Robotics and Autonomous Systems 61 (2013) 704–713

Contents lists available at SciVerse ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

End-user programming architecture facilitates the uptake of robots in social therapies

E.I. Barakova [a,*], J.C.C. Gillesen [a], B.E.B.M. Huskens [b], T. Lourens [c]

[a] Eindhoven University of Technology, P.O. Box 513, Eindhoven, The Netherlands
[b] Dr. Leo Kannerhuis Doorwerth, The Netherlands
[c] TiViPE, Kanaaldijk ZW 11, Helmond, The Netherlands

ARTICLE INFO

Article history:
Available online 20 August 2012

Keywords:
Robot control
Graphical programming
End-user programming
TiViPE

ABSTRACT

This paper proposes an architecture that makes programming of robot behavior of an arbitrary complexity possible for end-users and shows the technical solutions in a way that is easy to understand and generalize to different situations. It aims to facilitate the uptake and actual use of robot technologies in therapies for training social skills to autistic children. However, the framework is easy to generalize for an arbitrary human–robot interaction application, where users with no technical background need to program robots, i.e. in various assistive robotics applications. We identified the main needs of end-user programming of robots as a basic prerequisite for the uptake of robots in assistive applications. These are reusability, modularity, affordances for natural interaction and the ease of use. After reviewing the shortcomings of the existing architectures, we developed an initial architecture according to these principles and embedded it in a robot platform. Further, we used a co-creation process to develop and concretize the architecture to facilitate solutions and create affordances for robot specialists and therapists. Several pilot tests showed that different user groups, including therapists with general computer skills and adolescents with autism could make simple training or general behavioral scenarios within 1 h, by connecting existing behavioral blocks and by typing textual robot commands for fine-tuning the behaviors. In addition, this paper explains the basic concepts behind the TiViPE based robot control platform, and gives guidelines for choosing the robot programming tool and designing end-user platforms for robots.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

The uptake of robots as assistive tools in clinics and households, and in education will not become a reality before the robot's control is intuitive to everybody. Although some easy-to-use tools for programming of robots exist [1,2], the major hurdle for using robots in everyday life is the difficulty to combine the expertise of robot programmers and developers and the domain specialists in the way that robots can be used to the state of the art of their performance by end users. To address this problem we first select a concrete use case of training social skills to autistic children which defines the users of these assistive robots, namely the therapists as a primary user group that has the potential to facilitate the uptake of robots in this therapy, and the children and their parents as the second user group with no direct preference for this facilitation. Afterwards, we apply a participatory co-creation process that combines (1) a user centered design of a platform to support therapists to create and share behavioral training scenarios with robots and (2) acquisition of domain specific knowledge from the therapists in order to design robot–child interaction scenarios that accomplish specific learning goals [3,4]. The two aspects of the co-creation process are mutually dependent and therefore require an iterative design of a technological platform that will make gradual steps towards creating optimal affordances for therapists to create and share robot-mediated scenarios. So far different frameworks for controlling robots like the subsumption architecture of Brooks [5], or a three layered approach of reaction-control, sequencing, and deliberating/planning, emerged [6,7]. More recently, a number of researchers have developed executives that emphasize model-based approaches and deep integration of automated planning. For instance, constraint-based temporal planning (CTP) [8], and the extent to which it exploits automated planning and plan-based reasoning techniques is at the core of execution. CTP has been applied in a number of practical applications, e.g. [9].

The software architecture's taxonomy, used to implement these approaches, i.e. the ways to put the guidelines as proposed by the frameworks together in a functioning system is less clear.

* Corresponding author. Tel.: +31 402473563.
E-mail addresses: e.i.barakova@tue.nl (E.I. Barakova), j.c.c.gillesen@tue.nl (J.C.C. Gillesen), b.huskens@leokannerhuis.nl (B.E.B.M. Huskens), tino@tivipe.com (T. Lourens).

0921-8890/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2012.08.001

One reason why this is a problem is that there is no agreement on what the space of possible control software architectures is like, nor on the terminology for describing architectures or on criteria for evaluating and comparing them [10]. Objectively, in all these control software architectures, the burden is on the robot programmer to come up with details on how desired behaviors are to be accomplished. This becomes increasingly difficult as robots and their applications become more complex.

We distinguish two groups of robot software architectures, with respect to the user requirements. The first group is developed by robot developers and computer scientists and can be used exclusively in this community. Examples of such platforms are ROS Orca, URBI, and YARP. A common characteristic of these platforms is that they are difficult to use by non experienced programmers, and often they are developed solely for a UNIX platform, which is commonly used by people with a background in the exact sciences.

These software architectures can cope with multiple processes running on different machines, often with different operating systems and hardware. A recent development is the ROS open source operating system for robotics [11]. A ROS runtime graph is a peer-to-peer network of processes that are loosely coupled using the ROS communication infrastructure. ROS is a distributed framework of processes that enables executables to be individually designed and loosely coupled at runtime, and aims to support multiple textual programming languages. Yet Another Robot Platform (YARP) also supports building of a robot control system as a collection of programs communicating in a peer-to-peer way. Its main focus is on long-term software development [12,13]. The Open RObot COntrol Software (OROCOS) focuses on thread-safe and real time communication between tasks [14]. Orca, initially part of the Orocos project, is an open-source framework for developing component-based robotic systems. It provides the means for defining and developing the building-blocks which can be connected to form arbitrarily complex robotic systems, from single vehicles to distributed sensor networks [15]. The creators of Orca make a comparison between seven different robotics software systems [16], and conclude that: *one hopes to integrate existing software modules within a software framework*. The second group of robot programming architectures from the point of view of usability have elements of end-user programs, such as a graphical interface [2,1]. The merit of visual programming as expressed by Microsoft® using their Robotics Developer Studio is that *non-programmers can create robot applications using a visual programming environment* and *a Visual Programming Language* enables anyone to create and debug robotics programs very easily by just selecting the behavioral blocks from a menu, and connecting them in the desired flow of actions. The existing visual programming environments are rather simple and allow the user to connect the existing blocks but creation of novel behaviors is practically impossible for a non-specialist. To address the major shortcomings from the two clusters of platforms we adopted TiViPE which is a graphical programming environment [17], that emphasizes on the integration of existing software routines from (existing) libraries without additional programming. TiViPE also covers the aspects of multiple processes on multiple (embedded) computers, peer-to-peer communication, graphical programming, massively parallel (GPU) processing, and multiple operating system support [17]. Another platform that attempts to integrate the advantages of the two groups of approaches by providing a standard robotics interface through a low level language is the Universal Robotic Body Interface (URBI) [18]. However, TiVIPE, has the advantage of providing an advanced end-user interface. The interface of TiViPE uses colored blocks that are connected by clicking on the icon, while the textual robotics command language used within TiViPE allows a user to get more fine grained control over a robot by either typing these commands or using and connecting available graphical blocks. In addition a substantial set of modules for sensory data processing is available within TiVIPE.

Starting from the existing TiViPE programming environment that already had a modular structure and components for reading sensory information, an architecture capable of controlling robots and able to meet the challenges of the particular application domain was developed. The user group of therapists in autism practice who want to use robots to train social skills to autistic children define the requirements for the architecture in the needs to create and change training scenarios for autistic children. The issue of creating meaningful training sessions with a robot has been addressed previously by Bernd and colleagues [19]. However, in this study the therapist was a knowledge provider and he/she was not meant to take part in the training practice. Differently, we engaged the therapists in a co-creation process for the development of scenarios that they would like to use as an augmentation to their practice. The Therapist-in-the-loop approach was proposed by Colton and colleagues in [20]. These authors attempted to engage the children in social interactions with a team of therapists with the help of the robot. However, they do not consider designing a programming tool that empowers therapists to make their own training programs. Our aim is to engage the autistic children in interaction with therapists and with other children through a sustainable process that makes it possible that therapists and eventually parents can by themselves create, share, and reuse interactive scenarios with training and entertainment purposes. In this particular paper we identify the main technical requirements, determined by the user requirements, for a platform that can support this process and thus has the potential to facilitate the uptake in robots in social therapies. We do that by co-developing the programing architecture to be usable by therapists and other end-users, but generic enough to make possible implementation of state-of-the-art robot programming concepts. In addition, this paper can be a user manual for people that want to use the TiViPE based robot control platform, and is a contemporary guide for choosing an appropriate robot programming tool and designing new end-user platforms.

This paper is organized as follows. First, we define the requirements for this particular application in Section 2. In Sections 3–5 we introduce the technical concepts that are able to meet these requirements. The paper is meant for developers of user environments as well as end users who want to understand the basic concepts behind the programming of robot behavior. We summarize the results of several small pilot tests and the benefits for the mentioned communities in section 6, where we offer a discussion for further developments.

2. User requirements for the robot architecture

Autism Spectrum Disorders (ASD) are conditions where no curative treatments are available, but intensive behavioral interventions by young children during one year or longer may bring substantial improvements. Early intensive behavioral intervention has the greatest amount of empirical support and meets the criteria for evidence-based treatments [21]. Searching for the answer of how training can benefit by the use of robots, we identified two intertwined processes: a user centered design of a platform to support therapists to create and share behavioral training scenarios with robots, and the acquisition of domain specific knowledge from the therapists in order to design robot–child interaction scenarios that accomplish specific learning goals [4].

Although we identified that an effective collaboration between therapists, robot specialists, software developers and human–robot interaction researchers is a necessity for an efficient

Download English Version:

https://daneshyari.com/en/article/412674

Download Persian Version:

https://daneshyari.com/article/412674

Daneshyari.com