

# ViVe – the Visual Verifier

Verification, Validation and Certification of Automation Systems using Formal Methods

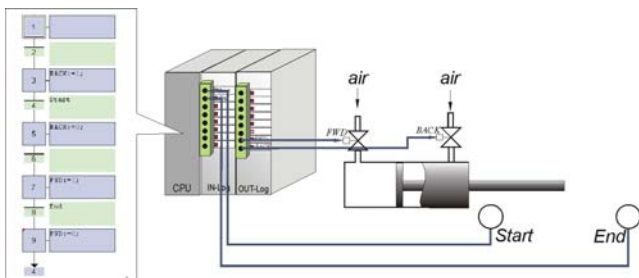
## What is formal verification ?

In computer science *formal verification* is an act of proving the correctness of programs by using mathematical methods and models.

In the industrial automation research formal verification is understood as an automatic alternative to the simulation-based testing and debugging, improving dependability and reliability of automation systems. Unlike testing and simulation, the formal verification can explore complete set of system's state space and mathematically prove that no undesirable or dangerous behaviour occurs. *This can be very helpful in proving compliance with various certification requirements.*

A *state* of the system is characterized by the values of its parameters (both in plant and in controller). The *state space* includes all states where the system can transition from the initial state driven by its controller. *Events*, such as on/off switching of some output variables, transition the system from one state to another.

Sometimes the state space is just a linear chain of states, but in a more realistic view it usually a more complex structure. Even for a simple cylinder as the one shown in the Figure below its *reachability graph* can include several alternative traces of behaviour as will be shown in the Figure on the right. Some states can have transitions to multiple states, and, as a result, multiple behaviors can appear in different circumstances. However, during a simulation session only one single trajectory is seen.



## What is it good for?

Formal verification can prove that the system behaves as expected. Usual software testing and debugging techniques are not much applicable in automation due to the following circumstances complicating the analysis:

- Programs are **cyclic**, executed millions times per second, so using step-by-step debugging and breakpoints is not practical

- The **logic of execution** can be further complicated by **multitasking, scheduling** and other particulars of the run-time platform
- **Communication delays** and data sampling rates can essentially influence the overall behavior of a system

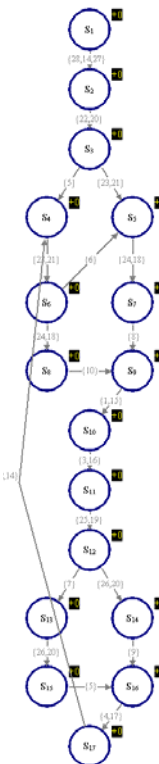
Verification can help to investigate:

- Influence of **malfunctions** (of sensors, actuators, or other equipment units) on the behavior of the system
- Impact of various combinations of **concurrent processes** in the plant that are difficult to predict in advance

- **Composition:** building system from **re-usable components** and validating its safety, correctness, timing, etc.

- **Reconfiguration:** i.e. execution of the same application on different HW/SW architectures

Thus, formal verification can have positive impact not only on dependability but also on flexibility and reconfigurability of automated manufacturing systems.



## How to do it?

First, the verified system is modelled using some *formal language*, e.g. state machines, Petri nets, NCES, etc. The model can represent a closed-loop plant-controller system, or only its controller (open – loop). The model of the plant has to be designed manually by control engineers, while the model of the controller can be built automatically given its code in some programming language.

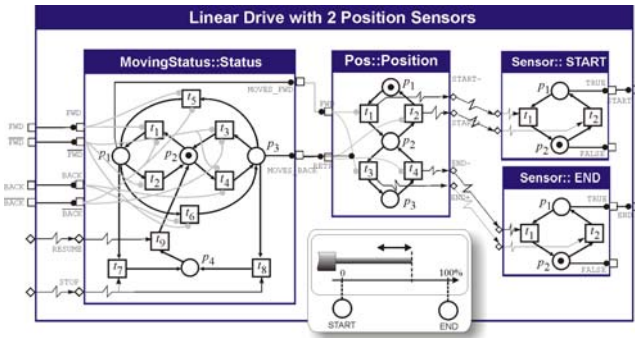
Second, specific software tools, called model-checkers, are used to prove certain properties with respect to the model.

Third, some analysis may be needed to understand why some undesirable behaviour occurs if it is revealed by the model-checker.

## Our modelling language

There are plenty languages used for formal modelling and verification, each has some strong features for a particular application domain. We use the graphical language called Net Condition/Event Systems, or *NCES* for short, which combines the block diagram view and modularity with mathematical precision of place-transition nets. Despite its graphical user-friendly form, the language has well defined mathematically rigorous rules. NCES was invented a decade ago by German scientists Mathias Rausch and Hans-Michael Hanisch and has been evolving through the application to modelling and verification of different automation systems.

The following Figure illustrates modelling of a simple linear drive with two position sensors.

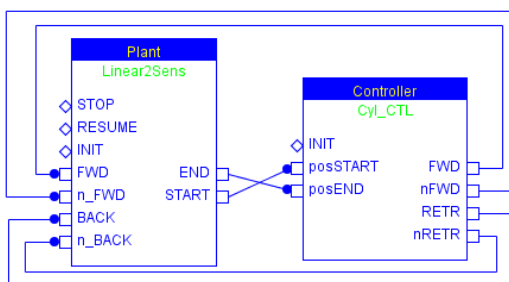


NCES's similarity with block diagrams used by control engineers makes it specifically suitable for modelling and analysis of automation systems. Its place-transition nature fits well for modelling of distributed systems.

Our sample model, for example, explicitly models sensors, reflects current position of the shaft and its reaction on the control signals and interactions from the environment.

### Closed-loop vs. Open-loop modelling

The NCES language is perfect for modelling closed-loop control systems as exemplified in the next Figure for the Cylinder case.



One module represents the model of the cylinder, which has exactly same signal interface as the real cylinder: logic signals FWD and BACK. In addition, the model can react on external events physically blocking the cylinder's movement (STOP) or allowing it again (RESUME).

The closed-loop modelling has a few benefits if compared to the open-loop verification of only controller, for example:

- It *reduces the complexity* by binding inputs to the controller to only some feasible reactions
- Freedom from deadlocks *cannot be checked* only for controller, sometimes even a deadlock-free controller can bring system to a deadlock state
- Specifications of desired or prohibited behaviour can be formulated *in terms of the object*, instead of referring to only input and output variables of the controller

### Problems and Limitations

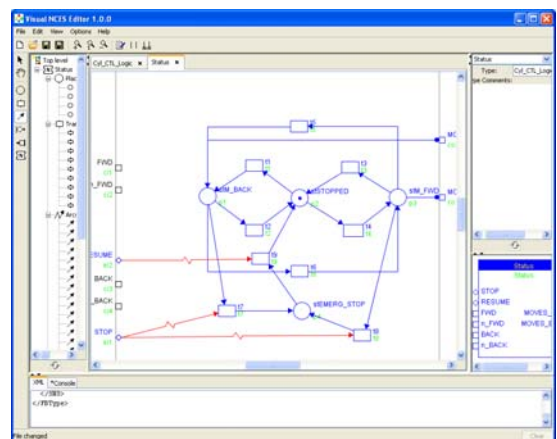
The need for modelling of plants is one of the most essential problems limiting the use of formal verification in industrial automation. While automatic model-generation is possible for the controller part, the plant modelling part requires some engineering methodology supporting systematic and structured model development.

**On the other hand, once developed, the model of plant can be re-used for verifying various controllers.**

Other problems include high computational complexity of the model-checking process. Nevertheless, the model-checking is widely used in digital design for exhaustive verification of electronic designs.

### The tool chain and the design flow

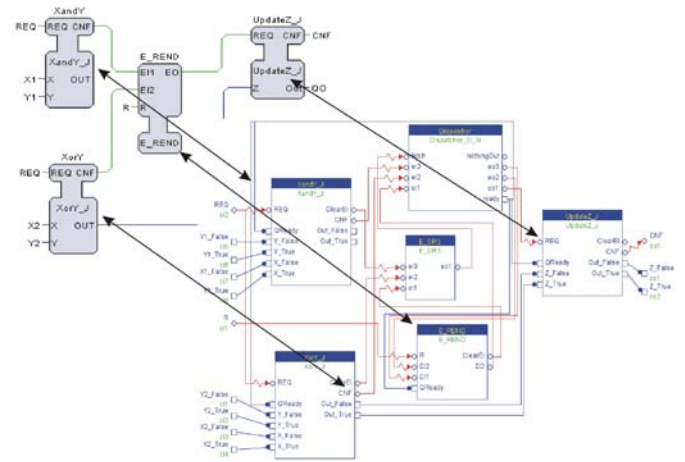
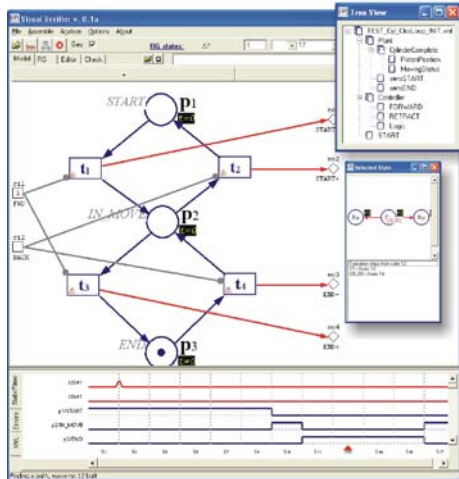
The tool chain supports the process of model development and analysis. It is specifically focused on the re-use of models and simplicity of their maintenance.



The models are created in the **graphical editor ViEd** and stored in XML – based format. Each module is designed separately, and consist of interface (event and condition inputs and outputs), and of internal logic defined by means of Condition/Event Net. Once designed, a model type can be instantiated over and over again in different designs.

A model can be basic (i.e. a single module as the one opened in the ViEd window above) or composite (i.e. a network of modules as the closed-loop model on the left).

The next in the tool-chain is the **Visual Verifier** tool (ViVe) which assembles the model of the whole system starting from the top level closed – loop view down to the models of every single element.



We have developed a model-generator which converts IEC 61499 function blocks to NCES modules, so the function block applications are modelled as networks of NCES modules, as shown in the Figure above. Their behaviour can be studied using ViVe.

**How we can help?**

The methodology of formal verification of automation systems has been in the focus of our research for the last decade. The methodology and supporting tools were applied in several application domains, such as assembly systems in electronics manufacturing, railway systems, batch processing. The toolset has been tested at the University of Halle (Germany), Tampere University of Technology (Finland) and Technical University of Vienna (Austria) in collaborative projects with industry.

We will be happy to help you benefit from using the Visual Verification framework for verifying automation and embedded systems. We can provide general training and consulting for particular applications.

**For further details please contact**

Dr. Valeriy Vyatkin  
**infoMechatronics and IndusTRIAL Automation Lab (MITRA)**  
 Department of Electrical and Computer Engineering  
**The University of Auckland**  
 Private bag 92019  
 Auckland Mail Centre  
 Auckland 1142, **New Zealand**  
 E-mail: [v.vyatkin@auckland.ac.nz](mailto:v.vyatkin@auckland.ac.nz)  
<http://www.ece.auckland.ac.nz/~vyatkin/>  
 Tel. +64-9-3737599 ext.89437 Fax: +64-9-3737461

**Formal verification and IEC 61499**

Distributed automation systems are especially hard to validate and here the formal verification can be the most helpful. The splendid similarities between IEC 61499 designs and NCES make the use of ViVe a natural complement to other testing tools for IEC 61499.