An integrated framework for analysing, simulating and testing UML models

Gustavo Carvalho¹, José Dihego², Augusto Sampaio¹ (ghpc@cin.ufpe.br | jose.dihego@ifba.edu.br | acas@cin.ufpe.br)

¹Universidade Federal de Pernambuco Centro de Informática, 50740-560, Brazil

²Instituto Federal de Educação, Ciência e Tecnologia da Bahia, Brazil





An integrated framework for UML models

Motivation

Context: UML as a notation for modelling object-oriented software systems

Goal: enable analyses supported by formal methods

Challenges:

- formal semantics (e.g., fUML, PSSM, PSCS)
- integration between structural and behavioural models (i.e., a unified semantics)





Contributions

Focus: behavioural (B) and structural (S) models

- B = state machine diagrams
- S = composite structure diagrams
- Integration with the NAT2TEST strategy
- Main contributions:
 - Systematic and compositional process
 - Validation of translation based on CSP
 - Tool support along with the NAT2TEST tool
 - Case studies:
 - dining philosophers
 - ring-buffer model

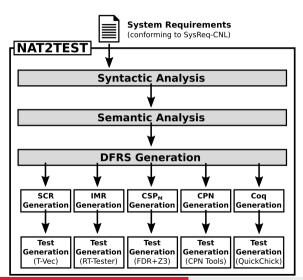




An integrated framework for UML models

December 5, 2024

NAT2TEST strategy



SysReq-CNL: guarded actions



Carvalho, Dihego, and Sampaio

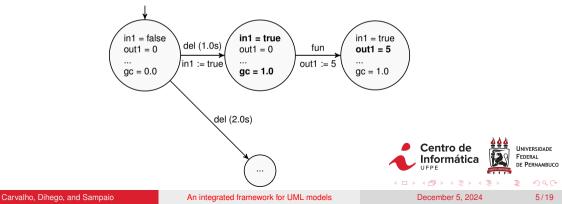
An integrated framework for UML models

December 5, 2024

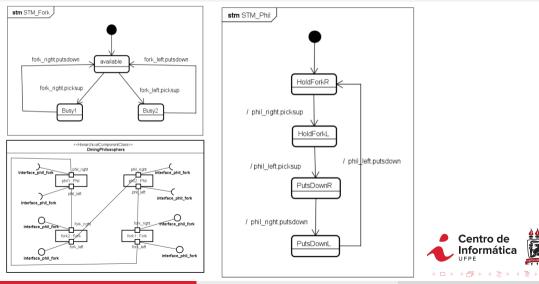
DFRSs: Data-flow Reactive Systems

s-DFRS: symbolic DFRS = $(I, O, T, gcvar, s_0, F)$ **•** $F = \{f_1, f_2, ...\}$, s.t. $f_1 = \{(static_guard_1, timed_guard_1) \mapsto \{action_1, action_2, ...\}, ...\}$ e-DFRS: expanded DFRS = $(I, O, T, gcvar, s_0, S, TR)$

del = delay transitions | fun = function transitions



Running example: dining philosophers



Carvalho, Dihego, and Sampaio

An integrated framework for UML models

December 5, 2024

6/19

UNIVERSIDADE

DE PERNAMBUCO

FEDERAL

3

From UML diagrams to CNL requirements: overview

DFRSs: communication via shared memory (events represented using variables)

Translation follows three phases¹

- SM: from transitions to sentences (conditions → guards, effect → actions)
- 2 CS: sentences replicated based on the number of instances (STM_Fork_state ~> STM_Fork_statefork1, ...)
- CS: sentences updated according to connections (sync.: one waits for the change, other performs the change)

Mapping rules: presented in the paper

¹SM = state machines, CS = composite structure

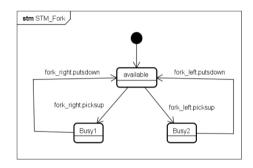


From UML diagrams to CNL requirements: phase 1

An integrated framework for UML models

Phase 1 – SM: from transitions to sentences

```
When the STM_Fork_state is available,
and the STM_Fork timer is greater than 0,
the STM_Fork_component shall:
assign true to the fork_right_picksup,
assign Busyl to the STM_Fork_state,
reset the STM_Fork timer.
```





From UML diagrams to CNL requirements: phase 2

Phase 2 – CS: sentences replicated based on the number of instances

```
When the STM_Fork_statefork1 is available, ...
the STM_Fork_componentfork1 shall:
assign true to the fork_right_picksupfork1,
assign Busy1 to the STM_Fork_statefork1, ...
```

When the **STM_Fork_statefork2** is available, ... the **STM_Fork_componentfork2** shall: assign true to the **fork_right_picksupfork2**, assign Busy1 to the **STM_Fork_statefork2**, ...



9/19

December 5, 2024

From UML diagrams to CNL requirements: phase 3

Phase 3 – CS: sentences updated according to connections

When the STM_Phil_statephil1 is HoldForkR, ..., and the STM_Fork_statefork1 is available, the STM_Phil_componentphil1 shall: assign true to the

phil_right_picksupphil1__fork_left_picksupfork1,

When the STM_Fork_statefork1 is available, ..., and the phil_right_picksupphil1__fork_left_picksupfork1 becomes true,

the STM_Fork_componentfork1 shall: assign false to the
 phil_right_picksupphil1__fork_left_picksupfork1, ...



From UML diagrams to CNL requirements: validation

Validation of translation based on a CSP semantics

- S_NAT2TEST: our semantics (derived by the NAT2TEST strategy)
- S_UML: obtained independently from the same diagrams [FLS+24]
- S_Classical: classical semantics written by Roscoe



From UML diagrams to CNL requirements: validation

S_NAT2TEST

```
SPECIFICATION(memory) = FUN(...)
SPECIFICATION2(memory) = ... delayTransition ...
SPECIFICATION3(memory) = INPUTS(memory)
SPECIFICATION4(memory) = SPECIFICATION(memory)
SYSTEM = SPECIFICATION(seq(initialBinding))
SYSTEM' = SYSTEM [[ ... <- ... ]]
S' = SYSTEM' \ {| ... |}</pre>
```

```
transparent sbisim, diamond
sbdia(P) = sbisim(diamond(P))
S'' = (sbdia(S'))
```

S_NAT2TEST = S'' S_NAT2TEST' = S_NAT2TEST \ { ... }

S_Classical

```
PHILS = ||| i:PHILNAMES @ PHIL(i)
FORKS = ||| i:FORKNAMES @ FORK(i)
PHILS_SYSTEM = PHILS [| {|picks, putsdown|} |] FORKS
```

S_Classical = PHILS_SYSTEM

```
S_Classical' = S_Classical [[ ... <- ...]] \ { ... }</pre>
```

Carvalho, Dihego, and Sampaio

S_UML

An integrated framework for UML models

```
...
processcomp =
  (inter_inter_inter_fork2_phil2_phil1_fork1)
  [| ... |] BFIO_INIT(fork_left.2, phil_right.1)
processcomp_com = processcomp
  [| ... |] BFIO_INIT(fork_right.2, phil_left.2)
processcomp_com_com = processcomp_com
  [| ... |] BFIO_INIT(fork_right.1, phil_left.1)
```

```
processcomp_com_com = processcomp_com_com
[| ... |] BFIO_INIT(fork_left.1,phil_right.2)
```

S_UML = processcomp_com_com S_UML' = S_UML [[... <- ...]] \ { ... }</pre>



From UML diagrams to CNL requirements: validation

We prove (via FDR) that:

```
assert S_NAT2TEST' [T= S_UML'
assert S_UML' [T= S_NAT2TEST'
```

```
assert S_NAT2TEST' [T= S_Classical'
assert S_Classical' [T= S_NAT2TEST'
```

All models have precisely the same trace semantics



Carvalho, Dihego, and Sampaio

Tool support

JavaScript code: obtain textual representation of AstahUML models

```
machine STM Fork
state Available ...
transition Available fork right.picksup Busy1 ...
```

Python scripts: implement the mapping rules in a very direct way

Uses text templates

```
When the {1}_state is {2},
  and the \{3\} timer is greater than 0,
the {4} component shall:
  assign true to the {5},
  assign {6} to the {7} state,
  reset the {8} timer.
```



December 5, 2024

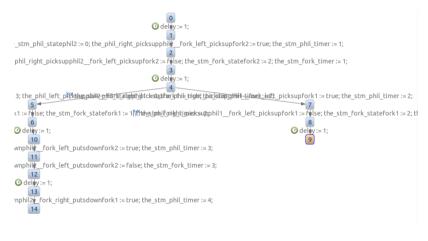
Tool support provided by the NAT2TEST strategy

NAT2TEST - From Natural Language Requirements To Test Cases _ a								
File Project Window Help								
11 12 11 1 1 A 11 12 13 14 12 10 14	a [剑 + 刮 +] 中 (中	12						
5 Navigation View SS 👘 🗈	Gil View All Requirements H+ Variables and Types If' Functions H: Animation 18							
* @PHLS	Layout: Crid Waximum of nodes:	Update Style						
General Info								
* III Requirements	Enabled Transitions							
Aliases								
RECOO2	@ delay := 1;							
RECO03	9 Delay DEL: +? time units; ?:=?;							
REQ004	som pri statepritz := 0; the pris sprice proved prior set set care the som pri curve := 1;							
REQ005	phil right, picksupphil2_fork_left_picksupfork2:= foice; the stm_fork_statefork2:= 2; the stm_fork_timer:= 1;							
RECOOD								
B REQ007	O de v >= 1;							
In REGOOD	3: the phil left offshing plate of the state							
III REQ009	×							
In REQ010	k1 := Twise; the_stm_fork_statefork1 := 1fff#eggelife@ightgi@elege							
B REQ011	O dollar (= 1) O dollar (= 1)							
B REQ012	Source Type Transition	Target						
EQ013	wrphing_fork_left_pubsdownfork2:=true; the_stm_phil_timer:=3; 0 Delay DEL:+1 time units;	1						
REQ014	wrohim fork. left: putsdownfork2 := false; the stm_fork_timer := 3; 1 Function _FUN: the phil_right_picksup;	2						
REQ016	2 Punction PUN: the phil right picksup;	3						
Dictionary	⊘ delay:= 1; 3 Delay DEL: +1 time units;	4						
* Z Data-flow Reactive System	11 4 Function FUN: the phil_left_picksuppl	5						
M-Variables and Types	mph/by_fork_night_put/downfork1:=true;the_stm_ph/Ltimer:=4; 14 14 14 14 15 14 14 14 14 14 14 14 14 14 14	6						
1/ Functions	- Function Force print procession	7						
#Animation	7 Punction FUN: the_phil_right_picksup;	8						
SCR .								
+ le Coq	Console 04-Variables Instances 31							
Coq Test Cases	State: 9 (15 variables)							
+ E CSPm	Kind Name Type Value							
Test Cases	OUTPUT the phil right putsdownphili_fork_left_putsdownfork1 BOOLEAN false							
(C) C2PW	OUTPUT the_stm_phl_statephl2 INTEGER 0							
	OUTPUT the_stm_pNi_statepNi1 INTEGER 0							
	OUTPUT the_phil_right_picksupphil_fork_left_picksupfork1 BOOLEAN false							
	OUTPUT the phil right picksupphil2_fork.left.picksupfork2 BOOLEAN faise OUTPUT the phil left.picksupphil2_fork.right.picksupfork1 BOOLEAN faise							
	OUTPUT the phillert_pickuppeni2_fork_pickupprox1 BOOLEAN faile OUTPUT the phillert_pickuppeni2_fork_pickupprox1 BOOLEAN faile							
	OUTPUT the gen (with pussion/pent_ore_gas_asson/pent/ OUTPUT the stm. fork.statefork2 INTEGR 2							
	OUTFUT the sm fork statefork1 NTEGR 2							



Universidade Federal de Pernambuco

Tool support provided by the NAT2TEST strategy





An integrated framework for UML models

December 5, 2024

Related work

	B. diagr.	S. diagr.	Simul.	Testing	Verif.	Code gen.
CA21	SM	BD, IB	•	0	•	0
LMC ⁺ 17	Ac, SM, Sq	BD, IB	0	0	•	0
HMG ⁺ 23	Ac, SM, Sq	BD, IB	•	•	•	•
EMM24	Ac, SM, Sq	—	•	0	0	0
FLS ⁺ 24	SM	CI, CS	•	0	•	0
Ours	SM	CS	•	•	•	0

CA21: Direct Model-checking of SysML Models.

- LMC⁺17: An integrated semantics for reasoning about SysML design models using refinement.
- HMG⁺23: Pragmatic verification and validation of industrial executable SysML models.
- EMM24: To Do or Not to Do: Semantics and Patterns for Do Activities in UML PSSM State Machines.
- FLS+24: A formal component model for UML based on CSP aiming at compositional verification.



Conclusion

Analysing, simulating, and testing UML models

- Combination of behavioural + structural diagrams
- Enabled by the NAT2TEST strategy

Future work:

- Complex constructs
- More UML diagrams
- Additional case studies
- Further explore the NAT2TEST strategy



An integrated framework for analysing, simulating and testing UML models

Gustavo Carvalho¹, José Dihego², Augusto Sampaio¹ (ghpc@cin.ufpe.br | jose.dihego@ifba.edu.br | acas@cin.ufpe.br)

¹Universidade Federal de Pernambuco Centro de Informática, 50740-560, Brazil

²Instituto Federal de Educação, Ciência e Tecnologia da Bahia, Brazil



