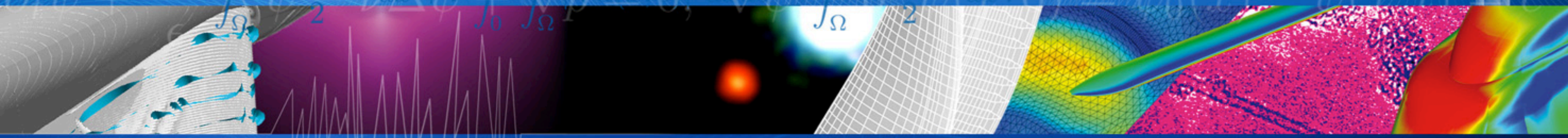


A formal language for safety-critical embedded user interfaces

Vincent LECRUBIER
ONERA, DTIM

ONERA
THE FRENCH AEROSPACE LAB

return on innovation

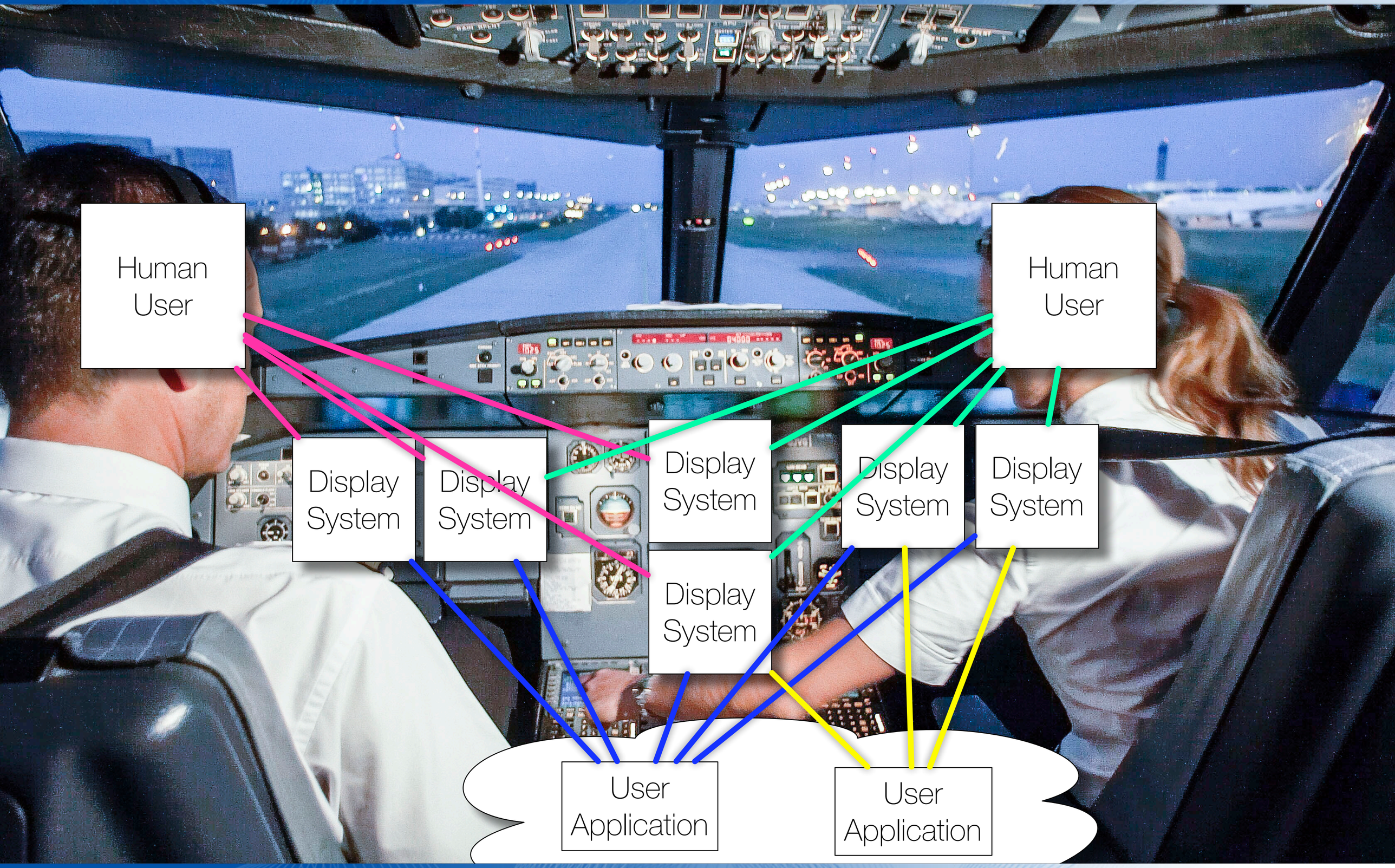
Context

Context

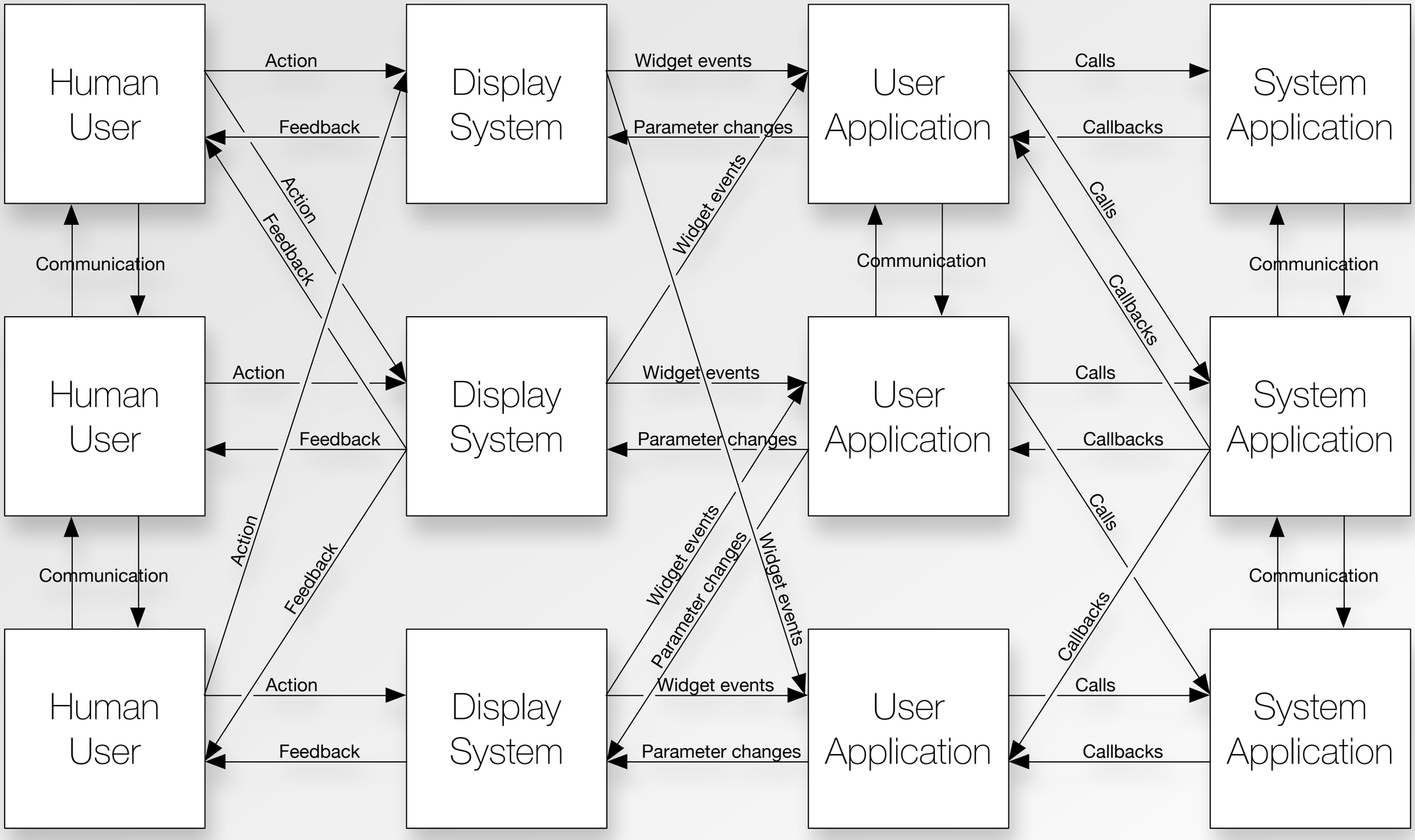
Context


Human User
Human User
Display System
Display System
Display System
Display System
Display System
Display System

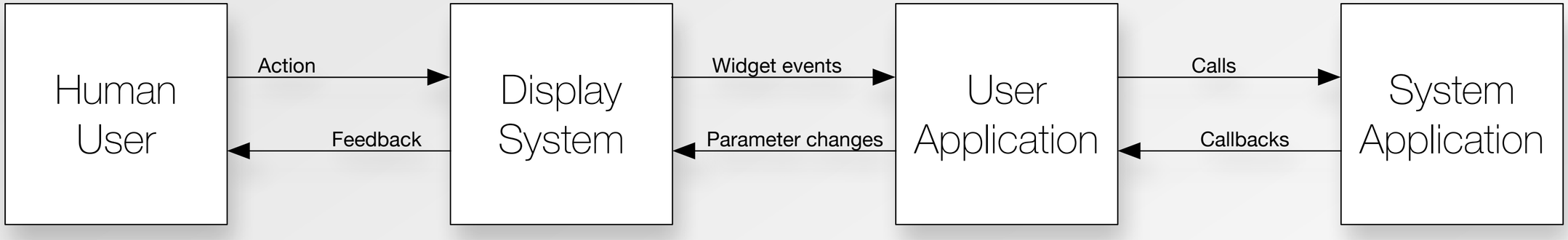
Context


Human User
Human User
Display System
Display System
Display System
Display System
Display System
Display System
User Application
User Application

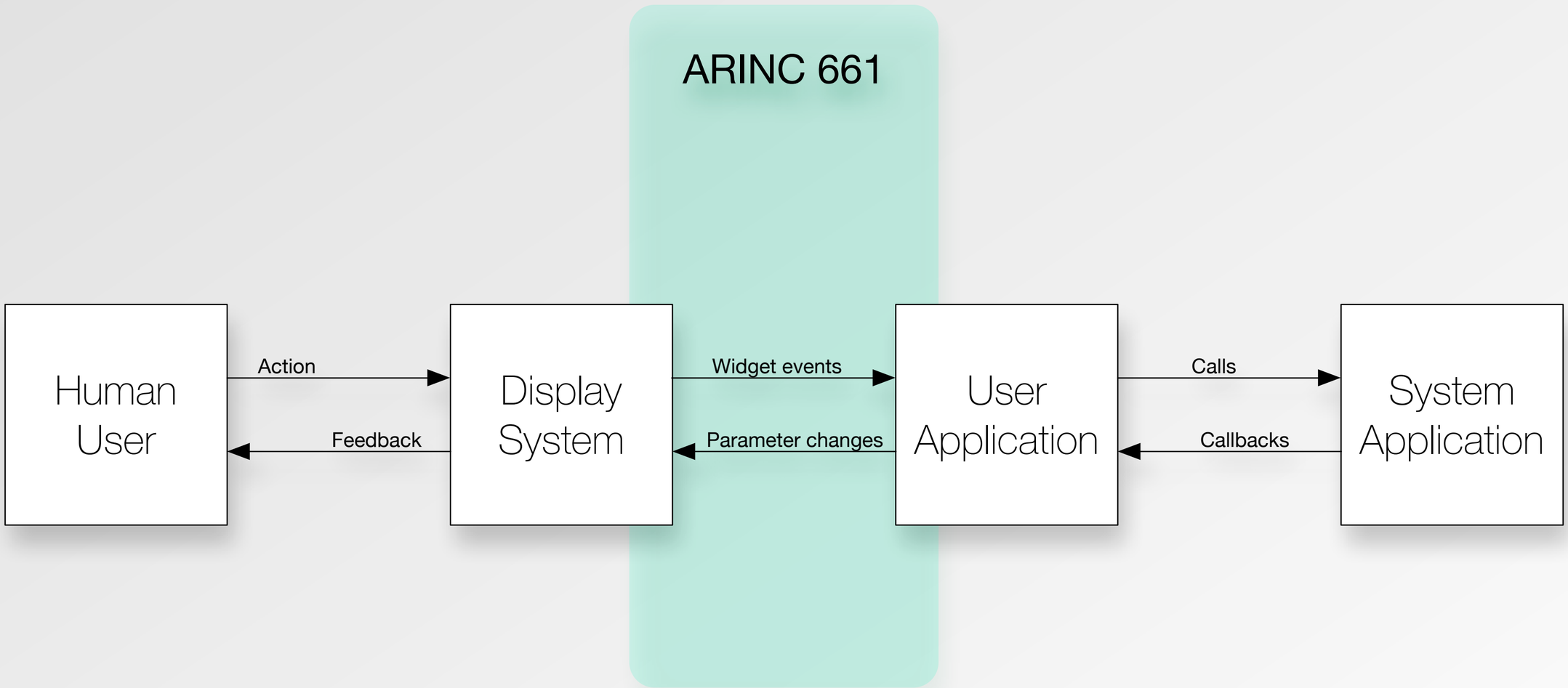
Context : Overview



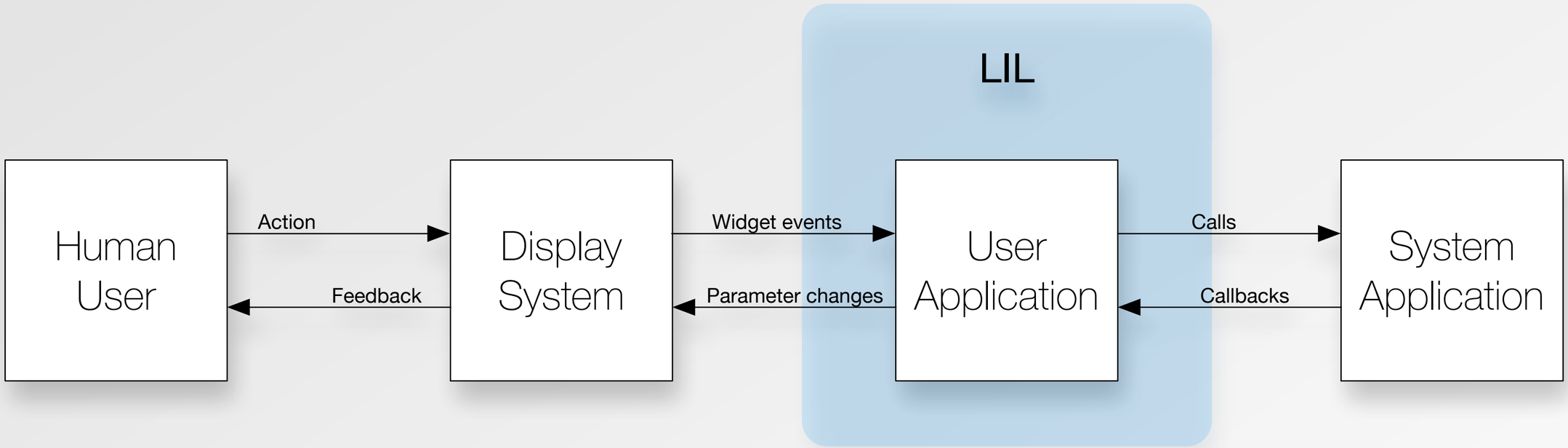
Context : Let's simplify



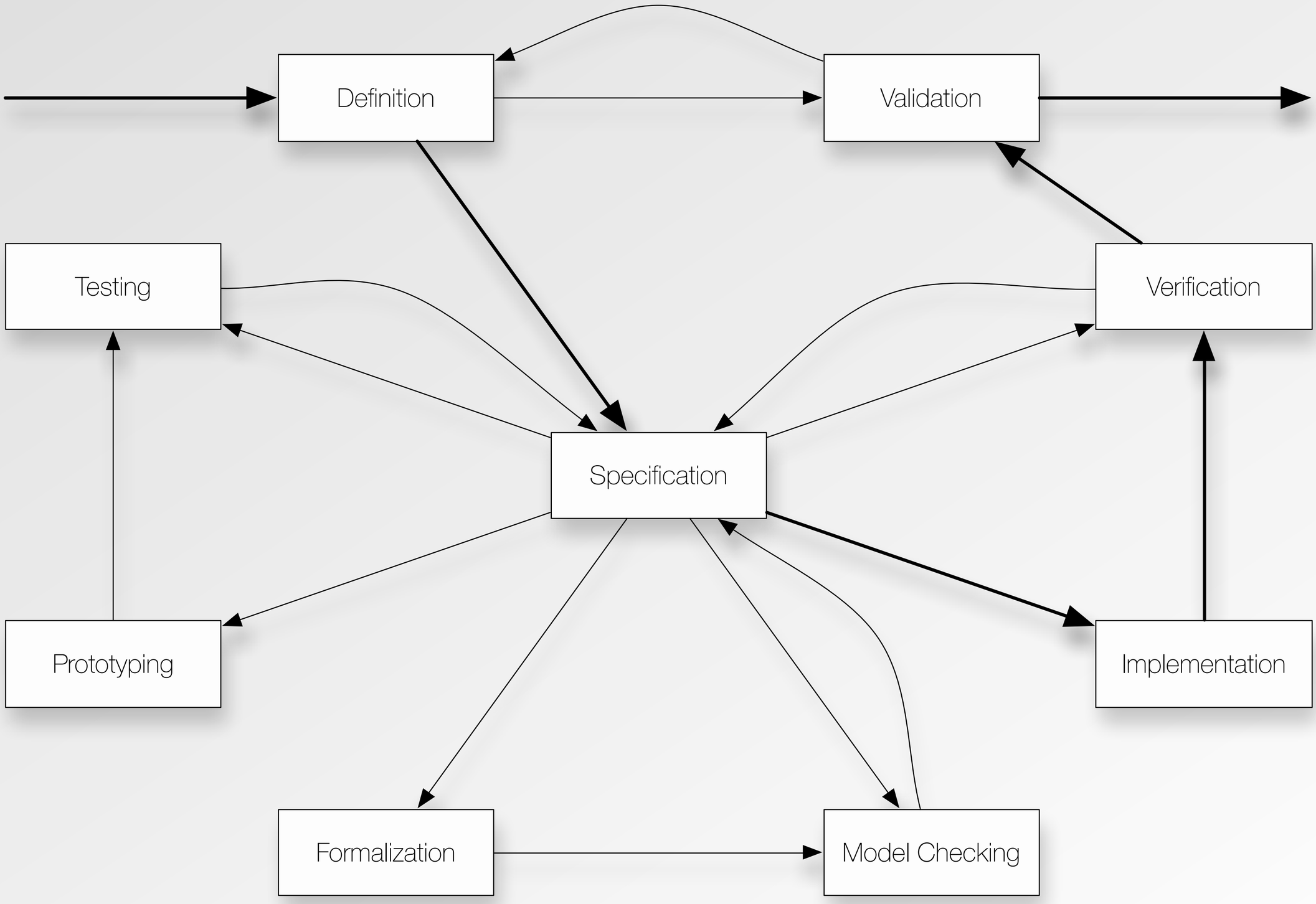
Context : ARINC 661



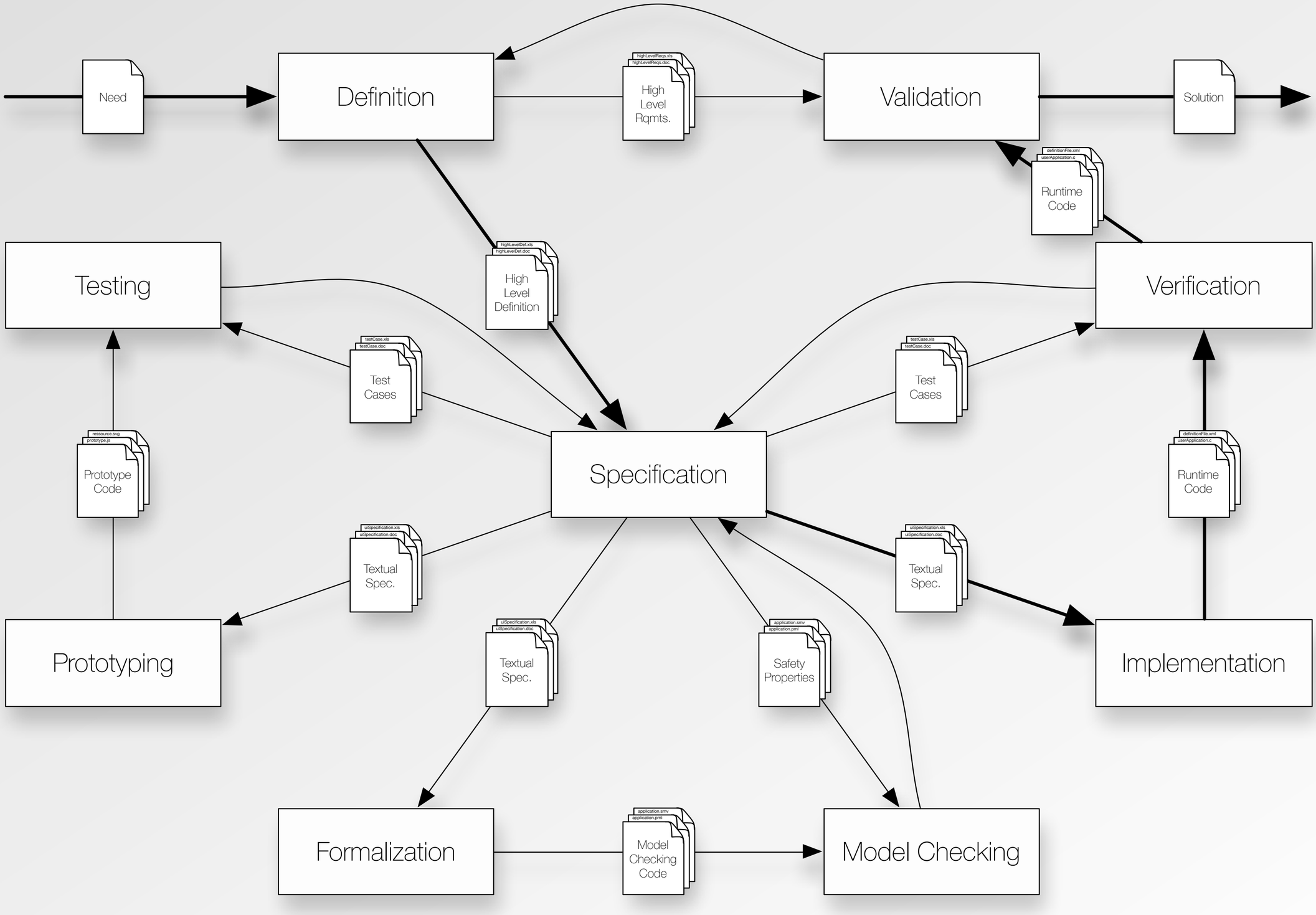
Context : LIL Perimeter



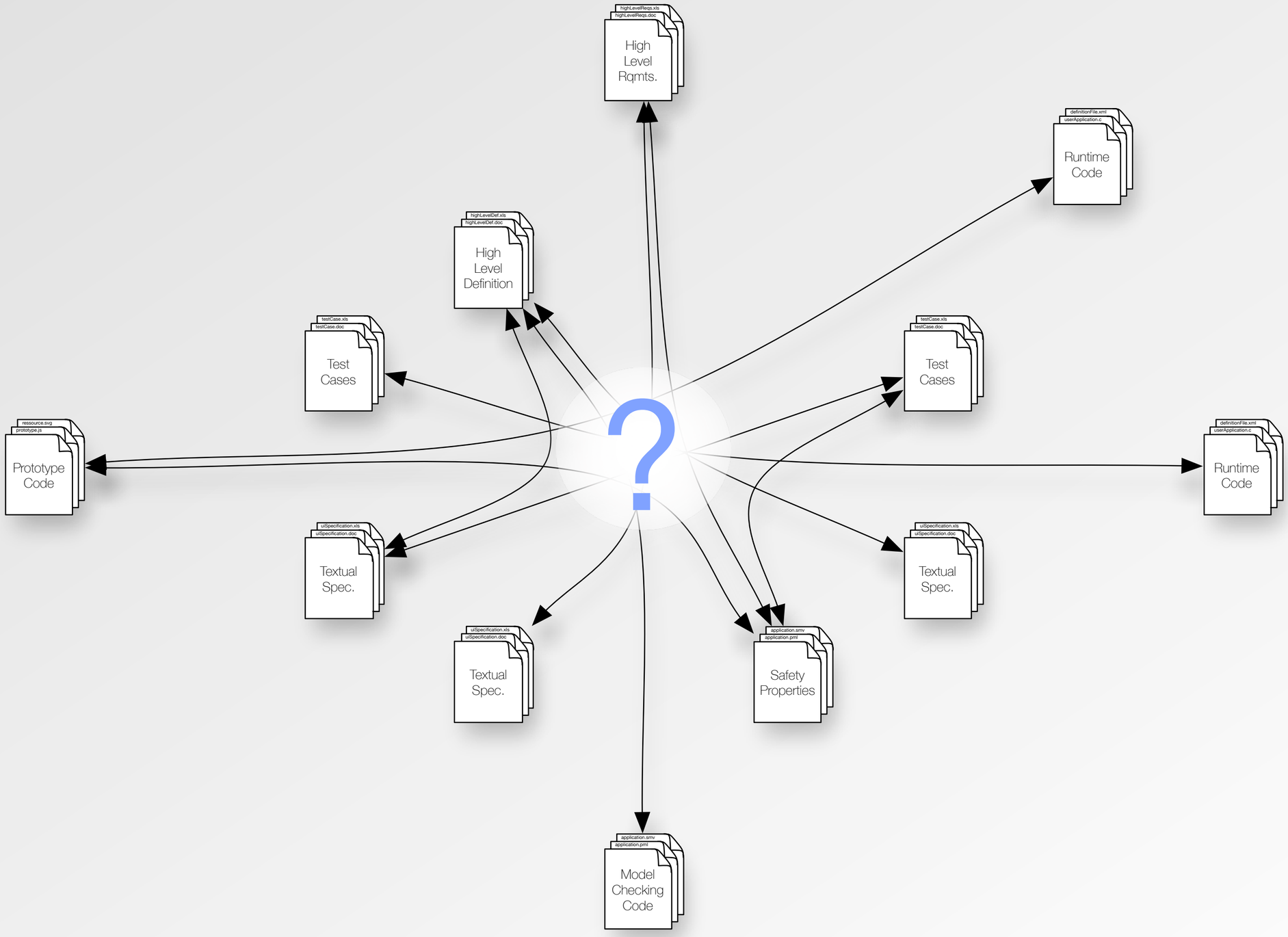
Why is this interesting : HMI Design process



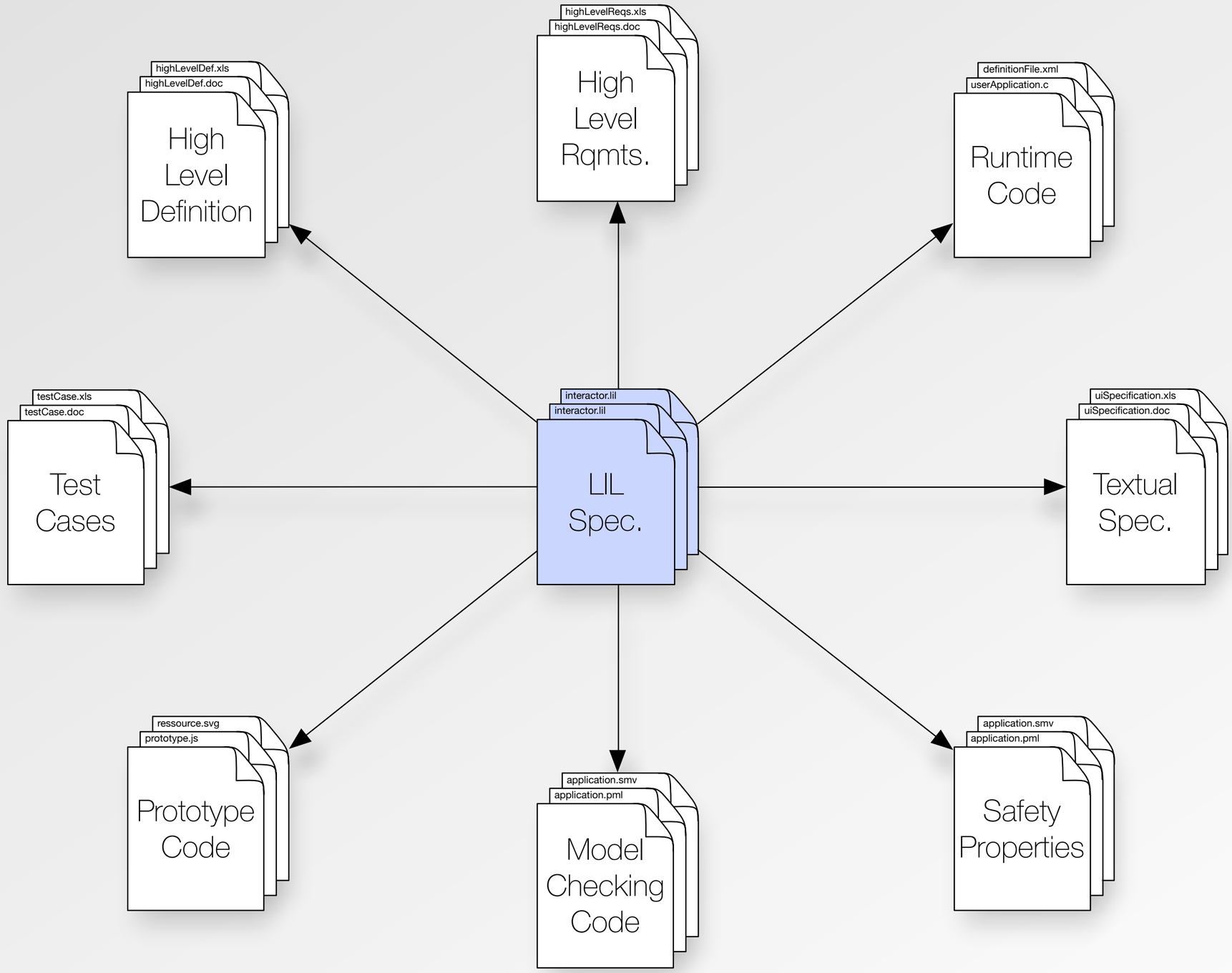
Why is this interesting : HMI Design process artifacts



Why is this interesting : How to link HMI design artifacts



Why is this interesting : A pivot language



Methodology

- State of the art
  - Scientific approaches to HMI design
  - Industrial approaches and languages
  - Model checking tools

- Definition of the LIL language
  - XText / Antlr
  - Mathematically

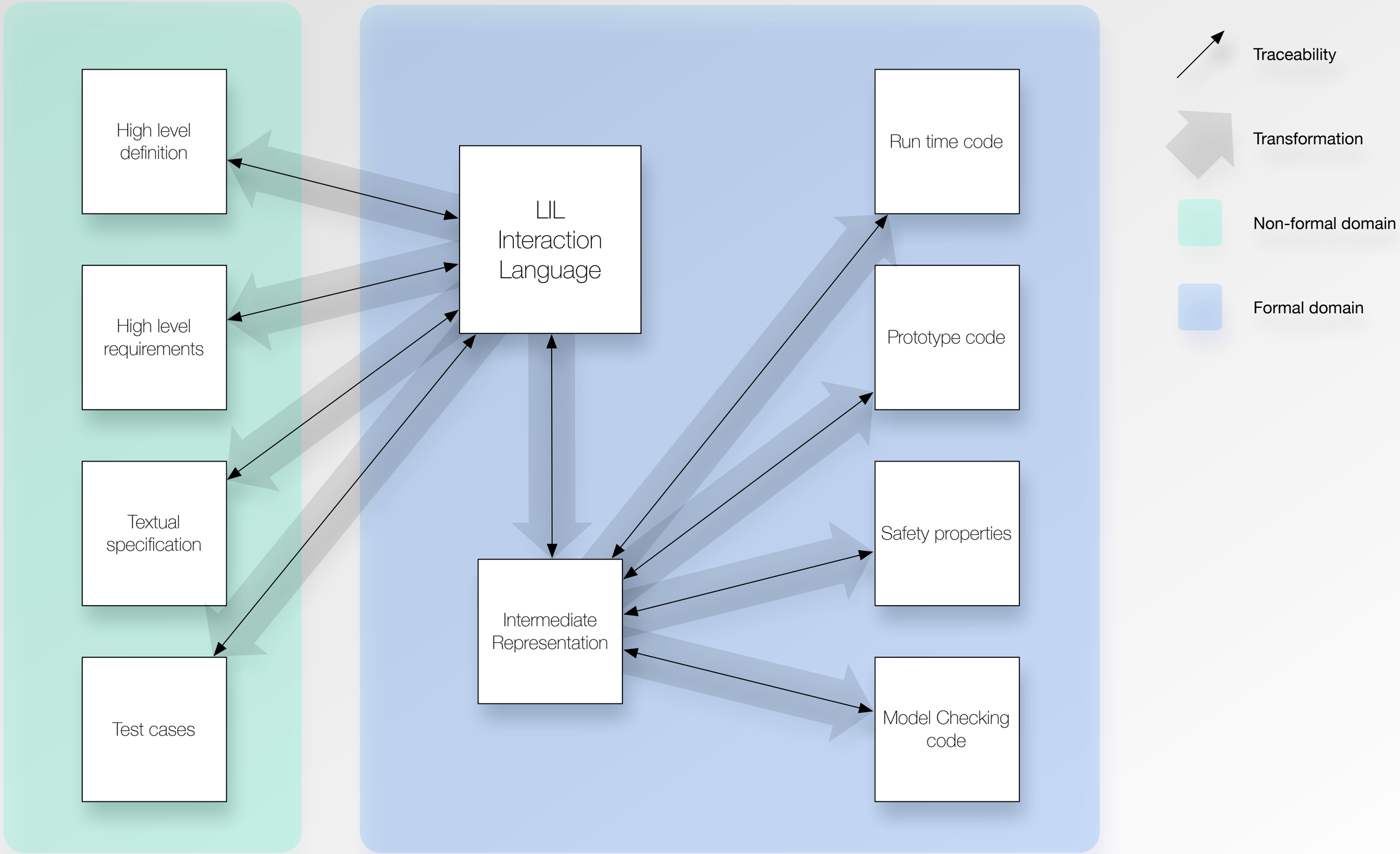
- Definition of transformations between
  - Study cases in LIL
  - Implementation code
  - Model checking code

Results : Overview

- LIL Interaction Language
  - Formal (maps to the intermediate representation)
  - Similar to natural language
  - Simple syntax
  - Specific constructs for HMI design
  - Structural and behavioral aspects

- Intermediate representation
  - Formal (directly mathematically defined)
  - Express compositions of state machines
  - Intermediate step for code generation
  - No structural aspects
  - Only behavioral aspects

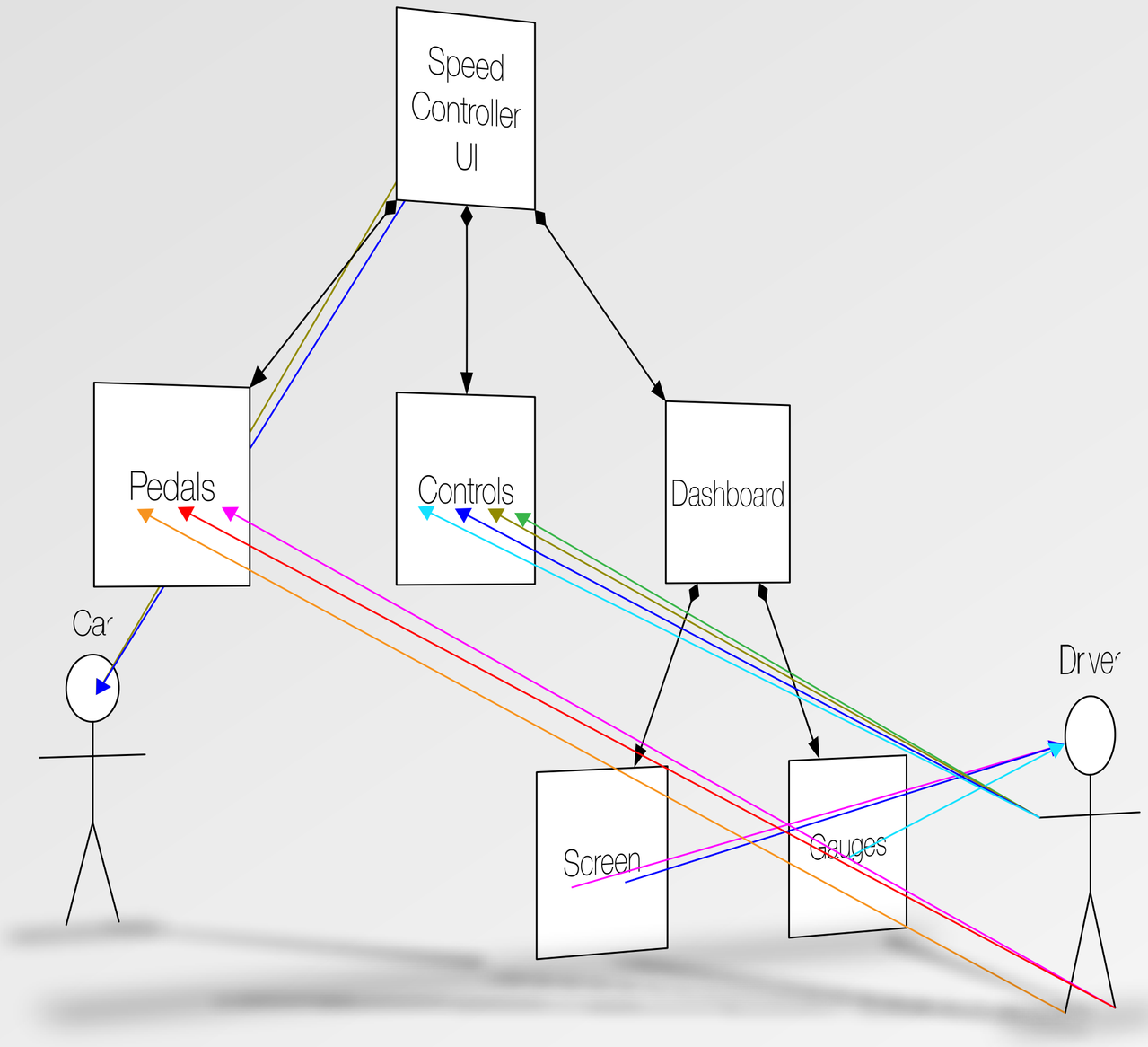
Results : Transformations and traceability



Results : LIL language overview

- Abstraction !
- Data types
- Actor types
- Interactor types
  - Composition
  - Data flow
  - Behavior

Example : Speed controller



Introducing Actors

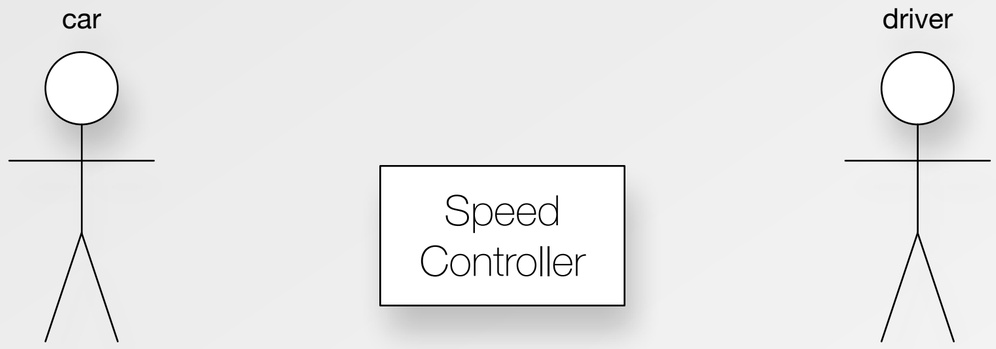
- Elements outside of the perimeter of the interaction system

- Express users, roles, compostion

- May be used to specify modality

Example : Expression of the need

```
SpeedController interactor:

  driver          : Human actor
  car             : System actor
```



Introducing Data flow

- Basic static properties of the flow of data through interactors

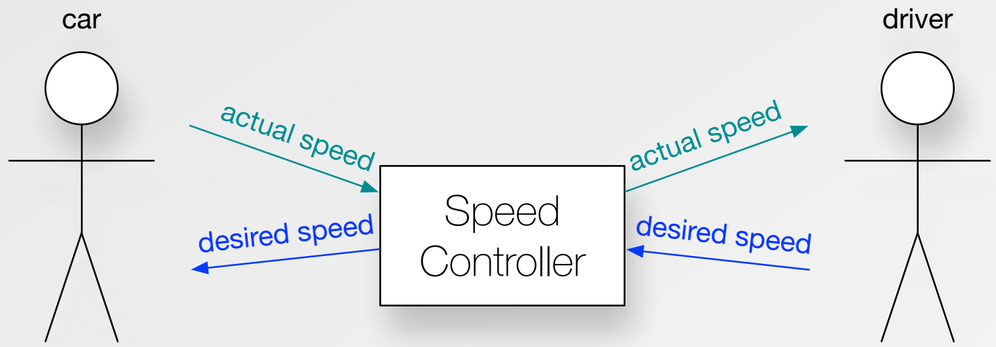
- Two categories
  - Events (Discrete)
  - Flows (Continuous)

- The usual data types
  - Boolean, Number, Text, Enums…
  - Structures and Unions
  - Lists, Sets, Arrays…

- Various sources and destinations
  - To/From the current interactor
  - To/From other interactors
  - To/From actors

Example : High level definition

```
SpeedController interactor:

   driver          : Human actor
   car             : System actor

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

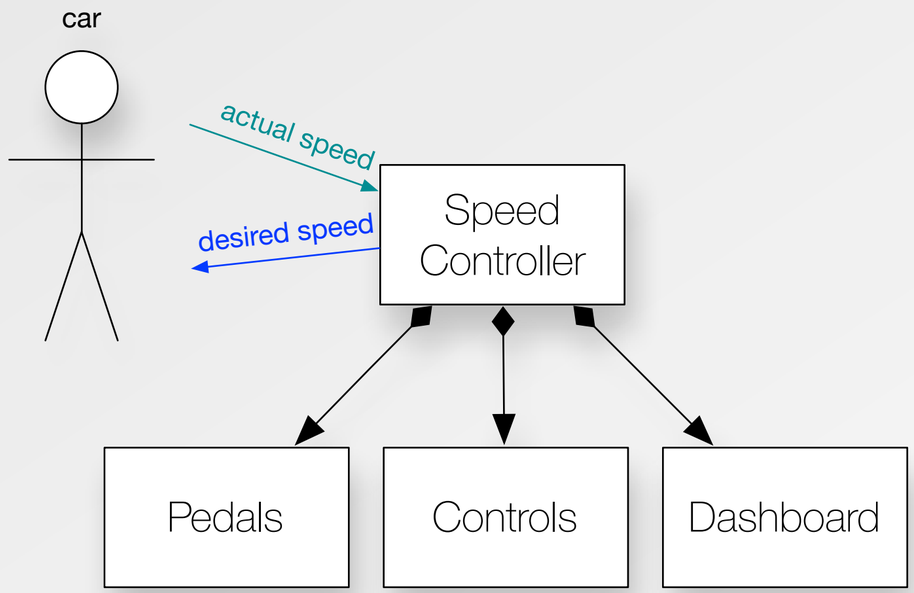


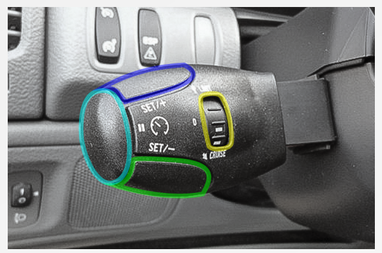
Introducing Composition

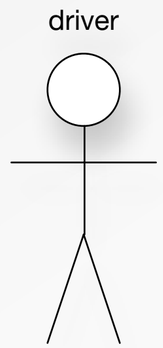
- Express Architecture of the HMI Hierarchically
- An interactor can be composed of instances of other interactor types
- Can be used to incrementally refine/add detail to specification
- Has an impact on data flow, since only neighbors interactions can communicate directly

Example : Architecture

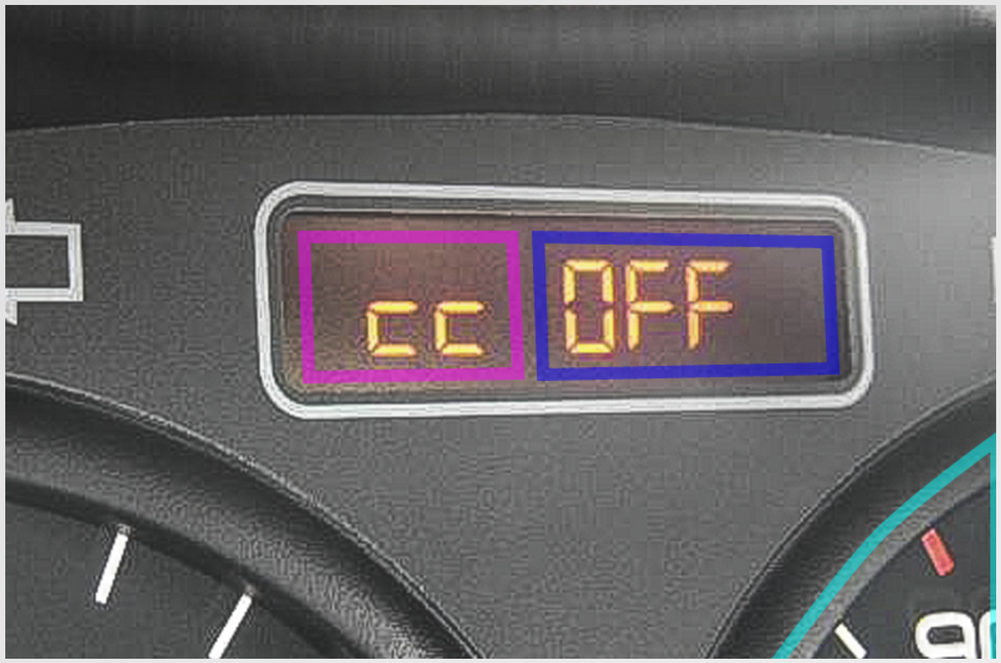
```
SpeedController interactor:

  driver          : Human actor
  car             : System actor

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

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

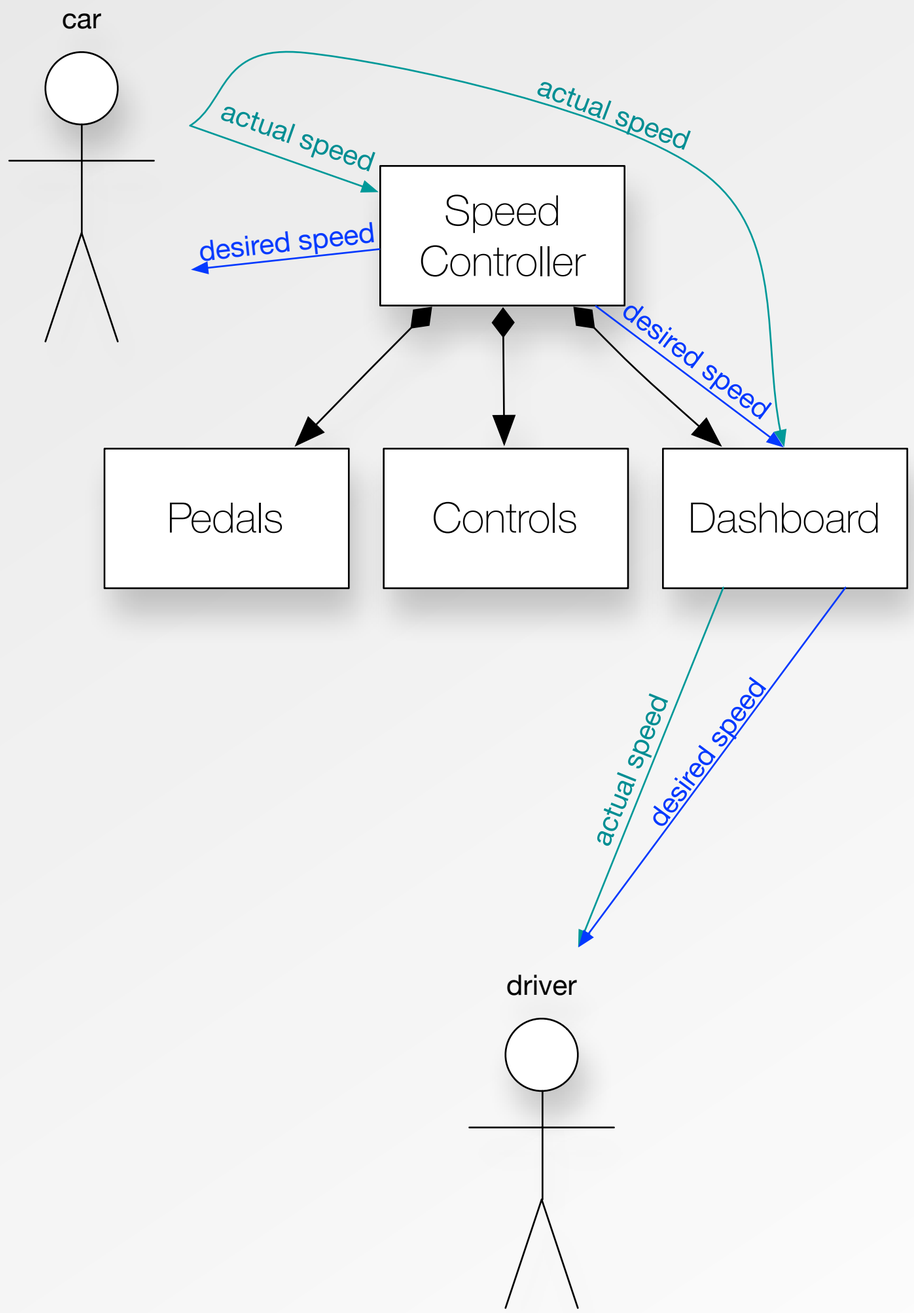






Example : The dashboard






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

  actualSpeed     : Number flow from car to driver
  mode            : Mode flow from parent to driver
  desiredSpeed    : Number flow from parent to driver
```

Introducing Behaviors

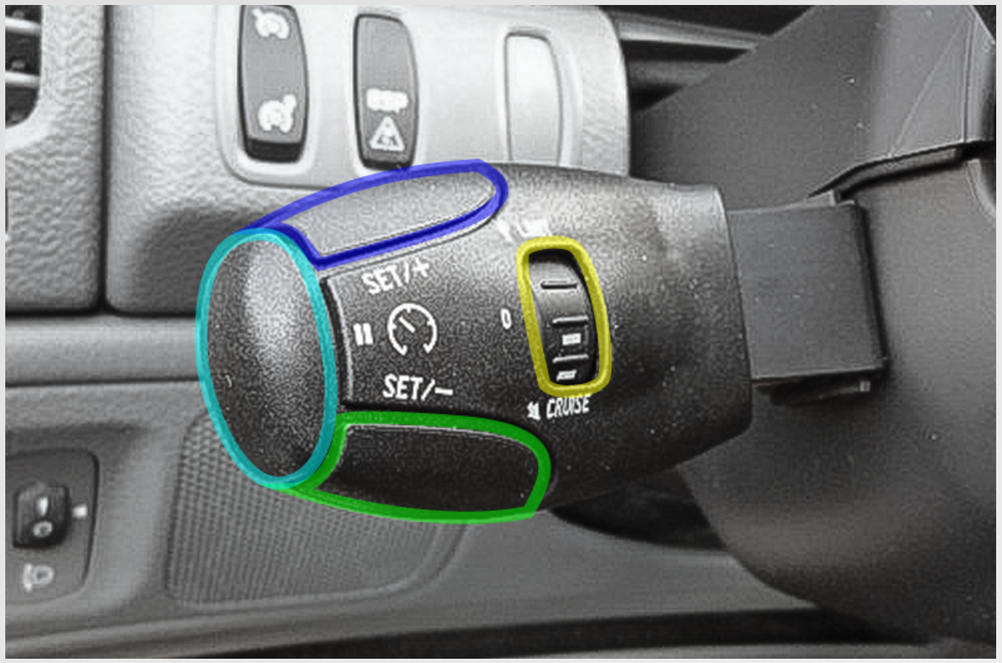
- Express interactions between different data flows and events

- Mix flows and events
  - Flows can trigger events
  - Events can have effects on flows

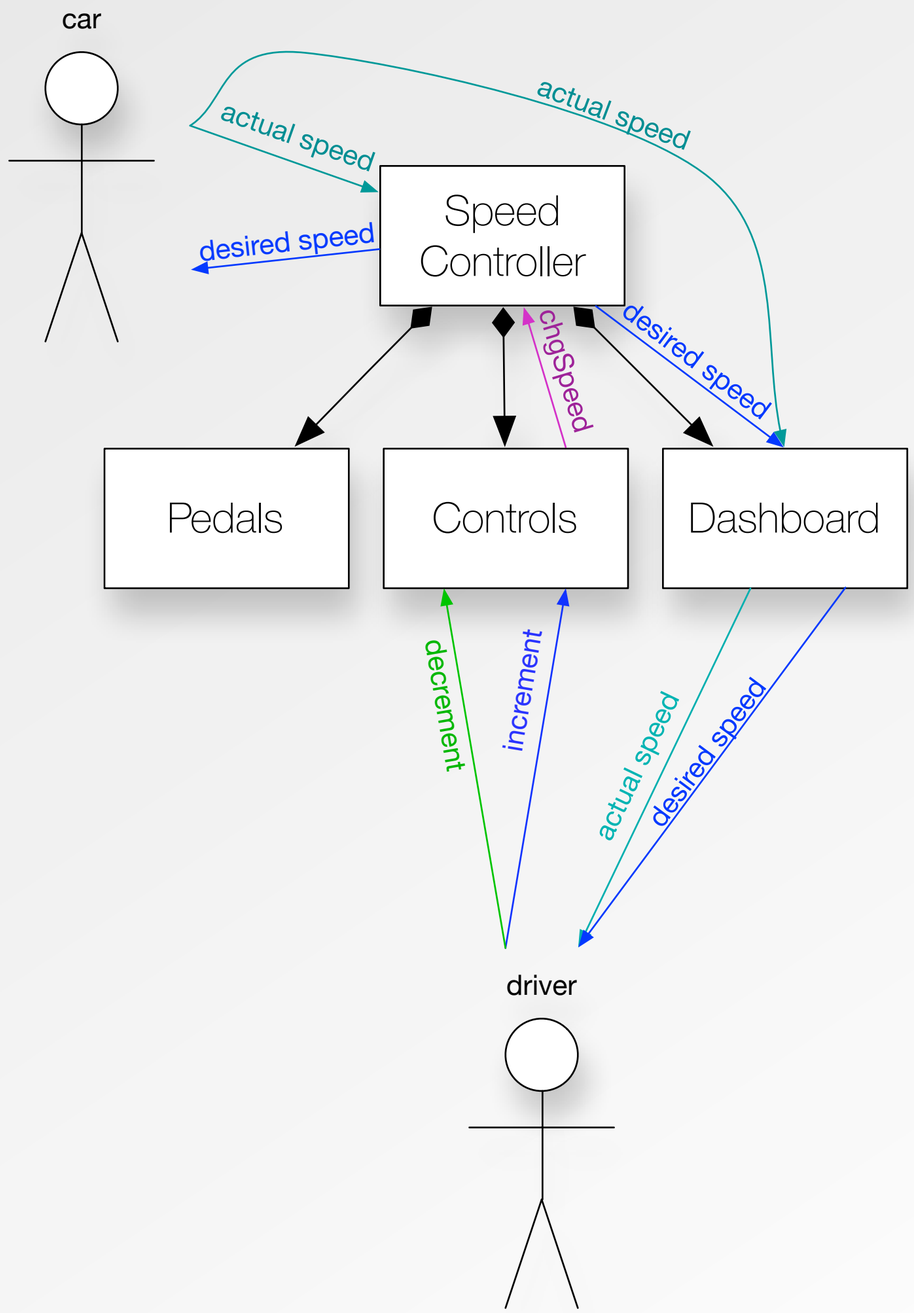
- Structuration and destructuration of data
  - Decompose incoming data to dispatch it
  - Aggregate incoming data from different sources

- Declarative
  - No imperative programming of behaviors
  - Specific operators to deal with common operations on complex data types

Example : The controls

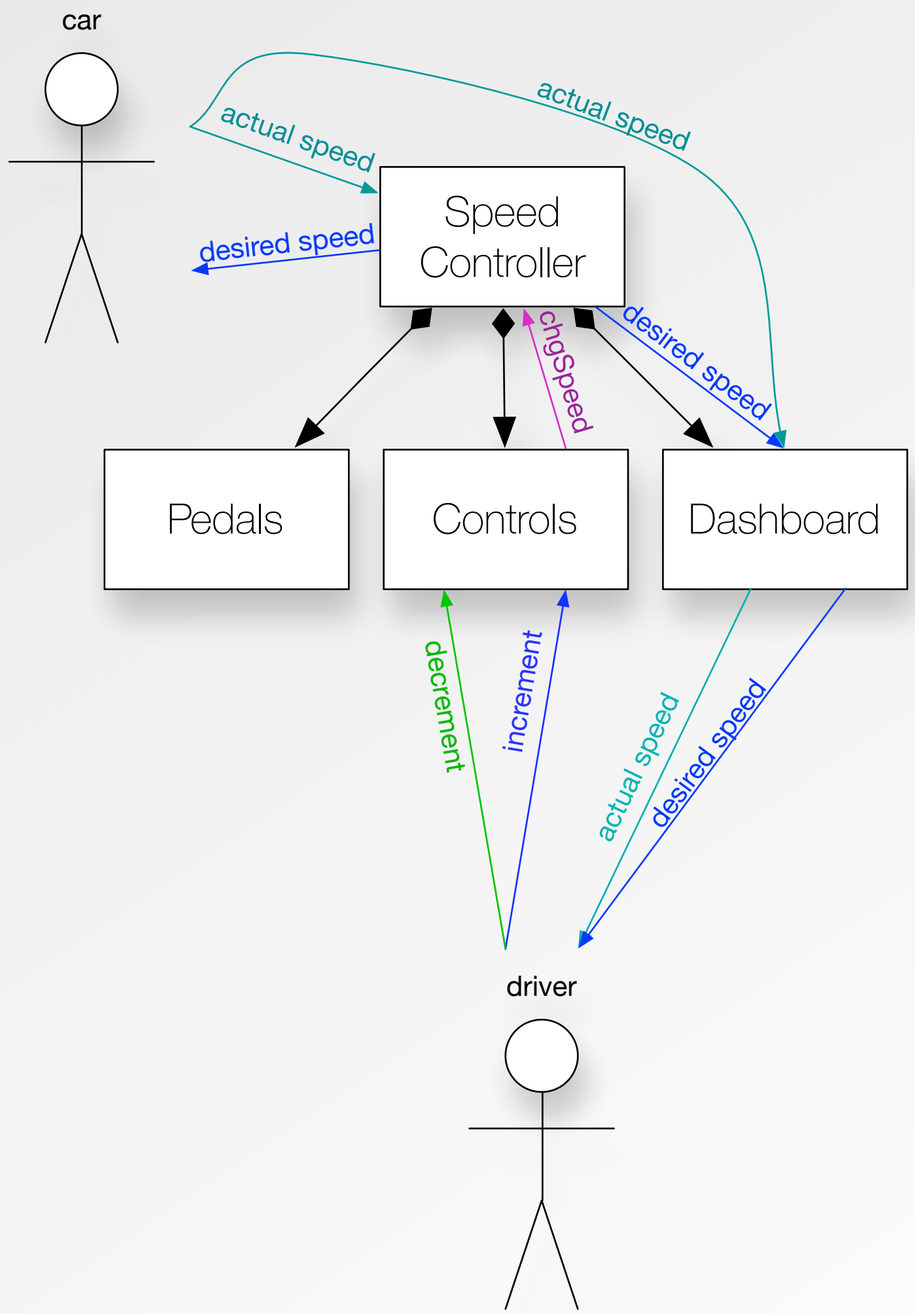



```
Controls interactor:
  driver          : Human actor

  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)
```



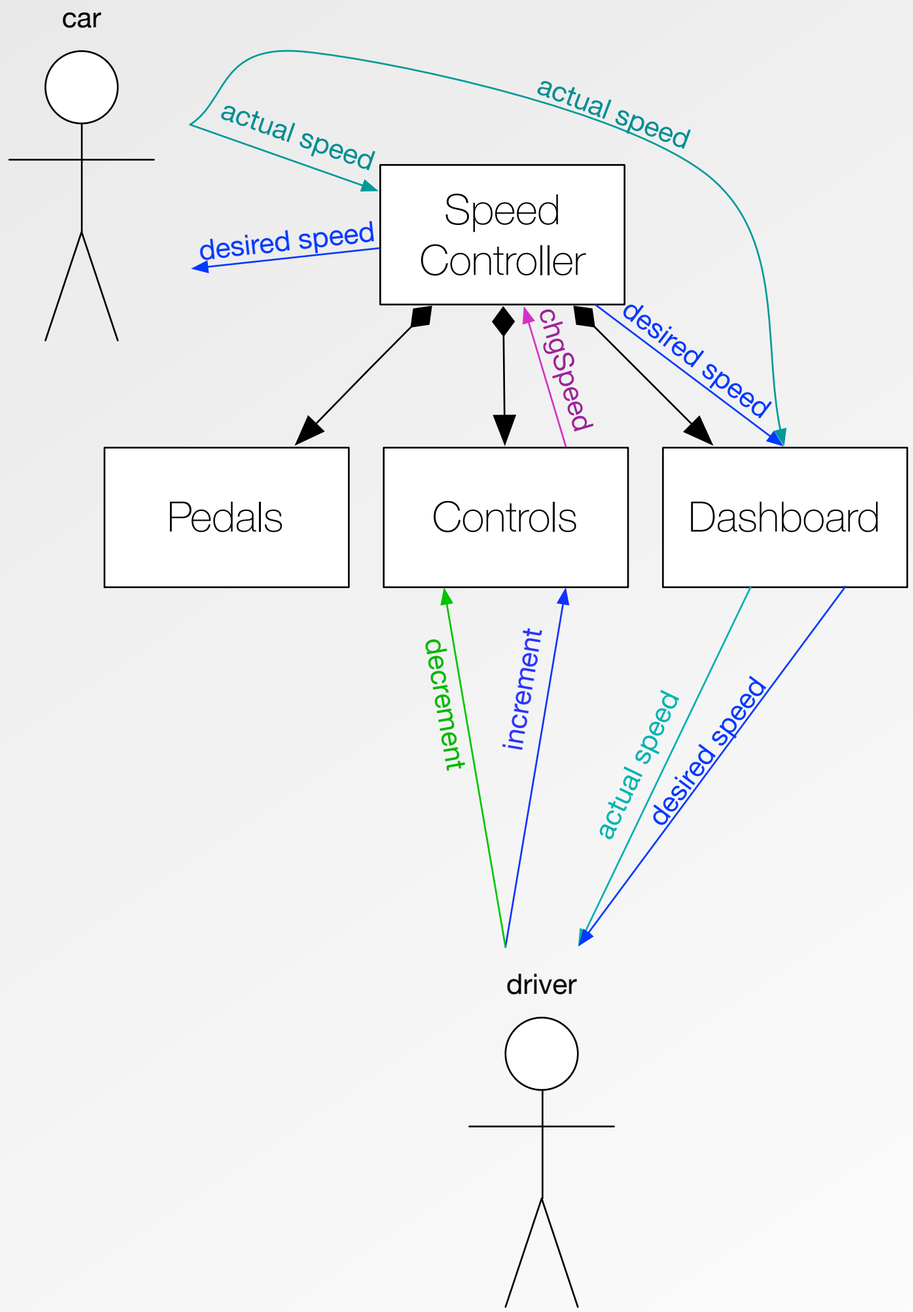
Example : "Final" UI

```
SpeedController interactor:

  driver          : Human actor
  car             : System 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
```

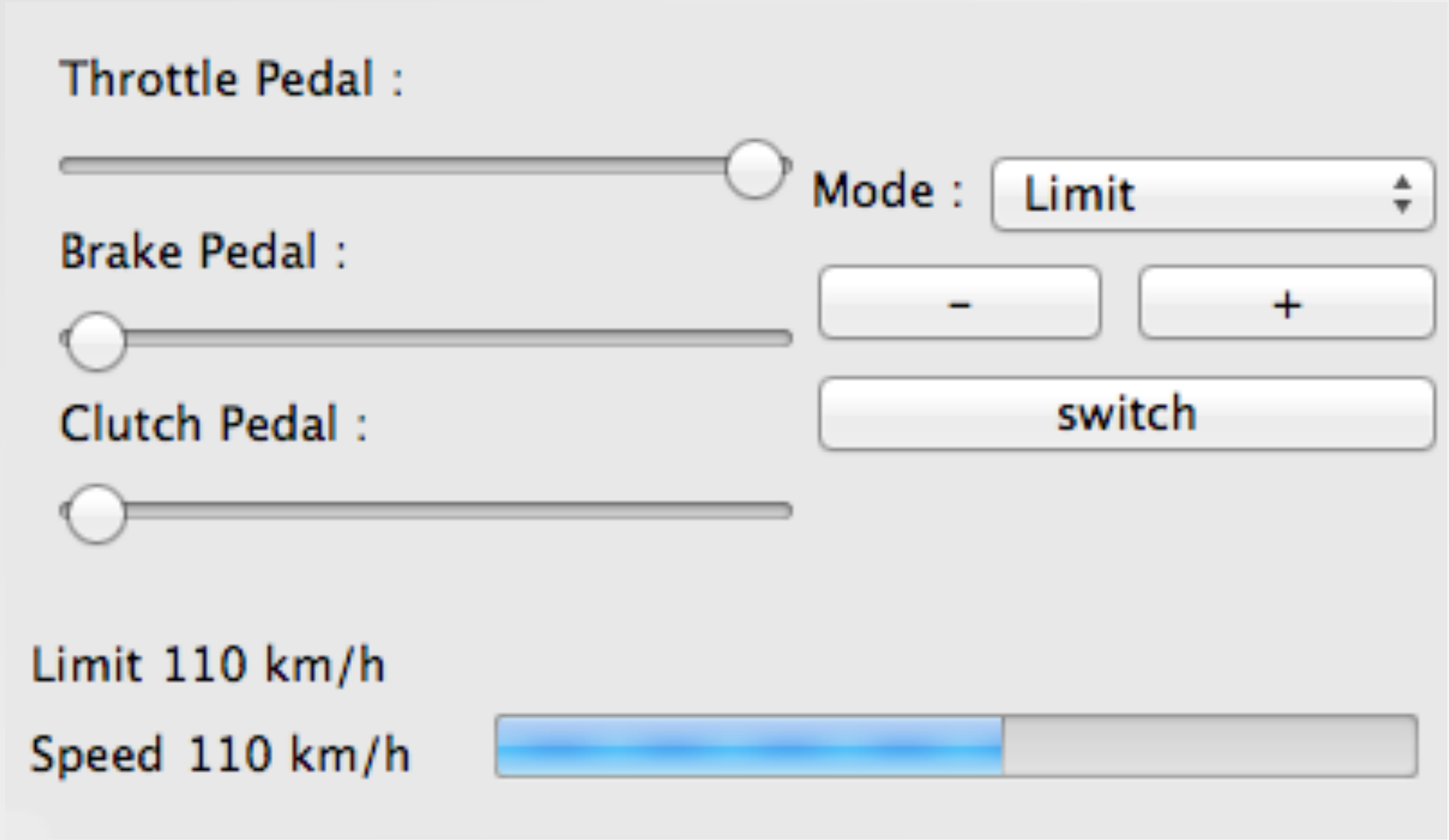


Example : "Final" UI 2

```
SpeedController interactor:

  driver          : Human actor
  car             : System 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
```

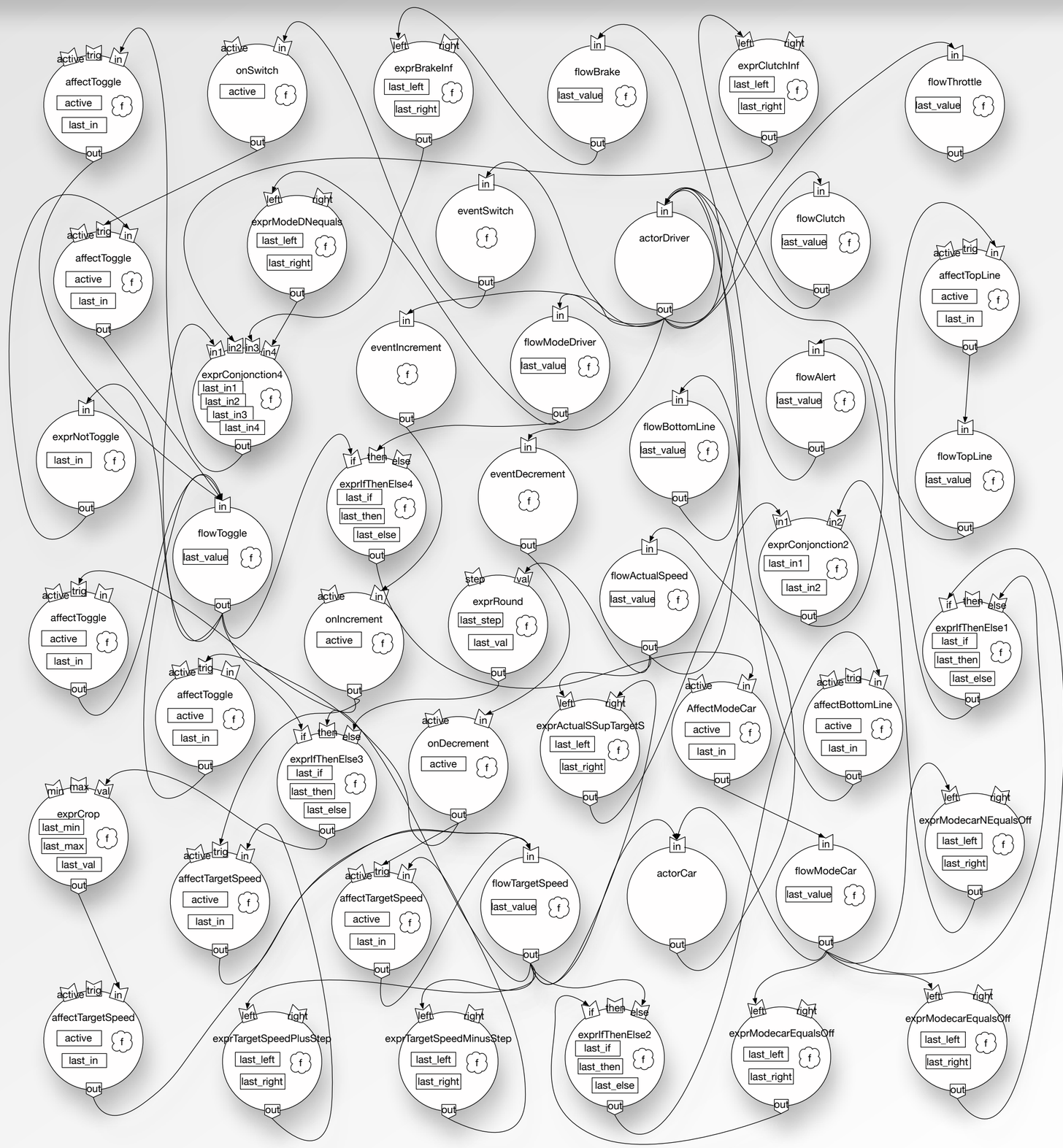


Example : Another concretization



Example : Intermediate representation

- Composition of state machines
- Each state machine represents one atomic behavior or structural element
- Connexions between state machines is determined by the specification of behaviors in LIL



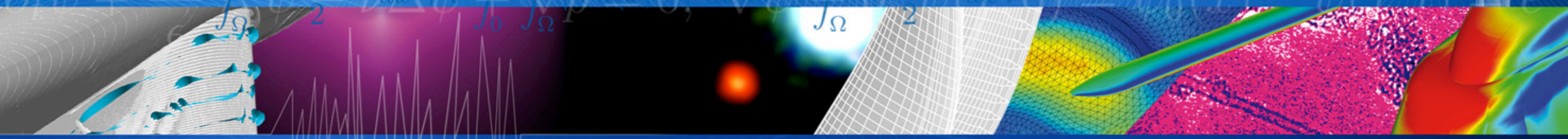
Ongoing work

- Develop tooling
  - Code generators
  - Model checking
  - Traceability

- Develop concretization frameworks
  - ARINC 661
  - Javascript / HTML
  - Java / Swing
  - Qt

Perspectives

- Expression of task model
- Enable (multi-)modality specification
- Enhance Code generation, concretization,
- Apply on bigger study cases

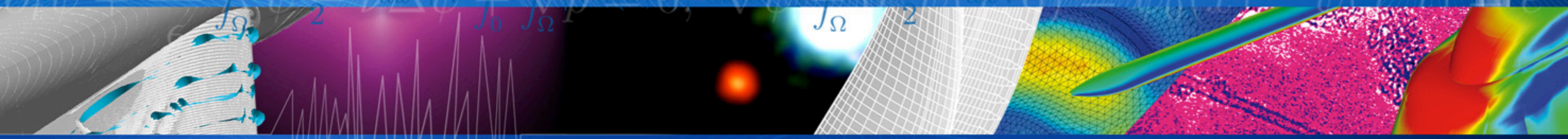


Thank you

ONERA

THE FRENCH AEROSPACE LAB

return on innovation



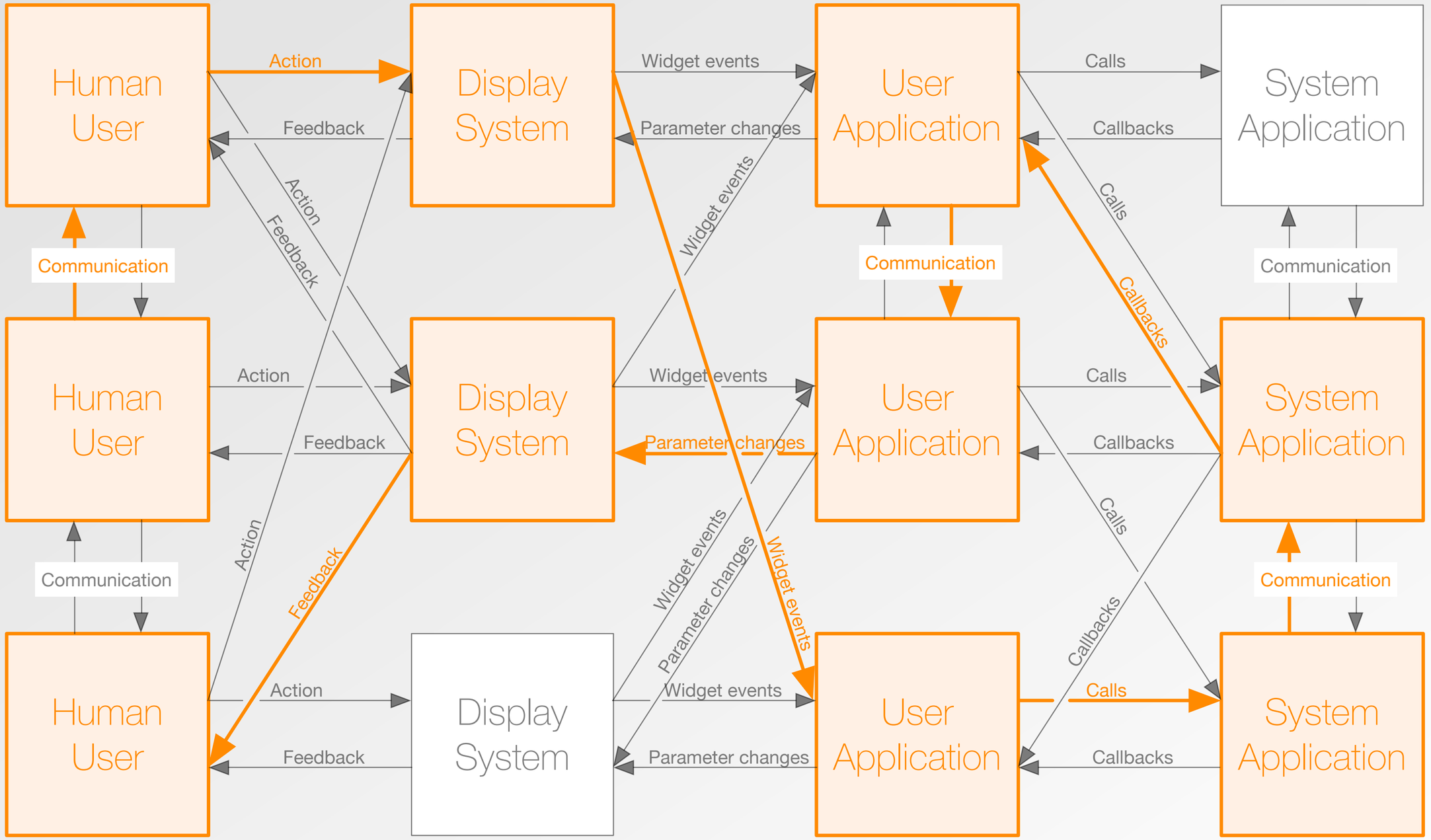
Appendix

ONERA

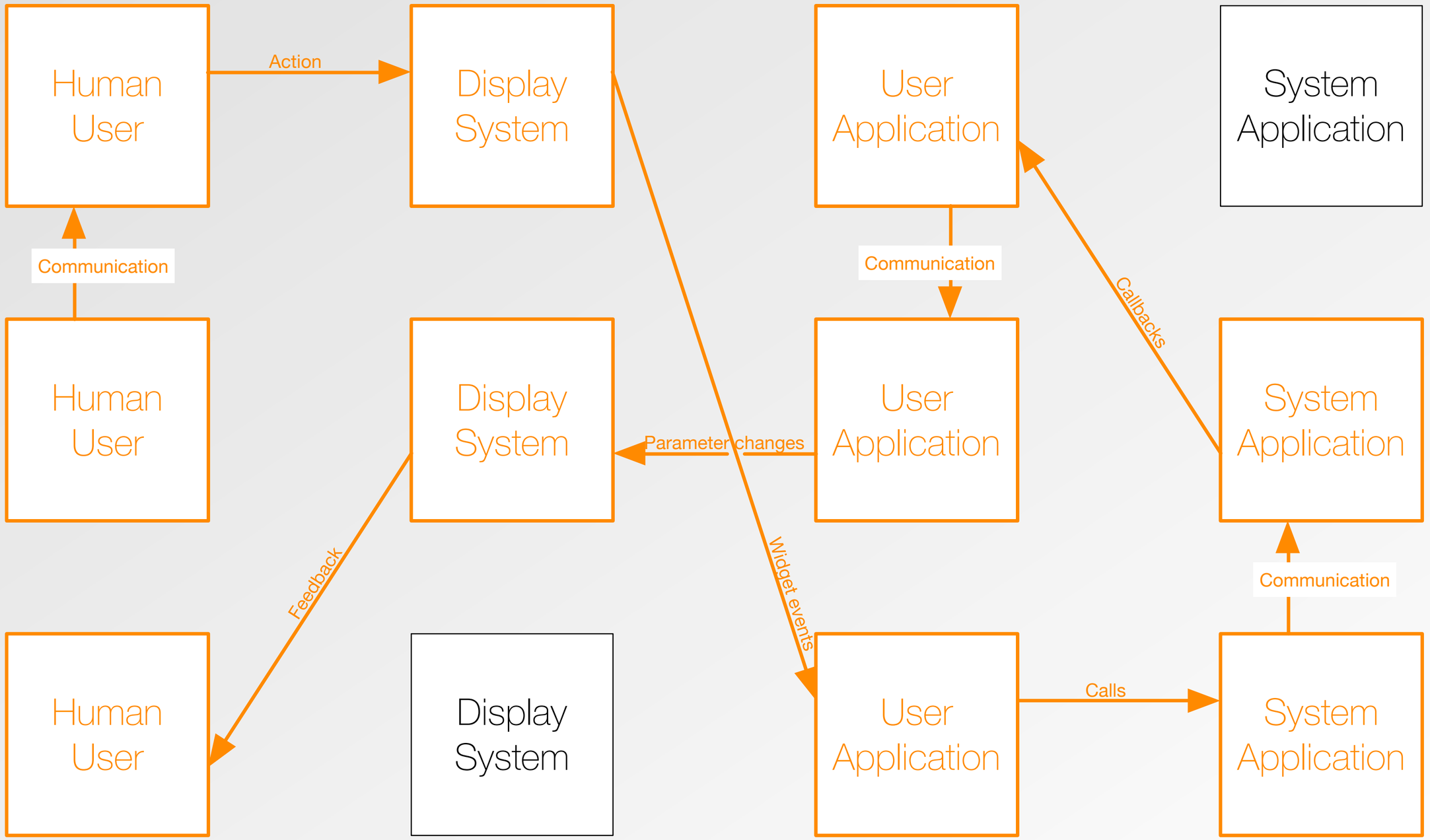
THE FRENCH AEROSPACE LAB

return on innovation

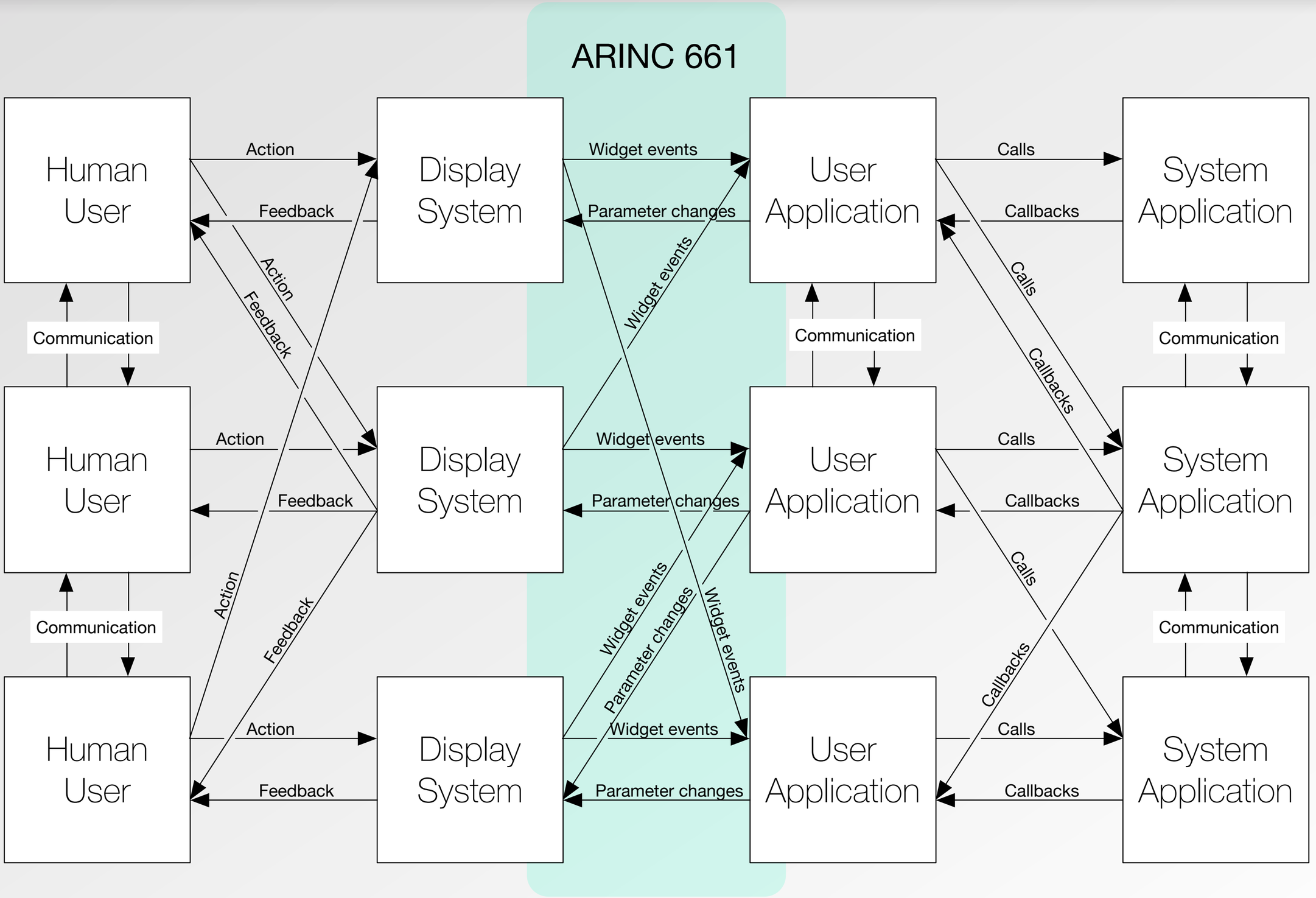
Appendix 1


Human User
Action
Feedback
Display System
Widget events
Parameter changes
User Application
Calls
Callbacks
System Application
Communication
Action
Feedback
Widget events
Calls
Callbacks
Communication
Communication
Human User
Action
Feedback
Display System
Widget events
Parameter changes
User Application
Calls
Callbacks
System Application
Communication
Action
Feedback
Widget events
Parameter changes
Widget events
Calls
Callbacks
Communication
Human User
Action
Feedback
Display System
Widget events
Parameter changes
User Application
Calls
Callbacks
System Application

Appendix 2

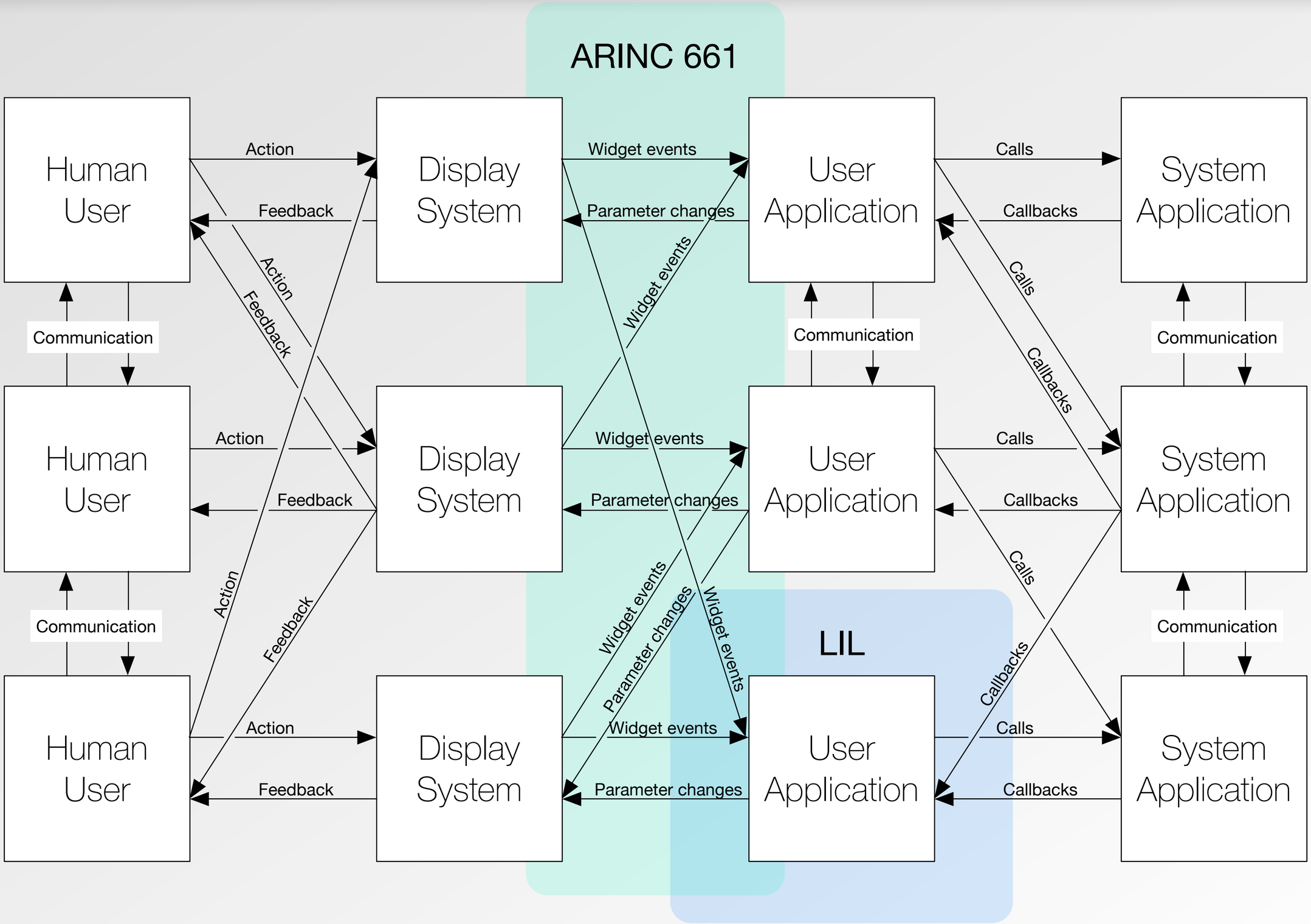

Human User
Action
Display System
User Application
System Application
Communication
Callbacks
Human User
Display System
Parameter changes
User Application
System Application
Feedback
Widget events
Communication
Human User
Display System
User Application
Calls
System Application

Appendix 3



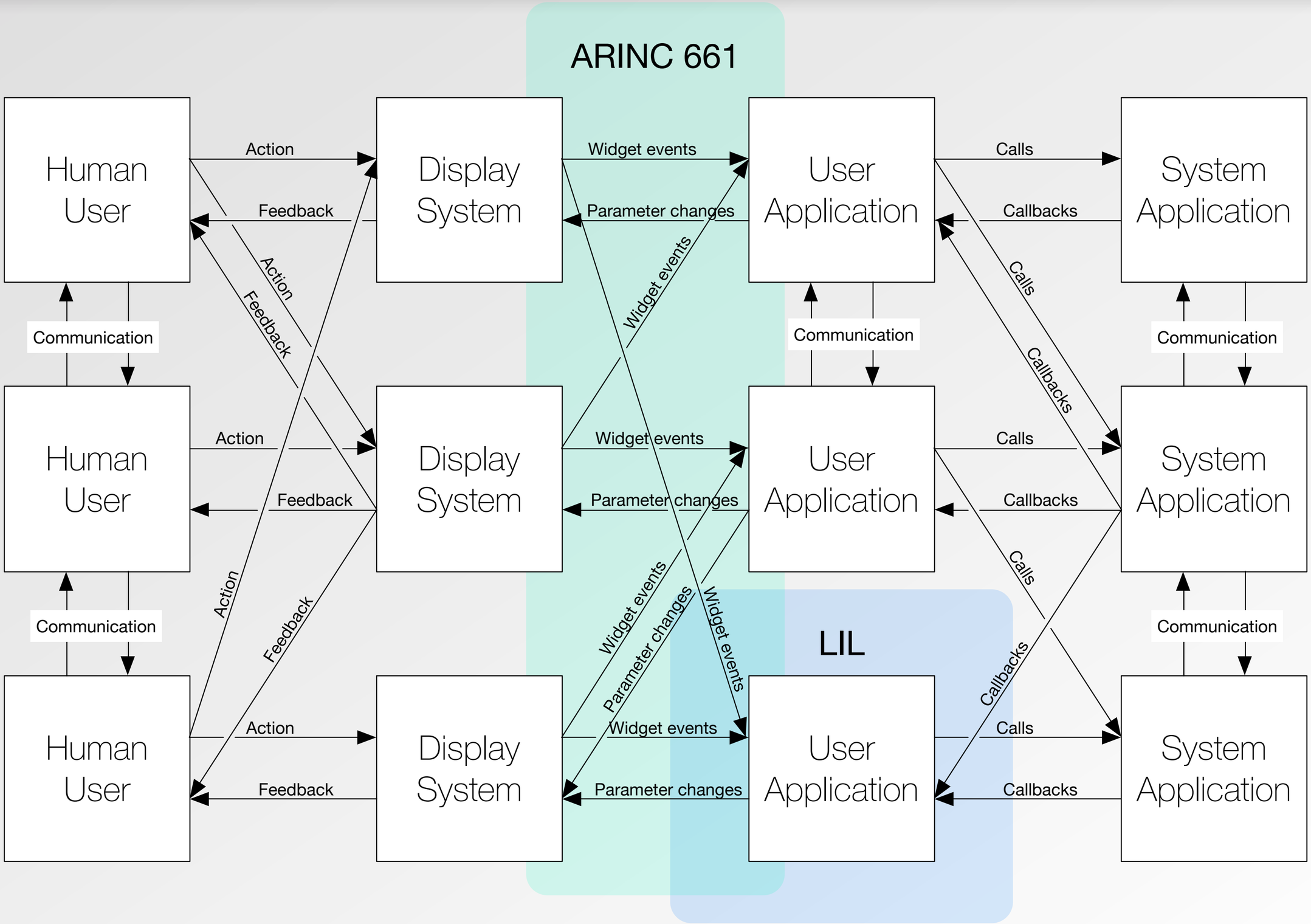
ONERA
THE FRENCH AEROSPACE LAB

Appendix 4



ONERA
THE FRENCH AEROSPACE LAB

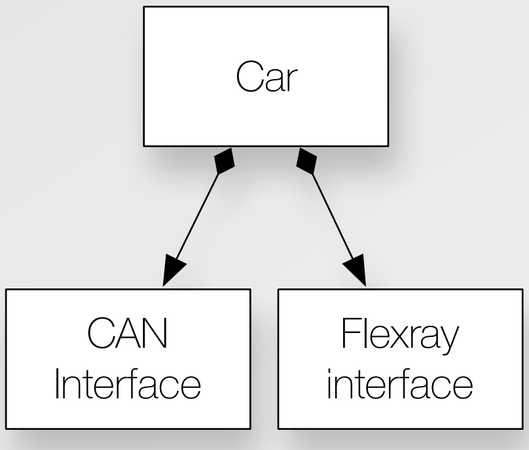
Appendix 5



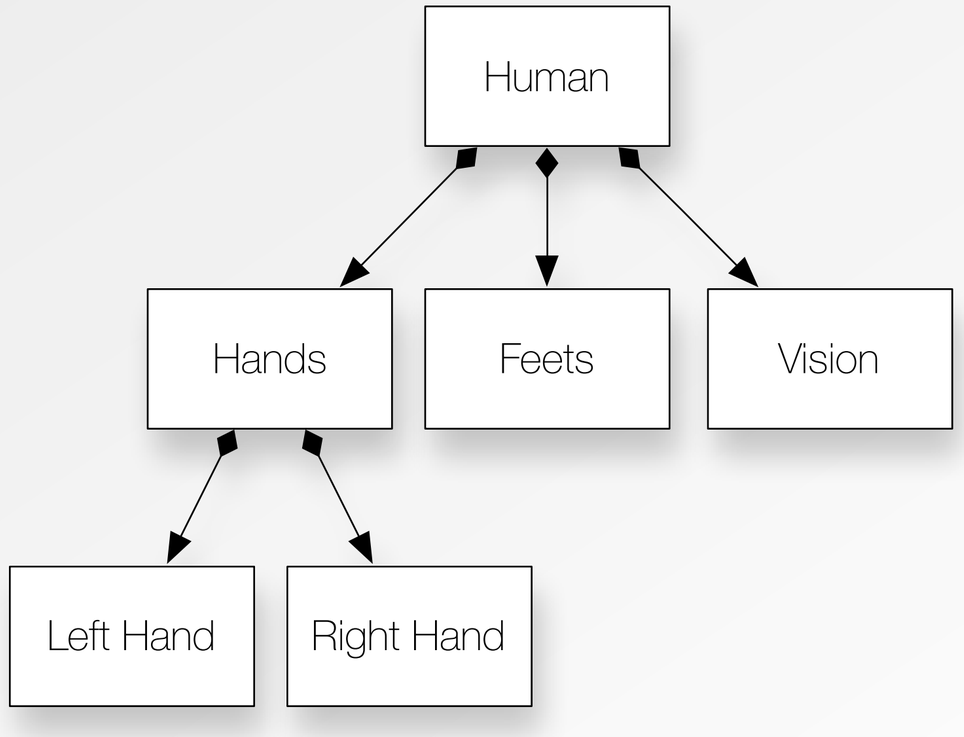
ONERA
THE FRENCH AEROSPACE LAB

Appendix 6

```
car actor :
  can actor
  flexray actor
```




```
human actor :
  hands actor
  feets actor
  vision actor
```



Appendix 7

Interactor Defintion

```
speedController interactor:
```

Structure

Actors

```
theDriver      : human actor
theCar         : car actor
```

Composition

```
theDashboard   : dashboard interactor
theControls    : speedControls interactor
thePedals      : pedals interactor
```

Data flow

```
increment      : void event from theControls
decrement      : void event from theControls
toggle         : boolean event from theControls
desiredSpeed   : number flow to theCar
actualSpeed    : number flow from theCar to theDashboard
selectedMode   : mode flow from theControls
desiredMode    : mode flow to theCar
actualMode     : mode flow from theCar to theDashboard
alert          : boolean flow to theDashboard
```

Behavior

```
on increment : desiredSpeed = desiredSpeed + 5
on decrement : desiredSpeed = desiredSpeed - 5
30 < desiredSpeed < 150
alert = actualSpeed > desiredSpeed or desiredMode != actualMode
desiredMode = if toggle then selectedMode else OFF
```