ONERA
THE FRENCH AEROSPACE LAB

# lil : A formal language for specification of safety-critical UIs

## Partial example application : A car speed controller UI

The LIL Interaction Language (LIL) is a domain-specific language (DSL) which aims at enhancing the design process of safety-critical user interfaces (UIs). LIL is a human-readable textual language, and has formal semantics. The ambition of LIL is to be a *lingua franca* between stakeholders of the critical UI specification process. This poster describes a partial example of the design process of a LIL application, along with the corresponding **`LIL code`** and the main concepts of LIL.

LIL enable specification of abstract user interfaces and their behavior. The concrete implementation is not part of the LIL language. So LIL can virtually specify any kind of modality and interaction techniques at an abstract level.

LIL is an ongoing work, and its syntax and semantics was inspired by different existing languages and prior work.

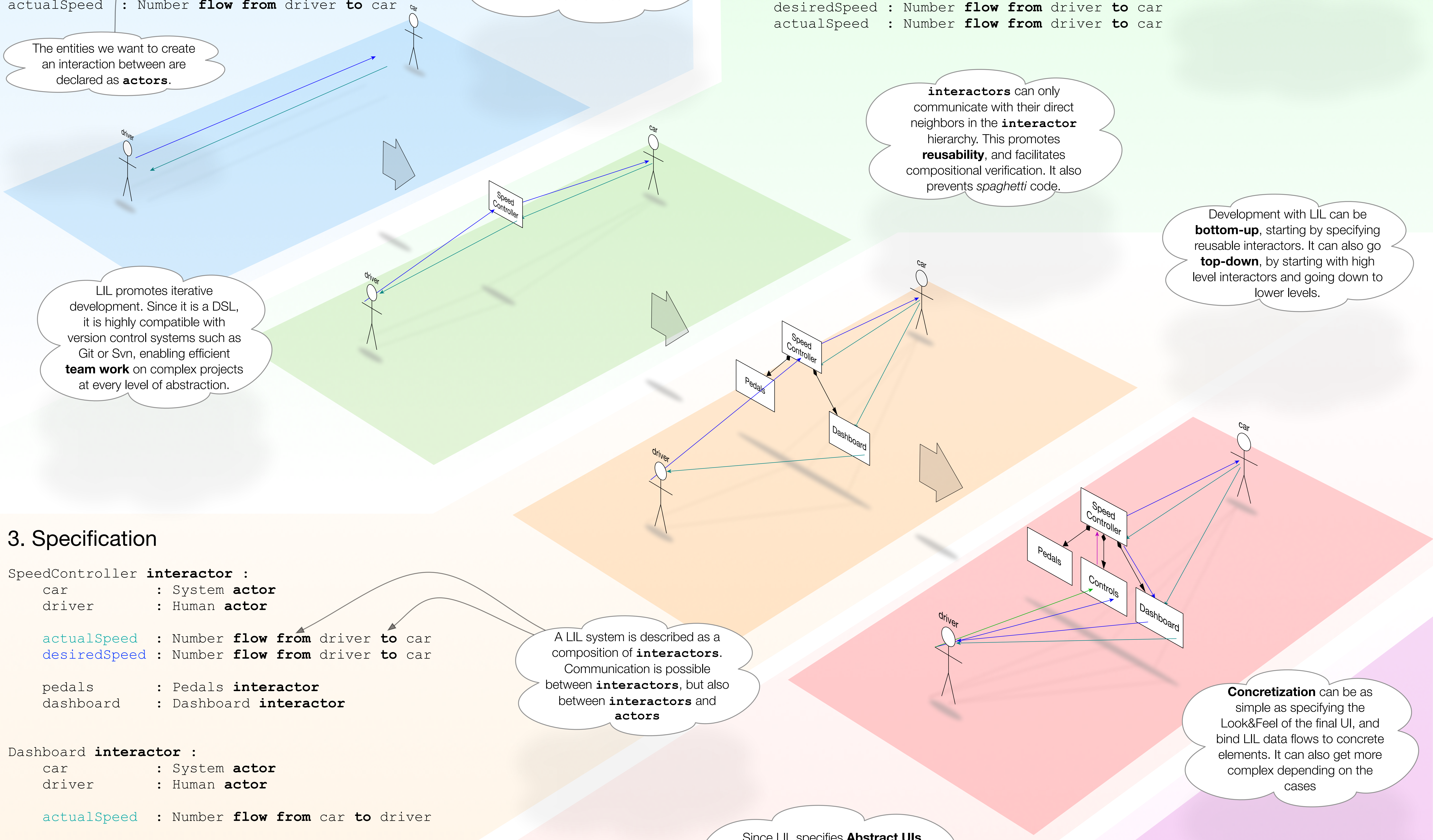
## 1. The need

```
car            : System actor
driver         : Human actor

desiredSpeed   : Number flow from driver to car
actualSpeed    : Number flow from driver to car
```

The two ways data can flow in LIL system are **flow** and **event**. A **flow** is continuous in time, while an **event** only happens at discrete times.

The entities we want to create an interaction between are declared as **actors**.

LIL promotes iterative development. Since it is a DSL, it is highly compatible with version control systems such as Git or Svn, enabling efficient **team work** on complex projects at every level of abstraction.



## 2. High level specification

```
SpeedController interactor :
    car            : System actor
    driver         : Human actor

    desiredSpeed : Number flow from driver to car
    actualSpeed  : Number flow from driver to car
```

With LIL you specify different **interactor** types. An **interactor** declares which **actors** an **interactors** it communicates with, and which data flows go through it.

**interactors** can only communicate with their direct neighbors in the **interactor** hierarchy. This promotes **reusability**, and facilitates compositional verification. It also prevents *spaghetti* code.

Development with LIL can be **bottom-up**, starting by specifying reusable interactors. It can also go **top-down**, by starting with high level interactors and going down to lower levels.

## 3. Specification

```
SpeedController interactor :
    car            : System actor
    driver         : Human actor

    actualSpeed    : Number flow from driver to car
    desiredSpeed   : Number flow from driver to car

    pedals         : Pedals interactor
    dashboard      : Dashboard interactor

Dashboard interactor :
    car            : System actor
    driver         : Human actor

    actualSpeed    : Number flow from car to driver
```

A LIL system is described as a composition of **interactors**. Communication is possible between **interactors**, but also between **interactors** and **actors**

**Concretization** can be as simple as specifying the Look&Feel of the final UI, and bind LIL data flows to concrete elements. It can also get more complex depending on the cases

Since LIL specifies **Abstract UIs** and their behavior, a specification can have different concrete implementations. LIL tries to reduce the concretization phase to the minimum.

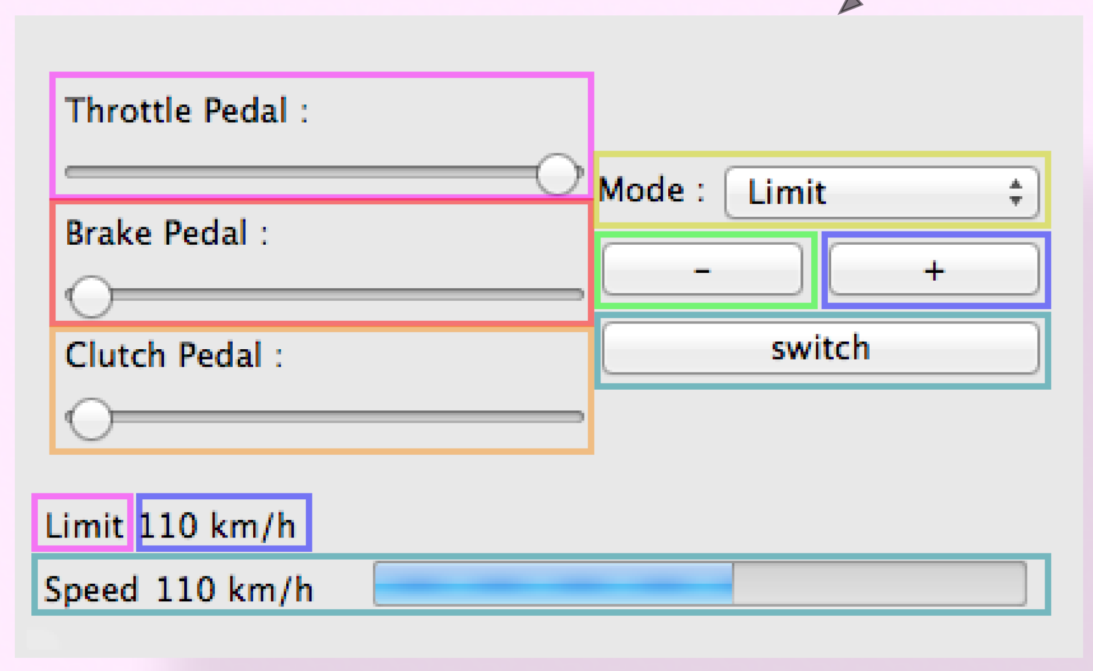
## 4. Abstract UI design

```
SpeedController interactor:
    car             : System actor
    driver          : Human actor

    actualSpeed     : Number flow from car
    desiredSpeed    : Number flow to car

    dashboard       : Dashboard interactor
    controls        : Controls interactor
    pedals          : Pedals interactor

    changeSpeed     : Number event from controls

    on changeSpeed(x)  : desiredSpeed = desiredSpeed + x
    30 < desiredSpeed < 150

Dashboard interactor:
    car             : System actor
    driver          : Human actor

    actualSpeed     : Number flow from car to driver
    desiredSpeed    : Number flow from parent to driver

Controls interactor:
    driver          : Human actor

    increment       : Void event from driver
    decrement       : Void event from driver
    changeSpeed     : Number event to parent

    on increment    : send changeSpeed(+5)
    on decrement    : send changeSpeed(-5)
```

One of the most important part of LIL is the declaration of **behavior** of UIs. Behaviors can mix **flows** and **events**, they can agregate and destructure data. But most of all, they are specified in a **declarative** manner.

## 5. Final products

- Protoyping code
- Final application code skeleton
- Model checking code
- Safety properties
- Test cases
- Textual specification
- ARINC 661 definition file skeleton for Cockpit Display Systems
- ...


Throttle Pedal :
Brake Pedal :
Clutch Pedal :
Mode : Limit
-
+
switch
Limit 110 km/h
Speed 110 km/h