RESEARCHERID
THOMSON REUTERS
[COTII 2019]
ISSN 2348 – 8034
Impact Factor- 5.070

GLOBAL JOURNAL OF ENGINEERING SCIENCE AND RESEARCHES

FLEXIBLE MANIPULATOR SYSTEM FOR RTOS ARCHITECTURE OF KUKA INDUSTRIAL ROBOT WITH ARTIFICIAL INTELLIGENCE

Suyash Srivastava[*1], Beena Rizvi[2], Dr. Alok Mishra[3] & Amit Mishr[a4]

[*1&2]Assistant Professor, Department of Electronics and Communication Engineering, Ambalika Institute of Management and Technology, Lucknow, India

[3]Professor, Department of Applied Science, Ambalika Institute of Management and Technology Lucknow, India

[4]Student, M.TECH (Automation and Robotics), Ambalika Institute of Management and Technology Lucknow, India

ABSTRACT

Industrial robots are characterized as a sophisticated mechanical components and it is used control algorithms. The efficient use of robotics systems is limited by existing programming which makes software development very complex and time-consuming too. The developed capabilities required for operating in industrial work place including features such as reliable and correct navigation, flexible manipulation and vigorous object recognition. The vision of the research project Robot was to easy robotics software development that is reduced development *time, by providing "robotics".*

Keywords: *Artificial Intelligence, KRL programs, sys.config, config.dat*

I. INTRODUCTION

Goal

To develop an object-oriented platform that permit to industrial robots using Java program. Like in traditional robotic languages (e.g. Kuka Robotic Language), the new framework is to provide real-time assurance for controlling robots. Developing robotics software, one should only be aware that real-time certain task steps, but should not be occupy oneself with low-level aspects of real-time programming. This is an important goal to reduce difficulty and allows to offer an application programming interface in plain Java instead of real-time ability programming languages. The framework had to support easy control flow, motion programming, motion blending and the motion trigger actions to achieve the performance. The framework support easy expressiveness of traditional robot programs. By one application controlling multiple robots or devices, as well as sensor-guided motions.

II. SOLUTION

The framework had developed a multi-layered software architecture, provided prototypical reference implementation (prototype-oriented, classless, or instance-based programming).

Robot Control Core which is implemented in C++ using Orocos. The real-time requirements of robot control which allows to control KUKA robots at the same time using the Research Interface. RCC is interfaced through an changeable dataflow language called Real-time Primitives Interface (RPI). We implemented the Java-based Robotics. The Command Layer, is used for describing real-time transactions critical robot as a composition and coordination of robotic Commands. Robot Commands consist of action. This layer includes an automatic translation of robot transactions into runtime. By the use of Activity Layer of the Robotics API that programmers can provides robot through an extensible set of interfaces. These Activities correspond to robot transactions with metadata that allow composition using composition patterns such as sequential or parallel execution.

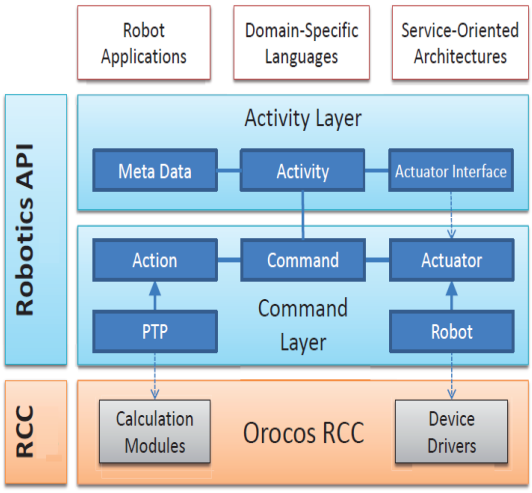

Figure 1. The Robotic architecture with API and RCC.

III. RESULTS

Using software architecture, the given goals could be achieved. Motion programming Language can be achieved by using Activities, by including the higher capabilities of the robot such as force-based motions. The start position of the robot is required for Motion planning, and it can retrieve from the meta-data of the existing activity and it does not have to be given by the programmer.

Figure 2. Manipulator

Meta-data is more flexible than using the current robot position it allows sequence of activities for pre-planning before execution. New modifications can put independent pieces of rough information together for making new rules to modify a program without affecting its structure.

The tasks is given by a human. Robots have many sensors to detect physical data from any ware such as light, heat, cold, deflection, temperature, movement, sound, bump, and pressure finding radio waves.

They have got efficient processors, number of multiple sensors and a huge memory to save the data to analyze exhibit intelligence. In addition, robots are capable of learning from their mistakes by the use of database stored in their memory and they can adapt to the new environment. This approach can be also applied for obey rules motions,

where an Activity may construct contact. By the first releasing force an activities have to cope with this case. The Activity make-up patterns allow to specify actions that are to be implement when a definite state occurs, and originate a multiple sequence of Activities into one transaction to remove unwanted delays between the two Activities. The co-ordination of multiple robots, as all robots can control by a single application. The direct interfaces differentiation shows that Robotics API code does not get the same syntactical compactness as written in KRL.

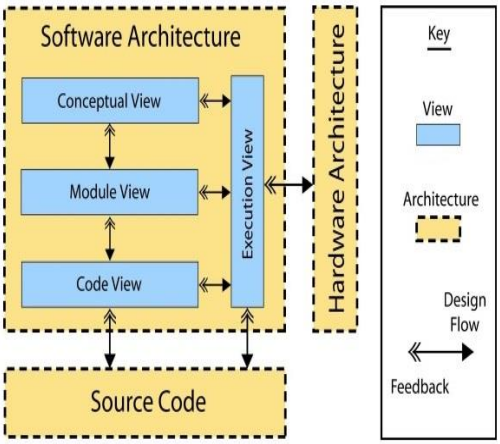

Figure 3. Software Architecture

IV. OUTLOOK

The consideration was evaluated successfully in robotic applications. As development environment with extension supporting the development of robotic applications in meta-data framework, but it's using Service oriented Architectures to control meta-data to greater automation solutions becomes feasible now a days. Moreover the Robotics API is a helpful principle for domain specific robot languages but also for graphical programming of KUKA robots.

REFERENCES

1. *M. Vistein, A. Angerer, A. Hoffmann, A. Schierl, and W. Reif,"Interfacing industrial robots using realtime primitives," in Proc. IEEE Intl. Conf. on Automation and Logistics, Hong Kong, 2010.*
2. *H. Bruyninckx, "Open robot control software: the OROCOS project," in Proc. 2001 IEEE Intl. Conf. on Robotics& Automation, Seoul, Korea, 2001.*
3. *H. Muhe, "On reverse-engineering the KUKA Robot Language," Workshop on DomainSpecific Languages and models for ROBotic systems, 2010 IEEE/RSJ Intl. Conf. on Intelligent Robots & Systems, Taipeh, Taiwan, 2010.*
4. *A. Angerer, A. Hoffmann, A. Schierl, M. Vistein, and W. Reif, "The Robotics API: An object-oriented framework for modeling industrial robotics applications," in Proc. 2010 IEEE/RSJ Intl. Conf. on IntelligentRobots & Systems, Taipeh, Taiwan, 2010.*
5. *Introduction to robotics, SAEED B. NIKU*
6. *KUKA bascis programming manual*
7. *(KUKA) www.kuka-roboter.de*
8. *https://en.wikipedia.org/wiki/KUKA*
9. *S. Lopes, and J. Connell, "Sentience in Robots: Applications and Challenges", IEEE Intelligent Systems, Computer Society, 5(16), 2001.*
10. *http://www.easy-rob.de/product.html.*
11. *BRICS) www.best-of-robotics.org "Best practices in robotics research"*
12. *(OROCOS) www.orocos.org "The Orocos Project, Smarter control in robotics & automation"*