument and writes IID document. IID should be a valid XML document and describe interface information (behavior, state) for instrument control

```
<Instrument Desc="Simple Robot made by RCX" Name="RCXRobot">
<MethodList>
<Method Name="backward" RetType="void" Desc="Causes robot to rotate backwards">
<ParamList>
<Param Desc="A period value">
<Name>aPeriod</Name>
<Type>int</Type>
</Param>
</ParamList>
</Method>
</MethodList>
</Instrument>
```

4.2. MAP (IID to API)

To connect with API of instrument with interface information (behavior, state), driver developer develops API and writes a MAP file which is a valid XML document.

```
<Instrument Name="Robot" Desc="Motor A,B,C">
  <MappingList>
    <CommandMap>
      <CommandReferenceMap
        CommandReference="backward"
        MapTo="RCXRobot.backward" />
    </CommandMap>
  </MappingList>
</Instrument>
```

4.3. CMIML (Control & Monitoring Instrument Markup Language)

CMIML which is generated by VI Wizard includes interface information (which is described) in IID and describes user interface of control and monitoring software, monitoring information of instrument, communication method between instrument and PC, schedule information of instrument.

```
<Instrument Name="Robot" Desc="RCXRobot">
  <Port Function="command" Name="Robot"
    Number="10001" Type="ascii" Hostname="New
    Port HostName">
    <Command Name="Robot.forward"
      RetType="void" Val="" Desc="Causes robot to
      rotate forward" UI="True" UI_ID="10">
    </Command>
  </Port>
  <ScheduleList>
    <Schedule Name="Robot" UI="True" I_IDS="1"
      UI_IDE="1">
      <Command Name="Robot.backward"
        RetType="void" Val="" Desc="Causes robot
        to rotate forward" UI="False" UI_ID="0">
      </Command>
    </Schedule>
  </ScheduleList>
</Instrument>
```

5. TWO TOOLS – VI WIZARD, GENERATOR

5.1. VI Wizard

VI Wizard supports editing environment based on GUI using IID file that describe interface information of instrument and generates CMIML from newly constructed VI. Generated CMIML is used to generate control software or can be reused for constructing other VI.

Fig. 2 presents VI Wizard system architecture. VI Wizard is consisting of four modules, three managers and two data pools.

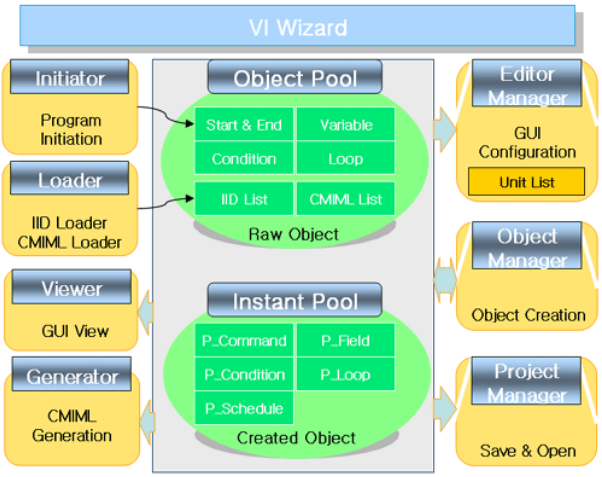

Fig. 2 VI Wizard System Architecture

Four modules are following:

・Initiator Module

Initiator Module performs initialization work at run time. This module registers Start, End, Variable, Condition and Loop elements to Object Pool and adds each element to Control group (see Fig. 4's (a)).

・Loader Module

Loader Module is consisted of IID Loader and CMIML Loader. IID Loader reads IID File and extracts Field and Method elements, then registers elements to IID_List in Object Pool and adds a new group which name is IID name to Tool Box in Fig. 4's (a) and adds method elements to newly added group. CMIML Loader reads CMIML file and extracts Schedule elements, then registers elements to CMIML_List in Object Pool and adds Schedule elements to User Custom menu in Fig. 4's (a).

・Viewer Module

Viewer Module previews user interface about control and monitoring software.

・Generator Module

Generator Module extracts data from Object Pool and Instant Pool, and then generates CMIML.

Three managers are following:

・Editor Manager

Editor Manager is responsible for constructing VI in GUI editing environment. It creates a new object at Instant Pool or removes a object in Instant Pool by help of Object Manager.

・Object Manager

Object Manager creates an instance of GUI object by using objects in Object Pool at the request of Editor Manager, and then adds a instance at Instant Pool or deletes a existing object in Instant Pool.

・Project Manager

Project Manager is responsible for management of project. It saves or loads working environment of project.

Two Data Pools are following:

・Object Pool

Object Pool stores object which is added by Initiator and Loader Module and offers necessary information when Editor Manager constructs VI.

・Instant Pool

Instant Pool has information of GUI Objects which is constructing VI. It stores instance of object when object in Object Pool becomes instance by Object Manager.

5.2. Generator

Generator generates software code automatically from CMIML which is generated by VI Wizard. Through Generator, control software can be generated without expert knowledge.

Fig. 3 presents Generator System Architecture. Generator is consisting of three modules and one data pool.

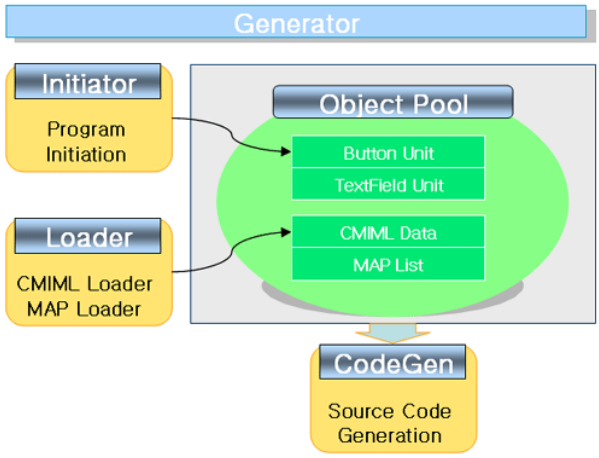

Fig. 3 Generator System Architecture

Three modules and one data pool are following:

・Initiator Module

Initiator Module performs initialization work at run time. This module is executed whenever Generator runs first time and new CMIML is loaded because new CMIML is loaded, existing data should be initialized.

・Loader Module

Loader Module is consisted of CMIML Loader and MAP Loader. Generator loads one CMIML document and one or more MAP documents. CMIML Loader reads CMIML document and extracts necessary elements in generating control and monitoring software. Extracted elements are saved CMIML Data in Object Pool. MAP Loader reads MAP documents and extracts link information between information which is described at CMIML and API. Extracted information is saved MAP List in Object Pool.

・CodeGen Module

CodeGen Module generates control and monitoring software code which is suitable any platform using data which is saved in Object Pool. CodeGen Module was implemented to generate JAVA code. However source code which is suitable various platforms can be generated by development of various CodeGen Module.

・Object Pool

Object Pool stores objects and data that are added by Initiator Module and Loader Module. Saved objects and data are used in generating control software code by CodeGen

6. IMPLEMENTATION OF VI WIZARD AND GENERATOR

VI Wizard and Generator were implemented using Visual Basic 6.0 and MSXML 3.0. Fig. 4 and Fig. 5 present generating control code which control RCXRobot of LEGO mindstroms[]. Major components of RCXRobot are RCXTM Microcomputer, two motors and two touch senses. RCXRobot is run through PC and infrared rays' communication. We loaded leJOS[] which is operating system based on JAVA to RCXTM Microcomputer and use API which is offered by leJOS.

Fig. 4 shows executive screen of VI Wizard. Fig. 4's (a) shows objects which were registered at Object Pool by Initiator and Loader Module(그룹별로). Fig. 4's (b) shows state information of selected objects. Fig. 4's (c), (d) and (e) shows VI which is edited based on GUI by Editor Manager. (d) shows internal action of condition state and (e) shows internal action of loop state and (f) shows CMIML document which is converted from newly constructed VI by VI Wizard.

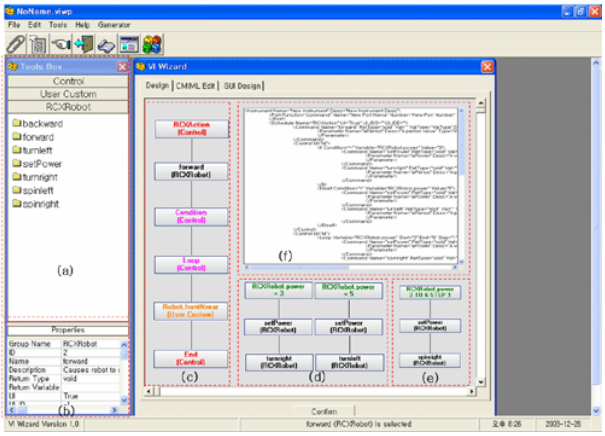

Fig. 4 VI Wizard

Fig. 5 presents executive screen of Generator. Fig. 5's (a) shows control software code which is generated based on CMIML document which is read by Loader Module (CMIML Loader, MAP Loader) and Data of MAP document. Present Generator was implemented to generate source code based on JAVA. Fig. 5's (b) shows result of compile when generated source code is compiled through external compiler.

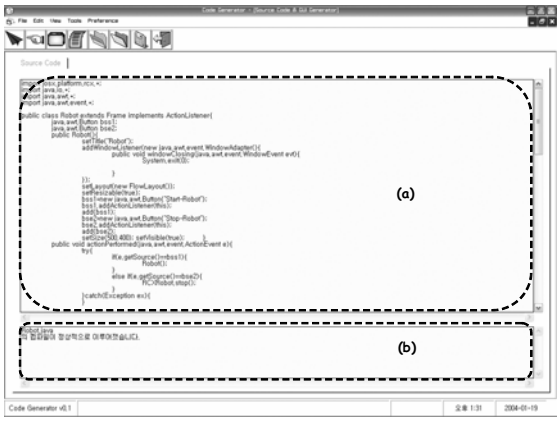

Fig. 5 Generator

7. CONCLUSION AND FUTURE WORKS

In this paper, we suggested a framework that supports easy development and maintenance of control and monitoring software for any kind of instruments in the field of production. The suggested framework is composed of three XML documents (IID, MAP, CMIML), VI Wizard and Generator.

The suggested framework that automatically generates control and monitoring software based on GUI provides easy development and maintenance. Also, the suggested framework minimizes dependency between instrument and software by using XML documents that describe instrument as well as software. Besides, reusability can be increased by reusing platform independent software description document and not reusing platform dependent software code.

The framework presented in this paper is still in progress. We are planning to extend three XML documents to reflect various requirements of control and monitoring software. And we continue to develop and evaluate the suggested framework.

REFERENCES

[1] National Instrument, “www.ni.com”
[2] Rockwell Automation, “www.rockwell.com”
[3] Vol. 2: EDDL Specification; Specification for PROFIBUS Device Description and Device Integration, "www.profibus.com"
[4] Vol. 1: GSD Specification; Specification for PROFIBUS Device Description and Device Integration, "www.profibus.com"
[5] CiA DSP 306 V 1.1: CANopen electronic data sheet (EDS) specification for CANopen, "www.canopen.org/canopen"
[6] Buhler Dieter, "The CANOpen Markup Language Representing Fieldbus Data with XML", Industrial Electronics Society, 2000. IECON 2000. 26th Annual Confjerence of the IEEE , Volume: 4 , 22-28 Oct. 2000 Pages:2449 – 2454
[7] Troy Ames, Lisa Koons, Ken Sall, Craig Warsaw, "Using XML and Java for Astronomical Instrumentation Control", Proc. of 6th Int'l Conf. on Space Operations(SpaceOps 2000), June 2000, France
[8] Dae-Sung yoo, Min-Suck Sim, Sung-ghue Park, Jong-Hwan Kim, Myeong-Jae Yi, "A Framework for automatic generation of instrument control and monitoring software", Proceedings of the 7th Korea-Russia International Symposium, KORUS 2003, vol2. pp, 428-432
[9] LEGO Mindstorms home, "http://mindstorms.lego.com/eng/default.asp"
[10] leJOS home "http://lejos.sourceforge.net/"