A Framework for Rapid Deployment of Devices and Robots on a Network

Raju P. Badapanda, Shivashankar B. Nair,
Department of Computer Science & Engineering,
Indian Institute of Technology Guwahati, Guwahati India

Dong Hwa Kim
Department of Instrumentation & Control Engineering, Hanbat National University, Daejon South Korea

Abstract: With the advent of an almost omnipresent Internet, the concept of embedding and controlling devices on a network is fast becoming feasible. This idea has found many implementations especially in the field of robotics but most of them are limited to the control of specific robots. A large amount of effort is required in actually realizing the practical implementation. The need of the day is to devise a tool that can aid device and robot owners to endeavour and make them available to others over a network. One can then envisage such devices and robots on a network to co-operate and share both their resources and knowledge. This paper describes an attempt at building such a tool that can facilitate the rapid deployment of a variety of robots and devices. The tool, which is made available as a *Wizard* in Microsoft .NET platform, enables deployers to expose the functionality of the devices and robots using web services. The paper also describes the evolution of the tool and the way its generic nature was tested by deploying dissimilar robots and a device on the network.

1. INTRODUCTION

Programming robots for real-world applications introduces considerable difficulties due to the complexity of the task at hand [1]. The advent of laptops, palmtops, wearable computers and the like, gadgets that help realize ubiquitous computing [2],[3], coupled with improved and reliable network connectivity, call for not just the use of computers running software at far off ends but for intelligent robots and also devices serving physical mobility at remote places. The goal of realizing a community of remote autonomous robots working in co-operation can be achieved only if the robot/device owners are encouraged to embed their robots on the Internet. For such amateur roboticists to graduate to professional robot deployers, numerous diverse issues have to be looked into and studied before actually embedding them on the Internet. While on one side the use of the Internet as a medium for transporting and realizing physical movement at distant and isolated locations is the need of the day, a common platform that provides robot-independence (analogous to device independence) is far more desirable. What has therefore to be evolved is a standard to embed, command and control robots and devices over the Internet. Several attempts have been made in this direction; the two more prominent ones being the Open Robot Interface (ORiN) [4] and the R-Cubed Transfer Protocol [5]. ORiN is a system that can be used for standardization of communication between a personal computer and the robot interface. It allows local information such as position, torque, number of parts assembled, etc. to be made available on a network.

There is still more to real world robotics – some robots may be capable of performing one set of jobs while others may be able to perform yet another. Though this may not be mutually exclusive, a strategy that will endorse co-operative behaviour in solving complex problems has to be evolved. A medium and protocol for communication in such multi-robot scenarios becomes a *sine qua non*. Connectivity to the Internet should also ensure availability of web-based information repositories and assisting agents [6,7] to transform them to useful knowledge.

As on date a number of robots and devices are already on-line [8-10] and ready to use via the Internet but each has its own custom interface catering to specific commands. A facility for programming and controlling such robots/devices to meet a user's needs does not exist.
In this paper we discuss the formalization of a methodology by which programmers can deploy robots on the Internet with ease and also allow their functionality to be used and programmed by remote users.

Based on the studies conducted on some typical robots and devices we have evolved a framework that can facilitate rapid deployment of robots and devices on the Internet. The framework has been formulated based on discernible functional decomposition and is suited against a stack of expertise levels. It is envisaged that the framework will enable users to command a variety of devices including robots and home appliances through normal desktops and smart devices. Users may connect to a robot through a web interface and use the functionality supported by it. They may, optionally, query a Centralized co-ordinating Server in case they are not aware of the Universal Resource Locator (URL) of the desired robot/device.

2. FEATURES OF THE FRAMEWORK

The framework enables users to embed robots/devices on Internet. Some noteworthy points of the framework that render near device independence are cited below.

1. It creates a vertical stack of expertise levels, each of which focuses on certain part of the complex issues possible to make devices remotely accessible to users and other peer devices already available in the Internet.
2. It assists developers in writing code for embedding robots in the Internet and invisibly leads them to conform to certain standards that aid them talk to their peers.
3. It veils the intellectual properties of the developers involved. Most manufacturers prefer not to divulge detailed information about the manner in which their robots and devices are interfaced.
4. It allows even amateur programmers to deploy robots on the Internet and allows remote users to use and customize the functions exposed facilitating meta-programming.

We have developed a *Wizard* that assists the process by automatically generating a VC++ .NET project with all necessary source-code template-files and conforms to the framework The underlying architecture follows an emerging paradigm for distributed computing known as Service Oriented Computing (SOC) [11]. This architecture effectively encapsulates all device-specific details at the local implementation level and enables devices to communicate with their clients in terms of higher level primitives

called as services, or more precisely, as web-methods that are accessible just like function calls across the web. As is evident from its name, the primary element of this paradigm rests upon a Service [12], an autonomous platform-independent computational element that can be described, published, discovered and programmed using standard protocols to build responsive networks of collaborating distributed applications.

3. ARCHITECTURE

Christened as RobIN-II, short for Robots on the Internet, the framework is primarily a peer-to-peer distributed architecture, with additional specifications of how to find one's peer. Figure 1 illustrates a broad overview of the overall architecture.

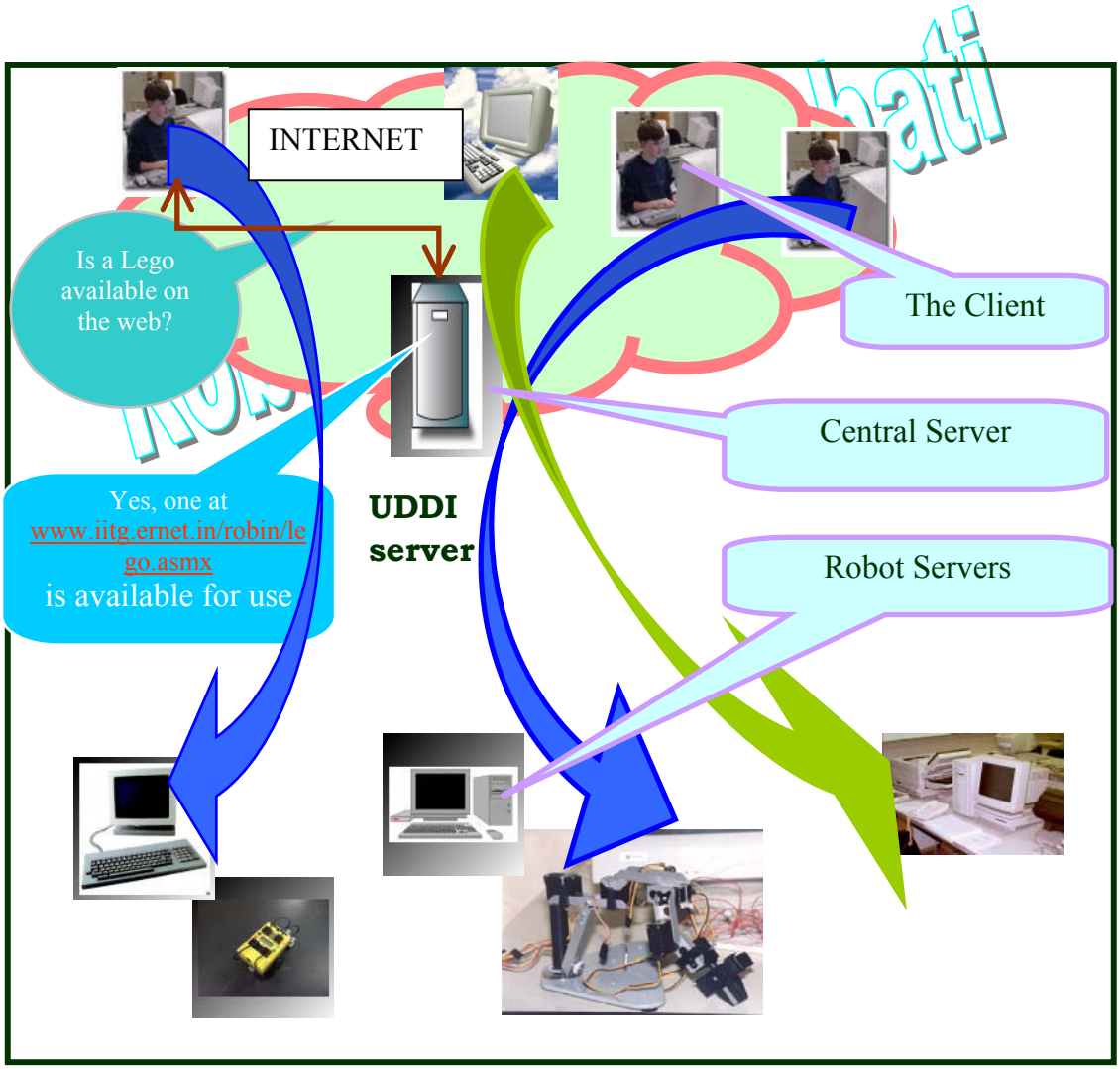

Figure 1 – RobIN-II Architecture

The figure reveals how the initial query is sent to the UDDI server that hosts all services available on the network. Once the URL is obtained, access is directly made

to the server hosting the robot/device. The figure shows three such robot servers hosting a LEGO robot, a Robix Rascal robot and an ink-jet printer respectively. RobIN-II follows a step by step improvement approach similar to the one used by Reid G. Simmons et al. [10] for Xavier, to make robots/devices appear on the Internet.

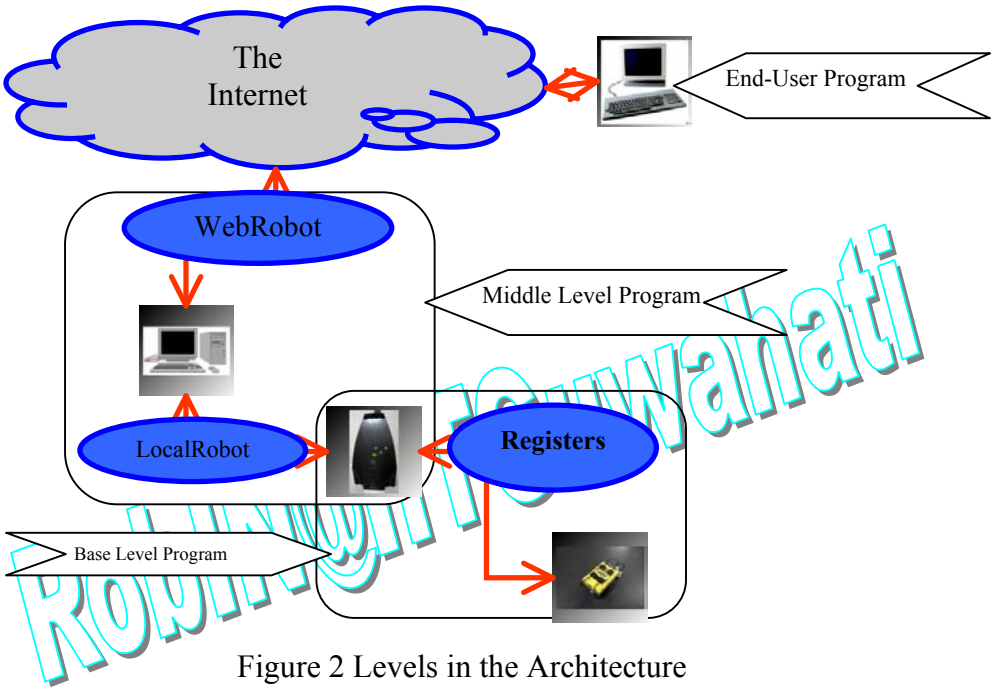

Figure 2 Levels in the Architecture

Using Microsoft .NET platform as a base for development, the RobIN-II framework assumes three types of programmers described below –

1) The Base Level Programmer (BLP)

Proficient in writing device drivers and tools for hardware, this programmer is the one who is looked upon synonymously with the manufacturer of the device/robot in question. The BLP has extensive knowledge of the device at hand and is the sole authority who writes and supplies device specific code for a customer. This software, which is usually the device driver itself, comes in the form of DLLs that work in conjunction with the hardware device controller. The rest of the world only sees and talks to the device driver to control and command the device.

2) The Middle Level Programmer (MLP)

Conventionally, this programmer is the one who can understand the intricacies of using and extending functionality of device drivers supplied by the BLP. He is proficient in at least one of the conventional programming languages and is capable of using the DLLs and APIs supplied by the BLP, to create custom code for the End-user. In RobIN-II, the Wizard, frees the MLP, to a great extent, the task of programming and deploying the robots/devices on the Internet.

3) The End User Programmer (EUP)

A user of a robot/device is finally forced to deliver a sequence of instructions or commands that form the end-user program. In RobIN-II, the EUP is basically a user commanding the robot through a client machine connected to the Internet. The EUP may also program the robot for command and control, if certain provisions are made by the BLP and endorsed by the MLP.

Figure 2 schematically depicts the broad levels of task distribution. The view at the base level is that of registers. The programmer at this level (generally the manufacturer) views the robot and its interface circuitry as a set of registers and logic gates that have to be controlled. This set is provided to the MLP, the person who buys and wishes to deploy the robot. The MLP in turn deploys the robot thereby exposing its functionality to the EUPs on the network who either merely use the robot or create their own meta-level programs. Each level handles the details the upper level has abstracted away, and directs some commands/abstractions to the lower level. The LocalRobot is the robot that is operable locally at the computer hosting it while the WebRobot is operable remotely, through the Internet.

4. ROBOT DEVICE DRIVERS – SOME GENERIC VIEWPOINTS

After a study of a variety of robots and devices, it was found that in order to be operable most devices need to pass through a definite set of sequences. The most prominent ones have been listed below.

A. Device initialization

After a robot/device interfaced to the hosting machine is powered on, the electronic or infrared controllers are activated. These active communication paths support the communication protocol between the device and the host computer as defined by the manufacturer. The major phases of device initialization are:

1) Configuration loading

Robots are basically a congregation of actuators and sensors, each of which has its own operation settings. For instance, servo motors of a Robix Rascal robot can be positioned within a range corresponding to 1 to 1400 units. Similarly the robot as a whole needs to keep track of a number of sensors and motors attached to it. Though most of the small scale robots have this information hard coded into their firmware, the larger robots with variable configuration support need to be initialized. This task is done at this phase.

2) Loading history-database for intelligent operation

In advanced robots where past experience and intelligence are preserved, reloading the relevant databases becomes a pre-requisite. This operation has to be performed each time the robot/device becomes functional.

B. Device operation

Once the device becomes directly operable by the device driver, it is in a position to accept a command from the user and convert it into a sequence of register operations supported by the hardware controller of the robot. It can also query the controller for feedback from the robot/device and pass on the information onto the user.

C. Device clean-up

This is the reverse process of initialization. Before the device is powered down or

disconnected it needs to update some book-keeping information, the most prominent ones being -

1) Back up of configuration data

If there is any change in the configuration of the device, like, for instance the elbow servo motor of a robotic hand has been shifted to its shoulder position, then this needs to be remembered so that the device functions correctly the next time it is referenced.

2) Updating the Intelligent data store

The more the experience gathered by an intelligent robotic agent the more decisive its operation. Thus each session can make it richer in terms of the knowledge it has gathered. This substantiates the need to update the existing databases.

All the programmable robot/devices come with their own device drivers and the rest of the world sends commands or requests to these drivers through the relevant Application Programming Interfaces (APIs) provided along with. Device drivers, as mentioned, are often made available in the form of a Dynamic Link Libraries (DLLs), which get loaded into the memory and make the concerned functions available to other applications. These libraries contain low-level functions and internal data structures to store a variety of state information about the robot. The low-level functions usually contain machine code to directly access system resources useful to the robot. During execution the application should be able to locate the DLL so that it can load it onto the memory to enable it to dynamically bind to the referred functions. Once the application has used the functionality provided by the DLL, it should be unloaded from memory.

5. A CONCISE CASE STUDY

In order to test the generic nature of the *Wizard*, two robots, a Lego robot rover and a Robix® Rascal robot, configured as a stationary robotic arm with 6 Degrees of Freedom were deployed on the Internet. The framework was also used to deploy an ink-jet printer successfully. Since the printer is a comparatively simple device and unlike specific robots does not need intricate explanations about the nature of functioning, we describe how we deployed it using the *Wizard*.

A Hewlett Packard DeskJet 840c printer, connected to a Windows 2000 machine was used. Instead of going through the complete documentation and viewing the device from the manufacturer's perspective, we created a simple DLL to print a representative "Hello World" file by invoking the Windows Notepad application on the hosting machine. This no doubt simplified the BLP's (manufacturer) task but in no way affected the credibility of the test. The RobIN-II Wizard shown in Figure 3 was invoked to assist the MLP in the deployment. Figure 4 lists the files generated using the *Wizard*.

Figure 3 The RobIN-II Wizard

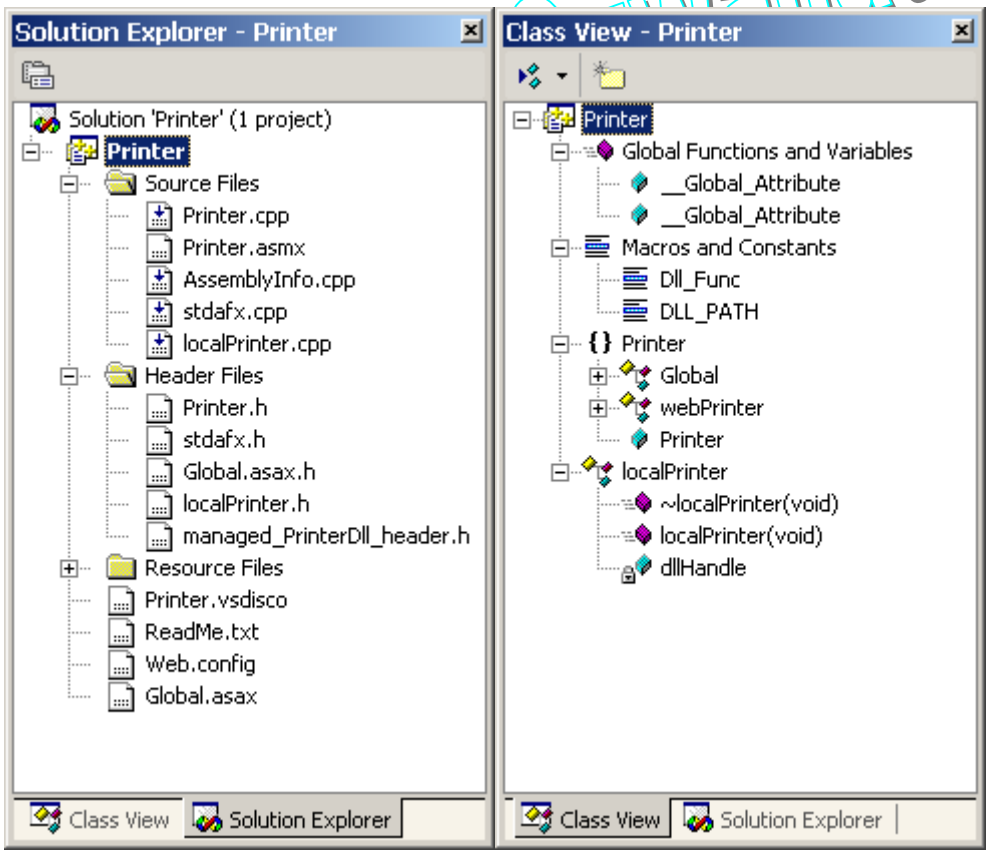

Figure 4 Files & Classes generated for the Printer

It was found that only a few lines of code had to be altered in five of the fourteen files created by the *Wizard* before compiling. The code as well as the files which had to be touched up are shown in Table 1.

Table 1. Code to be altered for deploying the Printer

File name	Line no.	Contents to be inserted
managed_printerdll_header.h	14	#define DLL_PATH "C:\\Documents and Settings\\puneet\\Desktop\\raju\\Deskjet 840c Printer\\printer_dll\\Debug\\printer_dll.dll "
	18	Dll_Func int helloWorld(void);
localprinter.h	21	int printHello(void);

localprinter.cpp	45	// other functions required to support the advertised webmethods
		int localPrinter::printHello(){
		return helloWorld();}

Printer.h	35	String __gc* Hello_World_to_the_Printer();
Printer.cpp	13-26	String __gc* webPrinter::Hello_World_to_the_Printer( ) { if (Printer->printHello()) return "Successfully printed"; else return "there is some problem"; }

These minor alterations clearly depict the drastic reduction in work on part of the MLP. A conventional C programmer can easily comprehend the meaning of each of the contents being altered or inserted. When the compiled code was run on the host machine, the printer became visible on the network in the form of a webservice. Using the *Wizard* and adding extra functionality, an MLP can now make all features of a standard printer accessible over the network.

6. CONCLUSIONS

After deploying two robots and a device, we concluded that Robin-II could certainly promoted amateur roboticists to professional deployers. With the added *Wizard* support it can potentially make almost every programming-savvy person an MLP. By enabling EUPs to program remotely hosted robots/devices, it opens a plethora of interesting scenarios. For instance, a EUP can graduate to a Meta level Internet programmer by using web methods exposed by different robot/device deployers (MLP). Assuming that each MLP exposes the entire set of primitives (instruction set) of a robot as web methods a large number of EUPs can pitch in to use web methods of different robots to program them over the Internet These EUPs may further expose their enhanced web methods for use by another generation of EUPs, leading to encapsulated web methods that are highly user-friendly. RobIN-II thus gradually motivates EUPs to narrow the man-machine barrier over the Internet.

ACKNOWLEDGEMENTS
The authors wish to thank Microsoft Inc. for the funding provided for this work under their Academic Alliance Scheme. The authors also express their gratitude for the support rendered by the South Korean Science and Engineering Foundation.

REFERENCES

[1] Harvey I., Husbands P., and Cliff D. Issues in evolutionary robotics. In J. -A. Meyer, H. L. Roitblat, and S. W. Wilson, Editors, From animals to animates 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior, Honolulu, Hawaii, 1992,. The MIT Press, pp. 364-373.

[2] G. S. Sukhatme, Maja J. Mataric, Embedding Robots into the Internet, Communications of the ACM, May 2000, Vol.43, No.5, pp. 67-73.

[3] Weiser, M. Some Computer Science problems in ubiquitous computing. Communications of the ACM, July 1993, Vol. 36,No.7.

[4] Open Robot Interface for the Network, http://www.jara.jp/E_ORiN/En_ORiN.htm

[5] R-Cubed Transfer Protocol, R-Cubed Manipulation Language Home Page, http://www.star.t.u-tokyo.ac.jp/projects/RCML

[6] Malviya M., Nair S.B., "WeBAssistant: An Intelligent Web Agent", Proceedings of the International Conference on Information Technology, ICIT'01, Gopalpur-on-sea, India, December 2001, pp. 351-354.

[7] Nair, S.B., Pandey, V.N., "A Remote Data Mining tool for Agents", Artificial Intelligence – Future Trends - Proceedings of the International Symposium on Artificial Intelligence, ISAI 2001, Fort Panhala, India, Akerkar, R. (Ed.), Allied Publishers, India, 2002, pp.311-316.

[8] http://ranier.hq.nasa.gov/telerobotics_page/realrobots.html

[9] Backes P.G., Tao K.S., and Tharp G.K., "Mars pathfinder mission Internet-based operations using WITS" in Proc. IEEE International Conference on Robotics and Automation, , May 1998, pp.284-291.

[10] Simmons Reid, Fernandez J., Goodwin R., Koenig S. and Sullivan J.O, "Xavier: An Autonomous Mobile Robot on the Web." Robotics and Automation Magazine, 1999. Available: http://www.cs.cmu.edu/~Xavier

[11] E-services: a cornucopia of digital offerings ushers in the next Net-based evolution, Communications of the ACM, Volume 46, Issue 6, and June 2003.

[12] ICSOC04, 2nd International Conference on Service Oriented Computing, New York City, NY, USA, November, 15-18, 2004.
Available: http://icsoc.org/icsoc04.html