User interface integration and remote control for modular surgical assist systems

Release 1.00

Stefan Bohn, Tobias Hilbert and Oliver Burgert

June 24th, 2010

Innovation Center Computer Assisted Surgery (ICCAS), University of Leipzig, Germany

Abstract

Today's operating rooms consist of numerous medical devices, clinical IT systems and systems for computer assisted surgery. The majority of these systems comes from different vendors and has different user interface designs and interaction schemes. This diversity introduces a high risk of accidental misuse, which is critical in the surgical domain. The proposed framework integrates user interfaces of heterogeneous components into one central control console using a uniform screen design and interaction scheme. To accomplish this, the standards Universal Remote Console (URC) and Device Profiles for Web Services (DPWS) have been combined and integrated. The prototype results are demonstrated in the ICCAS integrated operating room.

Contents

1 Introduction

In today's operating rooms (OR) numerous medical devices, clinical IT systems and systems for computer assisted surgery (CAS) support clinicians during surgical interventions. A large range of information is provided by these systems and devices, which usually are spatially distributed within the OR. This often leads to unergonomic conditions and an impaired surgical workflow [1].

Except in commercial integrated OR suites, the majority of today's clinical IT and CAS systems come from different vendors and are not networked or integrated in an appropriate manner. Thus, each of these systems has its own dedicated user interface design and often completely different operating schemes. This diversity has to be controlled and supervised by medical staff, which is expected to operate these disparate systems properly under every condition during surgery, while at the same time their primary focus should be the patient. A study by Matern et al. revealed that 70 % of the German surgeons and 50 % of the nursing personnel report difficulties regarding the handling of medical devices [2]. Furthermore, 40 % of all interviewed clinicians claimed that they have experienced multiple situations of potential danger for staff or patients. Several studies show that a large magnitude of errors result from insufficient, unclear, or complex human machine interfaces [3, 4].

Systems integration of medical devices and IT systems with centralized control of the integrated OR has been recognized for its potential to overcome the aforementioned issues and to increase the overall surgical efficacy, ergonomics and the clinical workflow.

At ICCAS Leipzig an open standards based OR integration framework has been designed, which is modeled as component-based service-oriented architecture [5]. Each medical device or CAS component is integrated as independent module and interconnected through an Ethernet network. The integration framework "TiCoLi" acts as middleware and facilitates service discovery, time synchronization, system diagnosis, messaging and event handling as well as streaming of continuous signals. Several core components such as central user console, managing and supervision module, COM server, context module, OR database, and session repository form the basic functionality of the integrated OR system.

To goal of this work is to define a generic framework, which enables heterogeneous components to seamlessly transfer their user interface (UI) to a central control console, where the UI is rendered in a uniform manner according to defined rules, the user's preferences and optionally a specific language. Using the central control console, the clinical user shall have control over the entire functionality of the integrated OR system as well as access to all relevant information. The central control console unifies the plurality of heterogeneous OR appliances into one uniform and clean user interface design with one dedicated interaction scheme.

2 Remote User Interface Technologies

Remote user interfaces (UI) are a method to render and interact with a user interface on other systems than those that execute the application logic. Current remote UI technologies can be categorized into two classes: 1st) graphics-level/framebuffer protocols and 2nd) markup language based protocols.

Remote UI methods based on graphics-level/framebuffer protocols try to create an exact copy of the pixel representation of the target systems user interface. Today, the most established technologies are protocols that act on graphics-level, e.g. X-Windows or Citrix-ICA as well as remote framebuffer based protocols such as Virtual Network Computing (VNC). Those protocols usually generate a certain amount of network load. Furthermore, they are not as flexible regarding the display size, ratio, or color depth of the output screen, especially when small or mobile devices are used as remote control.

Markup language based protocols describe the user interface in an abstract form based on a generic syntax and semantic. Using a certain transport mechanism, the target specific markup language file is transported

to the remote control system where the user interface is rendered on-the-fly according to the specific system properties and/or user preferences. The most common markup language based protocols today are XML User Interface Language (XUL), User Interface Markup Language (UIML), Extensible Application Markup Language (XAML), Universal Plug and Play (UPnP) as well as the Universal Remote Console (URC) standard. Those protocols mostly differ in the underlying transport mechanisms between target and remote control, the functionality of the overall framework they are embedded in as well as the render engines for the UI generation.

3 Framework

The proposed framework has three major parts and separates user interface representation, interaction scheme and physical design of the central control console from the integrated components and the network backend services (Fig. 1). Each component being controlled from the central remote console acts as a target. Targets describe and present not only their user interface in an abstract form but also their functionalities, constraints of use and the socket/protocol, which facilitates network access to the target. The central control console does not have any a priori knowledge about particular targets. This creates high flexibility and extensibility since UI modifications or updates require only moderate changes in the target and are instantly reflected at any central console. Using the service discovery and control session capabilities from the network protocol, the central control console is able to detect and connect to targets, retrieve the description documents, render the user interface according to defined rules, and to respond to user inputs.

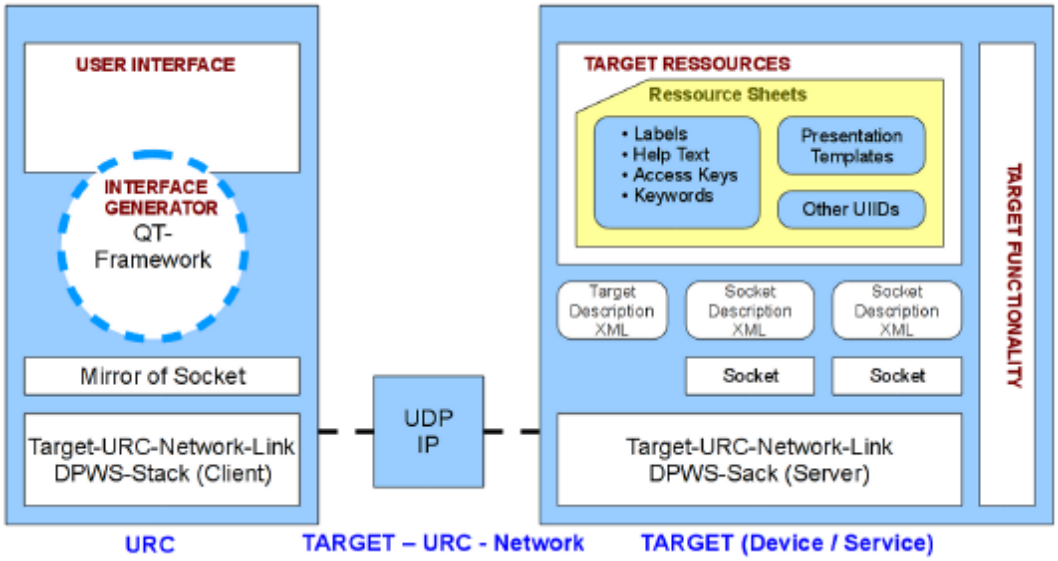

Figure 1: The framework for user interface integration and remote control of integrated components consists of three main parts: The target component to be controlled, the network as transport layer and the remote control console.

The basic technologies used in this framework are primarily based on two standards:

1) The overall process of user interface integration and remote control of targets, the syntax and semantics of exchanged data as well as the functional capabilities of central remote consoles are defined in the ISO/ANSI standards 389-2005 to 393-2005 - **Universal Remote Console (URC)**.

Communications between targets and the central remote console take place over a network, the Target-URC Network (TUN). The URC standard does not specify a particular network protocol to be used but rather defines (minimum) requirements such as service discovery and management of control sessions for TUN protocols.

2) A very promising network protocol, which fits these requirements, is the **Device Profiles for Web Services (DPWS)** standard [6]. DPWS is an IP based network protocol, which is based on the web-services architecture and provides means for discovery, events, metadata exchange and security. DPWS is generally light weighted and specifically tailored to low power and embedded systems with limited resources.

The overall framework is implemented in C and C++ and based on standard PC platforms. In this work URC and DPWS have been combined for the first time and are integrated into our OR integration framework.

3.1 Device Profiles for Web Services based Target-URC-Network (DPWS-TUN)

The basic framework used to realize the DPWS Target-URC-Network functionality is the open source stack DPWSCore of the SOA4D initiative [7]. A generic WSDL (Web Services Description language) file has been created, which defines the basic URC services: URCMessage, URCEvent and URCDocument. Additional tools such as wsdl2h and soapcpp2 are used to generate the skeleton and stub source codes for the client and server implementation. Several wrapper classes have been implemented to integrate the C DPWS stack with the URC functionality as defined in the URC-HTTP technical draft [8]. The overall DPWS-TUN functionality is realized as generic C++ library which is used in URC targets and the central control console.

3.2 URC Targets

URC Targets need to provide three XML description files (Target Description, Socket Description, and Presentation Template) that describe their functionality and user interface layout with (optionally multi-language) labels according to the URC standard. A separate URC-XML generator software with graphical user interface has been implemented, which supports the user in specifying the user interface elements, variables, and commands as well as specific layout rules. The URC-XML generator software then generates the standard conform XML files. Using the DPWS-TUN library the main functionality of the DPWS stack is available. Application specific methods need to be implemented to react to `URCMessage` or `URCEvent` requests from central control consoles such as get/set value or call method.

3.3 Central Control Console Application

The central console application is realized as widget engine based on the Qt application and UI framework [9] and realizes four major functionalities:

1. Discovery

A discovery process monitors the network for the appearance or disappearance of targets. Specific service types and scopes have been defined for the DPWS implementation, which limit the results of the discovery process to URC targets. A continuously updated list of all available targets is displayed on the screen where the user can select the target that should be controlled.

2. Description Retrieval

Upon user selection the application connects to the target through the TUN using the given socket details and requests the XML description files using the URCDocument service: Target Description, Socket Description, and the Presentation Template.

3. User interface generation

The presentation template describes the principal appearance of the user interface and defines the layout and optionally a grouping of UI elements. There are several basic types for UI elements such as text area, input, selection, trigger, output, modal dialogs, etc. A decision tree realizes the mapping of URC UI elements to specific Qt widgets (e.g. “text area” to QLabel). Finally, the user interface is rendered on-the-fly according to the layout and grouping guidelines. Variable values are retrieved from the target using the information provided in the socket description. With cascading style sheets (CSS) Qt provides a very flexible method to adapt the appearance of the user interface regarding colors, fonts, widget size etc.

4. User interaction

User interaction in Qt applications is realized using the Qt signal-slot-mechanism, which generates signals for any actions the user performs on the user interface. The events are mapped to `URCMessage` and forwarded to the corresponding target, which runs a listener-thread to react on any request. The central console application itself listens to `URCMessage` and `URCEvent` on the TUN interface, to react on incoming messages and to update the corresponding UI elements.

4 Results

The proposed framework enables spatially distributed and interconnected CAS components to describe their user interface using a standardized syntax and semantic. The UI description is transferred to the central control console, where the clinical user has control to the remote functionalities. The separation of UI presentation, interaction, and application logic has several advantages. The user interfaces of heterogeneous components appear within a uniform design, layout, and interaction scheme, which reduces the overall complexity for operating multiple systems. This approach provides enough flexibility for different vendors to create remote consoles with appropriate interfaces, input/output devices as well as specific corporate designs. At the same time vendors of CAS target components may focus on the actual application logic and functionality rather then on UI specific details.

Device Profile for Web Services (DPWS) meets all requirements for the Target URC Network protocol. Components with limited resources or embedded systems may also be integrated due to the light weighted nature of DPWS.

The creation of new targets using the developed DPWS-TUN library requires only moderate demands regarding implementation. The developed URC-XML generator software (Fig. 2) supports the user in creating standard conform XML description files for URC targets. The user can add variables, labels, commands, assign data types and define basic layout rules such as grouping of user interface elements.

Several prototype target components have been implemented, ranging from simple OR light controls to targets for neuro-navigation software. The remote UI application is implemented in C++ using the Qt Toolkit. The look and feel of the central control console (Fig. 3) is highly flexible due to easy configurable stylesheets. The list of available targets is displayed in a sidebar. Selecting a target renders the target specific user interface on-the-fly.

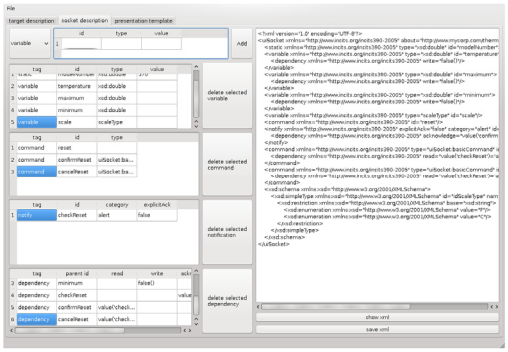

Figure 3: URC-XML generator software to support developers in creating new URC targets.

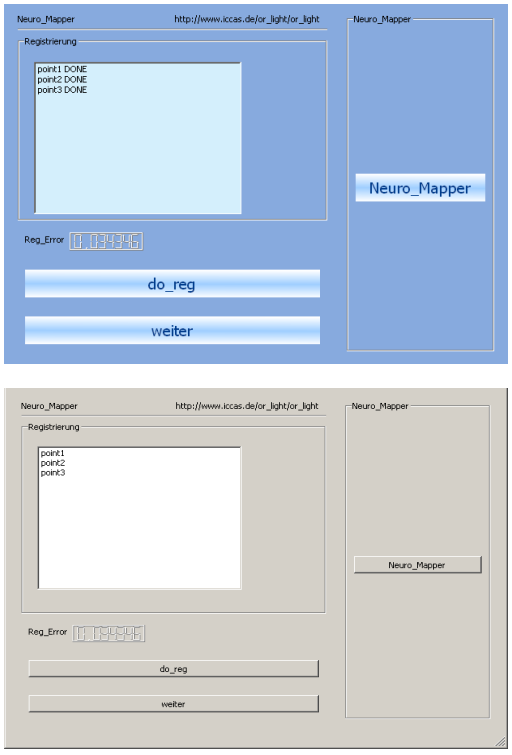

Figure 2: Example of remote user interface for a neuro-navigation system with different designs using style sheets.

The remote console software has been embedded into the TIMMS OR central console application (Fig. 4), which provides the surgeon access to the functionality of the integrated system, overview of the system status as well as access to preoperative planning data and acquired intratoperative data, e.g. screenshots from the video routing system.

Figure 5 shows the integrated TIMMS OR at the ICCAS lab. The surgical control console consists of ceiling mounted displays, which can be positioned close to the surgical situs. The displays consist of the master view and the central control console application. The master view can display any video source in the OR using video routing technology. A special control panel has been developed, which consists of a 7" touch screen and can be covered with sterile foil. The control panel gives the surgeon access to the video routing system and the remote user interfaces of the integrated components. Thus the surgeon is able to control any function of the integrated OR system within the sterile field.

Figure 4: Central console application with remote device control, system status and access to intervention related data sets.

Figure 5: Protoype integrated OR with ceiling mounted displays and TIMMS control panel (lower-middle) as user interface within the sterile field.

5 Conclusion

Within the proposed framework the standards Universal Remote Console and Device Profiles for Web Services have been successfully combined to facilitate uniform user interface integration and remote control of modular surgical assist systems. An appropriate user interface that meets the needs and preferences of the target user is a striking criteria concerning successful user interface design system. The architecture of the proposed framework offers high flexibility for the creation of custom-tailored solutions while at the same time the whole framework is based on established standards.

Acknowledgments

The Innovation Center Computer Assisted Surgery (ICCAS) at the Faculty of Medicine at the University of Leipzig is funded by the German Federal Ministry for Education and Research (BMBF) and the Saxon Ministry of Science and Fine Arts (SMWK) within the scope of the initiative Unternehmen Region with the grant numbers 03 ZIK 031 and 03 ZIK 032.

References

[1] Lemke HU, Ratib OM, Horii SC, Workflow in the Operating Room: Review of Arrowhead 2004 Seminar on Imaging and Informatics. Proc. SPIE Conf. on PACS and Imaging Informatics, San Diego, 2005.

[2] Matern U, Koneczny S, Scherrer M, Gerlings T, "Arbeitsbedingungen und Sicherheit am Arbeitsplatz OP / Working conditions and safety in the operating room“, Deutsches Aerzteblatt, Vol. 103, No. 47, pp. A-3187 - A-3192, 2006.

[3] Backhaus C, "Defizite durch eine unzureichende Gebrauchstauglichkeit (Deficits resulting from insufficient usability)" in Usability-Engineering in der Medizintechnik, Springer: Berlin Heidelberg, pp. 21 - 28, 2010.

[4] Dain S, "Normal accidents: human error and medical equipment design", The Heart Surgery Forum, Vol. 5, No. 3, pp. 254-257, 2002.

[5] Bohn S, Michael G, Franke S, Voruganti A, Burgert O, An integrated OR system based on open standards. The MIDAS Journal - Systems and Architectures for Computer Assisted Interventions; 12th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI), London, 2009.

[6] OASIS Web Services Discovery and Web Services Devices Profile (WSDD) Technical Committee. Oasis web services discovery and web services devices profile (ws-dd). http://www.oasis-open.org/committees/ws-dd/charter.php. Last visited: 2010-06-30

[7] DPWSCore-Community. The dpws core project. https://forge.soa4d.org/projects/dpwscore/, Last visisted: 2010-06-30

[8] URC-HTTP Protocol 2.0 (DRAFT). http://myurc.org/TR/urc-http-protocol2.0-20091103, Last visited: 2010-07-01

[9] Nokia, Qt Application and UI framework. http://qt.nokia.com/, Last visited: 2010-06-30