論文 / 著書情報
Article / Book Information

題目(和文)	
Title(English)	Normalizing Abstractions of Middleware-Based Software to Compose Highly-Integrated Robotic Systems
著者(和文)	セロンロペスアルトゥーロエドゥアルド
Author(English)	Arturo Eduardo Ceron Lopez
出典(和文)	学位:博士(工学), 学位授与機関:東京工業大学, 報告番号:甲第10145号, 授与年月日:2016年3月26日, 学位の種別:課程博士, 審査員:遠藤 玄,鈴森 康一,小田 光茂,塚越 秀行,長谷川 晶一,福島　E 文彦
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:甲第10145号, Conferred date:2016/3/26, Degree Type:Course doctor, Examiner:,,,,,
学位種別(和文)	博士論文
Category(English)	Doctoral Thesis
種別(和文)	論文要旨
Type(English)	Summary

（博士課程）
Doctoral Program

論 文 要 旨

THESIS SUMMARY

専攻： Department of	機械宇宙システム　専攻	申請学位（専攻分野）： Academic Degree Requested	博士 Doctor of （ Engineering ）
学生氏名： Student's Name	CERON LOPEZ, ARTURO EDUARDO	指導教員（主）： Academic Advisor(main)	遠藤 玄
		指導教員（副）： Academic Advisor(sub)	福島 E. 文彦

要旨（英文 800 語程度）
Thesis Summary（approx.800 English Words ）

This thesis is aimed to investigate methods for normalizing abstractions in middleware-based software to compose highly-integrated robotic systems, therefore increase the usability of such software and development tools.

Chapter 1 gives the background about this research. The concept of normalization of abstractions in middleware-based software can be defined as “to diminish as much as possible the amount of software abstractions in a robotic system by defining every entity as an element that can be mapped to one or various other related resources”. Service robots are complex systems in terms both of hardware and software, and their required capabilities are usually developed independently in the respective research fields. Due to this, the use of robotics middleware platforms has become a widely-accepted practice in the development of software for service robots. However, when a highly-integrated system is required, a platform-driven approach will produce discrepancy of abstractions among platforms, hindering the composition and usage of the highly-integrated system. Through a survey of different ways of normalizing abstractions, we identified that the normalization must be done at runtime, where we proposed a novel method for normalizing abstractions of middleware-based software through a framework that sets the paradigm of “Roles”. With the implementation of this framework, we expect to increase the usability in the administration of the software and development tools from the highly-integrated system.

In Chapter 2, the proposed Framework for Integration of Elements and Resources by Roles is presented. The definition of a “Role” is introduced, which is a way of grouping a set of elements that compose an activity. Roles are described by the declaration of Objects and Actions, where Objects group elements pertinent to data (e.g. files, data streams, etcetera) in the system, and Actions group elements pertinent to software elements (e.g. runnable programs, scripts, etcetera) in the system. Each element can become a resource when they can be resolved, accessed and retrieved. Elements can be mapped to one or various other synonym elements in the system that represents resources, in this way, creating normalized abstractions. Roles can be chained together when they share Objects in common, and can be reasoned through the Administration by Roles method, which declares a set of premises to be used with propositional calculus. In their implemented version, Roles can be serialized using the RDF and reasoned through a SPARQL engine.

Chapter 3 describes the implementation of the Framework for Integration of Elements and Resources by Roles in the Intelligent Cross-Platform Interface (ICPI). This interface provides two of the requirements for interoperability in the system: Piping resources across platforms and Executing basic administration tasks. The ICPI is composed of the ICPI-Server and the ICPI-Clients. An ICPI-Client can represent a Role in the system, taking care of the elements grouped in that Role. The ICPI-Server implements the Administration by Roles method into a module that has a SPARQL engine. The server can also read Roles from RDF/XML formatted files. Communication with the ICPI-Server and ICPI-Clients can be made through WebSockets and ICPI-QueryMessages to issue different tasks in the system. Also, a graphical user interface (GUI) that uses WebSockets can be hosted in the ICPI-Server (internal HTTP Server). Such GUI provides visual tools for administering the system.

Chapter 4 describes in detail an implementation of a GUI for the ICPI-Server; it is the Hyper-High Level Interface for Service Robots (HyperBot). This interface provides a third requirement of interoperability: Operation of the integral system. It comes with three main screens: The System Developer Screen, which is used to compose, visualize and operate each element composing the highly-integrated system; the Designer Screen, which is used to create monitoring/control panels for the system by inserting a set of widgets

(e.g. buttons, sliders, images, etcetera); and the User Screen, which can execute the designed panel and allows the user to input scripts to perform complementary administration tasks. Proofs of concepts were performed to compose and use a system by using HyperBot in several case studies, where its functionality could be confirmed. Also, the framework's concept was validated and verified through a usability test that was carried out with 12 subjects, which tested the concept of normalization of abstractions through FIERRo and the three requirements for interoperability through the ICPI and HyperBot. In overall, the results showed that our approach has a superior usability when compared against the benchmark approach.

Chapter 5 introduces HyperBot RAPIDE, an integrated development environment (IDE) implemented in the HyperBot interface. Rapid prototypes of robotics applications can be created with this IDE through a proposed method for the systematization of requests to resources in platform-agnostic system models. It features three screens to let the users work with the system and create visualization panels to operate and monitor their robotic systems. Features of the screens include panel design, scripting and 3D views. The systematization method and its implementation of the IDE were verified through a proof-of-concept and a usability test.

Chapter 6 summarized the contribution of this thesis work, gives final comments and discusses about the possible future work. The highlight in this chapter is in the possible applications of this concept that can improve the current technologies aimed to develop software for service robots, especially when fast prototyping is required in a project. As well, a discussion on the current situation and trend on standards for development systems aimed to service robots is presented.