Software Enginering Technique For Modularity Property In Component-Based Software Architecture

Fabrizio Boriero[1] ,Marta Capiluppi[1] , Paolo Fiorini[1] , Roberta Perrone[2] , Elena De Momi[2] , Giancarlo Ferrigno[2]

[1] University of Verona
[2] Politecnico di Milano

1 Introduction

Robots are widely used in surgical rooms of the main hospitals. The most common surgical robot is the Intuitive "Da Vinci" but other new high-tech devices are spreading to help the surgeons in their medical tasks. Such kind of devices are useful to decrees the surgeons physical and psychological stress and increasing the overall safety. The surgical room is a complex environment with a lot of heterogeneous devices made by different producers. Currently the devices work independently but in order to increase the functionalities to give to the users is necessary to think how to connect all of them.

For this reason in this paper we propose a methodology that comes from the *Eurosurge European project* (FP7-ICT-20) that shows a workflow that allows to build a surgical robotic software architecture that respect three properties: modularity, re-usability and safety.

To allow the cooperation between software modules it is important to define the functionalities of every module and how it is possible to integrate it into the global software architecture. In this paper we present a method that starts from the ontology of the components and design the structural model that allows an easy and modular integration. The *modularity* property is the capability of changing the connection between software modules without stopping or recompiling them. Sometime industrial or research partners want to exchange the software modules or use the same module in different tasks without a long and tedious integration phase. For this reasons a good software architecture has to follow the *reusability* property.

Surgical ecosystem, as the industrial one, has as primary characteristic the need of *safety* so it is important that the software architecture provides the real-time capability and the supervision capability.

A huge amount of robotic software architecture that allows the connection between robots and sensors were presented in past [1] but not all of this can achieve our properties. The Component-based software engineering [2], which is the base of robotic software architectures like ROS [3],OROCOS [4] and RT-Middlewere [5], is the technique chosen for our project due to its flexibility. These architecture, in-fact, allow and easy integration between the software modules

made by different vendors and the possibility of changing the connection topology between the software components without recompile everything.

To allow the *reusability* property it is necessary specify the characteristic and the role that the components have during the surgical or industrial case. An Ontology is a "specification of a conceptualization" [6] that can be used to formalize the workflow of a surgical procedure.

In our workflow, we use two different ontologies, with two different levels of granularity:

- The first ontology, named *TaskOntology*, will describe the considered task. It contains concepts related to all the possible states, transitions and commands that the component can assume. It could also contain information about the level of risk associated to the use of a component and related safety issues.
- The second ontology, named *ComponentOntology*, describes the set of components that participate to the scenario. All the components will be described by a set of property like inputs, outputs and specifications.

Once the components are described it is necessary to translate them in a model [7] that defines the interface of the software component (input/output ports, properties). The idea is that if we define a general description, we can assure that all the software components generated by different developers but from the same model can be integrated in the task assuring the reusability property.

The "TaskOntology" can be translated into a sequence of actions and automatic decision (i.e. a Finite State Machine) and connections between the components (architecture topology) that can change during the task, as required by the modularity property.

2 Case study

During the "Eurosurge" project, we chose a simple case study to show how it is possible to apply the method developed. A needle go to a pre-defined position and two tracker log the data into a text file. During the task it is possible that one tracker has a problem so we want to exchange it with another one. Fig 1 shows the hardware devices that compose the case study. To implement the two ontologies described in the previous section we used the tool "Protegé" and we saved the information into a OWL file description. We developed a tool, called "Oromodel", that allow to translate the "ComponentOntology" into a XML model file and automatically generate the software component skeleton for the OROCOS/ROS ecosystem. In general it is very easy to use the same XML model file descriptor also for other component-based architectures.

The OWL file that describe the "TaskOntology" was translated into a finite state machine and used to coordinate the surgical task.

Fig 2 shows that during the case of study, the tracker was exchanged and correctly integrated into the setup thanks to the description made by the component structural model.

Fig. 1. In this picture is possible to see the devices that compose the test case. There is one robot that move the surgical needle and two trackers that save the trajectories

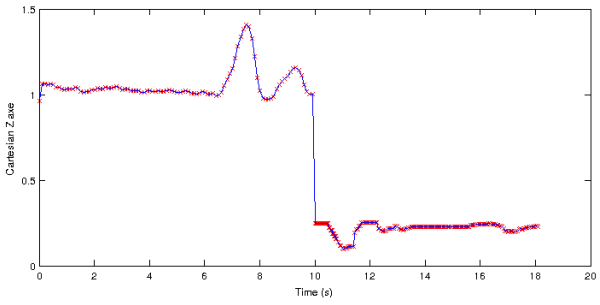

Fig. 2. In this picture there are the data logged by the trackers. It is possible to see that it was possible to change one broken tracker without stopping the whole architecture

3 Future works

The method proposed in this document allows to design a robotic software architecture allowing the re-usability, modularity and safety properties. During the project "Eurosurge" was done a first trial of connection between the Ontology, software modelling and real-time robotic software architecture. It is important to extend this road applying this technique to other contexts like mobile robotic, cyber-physical systems or industry. It is also important to find new modelling systems that allows to formally verify the desired properties.

References

1. Mohamed, N.: Middleware for robotics: A survey. Robotics, Automation and ... (Ram) (2008) 736–742
2. Heineman, G.T., Councill, W.T.: Component-based software engineering: putting the pieces together. Recherche **67** (2001) 2
3. Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., Ng, A.: ROS : an open-source Robot Operating System. (Figure 1) (2009)
4. Bruyninckx, H.: Open robot control software: the OROCOS project. In: Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on. Volume 3., IEEE (2001) 2523–2528
5. Ando, N., Suehiro, T., Kitagaki, K., Kotoku, T.: RT-Middleware: Distributed Component Middleware for RT (Robot Technology). IEEE/RSJ Conference on Inlelligent Robots and Systems (2005) 3555–3560
6. Gruber, T.: What is an Ontology. (1993) 1–11
7. Hochgeschwender, N., Gherardi, L., Shakhirmardanov, A., Kraetzschmar, G.K., Brugali, D., Bruyninckx, H.: A model-based approach to software deployment in robotics. Intelligent Robots ... (2013)