Using S-BPM for PLC Code Generation and Extension of Subject-Oriented Methodology to All Layers of Modern Control Systems

Harald Müller[1,2]

[1] Johannes Kepler University Linz
Department of Business Information Systems - Communications Engineering
Freistaedterstrasse 315, 4040 Linz, Austria
[2] Metasonic AG
Münchner Straße 29, Hettenshausen, 85276 Pfaffenhofen, Germany
harald.mueller@{jku.at,metasonic.de}

Abstract. Interdisciplinary collaboration has become a challenge in industrial enterprises. Substantially for overcoming departmental borders are supporting software systems and a common understanding of technical and also business processes. A Production Planning and Control System (PPC) supports the specialized divisions and provides data management tools with the aim to reduce processing time and increase in productivity. Embedded to an Enterprise Resource Planning System (ERP) a high level of integration can be reached. Communication between the parties involved is changing because of a SOA and standardization among all levels of modern control systems. Therefore BPM 2.0 can be used for business processes as well as control layer processes, if necessary definitions to fulfill communication and execution are added. IEC 61131, the only standard in automation control, and IEC 62541, a draft standard for vertical data integration, supports these technological change. This paper provides an approach for subject-oriented process modeling and inter-layer communication, starting at the control layer up to business process using subject-oriented methodology.

Keywords: IEC 61131, IEC 62541, SCADA, BPM 2.0, automation, process control, control process.

1 Introduction

Subject-oriented automation process modeling is an enhancement to S-BPM with the aim to influence the subject-oriented modeling methodology on different layers of industrial companies like Manufacturing Execution System (MES), Supervisory Control and Data Acquisition (SCADA) and the Control Layer. This is possible because of technological progress in automation industries which has rapidly increased since the early 1970s. If there were expensive hard-wired logic controllers at the beginning, today program implementation is based on high level computer-language. IEC 61131 defines the basic conditions for the development of programming languages and PLC applications. Part 3 of IEC 61131 deals

C. Stary (Ed.): S-BPM ONE 2012, LNBIP 104, pp. 182–204, 2012.

with programming languages and defines two textual (IL: Instruction List, ST: Structured Text) and two graphical (FBD: Function Block Diagram, LD: Ladder Diagram) PLC programming language standards and the Sequential Function Chart (SFC) which provides elements to organize programs for sequential and parallel control processing. The main advantage is the possibility to use textual and graphical programming languages within one PLC application. In context of automation process modeling, subject-oriented Model Driven Development (MDD), can be used on the Control Layer for code-generation of IEC 61131-3 applications using the PLC programming languages and application flow control. If we focus on the definition of Model Driven Development (MDD) "*... a set of approaches in which code is automatically or semi automatically generated from more abstract models, and which employs standard specification languages for describing those models and the transformations between them.*" [17] and contrast MDD with the graphical languages of IEC 61131-3 (e.g. FBD), there is a congruence. Indeed the abstraction level - the difference between specification and code - is very slightly [11, p.2]. Due to the fact that total costs of programming is often higher than the hardware itself, and a change of process control applications has a huge impact to change management leads to the logical consequence to raise the abstraction level of automation process applications which means that we need to allow developers to work in more abstract models. Model-based Software Engineering (MBSE), besides the discussion about Platform-Independent Models (PIMs) and Platform-Specific Models (PSMs), can be reduced to two fundamental notions [18, p. 384]:

1. "*Raising of the level of abstraction; that is, raising the level of software specifications even further away from underlying implementation technologies (relative to, say, traditional programming languages)*" and
2. "*Raising the degree of computer-based automation used to bridge the widening gap between design specifications and corresponding implementations.*"

The term "model" in context of MBSE "*is often used as a generic term to denote any specification expressed using a higher-level formalism, whether it is an abstraction that omits detail or a fully-fledged implementation specification from which a complete executable program can be auto-generated*" [18, p.384].

The IEC 61499, which is an object-oriented further development of IEC 61131, is partial included to the IEC 61131-3:2009 draft standard. Besides the discussion of including object-oriented aspects, this paper focuses on the subject-oriented approach pursuing the following purposes:

1. Transformation of subject-oriented model to IEC 61131-3 code on a model driven development approach should raise the level of abstraction. The method of modeling should focus on the used standard. The transformation uses the PLCopen XML Formats for IEC 61131-3 [16] definition to enable an exchange between different development environments and to fulfill the MBSE definition of "model".

2. A dynamic allocation of processes by the developer or system operator to PLC or Metasonic Flow, supported by vertical data integration functionality, should make a system more flexible and allow industrial enterprises exchange data by using a jCPEX! [13] plattform.
3. Direct integration of automation data into a Subject-oriented Business Process Model using a Behavioral Interface (BI).

PLCopen is a vendor- and product independent worldwide association trying to increase efficiency during the application development, while increasing the software quality and lowering life-cycle costs. One of the core activities of PLCopen is focused around IEC 61131-3, the only global standard for industrial control programming. The PLCopen XML specification allows to exchange programs, libraries and projects between development environments using the XML standard [16]. It harmonizes the way people design and operate industrial controls by standardizing the programming interface [15] and supports subject-orientation within the control-layer.

Supervisory Control and Data Aquisition (SCADA) as part of Industrial Control Systems (ICS) is subject-oriented enhanced by providing the possibility of vertical data integration using the OPC Unified Architecture (OPC UA) which provides a cohesive, secure and reliable cross platform framework for access to real time and historical data and events [14]. Therefor process data interchange between a programmable logic controller and a process flow control needs to be realized.

The goal of BPM 2.0 is to rapidly react to changing business environment in a complex business world. To reach this goal a BPM 2.0 approach must fulfill properties (see [10, p.86]):

- "*Only the participants in a process truly understand the complexity of the processes they are involved in.*" [10, p.86]
  If all layers of industrial processes (Sect. 2) are enhanced by subject-oriented methodology, divisional specialists are able to communicate using "the same language" and are able to define a common interface (in context of S-BPM: *Behavioral Interface*) in different processes, on different levels. This possibility supports that "*The parties involved in producing products or services have to agree on interaction behaviors for synchronizing their activities.*" [10, p.85]
- "*... the models should be executable without any additional programming or programming know-how ...*" [10, p.85].
  If processes on different levels are merged to one inter-divisional, multi-layer process using the same modeling methodology without having the same execution environments, standardized communication interfaces between these systems even though have to be a prerequisite for BPM 2.0. The ability to execute a control layer process, e.g. programmable logic controller applications, assumes the transformation of a subject-oriented process model to IEC 61131-3 application code. Therefore not the execution environment needs to be created but a Model Driven Software Engineering approach has to be assisted.

– "*... the process environment - the socio-technical system consisting of people, machines and software - should be easily integrated with the BPM model.*" [10, p.85].
  The integration of *machines* whether direct, using vertical data integration, or using a software interface on a higher layer of a modern industrial control systems (e.g ERP System) (Sect. 2) enables a company to illustrate the company wide communication model including people, machines and software.

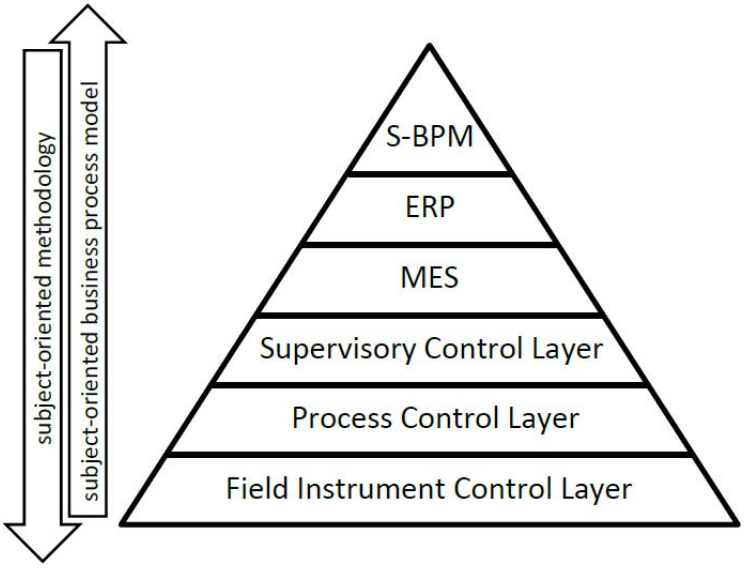
This document describes the possibilities to enhance the system layer architecture of modern control systems using subject-oriented methodology with a focus on the *Control Layer*. An introduction to modern control system architecture and S-BPM in context of automation is described in Sect. 2. Sect. 3 defines the *Subject-oriented Process Model* and introduces to the showcase used to explain the different layers. The use of subject-oriented methodology for MDSE within the *Control Layer* is shown in Sect. 4 and its subsections. The direct integration of control data to business processes and human-machine-interfaces is explained in Sect. 5, followed by the last section "Summary and Issues for Further Research".

2 A New Approach in Automation Engineering

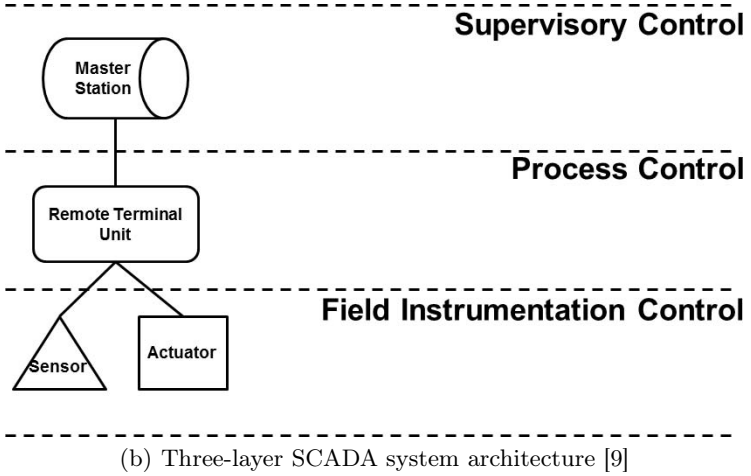
Subject oriented Process modeling in context of automation processes is not just to eliminate the word 'business' in S-BPM. There is a need to describe processes on different levels (Fig. 2). The structure of PLC-systems has changed. At the beginning a centralized PLC transfered the data to an associated server. This changed in the last years toward a more distributed architecture supported by standardization of communication infrastructure, a SOA and IEC 61131. Therefore most classifications divide modern control systems into four layers for control, visualization and production support systems: [1]

– *Field Layer with instrumentation.*
– *Control Layer with automation devices.*
– *Real Time HMI (Human Machine Interface) Layer with visualization devices.*
– *Real Time MES (Manufacturing Execution System) Layer with data processing devices.*
– *A Production Planning System (PPS) or Enterprise Resource Planning System (ERP) can be defined as a separate layer above these.*

Supervisory Control and Data Acquisition (SCADA) is the generic term for the hardware, software and procedures, used to control and monitor industrial processes. "*The latest trend of SCADA system is the three-layer SCADA architecture which depending on open system technology rather than a vendor controlled proprietary environment.*" [9, p.774] The PLCopen XML specification, used in this paper, as well as the OPC Unified Architecture is part of these open system architecture, supporting the communication between various vendors on an independent basis. Figure 2 (b) illustrates the three-layer SCADA system architecture:



(a) Subject-orientation in a modern control system



(b) Three-layer SCADA system architecture [9]

Fig. 1. Control system layer architecture

1. *Supervisory control layer (Master Stations)* are non-dedicated PCs processing automation duties (e.g. alarm handling, logging, trending) and have two main functions [9]:
   - *Periodically obtain data from RTUs (Remote Terminal Units)/PLCs*
   - *Control remote devices through the operator station*
2. *Process control layer (RTUs/PLCs)* usually consists of more than one device depending on the situation. The devices used are: [9]:
   - *Programmable Logic Controller (PLC)*
   - *Analog Input and Output Modules*
   - *Digital Input and Output Modules*
3. *Field instrument control layer (Sensors and Actuators)* "*This layer mainly consists of sensors and actuators. The sensors perform measurement and actuators perform control. Sensors get the data (supervision and data acquisition) and actuators perform actions dependent on this data (control). The processing and determination of what action to take, is done by the master control system (i.e. SCADA).*" [9]

Subject-oriented (business) process modeling can match or extend the different levels of the industrial IT system hierarchy. By illustrating the *Field instrument control layer* (*Field Layer*) using subject-oriented methodology, two advantages accrue:

- A standardized IEC 61499-1 [4] *Field Layer* documentation can be exported by code-transformation.
- The *Subject-oriented Hardware Model* (or IEC 61499-1 documentation) is the basic configuration for *Subject-oriented Code Generation* and therefor directly used.

A company's process control layer (*Control Layer*), in context of this document a programmable logic controller (RTUs/PLCs), needs to be described in a way that an inter-divisional process understanding is possible. Not the detailed control process provided by the vendor of a sensor, an actuator or a machine, needs to be illustrated but the process control information, the interaction between machines, sensors and actuators needs to be illustrated and embedded to a company wide process model. The subject-oriented methodology can help to bridge the widening gap between design specifications and corresponding implementations in context of automation applications. The illustration of program organization units (POUs) [7, p.51, 6.5] and configuration elements [7, p.126, 6.7] can be done with a subject interaction diagram wheras the logic of an application implemented in the body of *programs*, *function blocks* and *functions* could be modeled by using the subject behavior diagram.

A business process which mainly communicates with a PLC and works in context of automation, is a control process. If it uses the same methodology and workflow engine as a business process but is based on a different layer (*Control Layer*) the Subject-oriented Application Flow Model enhances this layer.

SCADA (*Supervisory Control*) is supported by *jCPEX! Automation Extension (AE) platform* (Sect. 3.1) which provides *Behavioral Interfaces* for communication purposes (e.g. to use in business processes, human-machine interfaces or for inter layer communication).

The *MES* (*Manufacturing Execution System*) in context of this paper is a technological use of the jCPEX! platform [13, p.176, 2]. If jCPEX! is not only used for cross-organizational business processes but enhanced with the aim to provide inter-layer communication, using the *Behavioral Interface*, a standard inter-layer communication interface is established. If this jCPEX! platform provides the possibility to distribute process data based on rules, a process control and control process functionality is added on all industrial system layers. Using jCPEX! platform, a MES is implemented when data exchange between SCADA and an ERP System or business process is realized.

The use of subject-oriented methodology on different layers of an industrial company allows a subject-oriented business process to communicate to all layers directly by using a standard interface, namely *Behavioral Interface* (BI). A SOA and the use of open system technology in combination with subject-oriented modeling methodology in context of BPM 2.0 enables direct communication to the socio-technical system - people, machines and software [10, p.85].

3 Subject-Oriented Methodology Enhances Automation on 3 Layers

Figure 2 illustrates the different layers of the subject-oriented process modeling including the automation and business process model. The *Subject-oriented Code Generation Model* (S-CGM) represents the IEC 61131-3 configuration elements [7, p.126, 6.7] and the programming model [7, p.21, 4.1]. Depending on the execution target (PLC or Metasonic Flow) the code-transformation and execution environment differs.

MDSE allows to generate IEC 61131-3 code which can directly be executed on a programmable logic controller using PLCopen XML specification as transformation target. Figure 7 shows the S-CGM and its sub models (S-HWM, S-TM, S-POUM, S-CM) which are described in Sect. 4. The *Subject-oriented Application Flow Model* (S-AFM) as part of *Subject-oriented Program Organization Unit Model* (S-POUM) communicates with the layer above by using a Behavioral Interface to enable communication between Control Layer (e.g. programmable logic controller (PLC)) and S-AFM or S-BPM using jCPEX! Automation Extension (AE) platform as described in Sect. 3.1.

The *Subject-oriented Application Flow Model* (S-AFM) as a separate layer can be used as a communication interface between hardware and S-BPM or by adding process control functionality as a (process) Control Layer similar to a software PLC.

PLC I/O variables (*DataItems* or a group of *DataItems* [5, p.8, 3.4.1]), illustrated as messages, are logically allocated to an Internal Subject and provided to the S-BPM by jCPEX! AE platform. The representation form used for inter-layer communication is a Behavioral Interface (BI) (an example is shown in Fig. 13).

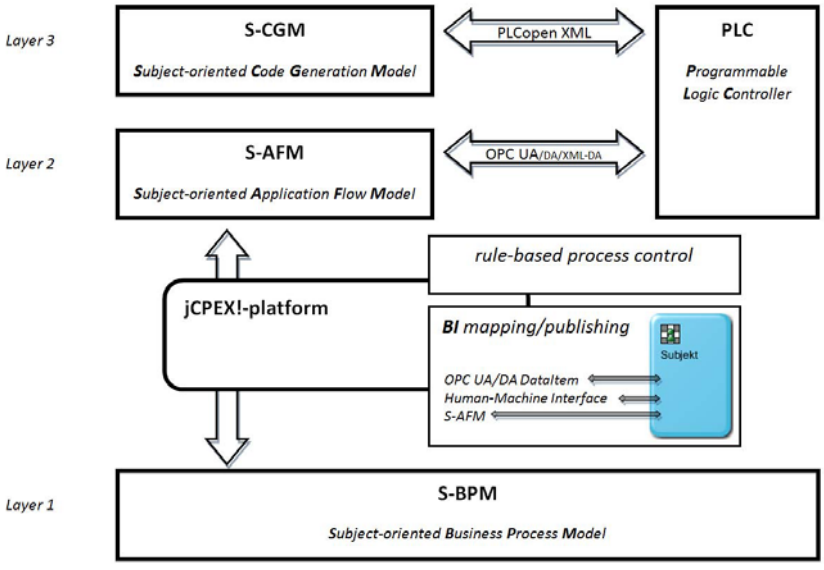


Fig. 2. Subject-orientation in context of automation

3.1 Inter-layer Communication with jCPEX! Automation Extension (AE)

In context of the jCPEX! approach, the communication between the involved partners is described as an implementation-independent choreography - the so called *Behavioural Interface* (BI) [13, p.176]. Thereby the BI can be seen as "*interface to the private process of the participation partners*"[13, p.176]. In context of automation processes we are less talking about inter-organizational communication than inter-layer communication, but the description of the observable behavior can be illustrated equally. Depending on the process, a corresponding possibility to administrate and distribute the BI has to be implemented (e.g. USDL Repository [13, p.184]).

As shown in Fig. 3, a database takes over the task of an *Behavioral Interface Repository*. A mechanism to allow communication between *Control Layer* and *Business Process* needs to be implemented. If we keep in mind, that a BI can be automatically derived from the internal private process [13, p.184], and the external subject is a known programmable logic controller *DataItem*, a mapping between an external Subject and a *DataItem*, supported by a Mapping Editor, is the basis to adopt the jCPEX! platform toward a inter-layer communication platform.

Figure 3 illustrates the communication between jCPEX! and a programmable logic controller. The modeling process and the export of a BI to the *Behavioral Interface Repository*, as well as the mapping of a BI to a PLC *DataItem* is enabled by the *Subject/DataItem Generator*, which provides a bidirectional data

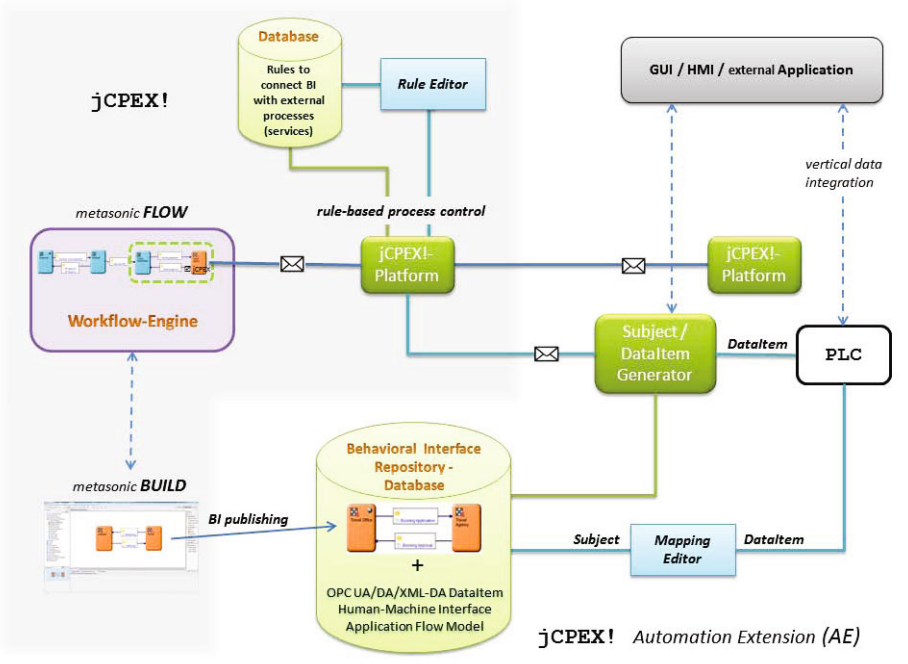


Fig. 3. jCPEX! Automation Extension (AE)

transfer between workflow engine and PLC by transforming a message to a corresponding *DataItem*. This communication can be realized by using the *OPC Unified Architecture (UA)* [6], *OPC DataAccess* or *OPC XML DA*.

jCPEX! platforms communicate via web services and use XML as data exchange format. A common interface allows to address different applications such as GUIs, human machine interfaces and other external applications. Figure 4 illustrates the communication between a jCPEX platform and an external application, namely *Subject/DataItem Generator* which provides a bidirectional access to a programmable logic controller and provides an interface itsself. The external application can be addressed using rules which "*facilitates replacement of a partner dynamically dependent on certain conditions - even at runtime*". [13, p.176] Therefore a *RuleEditor* can distribute incoming events to one or, if *MultiPartyBIs* are supported, more *Subjects*.

3.2 Showcase

In order to describe the advantage of subject-oriented process modeling, in context of automation process modeling, and the integration into S-BPM a showcase is prepared to explain the different levels of integration. Figure 5 illustrates three different domains: *household*, *factory* and *supplier of energy*.

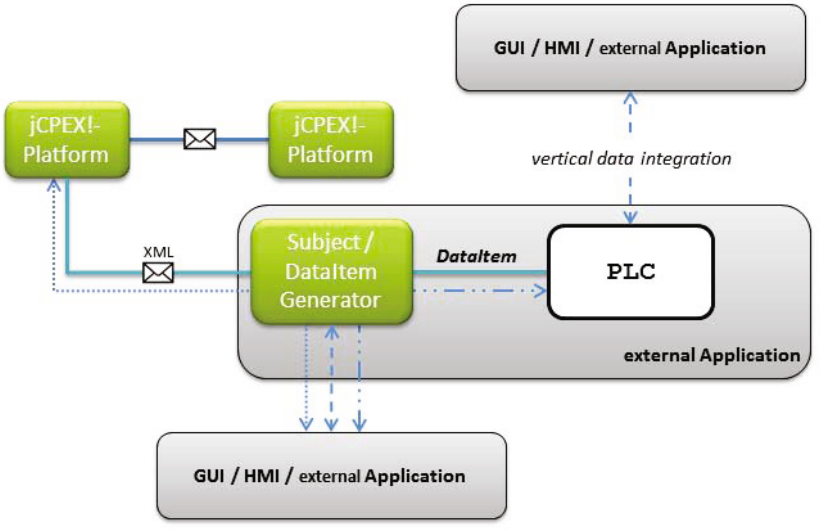


Fig. 4. jCPEX! DataItem Generator

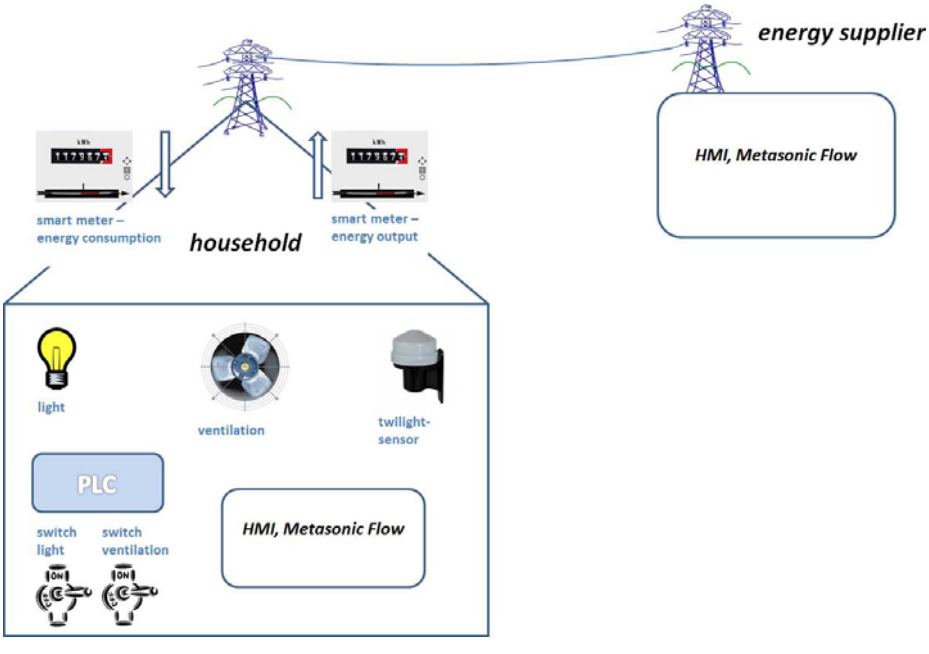


Fig. 5. Showcase - house control

The bidirectional communication between *supplier* and *household* as well as *supplier* and *factory* is a cross-organizational business process using the jCPEX! platform [13, p.176, 2]. The exchanged data is measured power consumption or measured power output. The process is described in S-BPM methodology using a Behavioral Interface which "*... contains the interaction behavior between involved actors and therefore represents the choreography for all concerned parties.*" [13, p. 184, 4.2]
Figure 6 illustrates the exchanged data between household and energy supplier. The S-BPM of household is kept simple with only one involved subject representing an tenant.

Within this document the subject-oriented implications are illustrated by using the example of house control. There are three sensors (*light switch*, *ventilation switch*, *twilight sensor*), two actuators (*light*, *ventilation*) and two energy meters (*energy consumption*, *energy output*). In addition to hardware I/O, the system interaction can be done using a Human-Machine Interface which is part of a SCADA system. *Energy consumption* and *energy output* is exchanged by a subject-oriented business process running in Metasonic Flow.

This showcase is used to describe the processes on the three layers illustrated in Fig. 2.

- *Layer 1*, the business process, is a cross-organizational business process with two involved subjects. The internal behavior of household includes the subject *AF automation* (Fig. 13 (b)) which provides the metered data and enables the possibility to switch the *light* and *ventilator*, ON and OFF, out of the business process.
- *Layer 2* enables the communication between *programmable logic controller* and the subject *AF automation*. Layer 2 represents the communication interface between control layer (PLC) and S-BPM using an jCPEX! AE platform.
- *Layer 3* generates IEC 61131-3 code which is described in Sect. 4 and uses Layer 2 for communication with the IT system.

The following Sect. 4 defines the prerequisites to transform a subject-oriented model to a IEC 61131-3 application using MDSE, and illustrates the transformation process using the example of house control as described in Sect. "Showcase". Layer 1 and 2 is described within one section (5) because the *Subject-oriented Application Flow Model*, as part of layer 2 and 3 (Fig. 2 and 7 (b)), is used as communication interface to integrate process data to S-BPM.

4 Subject-Oriented Code Generation Model (Layer 3: S-CGM)

Figure 7 (b) shows the S-CGM (***S***ubject-oriented ***C***ode ***G***eneration ***M***odel) which is matched to the procedure of implementing an IEC 61131-3 application.

The IEC 61131-3 Software Model shows the basic high-level language elements and their interrelationship which consists of programmed elements and configuration elements [7, p.21 4.2]. The Subject-oriented Code Generation Model is

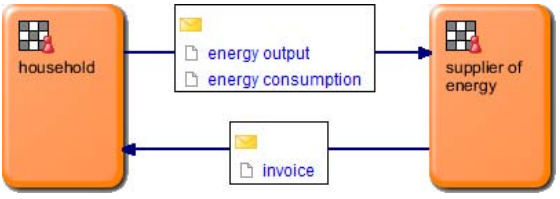


Fig. 6. BI: household - supplier of energy

a representation of a the Software Model described in Fig. 7 (a). Therefore it is important to distinguish between *programming* and *configuration elements* (in context of S-CGM: Subjects). In order to be able to communicate with a PLC, basic configurations need to be carried out. Therefore the Subject-oriented Configuration Model (S-CM) and Subject-oriented Task Model (S-TM) as well as the Subject-oriented Hardware Model (S-HWM) is established. A Behavioral Interface (BI), whether it is used for S-CGM or as a communication interface on Layer 2 (S-AFM), is the basic representation form for communication on all layers. The use of jCPEX! AE platform (Sect. 3.1) allows direct interaction between all layers of modern control systems.

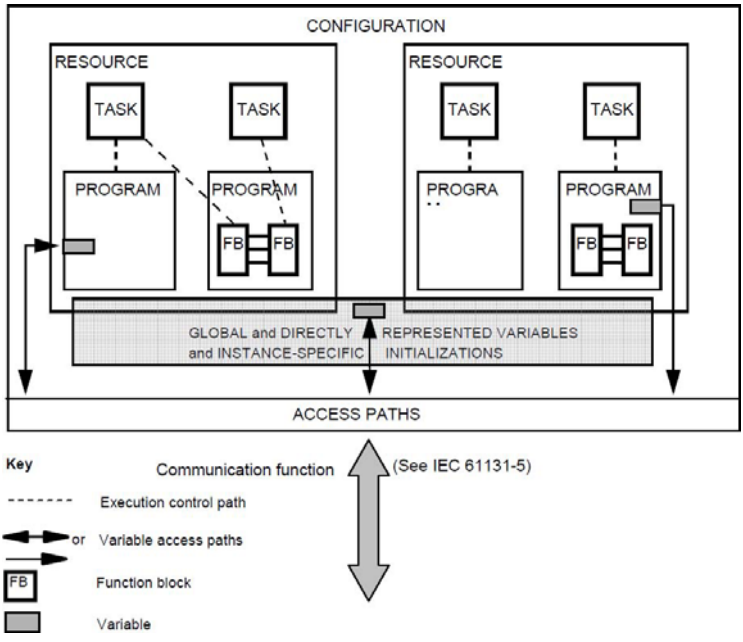
The Subject-oriented Program Organization Unit Model (S-POUM) represents the *programming elements* described in IEC 61131-3 and uses the S-CM and S-HWM.

4.1 *S*ubject-Oriented *H*ardware *M*odel (S-HWM)

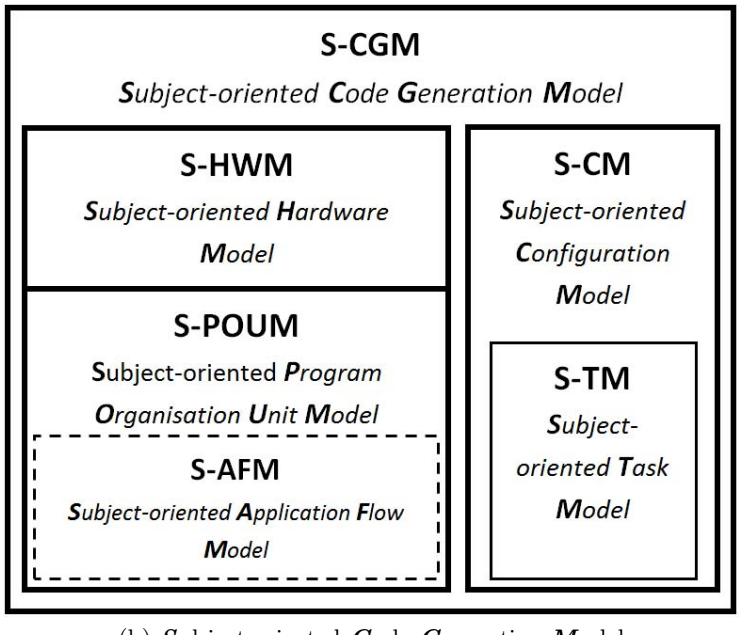
The ***S***ubject-oriented ***H***ardware ***M***odel associates a physical hardware device (sensors, actors) to a PLC I/O represented by a variable. In the domain of S-CGM the corresponding S-HWM represents a *configuration* Subject. It is the part of configuration elements, namely, *configurations*, *resources*, *tasks*, *global-variables*, *access paths*, and *instance-specific initializations* [7, p.21, 4.1], which represent hardware devices logically or physically .

The logical and physical connection of sensors and actuators and the assigned interfaces are available in numerous different documents. Even if the installation is hard-wired, the logical connection has to be documented in a further step which is the basis for advanced process modeling activities. IEC 61499-1 [4] defines a graphical and textual possibility to describe a system configuration which includes assignment of physical to logical port by a XML schema supporting system interoperability. IEC 61131-3 is compliant to IEC 61499 [4, p.85].

The S-HWM consists of sensor interfaces and/or actuator interfaces represented, by a Behavioral Interface, which is provided by jCPEX! AE platform. The exchanged messages are unidirectional or bidirectional depending on the hardware device. A multiple use of a BI, representing only one *DataItem* instance, within the subject interaction diagram has to be allowed because of modeling clarity issues.



(a) IEC 61131-3 - *Software Model* [7]



(b) ***S***ubject-oriented ***C***ode ***G***eneration ***M***odel

Fig. 7. IEC 61131-3 and corresponding subject-oriented model

No.	Feature/examples	
1[a]	**Declaration of locations of symbolic variables – AT keyword**	

	`VAR_GLOBAL`
	` LIM_SW_5 AT %IX27 : BOOL;`
	`END_VAR`
		Assigns input bit 27 to the Boolean variable `LIM_SW_5` (NOTE 2)

	`VAR`
	` CONV_START AT %QX25 : BOOL;`
	`END_VAR`
		Assigns output bit 25 to the Boolean variable `CONV_START`

	` TEMPERATURE AT %IW28: INT;`	Assigns input word 28 to the integer variable `TEMPERATURE` (NOTE 2)

	`VAR`
	`  C2 AT %Q* : BYTE;`
	`END_VAR`
		Assigns not yet located output byte to bitstring variable `C2` of length 8 bits

7[b]	**Declaration of enumerated variables**	

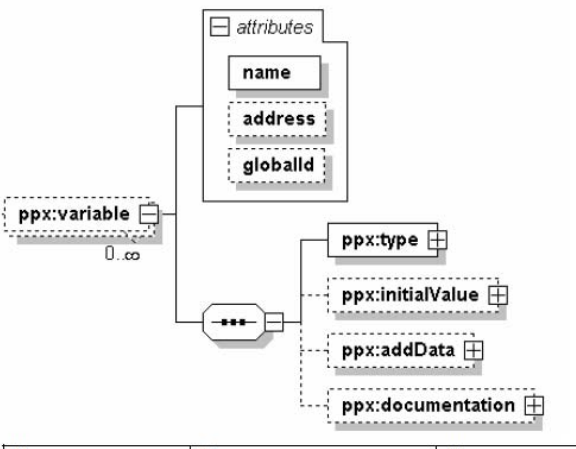
	`VAR`
	`  Y : (Red, Yellow, Green);`
	`END_VAR`
		Declaration of an enumerated variable

8[c,d]	**Declaration of subrange variables**	

	`VAR`
	`  Z : SINT(5..95);`
	`END_VAR`
		Declaration of a subrange variable

(a) IEC 61131-3 - variable assignment [7]

diagram:



attributes

Name	Type	Use	Default
name	xsd:string	required	
adress	xsd:string	optional	
globalID	xsd:ID	optional	

(b) PLCopen XML specification: *variable* [16]

Fig. 8. IEC 61131-3 and PLCopen *variable*

Figure 9 (a) illustrates the S-HWM. The *PRG light* represents a program organization unit, namely *program* which directly interacts with the I/O variables (digital input) *di light switch*, (analog input) *ai twilight sensor* and (digital output) *do light*. Because of the sensors and actuators used, the communication is uni-directional but, depending on the hardware device, a bi-directional communication is applicable. Depending on the selected hardware device the messages (business objects), sent and received, includes different values e.g. the raw value, the measurement value or value range. The data type of *single-element variables* [7, p.33, 6.3.2] can be mapped or defined whereas *multi-element variables*, namely *array* and *structure* have to be provided by the S-CM. Special to PLCs is the initialization of variables, according to the rules of IEC 61131-3, which can take one of the following initial values [7, p.45, 6.4.3]:

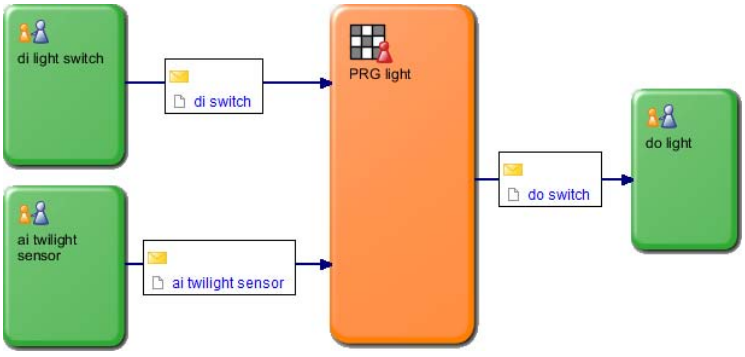
- *the value the variable had when the configuration element was "stopped" (a retained value)*
- *a user-specified initial value*
- *the default initial value for the variable's associated data type*

The declaration of the variables [7, p.45, Table 18] (e.g. *var*, *var_ input*, *var_ output*, *var_ global*, *var_ access* ...) has to be done automatically by model analyze, during the transformation process.

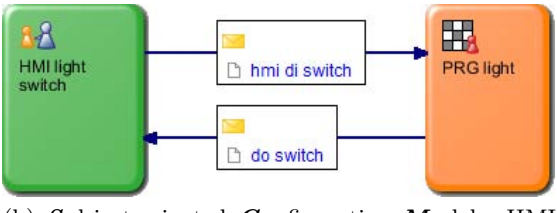
4.2 *S*ubject-Oriented *C*onfiguration *M*odel (S-CM)

The ***S***ubject-oriented ***C***onfiguration ***M***odel represents the *configuration elements* defined in part 6.7 of IEC 61131-3 [7]: "*A configuration consists of resources, tasks (which are defined within resources), global variables, access paths and specific initializations.*" In context of subject-oriented process modeling the configuration elements include interfaces to external applications e.g. Human-Machine Interface (HMI), webapplication or other programmable logic controller represented by a BI. Figure 9 (b) illustrates a Subject called *HMI light switch* which is an external HMI. Figure 9 (a) shows the hardware input represented by the single-element variable *di switch* and the hardware output represented by the variable *do light*. These variables are connected to the physical I/O register and represent the physical state of sensors and actuators whereas the Subject *HMI light switch* receives a variable *hmi di switch* which is changed by HMI interaction illustrated in Fig. 9 (b). Therefore *do light* changes the state of *hmi di switch* whenever the light status changes.

The ***S-TM*** (***S***ubject-oriented ***T***ask ***M***odel) is part of S-CM and defines the process execution. For the purposes of IEC 61131-3 [7, p.131, 6.7.3] a task is defined as: "*... an execution control element which is capable of calling, either on a periodic basis or upon the occurrence of the rising edge of a specified Boolean variable, the execution of a set of program organization units, which can include programs and function blocks whose instances are specified in the declaration of programs.*" [7, p. 131, 6.7.3] According to the definition of programming-language standards in IEC 61131-3 textual and graphical notation is available for definition of tasks.



(a) ***S***ubject-oriented ***H***ardware ***M***odel



(b) ***S***ubject-oriented ***C***onfiguration ***M***odel - HMI interface

Fig. 9. Subject-oriented communication with external hard- and software

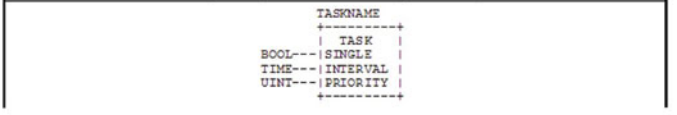
4.3 *S*ubject-Oriented *P*rogram *O*rganisation *U*nit *M*odel (S-POUM)

The ***S***ubject-oriented ***P***rogram ***O***rganisation ***U***nit ***M***odel represents the *function*, *function block* and *program* defined in the IEC 61131-3 [7, p.51, 6.5]. POUs are *programmed elements* and can be delivered by the manufacturer, or programmed by the user by the means defined in the IEC 61131-3.

A *program* is defined in IEC 61131-1 as a "*logical assembly of all the programming language elements and constructs necessary for the intended signal processing required for the control of a machine or process by a programmable controller system.*" [3]. *Program* as well as *function*, in context of S-POUM is represented by a *Internal Subject*. A *function* is defined as a program organizational unit (POU) which, when executed, yields no (VOID) or exactly one data element, which is considered to be the function result. [2]

A *function block* is represented by a *Multi Subject* which, "*when executed, yields no or exactly one data element, which is considered to be the function block result (like a function), and one or more values which are considered to be the function block outputs.*" [7, p. 74, 6.5.3.1] Multiple named instances (copies) of a function block type can be created on condition that each instance has an associated identifier (the instance name). Different to *functions*, the variables of a *function block* shall persist from one execution of the *function block* instance

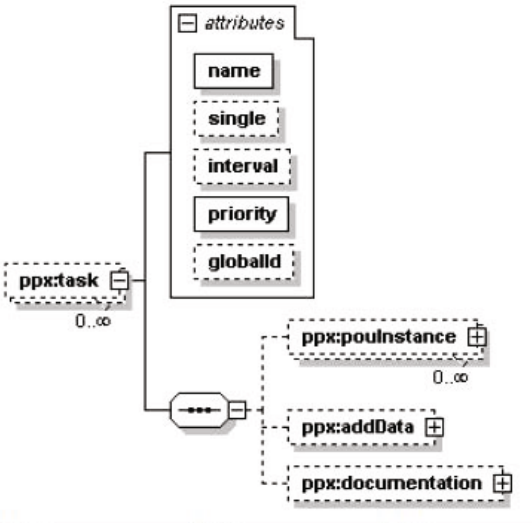
No.	Description/Examples
1a	Textual declaration of periodic TASK (feature 5a of Table 57)
1b	Textual declaration of non-periodic TASK (feature 5b of Table 57)
Graphical representation of TASKs (general form)	




```
1    CONFIGURATION CELL_1
2      VAR_GLOBAL  w: UINT;  END_VAR
3      RESOURCE STATION_1 ON PROCESSOR_TYPE_1
4        VAR_GLOBAL  z1: BYTE;  END_VAR
5a       TASK SLOW_1(INTERVAL := t#20ms, PRIORITY := 2) ;
5a       TASK FAST_1(INTERVAL := t#10ms, PRIORITY := 1) ;
6a       PROGRAM P1 WITH SLOW_1 :
8a                  F(x1 := %IX1.1) ;
9b       PROGRAM P2 : G(OUT1 => w,
6b                      FB1 WITH SLOW_1,
6b                      FB2 WITH FAST_1)  ;
```

(a) IEC 61131-3 - *task* graphical and textual [7]

diagram:



attributes:

Name	Type	Use	Default
name	xsd:string	required	
single	xsd:string	optional	
interval	xsd:string	optional	
priority	derived by: xsd:integer	required	
globalId	xsd:ID	optional	

(b) PLCopen XML specification: *task* [16]

Fig. 10. IEC 61131-3 and PLCopen *task*

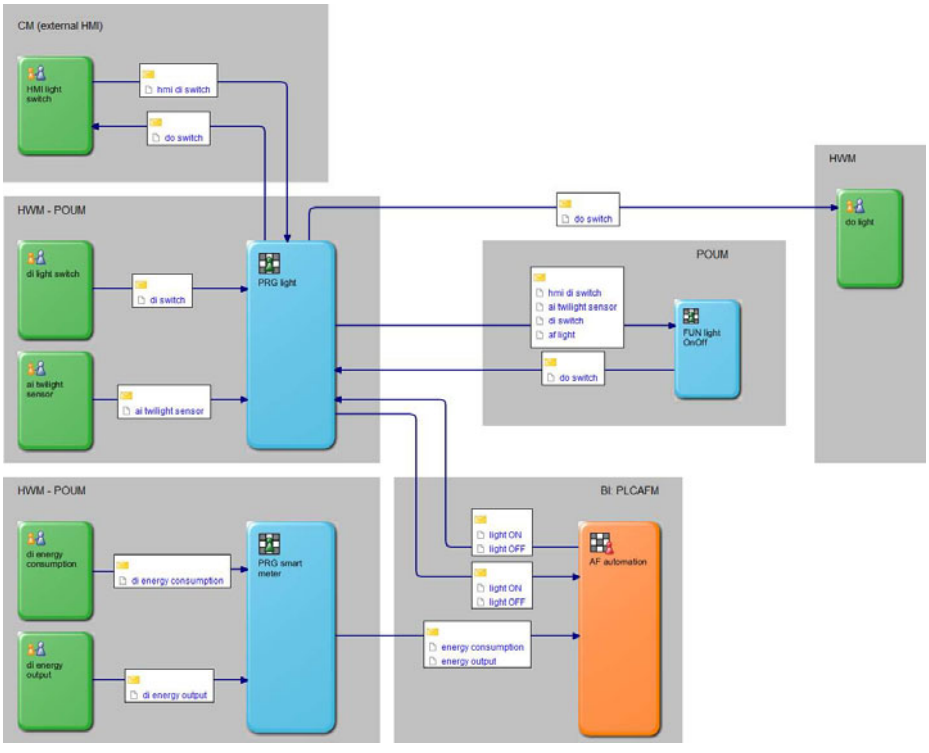


Fig. 11. ***S***ubject-oriented ***P***rogram ***O***rganisation ***U***nit ***M***odel

to the next [7]. The messages exchanged between Subjects are the corresponding variables in IEC 61131-3. A *Business Object* is represented by a single-element or multi-element variable [7, p. 43, 6.4.2.4]

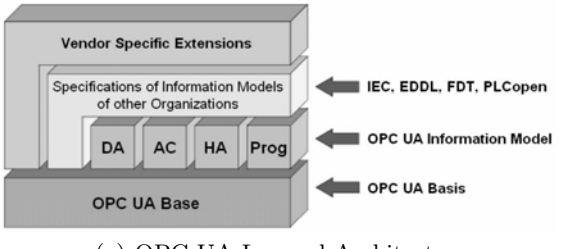
Figure 11 shows the S-POUM including the S-HWM, the S-CM and the BI for the Subject-oriented Application Flow Model. *PRG light* and *PRG smart meter* are two independent *programs* using the same BI for process communication with the S-AFM. The execution of the programs is configured in the Subject-oriented Task Model described in Sect. 4.2. The subject interaction diagram shows the communication structure of the PLC program. *PRG light* calls a function *FUN light OnOff* with a parameter list including the digital input *di switch*, the analog input *ai twilight sensor*, the external HMI variable *hmi di switch* and a variable *af light* which is a message out of the S-BPM represented by *light ON* and *light OFF*. The return value of function *FUN light OnOff* is the digital output *do light* which is a physical output and can not be declared twice. Therefore the variable *do light* just represents the return value type of the function. Figure 11 illustrates a IEC 61131-3 application using a subject oriented representation format. The subject interaction diagram represents the structure and the configuration of an application whereas a subject behavioral diagram represents the programming logic which is not shown in this document.

The use of a vendor specific application element, e.g. library or POU, is possible by importing them using the PLCopen XML standard format. The messages exchanged represent the parameter list, in context of IEC 61131-3 applications: input, output, and input-output variables.

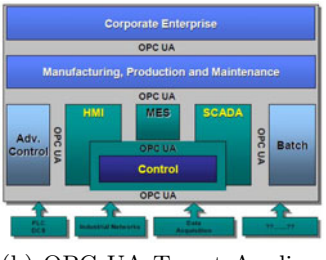
5 Subject-Oriented Application Flow Model (Layer 2: AFM) and S-BPM Communication (Layer 1)

Depending on the execution environment (PLC or Metasonic Flow) the transformation of the subjects is different. The S-CGM represents a IEC 61131-3 application using the hardware I/Os directly represented by a variable and the S-POUs represent IEC 61131-3 POUs. Whereas the Subject-oriented Application Flow Model can not use PLC I/Os directly because the S-AFM runs in Metasonic Flow and not on the PLC which is therefore used as hardware I/O interface. There are many possibilities to exchange data between a PLC and an IT system. The three-layer SCADA architecture using open system technology, allows, by using the OPC Unified Architecture, vertical data integration. The OPC Unified Architecture (OPC UA) will be known as IEC 62541 standards.

Figure 12 (a) shows the different layers of information models defined by OPC (OPC UA Basis, OPC UA Information Model) by other organizations (IEC, EDDL, FDT, PLCopen), or by vendors. The base specification covers OPC UA part 1- 7 including all known features from Classic OPC. OPC Unified Architecture is divided into 7 *Core Specific Parts* (part 1-7) and 5 *Access Type Specification Parts* (part 8-12). *Data Access* (DA, part 8) defines automation-data-specific extensions and allows a link to live automation data. Part 9 of the OPC UA is Alarm & Conditions (AC) which specifies an advanced model for process alarm management and condition monitoring. Part 10 Programs (Prog) specifies a mechanism to start, manipulate, and monitor the execution of programs. A mechanism to access historical data and historical events is defined in part 11 Historical Access (HA). [12, p.11]



(a) OPC UA Layered Architecture



(b) OPC UA Target Applications [6]

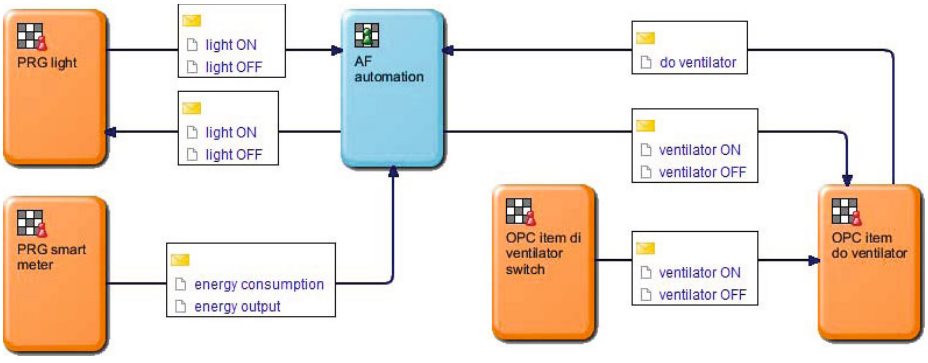
Fig. 12. OPC Unified Architecture

In context of S-AFM the Behavioral Interface illustrates the *DataItem* which is a link to arbitrary, live automation data (e.g. device data, calculated data, status information, dynamically-changing system data, diagnostic data) [5, p.8, 3.4]. The exchanged *message* (*business object*) represent two types of Variables, *Properties* and *DataVariables* [5, p.17] which differ in the kind of data they represent. Properties are server-defined characteristics of Objects, DataVariables and other Nodes, whereas *DataVariables* represent the content of an Object. [8, p.17] Figure 13 illustrates the subject-oriented Application Flow Model and its representation in the subject-oriented business model. *PRG light* and *PRG smart meter* represent the S-POUM and the data exchanged between programmable logic controller and the *AF automation* subject. The variable *OPC item di ventilator switch* represents a physical Input on the PLC which variable type is boolean. The *OPC item do ventilator* is an output variable representing a physical device. The subject interaction diagram shown in Fig. 13 (a) illustrates the subject interaction. A subject behavior diagram might show the process of switching on/off the ventilator using a subject oriented modeling methodology. The subject *AF automation* is the behavioral interface linking layer S-AFM and S-BPM shown in Fig. 2.

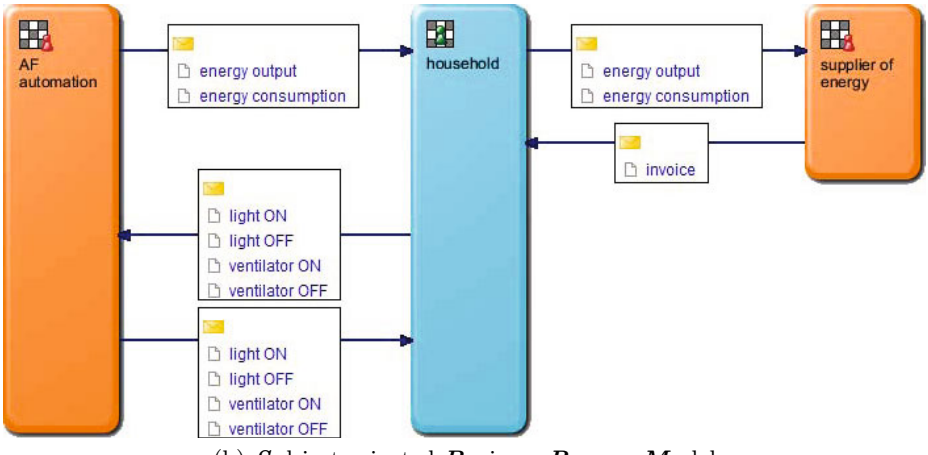
6 Summary and Issues for Further Research

A Behavioral Interface (BI) as shown in Fig. 6 "*... describes the observable behavior of the participation processes and their communication via message exchange. It can be automatically extracted from a partner's private process or modeled separately from scratch.*" [13, p.176] The use of a BI in combination with the jCPEX! platform allows cross-organizational business processes e.g. the communication between household and energy supplier who are exchanging the messages *energy output/consumption* and *invoice* as described in Sect. 3.2. A BI can also be used for mapping a *DataItem* (a link to arbitrary, live automation data) to a *business process subject*. A *DataItem* or a group of *DataItems* can be illustrated as a *Subject* used in a business process. The *messages* (*business objects*) exchanged between the two *external subjects* are the *DataItem* instances. To be able to communicate between a control layer device and a business process using vertical data integration, an application needs to be implemented for mapping the *DataItem(s)* to a *Subject* which can be done by *jCPEX! Automation Extension (AE) platform* including a *RuleEditor* which "*facilitates replacement of a partner dynamically dependent on certain conditions - even at runtime*". [13, p.176] In this context routing would not mean cross-organizational routing but rather inter-layer routing and the *replacement of partner* is on the one hand a rule-based adoption of automation processes influenced by a business process and on the other hand a rule-based initialization/instantiation of different business processes according to control layer data.

This *jCPEX! Automation Extension (AE) platform* is the basis for using a Subject-oriented Application Flow Model (S-AFM) which enables the control layer device act as an I/O interface, and use Metasonic Flow as workflow engine.



(a) ***S***ubject-oriented ***A***pplication ***F***low ***M***odel



(b) ***S***ubject-oriented ***B***usiness ***P***rocess ***M***odel

Fig. 13. Subject-oriented communication between S-AFM and S-BPM

A Human-Machine Interface (HMI) as part of the Subject-oriented Configuration Model (S-CM) might also use this platform if a SOA supports the communication between HMI and jCPEX! platform and supports the mapping between *HMI interface* and a *Subject*. A *RuleEditor* can distribute incoming events to one or, if *MultiPartyBIs* are supported, more *Subjects*. Another possibility would be to create a Subject-oriented Application Flow Model using the *jCPEX! AE* to dispatch messages to one or more business process Subjects, HMIs or *DataItems* represented by subjects.

The process control layer (RTUs/PLCs) (*Control Layer*) is enhanced by providing the ability to model PLC application code in a subject-oriented way. In a first step not the detailed control process (provided by the vendor of a machine or implemented by a specialist) is focused on, but the IEC 61131 *Configuration -, Communication - and Software Model* (illustrated as subject interaction diagram) are in the center of interest. The logic of an application implemented in the body of *programs*, *function blocks* and *functions* was not treated in this paper, but could be modeled by using the subject behavior diagram.

The integration of POUs (libraries, external code, ...), using a *Subject* or a *Multi Subject*, is adequate to illustrate processes. MDSE is possible because PLCopen provides a XML specification for the IEC 61131-3 standard, the only standard in industrial control programming, and the OPC Foundation provides the communication basis with the OPC Unified Architecture (draft standard IEC 62541) therefore a *BPM system* additionally has to provide *code-transformation ability* and the *ability to interact with these generated applications*.

Summing up the approach described, subject-oriented methodology can be used for business process modeling as well as IEC 61131-3 automation modeling. In combination with jCPEX! Automation Engineering (AE) platform and OPC UA an inter-layer and/or cross organizational use is applicable.

References

1. Cupek, R., Fojcik, M., Sande, O.: Object Oriented Vertical Communication in Distributed Industrial Systems. In: Kwiecień, A., Gaj, P., Stera, P. (eds.) CN 2009. CCIS, vol. 39, pp. 72–78. Springer, Heidelberg (2009), http://dx.doi.org/10.1007/978-3-642-02671-3_8, doi:10.1007/978-3-642-02671-3_8
2. DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE: Speicherprogrammierbare Steuerungen - Teil 3: Programmiersprachen, iCS 25.040.40; 35.060; 35.240.50 (2003)
3. DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE: Speicherprogrammierbare Steuerungen - Teil 1: Allgemeine Informationen (IEC 61131-1:2003); Deutsche Fassung EN 61131-1:2003, iCS 25.040.40; 35.240.50 (2004)
4. DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE: Function blocks - Part 1: Architecture (IEC 61499-1:2005); German version EN 61499-1:2005, iCS 25.040.40; 35.240.50 (2006)

5. DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE: OPC Unified Architecture - Part 8: Data Access (IEC 65E/98/CDV:2008); English version FprEN 62541-8:2008, iCS 35.200; 35.240.50 (2008)
6. DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE: OPC Unified Architecture - Teil 1: Übersicht und Konzepte (IEC 65E/92/CDV:2008); Englische Fassung FprEN 62541-1:2008, iCS 35.200; 35.240.50 (2008)
7. DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE: Programmable Controllers - Part 3: Programming languages; English version (IEC 65B/725/CD:2009), iCS 35.080 (2009)
8. DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE: OPC Unified Architecture - Teil 3: Adressraummodell (IEC 62541-3:2010); Englische Fassung EN 62541-3:2010, iCS 35.200; 35.240.50 (2011)
9. Endi, M., Elhalwagy, Y., Hashad, A.: Three-layer plc/scada system architecture in process automation and data monitoring. In: 2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE), vol. 2, pp. 774–779 (2010)
10. Fleischmann, A.: What is S-BPM? In: Buchwald, H., Fleischmann, A., Seese, D., Stary, C. (eds.) S-BPM ONE 2009. CCIS, vol. 85, pp. 85–106. Springer, Heidelberg (2010), http://www.springerlink.com/content/m4105554174112q7/
11. Frey, G., Thramboulidis, K.: Einbindung der iec 61131 in modellgetriebene entwicklungsprozesse, http://www.aut.uni-saarland.de/uploads/media/GF_KT_AUTOMATION_JUNE_2011.pdf
12. Mahnke, W., Leitner, S.H., Damm, M.: OPC Unified Architecture. Springer, Heidelberg (2009)
13. Meyer, N., Radmayr, M., Heininger, R., Rothschädl, T., Fleischmann, A.: Platform for Managing and Routing Cross-Organizational Business Processes on a Network Router. In: Schmidt, W. (ed.) S-BPM ONE 2011. CCIS, vol. 213, pp. 175–189. Springer, Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-23471-2_13, doi:10.1007/978-3-642-23471-2_13
14. Opc foundation (October 25, 2011), http://www.opcfoundation.org
15. Plcopen (October 25, 2011), http://www.plcopen.org
16. PLCopen: XML Formats for IEC 61131-3, 2.01 edn. (2009)
17. Selic, B.: From model-driven development to model-driven engineering. In: Euromicro Conference on Real-Time Systems, p. 3 (2007)
18. Selic, B.: Personal reflections on automation, programming culture, and model-based software engineering. Automated Software Engineering 15, 379–391 (2008), http://dx.doi.org/10.1007/s10515-008-0035-7, doi:10.1007/s10515-008-0035-7