TNO innovation for life

ESI
Powered by industry, academia and TNO

State-of-the-art study on formalisms and methods to specify and analyze flows in logistic processes

TNO Public › TNO 2023 R12176
20 December 2023

ICT, Strategy & Policy
www.tno.nl
+31 88 866 50 00
info@tno.nl

TNO 2023 R12176 – 20 December 2023

State-of-the-art study on formalisms and methods to specify and analyze flows in logistic processes

Author(s)	Jos Hegge Dennis Hendriks Pierre van de Laar Bram van der Sanden Wytse Oortwijn Nan Yang
Classification report	TNO Public
Title	TNO Public
Report text	TNO Public
Appendices	TNO Public
Number of pages	11 (excl. front and back cover)
Number of appendices	1
Sponsor	TNO, ASML, VDL-ETG
Project name	Poka Yoke 2023
Project number	060.55520/01.01

Our partners

Acknowledgements
This research is carried out as part of the Poka Yoka program under the responsibility of TNO-ESI in cooperation with ASML and VDL-ETG. The research activities are supported by the Netherlands Organization for Applied Scientific Research TNO, the Netherlands Ministry of Economic Affairs and Climate Policy, and TKI-HTSM.

Contents

1 Introduction

This report describes the results of the state-of-the-art (SotA) study of the Poka Yoke project on formalisms and methods to specify and analyze flow. The SotA study was conducted at the start of the Poka Yoke project from January till May 2023. Details on how the conclusions have been derived can be found in the slide set, attached to this document as appendix.

Poka Yoke project (2023-2026) - The focus of the Poka Yoke project is to create a methodology to formally specify, synthesize and analyze supervisory controllers for logistical processes in cyber-physical production systems. As a case study, we consider the wafer logistics in the wafer handler subsystem of ASML lithography machines. The specification describes the environment, components, and requirements. Having a formal and complete specification and describing change specifications that contain pre- and postconditions makes it possible to synthesize flow specifications automatically. Various kinds of design assistance can be used to (automatically) analyze the specification, including simulation, verification, and synthesis. Results of this analysis provide feedback to further improve and/or refine the specification. From the formal specifications also artifacts can be generated, like documentation and control software.

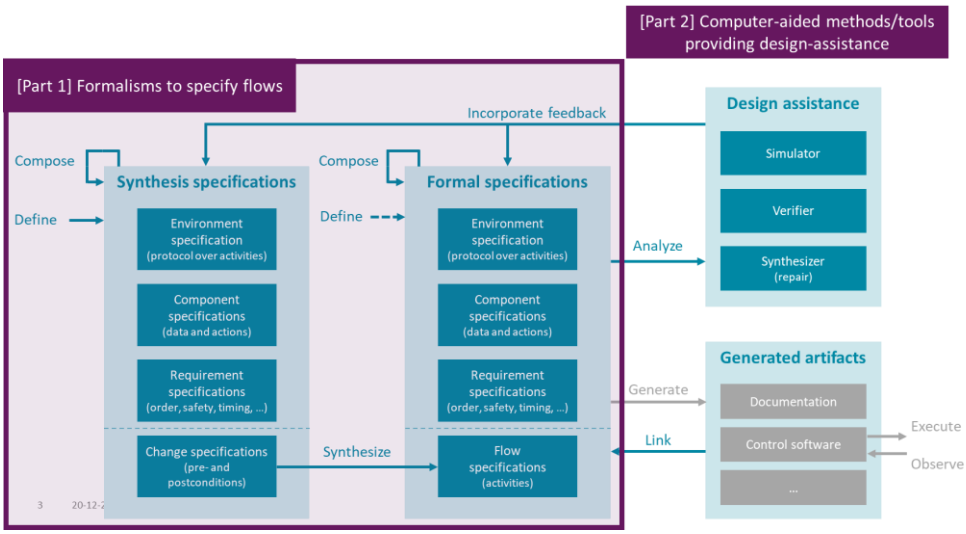


Figure 1.1: Overview of the Poka Yoke project focus.

Scoping of SotA study - In this report, we describe the SotA study results on the two parts shown in Figure 1.1.

- Part 1: Formalisms to specify flows.
- Part 2: Computer-aided methods/tools providing design-assistance.

Approach - We created the SotA study overview based on the team's expertise, by trying out various formalisms and tools, by consulting relevant experts both within TNO and in academia, and based on a literature survey, including documentation available online.

Scoping – The SotA study was restricted to Poka Yoke project context, with a specific focus on logistical processes and supervisory controllers. As target users for the formalisms and analysis techniques, we considered software and mechatronics engineers. In the SotA study, we looked primarily at the current status/situation of the formalisms, methods, and tools, and not at roadmaps for planned additions.

2 Formalisms to specify flows

In the first part of the SotA study, we identified the most suitable formalisms to specify flows as encountered in logistical processes. We first selected the most promising **formalism category** based on a set of aspects, and subsequently we identified the most promising **formalisms within the selected category** based on a set of aspects.

2.1 Formalism categories to specify flows

We investigated five promising formalism categories, shown in Table 2.1.

Table 2.1: Formalism categories

Formalism category	**Examples**
Mathematical logics	Alloy, first-order logic, Prolog, SMT-lib, TLA+, Z
Process algebras	E-LOTOS, mCRL2, POOSL
Petri Nets	PNML, TAPAAL, TINA
Synchronizing state machines	ASD/Dezyne, CIF, Coco, ComMA, MATLAB Stateflow, Modelica, UPPAAL
Flowcharts	UML/SysML/SysML v2/LSAT activity diagrams, BPMN, MATLAB

We evaluated these categories based on four aspects, using a score from -- (worst) to ++ (best), shown in Table 2.2.

Table 2.2: Aspects to evaluate the formalism categories

Aspect	Explanation	Range
Expressivity	Are all relevant capabilities supported?	-- to ++
Intuitiveness	Are the behavior effects of the specification constructs locally explicitly clear? (e.g., no impact on global behavior in non-intuitive ways)	-- to ++
Familiarity	How familiar are software and mechatronics engineers with the formalism category?	-- to ++
Adoptability	How much effort would it be to train people and integrate the tools related to the formalism category?	-- to ++

To assess expressivity of the formalism categories, we used the following list of relevant capabilities:

- Actions
- Sequencing
- Repetition (i.e., iteration)
- Choice
- Hierarchy
- Parallelism and synchronization
- Data
- Configuration (i.e., to represent system variants)
- Time
- Stochastics
- Requirements
- Design decision

Conclusions – As shown in Table 2.3, we concluded that the category of flowcharts is the best option to specify product flow. Alternative categories are, however, not completely discarded, as various aspects of different formalisms can be combined.

Table 2.3: Evaluation of flowchart categories

Formalism category	Expressivity	Intuitiveness	Familiarity	Adoptability
Mathematical logics	++	--	--	--
Process algebras	+	-	-	--
Petri Nets	+	+	+	+
Synchronizing state machines	+	+	-/+	+
Flowcharts	+	++	++	++

2.2 Formalisms to specify flowcharts

We investigated five promising formalisms within the flowcharts category:
1. UML/SysML activity diagrams
2. SysML v2 activity diagrams
3. BPMN (Business Process Model and Notation)
4. LSAT activity diagrams
5. MATLAB

We evaluated the formalisms based on the eight key aspects shown in Table 2.4.

Table 2.4: Aspects to evaluate flowchart formalisms

Aspect		Explanation	Range
Suitability	Expressivity	Are the required capabilities natively supported?	-- to ++
	Extensibility	Can the missing capabilities be easily added?	-- to ++
	Formal semantics	Can the formalism be used for automated design assistance?	-- to ++
	Representability	Is there a textual and graphical syntax?	-- to ++

Aspect		Explanation	Range
Maturity	Industrial acceptance	Is the formalism currently widely used in industry?	-- to ++
	Tool ecosystem	Is the formalism well supported by (many) well-maintained tools?	-- to ++
Adoptability		How much effort would it be to train people and integrate the tools?	-- to ++
Openness		Is there an open standard, without vendor lock-in?	-- to ++

Conclusions – As shown in Table 2.5, there is not a single formalism that perfectly fits our needs. We conclude that we can combine various aspects of different formalisms, e.g., properties and timing. We propose to connect to SysML v2, which is most suitable on the long term, and to start from UML/SysML, which is most mature and adoptable on the short term.

Table 2.5: Evaluation of flowchart formalisms

Aspect	Formalism	UML / SysML activity diagrams	SysML v2 activity diagrams	BPMN	LSAT activity diagrams	MATLAB
Suitability	Expressivity	+	+	+	-	-/+
	Extensibility	+/++	++	-/+	-/+	-
	Formal semantics	-	++	++	++	+
	Representability	-	++	-	++	-
Maturity	Industrial acceptance	++	--	-	-	++
	Tool ecosystem	++	--	+	-	++
Adoptability		++	+	+	++	+
Openness		++	++	++	++	--

3 Computer-aided methods and tools for synthesis and analysis

In the second part of the SotA study, we focused on computer-aided methods and tools for synthesis and analysis. We distributed a questionnaire to the Poka Yoke stakeholders to identify the most important design-assistance questions. The importance was determined by ranking the questions based on relevance, frequency, and total effort. Based on the ranking, we identified the following top 5 questions:

1. **Behavior** – Is the behavior realized by the specification as expected?
2. **Safety** - Does the specification guarantee a particular safety property, i.e., bad things never happen, such as deadlock and collisions?
3. **Reachability** - Are the expected parts of the specification reachable?
4. **Realizability** - Can the specification be realized, i.e., are the requirements non-conflicting?
5. **Timing guarantees** - Does the specification guarantee a particular timing property?

For each question, we identified suitable methods and tools to provide design-assistance, as shown in Table 3.1.

Table 3.1: Suitable methods and tools to provide design-assistance for the top 5 questions

Question	Method(s)	Example tools
1. Behavior	**System simulation**	
	Simulation	Eclipse ESCET / CIF, MATLAB Simulink, POOSL
	Visualization engine	Blender, Cry, Panda3D, Unity, Unreal Engine
	Activity diagram simulation	LSAT, fUML execution engine, Cameo, Papyrus Moka
2. Safety	**Transformations to aspect models** Examples: UML activity to Petri Net, CSP, Pi-calculus, PROMELA, NuSMV, Automata	
	Verification on aspect models	
	Checking properties using model checker	mCRL2, TAPAAL, TINA, SPIN, ITS-tools, NuSMV
	Check satisfiability using SMT solver	Z3, cvc5
	Check non-blockingness using supervisory controller synthesis	Eclipse ESCET / CIF, Supremica

Question	Method(s)	Example tools
3. Reachability	Model checking	mCRL2, TAPAAL, TINA, SPIN, ITS-tools, NuSMV
4. Realizability	Automata-based approach	
	Ontology-based approach	
5. Timing guarantees	Transformations to aspect models Examples: UML/SysML activity to PROMELA, SPIN, NuSMV, UPPAAL, PES, UPPAAL-SMC, PRISM	
	Verification on aspect models Check properties using model checker	UPPAAL, TAPAAL, TINA, SPIN, PRISM

Conclusions – In our SotA study, we concluded that depending on the required design assistance, different aspect models might be needed. Such aspect models can be automatically generated from a formal specification. Also, the outcome of the analysis can be transformed back in terms of the specification. Figure 3.1 graphically shows this link between the specification and aspect models.

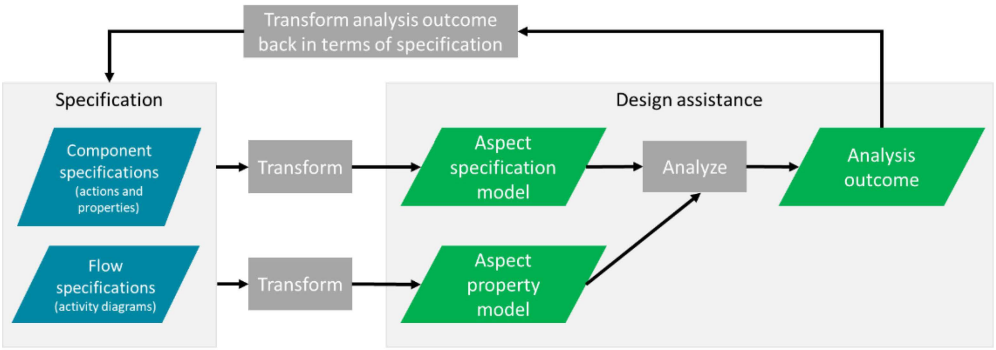


Figure 3.1: Aspect models can be used to offer specific design assistance given the same formal specification.

We conclude that multiple suitable methods/tools exist for each question, giving confidence that the questions can likely be answered using available methods and tools.

Appendix A
Slide deck

Attached with this report is a slide deck containing additional details on the formalisms and analysis methods/tools, as well as how the conclusions have been derived.

ICT, Strategy & Policy

High Tech Campus 25
5656 AE Eindhoven
www.tno.nl

TNO innovation for life

Poka Yoke applied research project

Appendix A of TNO report TNO 2023 R12176

State-of-the-art study on formalisms and methods to specify and analyze flows in logistic processes

VDL ASML ESI

Powered by industry, academia and TNO

State-of-the-art study: Approach

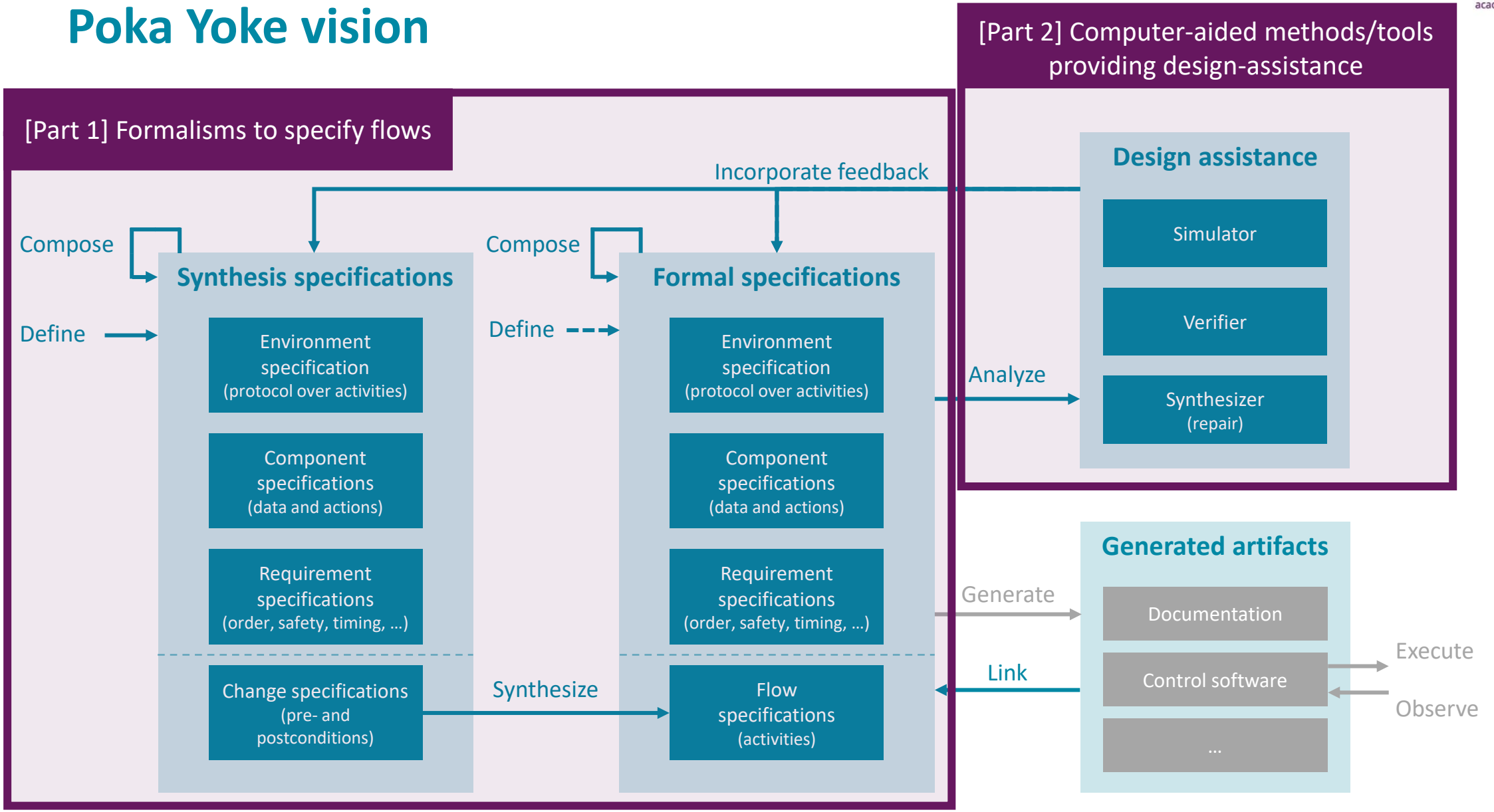
Sources

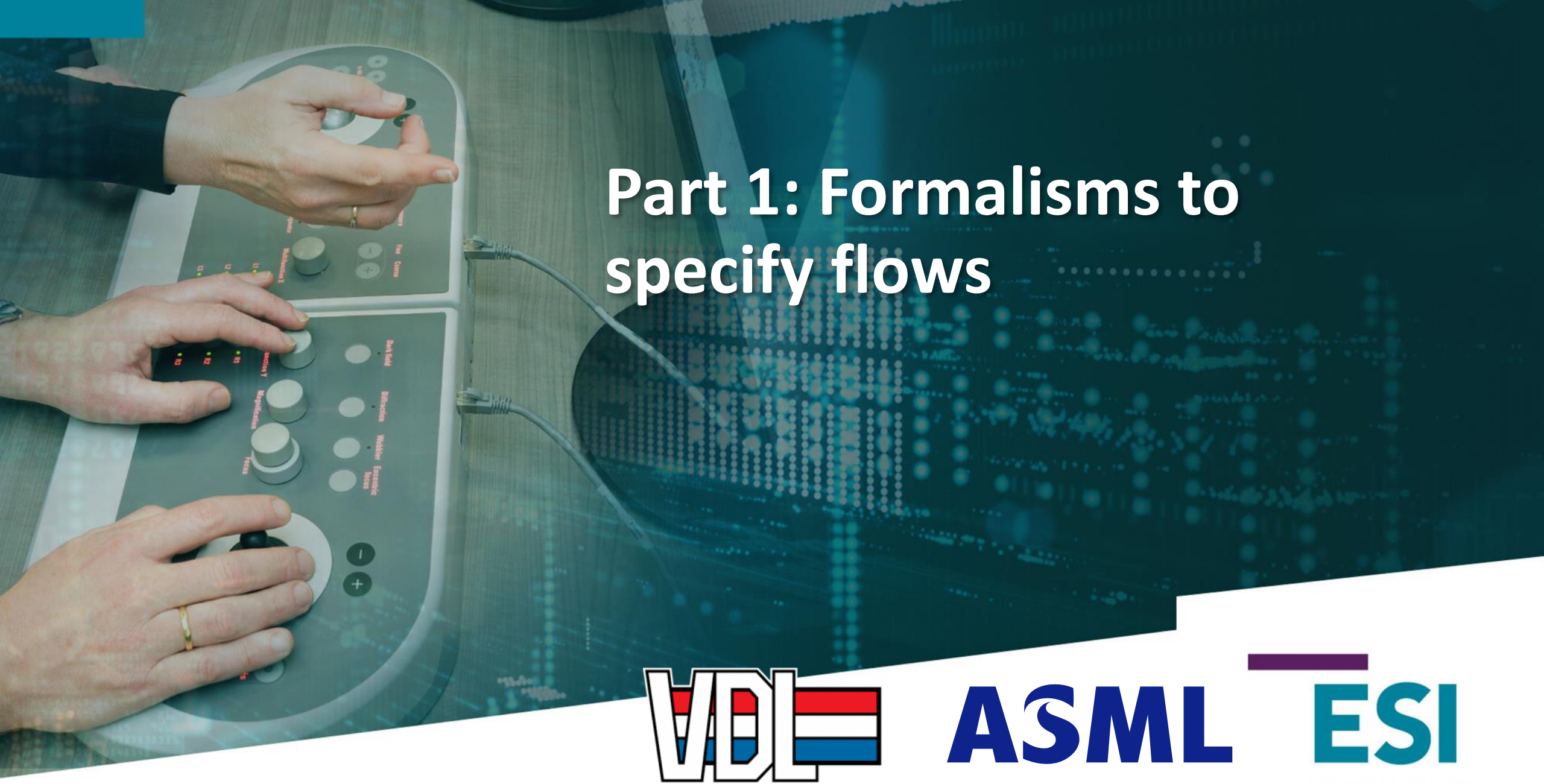
- Used various sources: own expertise, literature/internet search, experts

Scoping

- Restricted to Poka Yoke context: wafer handling/logistics systems, supervisory controllers, software …
- Looked primarily at current status/situation, of the formalisms, methods and tools
- Target users: software (SW) and mechatronics (MSD) engineers

Poka Yoke vision





Part 1: Formalisms to specify flows

VDL ASML ESI

Powered by industry, academia and TNO

Categories of formalisms for specifying flow

We investigated 5 promising categories of formalisms based on 4 key aspects

1. Mathematical logics — e.g., Alloy, first-order logic, Prolog, SMT-lib, TLA+, Z
2. Process algebras — e.g., E-LOTOS, mCRL2, POOSL
3. Petri Nets — e.g., PNML, TAPAAL, TINA
4. Synchronizing state machines — e.g., ASD/Dezyne, CIF, Coco, ComMA, MATLAB, Modelica, UPPAAL
5. Flowcharts — e.g., UML/SysML/SysML v2/LSAT activity diagrams, BPMN, MATLAB

Aspect	Explanation	Range
Expressivity	Are all relevant capabilities supported?	-- to ++
Intuitiveness	Are the behavior effects of the specification constructs locally explicitly clear?	-- to ++
Familiarity	How familiar are SW and MSD engineers with the formalism category?	-- to ++
Adoptability	How much effort would it be to train people and integrate the tools?	-- to ++

Formalism categories: 1. Mathematical Logics

```
fact trans {
  always (empty or (some f : File | delete[f] or restore[f]))
}
```

Alloy

(https://haslab.github.io/formal-software-design/overview/index.html)

```
(=> x (=> y z))
(and (and x y) z)
(and (= x y) (= y z))
(and (distinct x y) (distinct x z) (distinct y z)) .
```

SMT-lib

(https://haslab.github.io/formal-software-design/overview/index.html)

Aspect	Findings	Conclusion
Expressivity	All relevant capabilities can be represented mathematically.	++
Intuitiveness	• Wafer flow is encoded, rather than expressed naturally. • Local changes to the formula may indirectly impact behavior globally.	--
Familiarity	SW and MSD engineers may be unfamiliar with mathematical logics.	--
Adoptability	Translation to mathematical logics is non-trivial, extensive training is essential.	--

Formalism categories: 2. Process algebras

```
process duplex-buffer [in-a, in-b, out-a, out-b] :=
        simplex-buffer [in-a, out-a]
    |||  simplex-buffer [in-b, out-b]

where
        process simplex-buffer [in, out] :=
                in; out; stop
        endproc
endproc
```

LOTOS

(Introduction to the ISO Specification Language LOTOS, T. Bolognesi and E. Brinksma, 1987, page 15)

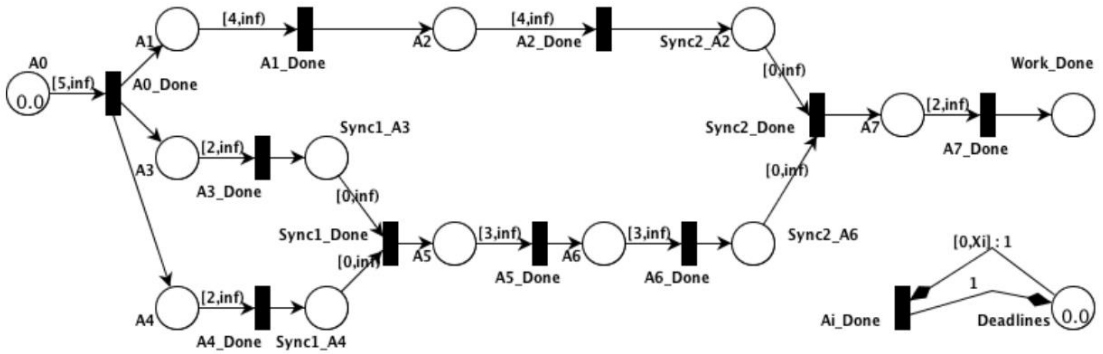
```
act coin, good, bad;
proc P = coin .
           (bad . P +
            coin . good . P);
init P;
```

mCRL2

(https://www.mcrl2.org/web/user_manual/articles/basic_modelling.html)

Aspect	Findings	Conclusion
Expressivity	Many different variants (e.g., probabilistic/timed process algebras, …).	+
Intuitiveness	Some local constructs can impact global behavior in a non-obvious way.	-
Familiarity	The use of algebraic terms and operators is distant from current practice.	-
Adoptability	Translation to process algebra is non-trivial, extensive training may be needed.	--

Formalism categories: 3. Petri Nets



TAPAAL

(TAPAAL: Editor, Simulator and Verifier of Timed-Arc Petri Nets, Byg et al., ATVA 2009)

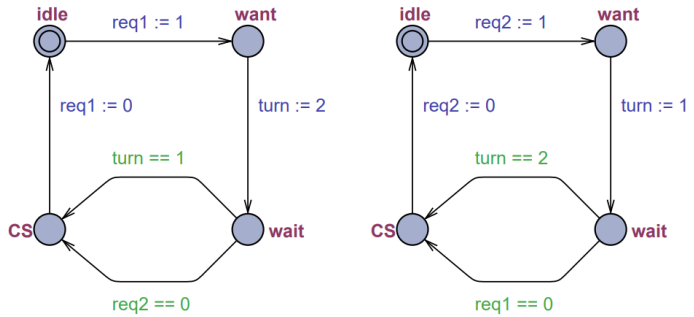
Aspect	Findings	Conclusion
Expressivity	Many different variants (e.g., colored/high-level/probabilistic Petri Nets, …).	+
Intuitiveness	The concept of moving tokens is more abstract than concrete activities.	+
Familiarity	The concepts are not so distant from activity flows.	+
Adoptability	Translation may be laborious but various transformations are available. Training may be needed.	+

Formalism categories: 4. Synchronizing state machines

```
automaton lamp:
  event turn_on, turn_off;

  location on:
    initial;
    edge turn_off goto off;

  location off:
    edge turn_on goto on;
end
```

CIF

(https://eclipse.org/escet/cif/language-tutorial/basics/automata.html)

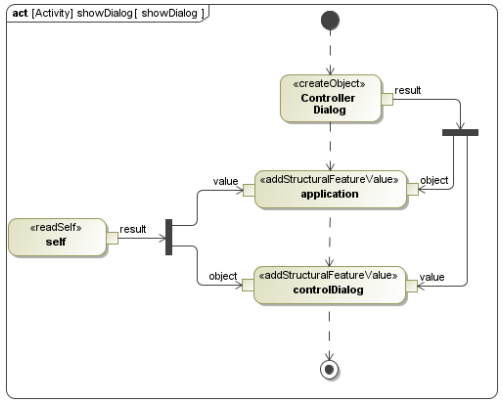


UPPAAL

(https://www.it.uu.se/research/group/darts/uppaal/small_tutorial.pdf)

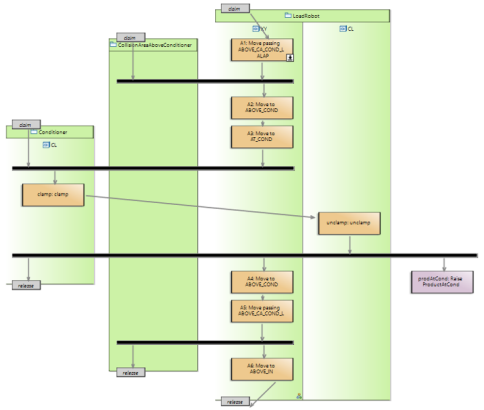
Aspect	Findings	Conclusion
Expressivity	Many different variants (e.g., EFAs, timed automata, …)	+
Intuitiveness	Some local constructs can impact global behavior in a non-obvious way.	+
Familiarity	• Activity diagrams and state machines use inverted “blocks and arrows”. • Already used by SW, but less by MSD.	-/+
Adoptability	Translation may be feasible; training may be required.	+

Formalism categories: 5. Flowcharts



UML

(Source: https://docs.nomagic.com/display/SYSMLP190/SysML+Activity+Diagram)



LSAT

(Twilight demo model, LSAT User Guide)

Aspect	Findings	Conclusion
Expressivity	Various formalisms and extensions exist that support the required capabilities.	+
Intuitiveness	The behavior effects of most specification constructs are locally clear.	++
Familiarity	Already known to both SW and MSD engineers.	++
Adoptability	Translation is relatively easy, relatively little training would be required.	++

Categories of formalisms for specifying flow: Conclusion

Formalism	Expressivity	Intuitiveness	Familiarity	Adoptability
Mathematical logics	++	--	--	--
Process algebras	+	-	-	--
Petri Nets	+	+	+	+
Synchronizing state machines	+	+	-/+	+
Flowcharts	+	++	++	++

Conclusions

- Flowcharts are the best option for specifying wafer flow
- Alternatives are not completely discarded; different aspects of different formalisms can be combined

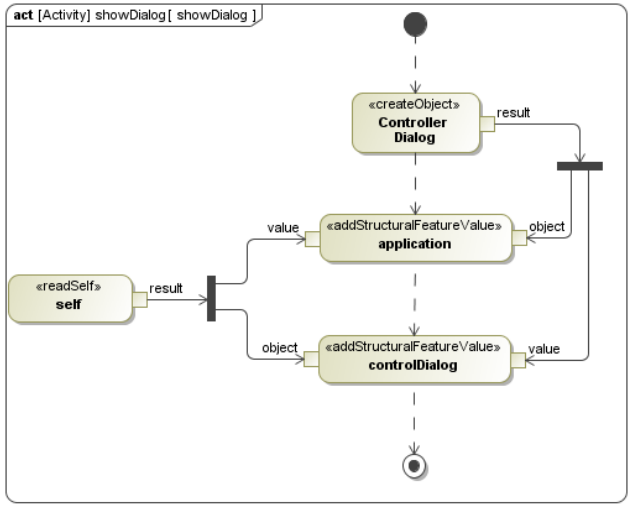
Formalisms for specifying flowcharts

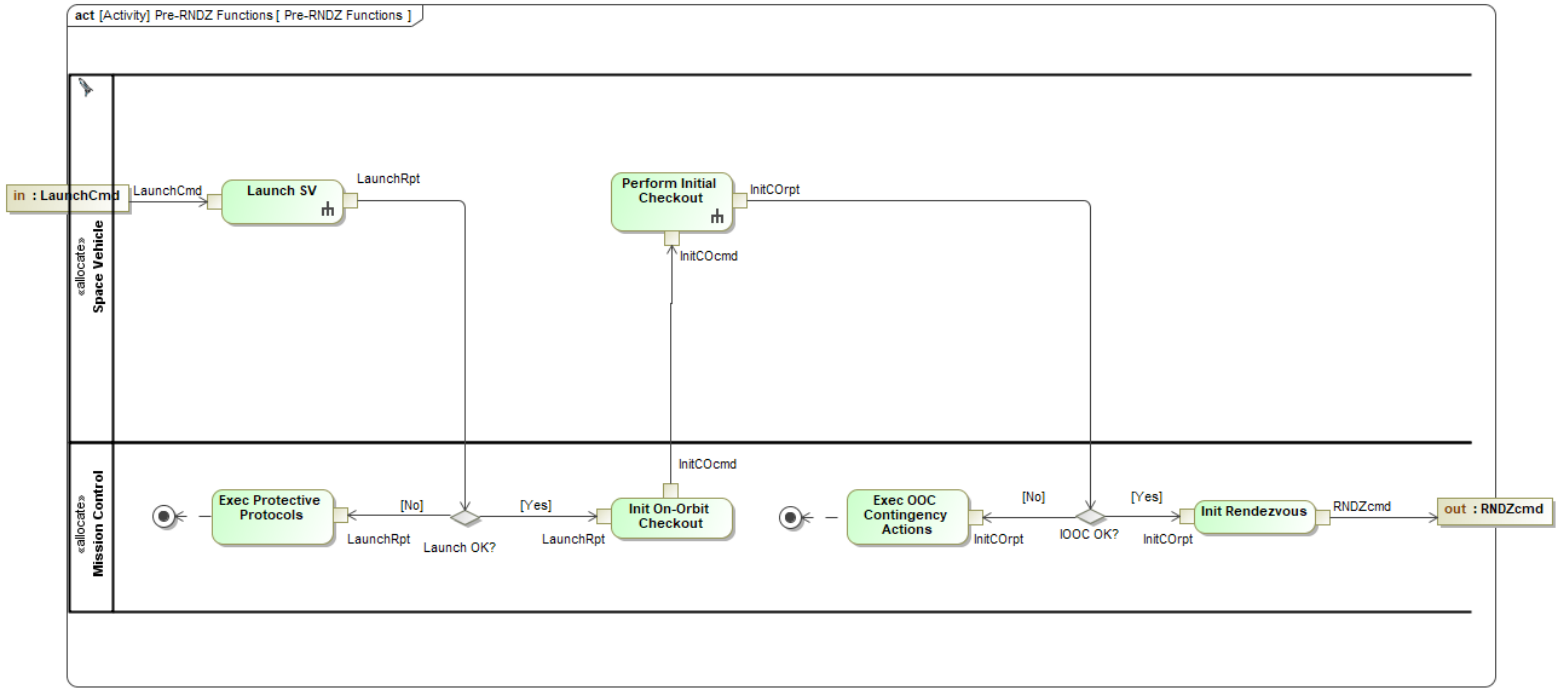
We investigated 5 promising formalisms for specifying flowcharts based on 8 key aspects

1. UML/SysML activity diagrams
2. SysML v2 activity diagrams
3. BPMN (Business Process Model and Notation)
4. LSAT activity diagrams
5. MATLAB

Aspect		Explanation	Range
Suitability	**Expressivity**	Are the required capabilities natively supported?	-- to ++
	Extensibility	Can the missing capabilities be easily added?	-- to ++
	Formal semantics	Can the formalism be used for automated design assistance?	-- to ++
	Representability	Is there a textual and graphical syntax?	-- to ++
Maturity	**Industrial acceptance**	Is the formalism currently widely used in industry?	-- to ++
	Tool ecosystem	Is the formalism well supported by (many) well-maintained tools?	-- to ++
	Adoptability	How much effort would it be to train people and integrate the tools?	-- to ++
	Openness	Is there an open standard, without vendor lock-in?	-- to ++

Flowchart formalisms: 1. UML/SysML activity diagrams





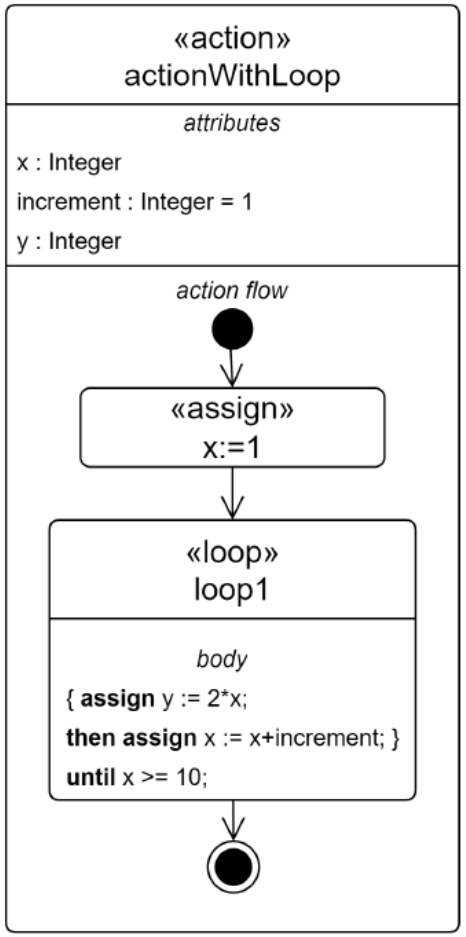
(Source: https://docs.nomagic.com/display/SYSMLP190/SysML+Activity+Diagram, accessed 2023-11-08)

Source: https://pivotpt.com/training/mbse-sysml, accessed 2023-04-26 (modified).

Flowchart formalisms: 1. UML/SysML activity diagrams

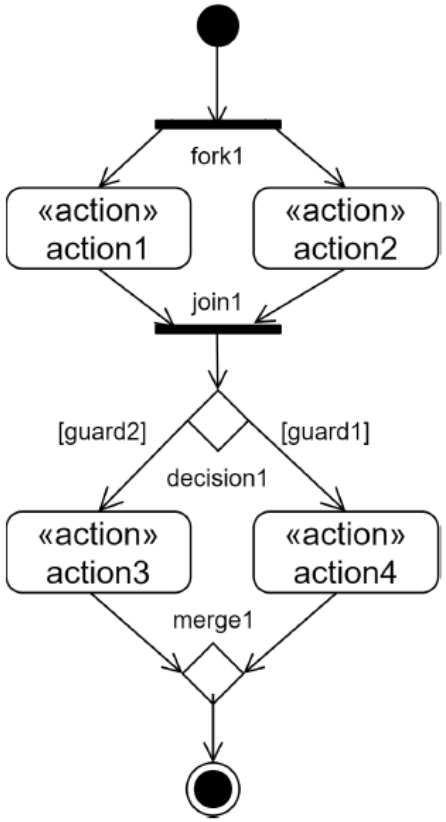
Aspect		Findings	Conclusion
Suitability	**Expressivity**	Many required capabilities can be natively expressed.	+
	Extensibility	Via stereotypes, extension profiles, and annotated properties.	+/++
	Formal semantics	Generally lacking; some variants have a (somewhat) formal semantics.	-
	Representability	The formalism is graphical, lacking textual syntax.	-
Maturity	**Industrial acceptance**	Widely used industry standards.	++
	Tool ecosystem	Many commercial and open-source tools are available.	++
	Adoptability	Already used; going from UML to SysML is a relatively minor step.	++
	Openness	Both are open standards.	++

Flowchart formalisms: 2. SysML v2 activity diagrams




```
package Loop {
   action
actionWithLoop {
      attribute
x:Integer := 1;
      attribute
increment:Integer = 1;
      attribute
y:Integer;
      loop action loop1
{
         assign y :=
2*x;
         then assign x
:= x+increment;
      } until x >= 10;
      then done;
   }
}
```

Source: OMG Systems Modeling Language, page 196.



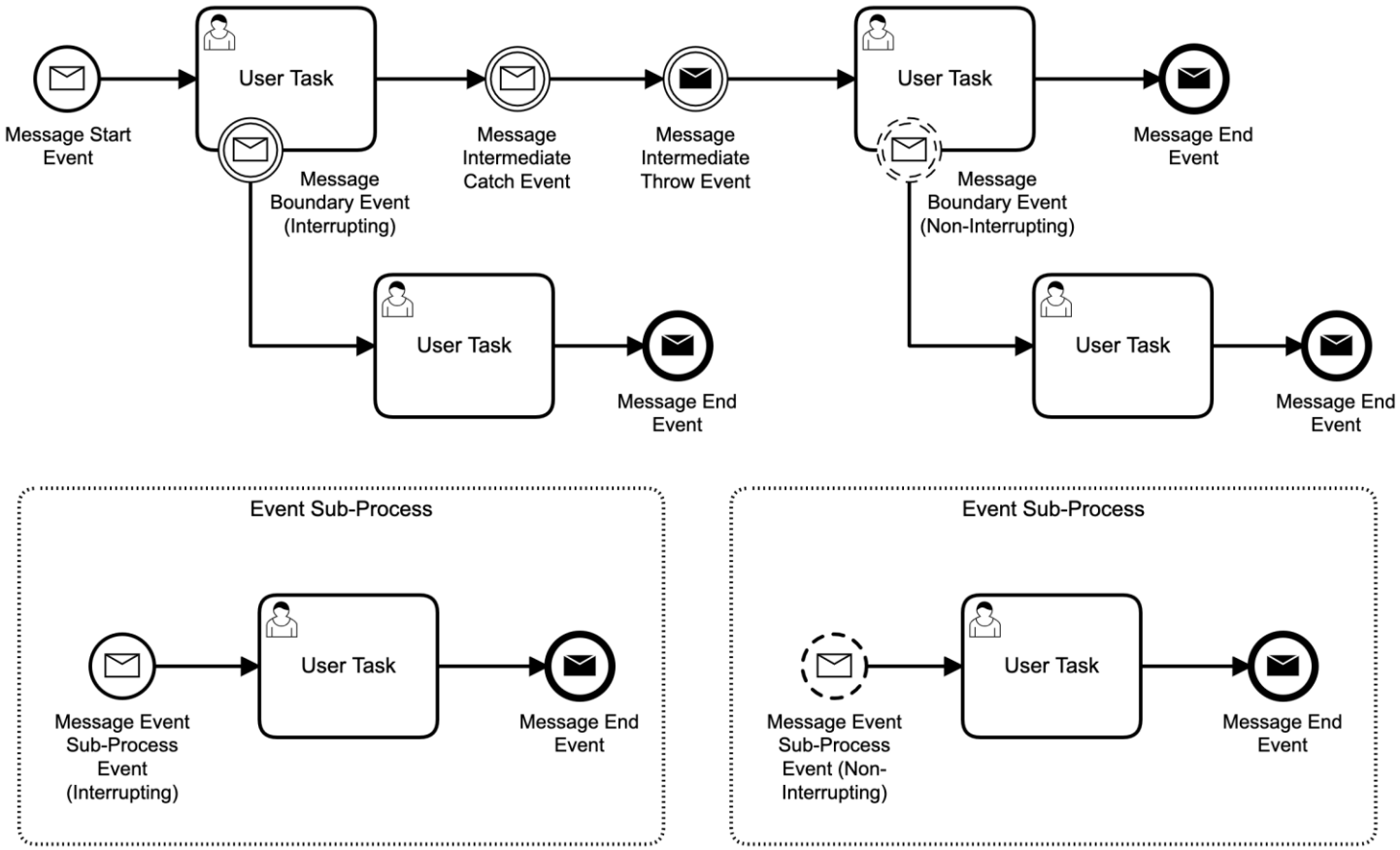

```
first start;
then fork fork1;
   then action1;
   then action2;

action action1;
   then join1;
action action2;
   then join1;

join join1;
then decide decision1;
   if guard2 then
action3;
   if guard1 then
action4;

action action3;
   then merge1;
action action4;
   then merge1;

merge merge1;
then done;
```

Source: OMG Systems Modeling Language, page 198.

Flowchart formalisms: 2. SysML v2 activity diagrams

Aspect		Findings	Conclusion
Suitability	**Expressivity**	Many required capabilities can be natively expressed.	+
	Extensibility	Has extensive support for extensions.	++
	Formal semantics	Has a well-defined formal semantics.	++
	Representability	Has both a graphical and textual syntax.	++
Maturity	**Industrial acceptance**	Currently beta specification; expected to become an industry standard.	--
	Tool ecosystem	Few tools available; expected to grow significantly.	--
	Adoptability	Translation is straightforward; training is required.	+
	Openness	Designed as open standard; interoperability is a key focus.	++

Flowchart formalisms: 3. BPMN

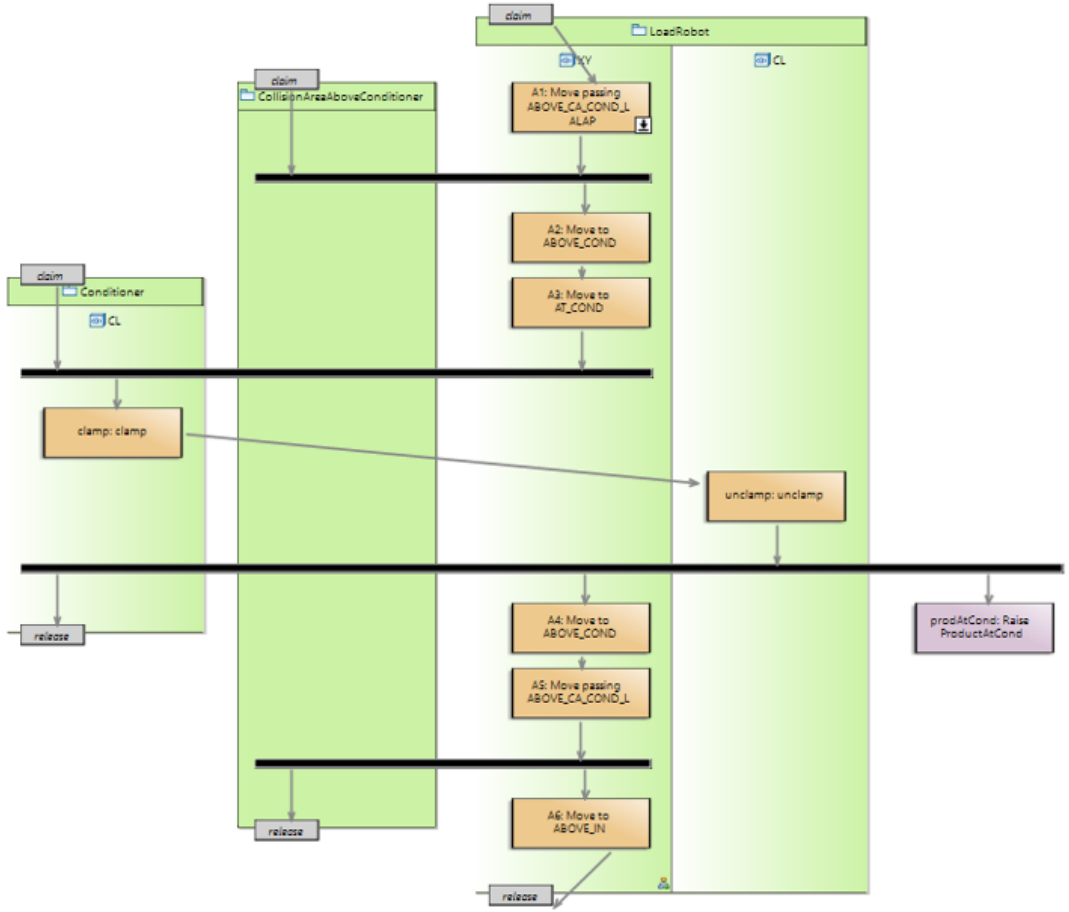


Source: https://www.javanibble.com/bpmn-message-event/

Flowchart formalisms: 3. BPMN

Aspect		Findings	Conclusion
Suitability	**Expressivity**	Expressivity is very close to UML/SysML activity diagrams.	+
	Extensibility	Can be extended, but not easily for all capabilities.	-/+
	Formal semantics	Has a well-defined formal semantics.	++
	Representability	The formalism is graphical, lacking textual syntax.	-
Maturity	**Industrial acceptance**	Well-accepted for business domain, less for software domain.	-
	Tool ecosystem	Commercial and open-source tools available (less than UML/SysML).	+
	Adoptability	Close to activity diagrams, little training required.	+
	Openness	Is an (OMG) open standard.	++

Flowchart formalisms: 4. LSAT activity diagrams

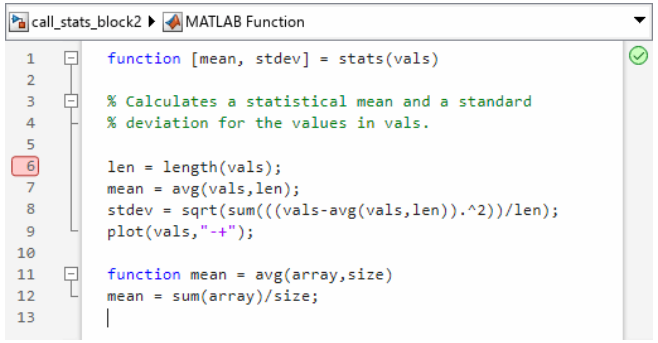


Source: LSAT v0.2 User Guide, Section 2.4.1

Flowchart formalisms: 4. LSAT activity diagrams

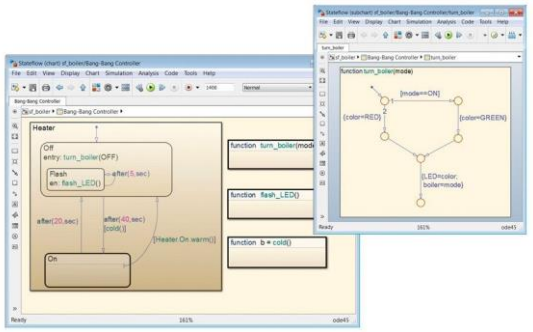
Aspect		Findings	Conclusion
Suitability	**Expressivity**	Multiple capabilities cannot be expressed at all.	-
	Extensibility	Can be extended, for which in-house knowledge is available.	-/+
	Formal semantics	Well-defined formal semantics to enable timing analysis.	++
	Representability	Has both a graphical and textual syntax.	++
Maturity	**Industrial acceptance**	Few companies use it.	-
	Tool ecosystem	No commercial support; small developer community.	-
	Adoptability	Already used by SW and MSD.	++
	Openness	Is open source.	++

Flowchart formalisms: 5. MATLAB



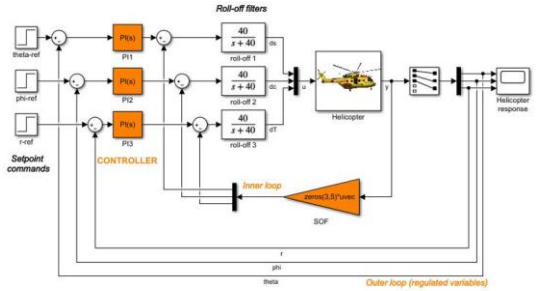
MATLAB

(https://www.mathworks.com/help/simulink/ug/debugging-a-matlab-function-block.html)



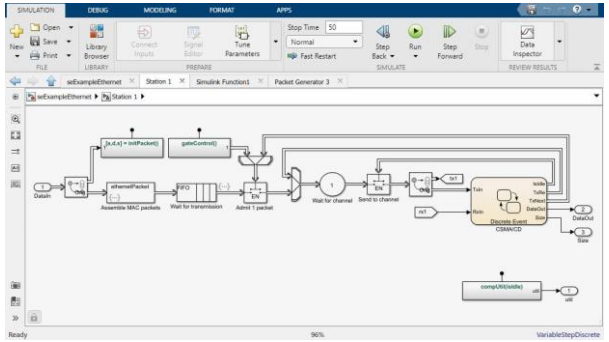
Stateflow

(https://www.mathworks.com/products/stateflow.html)



Simulink

(https://nl.mathworks.com/products/simulink.html)



SimEvents

(https://nl.mathworks.com/products/simevents.html)

Flowchart formalisms: 5. MATLAB

Aspect		Findings	Conclusion
Suitability	**Expressivity**	Some capabilities cannot be expressed at all, but most can.	-/+
	Extensibility	Some extensibility, but limited in adding new concepts.	-
	Formal semantics	Has a well-defined execution semantics.	+
	Representability	Some features are only supported in a graphical editor.	-
Maturity	**Industrial acceptance**	Widely used in industry.	++
	Tool ecosystem	Commercially supported.	++
	Adoptability	Training is required for people unfamiliar with MATLAB.	+
	Openness	There is vendor lock-in.	--

Formalisms for specifying flowcharts: Conclusion

Formalism	Suitability				Maturity		Adopt ability	Openness
	Expressivity	Extensibility	Formal semantics	Represent ability	Industrial acceptance	Ecosystem		
UML / SysML activity diagrams	+	+/++	-	-	++	++	++	++
SysML v2 activity diagrams	+	++	++	++	--	--	+	++
BPMN	+	-/+	++	-	-	+	+	++
LSAT activity diagrams	-	-/+	++	++	-	-	++	++
MATLAB	-/+	-	+	-	++	++	+	--

Conclusions

- There is not a single formalism that perfectly fits our needs
- Can combine different aspects of different formalisms, e.g., properties and timing, to be investigated further
- We propose to connect to SysML v2, which is most suitable on the long term
- We propose to start from UML/SysML, which is most mature and adoptable on the short term

There is no single perfect UML modeling tool for our purposes

	Required/desired capabilities						
	Formal semantics		Activity diagram simulation	Property analysis	Activity diagram synthesis	Extensible / open source	Currently used
	fUML	ALF					
UML Designer	×	×	-	×	×	++	✓
Enterprise Architect	×	×	+	×	×	+	✓
Eclipse Papyrus™, with Moka plugin	✓	✓	+	×	×	++	×
MagicDraw / Cameo Simulation Toolkit	✓	✓	++	×	×	+	×

No tool covers all required capabilities

Conclusion

We tried modeling existing wafer flow specifications in fUML/ALF

- The meaning of current WH modeling concepts do not directly map to native fUML concepts
- These concepts can however be encoded in fUML (but this requires a translation)

Conclusion

- There is no existing formalism nor tool that perfectly fits our need
- We start with a custom version of activities that translates to fUML
  - We do that by extending activities in UML Designer
- We keep an eye on SysML v2 and reconsider its suitability later in the project

Part 2: Computer-aided methods/tools providing design-assistance

Powered by industry, academia and TNO

State-of-the-art study: Process

Data collection

Distributed **design-assistance questionnaire** to focus the study, got 5 responses.

Data processing

- Computed how often each value is ticked for each question, given the responses; “x” = 1, else 0.
- Computed the total score of each question by summation of the following values;
  - Total relevance of question = 4 * Critical + 3 * Significant + 2 * Informative + 1 * None
  - Total frequency of question = 4 * Daily + 3 * Weekly + 2 * Monthly + 1 * Yearly or less
  - Total effort of question = 4 * Month or more + 3 * Week + 2 * Day + 1 * Hour

Investigation

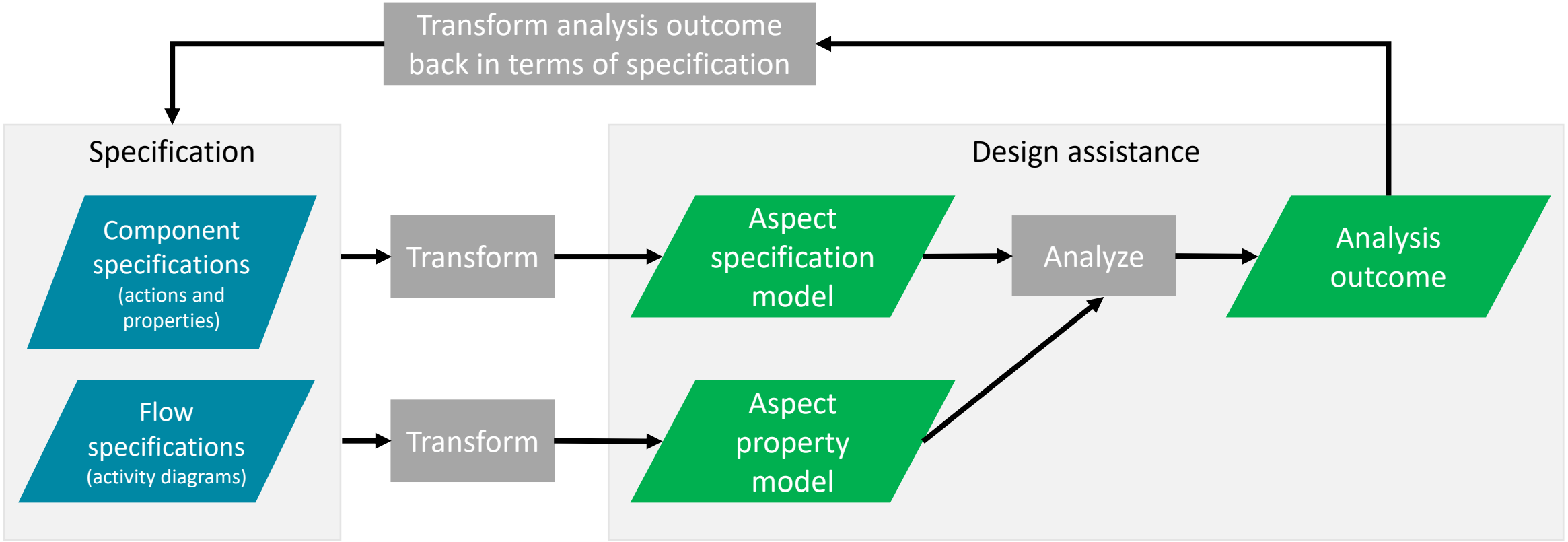
Selected the top 5 questions. For each question, we made an overview of available analysis methods.

Design assistance questionnaire: Results

Nr	Score	Question
1	50	How does the specification behave, i.e., is the exposed behavior as expected?
2	48	Does the specification guarantee a particular safety property, i.e., bad things never happen, such as deadlock and collisions?
3	46	Are the expected parts of the specification reachable?
4	45	Can the specification be realized, i.e., are the requirements non-conflicting?
5	44	Does the specification guarantee a particular timing property?
6	43	What is the optimal KPI value given the happy flow of the specification?
7	42	Which parts of the specification relate to / cause a particular (unexpected) behavior?
8	42	How many different solutions are possible given the specification?
9	42	Which part of the specification limits a particular KPI the most?
10	42	Does the specification satisfy its interface contracts?
11	41	Does the specification guarantee a particular liveness property, i.e., something good eventually happens?
12	40	What is the average KPI value given the stochastics of all possible behavior?
13	39	When the design space is empty, which parts of the specification are conflicting?
14	38	What is the design space, i.e., what are the possible design choices?
15	38	How (much) does a part of the specification limit the design space?
16	37	How does a sequence observed at a customer relate to the specification, such as states of components and actions in activity diagrams?
17	36	How do parts of the specification interact, i.e., restrict each other's behavior?
18	35	What is the worst case KPI value given all possible behavior?
19	34	Is a particular sequence (not) possible given the specification? And why (not)?
20	34	What are the trade-offs between multiple KPIs?
21	33	Does the specification guarantee a particular ordering property, such as FIFO (first in first out)?
22	31	Does the specification contain redundant information?
23	30	Does the specification adhere to certain checks and guidelines?

Design assistance: Different aspect models for different techniques/tools

Depending on the required design assistance, different aspect models might be needed.



Design assistance questionnaire: Top 5 questions

Nr	Score	Question
1	50	How does the specification behave, i.e., is the exposed behavior as expected?
2	48	Does the specification guarantee a particular safety property, i.e., bad things never happen, such as deadlock and collisions?
3	46	Are the expected parts of the specification reachable?
4	45	Can the specification be realized, i.e., are the requirements non-conflicting?
5	44	Does the specification guarantee a particular timing property?
6	43	What is the optimal KPI value given the happy flow of the specification?
7	42	Which parts of the specification relate to / cause a particular (unexpected) behavior?
8	42	How many different solutions are possible given the specification?
9	42	Which part of the specification limits a particular KPI the most?
10	42	Does the specification satisfy its interface contracts?
11	41	Does the specification guarantee a particular liveness property, i.e., something good eventually happens?
12	40	What is the average KPI value given the stochastics of all possible behavior?
13	39	When the design space is empty, which parts of the specification are conflicting?
14	38	What is the design space, i.e., what are the possible design choices?
15	38	How (much) does a part of the specification limit the design space?
16	37	How does a sequence observed at a customer relate to the specification, such as states of components and actions in activity diagrams?
17	36	How do parts of the specification interact, i.e., restrict each other's behavior?
18	35	What is the worst case KPI value given all possible behavior?
19	34	Is a particular sequence (not) possible given the specification? And why (not)?
20	34	What are the trade-offs between multiple KPIs?
21	33	Does the specification guarantee a particular ordering property, such as FIFO (first in first out)?
22	31	Does the specification contain redundant information?
23	30	Does the specification adhere to certain checks and guidelines?

Question 1

How does the specification behave, i.e., is the exposed behavior as expected?

Simulation can be used to evaluate the exposed behavior. There are two main approaches:

Approach 1: system simulation

Simulate system scenarios. Observe whether system behavior is as expected using execution logs or visualizations.

Examples of simulators

Eclipse ESCET / CIF**,** Matlab simulink, POOSL

Examples of visualization engines

Blender, Cry, Panda3D, Unity, Unreal Engine

Approach 2: activity diagram simulation

Simulate individual activities. Observe whether the behavior is as expected using Gantt charts or animation of activity.

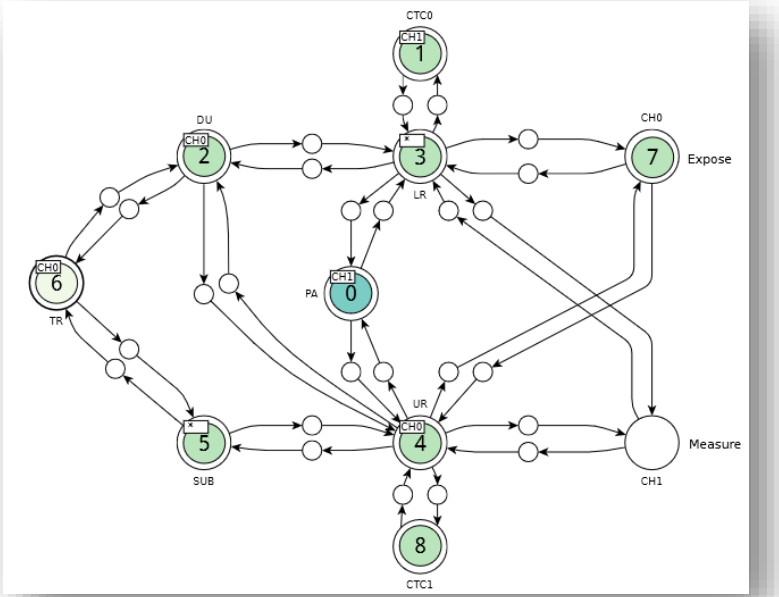
Examples of simulators

- **LSAT:** focus on timing, visualized in Gantt chart
- **fUML execution engine:** no logging/visualization
- **Cameo Simulation Toolkit:** animation, basic Gantt chart
- **Papyrus Moka:** animation

Question 1

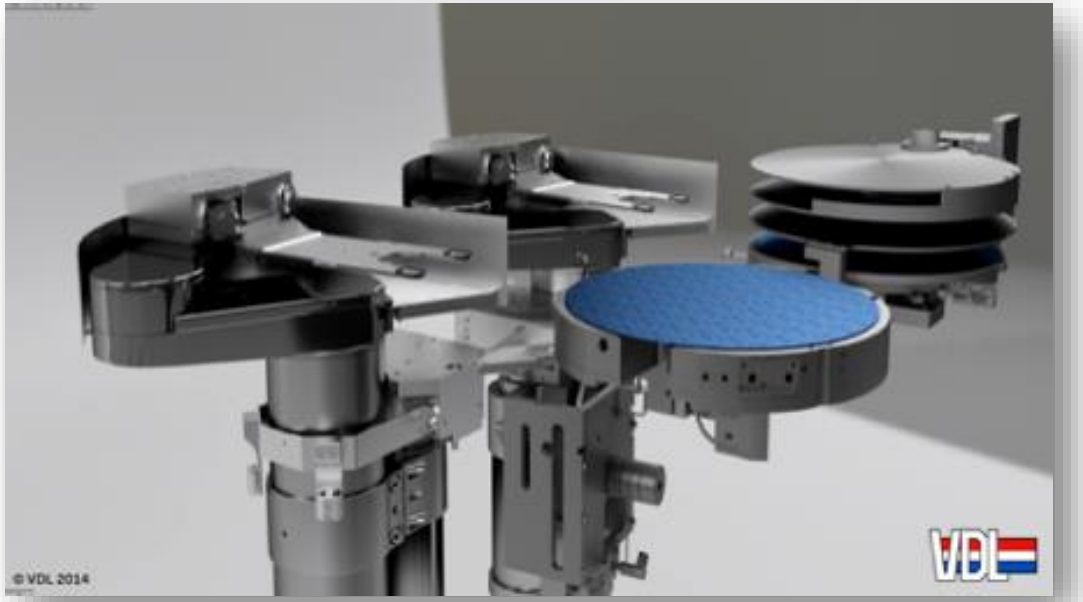
How does the specification behave, i.e., is the exposed behavior as expected?

Examples of system simulation



Interactive visualization in CIF

van der Sanden et al., “Modular Model-Based Supervisory Controller Design for Wafer Logistics in Lithography Machines”, MODELS 2015



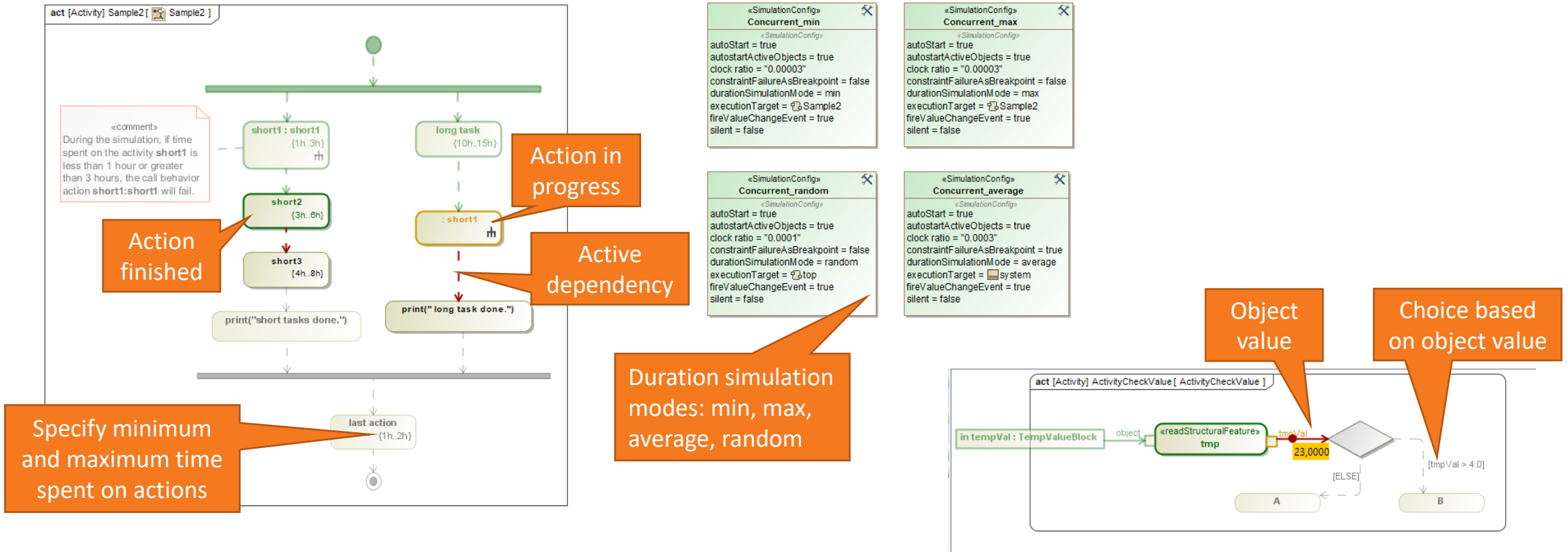
Visualization in Blender

Video rendered with Blender, created by Adrie Bovenhof

Question 1

How does the specification behave, i.e., is the exposed behavior as expected?

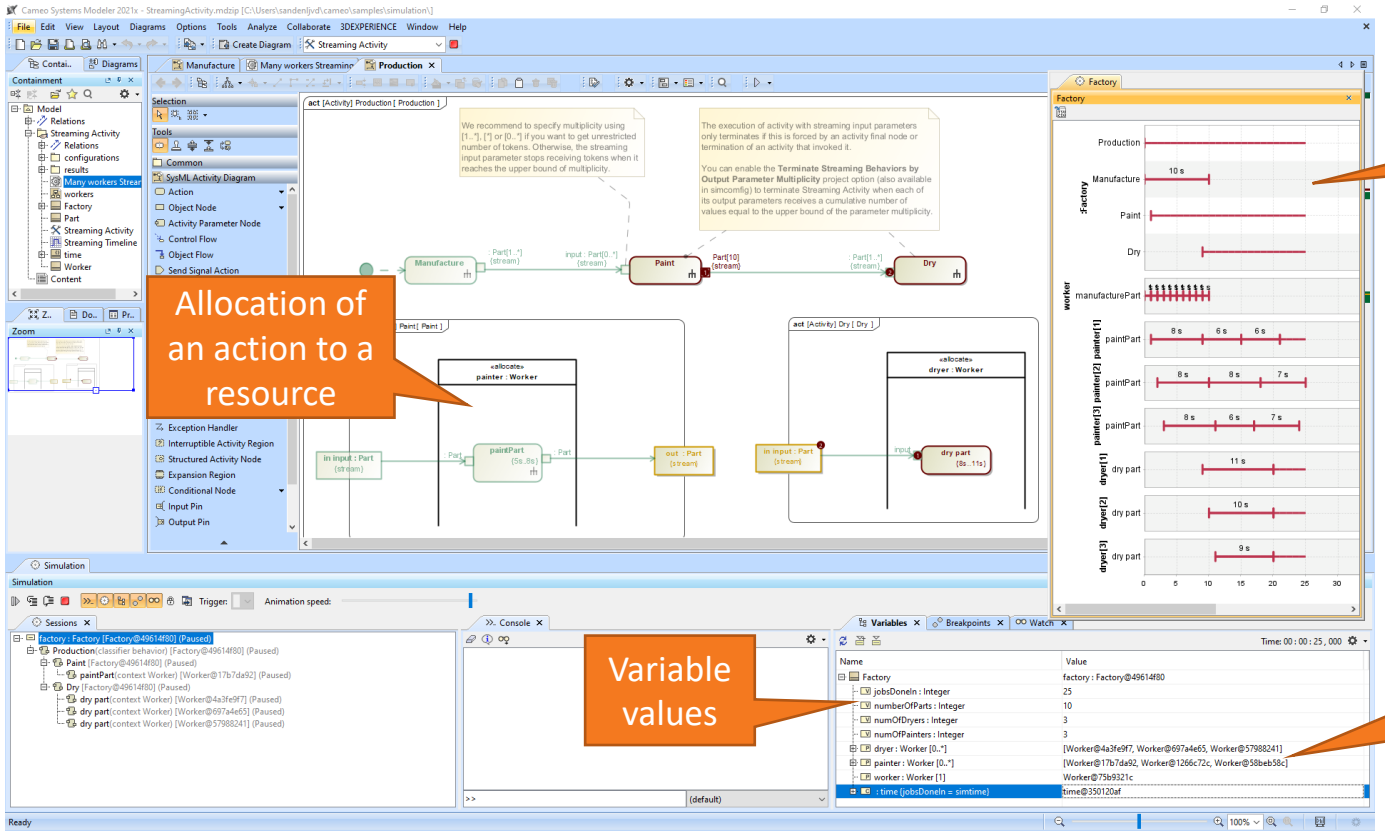
Examples of activity diagram simulation in Cameo Simulation Toolkit (1/2)



Question 1

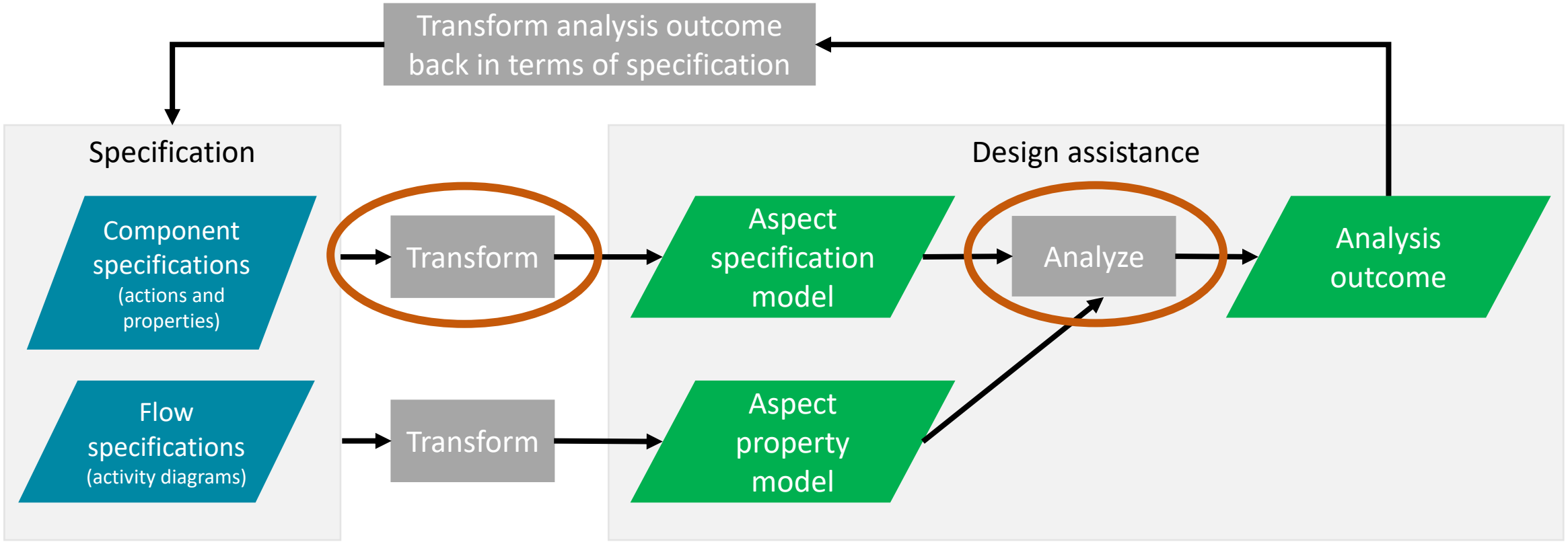
How does the specification behave, i.e., is the exposed behavior as expected?

Examples of activity diagram simulation in Cameo Simulation Toolkit (2/2)



Question 2

Does the specification guarantee a particular safety property, […]?



Question 2

Does the specification guarantee a particular safety property, […]?

"Transform" – examples of available transformations, starting from an activity diagram

Source	Target	Implementation?	Reference
UML2 activity	Petri Net	Yes; VIATRA2	https://wiki.eclipse.org/VIATRA2/Activity_Diagrams_to_Petri_Nets
UML activity	Petri Net	No	Automatic translation UML activity diagrams to Petri net https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7147141
UML activity	Petri Net	Yes; via QVTO	https://github.com/MDE4CPP/AD2PN
UML activity	CSP	No	https://www.researchgate.net/publication/228745131_Case_study_UML_to_CSP_transformation
UML activity	Pi-calculus	No	https://ceur-ws.org/Vol-2326/paper14.pdf
UML activity	PROMELA	No	https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5363181
UML activity	NuSMV	No	https://dl.acm.org/doi/10.1145/1125808.1125809
UML activity	Automaton	Yes	https://www.se-rwth.de/materials/semdiff/

Question 2

Does the specification guarantee a particular safety property, […]?

"Verify property" – examples of property checking tools

Type	Tool	Specification language	Property specification language
Model checkers Check property	mCRL2	mCRL2; process algebra	Modal mu-calculus with data
	TAPAAL	Timed-Arc Petri Net	CTL, LTL, subset of CTL considering time
	TINA	Timed-Arc Petri Net	Modal mu-calculus, state/event LTL
	SPIN	PROMELA	LTL
	ITS-tools	LTSmin, Spin, PNML, Uppaal, Romeo, Tina, DiVinE	CTL, LTL, safety in propositional logic
	NuSMV	NuSMV specification	LTL, CTL
SMT solvers Check satisfiability	Z3, cvc5	SMTLIB2; first-order logic	SMTLIB2 first-order logic
Supervisory controller synthesis Check nonblocking	CIF	CIF specification; EFA	CIF specification; EFA, invariant
	Supremica	Supremica specification; EFA	Supremica specification; EFA

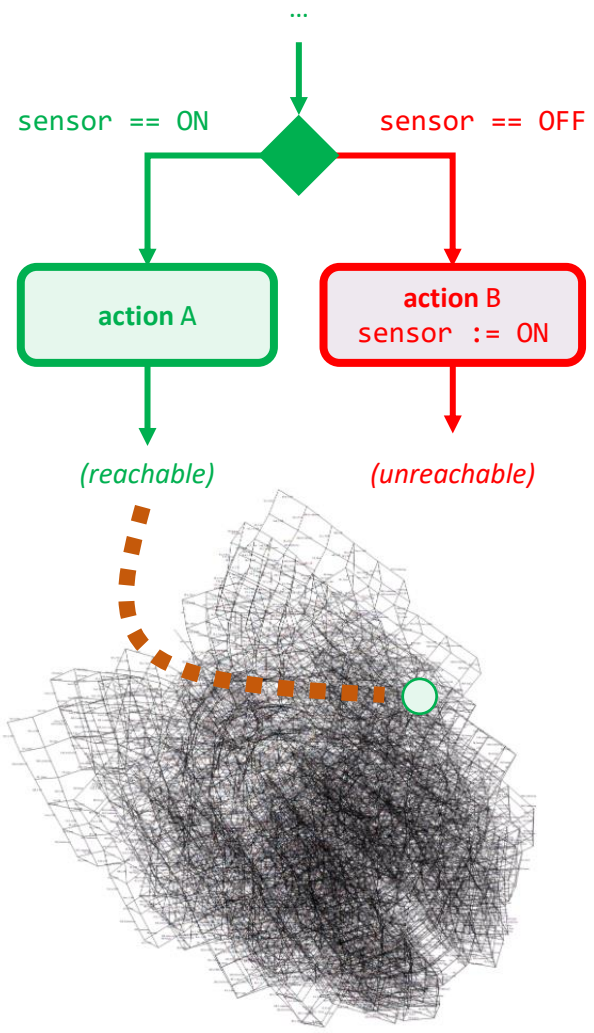
Note: this is a compact overview of well-known tools. For many formalisms, there is a large variety of tools supporting different flavors.

Question 3

Are the expected parts of the specification reachable?

Typical method to assess reachability is by model checking:

- A **state space exploration** can be performed to compute the reachable part of the specification.
- A **model checker** can be used to check whether certain states can always be reached, given the (possible) data values.
- **Reachability properties** can be formulated in a property language like LTL or CTL (e.g., we can eventually always reach a certain state).



Source of bottom image: van der Sanden, “Performance analysis and optimization of supervisory controllers”, PhD thesis 2018

Question 4

Can the specification be realized, i.e., are the requirements non-conflicting?

Requirements are conflicting if they are inconsistent, i.e., the contradict each other.

Various types of requirements:

- Functional — e.g., action ordering, data, no deadlocks, no livelocks, controllable
- Timing — e.g., duration within time bound
- Other qualities — e.g., cost, contamination, overlay impact within specified bounds, stability of the schedule

Depending on the type of requirements, different approaches are available:

1. Automata-based — functional (action ordering, data), timing
2. Ontology-based — functional, timing, other qualities

Question 4

Can the specification be realized, i.e., are the requirements non-conflicting?

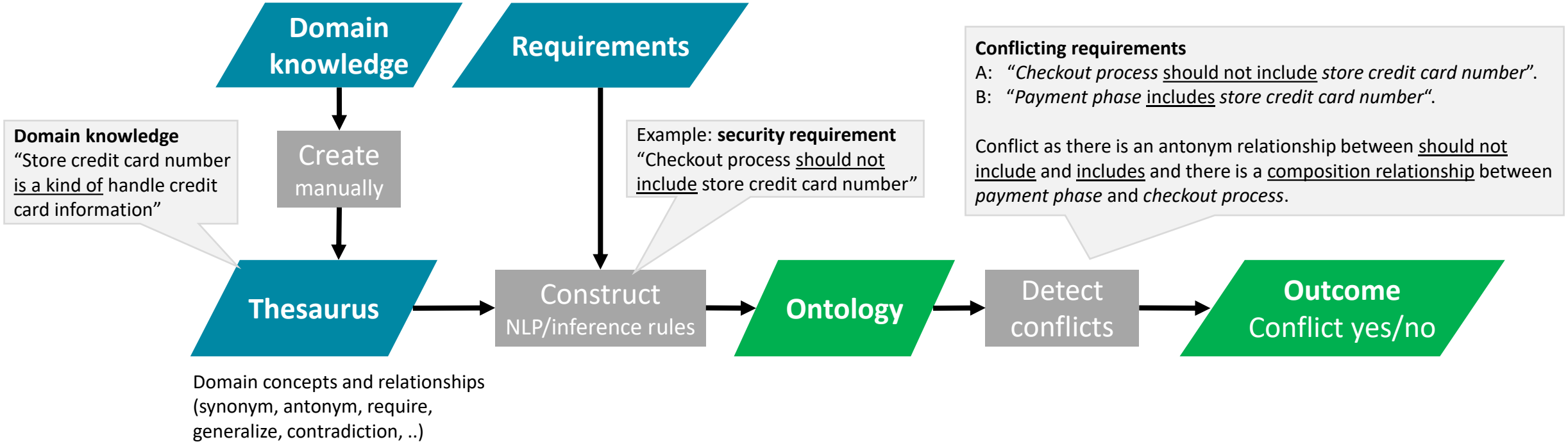
Automata-based approach to automatically identify conflicting requirements.

- **Context of properties in model checking**
  - Functional requirements can typically be translated to an automaton, e.g., an LTL expression.
  - To check whether two requirements are conflicting, we compute the intersection of the corresponding automata. If the intersection is empty, then the requirements are conflicting.
- **Context of requirement automata in synthesis**
  - Requirements regarding events and states:
    - Event-based requirements — specify ordering of events
    - State-based requirements — event(s) only allowed in certain state, mutual state exclusion
    - Time-based requirements — associating timing properties of and between states
  - Each requirement of this type can be translated to an automaton.
  - To check whether two requirements are conflicting, we compute the synchronous composition. If the synchronous composition is blocking, then the requirements are conflicting.

Question 4

Can the specification be realized, i.e., are the requirements non-conflicting?

Ontology-based approach to semi-automatically identify conflicting requirements.



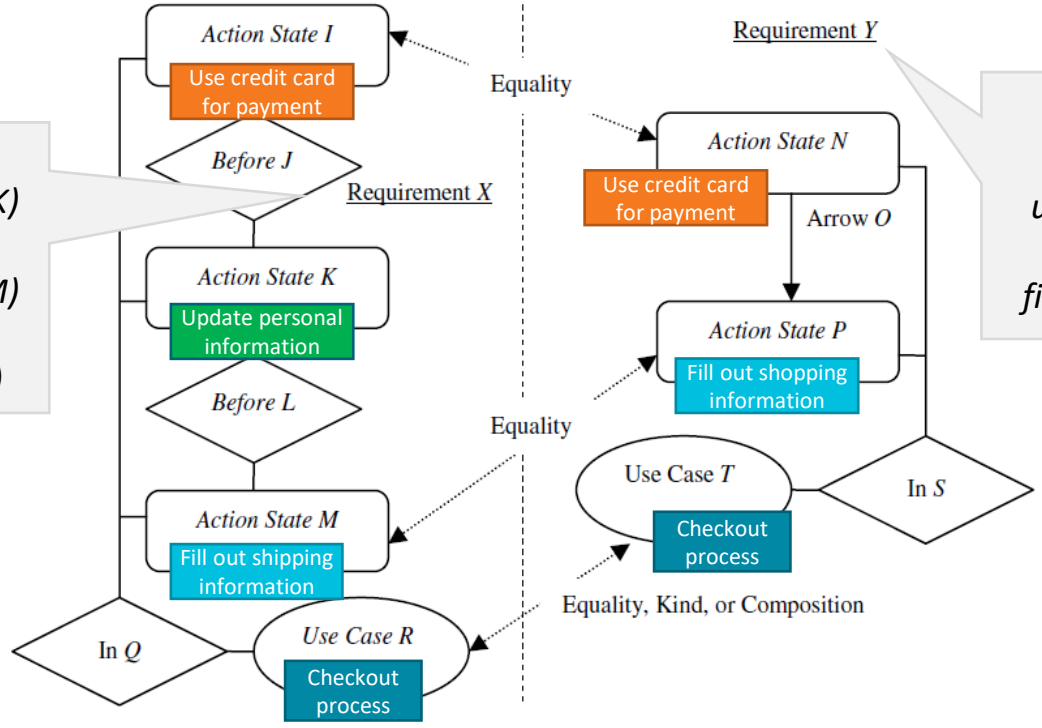
Question 4

Can the specification be realized, i.e., are the requirements non-conflicting?

Ontology-based approach to identify action ordering conflict, using predefined detection rules:

Example rule “Shortcut conflict”: Update personal information (action state K) is omitted and the shortcut conflict is occurred between requirement X and Y.

Requirement X:
Update personal information (action state K)
comes before
Fill out shipping information (action state M)
and comes after
use credit card for payment (action state I)



Requirement Y:
“The next step of
use credit card for payment (action state N)
is
fill out shipping information (action state P)”

Source of image: Ontology-Based Requirement Conflicts Analysis in Activity Diagrams, Liu (2009)

Question 5

Does the specification guarantee a particular timing property?

"Transform" – examples of available transformations considering time, starting from an activity diagram

Source	Target	Implementation?	Reference
UML activity	PROMELA, SPIN	Yes, UML-VT	Comparing model checkers for timed UML activity diagrams https://www.sciencedirect.com/science/article/pii/S0167642315001045?ref=cra_js_challenge&fr=RR-1 https://ceur-ws.org/Vol-1554/PD_MoDELS_2015_paper_16.pdf http://www.cs.umd.edu/~rance/projects/uml-vt/
UML activity	NuSMV		
UML activity	UPPAAL		
UML activity	PES		
UML activity	UPPAAL-SMC	No	Quantitative Timing Analysis of UML Activity Diagrams Using Statistical Model Checking https://past.date-conference.com/proceedings-archive/2016/pdf/0339.pdf
SysML activity	PRISM	No	https://www.sciencedirect.com/science/article/pii/S0957417413008968

Question 5

Does the specification guarantee a particular timing property?

“Verify property” – examples of property checking tools for timing properties

Type	Tool	Specification language	Property specification language
Model checkers Check property	UPPAAL	Networks of timed automata, extended with data types	UPPAAL requirement specification language
	TAPAAL	Timed-Arc Petri Net	CTL, LTL, subset of CTL considering time
	TINA	Timed-Arc Petri Net	Modal mu-calculus, state/event LTL
	SPIN	PROMELA	LTL
	PRISM	PRISM language	Probabilistic temporal logics: PCTL, CSL, Probabilistic LTL, PCTL*; quantitative specification; costs; rewards

State-of-the-art study: Process summary and conclusions

Process summary

- Distributed design-assistance questionnaire to focus the study
- Analyzed the responses and identified the top 5 questions
- Created overviews of available analysis methods/techniques for each top-5-question

Conclusions

- Multiple suitable methods/tools exist for each question
- This gives confidence that the questions can likely be answered
- Possible methods/tools will be investigated further, starting from the feasibility study

Acknowledgements

This research is carried out as part of the Poka Yoka program under the responsibility of TNO-ESI in cooperation with ASML and VDL-ETG. The research activities are supported by the Netherlands Organisation for Applied Scientific Research TNO, the Netherlands Ministry of Economic Affairs and Climate Policy, and TKI-HTSM.