VTT Technical Research Centre of Finland

System switch specification

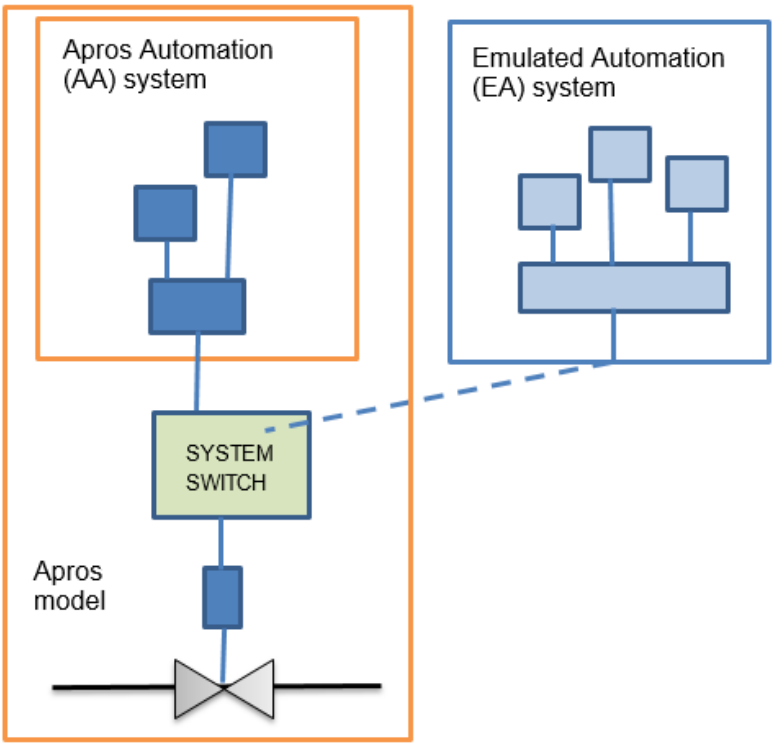
Lappalainen, Jari; Lackman, Tuomas

Published: 01/01/2018

Document Version
Publisher's final version

RESEARCH REPORT

VTT-R-04126-18



System switch specification

Authors: Jari Lappalainen, Tuomas Lackman

Confidentiality: Public

Report's title

System switch specification

Customer, contact person, address	**Order reference**
Business Finland, Kari Koskela PL 358, 00181 Helsinki, Finland	
Project name	**Project number/Short name**
Raising the level of automation and quality in plant engineering (Engineering Rulez)	Tekes_Engineering Rulez / 112220
Author(s)	**Pages**
Jari Lappalainen, Tuomas Lackman	17
Keywords	**Report identification code**
Simulation aided automation testing, automation system, system switch, process plant, simulation	VTT-R-04126-18

Summary

This work contributes towards straightforward simulation model transformation from early engineering phases into automation testing of process plants. The work was conducted in the Engineering Rulez research project, under the task T2.2.2 "Toolset for automatic transformation of analyses simulation model into testing model". The focus was in easy switching between different automation solutions, such as safety logics in the Apros model and an emulated actual system, and comparison of the results from these systems. Besides Apros, the environment for automation testing includes the new version of the Testing Station software that has been developed in other tasks and sub-projects of the Engineering Rulez project.

New modules were designed to support the switching between alternative automation circuits in the Apros simulation software, taking into account the result comparison feature. The relevant options for implementation are discussed in this report. In addition, sample implementation of Analog and Binary System Switches using the User Component approach were experimented and the main findings are reported and discussed.

The value comparison part of the original project plan did not proceed to implementations. That was decided to be left done after the current active development phase of the Testing Station 3 software. An important detail question discussed in this report is to find the best way to support multiple violations for the Violation Monitors feature so that the comparison continues through the entire test sequence.

Confidentiality	Public

Espoo 26.9.2018

Written by	**Reviewed by**	**Accepted by**
Jari Lappalainen Senior Scientist	Jouni Savolainen Senior Scientist	Juha Kortelainen Research Team Leader

VTT's contact address

Vuorimiehentie 3, Espoo, PL 1000, 02044 VTT, Finland

Distribution (customer and VTT)

Business Finland, one copy
VTT Archive, one copy

Preface

This brief study is to facilitate future steps towards efficient model transformation from early engineering phases into automation testing of process plants. This promotes increased efficiency and quality in the automation tests. We would like to thank Topi Tahvonen from Fennovoima, and Vesa-Matti Tikkala from Fortum for the valuable discussions and ideas that this work is largely based on.

In Espoo, Sep 4th 2018

Authors

Contents

1. Introduction

Dynamic simulation models have many important roles in nuclear engineering. The various uses of the system model set different requirements for the process model and for the automation. The automation functionality can be configured as part of the model, or used via a link with the actual automation product or its virtual version. Despite of the different use cases, typically major parts of the model can be re-used. When an engineering project has a good commitment to modelling and simulation activities, the tools and methods should support transformation from a task to another. As an example, a model that is first used for accident analysis can be developed further to serve in the verification of the automation system. Accordingly, a large potential for cost savings and quality improvement exists in the model development and transformation phase. With this background, we discussed the needs and ideas with the domain experts from Fennovoima and Fortum, and focused on the advanced interface that could support switching of one automation solution into another, and more important, provide new information of the functional differences between the systems in practical tests. This work is linked with the parallel development of a new automation testing tool, Testing Station 3, parts of which have been conducted in the parallel Engineering Rulez sub-projects.

2. Goal

The goal is to provide early configuration of a dynamic simulation model with features that later on will support the model's utilisation in advanced automation testing. The new features would benefit the tests by enabling easy switching of alternative automation circuit implementations and comparing of their outputs. The Apros simulation software was used as the working environment in the study.

The focus was set in cases, which contain the automation functionalities as Apros models and as an emulated automation solution. The basic idea is to introduce an interface module, hereafter called as a System Switch. The modules would be used in the model configuration in the early phase of the project, even if they do not have a role in the early engineering tasks, such as accident analysis. When the project reaches the phase of automation testing, the simulation system will be connected with alternative automation circuits, e.g. an emulated version of the actual automation of the targeted plant. The System Switch modules make switching between the Apros model and emulated automation solutions easy and moreover, produces new, valuable information of the differences between the two solutions. The comparison functionality is planned to be implemented in the Testing Station 3 software. Figure 1 illustrates the use of a System Switch in automation testing.

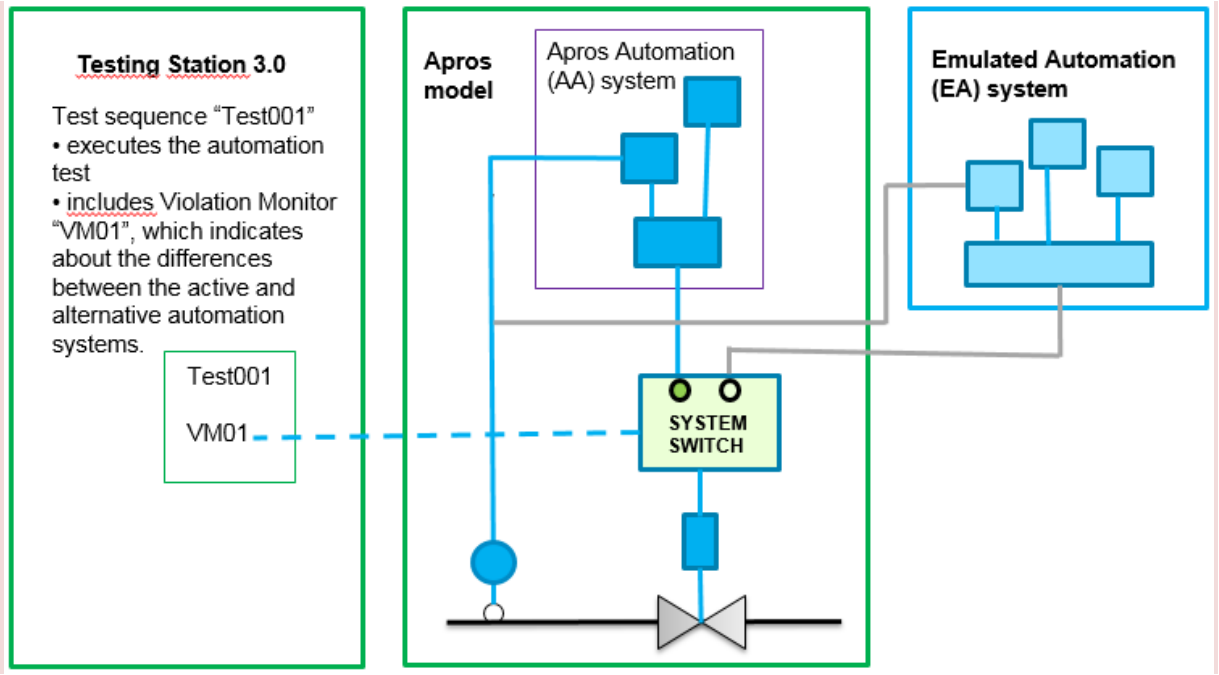


Figure 1. A System Switch selects the active source for each automation interface signal. The Testing Station software tracks and reports the possible differences between the active and the alternative signals.

3. Description

The name System Switch does not have any earlier reference. It was considered to characterise the module, yet other name candidates were proposed as well, for example: System Switcher, Source Switch/Switcher/Selection and Circuit Switch/Switcher/Selection. The words source or circuit are relevant options in the sense that often the 'system' behind is not large at all, but just a simple logic circuit. Furthermore, the final output is single signal. However, for simplicity, the name *System Switch* is used in this document.

The concept of System Switch is applicable for binary and analog automation systems. The word Analog or Binary is intuitive to add in all of the name candidates above. The primary interest, however, is in the binary side, because the comparison of alternative binary automation solutions is much more straightforward. This is mainly due to the continuous nature of signals in analog systems, which produces a practically infinite number of states. Even in most simple analog comparison, limits must be defined which is always prone to subjective choices. The number of analog implementation variants is nearly boundless, and the dependency on the process inputs is continuous. Consequently, the comparison might thus become misleading. Despite of these issues, Analog System Switch is discussed here equally to the binary counterpart.

3.1 User stories

The need and use case for the System Switch was written in a form of user stories, as follows:

1. As an engineer of Automation testing, I'd like to use an interfacing module, which enables clear management of alternative automation circuits/solutions, and allows easy comparison between their operation.

2. I will first configure a logic circuit with Apros Automation (AA), but later on, I'll set-up and run the same logic via an emulated automation (EA) solution. It's worth emphasising that only one solution is truly controlling at a time. That one can be called as *active*, while the other one is *inactive*.

3. I'd like to prepare for and make switching between the AA/EA solutions fluently. I'd like to use some grouping by name for the components in a system, say RPS (Reactor Protection System). I'd like to see clear notification event when the switching has taken place.

4. I'd like to use the results of the inactive automation solution as a reference for the active one. Often, I get results from both automation solutions simultaneously during the same simulation run. This is especially useful when AA and EA are functioning independently. Then important signal(s) value(s) should be continuously evaluated against each other. Possibly a certain relaxation (resolution) with respect to time is needed. Often the inactive system's execution depends on the active counterpart, because of the calculated outputs into and the feedback from the process. This method may be used also in these dependent cases, when carefully considering the role of the dependency when interpreting the results.

5. I'd like to prepare for this comparison need already in the AA modelling phase, and later on, when the EA is available, set-up respective Violation Monitors (VM) in the Testing Station tool. The VMs should be capable to reactivate themselves after each violation to capture possible multiple violations between the alternative systems' signals.

The business value comes from the early preparation in the model building for the needs of the coming automation testing, from the clarity of the operation with the new interfacing module, and from the insights and added efficiency due to the intuitive functional comparison of the alternative system implementations.

3.2 Implementation options

The current working method with alternative automation solutions uses binary and analog switches (in Apros, BINARY_SWITCH and ANALOG_SWITCH), which have signal inputs from AA and from external EA. The latter signal is only an interfacing module, which gets its value externally, e.g. via OPC UA or ACL. For the new System Switch, a requirement is to operate with a single module that allows both the switching function and the comparison function. The comparison function is planned to be implemented in the Testing Station software, which should have features for flexible configuration of Violation Monitors for System Switches. Additionally, the module should send notifications (events) when the output value changes.

The implementation alternatives are:

- User Component
- New Apros Automation module type
- Extending a current one (BINARY_SWITCH and ANALOG_SWITCH).

The latter option could be realised by introducing some new attributes, and creating a dedicated symbol for the new purpose. It was not investigated or specified further, because the User Component approach was considered better due to less effort and fast development cycles. The option of a genuine Apros module type for System Switches might become relevant if issues are revealed with the User Component solution, for example, related to Violation Monitor implementation or connection with external automation solutions. More details on these topics will follow in the next sections.

3.3 System Switch properties

The following introduces the binary and analog versions of System Switch, giving the properties of the module.

3.3.1 Binary System Switch

Table 1 below lists and describes the properties of the Binary System Switch. The comparison task is proposed to be done in the Testing Station side. If the violation state is solved in the Apros side, an additional property is needed, e.g. 'Violation ON'.

The model builder is supposed to know in most cases, whether the comparison is a proper thing to do regarding the nature of the logic circuit. In other words, the property 'Violation monitor OK' is a hint from the model builder for the Testing Station configurator.

Relaxation time is considered useful as the alternative systems may have slightly different timing due to minor functional differences and communication delays.

Table 1. Binary System Switch properties.

Property/terminal	Type	Default	Description
Input signal 1	Binary_signal	-	Name of the 1st input signal. This is for the signal coming from the AA solution.
Input 1 value	Boolean	false	Value of the 1st input signal.
Auxiliary input value	Boolean	false	Value of the 2nd input. Normally this value comes externally.
Selection	Integer	1	Selection of the active system (1 or 2).
System identifier	String	-	Short name/abbreviation of the automation sub-system if applicable. Examples: PPS, PAIS, RTS. This can be used to find (e.g. SCL query) all modules for a specific emulated system.
Relaxation time	Double	1.0 s	Time interval, which the active and inactive systems' values are allowed to deviate without any violation. Time is counted from the active system's change symmetrically backward and forward in time.
Event	String	'input 1/input 2'; 'true/false'; 'on/off'	Event text that Apros sends if the user has 'Events OK' = true. Alternative implementations: 1) when the output changes, 2) when the input source changes.
Events OK	Boolean	false	Selection for sending events: • False: no events sent • True: events are send
Violation monitor OK	Boolean	true	Selection for informing if the value comparison VMs are reasonable in this case:

			• False: VMs not recommended • True: VMs are reasonable

An unclear situation takes place if the output values behave pulse-wise, i.e. it is desired that the signal changes for a short time interval only. Then comparing the two signals within the Relaxation time needs careful specification and maybe optional functional modes.

3.3.2 Analog System Switch

Table 2 below lists and describes the properties of the Analog System Switch. The comparison task is more complicated in this case, and it must be considered, whether the proper place to solve the status is in Testing Station side or in the Apros side. In the latter case, a property for the violation is needed, e.g. 'Violation ON'.

Table 2. Analog System Switch properties.

Property/terminal	Type	Default	Description
Input signal 1	Analog_signal	-	Name of the 1st input signal. This is for the signal coming from the AA solution.
Input value 1	Double	0.0	Value of the 1st input signal.
Auxiliary input value	Double	0.0	Value of the 2nd input. Normally this value comes externally.
Selection	Integer	1	Selection of the active system (1 or 2).
System identifier	String	-	Short name/abbreviation of the automation sub-system if applicable. This can be used to find (e.g. SCL query) all modules for a specific emulated system.
Relaxation time	Double	1.0 s	Time interval, which the active and inactive systems' values are allowed to deviate without any violation. Time is counted from the active system's change symmetrically backward and forward in time.
Relaxation value	Double	1.0	Value zone (deadband), which the active and inactive systems' values are allowed to differ (within Relaxation time) without any violation. The allowed deviation zone is altogether 2x this value, as it covers upwards and downwards of the current active system's value.
Event	String	'input 1/input 2' 'high/low'	Event text that Apros sends if the user has 'Events OK' = true. Alternative implementations: 1) when the output changes, 2) when the output value exceeds the allowed zone in respect of time and value.
Events OK	Boolean	false	Selection for sending events: • False: no events sent

			• True: events are sent
Violation monitor OK	Boolean		Selection for informing if the value comparison VMs are reasonable in this case: • False: VMs not recommended • True: VMs are reasonable

3.4 Sample implementation as a User Component

3.4.1 Configuration

The System Switch modules were configured as User Components to initially test the feasibility of the approach. Figure 2 shows the configuration of the Analog and Binary System Switch modules. As can be seen, the basic function is simple: switching between two input signals. Figure 3 shows the property definitions.

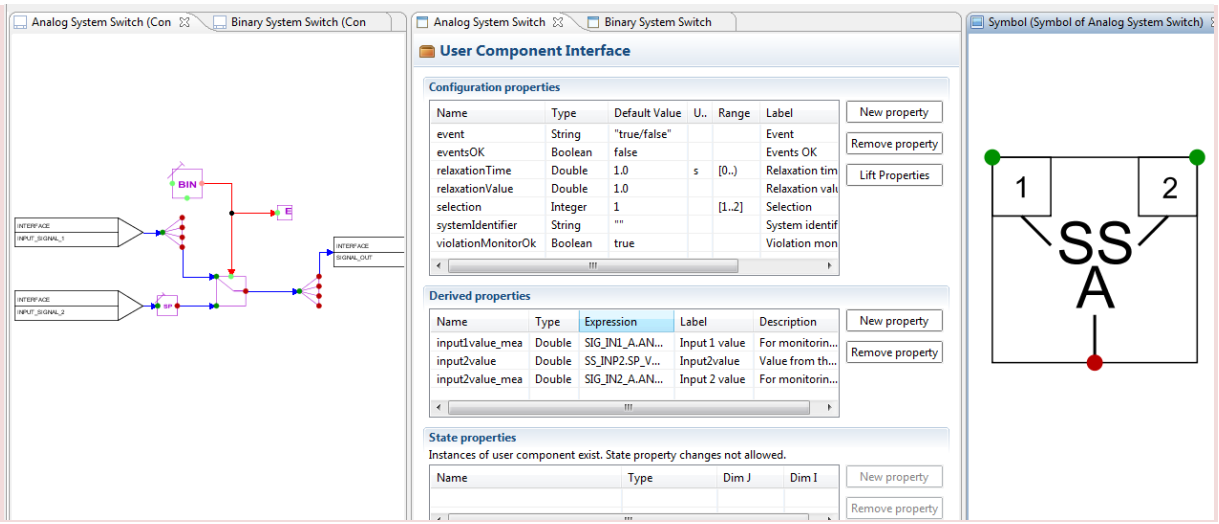


Figure 2.An Analog System Switch configuration, properties and symbol as User Component.

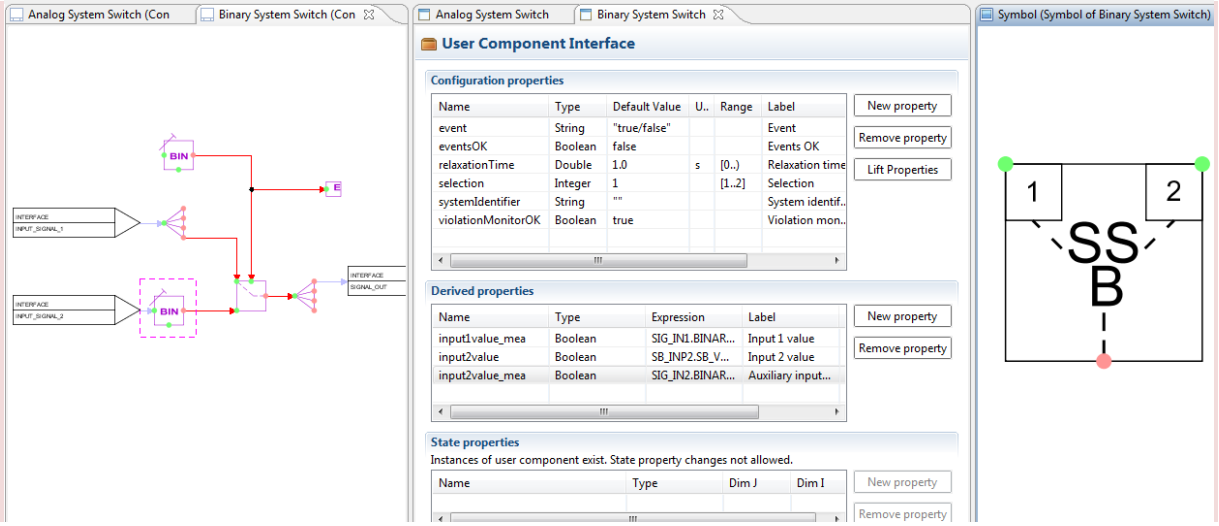


Figure 3. A Binary System Switch configuration, properties and symbol as User Component.

3.4.2 Testing, case External Automation via OPC UA

Figure 4 presents a test case conducted with the modules, and Figure 5 repeats the chart windows for clarity. In this demonstration, the Apros automation (AA) is represented by a signal generator. The external automation (EA) was realised with an OPC UA connection from the Unified Automation GmbH's UaExpert software, version 1.4.4275 (Unified Automation 2018).

The starting condition in the test for both modules is that Selection 1 is active, i.e. the Apros automation signal (input 1, green colour in the charts) is outputted (blue). Then the milestones indicate the following steps. Note, for clarity, the actions taken slightly after the milestone moment and Analog System Switch's output signal's scaling is slightly broader.

1. The EA values changed: −0.5 → 1.23; true → false. Both System Switches follow input 1 (green curves).
2. Selection changed from 1 → 2. The outputs start following the EA signals (red).
3. The analog EA value changed 1.23 → −0.9. The output of the System Switches follows.
4. The binary EA value changed true → false. The output of the System Switches follows.
5. Selection changed back 2 → 1. The outputs start following the AA signals again.

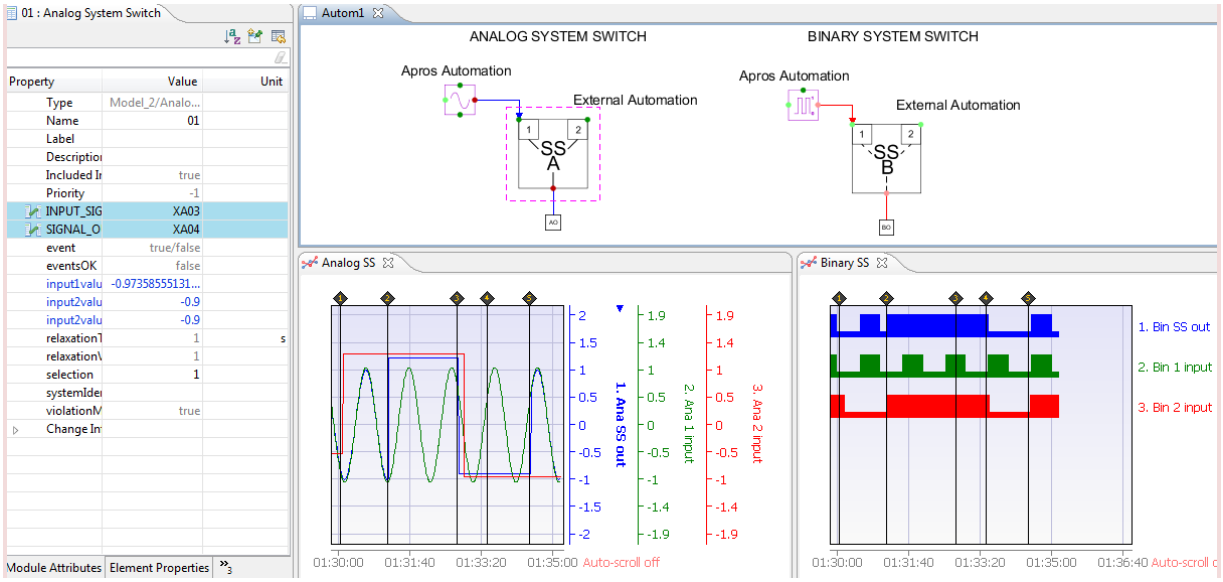


Figure 4. A demonstration of source switching with Analog (left) and Binary (right) System Switches.

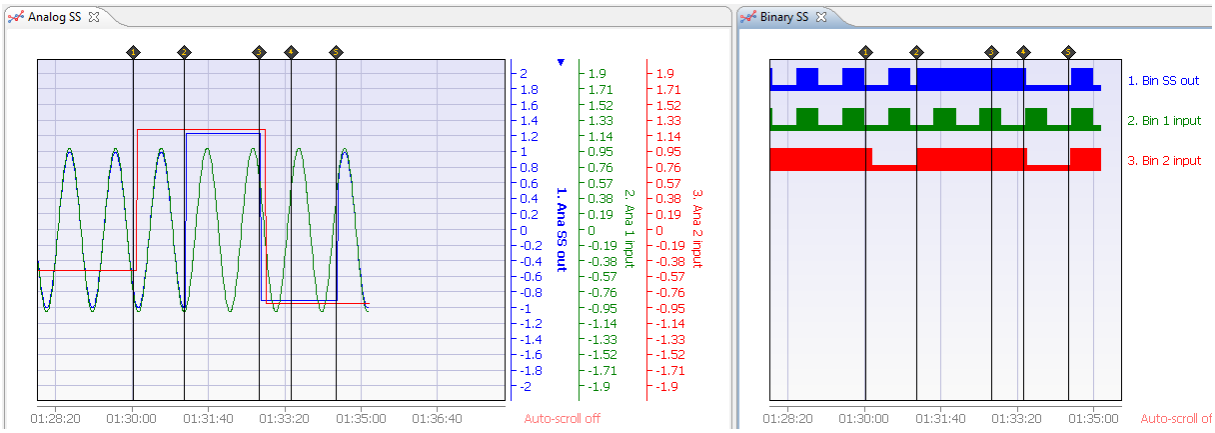


Figure 5. More detailed view of the trends in the test case.

Figure 7 illustrates how the input 2 was searched and modified via OPC UA. The UaExpert session was connected with the Apros OPC UA server and the model database was searched for the appropriate tags. Here we see a drawback of using the User Component approach: the System Switch instance does not express itself in the TYPES list, only the Apros module types are listed. Consequently, the input signal 2 of the Analog System Switch must be searched from the User Component's substructure. In this case, the value is given for a SET_POINT named 'SS_INP2@1', see Figure 6 and the left side of Figure 7. The value itself, −0.9 in this case, can be seen in the right side of Figure 7.

Yet not shown in the figures, the next Analog System Switch, when added into the Apros model, received a corresponding SET_POINT name: 'SS_INP2' and the next one 'SS_INP2@2'. The server needed a reconnection to see these updates in the Apros database.

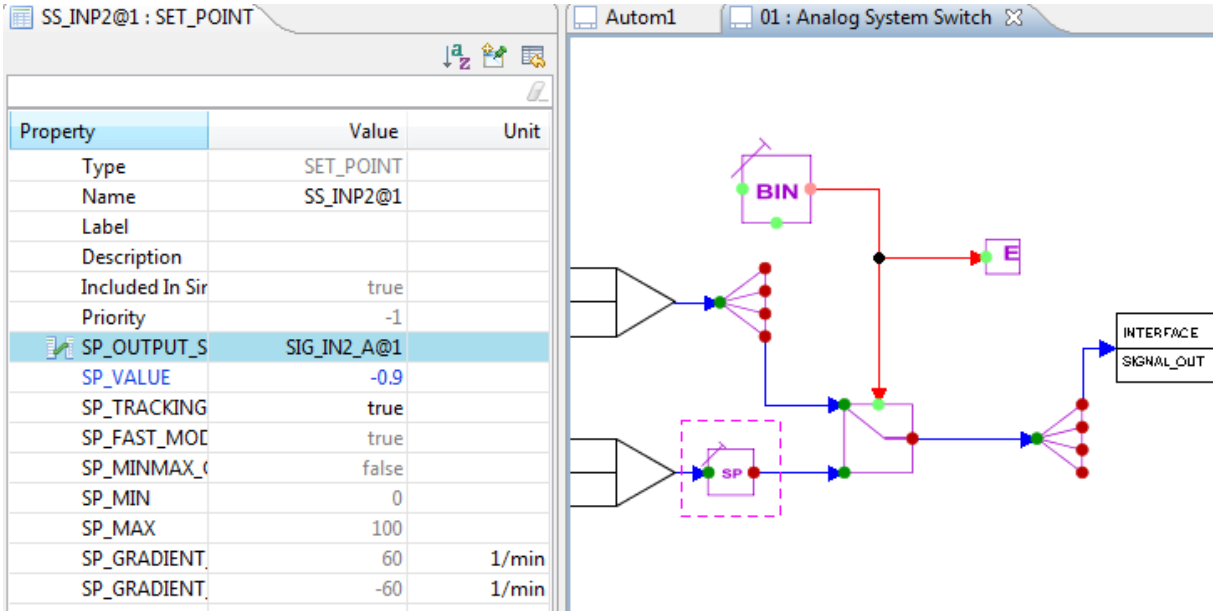


Figure 6. The substructure of Analog System Switch reveals the module name where the external connection and the value updates can be made.

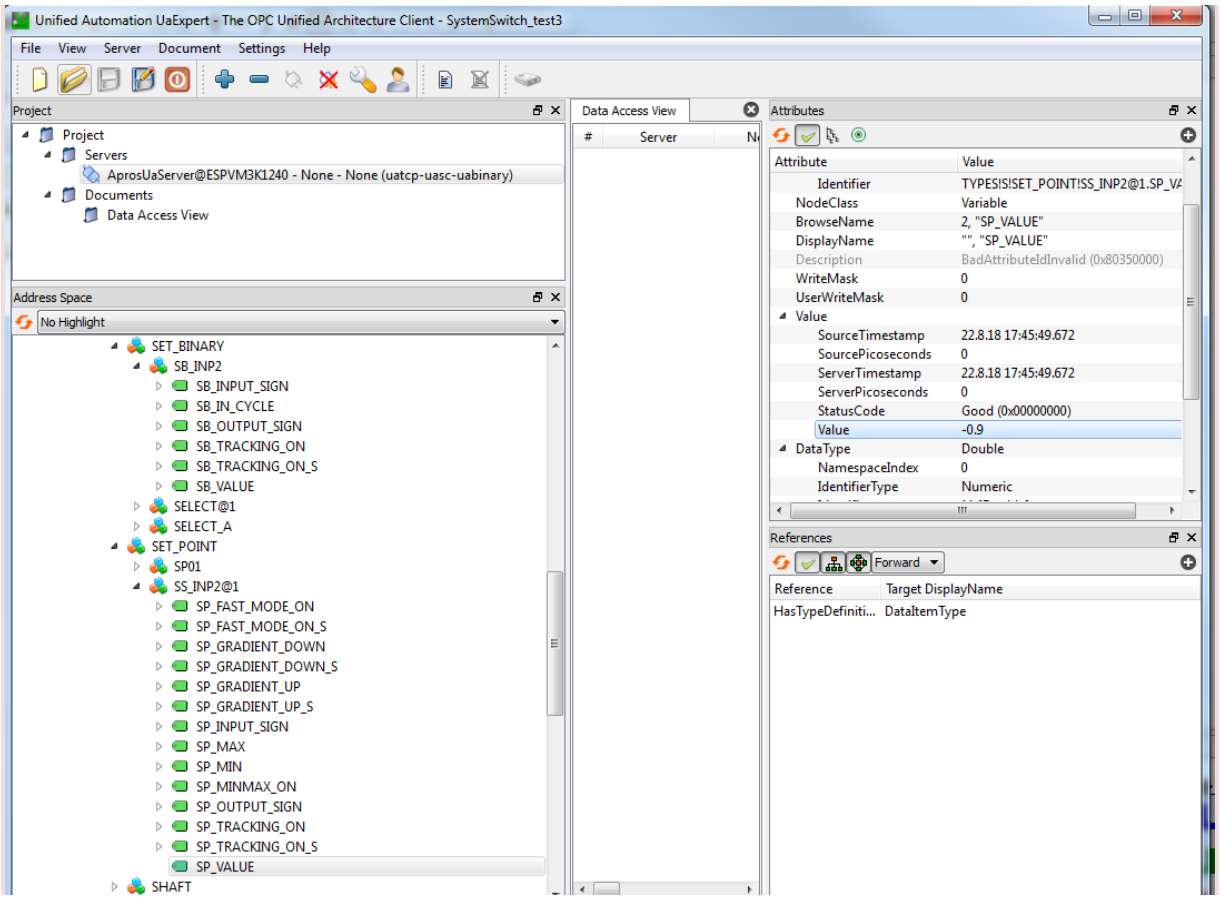


Figure 7. A view of the UaExpert. The value of the EA signal was changed in the right side view, in the field 'value' (now −0.9).

The use of the System Switches would be more straightforward if the properties of the User Component itself could be browsed and connected via OPC UA. However, as the User Component is a Simantics database entity, not having any Apros database correspondence, the OPC UA sees all the created substructure as unstructured objects (flat). It is worth mentioning that the third property group of the User Component, the State properties (note, the name has

been recently updated, earlier this naming was used for Derived properties), see Figure 3, does create a new module type with real attributes. This would allow more intuitive browsing via OPC UA. However, the main purpose of this property type is to allow User Components to store data in the Apros database, and its use differs (has restrictions) from the other User Component properties. Thus, the state variable method was not suitable for this purpose. It is worth further discussions for possible future development.

3.4.3 Testing, case external automation via Value Transmitter

Figure 8 presents the set-up in this test case. This demonstration mimics one possible way of attaching an external automation: using Apros external models, such as EXT_10_REF_MODEL. In this method, the external automation is called from an external Fortran or C program, which inputs/outputs are connected using the Apros module type VALUE_TRANSMITTER. In this case, the Apros automation is represented by two signal generators, a sine wave for Automation 1 and triangular wave for Automation 2. For simplicity, the test model moves the output from the wave generators to the value transmitter and thereupon to the System Switch's input 2.

The starting condition in the test case for both modules is that Selection 1 is active, i.e. the Apros automation signal (input 1, green colour in the charts) is outputted (blue). The milestones indicate the following steps:

1. Selection changed from 1 → 2. The outputs start following the input 2 signals (red; the values are triangular wave and more frequent binary pulses).

2. The values from the source 2 (mimicking the external automation) are generated using doubled period times.

3. Selection changed back 2 → 1. The outputs start following the Apros automation signal again.

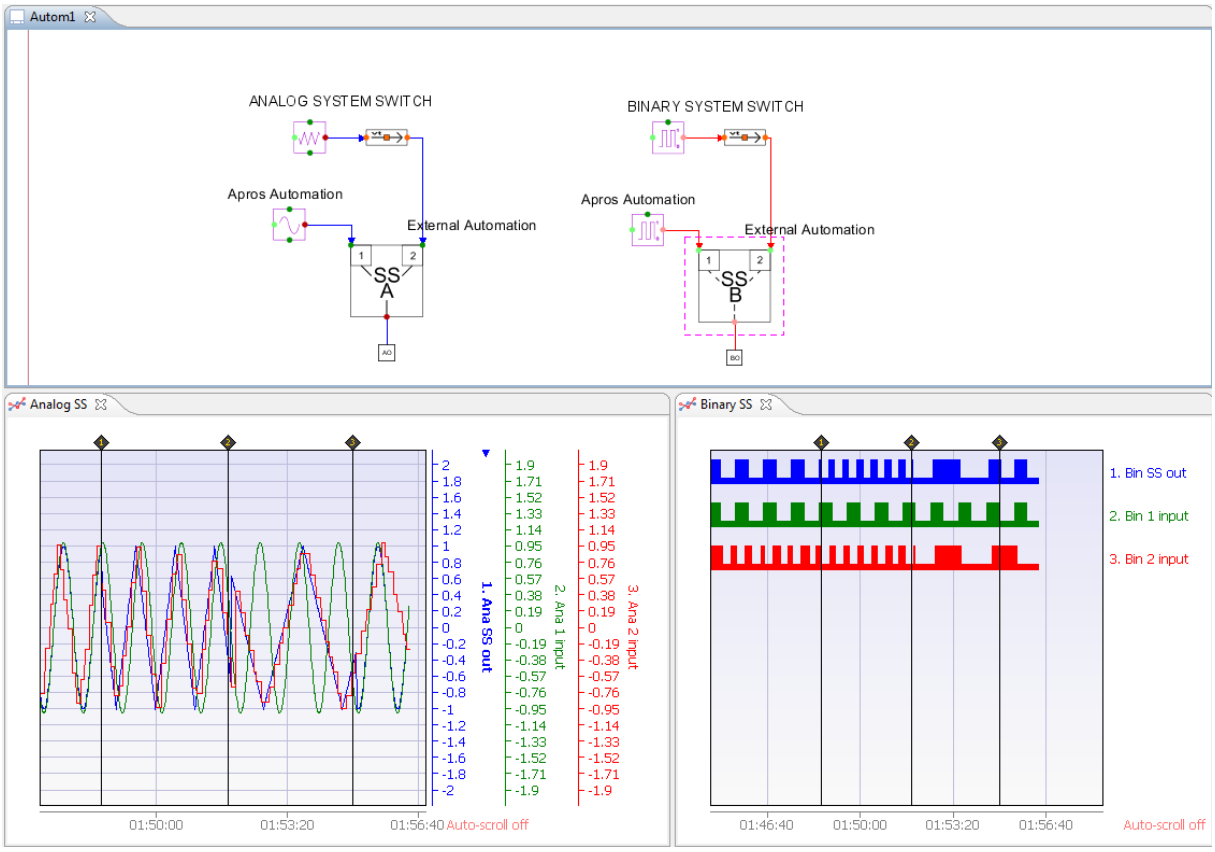


Figure 8. A demonstration of source switching with Analog (left) and Binary (right) System Switches. In this case, both input sources are in Apros. The input source 2 mimics the case that the external automation is linked with the external model approach and values are transferred via VALUE_TRANSMITTERs.

4. Results and discussion

The System Switch test implementation with the User Component approach was straightforward with relation to the switching function. The comparison function was not addressed in these tests, but it is considered to be implemented in the Testing Station 3 software. Thus, the new components were tested from the interfacing and switching points of view only, and not from the comparison point of view. The tests revealed that the Apros database provides only a flat view to the modules via the OPC UA connection, so corresponding tags for the User Components' properties are not intuitive to find. This limitation might change in the future or some shortcut way might be found. Alternatively, the implementation of the switching functionality could be Fortran-coded in the Apros ANALOG_SWITCH and BINARY_SWITCH module types, or in totally new module types.

With respect to the value comparison functionality, which after all is the real motivation and value provider in this approach, the results are scarce. The active development phase of the new Testing Station software was challenging, affecting to the work in this project task.

The following issues were discussed and recognised:

- The System Switch implementation may introduce delays of one time step, depending on the implementation approach used. This issue should be carefully addressed and additional delays avoided if possible.

- Output value changes pulse-wise within the Relaxation time – desired function should be clearly specified.
- The Violation Monitor (VM) definition language (MTL in SCL) should allow easy definition of both Analog and Binary System Switches.
- Currently the VM implementation supports only three outputs: 1) the monitor has been violated (with the time it was first violated), 2) the monitor has not been violated, and 3) it is unknown whether the monitor has been violated or not. It will need additional effort to allow the monitors to have multiple violation occurrences.
- The VM purpose has been so far to announce when the condition (requirement) breaks. In the context of System Switch, there is an alternative approach: report the start and stop times for a violation. This feature would provide more compact way of recording violations, which originate from comparison of two signals. Perhaps single VM could present a list of this kind of violation periods.
- When developing the implementation as a User Component, two issues have arisen:
    - SCL functions within User Components do not currently work in the headless Apros. This disallows the User Component implementation, when Apros is run in a server environment, so additional development needs to be done. Before that, the implementation can be done so that it works properly in the desktop environment.
- The events' role in the System Switches needs to be defined more precisely. There could be many specific event functions for different purposes. The current test implementation gives an event when the source is changed, which is symmetric function for both System Switches.
- One problem with the definition of the difference between the two signals by VMs is that the current implementation does not have support for saving the state of the violation monitor execution. That is, if a model export and import is made, the violation monitor acts differently to a simulation which was run in one piece (in case that there is a monitor that needs to carry an event over the export/import boundary).

It is worth emphasising that since almost every automation circuit uses some process inputs, potentially the currently inactive system uses such inputs, which the active system has influenced. In this sense, the alternative automation systems are not fully independent of each others. Accordingly, the comparison must be conducted with special care. This is seen as a limitation for the System Switch approach, especially in studies of analog systems.

5. Summary

The work in this research task produced specification for the System Switches of both analog and binary types in the Apros software. Sample User Component implementations were accomplished and their testing reported. The value comparison part did not proceed to implementations. Considering the entire functionality and the current active development of the Testing Station 3 software, this part still requires a thorough discussion about the desired features and finding the best option for implementation. Focal question is how the value comparison can be executed in the Testing Station software so that multiple violations are supported in cloud simulations, or some other way to report on multiple violations revealed by the alternative automation signals mutual comparison. Because these violations may be numerous, special requirements for the results management, visualisation and reporting must be tackled. In addition, the nature of automation testing, where external automation solutions are involved, substantially differs from the currently used simple test cases in the Testing Station 3 software development, where all the functionality is within the Apros model. Some limitations for the test case architecture are probable, if tests are to be computed in the cloud. The system switch approach might therefore be more relevant for desktop type of testing sessions.

References

Unified Automation, OPC UA Clients – Downloads, UaExpert v1.4.4. https://www.unified-automation.com/downloads/opc-ua-clients/file/download/details/uaexpert-v144.html. Accessed 7.8.2018.