www.icgst.com

ARCSUS
Advanced Robotic Control System Using SVG

F. Tampalini, R. Cassinis
University of Brescia, DEA,
Brescia, Via Branze 38, Italy,
[fabio.tampalini, riccardo.cassinis]@ing.unibs.it,
http://bsing.ing.unibs.it/~cassinis/ARL/

Abstract

We present the study and realization of a control system created for distributed systems. We propose to use a browser to control robotics system with XML and SVG formatted files; so the system exploits the web server power and allows different control operations through any OS and any browser equipped with the SVG plug-in. We created ARCSUS (Advanced Robot Control System Using SVG) to monitor and control different devices linked.

Keywords: *Autonomous robot, Distributed system, SVG, XML.*

1 Introduction

In this article we present the study and realization of a control system created for distributed systems. We planned ARCSUS (Advanced Robot Control System Using SVG) to monitor and control a great number of different devices linked by Internet. Nowadays, we are surrounded by a lot of cooperative agents, for example PDAs link to remote computers or satellite systems to get data about their position at any time. In the last years, UMTS technology has allowed the production of mobile phones that are able to send real-time shots or make video calling. At the same time we have an increasing presence of low-cost webcams in working and domestic environments. Many factories producing these device agents are pointing at an easy integration to simplify and empower any of them. The idea of distributed systems is the same one we have carried out in our research. It let us create a system of distributed agents, which was used for video-surveillance in our University Faculty of Engineering, called SAURON (Surveillance AUtonomous Robot Over Network)(in Fig. 1). We have created a system composed by:

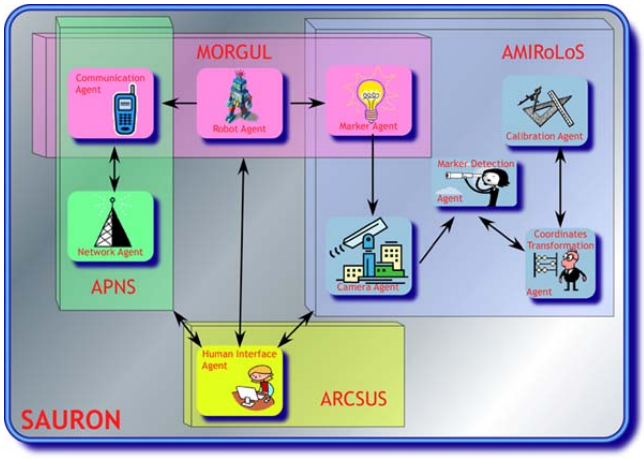

Figure 1: SAURON (Surveillance AUtonomous Robot Over Network): video-surveillance system composed by cooperative distributed agents.

- MORGUL (Mobile Observation Robot for Guarding the University Laboratories): this robot carries a laptop containing its hybrid control system, which allows it to stay connected to Internet. When the robot has flat batteries it moves autonomously to its docking station [5]; MORGUL is composed by:
  - Communication Agent, composed by hardware and software that allow robot to stay linked to Access Points;
  - Robot Agent, the hardware robot and laptop with OS and programs that permit robot to perform its tasks;
  - Marker Agent, composed by a set of LEDs and software that control the marker itself.
- AMIRoLoS (Active Marker Internet-based Robot Localization System): composed by a system of low-cost webcams, which is controlled

by a server determinating the robot pose [6]; AMIRoLoS is composed by:

  - Marker Agent;
  - Camera Agent, a commercial off-the-shelf webcam connected via a USB link to a PC;
  - Marker Detection Agent, the heart of AMIRoLoS. Its purpose is to examine the streams of images from the camera and to find blobs whose behavior matches the behavior expected from the marker;
  - Coordinates Transformation Agent has the main task of transforming the coordinates of the marker seen by the camera into actual real world coordinates and to transmit them to the robot navigation agent;
  - Calibration Agent, a software system that human operator must use to set-up AMIRoLoS system.

- APNS (Automatic Predictive Network Selection): it allows the robot to stay connected to Internet [4]; ANPS is composed by:
  - Communication Agent;
  - Network Agent, a program working on MORGUL laptop.

While developing the system, we faced some typical limitations of normal graphic interfaces made to monitor and control the agents:

- strong dependence on libraries and their cycle of life (consider backward incompatibility, e.g. between Qt3 and Qt2);
- in order to export the interface to remote systems, the necessity to use some ports that are probably unsafe;
- hard coexistence of different graphic interfaces on the same screen;
- inconsistence between interfaces;
- insufficient analysis of human-machine interaction;
- strong dependence on OSs;
- ...

In order to face and overcome these limitations, we have developed the system described in the following sections.

2 State of the Art

At present, there are some models of controllers for energetic power stations [8] but there are no real specific applications based on SVG able to control a robotic system; we can also assert that a real Human-Robot interface analysis does not exist at all [10].
The main studies are now focusing on the different kinds of prepping data need in order to be exchanged between devices. It is now clear that they will have XML prepping. The lack of H-R interface analysis is due to the fact that graphic interfaces, in robotics, are often bare , as they are mainly used by developers who monitor and eventually set each device in the correct way, without a general vision of the cooperative system. This is partly explainable by the fact that we are talking about autonomous robotic systems. It is also true that, in a few years, autonomous robots, and cooperative distributed devices systems in general, will have an increased presence in our daily life. As a consequence, systems will be no longer used just by experts or developers but even by common people, so we will need powerful and user-friendly interfaces, allowing everyone to control these kinds of devices. Therefore, developing graphic interfaces in SVG is a reasonable choice; in fact, besides the benefits (described in this article), SVG is nothing but a XML dialect. Flash could have been another useful instrument [9] but we chose SVG after several compared analysis. Many similarities exist between Flash and SVG:

- both SVG and Flash files are vector-based;
- both SVG and Flash files can be animated;
- both required plug-ins to view images;
- both SVG and Flash files can be zoomed and panned;
- both SVG and Flash files can contain sound;
- both SVG and Flash files are script-able, though Flash's Actionscript is proprietary. SVG can be scripted with Javascript, Java Bindings, and ActiveX controls;
- both SVG and Flash files allow hyperlinking;
- both SVG and Flash files have database connectivity;
- ...

It turns out that SVG has some very distinct advantages over Flash:

- SVG is open-source code. This means you can easily see, read, and edit the code underlying an SVG image. Flash's code is proprietary, or owned by Macromedia, and therefore is hidden

from public view. Viewing Flash code wouldn't do most of us much good anyway, because it is written in binary format, which consists of strings of zeros and onesâĂŤnot of great use to most flesh-and-blood-based life forms;

- SVG is text-based. So, it is easy to read an SVG file and easy to create one. Text used in SVG remains selectable and searchable. To make or edit a Flash file, you must have access to Flash software, and even then, you won't be able to read what's really going on under the hood;
- You only need a texteditor to create SVG, no need to buy Flash, there are a few SVG editors available, but only just starting to become useable;
- SVG is fully scriptable - using a DOM1 (part DOM2) interface and Javascript. That means you can start with an empty SVG image, and build it up using Javascript. This is great for real-time display of , for instance, a machine's status online. You can have dials and readouts replicating those on the machine, showing the values in buffered real-time;
- SVG can easily be created by ASP, PHP, Perl or whatever, and extracted from a database. (take care to set correct mime-types on the server);
- SVG has a built-in ECMA-script (javascript) engine, so you don't have to code per browser, and you don't need to learn Flash's action-script;
- SVG **is** XML, meaning it can be read by anything that can read XML . Flash can **use** XML, but needs to convert it before use;
- This also allows SVG to be transformed through an XSLT stylesheet/parse;
- SVG supports standard CSS1 style-sheets.

However SVG and Flash are not direct competitor. SVG is basis as a subset of XML gives it the ability to do things that Flash cannot, such as coordinate seamlessly with other Web standard technologies. Flash is a proprietary, binary-coded application used primarily for Web animation, which it does quite wells [7].

3 Architecture

ARCSUS is a meta-system useful to create graphic interfaces. Many characteristics make this system more useful and powerful than common graphic interfaces, usually developed for these applications. The system is based on the following widespread standard technologies:

- SVG (Scalable Vector Graphics) is a language for describing two-dimensional graphics in XML. SVG allows three types of graphic objects: vector graphic shapes (e.g., paths consisting of straight lines and curves), images (raster) and text. Graphical objects can be grouped, styled, transformed and composed into previously rendered objects. The feature set includes nested transformations, clipping paths, alpha masks, filter effects, template objects and extensibility [12, 16].
- XML (Extensible Markup Language) is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere [15, 13].
- XSLT (eXtensible Stylesheet Language Transformation) is designed to be used as a part of XSL, which is a style sheet language for XML. In addition to XSLT, XSL includes an XML vocabulary for specific formatting. XSL specifies the style of XML documents: by means of XSLT, it describes the transformation of a document into another XML document using the formatting vocabulary. XSLT is also designed to be used independently from XSL. However, XSLT is not supposed to be a complete general-purpose XML transformation language, rather it is developed primarily for the transformations that are necessary every time XSLT is used as part of XSL [11].
- PHP (Personal Home Page Tools) is a widely used general-purpose scripting language especially suited for Web development, which can be embedded into HTML [2].

While developing the system, we focused on the importance of H-R interaction and on the possibility of exporting and visualizing the interface by means of several kinds of agents: not only common computers or notebooks but even mobile phones or PDAs, thanks to empowering systems visualizing SVG pages even on these kinds of devices [3, 14]. Making the interface compatible with a wide range of devices is already a SVG function; in fact, since it is based on vector images, it is also easily resizable and suitable without changing the original source code.
Let is underline the initial idea of ARCSUS: creating a component useful to human operator to monitor and control a system focused on the use of **autonomous** patrolling robots for surveillance purposes. In this way, it is possible to modify some parameters but these variables are not fundamental for hard real time tasks. The hard real time tasks are autonomously processed and controlled in stand-alone programs. In

this way the server can be delayed in time to respond the required data.

4 Realization

In this section, we describe the system realization and show the interface realized to monitor AMIRoLoS system (in Fig. 2).

At device level, the system works in this way; if the

Figure 2: ARCSUS page about AMIRoLoS monitoring.

robot is not sure about its own pose or needs certain data, it asks for more information through a middleware in which all the cooperative agents are plunged.

One of the cooperating devices is a server getting the robot requests. After processing the images of the robot taken by a webcam, it converts them in coordinates related to the robot environment and sends them to the robot. These exchanges of requests and answers between agents are all XML formatted. After calculating the coordinates, the server transmits XML file to the devices needing it, by means of the middleware we have already described. A web server is interested in these data, too; ARCSUS is installed on this web server. In Fig. 3 we can see an example of XML formatted data sent from AMIRoLoS server. This file contains information about date, time, pose and further details about the data acquired during the images elaboration by AMIRoLoS server. These data are the ones visible in Fig. 2 on a browser after SVG conversion. At web server level (strictly about ARCSUS), the system works in this way: data in XML format (in Fig. 3) get into the web server and, by means of a PHP code (in Fig. 4), they are processed by an XSLT file (in Fig. 5) to obtain an SVG file (in Fig. 6). The main aspects of the PHP program are:

- creating and giving a now XSLT processor:

```
$xh = xslt_create();
```

- `xslt_process()` function is one of the most important in XSLT extension. It allows any XSLT transformation by using almost any source as an input. The simplest kind of transformation based on `xslt_process()` function is the transformation of an XML file by means of an XSLT file; the result is another XML (or HTML) document:

```
xslt_process($xh,$inputXML,$inputXSL,$outputXML);
```

- video printing of SVG page created by `xslt_process()` function:

```
echo '<embed src="./'.$outputXML.'"type="image/xml+svg"/>';
```

- releasing the XSLT processor identified by the given handle:

```
xslt_free($xh);
```

As we can see, the operative way is the same for each controller of each device present in SAURON; it is definitively the same for any system we want to control. We created an XSLT file respecting all the specifications and, as any new data comes from the agents, it is processed by PHP file. In a similar way, when human agents modify the control settings, these new data are saved in an XML file which is sent to the involved devices by means of the middleware. We have created SVG pages for APNS, for images acquired by the webcam mounted on-board MORGUL, etc. with similar XSLT files and respective XML files. Now we have a web site in which we can monitor and control SAURON system.

5 Experiment

ARCSUS was tested to be free from programming problems. The system (still in beta version at that time) was used in the demonstration we gave at SMAU 2005 to introduce MADSys (Multi-Agent Development System) [1] where SAURON was present. During the exposition, the robot moved itself in the Faculty of Engineering Campus, at Brescia University, and it was possible to monitor it from the exposition stand (in Milan) thanks to a big screen employed as a monitor for a browser interfacing with ARCSUS. It is important to stress that time slice necessary to load one SVG page is similar to the one necessary to load a simple HTML page (as SVG page is a XML page). So our interface is lighter than a Flash site. ARCSUS system was studied and realized to completely respect W3 specifications; therefore it is certificated and validated by on-line validator of W3 Consortium for SVG and XML files.

6 Conclusion and Future Research

The ARCSUS system presented in this article is a really powerful instrument for autonomous robotic and it is very useful for cooperative distributed agents

```
<?xml version="1.0"?>
<!DOCTYPE amirolos [
        <!ELEMENT amirolos (date,time, image, world)>
        <!ELEMENT date (#PCDATA)>
        <!ELEMENT time (#PCDATA)>
        <!ELEMENT image (marker_coord, max_value, pixel_max_value,
                average_value, pixel_average_value)>
        <!ELEMENT marker_coord (x, y)>
        <!ELEMENT x (#PCDATA)>
        <!ELEMENT y (#PCDATA)>
        <!ELEMENT max_value (#PCDATA)>
        <!ELEMENT pixel_max_value (#PCDATA)>
        <!ELEMENT average_value (#PCDATA)>
        <!ELEMENT pixel_average_value (#PCDATA)>
        <!ELEMENT world (robot_coord, distance)>
        <!ELEMENT robot_coord (x, y)>
        <!ELEMENT distance (#PCDATA)>
]>
<amirolos>
        <date>
                14&#47;11&#47;2005
        </date>
        <time>
        14&#58;17&#58;06
        </time>
        <image>
                <marker_coord>
                        <x>445</x>
                        <y>195</y>
                </marker_coord>
                <max_value>75</max_value>
                <pixel_max_value>1</pixel_max_value>
                <average_value>30</average_value>
                <pixel_average_value>5</pixel_average_value>
        </image>
        <world>
                <robot_coord>
                        <x>11657</x>
                        <y>25108</y>
                </robot_coord>
                <distance>28554</distance>
        </world>
</amirolos>
```

Figure 3: Information about robot pose formatted in XML file.

systems. We showed an application of ARCSUS in MADSys project: we realized a system composed by robot, web server and webcam, which is able to make video surveillance in open or close environments; in this article we focused specially on AMIRoLoS interface, the system used by the robot to update and modify the informative data about its own pose. By using an SVG page inside ARCSUS, it is possible to monitor the server behaviors which provides correct information about the robot pose, thanks to the images taken from the webcams. ARCSUS was developed so that it could be validated according to W3 Consortium specifications. Therefore, it could be a possible standard for interfaces visible not only on monitors (with different resolutions) but also on PDAs or mobile phones. ARCSUS characteristics and formats make it usable and integrable in working and operative systems. For the future we are:

- developing SVG pages to completely control the related agents that will make the human operator able to set the parameters from the browser and to give directives;
- increasing and verifying this interface application even on agents different from computers (e.g. PDAs, mobile phones,...). In this way human operators can control SAURON (or other systems) and monitor the environment in every moment, even making other tasks or being outside their office;
- PDAs can be used as specific agents for other tasks;
- increasing the number of agents controlled by ARCSUS.

Acknowledgements

The authors acknowledge partial support by MIUR COFIN-03 contract.

References

[1] Multi-agent development system. http://www.airlab.elet.polimi.it/MADSys/, 2005.

[2] Personal home page tools, php hypertext processor. http://php.net, 2005.

[3] Tinyline. http://www.tinyline.com, 2005.

[4] R. Cassinis, F. Tampalini, and P. Bartolini. Wireless network issues for a roaming robot. In *Pro-*

ceedings of the ECMR'05, pages 74–79, Ancona, Italy, September 7-10 2005.

[5] R. Cassinis, F. Tampalini, P. Bartolini, and R. Fedrigotti. Docking and Charging System for Autonomous Mobile Robots. Technical Report Tech. Rep. no. R.T.2005-02-4, DEA, University of Brescia, (Italy), 2005.

[6] R. Cassinis, F. Tampalini, and R. Fedrigotti. Active markers for outdoor and indoor robot localization. In *Proceedings of the TAROS 2005*, pages 27–34, London, England, September, 12-14 2005.

[7] G. Held, T. Ullrich, A. Neumann, and A. M. Winter. Comparing .swf (shockwave flash) and .svg (scalable vector graphics) file format specifications. http://www.carto.net/papers/svg/comparison_flash_svg/, 2005.

[8] N. Jay. Scalable vector graphics, an open solution for real-time graphics. http://www.wpsenergy.com/JayNick/, 2005.

[9] Macromedia. Flash. http://www.macromedia.com/, 2005.

[10] M. Makatchev and S. K. Tso. Human-robot interface using agents communicating in an xml-based markup language. In *Proceeding of the IEEE Int. Workshop on Robot and Human Interactive Communication, RO-MAN 2000*, pages 270–275, Osaka, Japan, 2000.

[11] W3C. Xsl transformations (xslt), version 1.0, w3c recommendation 16 november 1999. http://www.w3.org/TR/xslt/, 1999.

[12] W3C. About svg, 2d graphics in xml. http://www.w3.org/Graphics/SVG/About, 2004.

[13] W3C. Extensible markup language (xml) 1.1, w3c recommendation 04 february 2004. http://www.w3.org/TR/xml11/, April, 15 2004.

[14] W3C. Mobile svg profiles: Svg tiny and svg basic, w3c recommendation 14 january 2003. http://www.w3.org/TR/SVGMobile/, 2004.

[15] W3C. Extensible markup language (xml). http://www.w3.org/XML/, 2005.

[16] W3C. Scalable vector graphics (svg) full 1.2 specification, w3c working draft 13 april 2005. http://www.w3.org/TR/SVG12/, 2005.

Vitae

Riccardo Cassinis got his degree in Electronic Engineering in 1977 at Polytechnic University of Milan., and has worked with that Institution until 1987, as Fellow, Assistant Professor and Research Associate.
In 1987 he was appointed Associate Professor of Robotics and of Numerical Systems Design at the University of Udine. Since 1991 he is Associate Professor of Computer Science and of Robotics at the University of Brescia. He has been founder and director of the Robotics Laboratory of the Department of Electronics of Milan Polytechnic University, of the Robotics Laboratory of the University of Udine, and is now Director of the Advanced Robotics Laboratory of the University of Brescia. After graduation, he has been working for about fifteen years on several topics related to industrial robots, and has then addressed navigation and sensing problems for advanced mobile robots. His last research interest aims at taking advantage of Internet technologies for building robots whose sensing and processing capabilities, rather than being concentrated in a single machine, are distributed over a network, allowing the construction of very simple and small devices.

Fabio Tampalini received his diploma in Electronic Engineering from the University of Brescia, Italy in 2003. He is currently working as a Ph.D. student of Information Engineering at the Department of Electronics for Automation of University of Brescia. His research interesting include fuzzy logic, swarm robots and distribute systems.

```php
<?php
$upload_dir = "./";
$xh = xslt_create();
$inputXML  = $upload_dir . "data.xml";
$inputXSL  = $upload_dir . "data.xsl";
$outputXML = "OutSVG.svg";

if ( xslt_process($xh, $inputXML, $inputXSL, $upload_dir . $outputXML) ) {
 $fo = fopen( "./Tmp-$outputXML", "w" );
 fputs( $fo, '<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>' );
 fputs( $fo, '<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
              "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">' );

 $fi = fopen( "./$outputXML", "r" );
 fgets($fi);
 while (!feof ($fi)) fputs( $fo, fgets($fi) );
 fclose( $fo );
 fclose( $fi );

 echo "<html><head><title>ARCSUS .::. AMIRoLoS</title></head><body>";
 echo '<embed src="./Tmp-' . $outputXML . '" type="image/xml+svg" />';
 echo "</body></html>";
}
else {
 echo "<html><head><title>Error</title></head><body>";
 echo "Error " . $inputXML ./* " in " . $outputXML .*/ " " . $inputXSL;
 echo "<br/>" . xslt_error($xh) . "<br/>";
 echo "Error Code: ". xslt_errno($xh);
 echo "</body></html>";
}
xslt_free($xh);
?>
```

Figure 4: PHP file content converting from XML to SVG, by means of XSLT file.

```xml
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns="http://www.w3.org/Graphics/SVG/SVG-19990812.dtd">

<xsl:output method="xml" indent="yes" media-type="image/svg"/>

<xsl:template match="/">

<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<?xml-stylesheet type="text/css" href="style.css" ?>
<svg width="100%" height="100%" xmlns="http://www.w3.org/2000/svg" version="1.1"

<!-- Colore sfondo totale -->

        <rect class="Background" x="0" y="0" width="100%" height="100%" />

<!-- END Colore sfondo totale -->

        <svg onload="init(evt)" xml:space="preserve" width="100%" height="100%"
                <g transform="translate(0,-15)">
<!-- Intestazione -->

                        <rect class="Background" x="0" y="0" width="800" height=
                        <g transform="translate(0,17)">
                                <a xlink:href="./AMIRoLoS.svg" id="AMIRoLoS">
                                        <rect class="EtichetteON" x="220" y="20"
                                                <set begin="AMIRoLoS.mouseover"
                                        </rect>
                                        <text class="EtichetteON" x="226" y="36"
                                </a>
```

Figure 5: Subpart of XSLT file used to convert from XML to SVG file.

```xml
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1
<?xml-stylesheet type="text/css" href="style.css" ?>
<svg width="100%" height="100%" xmlns="http://www.w3.org/2000/svg" version="1.1"

<!-- Colore sfondo totale -->

        <rect class="Background" x="0" y="0" width="100%" height="100%" />

<!-- END Colore sfondo totale -->

        <svg onload="init(evt)" xml:space="preserve" width="100%" height="100%"
                <g transform="translate(0,-15)">
<!-- Intestazione -->

                        <rect class="Background" x="0" y="0" width="800" height=
                        <g transform="translate(0,17)">
```

Figure 6: Subpart of SVG content finally shown from the browser.