JCSE Journal of Computing Science and Engineering

Regular Paper
Journal of Computing Science and Engineering,
Vol. 14, No. 1, March 2020, pp. 19-25

MIORPA: Middleware System for Open-Source Robotic Process Automation

Myeong-Ha Hwang, Ui-Kyun Na, Seungjun Lee, ByungJoo Cho, Yeri Kim, DongHyuk Lee, and JiKang Shin*
Digital Solution Laboratory, Korea Electric Power Research Institute (KEPRI), Daejeon, Korea
{mh.hwang, uikyun.na, seungjoon.lee, byungjoo_cho, yeri.kim, leedh8036, jk.shin}@kepco.co.kr

Abstract
Introduction of robotic process automation (RPA) in simple repetitive task automation initiated its research and development in various fields. The high demand for RPA has expanded the global market. However, the disadvantages include the high cost of commercial RPA products and limited functional expansion. Therefore, it is necessary to design open-source RPA to minimize the cost and manage the execution of multiple RPA jobs. We propose a middleware system called MIORPA to control open-source RPA robots. The proposed middleware system provides a job-scheduling algorithm for assignment of tasks to multiple RPA robots in multiple middleware environments. Further, MIORPA provides watchdog-based RPA robot status monitoring, and the status of the RPA robot can be identified and managed in real time. Therefore, when a large number of users request RPA, the work is distributed and processed efficiently. Thus, this study contributes towards research into the control of RPA robots.

Category: Smart and Intelligent Computing

Keywords: Robotic process automation; Middleware; Job scheduling; Watchdog

I. INTRODUCTION

Robotic process automation (RPA) is a software-based technology that helps automate repetitive tasks based on a pre-defined workflow [1]. RPA helps automate low-value manual tasks and allows people to focus on high-value creative tasks such as identifying differentiated business values. Therefore, many companies have recently adopted RPA to strengthen their competitiveness. In South Korea, various industrial sectors including the financial sector have introduced RPA as well [2].

The high demand for RPA has led to an increase in its global market share and companies specializing in RPA implementation, such as UiPath, Automation Anywhere, and Blue Prism. However, the disadvantages of commercial RPA products include high cost of implementation and limited functional expandability. Therefore, it is necessary to design an open-source RPA to help minimize costs.

Furthermore, the main principle of RPA is to imitate the manipulation of keyboard and mouse by a user. When RPA performs a task, the personal computer does not need to be operated by a user, and a single RPA robot can only perform a single task. Therefore, to efficiently perform task automation using multiple RPA robots, a middleware that schedules and manages the execution of RPA jobs is necessary.

In this study, we propose a middleware system called MIORPA for handling multiple open-source-based RPA

Open Access http://dx.doi.org/10.5626/JCSE.2020.14.1.19 http://jcse.kiise.org

Received 31 January 2020; **Accepted** 05 March 2020
*Corresponding Author

 pISSN: 1976-4677 eISSN: 2093-8020

robots. This system uses a job scheduling algorithm to assign tasks to multiple RPA robots in multiple middleware environments and monitor the status of RPA robot in real time. Therefore, when a large number of users request RPA tasks, the middleware decentralizes and expedites the tasks, which contributes to continued research on RPA robots.

II. RELATED WORK

A. Robotic Process Automation

Companies focus on business process outsourcing (BPO) and enterprise resource planning (ERP) to minimize costs. However, RPA improves the quality of work and job satisfaction as it reduces the burden of repetitive work and facilitates multiple applications [3]. Further, the recent revision in the Labor Standards Act of Korea has led to a decrease in labor time and increased RPA introduction to automate simple repetitive tasks for enhanced work efficiency [2].

RPA can be divided into three automation categories: (1) basic, for rule-based processing; (2) intelligent, for processing information based on accumulated data and machine learning technology; and (3) cognitive, which makes complex decisions based on deep learning and predictive analysis [4]. Current RPA technology is in the basic automation stage. Therefore, middleware is essential for assigning and managing RPA robot tasks. In addition, commercialized RPA products are associated with high implementation costs and limited functional expandability. Therefore, there is need for an open-source RPA that can be used without additional business expenses and a middleware program to handle it.

B. Middleware

Middleware is a software program that serves applications in addition to those served by the operating system [5]. Finkemeyer et al. [6] proposed middleware for robotic and process control applications (MiRPA), which enables real-time communication between publishers and subscribers as well as between clients and servers. Thus, middleware provides a standardized interface and maintains data consistency by processing distributed tasks simultaneously. It also facilitates workload distribution.

Job scheduling—a type of middleware system—is a computer application that controls the execution of waiting jobs [7]. Job scheduling can be categorized into workload, resources, and requirements [8]. When classifying job scheduling under these categories, it is clear that research and development of a scheduling algorithm to control the execution of a large number of RPA robots for various types of jobs in multiple middleware systems is lacking.

A watchdog is a type of middleware that uses electronic timers to detect and repair computer malfunction. In the event of a hardware defect or software program error, the timer generates a timeout signal. This timeout signal then triggers several corrective actions [9]. Dias et al. [10] developed a watchdog for detecting misbehavior nodes at vehicular delay-tolerant networks (VDTNs), while Ma et al. [11] have actively developed a watchdog for collision detection in smart cities. By developing watchdogs that detect the status of multiple RPA robots, it is possible to perform tasks rapidly and efficiently to address the needs of users.

III. MIORPA: MIDDLEWARE SYSTEM FOR OPEN-SOURCE ROBOTIC PROCESS AUTOMATION

A. System Architecture

The RPA system designed by our research team was developed for ERP. Fig. 1 shows the system architecture of MIORPA. The user inputs the necessary information using the web server and requests delegation execution (Fig. 1(a)). The web server transfers the information to the database after receiving the request for delegation execution. It saves the necessary attachments using the security file transfer protocol (SFTP), which is used in MIORPA (Fig. 1(b), 1(c)).

Job scheduling and watchdog functions are implemented in MIORPA to handle multiple RPA robots in multiple middleware environments to efficiently handle the delegation execution requested by multiple users (Fig. 1(d)). During delegation, the status of RPA robots, such as waiting, execution, shutdown, or error, is updated in real time in the database. The updated RPA status is communicated to the user via a web server for the user to monitor the status in real time. The RPA robot was developed using AutoIt [12].

B. Database Construction

The database schema of MIORPA is shown in Fig. 2. The Submit_Result table stores information about jobs requested by users and conveys the status of the job to the users (Fig. 2(a)). It contains parameters such as the number (ID), request time (GEN_TIME), job name (TITLE), middleware job assignment (ALLOT), job classification information (SUBMIT_CD, SUBMIT_CD_NAME), authorization number (APPROVE_NUM), user number and name (USER_EMPNO, USER_NAME), CPU-based job status (STATUS_CD, STATUS_NAME), and RPA robot-based job status (JOB_STATUS_CD, JOB_STATUS_NAME, JOB_STATUS_DETAIL) pertaining to the requested job. The CPU-based job status (STATUS_CD, STATUS_NAME) is used to determine the status of each RPA robot in real time while monitoring the list of CPU processes in each

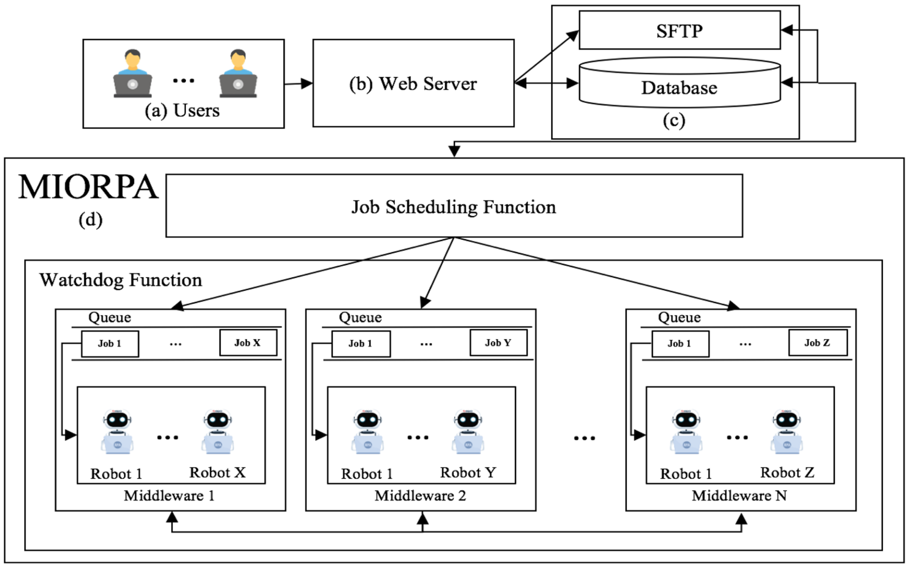

Fig. 1. System architecture of MIORPA.

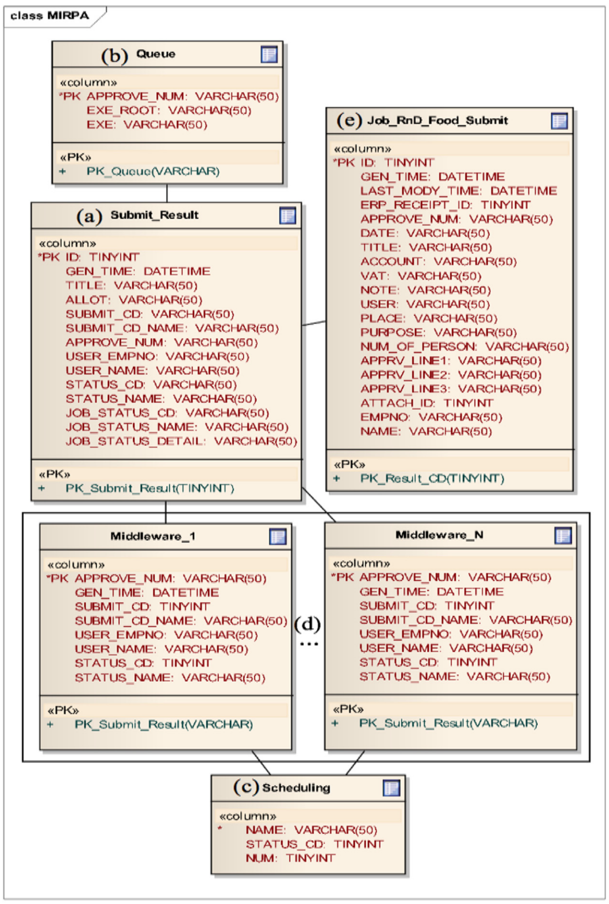

Fig. 2. Schematic diagram of the MIORPA database.

middleware environment. The RPA robot-based job status (JOB_STATUS_CD, JOB_STATUS_NAME, JOB_STATUS_DETAIL) is used to convey the execution status of the RPA robot to the user.

The Queue table stores the executable and root information of the RPA robot (Fig. 2(b)). It consists of the job approval number (APPROVE_NUM), the root where the RPA robot executable is located (EXE_ROOT), and the RPA robot executable filename (EXE).

The Scheduling table assigns jobs requested by the users to several middleware programs (Fig. 2(c)). It consists of the name (NAME) and state (STATUS_CD) of each middleware. In particular, the order of middleware (NUM) is used to assign the corresponding tasks.

As shown in Fig. 2(d), 'N' number of middleware tables were generated. The work requests were distributed to each middleware table according to the MIORPA scheduling algorithm, thereby generating the data in the tables. The middleware table consists of the job approval number (APPROVE_NUM), request time (GEN_TIME), job classification information (SUBMIT_CD, SUBMIT_CD_NAME), user number and name (USER_EMPNO, USER_NAME), and CPU-based job status (STATUS_CD, STATUS_NAME).

Finally, the Job_RnD_Food_Submit table allows each open-source RPA robot to execute ERP tasks (Fig. 2(e)). The table consists of the job number (ID), request time (GEN_TIME), and request change time (LAST_MODY_TIME) as well as the remaining data for the ERP task.

C. Job Scheduling Function

The job scheduling function of MIORPA can be classified into two sub-functions, as shown in Algorithm 1. The first sub-function assigns multiple jobs to each middleware: when the Job Scheduling algorithm is first executed,

Algorithm 1. Job Scheduling

```
procedure
 Connect to database
   ML ← middleware list
   MTn ← n'th table in database of middleware
   MJn ← n'th job of middleware
   NJ ← the number of new jobs
   PTL ← previous waited job list of each table
   CTL ← current waited job list of each table
   while true do
     if ML exist then
       if ML == 1 then
         ML += MJn and new_job
       else
         job_list = MJn, count = max(job_list)
         for i to the_range_of_job_list do
           sub_list.append(count- job_list[i])
         end for
         W = NJ – sum(sub_list)
         for j to the_range_of_ML do
           MTj += NJ(sub_list [j]+(W/len(ML))
         end for
         for x to the_range_of_W % len(ML) do
           MTx += NJ(1)
         end for
       time_run = 0
       while time_run < periodic_time do
         if time_run == 0 then
           PTL = MJn
         end if
         time_run += 10sec
         if time_run == periodic_time then
           CTL = MJn
           for i to the rang of length ML do
             if PTL[i] != 0 then
               if PTL[i] == CTL[i] then
                 new_job += CTL
                 MTn drop in database
               end if
             end if
           end for
         end if
       end while
     end if
end procedure
```

many jobs are allocated by dividing the middleware number by N. The new jobs are entered into the database at regular intervals.

The second sub-function is used to evaluate the remaining jobs in the database table linked to each middleware at regular time intervals. If there is no change in the database table, it labels the middleware as a failure and assigns the jobs to other middleware.

D. Watchdog Function

The MIORPA watchdog function is implemented via three algorithms: Job Load Timer, Waited Job Load, and Watchdog. The Job Load Timer and Waited Job Load algorithms are shown as Algorithms 2 and 3, respectively, which are used to assign newly added jobs to the watchdog queue at specific time intervals.

The Watchdog algorithm is used to handle multiple RPA robots within each middleware environment. It monitors the status and updates the database according to the presence or absence of the corresponding RPA robot executable file from the list of executable files in the CPU process. The states of each RPA robot are defined as

Algorithm 2. Job Load Timer

```
procedure
  if first_run = True then
    time_check = 0
  else
    time_check += 3sec
    if time_check = periodic_time then
      Perform Algorithm 3: Waited job load
      first_run = True
    end if
  end if
end procedure
```

Algorithm 3. Waited Job Load

```
procedure
 Connect to database
  T ← the time of tail value in Queue
  WT ← the time list of waited job in Database
  for i to WT do
    if T < WT then
      Insert the WT into Queue
      Perform Algorithm 4: Watchdog
    end if
  end for
end procedure
```

Algorithm 4. Watchdog

```
procedure
    Connect to database
    Insert job into Queue
        first_run set to True and cnt_sec set to 3sec
        CPU_check ← the executing list in CPU
        for cnt_sec to 0sec do
            if cnt_sec = 0sec then
                1sec sleep
            else
                if not exist CPU_check then
                    if first_run = True then
                        time_check += 3sec
                        Perform Algorithm 2: Waited Job Load
                        if time_check > 10sec then
                            Execute the RPA robot
                            time_check = 0 and first_run = False
                        end if
                    else
                        Dequeue the head value in Queue
                        Update the Robot status to Executing
                        Perform Algorithm 2: Waited Job Load
                    end if
                else
                    if first_run = True then
                        Perform Algorithm 2: Waited Job Load
                    else
                        Update the Robot status to Done
                        Perform Algorithm 2: Waited Job Load
                        first_run = False
                    end if
                end if
            end if
        end for
end procedure
```

waiting (0), running (1), finished (2), and error (3). If an RPA robot terminates a job normally after execution, the next RPA robot waits for 10s before performing the next job to avoid conflicts. The Watchdog algorithm is shown in Algorithm 4.

IV. EXPERIMENTS AND RESULTS

MIORPA experiments were performed using five middleware programs on a Windows 10 Pro operating system with Intel Xeon CPU E3-1226 v3 with a 3.30 GHz Processor and 32 GB RAM. For the Job Scheduling algorithm, we distributed a large amount of jobs to multiple middleware programs. The three methods (FIFO algorithm, the FIFO algorithm with database optimization, and MIORPA) were compared, and the results are illustrated in Fig. 3.

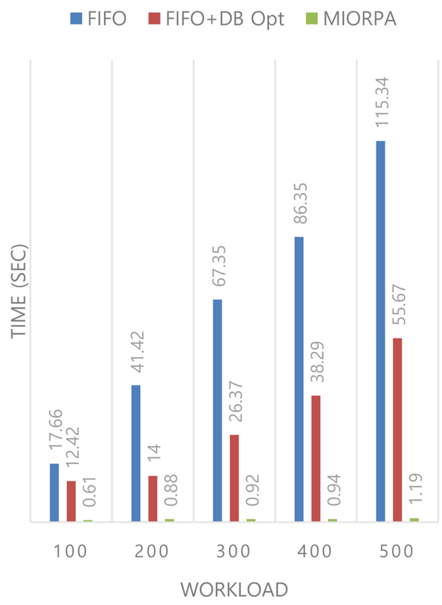

Fig. 3. Comparison of execution time for the scheduling functions.

The FIFO algorithm allocates jobs to the middleware with the smallest number of remaining jobs. However, the disadvantage is that the algorithm is expected to use as many database connections as there are middleware. Therefore, when the FIFO algorithm is used, the job allocation time can be reduced by implementing query optimization to optimize the database connections. However, if the algorithm is utilized for a large enterprise, it is necessary to prepare for large workloads. Application of Algorithm 1 resulted in a significant reduction in the job allocation time compared with the FIFO algorithm and the FIFO + database optimization methods.

An example of watchdog implementation is shown in Fig. 4. When running watchdog for the first time (Fig. 4(a)), the RPA robots executed in the middleware are inserted into the queue, and the RPA robot located in the head is highlighted. The program evaluates whether the RPA robot is running on the CPU of the middleware at 3-second intervals and executes the appropriate RPA robot after at least 10 seconds (Fig. 4(b)). When the RPA robot is running, the database is updated with “Processing” as shown in Fig. 4(c). Once the RPA robot execution is completed, the program executes Dequeue, updates the program with “Completed” and updates the next RPA

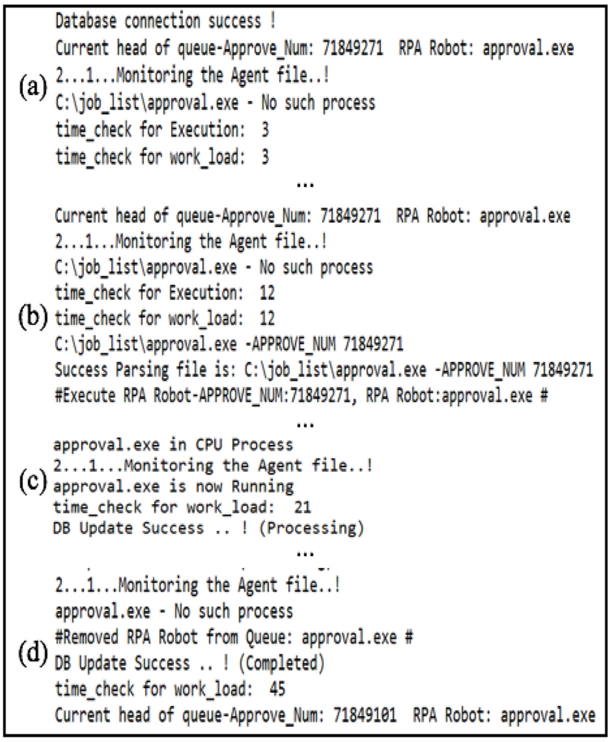

Fig. 4. An example of watchdog function of MIORPA: (a) pre-execution state, (b) starting point of execution state, (c) execution state, and (d) completion state.

robot as the head of the queue (Fig. 4(d)). Thus, the process is completed by repeating the steps shown in Fig. 4. Eventually, new jobs and RPA robots stored in the database are allocated to the queue.

V. CONCLUSION

In this study, we introduced MIORPA, a middleware system for handling a large number of open-source RPA robots. It includes a job-scheduling algorithm that efficiently distributes jobs and a watchdog function that processes jobs using a queue based on the CPU processing status of each middleware program. The results show that MIORPA facilitated efficient processing of ERP jobs by assigning and processing jobs to multiple RPA robots in multiple middleware environments.

In the future, job scheduling will be optimized via deep learning to develop a universal job scheduling algorithm that can be used not only for RPA robots, but also for other applications. Further, we will develop a user-friendly middleware system by investigating the watchdog function for system stabilization using various use-case simulations.

AKNOWLEDGMENTS

This work was funded by the Korea Electric Power Corporation (KEPCO).

REFERENCES

1. W. M. van der Aalst, M. Bichler, and A. Heinzl, “Robotic process automation,” *Business & Information Systems Engineering*, vol. 60, pp. 269-272, 2018.
2. S. Y. Yang and D. W. Park, “Case study on the application of robotic process automation technology to public institutions,” *The Journal of Korean Institute of Communications and Information Sciences*, vol. 43, no. 9, pp. 1517-1524, 2018.
3. S. Aguirre and A. Rodriguez, “Automation of a business process using robotic process automation (RPA): a case study,” in *Applied Computer Sciences in Engineering*. Cham: Springer, 2017, pp. 65-71.
4. L. Willcocks, M. Lacity, and A. Craig, “The IT function and robotic process automation,” London School of Economics and Political Sciences, The Outsourcing Unit Research Paper 15/05, 2015.
5. P. A. Bernstein, “Middleware: a model for distributed system services,” *Communications of the ACM*, vol. 39, no. 2, pp. 86-98, 1996.
6. B. Finkemeyer, T. Kroger, D. Kubus, M. Olschewsk, and F. M. Wahl, “MiRPA: middleware for robotic process control applications,” in *Workshop on Measures and Procedures for the Evaluation of Robot Architectures and Middleware at the IEEE International Conference on Intelligent Robots and Systems*, San Diego, CA, 2007, pp. 76-90.
7. D. G. Feitelson, L. Rudolph, U. Schwiegelshohn, K. C. Sevcik, and P. Wong, “Theory and practice in parallel job scheduling,” in *Job Scheduling Strategies for Parallel Processing*. Heidelberg: Springer, 1997, pp. 1-34.
8. R. V. Lopes and D. Menasce, “A taxonomy of job scheduling on distributed computing systems,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 12, pp. 3412-3428, 2016.
9. N. Murphy and M. Barr, “Watchdog timers,” *Embedded Systems Programming*, vol. 14, no. 11, pp. 79-80, 2001.
10. J. A. Dias, J. J. Rodrigues, F. Xia, and C. X. Mavromoustakis, “A cooperative watchdog system to detect misbehavior nodes in vehicular delay-tolerant networks,” *IEEE Transactions on Industrial Electronics*, vol. 62, no. 12, pp. 7929-7937, 2015.
11. M. Ma, S. M. Preum, and J. A. Stankovic, “Cityguard: a watchdog for safety-aware conflict detection in smart cities,” in *Proceedings of the 2nd International Conference on Internet-of-Things Design and Implementation*, Pittsburg, PA, 2017, pp. 259-270.
12. AutoIt Consulting Ltd., “AutoIt v3,” https://www.autoitscript.com/site/.

Myeong-Ha Hwang

Myeong-Ha Hwang received B.S. degree in Department of Information and Communication Engineering from Chungnam National University (CNU), South Korea in 2015 and M.E. degree in Information and Communication Network Technology from University of Science and Technology (UST), South Korea in 2018, and currently works for Korea Electric Power Research Institute (KEPRI). His research area covers deep learning, natural language processing, and robotic process automation.

Ui-Kyun Na

Ui-Kyun Na is received B.S. degree in Department of Information and Telecommunication Engineering from Incheon National University (INU), South Korea in 2017 and M.E. degree in Information and Telecommunication Engineering from Incheon National University (INU), South Korea in 2019. His current research interests include energy management system, fog/edge computing, and robotic process automation.

Seungjun Lee

Seungjun Lee received B.S. degree in Department of Statistic and Computer Science from Chosun University, South Korea in 2019. His current research interests include robotic process automation and middleware.

ByungJoo Cho

ByungJoo Cho received a bachelor's degree in Department of Computer Engineering and Economy Commerce from Sejong University, South Korea in 2017. He is interested in data science, especially, data analysis. Currently, He is working for Korea Electric Power Corporation (KEPCO) and studying robotic process automation.

Yeri Kim

Yeri Kim received B.S. degree in Department of Information Systems from Sungshin Women's University (SSWU), South Korea in 2019. Her current research interests include web programming, design and construction of database. She manages server and database in the robotic process automation (RPA) project.

DongHyuk Lee

DongHyuk Lee received B.S. degree in Department of Computer Science from Kookmin University (KMU), South Korea in 2018. He is working for Korea Electric Power Corporation (KEPCO). His current research interests include Software QA (quality assurance) and robotic process automation.

JiKang Shin

JiKang Shin received B.S. degree in Department of Computer Science and Electrical Engineering from Handong University, South Korea in 2008, and M.E degree in Information and Communication Engineering from Korea Advanced Institute of Science and Technology (KAIST), South Korea in 2010, and currently works for Korea Electric Power Corporation Research Institute. His research area covers intelligent applications for smart-grid and robotic process automation.