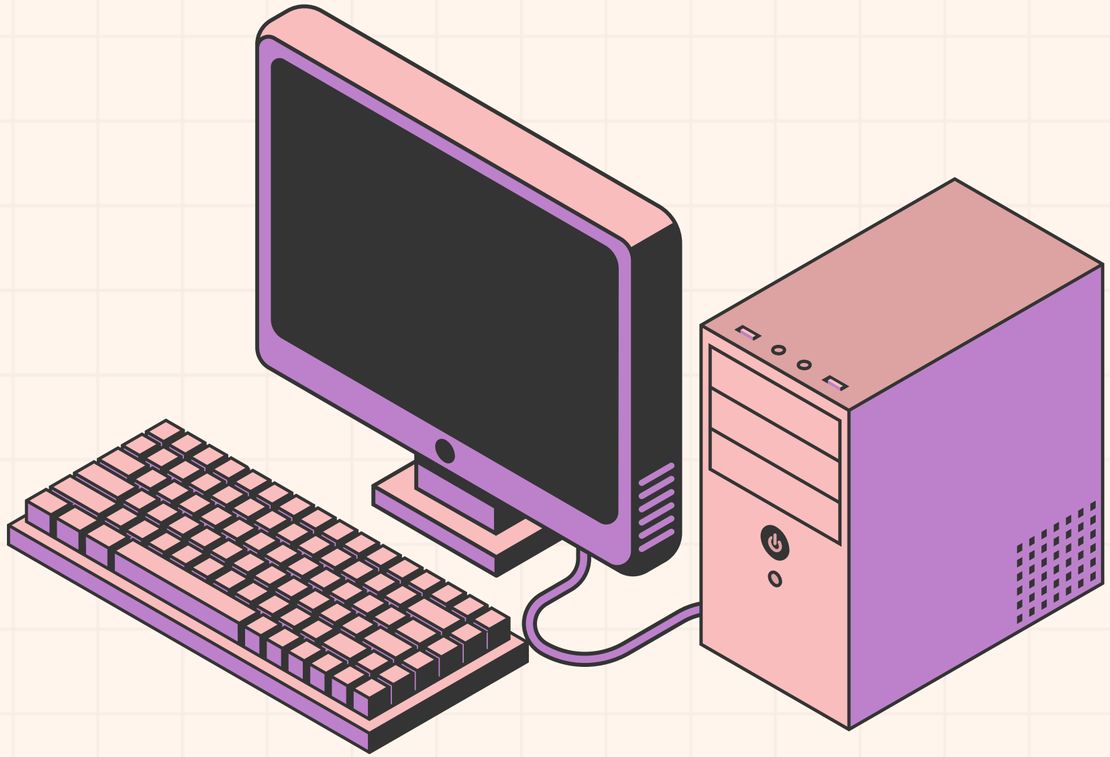
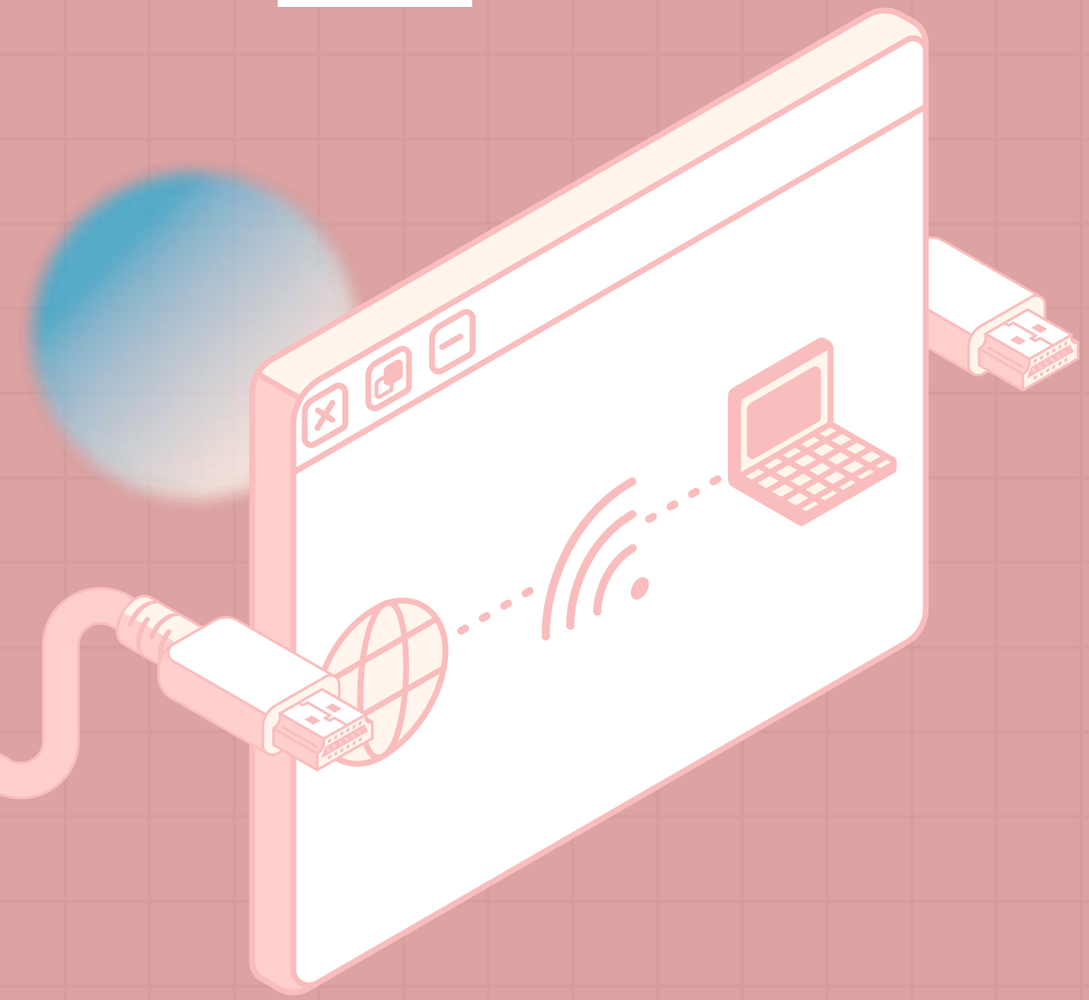
SBMF 2024 - 05/12/2024

VERIFYING INTEGRATED STATE MACHINES DESIGNS OF UML AND ACTIVITIES USING CSP



Diego Ferreira, **Lucas Lima**
Departamento de Computação - UFRPE



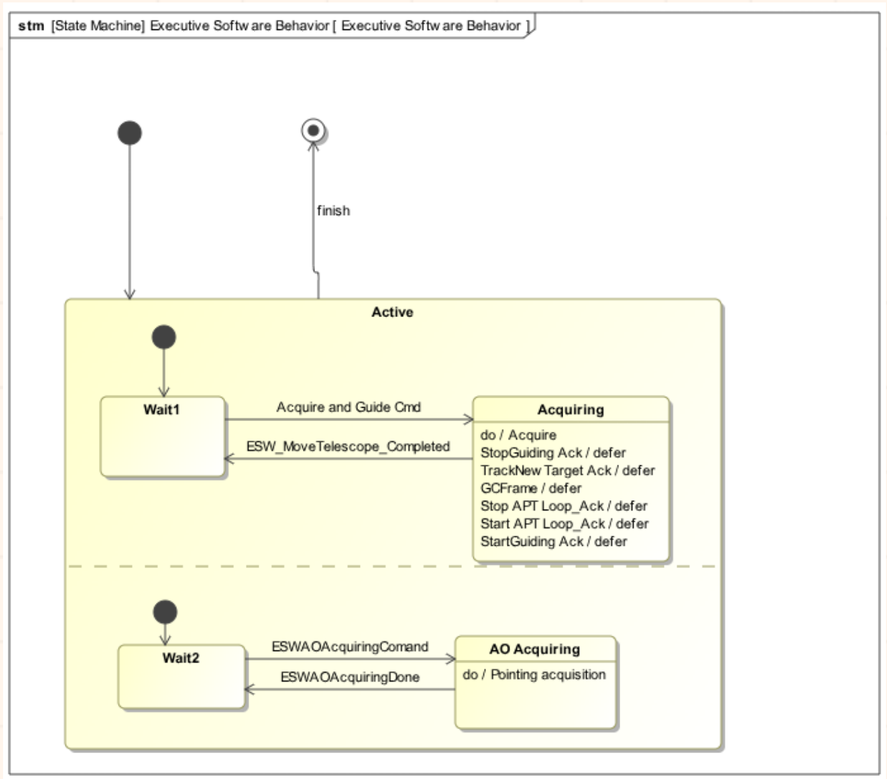
Context

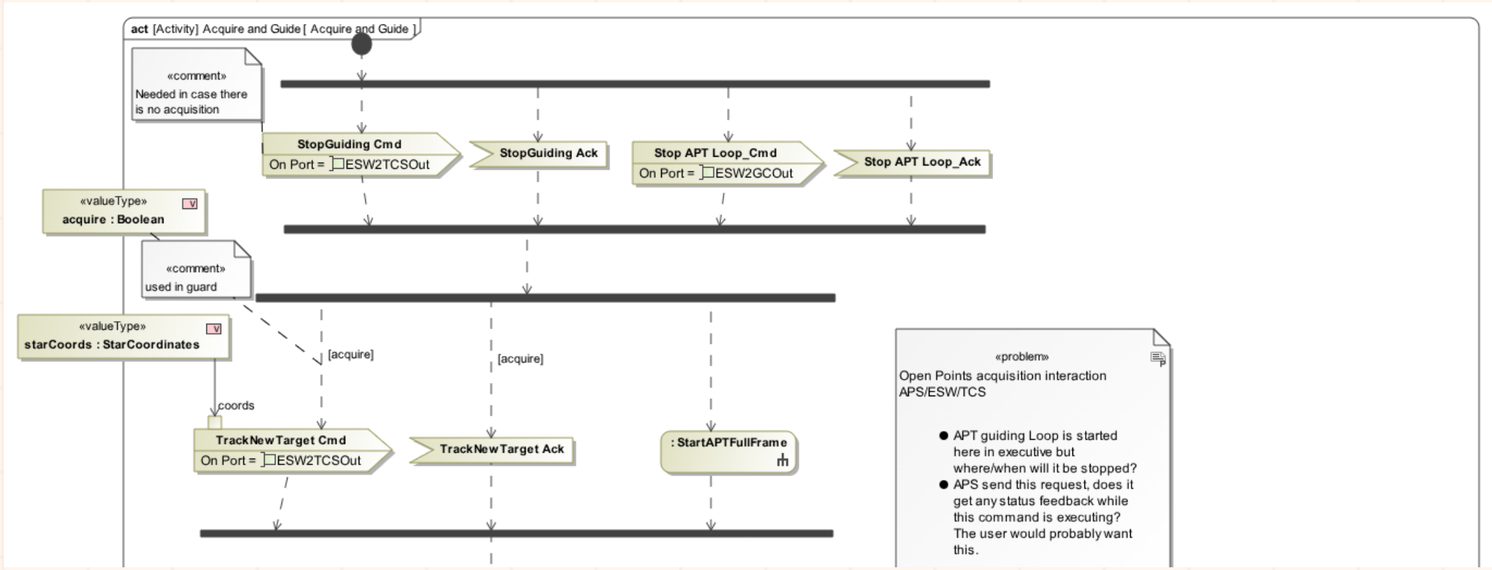
- Popular UML Diagrams
- Activity Diagram and State Machine Diagram integrated
- Common pattern used in industry

UFRPE

SBMF 2024

TMT - OpenMBEE





OpenMBEE

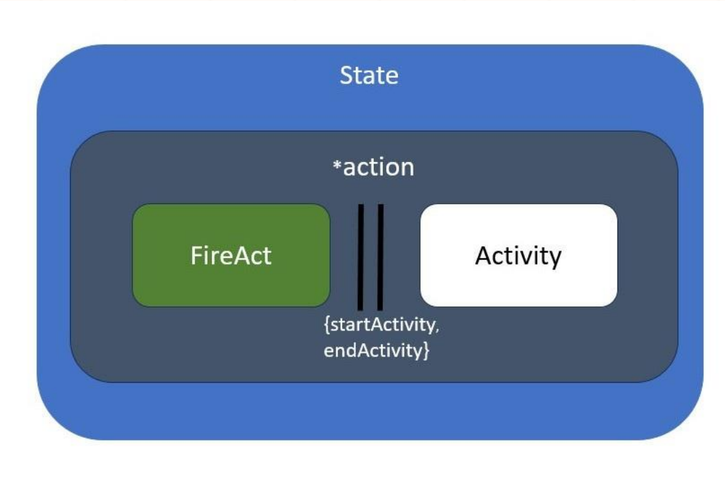
UFRPE

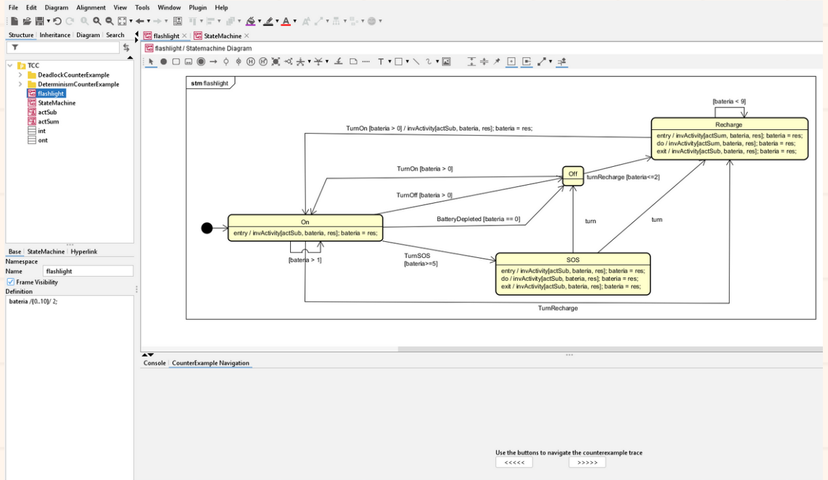
SBMF 2024 - 05/12/2024

RESEARCH PROJECT

Develop a framework to automatically verify deadlock and nondeterminism in integrated UML state machine and activity diagrams.

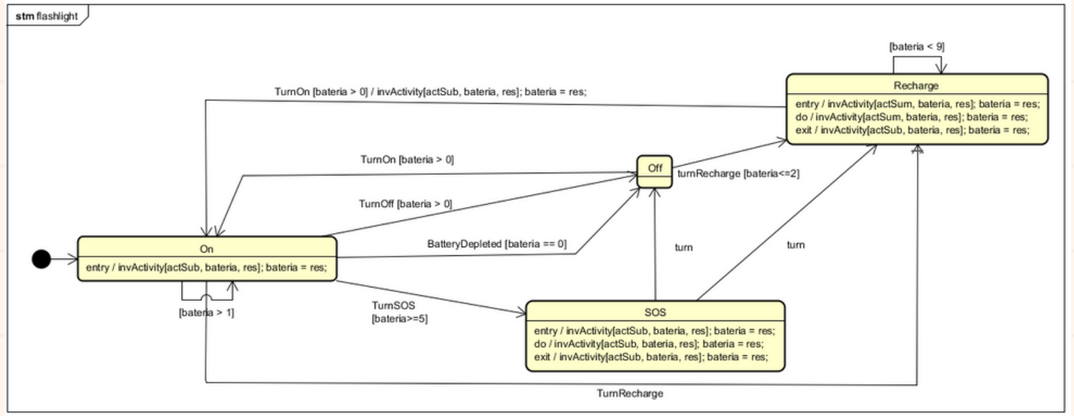
Contributions:



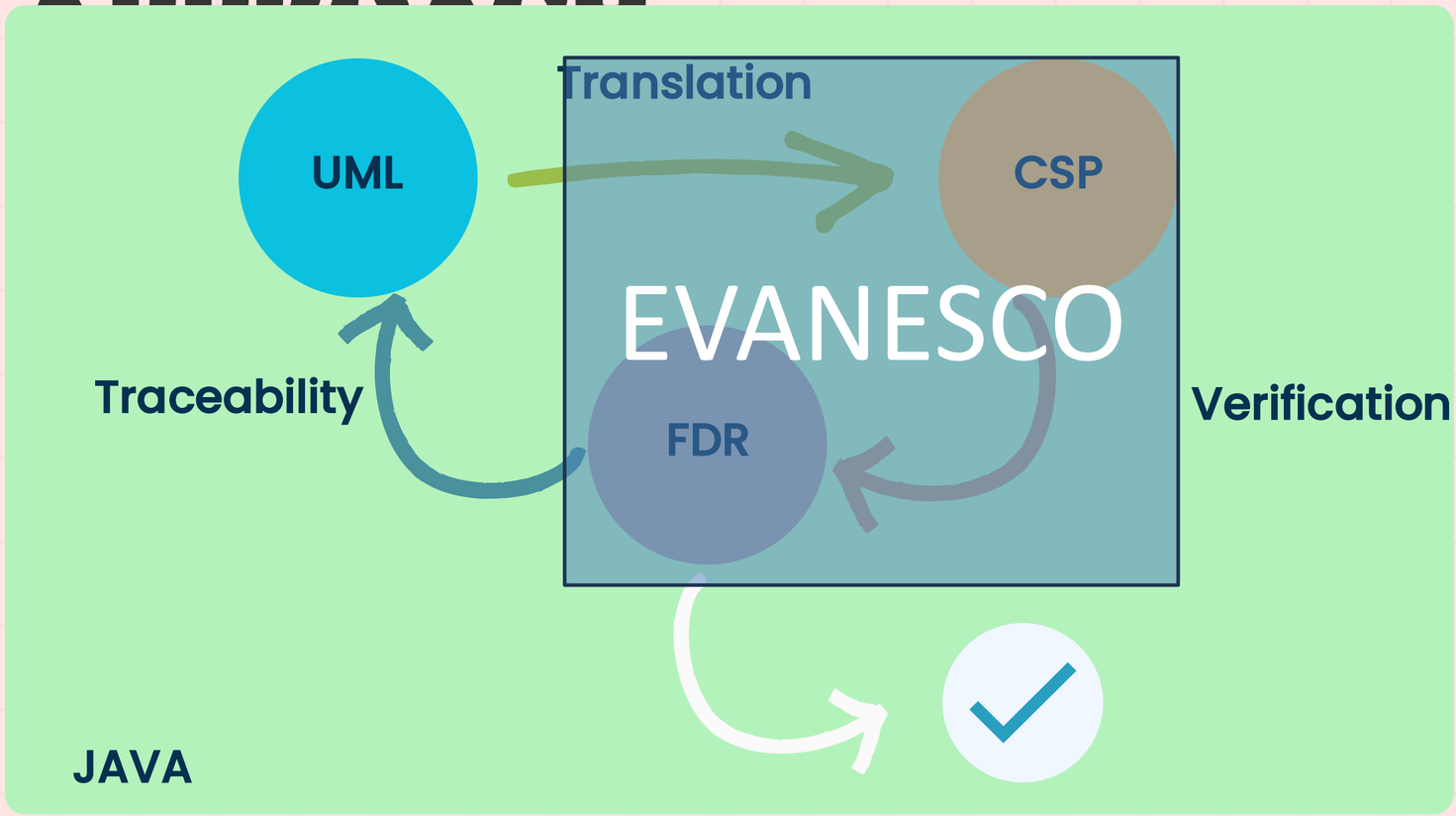


FORMALAN INFORMAL D

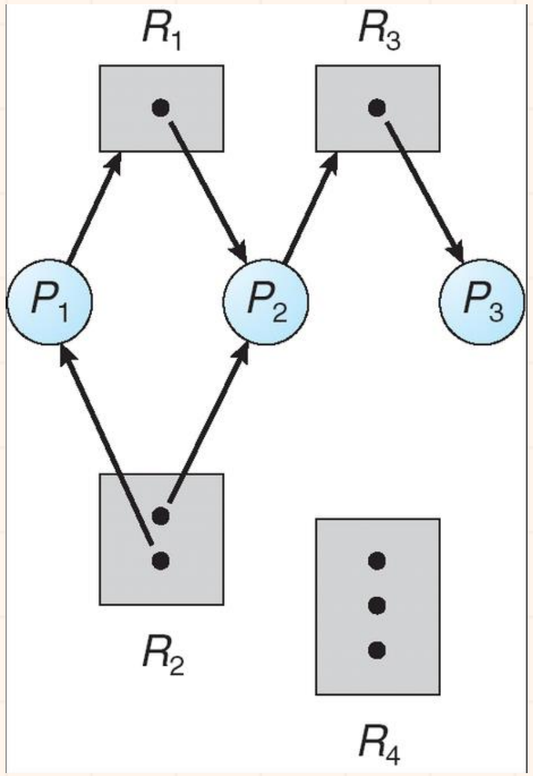
```
transparent wbisim

datatype STATES_ID_flashlight = st_On_flashlight | st_SOSMode_flashlight

channel tr_7hg_51174a4c4cd23019c4adece90e122ab1, tr_8ja_51174a4c4cd2301
channel enter, do, entry, exit, exited: STATES_ID_flashlight
channel final_flashlight
channel internal, turnRecharging, turnOn, turnOff, batteryDepleted, tur
channel turnRecharging_tr_7hg_51174a4c4cd23019c4adece90e122ab1, interna
channel get_battery_flashlight, set_battery_flashlight: {0..100}
channel end, Loop, final_Compound_st_On_flashlight
channel interrupt: STATES_ID_flashlight

LOOP = Loop -> LOOP

max (a, b) = if a < b then b else a
min (a, b) = if a < b then a else b

MEMORY_flashlight(battery) = (end -> SKIP) []
get_battery_flashlight!battery -> MEMORY_flashlight(battery) [] set_bat
 [] (battery > 0) & turnOn_tr_9oh_51174a4c4cd23019c4adece90e122ab1 -> M
 [] (battery < 95) & internal_flashlight_tr_d7e_51174a4c4cd23019c4adece
 [] (battery > 0) & turnOff_tr_el8_51174a4c4cd23019c4adece90e122ab1 -> M
 [] (battery == 0) & batteryDepleted_tr_fvz_51174a4c4cd23019c4adece90e1
 [] (battery > 0) & internal_flashlight_tr_1vjo_cb828eff8f50820142594d1

State_st_On_init_flashlight = State_st_On_flashlight[|{|interrupt.st_On
State_st_On_flashlight = (enter.st_On_flashlight -> State_st_On_Entry_f
State_st_On_Entry_flashlight = EntryProc(st_On_flashlight)
State_st_On_Do_flashlight = (DoProc(st_On_flashlight) [|{|interrupt.st_

State_st_SOSMode_flashlight = (enter.st_SOSMode_flashlight -> State_st_
State_st_SOSMode_Entry_flashlight = EntryProc(st_SOSMode_flashlight)
State_st_SOSMode_Do_flashlight = interrupt.st_SOSMode_flashlight -> SKI

State_st_NormalMode_flashlight = (enter.st_NormalMode_flashlight -> Sta
State_st_NormalMode_Do_flashlight = interrupt.st_NormalMode_flashlight

State_st_Off_flashlight = (enter.st_Off_flashlight -> State_st_Off_Do_f
State_st_Off_Do_flashlight = interrupt.st_Off_flashlight -> SKIP; exit.
```

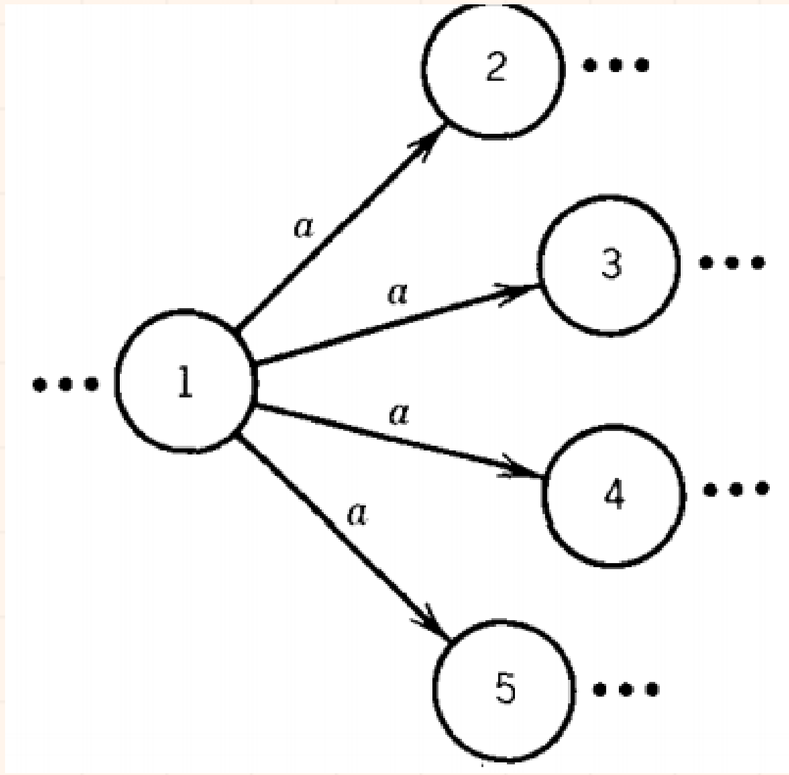


OVERVIEW OF THE APPROACH

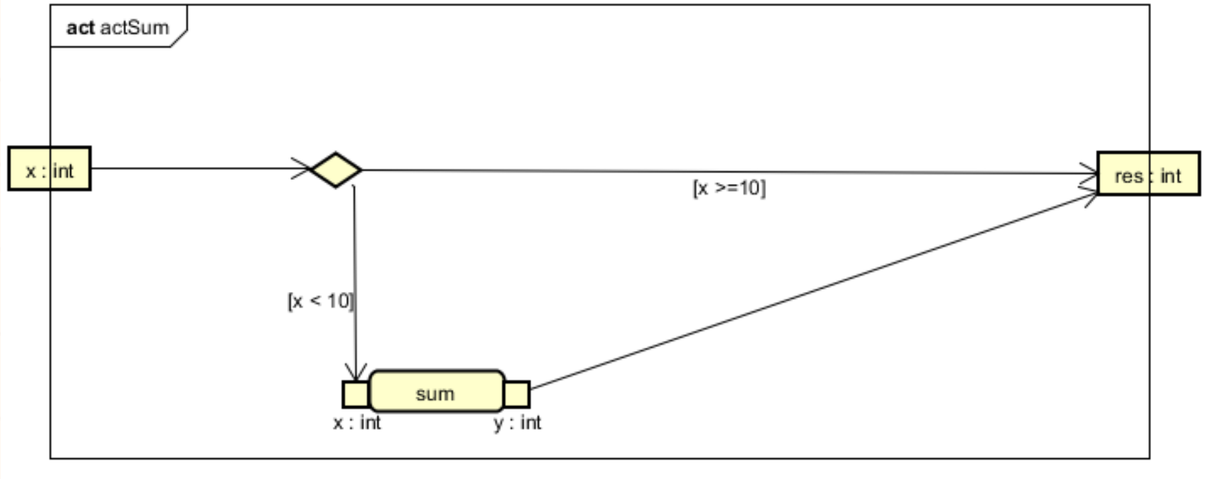


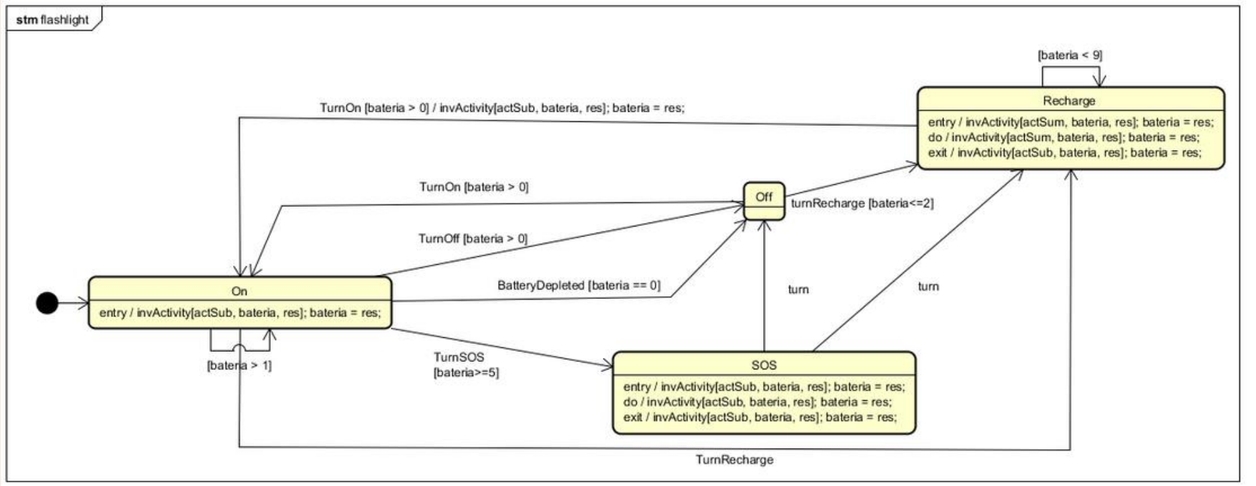
DEADLOCKAN NONDETERMINI D SM



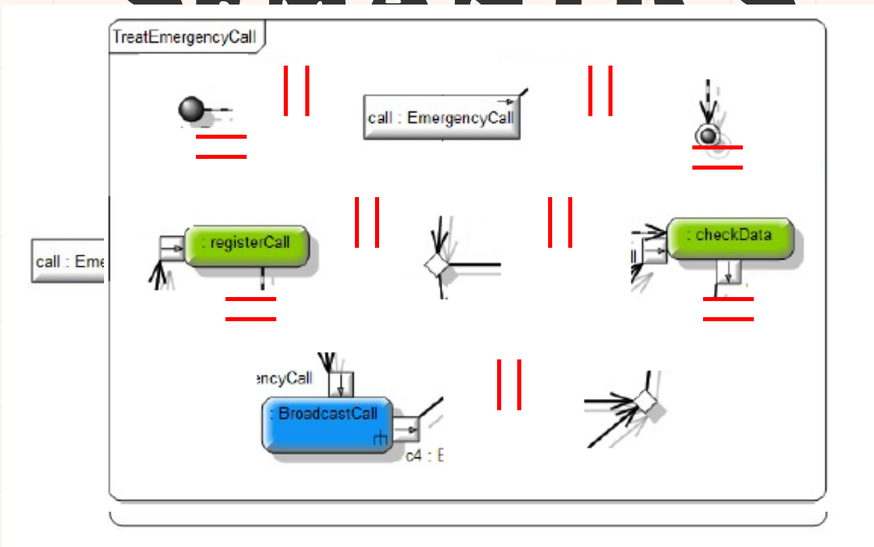


ACTIVITYAN STATE MACHINE D



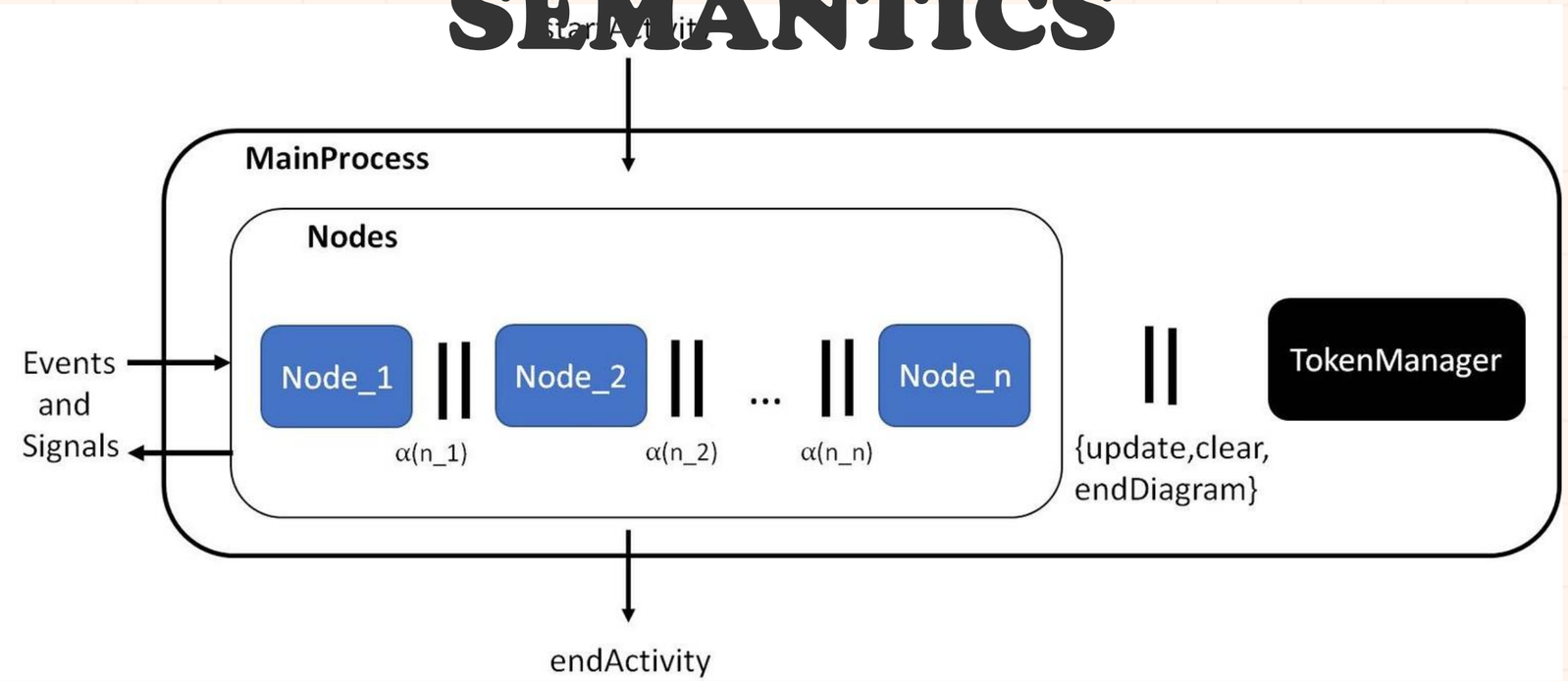


ACTIVITY DIAGRAMS SEMANTICS



Lima, L., Tavares, A., Nogueira, S., A framework for verifying deadlock and nondeterminism in UML activity diagrams based on CSP, Science of Computer Programming, Volume 197, 2020, 102497, ISSN 0167-6423

ACTIVITY DIAGRAMS SEMANTICS



Lima, L., Tavares, A., Nogueira, S., A framework for verifying deadlock and nondeterminism in UML activity diagrams based on CSP, Science of Computer Programming, Volume 197, 2020, 102497, ISSN 0167-6423

UFRPE

SBMF 2024 - 05/12/2024

STATE MACHINE DIAGRAMS SEMANTICS

A CSP semantics for UML state machines aiming at hidden formal methods verification

Diego Ferreira and Lucas Lima

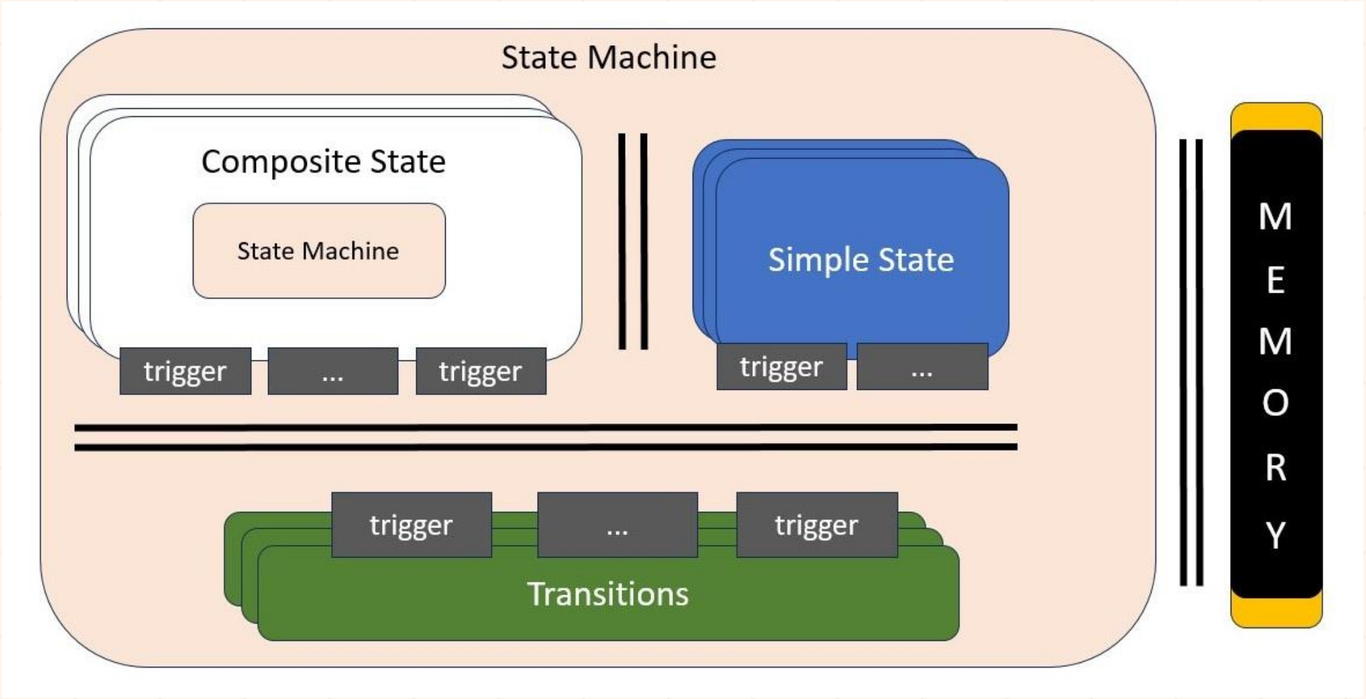
Departamento de Computação, Universidade Federal Rural de Pernambuco, Recife-PE, Brazil
{diego.pires,lucas.albertins}@ufrpe.br

Abstract. The increasing complexity of software systems, especially in safety-critical domains, requires rigorous verification methodologies to ensure reliability and correctness. This paper presents a semantics for UML state machines using the formal language CSP to support automatic property verification. Our approach integrates the intuitive modeling capabilities of UML with the precise verification tooling available for CSP, thus facilitating the detection and correction of design errors at the early stages of system development. We implemented a framework as a plugin for the Astah modeling tool, which translates UML diagrams into CSP specifications and utilizes the FDR model checker for verification. The results are traced back to the diagram level, thus hiding the complexity of formal notations and tools. A case study of a flashlight system demonstrates the practical applicability and benefits of our approach, highlighting its ability to identify and solve design issues early in the development process.

Keywords: state machine · semantic · deadlock · nondeterminism · CSP

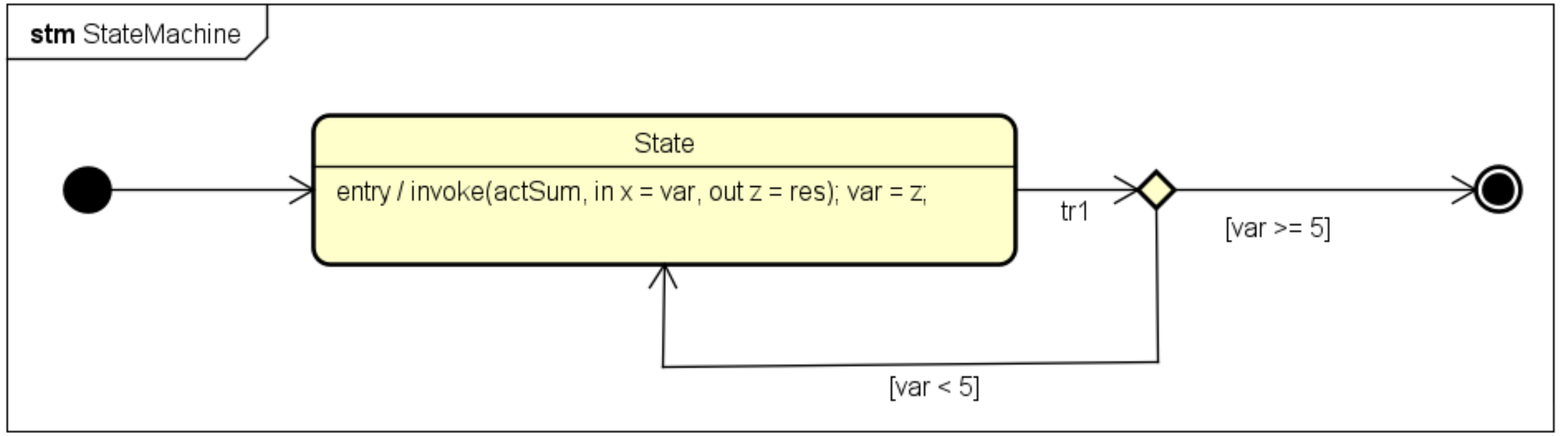
1 Introduction

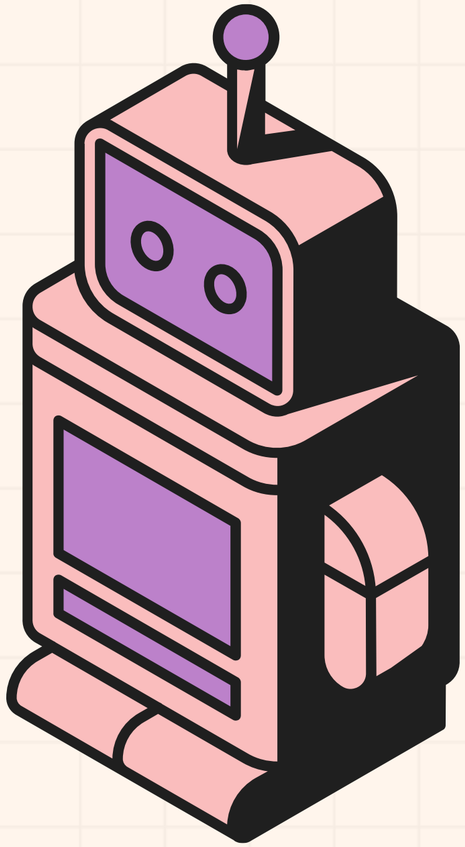
We have recently noticed a paradigm shift from traditional system engineering towards a model-based approach, known as Model-based system engineering (MBSE). Some of the reasons for this include better support for digitalization



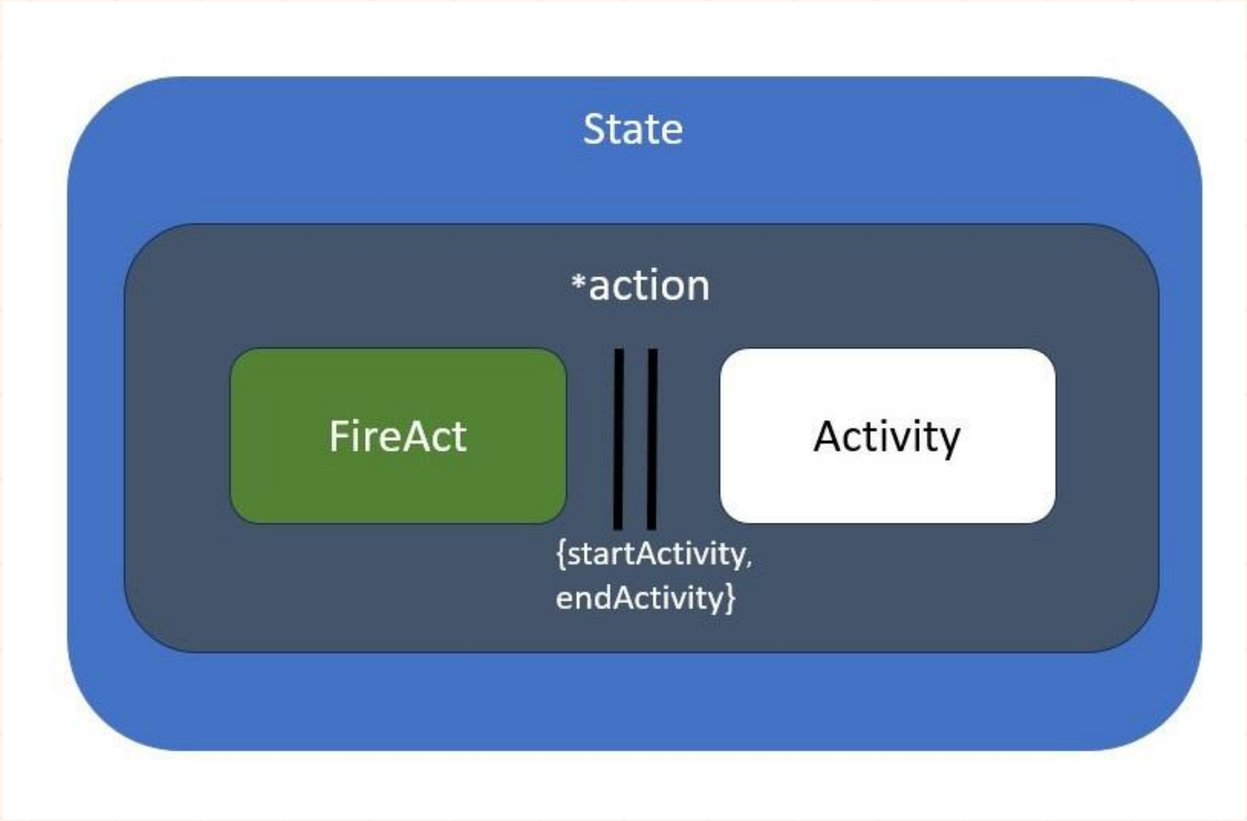
HOW TO INTEGRATE @ UML level

```
invoke(actName, in i1 = q, ..., out j = o1, ...)
```

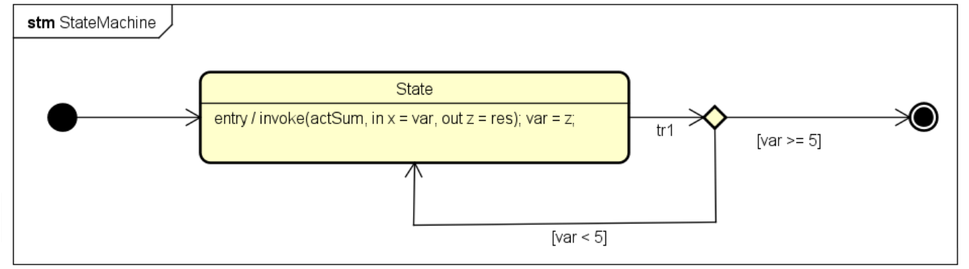




SEMANTIC RELATION



PARSER INTEGRATI



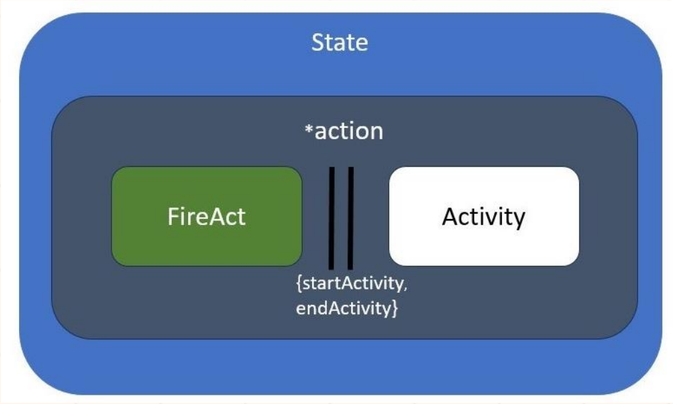
$$EntryProc(st) = entry.st \rightarrow (actSum(id)$$

$$\Vert_{startActivity\_actSum,endActivity\_actSum}$$

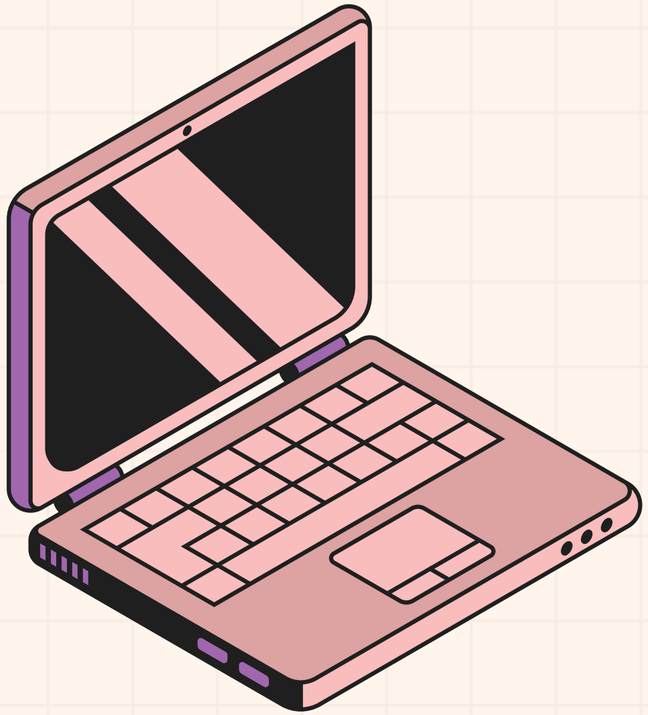
$$FireAct(id)) \; ; State\_st\_State\_Do\_StateMachine$$

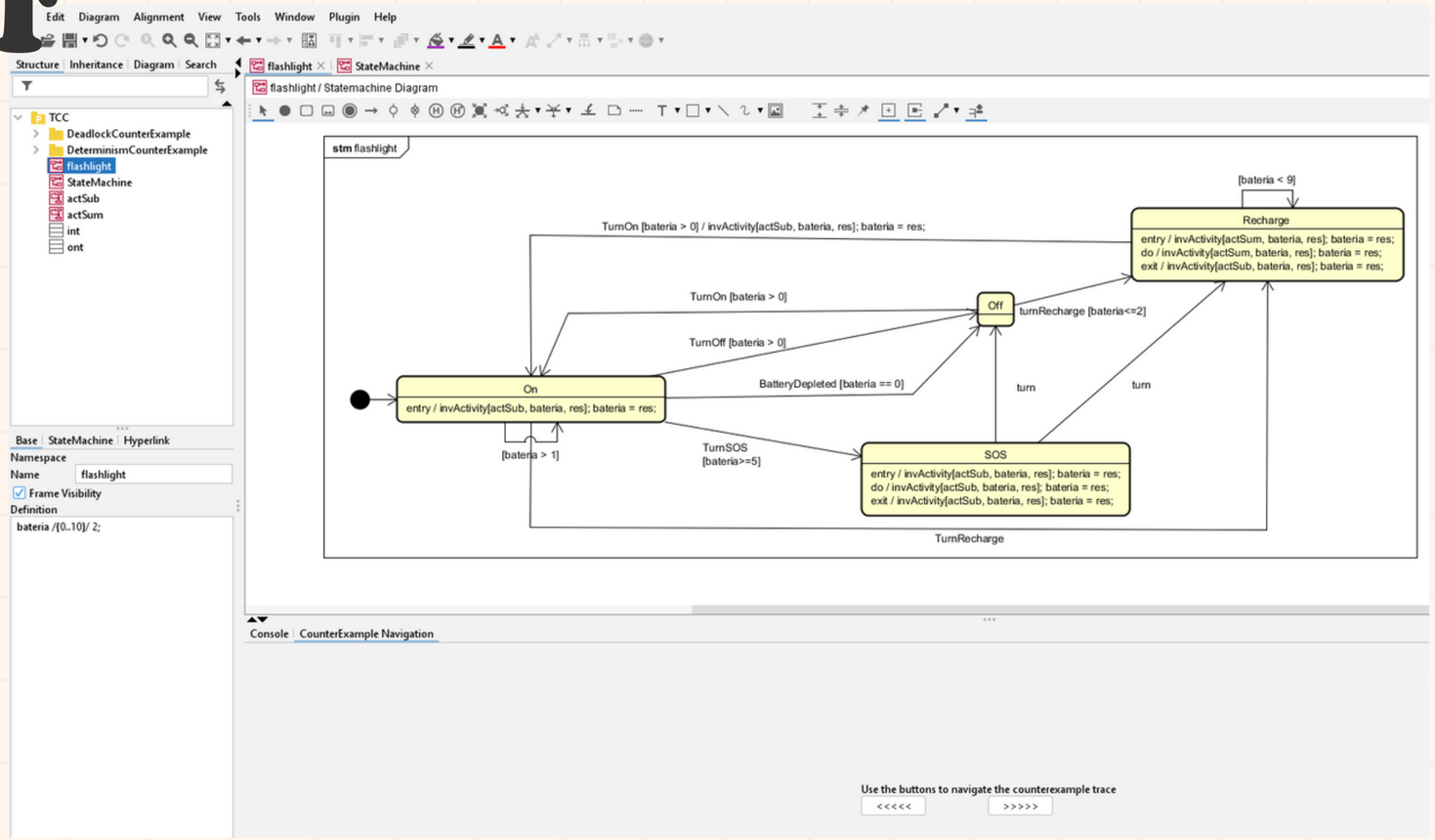
$$FireAct(i) = get\_var?var \rightarrow startActivity.i!var \rightarrow endActivity.i?z \rightarrow$$

$$set\_var!(z) \rightarrow SKIP$$

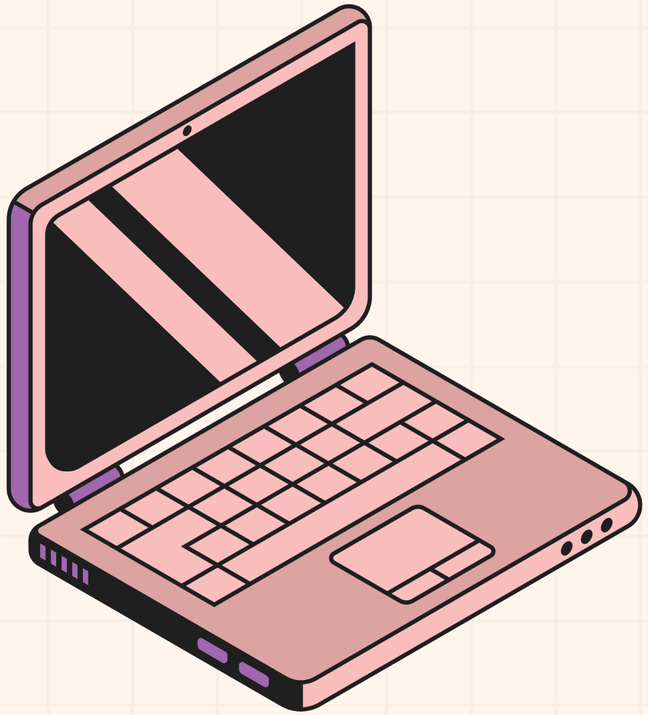


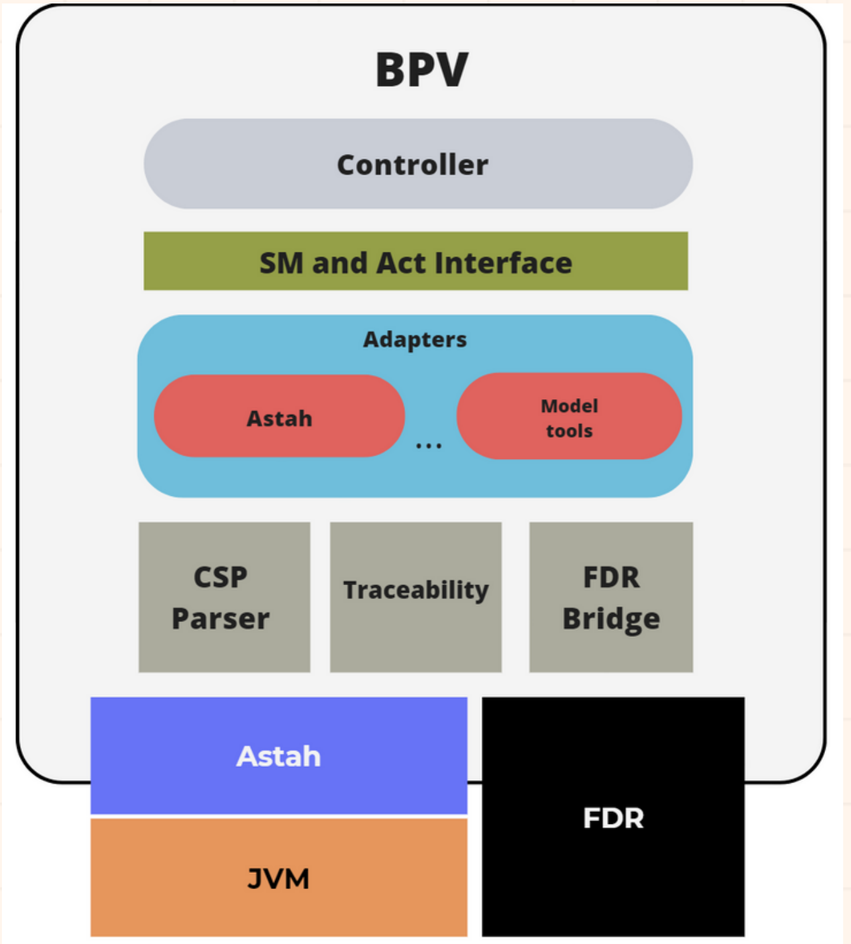
TOOL SUPPORT



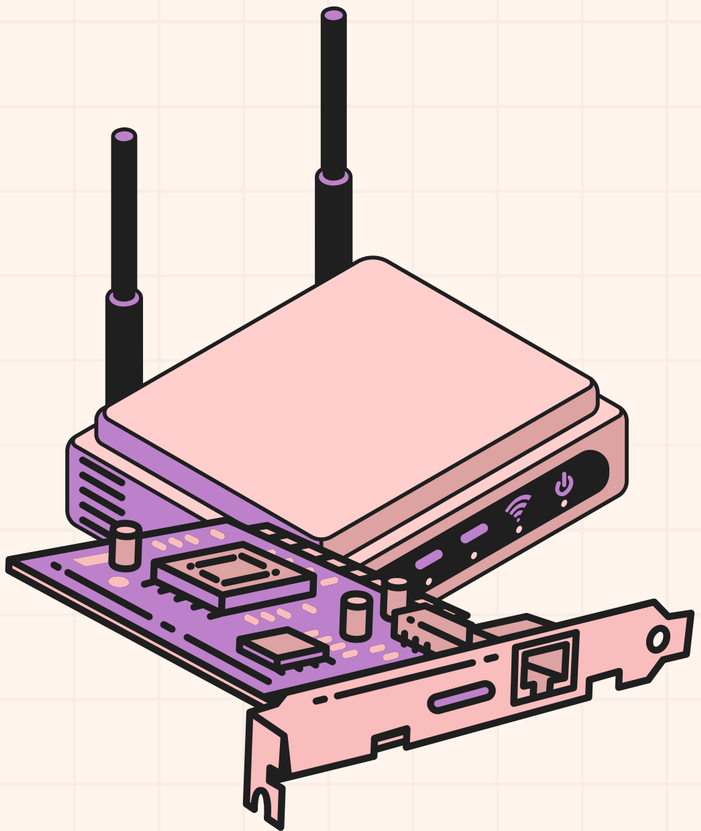


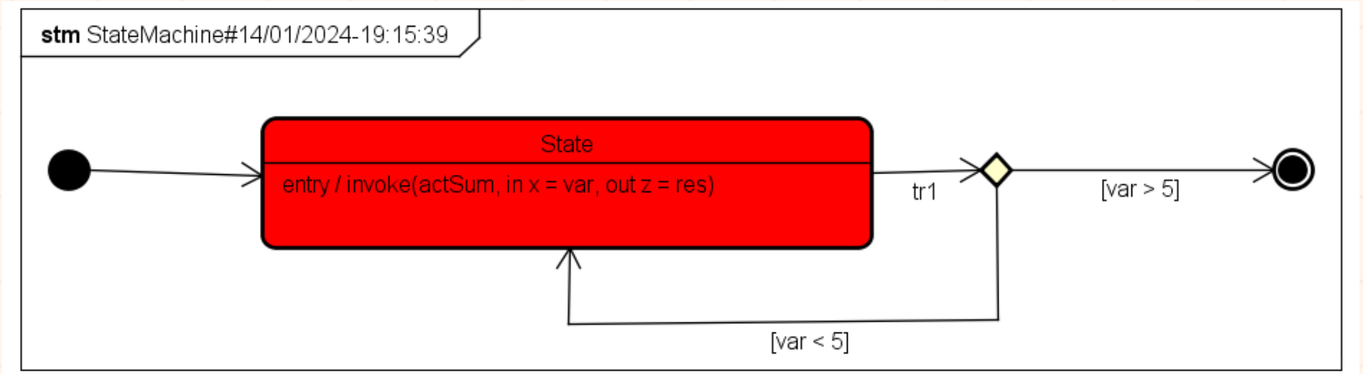
ARCHITECTURE

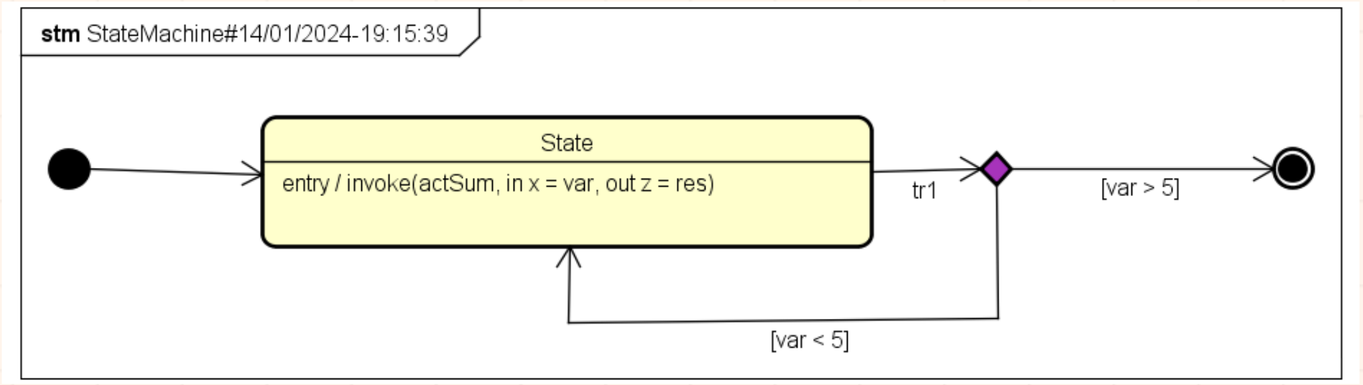




COUNTER EXAMPLE





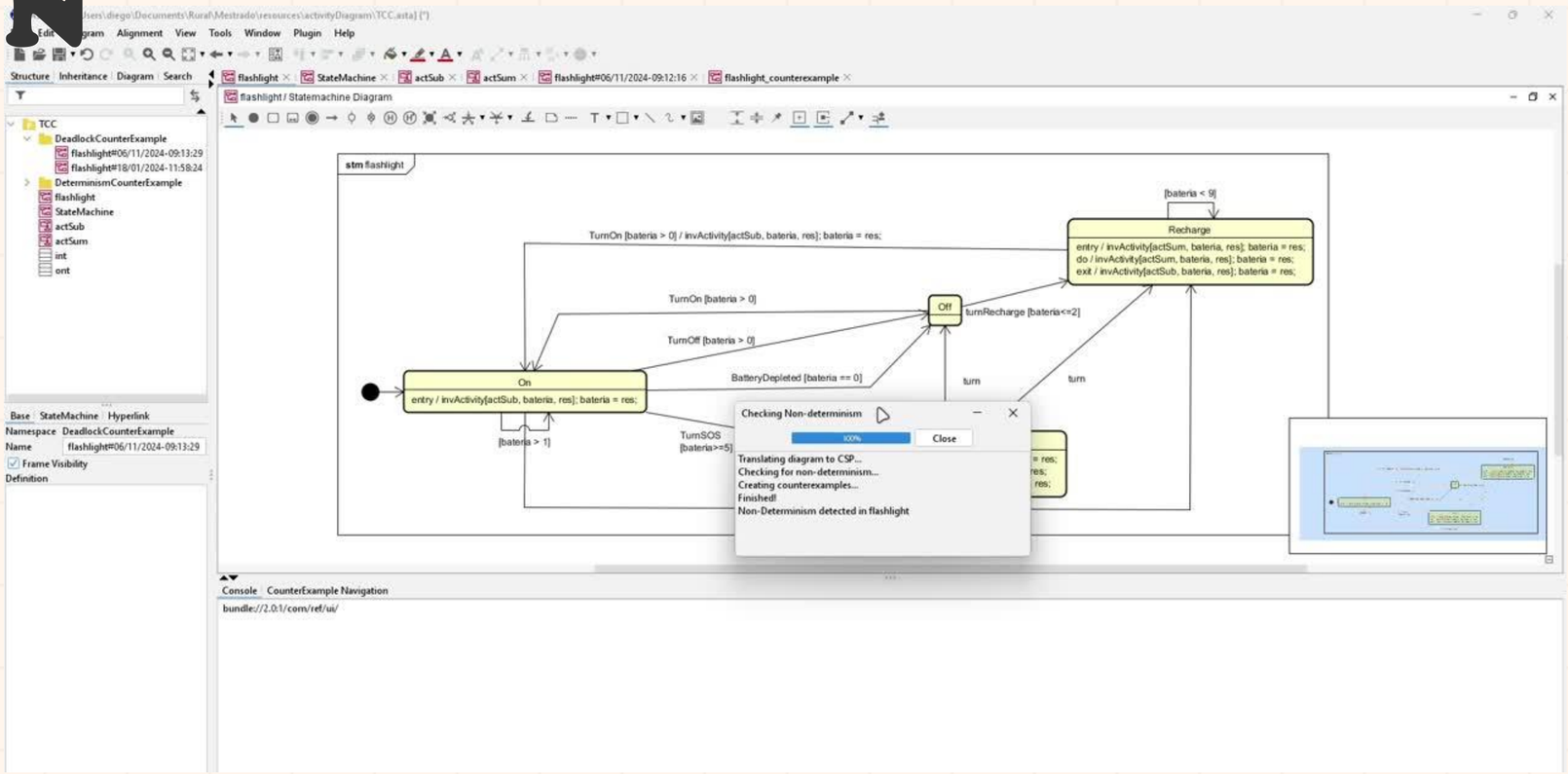


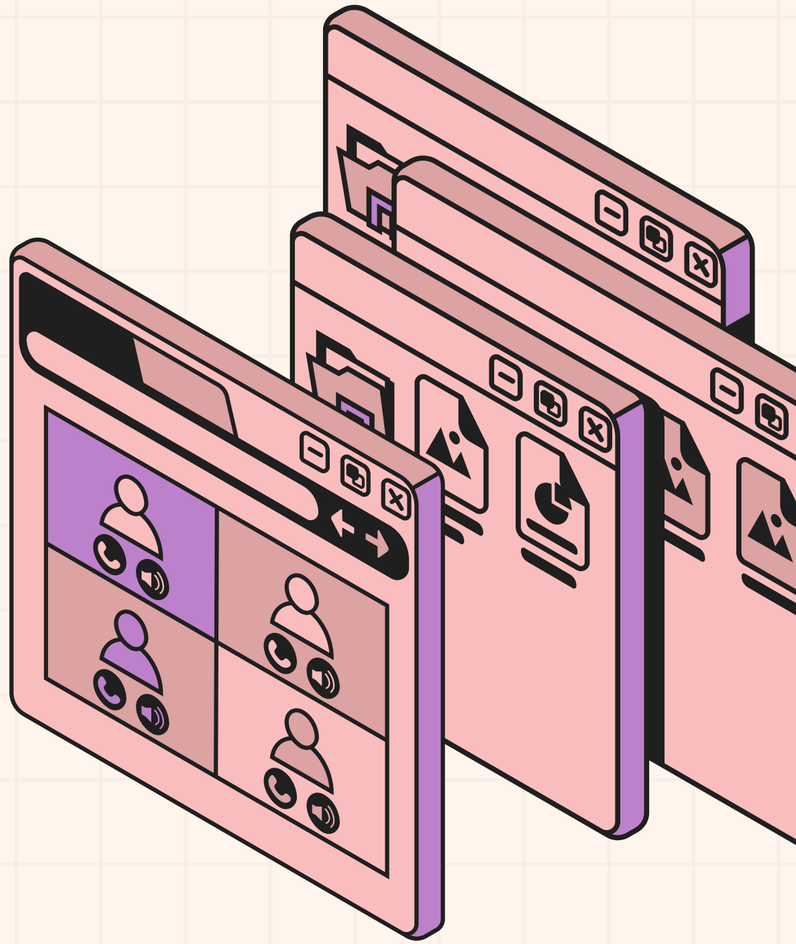
UFRPE

SBMF 2024 - 05/12/2024

DEMONSTRATIO

N





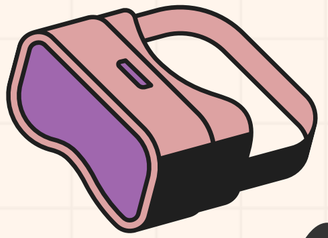
Related Work

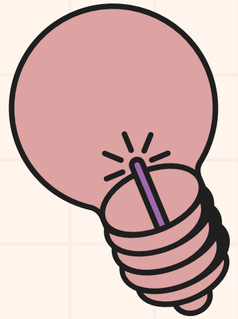
	Verify Activity	Verify State Machines	Intergration	Traceability	Formalism	Purpose
Abdelhalim et al.	✓	✗	✗	✗	CSP	Deadlock
Elmansouri et al.	✓	✗	✗	✗	CSP	Property Verification
Horvath et al.	✓	✓	✓	✓	Gamma	Reachability properties
Kohlmeyer et al.	✓	✓	✓	✗	ASM	Property Verification
Lima et. al	✓	✗	✗	✓	CSP	Deadlock and nondeterminism
Miyazawa et al.	✗	✓	✗	✗	CSP	Deadlock
Our Work	✓	✓	✓	✓	CSP	Deadlock and nondeterminism

Next Steps

- Bidirectional Diagram Integration
- Scalability Analysis
- Expand Model Element Support

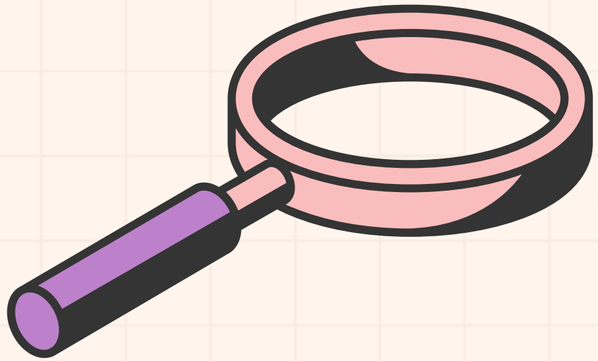
SBMF 2024 - 05/12/2024





THANK YOU

QUESTIONS?



Contact:
diego.pires@ufrpe.br
lucas.albertins@ufrpe.br