LUND UNIVERSITY

Service Orchestration with OPC UA in a Graphical Control Language

Theorin, Alfred; Hagsund, Johan; Johnsson, Charlotta

Link to publication

Service Orchestration with OPC UA in a Graphical Control Language

Alfred Theorin, Johan Hagsund, and Charlotta Johnsson
Department of Automatic Control, Lund University, Lund, Sweden

Automatic Control
www.control.lth.se
LUND UNIVERSITY

Background

Increasing market demands require highly flexible automation systems. A successful approach to compose flexible software is Service Oriented Architecture (SOA). In this work a convenient way to do SOA service orchestration with OPC Unified Architecture (OPC UA) is presented.

Me

Alfred Theorin
Ph.D. student
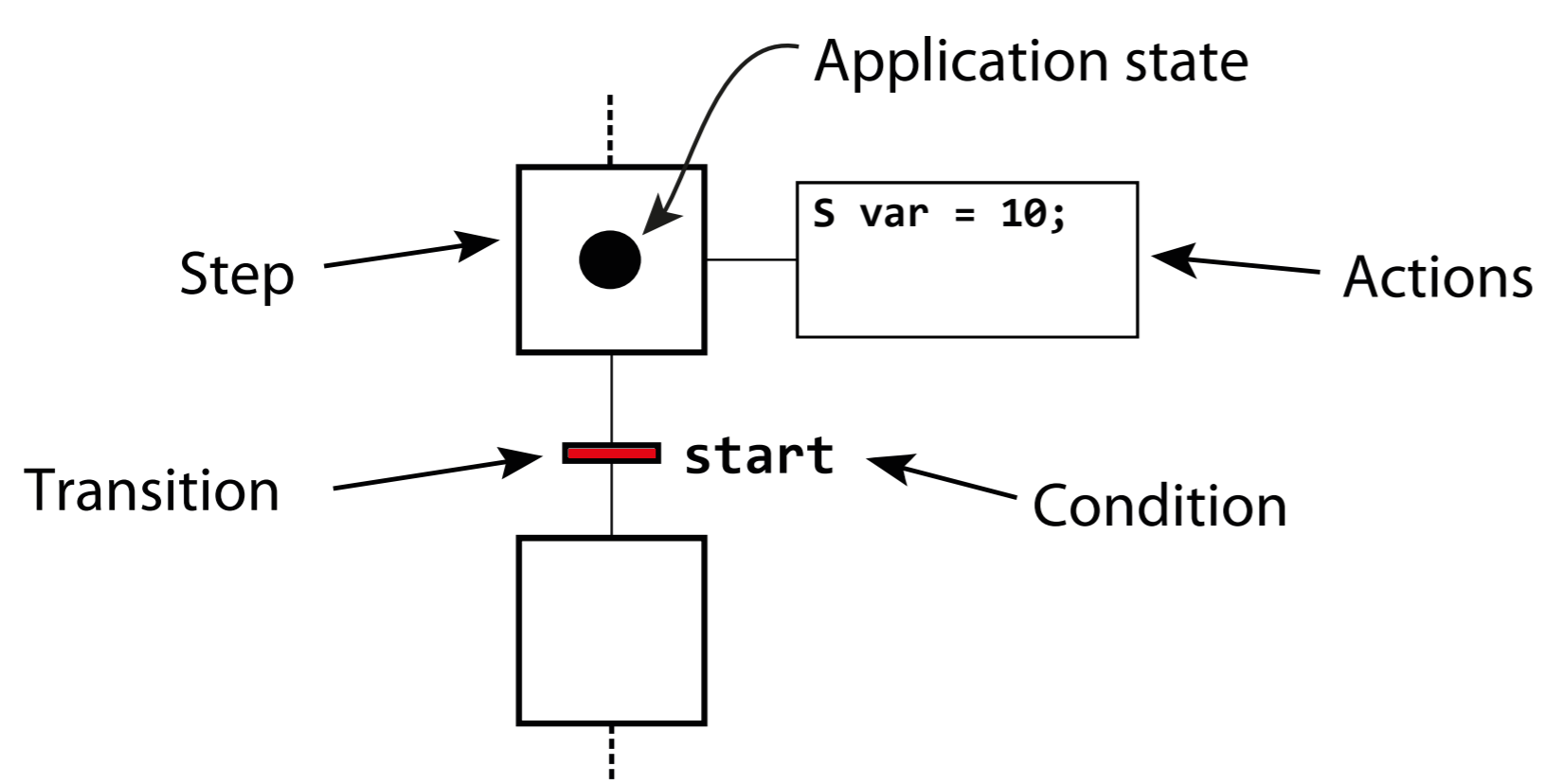alfred.theorin@control.lth.se

OPC Unified Architecture

OPC UA offers robust and secure communication for distributed systems. It is backward compatible feature-wise with the classic OPC standards and is platform and language independent. It also offers SOA features and more powerful modeling capabilities with types and instances.
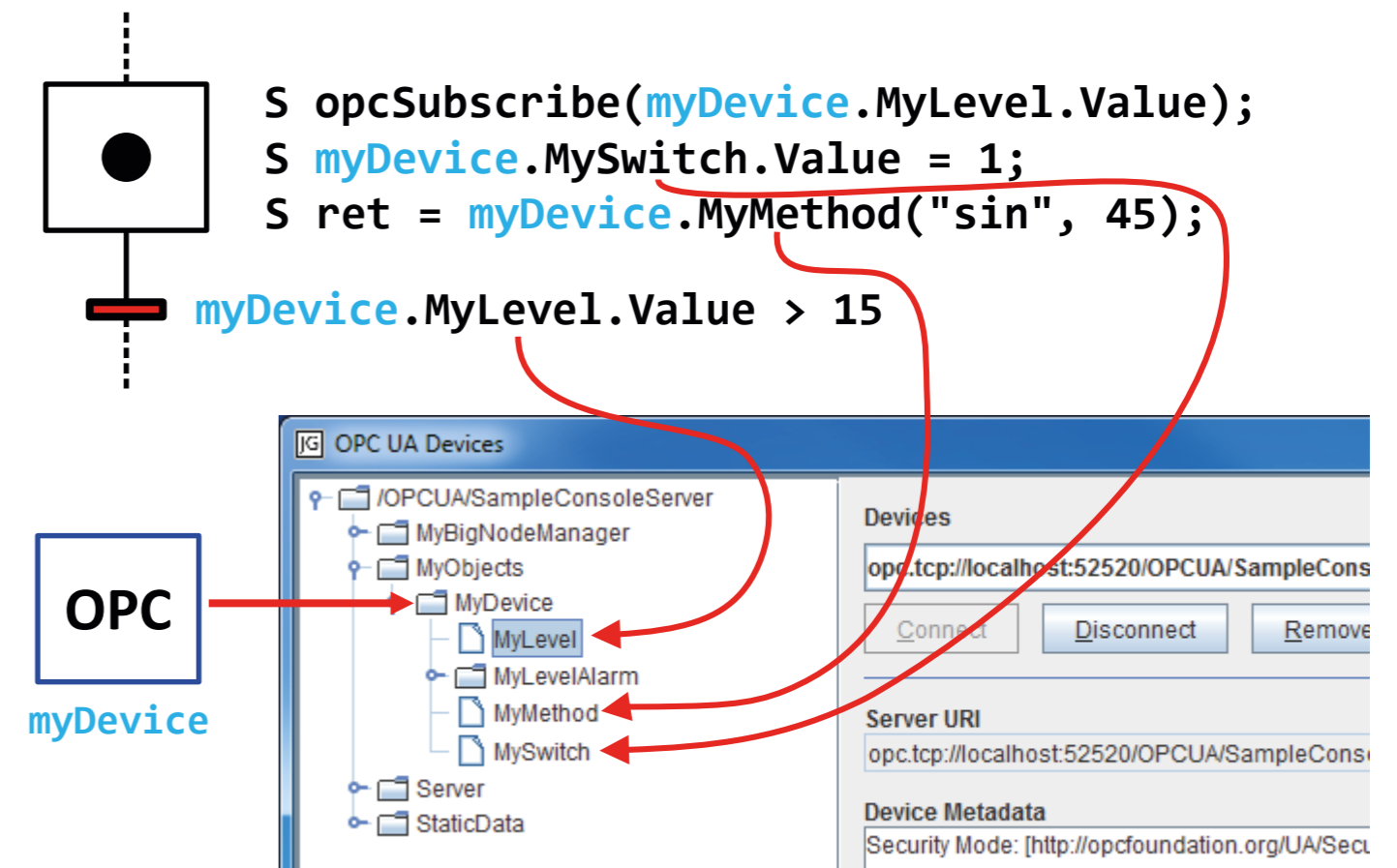
| Java	C++	C#	
OPC UA API			Application layer
UA Binary		**UA XML**	Data encoding layer
UA-SecureConversation		WS-SecureConversation	Security protocol layer
OPC UA TCP		SOAP	Transport protocol layer
		HTTP / HTTPS	
TCP/IP		TCP/IP	

The Grafchart Language

Grafchart is a graphical programming language based on the PLC standard language SFC. The basic building blocks are steps (application states) with actions (what to do when) and transitions (how and when to change application state).
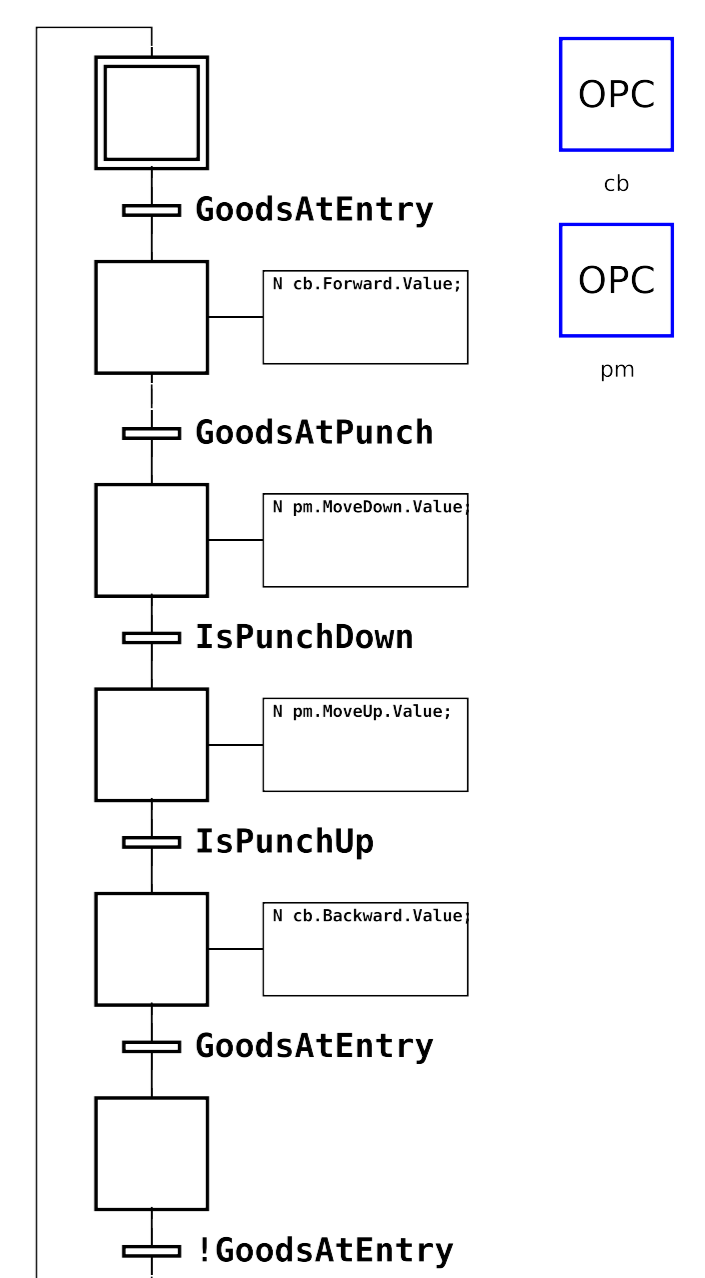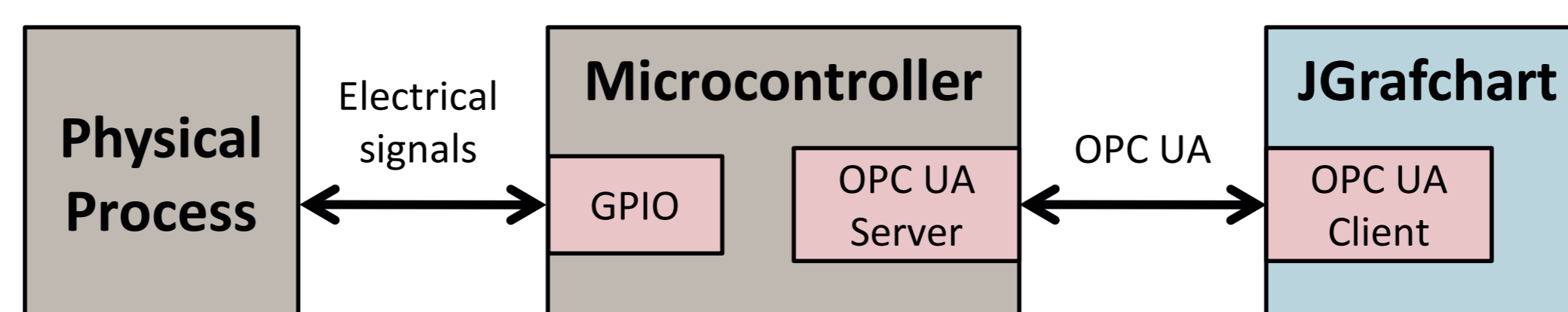
OPC UA Integration in JGrafchart

```
S opcSubscribe(myDevice.MyLevel.Value);
S myDevice.MySwitch.Value = 1;
S ret = myDevice.MyMethod("sin", 45);

myDevice.MyLevel.Value > 15
```


An *OPC UA Object* is bound to a node in an OPC UA server. The OPC UA variables and methods are then used like native variables and methods.

Use Case
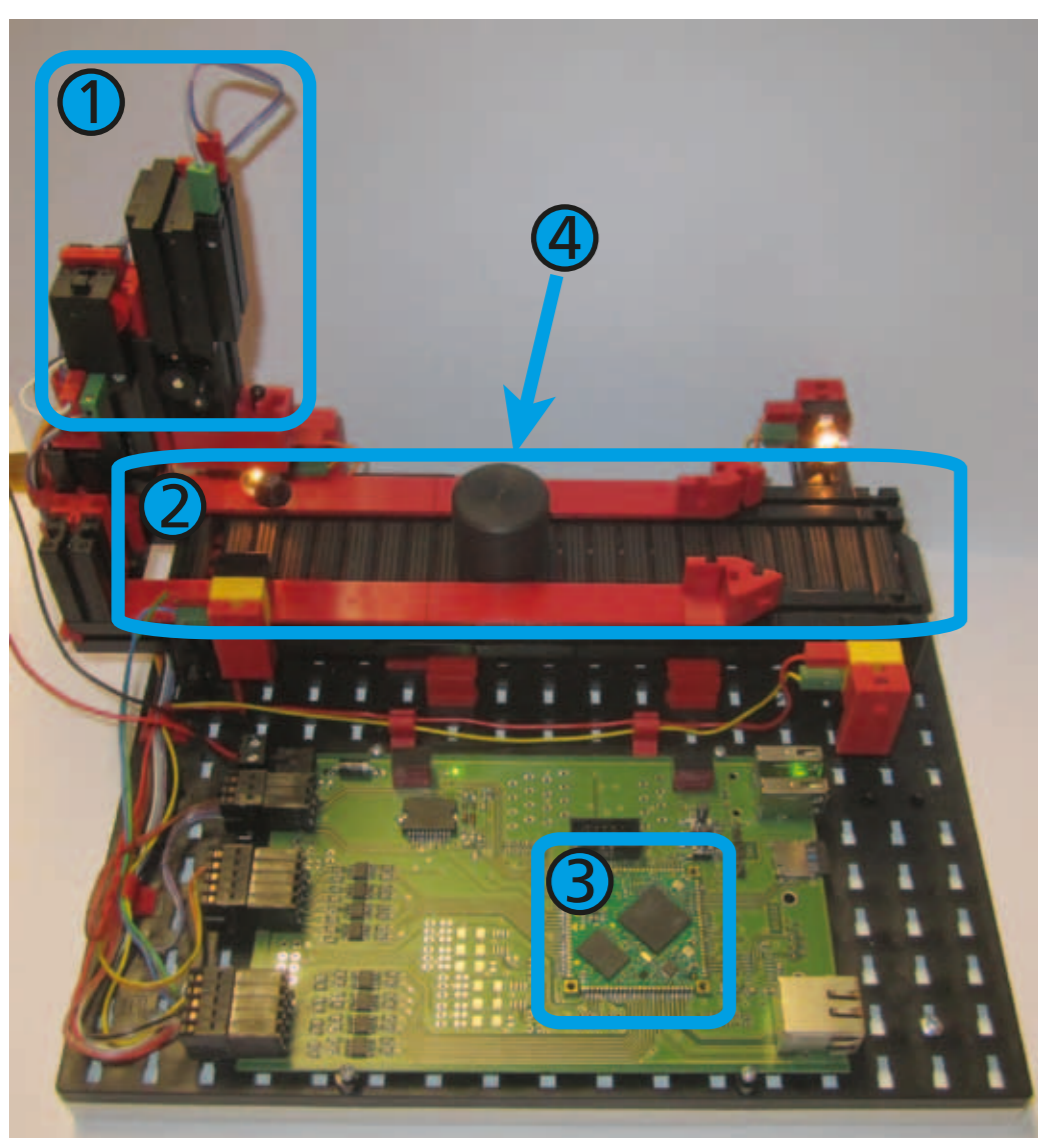
The OPC UA support was tried out on a physical device in an Industry 4.0 setting. The device consists of a punching machine (1) and a conveyor belt (2). A microcontroller (3) was used to expose the device as variables in an OPC UA server.

The task to process incoming goods (4) was straightforward to implement in JGrafchart with the new OPC UA support.

Physical Process — Electrical signals → **Microcontroller** [GPIO | OPC UA Server] — OPC UA → **JGrafchart** [OPC UA Client]

Conclusions

The new integrated OPC UA support in JGrafchart is a convenient way to do SOA service orchestration in the automation domain. Both DPWS and OPC UA are required to cope with industrial requirements on device level and now JGrafchart supports both. With an OPC classic adapter, JGrafchart can now also interact with most industrial automation systems.

Download: **http://www.control.lth.se/Research/tools/grafchart.html**

Acknowledgments

LISA. ELLIIT