AI- and Formal Methods-Based Model Design

Ludovic Apvrille & Bastien Sultan
ludovic.apvrille,bastien.sultan@telecom-paris.fr

Journée thématique GDR SoC2

Outline

A few preliminary words

Our scope

- **M**odel-**D**riven **E**ngineering
  - Uses models for system analysis (including requirement engineering) and design
- Software/hardware partitioning, and specific focus on software design
- Automation:
  - Pattern integration
  - Model transformation and evolution (mutation)
  - AI-based assistance

A few preliminary words

AI-based assistance in MDE

Juri Di Rocco et al. 2025. **On the use of large language models in model-driven engineering**. Softw. Syst. Model. 24, 3

- Model completion
- Model generation

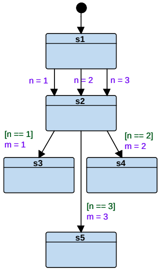
Complementary tasks

- Consistency analysis
- Requirement elicitation
- Derivation of properties from requirements
- Model mutation

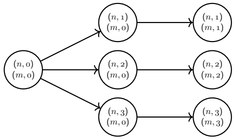
A few preliminary words

Our (first) scope: **formal** MDE

- Mathematically-grounded modeling languages
- Expression of requirements in temporal logic properties
- Formal verification (including model checking)
- Simulation

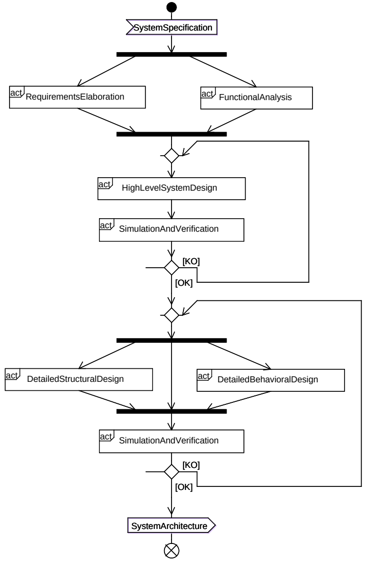


$\langle s_0, S, T\rangle \mapsto \langle V, E, A, val\rangle$
– $E \subseteq V \times V$
– $val : V \to \{(a,x) | a \in A, x \in type(a)\}$
– $type : A \to \{\mathbb{B}ool, \mathbb{Z}\}$



T $E <> n = m$
F $A[]\ n = m$
T $A <> n = m$

A few preliminary words


SystemSpecification
act RequirementsElaboration
act FunctionalAnalysis
act HighLevelSystemDesign
act SimulationAndVerification
[KO]
[OK]
act DetailedStructuralDesign
act DetailedBehavioralDesign
act SimulationAndVerification
[KO]
[OK]
SystemArchitecture

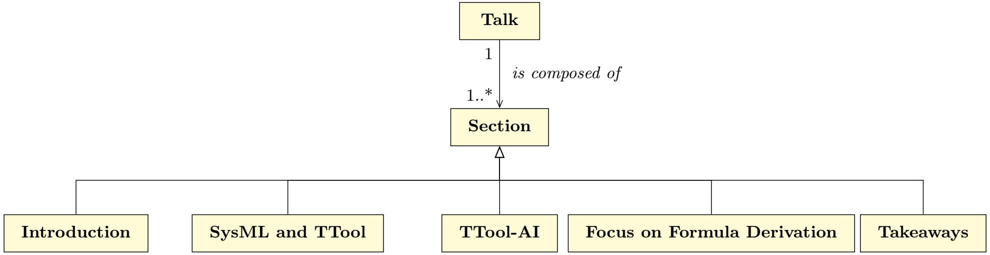
Our (second) scope: **incremental** MDE

- MDE as a dynamic, iterative process
- Model mutations
- Incremental model checking

AI-assisted MDE

- Drafting models from specification, generating model mutation scripts, deriving CTL formulas from requirements, consistency analysis, ...
- Continuous AI modeling assistance

Outline


Talk
1
is composed of
1..*
Section
Introduction
SysML and TTool
TTool-AI
Focus on Formula Derivation
Takeaways

Outline

Some context – the Systems Modeling Language

Scope

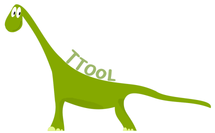
- Systems engineering
- Analysis (requirements, functional, structural) and design (structural, behavioral)

$\rightarrow$ Supports most MDE stages

Strongly anchored

- Standardized by the OMG (current members include Airbus, Dassault, Microsoft, NASA, etc.)
- Widely supported by software tools

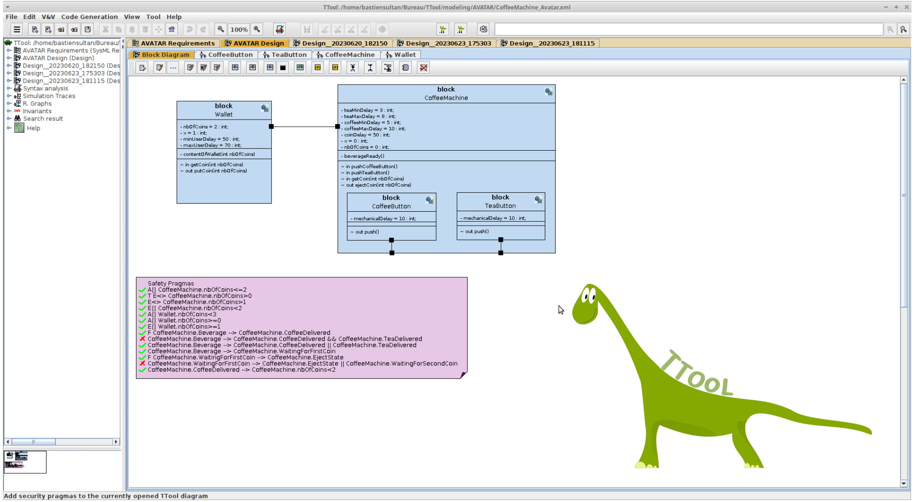
TTool



In a nutshell

- Free and open-source MDE toolkit
- Formally defined SysML profiles targeting generic systems as well as real-time embedded systems design
- Simulation, formal verification (model checking of CTL formulas, or translation to ProVerif for security properties)
- Code generation
- User assistance: automated pattern integration, mutation language, AI-based assistance

TTool


TTool: /home/bastiensultan/Bureau/TTool/modeling/AVATAR/CoffeeMachine_Avatar.xml
File Edit V&V Code Generation View Tool Help
100%
TTool: /home/bastiensultan/Bureau/
AVATAR Requirements (SysML Re
AVATAR Design (Design)
Design__20230620_182150 (Des
Design__20230623_175303 (Des
Design__20230623_181115 (Des
Syntax analysis
Simulation Traces
R. Graphs
Invariants
Search result
Help
AVATAR Requirements
AVATAR Design
Design__20230620_182150
Design__20230623_175303
Design__20230623_181115
Block Diagram
CoffeeButton
TeaButton
CoffeeMachine
Wallet
TTool

block
Wallet

- nbOfCoins = 2 : int;
- x = 1 : int;
- minUserDelay = 50 : int;
- maxUserDelay = 70 : int;

- contentOfWallet(int nbOfCoins)

~ in getCoin(int nbOfCoins)
~ out putCoin(int nbOfCoins)

block
CoffeeMachine

- teaMinDelay = 3 : int;
- teaMaxDelay = 8 : int;
- coffeeMinDelay = 5 : int;
- coffeeMaxDelay = 10 : int;
- coinDelay = 50 : int;
- x = 0 : int;
- nbOfCoins = 0 : int;

- beverageReady()

~ in pushCoffeeButton()
~ in pushTeaButton()
~ in getCoin(int nbOfCoins)
~ out ejectCoin(int nbOfCoins)

block
CoffeeButton

- mechanicalDelay = 10 : int;

~ out push()

block
TeaButton

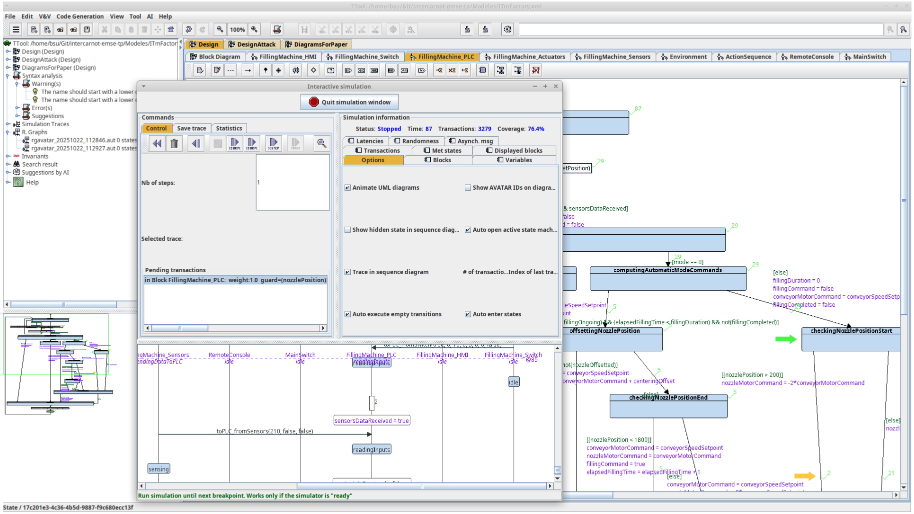
- mechanicalDelay = 10 : int;

~ out push()

```
Safety Pragmas
✓ A[] CoffeeMachine.nbOfCoins<=2
✓ T E<> CoffeeMachine.nbOfCoins>0
✓ E<> CoffeeMachine.nbOfCoins>1
✓ E[] CoffeeMachine.nbOfCoins<2
✓ A[] Wallet.nbOfCoins<3
✓ A[] Wallet.nbOfCoins>=0
✓ E[] Wallet.nbOfCoins>=1
✓ F CoffeeMachine.Beverage --> CoffeeMachine.CoffeeDelivered
✗ CoffeeMachine.Beverage --> CoffeeMachine.CoffeeDelivered && CoffeeMachine.TeaDelivered
✓ CoffeeMachine.Beverage --> CoffeeMachine.CoffeeDelivered || CoffeeMachine.TeaDelivered
✓ CoffeeMachine.Beverage --> CoffeeMachine.WaitingForFirstCoin
✓ F CoffeeMachine.WaitingForFirstCoin --> CoffeeMachine.EjectState
✗ CoffeeMachine.WaitingForFirstCoin --> CoffeeMachine.EjectState || CoffeeMachine.WaitingForSecondCoin
✓ CoffeeMachine.CoffeeDelivered --> CoffeeMachine.nbOfCoins<2
```

Add security pragmas to the currently opened TTool diagram

Simulation with TTool


TTool: /home/bsu/Git/intercarnot-emse-tp/Modeles/ITmFactory.xml
File Edit V&V Code Generation View Tool AI Help
Design DesignAttack DiagramsForPaper
Block Diagram FillingMachine_HMI FillingMachine_Switch FillingMachine_PLC FillingMachine_Actuators FillingMachine_Sensors Environment ActionSequence RemoteConsole MainSwitch
Design (Design)
DesignAttack (Design)
DiagramsForPaper (Design)
Syntax analysis
Warning(s)
The name should start with a lower
Error(s)
Suggestions
Simulation Traces
R. Graphs
rgavatar_20251022_112846.aut 0 states
rgavatar_20251022_112927.aut 0 states
Invariants
Search result
Suggestions by AI
Help
Interactive simulation
Quit simulation window
Commands
Control Save trace Statistics
Simulation information
Latencies Randomness Asynch. msg
Transactions Met states Displayed blocks
Options Blocks Variables
computingAutomaticModeCommands
offsettingNozzlePosition
checkingNozzlePositionStart
checkingNozzlePositionEnd
sensing
readingInputs
idle

Status: Stopped Time: 87 Transactions: 3279 Coverage: 76.4%

Nb of steps: 1

Selected trace:

Pending transactions

in Block FillingMachine_PLC: weight:1.0 guard=(nozzlePosition):

☑ Animate UML diagrams ☐ Show AVATAR IDs on diagra...

☐ Show hidden state in sequence diag... ☑ Auto open active state mach...

☑ Trace in sequence diagram # of transactio...Index of last tra...

☑ Auto execute empty transitions ☑ Auto enter states

FillingMachine_Sensors RemoteConsole MainSwitch FillingMachine_PLC FillingMachine_HMI FillingMachine_Switch

sensorsDataReceived = true

toPLC_fromSensors(210, false, false)

[mode == 0]

[else]
fillingDuration = 0
fillingCommand = false
conveyorMotorCommand = conveyorSpeedSetp
fillingCompleted = false

fillingOngoing) && (elapsedFillingTime < fillingDuration) && not(fillingCompleted)]

not(nozzleOffsetted)]
= conveyorSpeedSetpoint
onveyorMotorCommand + centeringOffset

[(nozzlePosition > 200)]
nozzleMotorCommand = -2*conveyorMotorCommand

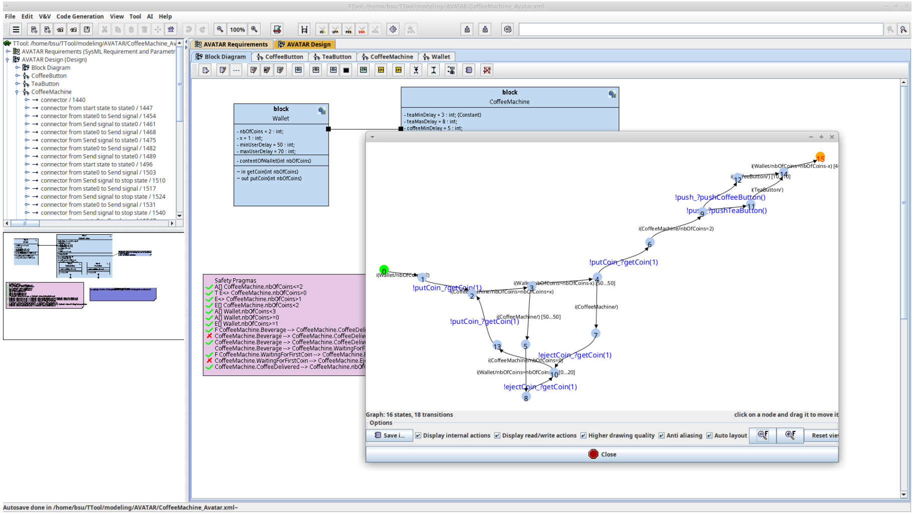
[(nozzlePosition < 1800)]
conveyorMotorCommand = conveyorSpeedSetpoint
nozzleMotorCommand = conveyorMotorCommand
fillingCommand = true
elapsedFillingTime = elapsedFillingTime + 1

[else]
conveyorMotorCommand = conveyorSpeedSetpoint

Run simulation until next breakpoint. Works only if the simulator is "ready"

State / 17c201e3-4c36-4b5d-9887-f9c680ecc13f

Formal verification with TTool


TTool: /home/bsu/TTool/modeling/AVATAR/CoffeeMachine_Avatar.xml
File Edit V&V Code Generation View Tool AI Help
100%
AVATAR Requirements
AVATAR Design
Block Diagram
CoffeeButton
TeaButton
CoffeeMachine
Wallet
Graph: 16 states, 18 transitions
click on a node and drag it to move it
Options
Save i...
Display internal actions
Display read/write actions
Higher drawing quality
Anti aliasing
Auto layout
Reset view
Close
!putCoin_?getCoin(1)
!push_?pushCoffeeButton()
!push_?pushTeaButton()
!ejectCoin_?getCoin(1)
i(CoffeeMachine/nbOfCoins=2)
i(CoffeeMachine/)
i(CoffeeMachine/) [50...50]
i(Wallet/nbOfCoins=nbOfCoins-x) [50...50]
i(CoffeeMachine/nbOfCoins=nbOfCoins+x)
i(CoffeeMachine/nbOfCoins=0)
i(Wallet/nbOfCoins=nbOfCoins+x) [0...20]
i(TeaButton/)
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

TTool: /home/bsu/TTool/modeling/AVATAR/CoffeeMachine_Av
- AVATAR Requirements (SysML Requirement and Parametr
- AVATAR Design (Design)
  - Block Diagram
  - CoffeeButton
  - TeaButton
  - CoffeeMachine
    - → connector / 1440
    - → connector from start state to state0 / 1447
    - → connector from state0 to Send signal / 1454
    - → connector from Send signal to state0 / 1461
    - → connector from state0 to Send signal / 1468
    - → connector from Send signal to state0 / 1475
    - → connector from state0 to Send signal / 1482
    - → connector from Send signal to state0 / 1489
    - → connector from start state to state0 / 1496
    - → connector from state0 to Send signal / 1503
    - → connector from Send signal to stop state / 1510
    - → connector from state0 to Send signal / 1517
    - → connector from Send signal to stop state / 1524
    - → connector from state0 to Send signal / 1531
    - → connector from Send signal to stop state / 1540

block
Wallet
- nbOfCoins = 2 : int;
- x = 1 : int;
- minUserDelay = 50 : int;
- maxUserDelay = 70 : int;

- contentOfWallet(int nbOfCoins)

~ in getCoin(int nbOfCoins)
~ out putCoin(int nbOfCoins)

block
CoffeeMachine
- teaMinDelay = 3 : int; {Constant}
- teaMaxDelay = 8 : int;
- coffeeMinDelay = 5 : int;

Safety Pragmas
- ✓ A[] CoffeeMachine.nbOfCoins<=2
- ✓ T E<> CoffeeMachine.nbOfCoins>0
- ✓ E<> CoffeeMachine.nbOfCoins>1
- ✓ E[] CoffeeMachine.nbOfCoins<2
- ✓ A[] Wallet.nbOfCoins<3
- ✓ A[] Wallet.nbOfCoins>=0
- ✓ E[] Wallet.nbOfCoins>=1
- ✓ F CoffeeMachine.Beverage --> CoffeeMachine.CoffeeDeli
- ✗ CoffeeMachine.Beverage --> CoffeeMachine.CoffeeDeliv
- ✓ CoffeeMachine.Beverage --> CoffeeMachine.CoffeeDeliv
- CoffeeMachine.Beverage --> CoffeeMachine.WaitingForF
- ✓ F CoffeeMachine.WaitingForFirstCoin --> CoffeeMachine.E
- ✗ CoffeeMachine.WaitingForFirstCoin --> CoffeeMachine.Ej
- ✓ CoffeeMachine.CoffeeDelivered --> CoffeeMachine.nbOf

Autosave done in /home/bsu/TTool/modeling/AVATAR/CoffeeMachine_Avatar.xml~

Outline

LLM-based assistance: basics

Figure: A monkey.

First requirement

Filter LLM responses with algorithmic, reliable filters

LLM-based assistance: basics

Figure: A monkey.

Figure: An infinite monkey.

First requirement

Filter LLM responses with algorithmic, reliable filters

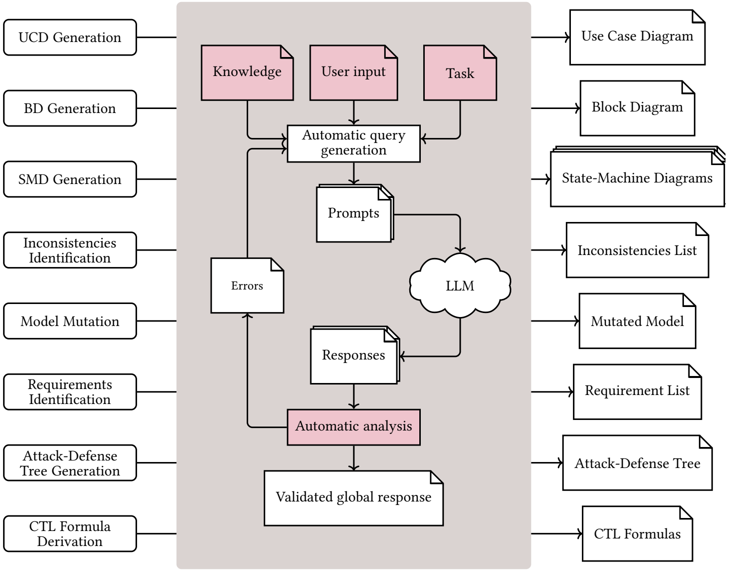
Second requirement

Retrieval-**A**ugmented **G**eneration + automatic feedback

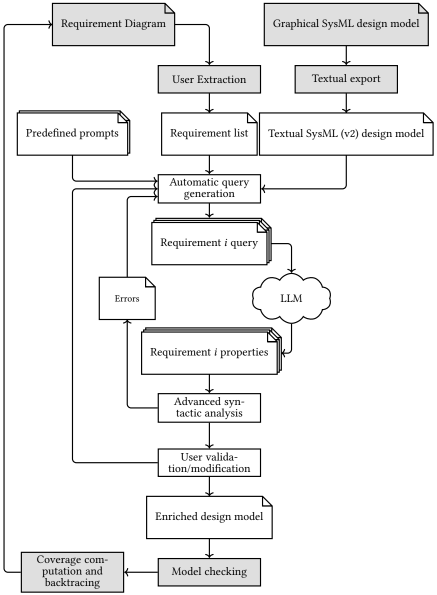
AI-based assistance in TTool

TTool-AI

- Provides automation in several MDE tasks
- RAG + algorithmic analysis of the response correctness
- Automatic feedback loop until the AI converges towards a response that meets the correctness criteria
- Syntax checking and model checking are in the loop
- User is in the loop


UCD Generation
BD Generation
SMD Generation
Inconsistencies Identification
Model Mutation
Requirements Identification
Attack-Defense Tree Generation
CTL Formula Derivation
Knowledge
User input
Task
Automatic query generation
Prompts
LLM
Responses
Errors
Automatic analysis
Validated global response
Use Case Diagram
Block Diagram
State-Machine Diagrams
Inconsistencies List
Mutated Model
Requirement List
Attack-Defense Tree
CTL Formulas

Outline


Requirement Diagram
Graphical SysML design model
User Extraction
Textual export
Predefined prompts
Requirement list
Textual SysML (v2) design model
Automatic query generation
Requirement *i* query
Errors
LLM
Requirement *i* properties
Advanced syntactic analysis
User validation/modification
Enriched design model
Model checking
Coverage computation and backtracing

Predefined prompts (knowledge)

Temporal logic grammar we want to inject

$v ::= block.id$

$p ::= v \mid v == v \mid v! = v \mid v < v \mid v \leq v \mid v > v \mid v \geq v \mid v == cst \mid v! = cst \mid$
$v < cst \mid v \leq cst \mid v > cst \mid v \geq cst$

$\varphi ::= \bot \mid \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi$

$\Phi ::= A[]\varphi \mid E[]\varphi \mid A <> \varphi \mid E <> \varphi \mid \varphi - - > \varphi$

Predefined prompts (knowledge)

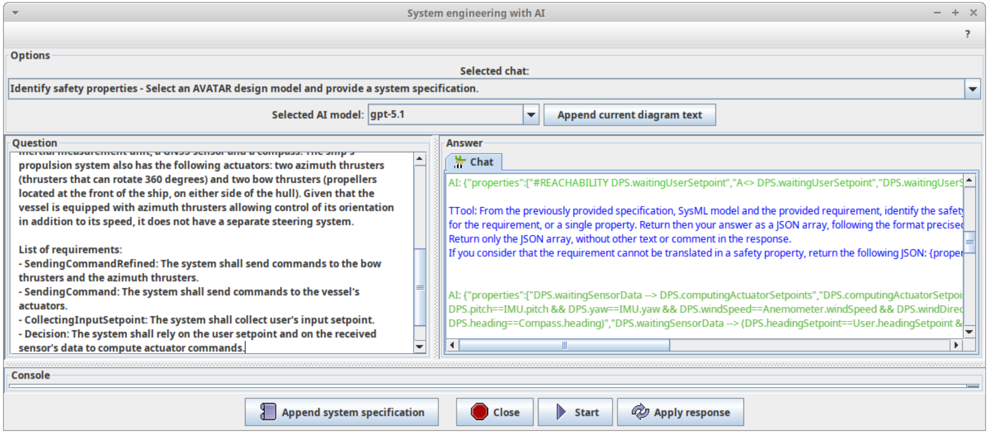
Resulting prompt

```
‘‘[...]  You shall respect the following rules:
#You can only use the following quantifiers, followed by a
property:  A<> prop, A[] prop, E<> prop, E[] prop, EXCEPT if you
use a --> in the formula.  In that case, DO NOT USE quantifiers.
So, for instance A<> prop is valid.  prop1 --> prop2 is also
valid.
#--> means ‘leads to‘.  It DOES NOT mean ‘implies‘.  ‘p --> q‘
means ‘whenever p is true, q will eventually hold true‘.  When
using a --> in a CTL formula, NEVER USE A QUANTIFIER in this CTL
formula.  So, in prop1 --> prop2, prop1 and prop2 are boolean
properties, without quantifiers.  [...]’’
```

Predefined prompts (feedback loop)

```
‘‘Your answer was not correct because of the following errors:
Safety Pragma A<> block.n = 42 cannot be parsed.  If you
intended to write an equality, use == instead of =.
In your new response, correct the properties that have errors as
listed in the errors list.  Provide one corrected property per
property that has errors.  In addition, keep the correct
properties (those for which no error was listed) that you
previously provided [...]’’
```

In practice



System engineering with AI

Options

Selected chat: Identify safety properties - Select an AVATAR design model and provide a system specification.

Selected AI model: gpt-5.1

Question

propulsion system also has the following actuators: two azimuth thrusters (thrusters that can rotate 360 degrees) and two bow thrusters (propellers located at the front of the ship, on either side of the hull). Given that the vessel is equipped with azimuth thrusters allowing control of its orientation in addition to its speed, it does not have a separate steering system.

List of requirements:
- SendingCommandRefined: The system shall send commands to the bow thrusters and the azimuth thrusters.
- SendingCommand: The system shall send commands to the vessel's actuators.
- CollectingInputSetpoint: The system shall collect user's input setpoint.
- Decision: The system shall rely on the user setpoint and on the received sensor's data to compute actuator commands.

Answer

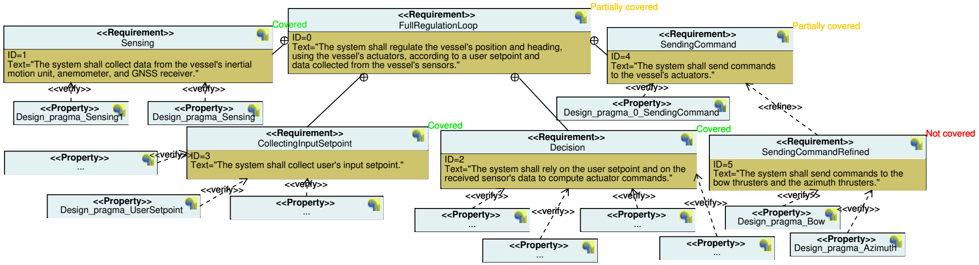
```
AI: {"properties":["#REACHABILITY DPS.waitingUserSetpoint","A<> DPS.waitingUserSetpoint","DPS.waitingUserS

TTool: From the previously provided specification, SysML model and the provided requirement, identify the safety
for the requirement, or a single property. Return then your answer as a JSON array, following the format precised
Return only the JSON array, without other text or comment in the response.
If you consider that the requirement cannot be translated in a safety property, return the following JSON: {proper


AI: {"properties":["DPS.waitingSensorData --> DPS.computingActuatorSetpoints","DPS.computingActuatorSetpoi
DPS.pitch==IMU.pitch && DPS.yaw==IMU.yaw && DPS.windSpeed==Anemometer.windSpeed && DPS.windDirec
DPS.heading==Compass.heading)","DPS.waitingSensorData --> (DPS.headingSetpoint==User.headingSetpoint &
```

Console

In practice


Partially covered
<<Requirement>>
FullRegulationLoop
ID=0
Text="The system shall regulate the vessel's position and heading, using the vessel's actuators, according to a user setpoint and data collected from the vessel's sensors."
Covered
<<Requirement>>
Sensing
ID=1
Text="The system shall collect data from the vessel's inertial motion unit, anemometer, and GNSS receiver."
<<verify>>
<<Property>>
Design_pragma_Sensing1
<<verify>>
<<Property>>
Design_pragma_Sensing
Partially covered
<<Requirement>>
SendingCommand
ID=4
Text="The system shall send commands to the vessel's actuators."
<<verify>>
<<Property>>
Design_pragma_0_SendingCommand
<<refine>>
Covered
<<Requirement>>
CollectingInputSetpoint
ID=3
Text="The system shall collect user's input setpoint."
<<Property>>
...
<<verify>>
<<verify>>
<<Property>>
Design_pragma_UserSetpoint
<<verify>>
<<Property>>
...
Covered
<<Requirement>>
Decision
ID=2
Text="The system shall rely on the user setpoint and on the received sensor's data to compute actuator commands."
<<verify>>
<<Property>>
...
<<verify>>
<<Property>>
...
<<verify>>
<<Property>>
...
<<verify>>
<<Property>>
...
Not covered
<<Requirement>>
SendingCommandRefined
ID=5
Text="The system shall send commands to the bow thrusters and the azimuth thrusters."
<<verify>>
<<Property>>
Design_pragma_Bow
<<verify>>
<<Property>>
Design_pragma_Azimuth

In practice

(a) Dynamic Positioning System

Before manual refinement			After manual refinement		
	Average	Min–Max		Average	Min–Max
Generation time (s)	29	11.5–40.3	Nb. of modified properties	0.8	0–2
Nb. of generated properties	12.4	8–21	Nb. of deleted properties	2.6	2–4
Nb. of relevant properties	9.2	5–19	Nb. of added properties	4.6	4–6
Rate of relevant properties (%)	70.1	55.6–90.5	Nb. of properties after modification	14.6	12–23
Nb. of associated requirements	4/5	3–5/5	Refinement time (min)	7 min 37 s	6 min 32 s – 8 min 23 s
			Quality score /10	5.3	3.7–7.6

(b) Satellite System

Before manual refinement			After manual refinement		
	Average	Min–Max		Average	Min–Max
Generation time (s)	51.3	30–85.1	Nb. of modified properties	0.7	0–2
Nb. of generated properties	8.7	5–11	Nb. of deleted properties	0.7	0–1
Nb. of relevant properties	7	2–10	Nb. of added properties	2	1–4
Rate of relevant properties (%)	73.6	40–90.9	Nb. of properties after modification	10	8–12
Nb. of associated requirements	3.2/5	1–4.5/5	Refinement time (min)	6 min 36	5 min – 7 min 42 s
			Quality score /10	6.9	3.3–9.2

About carbon footprint

Two interaction modes

- Remotely-hosted LLMs (on OpenAI, Mistral, ... servers)
- Self-hosted LLMs (on a machine hosted at Télécom Paris)

Some insights

Feature: model completion suggestions, self-hosted LLMs

- Grams of CO_2 per suggestion: 0.0041g to 0.0065g on average, depending on the LLM
- Average Google request footprint: 0.2g CO_2, 38 times more
- Obviously depends on the national energy mix...

Outline

Takeaways

LLMs for automating MDE: general toughts

- Widely integrated with MDE processes and tools
- Three key principles for a reliable integration: RAG, algorithmic verification, feedback loop
- Overall, good to excellent ability to generate or complete syntactically correct models
- Future works need to focus on semantic correctness

Takeaways

LLMs for automating MDE: lessons learnt from TTool-AI

- In model creation, the main benefit is saving time.
- Moreover, compared to Master's-level students, TTool-AI tends to perform better.
- Another particularly interesting feature is its ability to easily generate alternative designs.
- It is also very helpful for detecting and correcting inconsistencies, as well as suggesting model improvements.

Future works

Improving semantic correctness by integrating model checking of CTL formulas in model generation or completion loop

Questions?

The GDR SoC2 at the end of this talk
Hergé – *Coke en stock*, Casterman. 1958

Some references (1/2)

[1] Bastien Sultan & Ludovic Apvrille. 2025. **TTool-AI: A Large Language Model-Based Assistant for Model Driven Engineering**. SN Comput. Sci. 6, 7 (Oct 2025).

[2] Bastien Sultan & Ludovic Apvrille. 2024. **AI-Driven Consistency of SysML Diagrams**. In Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (MODELS '24). Association for Computing Machinery, New York, NY, USA, 149–159. *ACM SIGSOFT Distinguished Paper Award*.

[3] Ludovic Apvrille & Bastien Sultan. 2024. **System Architects are not alone Anymore: Automatic System Modeling with AI**. In Proceedings of the 12th International Conference on Model-Based Software and Systems Engineering (MODELSWARD 2024). 27-38. *Best paper award*.

Some references (2/2)

[4] Bastien Sultan, Ludovic Apvrille & Pierre de Saqui de Sannes. 2026. **Automated Derivation of Formal Properties from Requirements**. In Proceedings of the 20th IEEE International Systems Conference (SysCon '26). To appear.

[5] Ludovic Apvrille & Bastien Sultan. 2026. **Continuous AI Assistance for Model-Driven Engineering**. In Proceedings of the 14th International Conference on Model-Based Software and Systems Engineering (MODELSWARD 2026). 61-72.