Modeling of Natural Language Requirements based on States and Modes

Postdoc: Yinling LIU[1]
Supervisor: Jean-Michel BRUEL[1]
[1]SM@RT, IRIT, University of Toulouse

August 16th, 2022

SM@RT

Content

Conflicts in companies about modes

SMARES ENGINEERING:

- conflicts with clients;
- conflicts among engineers;
- ...

Conflicts in companies about modes

SMARES ENGINEERING:

- conflicts with clients;
- conflicts among engineers;
- ...

A pragmatic approach to detecting modes in requirements

An approach based on tables

			Phase	Pre-mission			Mission							Post-mission
			Mode	WAIT	LOAD	READINESS	FLY		WORK			FLY		UNLOAD
			State	Flight plan waiting	Flight plan loading	Ready to work	Flying to destination		Working			Flying to base		Off
							Localising	Thrust generating	Camera controlling	Records managing	Records sending	Localising	Thrusting generating	
Function				✅										✖
Manage HMI				✅										
Sense Wind Force				✅		✅				✅				✖
Manage Mission	Manage Mission Modes			OK	OK	✅	✅					✅		
	Retrieve POI				✅	✅								✖
Build Flight Plans						OK	✅					✅		
Fly to Position	Control UAV Altitude					OK	✅					✅		✖
	Control UAV Position							✅					✅	
	Manage Thrust	Generate Thrust						✅					✅	✖
		Compute Thrust					OK					OK		
Monitor UAV Control							✅					✅		✖
Sense	Perceived Position													
	Acquire Altitude Position								✅	✅	✅			
Make Photos and Record videos	Manage Recording								✅	OK				✖
	Control Camera Orientation									OK	✅			
	Send Records to DB									OK	✅			✖
	Record Photos									OK	✅			
	Record videos										OK			✖
Identify Defects														

Our objective

How to formally verify the requirements based on states and modes?

What are states and modes ?

Wasson's definitions on states and modes:

State

States are observable and measurable physical attributes of a SYSTEM and ENTITY.

For example, the state *Idle* of a washing machine : Temperature = 0, PreWash = False, SpinningSpeed = 0, ...

What are states and modes ?

Wasson's definitions on states and modes:

State

States are observable and measurable physical attributes of a SYSTEM and ENTITY.

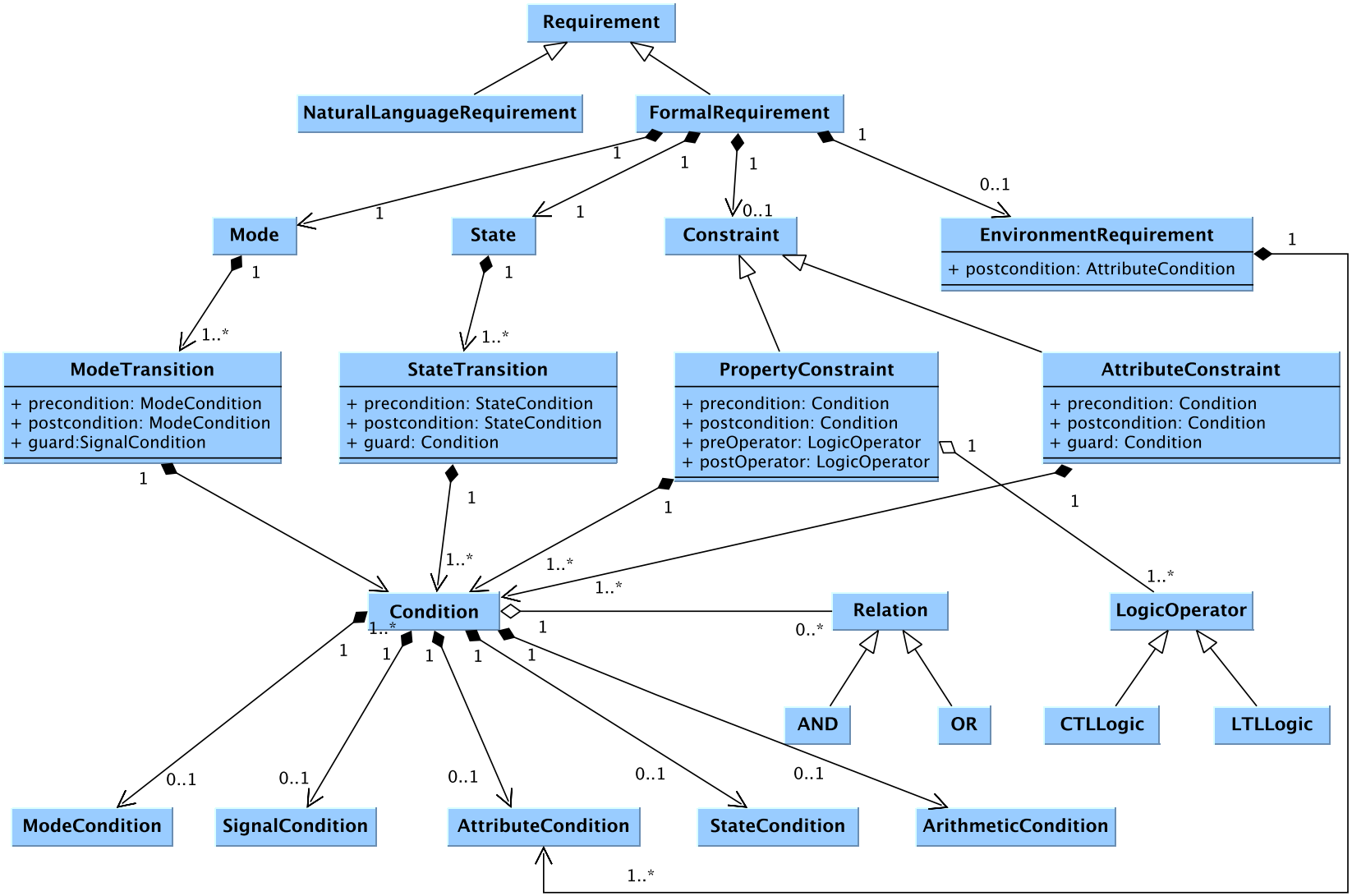
For example, the state *Idle* of a washing machine : Temperature = 0, PreWash = False, SpinningSpeed = 0, . . .

Mode

Modes are abstract labels applied by System Developers to User selectable options required to command and control an Enterprise or Engineered System.

For example, for a washing machine, if the system is in mode *Jeans*, we configure the system : Temperature = 40, PreWash = False, SpinningSpeed = 800.

The MoSt (Modes and States) meta model



MoSt validator

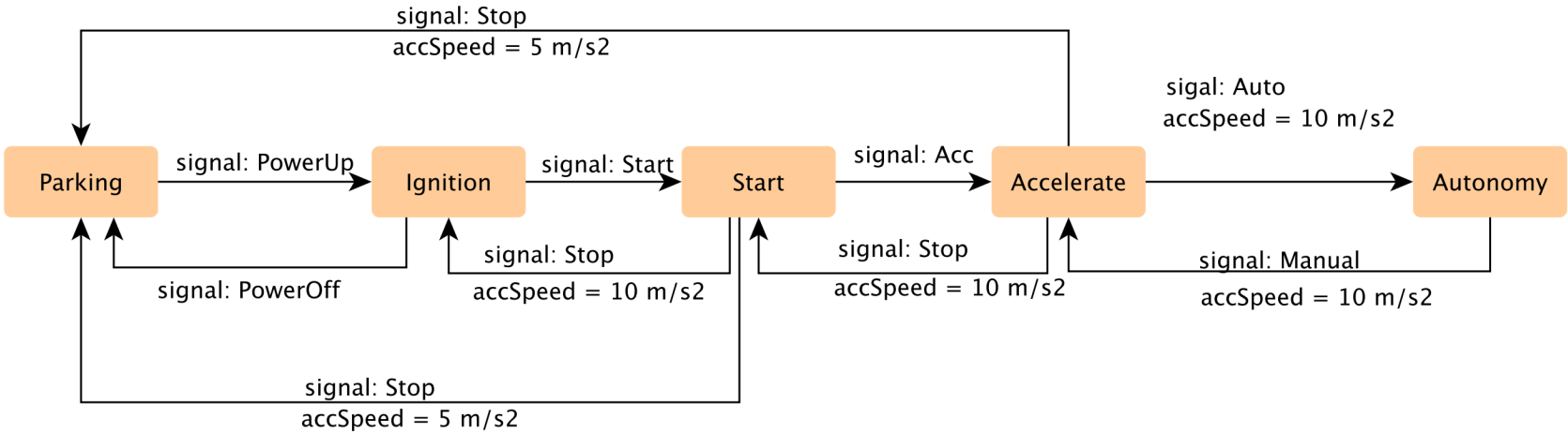
Naming checks (NC):

1. State names should start with a lower case;
2. Signal names should start with an upper case;
3. ...

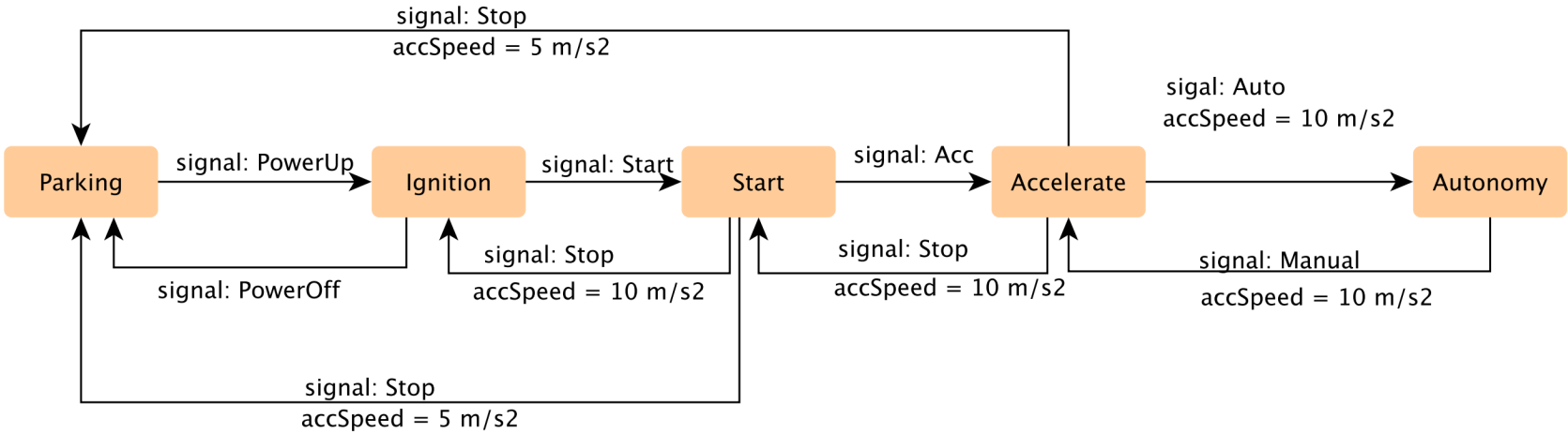
Consistency checks:

1. Non-integer variables should only be initialised once;
2. The variables mentioned in attribute requirements should be initialized;
3. The variable of *Integer* should be given the scope;
4. The repetition of requirement IDs is not allowed;
5. The repetition of requirements is not allowed;
6. Different post-conditions of one type of requirements cannot have the same preconditions.
7. The variable of attribute conditions must be defined before using it.
8. ...

A car example



A car example



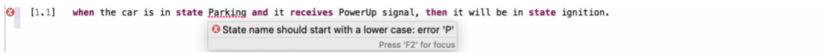
The examples for states, attributes, modes, and property requirements, which are illustrated as follows:

- State transition: [1.1] when the car is in state parking and it receives PowerUp signal, then it will be in state ignition.
- Attribute declaration: [2.1.1] The speed should be initialised to 0 km/h.
- Mode transition: [6.1] when the car is in mode sportive and it receives DeAC signal and its speed is greater than 40 km/h, then it is in mode economic.
- Property declaration: [7.1] when all globally the car is state autonomy and it is in mode economic, then all next it is not in state accelerate.

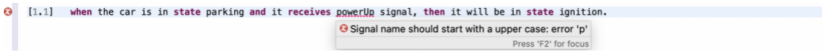
Static checks results

Naming Check (NC)

NC1:

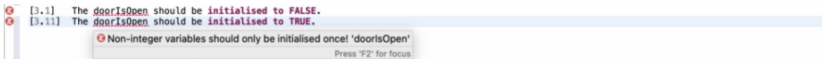


NC2:



Consistency Check (CC)

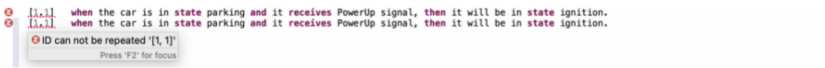
CC1:



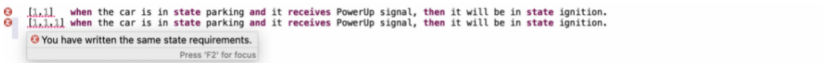
CC2:

CC3:

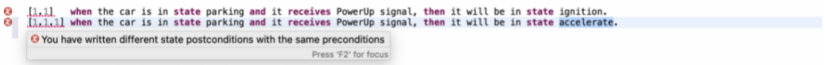
CC4:



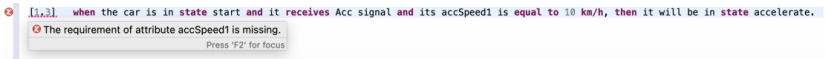
CC5:



CC6:



CC7:



Static checks results

Naming Check (NC)

NC1:

```
[1.1]  when the car is in state Parking and it receives PowerUp signal, then it will be in state ignition.
```

State name should start with a lower case: error 'P'

Press 'F2' for focus

NC2:

```
[1.1]  when the car is in state parking and it receives powerUp signal, then it will be in state ignition.
```

Signal name should start with a upper case: error 'p'

Press 'F2' for focus

Consistency Check (CC)

CC1:

```
[3.1]  The doorIsOpen should be initialised to FALSE.
[3.11] The doorIsOpen should be initialised to TRUE.
```

Non-integer variables should only be initialised once! 'doorIsOpen'

Press 'F2' for focus

CC2:

```
[2.2.3] when the car is mode economic, then its accSpeed is equal to 5 m/s2.
```

You have not initialised this variable

Press 'F2' for focus

CC3:

```
[2.2.3] when the car is mode economic, then its accSpeed is equal to 5 m/s2.
[2.2.1] The accSpeed should be initialised to 0 m/s2.
```

Scope should be given to environment variables 'accSpeed'

Press 'F2' for focus

Conclusions and future work

Conclusions

- MoSt meta model;
- MoSt validator.

Future work

- transforming MoSt models into formal models;
- automating the formal analysis.

Q & A

THANK YOU FOR YOUR ATTENTION!