Graphical Programming Language Support for Service Oriented Architecture in Automation

Theorin, Alfred; Johnsson, Charlotta

2012

Graphical Programming Language Support for Service Oriented Architecture in Automation

Alfred Theorin and Charlotta Johnsson

Department of Automatic Control, Lund, Sweden
{alfred.theorin,charlotta.johnsson}@control.lth.se

The trend in manufacturing industry today is that the complexity of the manufacturing equipment is increasing. At the same time the requirements on vertical integration and faster set-up and adaptations are increasing. To cope with this, highly re-usable, flexible, and adaptable automation systems are needed. One promising technology is Service-oriented architecture (SOA) which has already been recognized in several research projects, e.g. the EU projects SIRENA [2] and SOCRADES [1] [3]. The outcome of the SIRENA project was an early Devices Profile for Web Services (DPWS) implementation targeted at embedded devices. However, in practice SOA is still not widely used for automation. This should be of no surprise as it is a fairly new technology, and in the automation world only mature technologies that are known to work well in practice are normally considered. One step towards convincing the industry that using this approach is worth considering is to present realistic examples for when the approach has been applied successfully and with the anticipated advantages.

In [4] Grafchart, a graphical programming language for sequential control applications, was extended with a generic DPWS integration that is very easy to use. This enables anyone to try out SOA for automation with a minimal effort, and to experience the advantages of using this approach. It was also shown that the approach works very well on a realistic demonstrator process, and that the resulting coordination application turned out to be practically identical to the way that you would typically think about and model the coordination for the process, a usual advantage of graphical programming languages.

The top level entity of DPWS is the devices and below each device is the ordinary web service structure with *services*, *port types*, and *operations*. In DPWS it is also mandatory for devices to be discoverable and self-describing, i.e. to supply the WSDL file for the services. It is possible to probe the network for all running DPWS devices as well as detecting when a DPWS device on the network starts/stops. The WSDL file contains all the information required to be able to use the services, i.e. all avaiable operations and their signatures. Together this makes it possible to create a generic, Plug n' Play-like, DPWS client and this is what has been done for JGrafchart, the free Java implementation of Grafchart.

The link between the DPWS devices and the JGrafchart applications consists of the new IO element *DPWS Object* in JGrafchart. Each DPWS Object is configured to be bound to a *port type*. Since DPWS devices have a mandatory unique identifier the port type can be, and is, re-bound automatically, e.g. when a saved JGrafchart application is opened or when a DPWS device is restarted. As all other IO in JGrafchart the DPWS Object is given a name that is used to access it from the actions and conditions.

WSDL specifies four kinds of operations; one-way, request-response, solicit-response, and notification. One-way and request-response calls are initiated by JGrafchart and the difference is that request-response operations return something while one-way operations do not. Symmetrically solicit-response and notification calls are initiated by the device and solicit-response operations return something. All except solicit-response are supported by JGrafchart and calling of DPWS operations from JGrafchart is designed to look like other method calls in JGrafchart.

Example: Looking in the DPWS services dialog in JGrafchart you find a discovered DPWS device which contains a port type that you would like to use. It has the one-way operation `oneWayOp` which takes no parameters, the request-response operation `reqRespOp` which takes one parameter, and the notification operation `eventOp`. You add a DPWS Object to the application and bind it to this port type. Assuming that you give the DPWS Object the name `myDPWSObj`, using these operations from your application may look like:

```
S myDPWSObj.oneWayOp();
S ret = myDPWSObj.reqRespOp("par");
S dpwsSubscribe(myDPWSObj, "PT10M");
...
S e = dpwsHasEvent(myDPWSObj, "eventOp");
S ev = e ? dpwsGetEvent(myDPWSObj, "eventOp") : 0;
```

On the second line the returned value from the call is stored in `ret`. On the third line a subscription of events (notifications) for 10 minutes on `myDPWSObj` is initiated. On the last two lines it is checked if any `eventOp` calls have been received and if so stores the value from the oldest call in `ev`.

References

1. De Souza, L.M.S., Spiess, P., Guinard, D., Köhler, M., Karnouskos, S., Savio, D.: Socrades: A web service based shop floor integration infrastructure. Networks 4952, 50–67 (2008), http://www.springerlink.com/index/05581001K35585K4.pdf
2. Jammes, F., Mensch, A., Smit, H.: Service-oriented device communications using the *devices profile for web services*. In: Terzis, S., Donsez, D. (eds.) MPAC. ACM International Conference Proceeding Series, vol. 115, pp. 1–8. ACM (2005)
3. Kirkham, T., Savio, D., Smit, H., Harrison, R., Monfared, R., Phaithoonbuathong, P.: Soa middleware and automation: Services, applications and architectures. In: Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on. pp. 1419 –1424 (july 2008)
4. Theorin, A., Ollinger, L., Johnsson, C.: Service-oriented process control with grafchart and the devices profile for web services. In: INCOM 2012 (May 2012)