Pragmatic Incremental Approach to an Affordable Certification Process for RPAS - Building-up from core Safety Functions.

An example with a Smart Hybrid Parachute System

Laurent Ciarletta, Loïc Fejoz, Adrien Guenard, Nicolas Navet

RPAS:

- Raise safety concerns
- How can we increase safety?
- How can we have guarantees on the performances of RPAS?
- Can hardly use same processes and standards used in aeronautic industry for now

The idea:

- Use a software development tool-chain which could guarantee requirements

- Begin with a small set of safety functions

- Add safety incrementally

Contribution of this study:

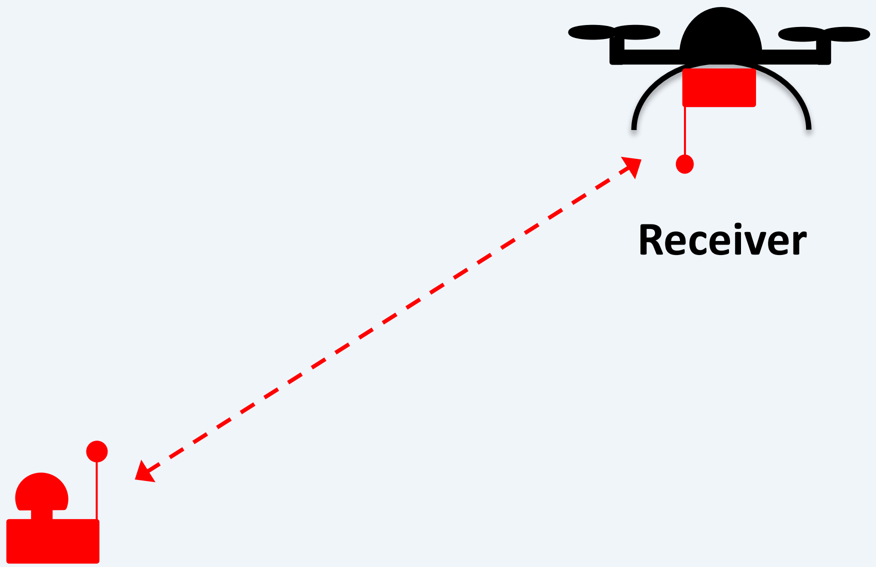
- Bring pragmatic solutions to develop provably safe software in a time and cost-affordable manner

- Add the minimum level of safety requirements to allow a safe-crash solution

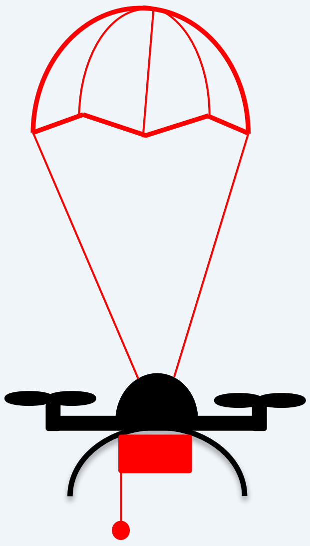
Use case: Intelligent parachute deployment system

Add-on to UAV

Independent safety module:
- own communication channel
- own computational unit
- own power supply



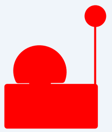
Use case: Intelligent parachute system



In case of emergency: on user demand or if link down

Emergency procedure:
- stops motors
- deploys parachute
- stops power supply

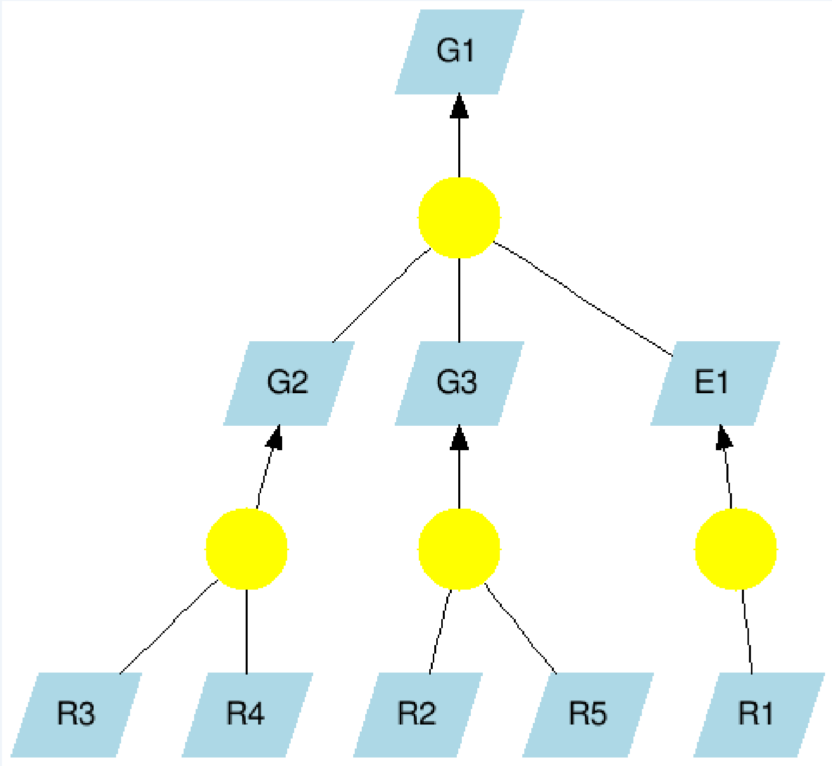
Receiver



Transmitter

« Red Button »

RTaW ReqLab : Requirements definition



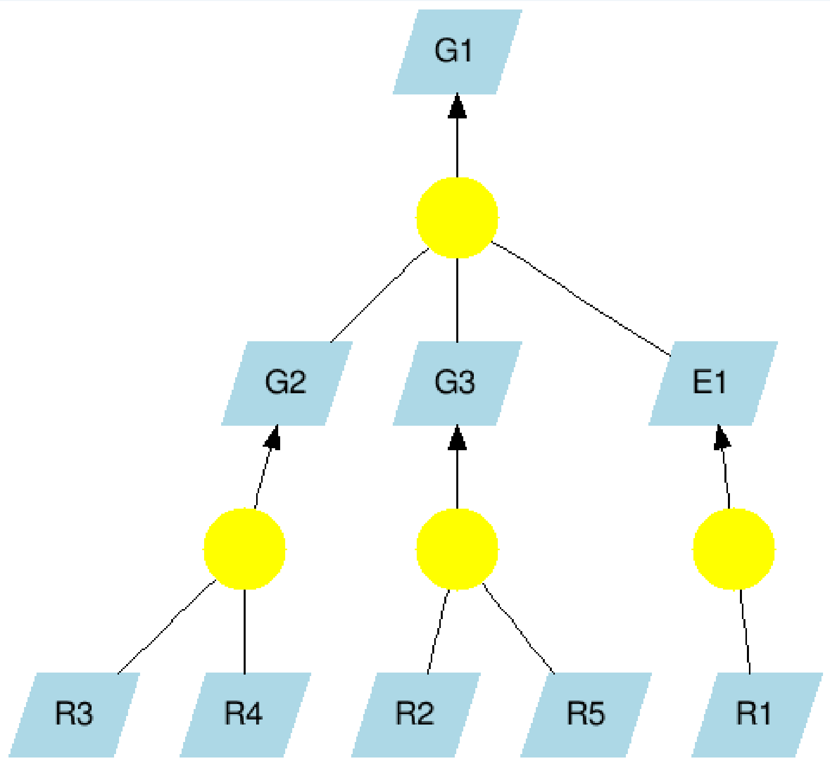
G1: Reduce property damage.

G2: Remote safety procedure shall deploy a parachute.

G3: When communication link loss is detected, the remote safety procedure shall be engaged.

E1: The pilot shall engage the remote safety procedure every time a hardware failure occurs, or
when an emergency is going to happen.

RTaW ReqLab : Requirements definition



G2: Remote safety procedure shall deploy a parachute.

[R3] The safety process shall turn the propellers off before deploying the parachute.

[R4] Once the safety process engaged, the parachute shall be deployed in less that 1.43s.

- **CPAL**: Cyber-Physical Action Language: model, simulate, verify and program embedded systems

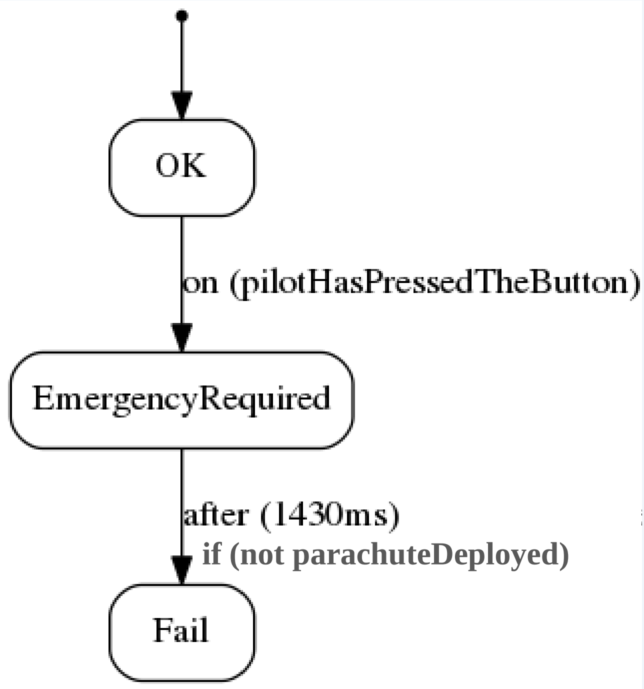
- Refines requirements to a specification:

  list of requirements which are SMART (Specific, Measurable, Assignable, Realistic and Testable)

- The fulfillment of SMART requirements can be verified in a dedicated CPAL task

Example: [R4] could be verified with the code shown

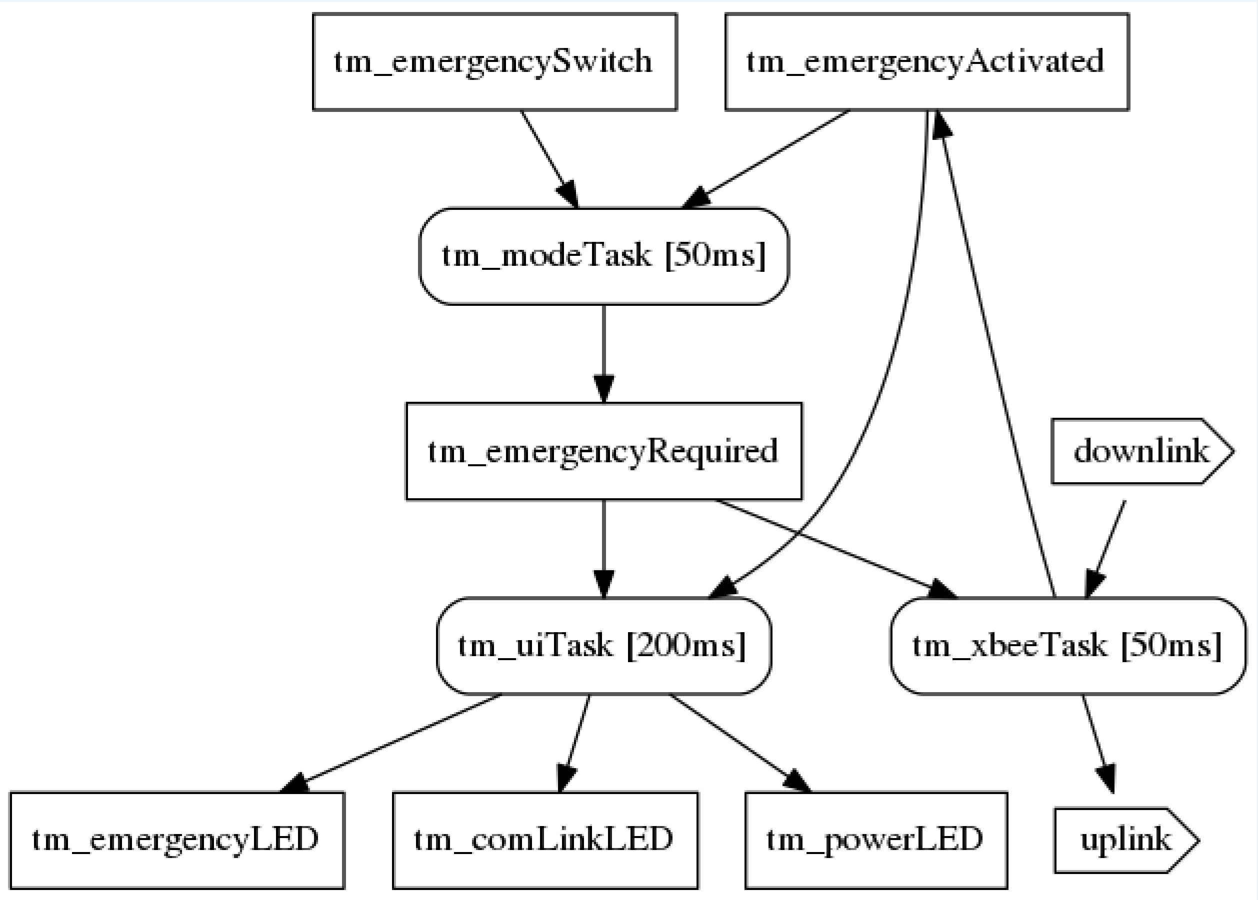
[R4] Once the safety process engaged, the parachute shall be deployed in less that 1.43s.




```
processdef R4Observer (
   in bool : pilotHasPressedTheButton,
   in bool : parachuteDeployed)
{
   state OK {
   }
   on (pilotHasPressedTheButton)
     to EmergencyRequired;
   state EmergencyRequired {
   }
   after (1430ms) if (not parachuteDeployed)
     to Fail;
   state Fail {
     /* println("R4 FAILED"); */
     assert(false);
   }
}
```

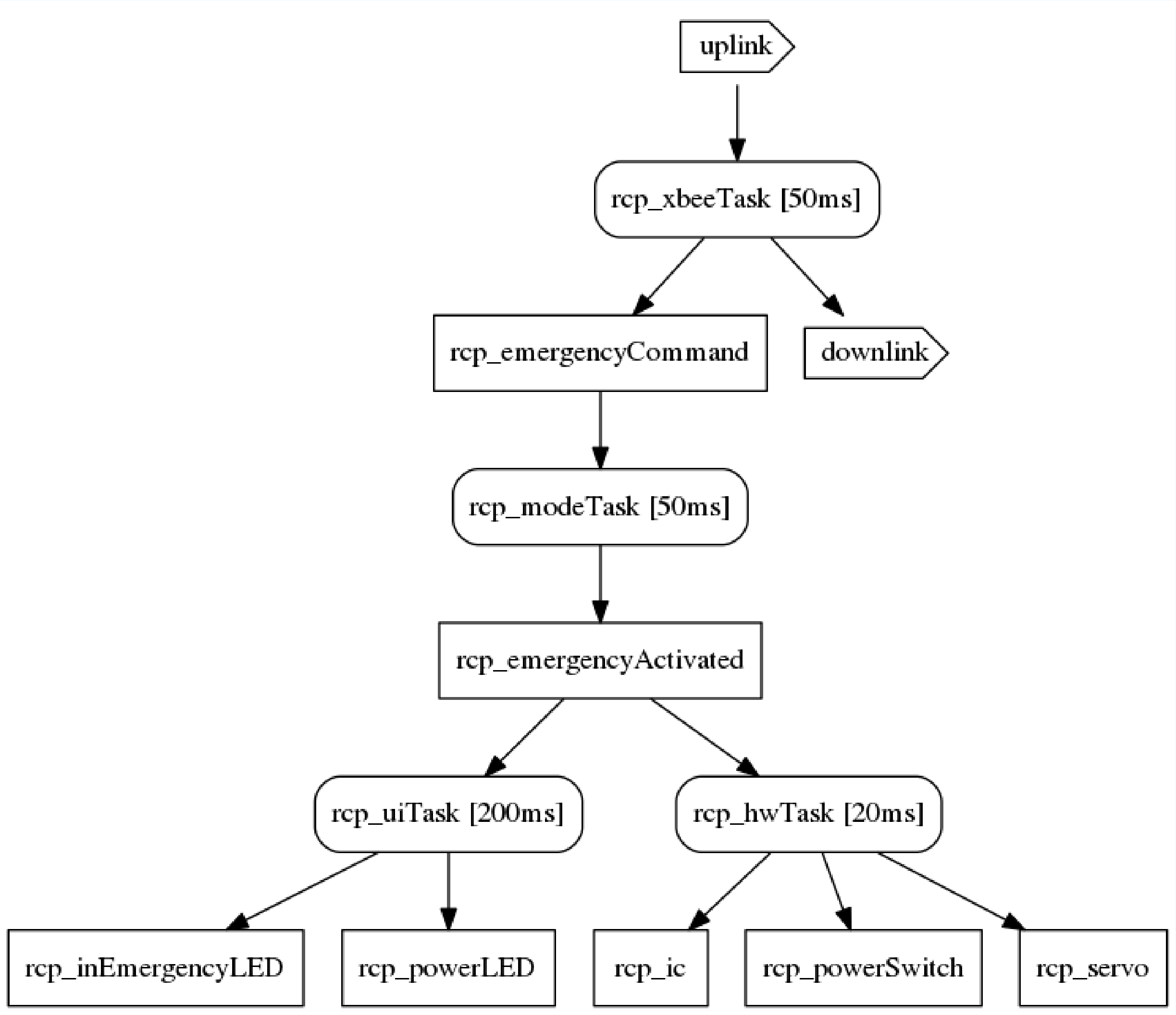
CPAL models of software architecture

Transmitter

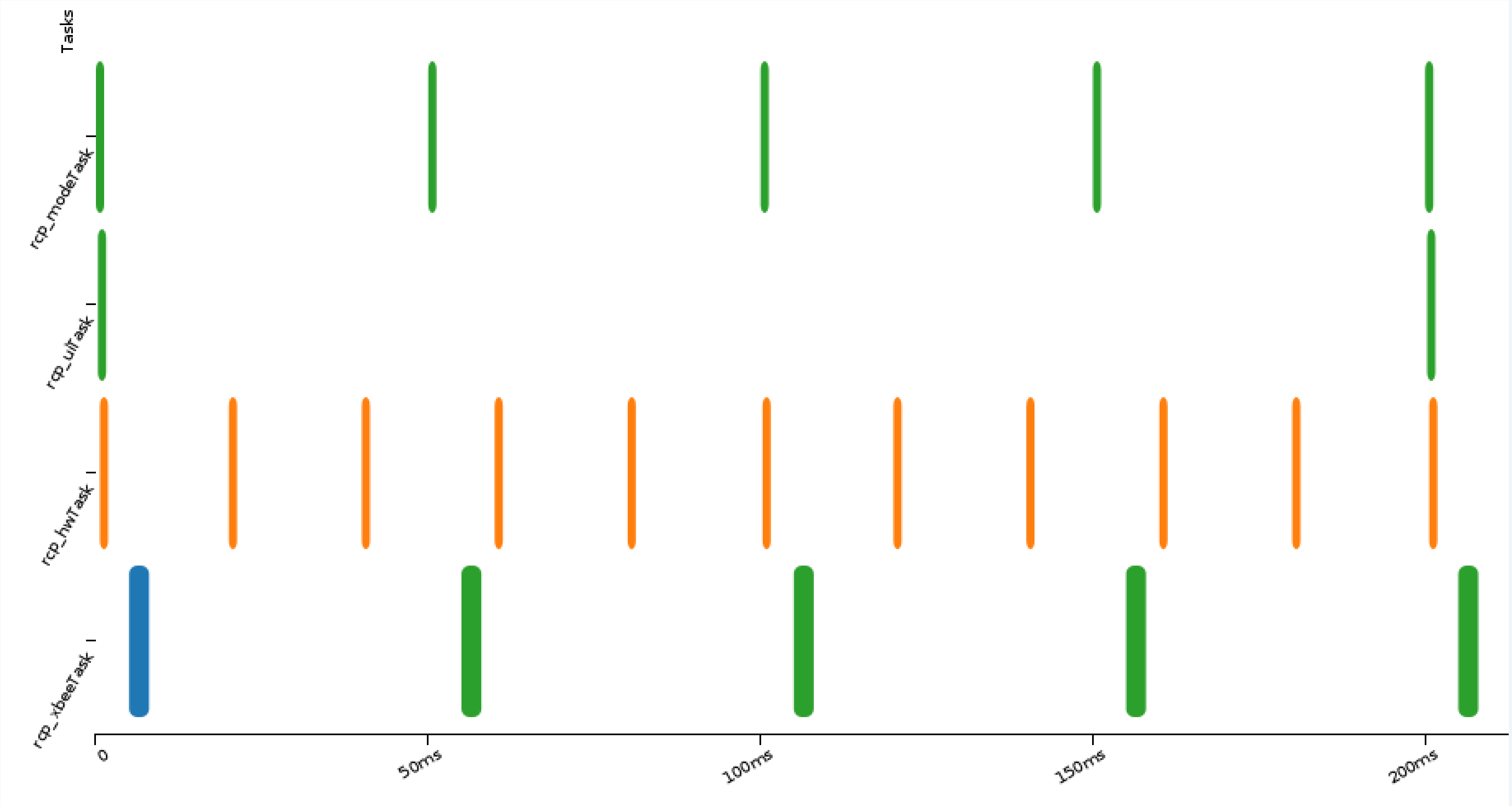


CPAL models of software architecture

Receiver



Gantt chart of the tasks execution



- Return of experience
- Short-term pragmatic solution to bring safety in RPAS
- CPAL development environment and RTaW ReqLab free to use at
  http://www.designcps.com and https://www.requirements.fr
- Models available
- Long term: adaptation and participation to regulation and standardisation effort