Interconnectable Gadgets and Web Services Usage in Supervisory Control of Unmanned Underwater Vehicles

Tomislav Lugarić*, Antonio Vasilijević* and Siniša Srbljić **
* University of Zagreb, Faculty of Electrical Engineering and Computing/Laboratory for Underwater Systems and Technologies, Zagreb, Croatia
** University of Zagreb, Faculty of Electrical Engineering and Computing/Consumer Computing Laboratory, Zagreb, Croatia
tomislav.lugaric@fer.hr

***Abstract* – Unmanned Underwater vehicles (UUVs) are routinely used for data collection during underwater research missions. UUV operators which perform advanced data collection are usually not qualified for data interpretation. On the fly adaptation of data collection methods based on interpreted data can increase data quality and lower the operator's effort. However, this requires the presence of an expert on site. In order to avoid this, a system of remote monitoring and control over the Internet is proposed. Closing the vehicle control loop over the Internet is problematic due to latency issues, therefore a supervisory control approach is used. This requires only high-level commands to be sent over the Internet while closing the control loop locally. Service oriented architecture (SOA) is used as an API for vehicle monitoring and mission control, while software gadgets are used to display collected data and to send commands for mission adaptation. Gadgets provide support for modifying and displaying data as well as defining and detecting logical conditions. Usage of connectible gadgets as building blocks eliminates the need for expertise in programming languages while increasing the scalability and flexibility of the system.**

I. INTRODUCTION

Remotely operated systems offer advantages in situations where the system is in a remote or hazardous location. Teleoperation is a process in which a human operator directly controls the system from a safe location. Because of significant data flow requirements, teleoperation systems are unpractical for deployment in situations where data flow is limited. Supervisory control is used as an alternative to teleoperation when the conditions do not allow sufficient flow of sensory data. Supervisory control is a method of controlling robotic systems that requires only intermittent operator input. The name comes from the analogy between interaction of a supervisor and his subordinates in an organization and interaction of a person with automated systems[1]. In a supervisory control system, a computer closes the control loop. The data is continually presented to the supervisor, who sends commands to the control computer when necessary. The computer has autonomous control of the variables of the system, and converts high level supervisor commands into low level vehicle commands. Since all time-critical

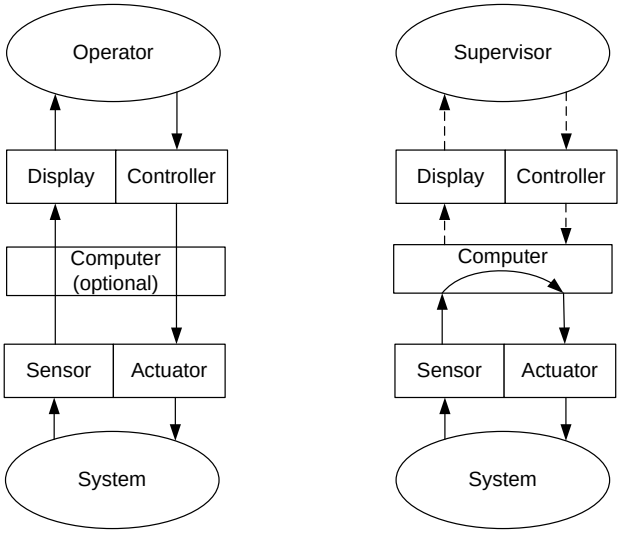

Figure 1: Teleoperation control (left) and supervisory control (right)[1]

operations are done locally on the computer that closes the control loop, a supervisory control system can be implemented over local or wide area networks. Network latency has minimal effect on supervisor interaction since the amount of data exchanged between supervisor and system is smaller than the amount of data needed in a teleoperation system. Figure 1 shows differences between a teleoperation system and a supervisory control system. Solid arrows represent control loops through which a major fraction of control is accomplished and dashed arrows represent the supervisory control loop.

Service oriented architecture (SOA) is a concept of designing and deploying systems over the Internet which takes advantage of loose coupling and high modularity of its components [2]. Entities in a SOA system are divided into two roles: service providers and service consumers [3], which interact as shown on figure 2.

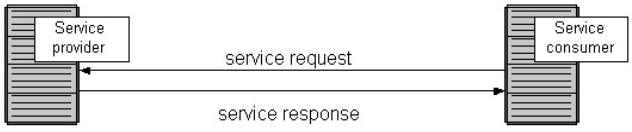

Figure 2: Service provider and service consumer [2]

In order to connect several entities together; they must have standardized interfaces and exchange messages that are in a standardized format. Each entity in the system may act as a service provider and as a service consumer at the same time. Service providers allow consumers to access a specified functionality, similar to function calls in programming languages. The consumer sends parameters to the service provider that processes data and responds to the consumer with the result of the operation. While processing data, the service provider may send data to other service providers for processing assuming the role of a service consumer. Services allow developers to easily expose functionalities of a system over the Internet. A standard for designing interfaces and messages as well as a service directory is defined by the Web Services (WS-*) technology specification [2, 4, 5].

Gadgets, sometimes also called widgets, are small standalone web sites that run inside a gadget container. The concept of software gadgets was first introduced in 1985 by the Commodore company on their Amiga line of computers [6]. An environment called Intuition was responsible for handling user generated events and forwarding them to programs running in background. User event handling and data displaying was handled by elements of the interface called gadgets, which were independent of the programs they communicated with. This prevented the graphical user interface from freezing while the program was running. At present, the term *gadget/widget* is widely accepted a standalone platform independent program running inside a container [7, 8]. Gadgets are used on the Web for customizing interfaces of personal web pages such as iGoogle. Each gadget in a web container implements a specific functionality and acts as an interface to a web based resource. The user can enter parameters in the gadget's interface and receive results of the performed operation. By selecting the gadgets that are relevant to his field of interest the user can design the interface which displays the desired data.

This paper describes a proposal for a supervisory control system that utilizes SOA and WS-* as the underlying communication technology, and gadgets as building blocks of the supervisory and control system graphical user interface (GUI). The Geppeto gadget container [9] which is also described allows the user an additional level of flexibility by allowing automatisation of operations executed in the GUI.

II. PROBLEM STATEMENT

Underwater missions performed by robotic vehicles are an example of a hazardous and remote environment. The main hazard for a vehicle is depth and lack of air for the operator. The vehicles with operators onboard must carry their own reserves of air in order to carry out their missions. To minimize the risk to operators, remotely operated vehicles (ROV) or autonomous underwater vehicles (AUV) are used. In case of ROVs, the operator is located away from the vehicle, typically on a boat, as shown on figure 3. The vehicle and operator's console are connected by a tether, which provides power, control and status data links.

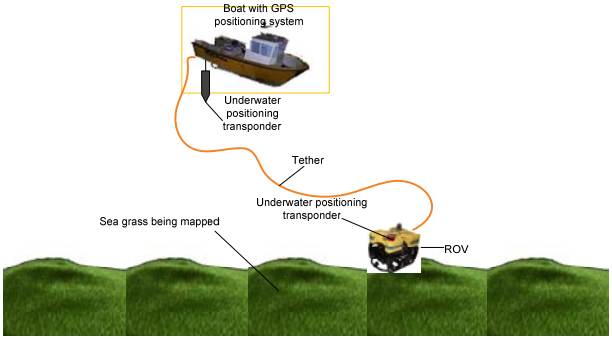

Figure 3: ROV mission

In case of AUVs, the mission is preloaded into the control computer onboard the vehicle which carries out the mission automatically, optionally reporting its status at specified waypoints during the mission. At the end of the ROV or AUV mission, the gathered data is delivered to the experts for further analysis.

Since experts qualified for operating AUVs or ROVs lack the knowledge required for interpretation of the data gathered during a mission, additional expert supervision is typically required for a successful mission. The expert posessing relevant knowledge provides supervisory control to the human operator. The workload of the operator is often lessened by automating low level vehicle commands and simplifying high level ones. An expert could provide all the high level commands, such as depth, heading and speed, without involving the operator except in case of an emergency. In many cases it is not practical to have the expert on site, either because the boat lacks the necessary facilities and space, or because the expert is not available to travel to site. Since the local control loop is already closed by the computer, and only high level commands are necessary for a successful mission, the commands could be issued remotely. In order to achieve this capability, an infrastructure allowing experts to monitor the vehicle remotely and issue commands to it needs to be developed.

A remote supervisory control system would enable the experts to guide a mission without travelling to the site. Since a significant portion of the mission's duration will be uneventful, the experts may wish to be notified only if specific conditions occur. When monitoring parameters received from the vehicle the expert may want to perform some additional processing or data fusion before the data is displayed. These tasks are not difficult to accomplish using programming languages or specific tools like MATLAB, but this approach requires the expert to be proficient in programming. In order to allow experts to define and handle events or to process data without requiring them to be proficient in programming a simple but powerful tool is required. The tool needs to offer functionalities comparable to programming languages while utilizing only the operations that are common to any user familiar with graphical user interfaces of current operating systems.

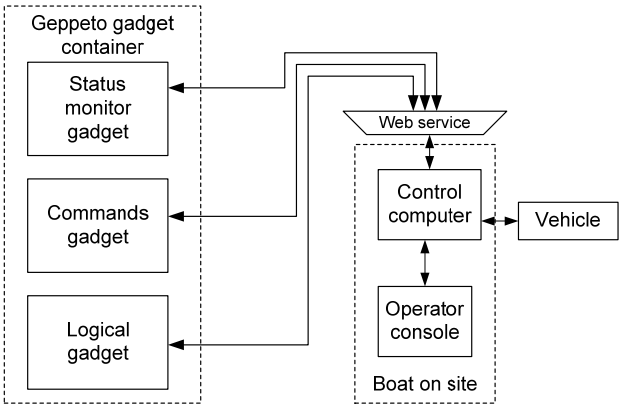

Figure 4: System architecture

III. System architecture

The proposal described in this paper is a flexible distributed supervisory control system for underwater missions. The system has three key elements: the underwater vehicle, the on-site support infrastructure and the gadget container with gadgets repository.

The underwater vehicle is the working component of the system, because it is the source of all data that is obtained and the sink for all commands sent. It can be an ROV or an AUV. If the vehicle is an ROV, it is permanently connected to the control computer which is typically located on a support boat. If it is an AUV, the connection is only available while the vehicle is on the sea surface and within range of the control computer's wireless network. To enable an AUV to deliver data and receive mission changing commands during its mission, the AUV is programmed to periodically surface in order to establish communication with the control computer. AUVs have an additional onboard computer which closes the control loop. In essence, AUVs already have a supervisory control system in place, and the proposed system represents an extension of the existing system over the Internet.

The on-site support infrastructure consists of the control computer and the operator's console. The control computer has a communication link which connects to all sensors, the vehicle and all additional equipment on site. It also hosts web services which allow clients to read data from specific pieces of equipment and send commands to it. Every piece of equipment is visible to the client as a set of variables that it provides and as a set of commands it accepts. The web service is connected to the communication link in the same way as other pieces of equipment. The operator's console may be connected to the vehicle through the computer, or vice versa, depending on type of hardware used. It is normally not used in the system, but it needs to be configured in such a way that it overrides any other commands in the system. This way, the operator can take full control of the vehicle and drive it to safety if an emergency arises.

The gadget container and the gadgets are the expert's control panel for the vehicle. Gadgets are grouped into three groups: status readers, logical gadgets and command gadgets. Status readers fetch data from the vehicle, command gadgets send commands to the vehicle and logical gadgets detect when specified logical conditions are realized. A special TouchMe gadget deployed in the container provides support for automatization of operations performed on the gadgets' GUIs.

IV. Exposing vehicle functionality through web services

The web service deployed on the control computer determines how the clients perceive equipment on site. The developer decides which functionalities to expose over the Internet and which ones to reserver for local control. For maximum modularity of the system, the exposed functionalities should have a certain level of uniformity. Every piece of equipment can offer different functionalities and data sets, but a core set of basic functionalities must to be established. All pieces of equipment can be placed in two groups: vehicles and sensors.

The core vehicle dataset should include position data, vehicle status and mission status. The core position data may consist of the vehicle's depth, latitude, longitude, roll, pitch, yaw, forward speed (surge), sideways speed (sway) and vertical speed (heave). It is safe to assume at least some of this data is always available because it is essential to closing the vehicle's control loop [10]. Any positional data that is not available may be safely set to a predetermined value (such as zero) because it is not applicable to the vehicle in question. For example, it is safe to assume that an ROV with no roll control will always have roll value of 0, or that a torpedo-type AUV will only have a forward speed. The vehicle may provide additional position data such as altitude from bottom, quality of positional fix, current positioning method in use or similar.

Vehicle status can only have a small standardized portion because it is highly on the vehicle's hardware. The core vehicle status should consist of warning messages such as leak, short circuit or general failure. Each warning message should include a description of the malfunction and an indicator of the malfunction's severity. The vehicle may provide any additional data available from its internal or external sensors (water temperature, salinity, battery status and thruster settings).

Mission status dataset gives information about the state of the mission a vehicle is undertaking. The core mission status should include a mission running indicator, type of action being carried out (station keeping, traveling, parked etc), current waypoint the vehicle is traveling to, distance and relative bearing to the waypoint, desired depth, speed, heading and mission running time. Additional data the vehicle provides depends on the mission management featured build into the control computer. Since the mission management software is not necessarily dependent on the vehicle, the mission status core dataset will likely be expanded to include additional status data added in the future.

The core command set includes high level commands controlling the vehicle's mission, similar to an autopilot system. Simpler commands include heading, depth and altitude changes, while more complex ones may include

waypoint changes, sequences of preprogrammed maneuvers or triggering of missions that have been preloaded into the control computer. In case the vehicle is outfitted with additional advanced guidance subsystems, commands for those systems may be included in the additional command set.

Sensors have a significantly smaller dataset and command set. The core dataset for sensors is simply an indicator of their operation and the data they record in any appropriate format which may include necessary metadata. The core dataset is simply a command to turn a sensor on or off.

The client must have the option to receive only core datasets or to receive additional data from the vehicle or sensors. The presence or absence of additional datasets must not interfere with the core data set.

When exposing the on site equipment through web services, the functionalities exposed to clients should be grouped according to the piece of equipment they are related to. Each piece of equipment should be mapped to a single web service. The web service will have at least two operations for reading data – one for the core dataset and one for the additional dataset. Each command will be mapped to a different operation. With command operations, care needs to be taken to avoid conflicts in case different clients attempt to control the vehicle at the same time as well as preventing malicious usage of commands. This is achieved by assigning each client one of three authorization levels: observer, supervisor and master supervisor. Each client is initially given the observer role. An observer may read data from the vehicle, but any commands sent from an observer are ignored by the web service. An observer may obtain the supervisor role for a specified amount of time by sending the appropriate request. Only one client may receive the supervisor authorization at a time. The master supervisor role is predefined in the system, and is typically reserved for the owner of the equipment. If needed, there may be more master supervisors in the system. A command issued by the master supervisor overrides any commands issued by supervisors. Additionally, the master supervisor may terminate the current supervisor's authorization, demoting the client back to observer.

The described web service allows multiple clients to provide supervisory control over the equipment using the Internet. An authorization mechanism ensures that there are no conflicts between multiple supervisors, and the role of master supervisor allows handling of emergency situations remotely. As a failsafe, the operator on site can interrupt the supervisory control loop at any time.

V. GADGET BASED MONITORING AND CONTROL

The web service which exposes the on site equipment over the Internet provides a powerful API. To utilize a service the user must design an application that interacts with it. The target audiences of the proposed system are experts in fields other than programming, therefore providing an API is not sufficient. In order to allow the users to interact with the equipment, a set of gadgets serves as endpoints of the web service. The user can load the gadget in a compatible gadget container and interact with the equipment through them. Three distinct groups of gadgets are defined according to their functionalities: data readers, data processing & visualization, command senders.

Data reader gadgets allow the user to read data the equipment provides. The data is typically in numerical form but may include text or images. The operator of the equipment provides the gadgets which connect to his web service, ensuring they provide at least the core dataset.

Data processing & visualization gadgets do not necessarily depend on the operator of the equipment. They can be provided by any third party, or can be deployed by the user. These gadgets can allow data handling operations such as selecting elements from tables, merging tables together, drawing charts or maps or data processing such as simple mathematical operations or statistical calculations. A special type of gadget is the logical gadget. The logical gadget can be used to define and detect specific conditions. When those conditions happen, events are generated in the system which can be used to trigger operations.

Command sender gadgets allow the user with a supervisor role to control the equipment. The operator of the equipment should provide them. Each piece of equipment must have the accompanying gadgets that accept the core command set. The operator may provide additional gadgets for the additional command set. Command sender gadgets must handle client authorization, which can be integrated into the gadget container.

Since the reader gadgets provide only raw data, they are not practical for vehicle monitoring as such. To visualize data the user relies on data processing & visualization gadgets. This requires data to be transferred between them, presenting a problem if the data is not standardized. Data reader gadgets must provide data in a standard format, and its specification must be available to all developers of additional gadgets. This way, connection of multiple gadgets is allowed. The user can copy the received data from a data reader and paste it into a graphing gadget. The graphing gadget then draws the data as a chart. The user can examine the chart and make decisions about the mission. If needed, the user can copy data from another sensor into a command gadget and send the command to the vehicle on site (e.g. new course).

By utilizing multiple gadgets, the user can design custom data flows. Flows can be utilized to simply monitor data and make observations or to modify the mission on the fly. The user can perform tasks using operations such as copying, pasting, clicking buttons, selecting options from drop down lists or clicking checkboxes. These operations are common on the Internet and in graphical operating systems, therefore this mode of programming is more understandable to users than coding in a programming language [8]. The standard data format ensures that any interested party may develop new gadgets to be connected with existing ones.

VI. GEPPETO-BASED AUTOMATION

Combination of multiple gadgets is a highly modular and easily understandable way of building applications. While performing an operation while utilizing multiple gadgets, the user typically performs a fixed sequence of simple operations on their interfaces. The same sequence is repeated each time the operation is carried out; therefore a tool to automate execution of such sequence is required. Additionally, if the user wishes to group a set of gadgets together permanently, the tool should provide a way of defining a new composite gadget which would serve as an interface to the permanently connected ones.

An example of such system is Geppeto (Gadget Parallel Programming Tool), developed in the Comsumer Computing Laboratory [11] at the University of Zagreb, Faculty of Electrical Engineering and Computing (FER). The system utilizes a gadget container based on Apache Shindig to render gadgets. The same container is used on sites like as iGoogle, LinkedIn and Yahoo! On a webpage with a Geppeto [9] container, the user can load gadgets from the built-in repository or by entering a URL from which to load them. Designing the user interface of a composite gadget and logic programming can be done via the right click menu which can be brought up when the mouse pointer is over a user interface element. Firstly, the user adds a blank gadget. The added gadget is automatically marked as the active one. In case multiple composite gadgets exist on the currently loaded page, the user must select the correct gadget before adding elements. All other gadgets can be used as sources of user interface elements for the new composite gadget. To add an element to the composite gadget, the user right-clicks the element, selects “add” and the element is copied into the composite gadget. The right click menu also has the option of removing a user interface element. When programming the logic behind the composite gadget’s interface, the right-click menu is used to specify actions. A sequence of actions is generated by selecting the “When clicked” action on an element and then defining actions on GUI elements on other gadgets from the right-click menu. The same sequence is repeated each time the element is clicked. All sequences the user generates can be viewed and reorganized in a table which is stored together with the generated composite gadget. In addition to click-driven gadgets (TouchMe), the system also offers time driven (TickMe) and event driven (TriggerMe) gadgets. Composite gadgets can be used as building blocks in new, more complex gadgets. Utilizing the Geppeto system the user can create complex applications with similar functionalities to those designed in traditional programming languages. The benefit is that the user does not need to be proficient in programming, and uses only operations familiar to him while designing applications.

Geppeto allows users to design applications that continuously monitor equipment located on site and alert them if an event of interest happens. For example, the user can design an application to issue a notification when the vehicle passes over a specific point of interest. The user can then take control of specific sensor or the entire vehicle while it is in the area of interest and adjust it to get the data he wants. Using the same concepts, the users can program the vehicle to perform entire missions and even execute special maneuvers in case of an emergency.

VII. EXAMPLE USE CASE

The following use case explains a survey mission carried out by an unmanned surface vehicle (USV). The USV is equipped with down-facing cameras, side scanning sonars, a sub-bottom profiler and a probe for measuring biological parameters. The USV is wirelessly connected to a computer in a base station located on shore, its functionalities exposed via web services. The mission area has several potential mine sites, a few interesting marine habitats and a preserved medieval wreck. There are also a few dangerous shallows nearby. The USV is initially programmed to perform a sweep of the whole area using only the cameras and side-scanning sonars. Three experts have expressed interest for the mission: an archaeologist, a biologist and a navy official.

Each of the experts was given access to a gadget container and a gadget repository containing gadgets that interface to the vehicle. The experts have each designed their own gadget-based applications shown below. Figure 5 shows the biologist's application. In the application, a gadget loads data for crude oil in water and chlorophyll levels. The reader gadget is driven by ticks provided by Geppeto TickMe gadget. In each iteration the system will check if the crude oil concentration is above normal or chlorophyll is below normal. If there is too much oil, the system will order the vehicle to take a photo of the sea bottom, and if there is not enough chlorophyll, the system will order the vehicle to perform a scan for all biological parameters in a 10 meter circle. Additionally, the system will continuosly draw a graph of those two values.

Similarly, the archaeologist’s application on figure 6 continuously loads vehicle’s position. It is preprogrammed with the location of the medieval wreck, and takes photos and sub-bottom scans while in proximity of the wreck.

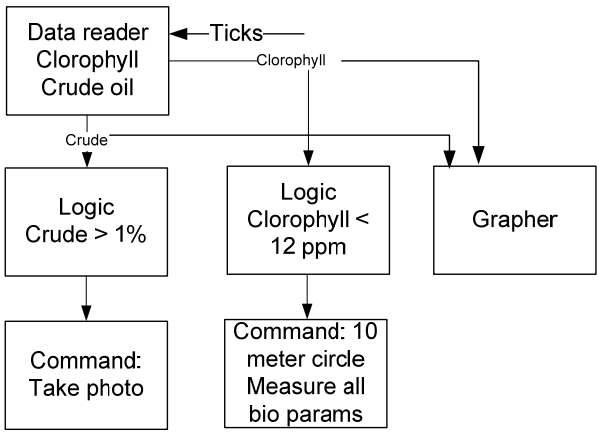

Figure 5: Biologist's application

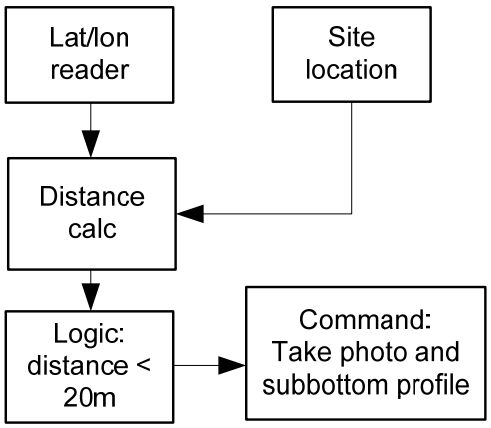

Figure 6: Archaeologist's application

The navy official's application constantly checks the side scanning sonar for objects larger than 2 meters and sends images of such object via email.

These three applications allow three users to benefit from the same mission. This greatly improves the efficiency of the vehicle and increases the value of obtained data. The owner of the equipment may monitor the vehicle in his own application and terminate the mission if he feels that any of the supervisors is misusing the vehicle.

VIII. CONCLUSION

The system proposed in this paper is designed to simplify usage of unmanned marine vehicles and to facilitate remote controlling over distances. The benefits for the owner of the vehicle include easier mission management, higher value of obtained data, faster and more flexible reporting. Because experts can provide supervision as the mission is progressing, efficiency of the missions is improved compared to the usual approach of performing the mission and providing data to experts upon completion. Since experts do not need to travel to the site of the mission, this system potentially reduces costs of such missions. The owner can easily protect the vehicle from misuse and always know what mission it is performing. Using the proposed system, experts have the ability to form their own reports and to adapt them to the current mission as it is being carried out. The ability to automate some of their supervision process reduces the effort needed for a successful mission and the possibility to tightly cooperate with on site teams increases quality of obtained data.

ACKNOWLEDGMENT

The work was carried out in the framework of a Coordination and Support Action type of project supported by European Commission under the Seventh Framework Programme "CURE – Developing Croatian Underwater Robotics Research Potential" SP-4 Capacities (call FP7–REGPOT–2008–1) under Grant Agreement Number: 229553.

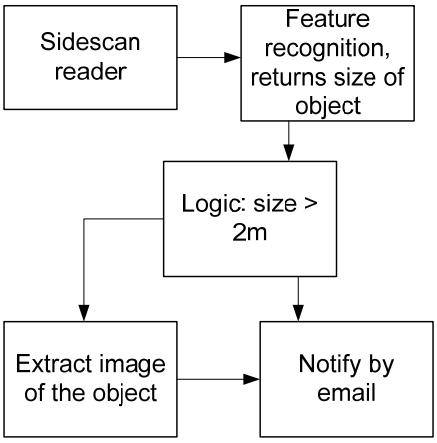

Figure 7: Navy official's application

We would like to thank Miroslav Popović from Faculty of Electrical Engineering and Computing at University of Zagreb, Vjekoslav Mesaroš and Jurica Štimac from faculty of Economics and Business at University of Zagreb for providing valuable input during development of prototype data handling and logical gadgets. We would also like to thank Đula Nađ and Zvonimir Pavlić from from Faculty of Electrical Engineering and Computing at University of Zagreb for ongoing support in development of a standard for vehicles, sensors and gadget integration.

Finally, we thank all members of Consumer Computing Laboratory (CCL) and Laboratory for Underwater Systems and Technologies (LABUST) from Faculty of Electrical Engineering and Computing at University of Zagreb for making work on this paper possible.

REFERENCES

[1] Sheridan, Thomas B., "Telerobotics, automation and human supervisory control", MIT Press, 1992.

[2] Hao He, What is Service Oriented Architecture; 31. 3. 2006; http://pesona.mmu.edu.my/~wruslan/SE2/Readings/detail/Reading-28.pdf; 24. 1. 2011.

[3] Douglas K. Barry, Service-Oriented Architecture (SOA) definition; http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html; 24. 1. 2011.

[4] Douglas. K. Barry; Web Services Explained; http://www.service-architecture.com/web-services/articles/web_services_explained.html; 24. 1. 2011.

[5] W3C, "Web Services Activity", http://www.w3.org/2002/ws/, 24. 1. 2011.

[6] Mical, R. J., Deyl, S., "Amiga Intuition Reference Manual", Addison – Wesley Pubnishing Company, 1986.

[7] Srbljić, S., Škvorc, D., Skrobo, D., Widget – Oriented Consumer Programming, Autimatika 50(2009), 3-4, str 252-264

[8] Škvorc, D., "Consumer programming", PhD thesis, , University of Zagreb Faculty of Electrical Engineering and Computing

[9] Geppeto Home Page, http://www.ris.fer.hr/, 24.1.2011.

[10] Fossen, Thor I., "Guidance and control of ocean vehicles", JohnWiley and Sons, 1999.

[11] CCL home page, http://ccl.fer.hr, 24.1.2011.