Design-driven Development of Dependable Applications: A Case Study in Avionics

Quentin Enard[1,2], Stéphanie Gatti[1,2], Julien Bruneau[1,2], Young-Joo Moon[1], *Emilie Balland[1]* and Charles Consel[1]

[1]Phoenix, Inria, France

[2]Thales Airborne Systems, France

Dependable Applications

- Dependability [Avizienis et al, 2004] includes the following attributes
  - Safety: absence of catastrophic consequences on the user(s) and the environment
  - Availability: readiness for correct service
  - Reliability: continuity of correct service
  - …
- Need of a certification process
  - Safety analysis
  - Specification of the functional and non-functional requirements (e.g., fault tolerance, QoS)
  - Conformance across the development stages (requirements traceability)

Case Study: an Avionics Flight Guidance Application

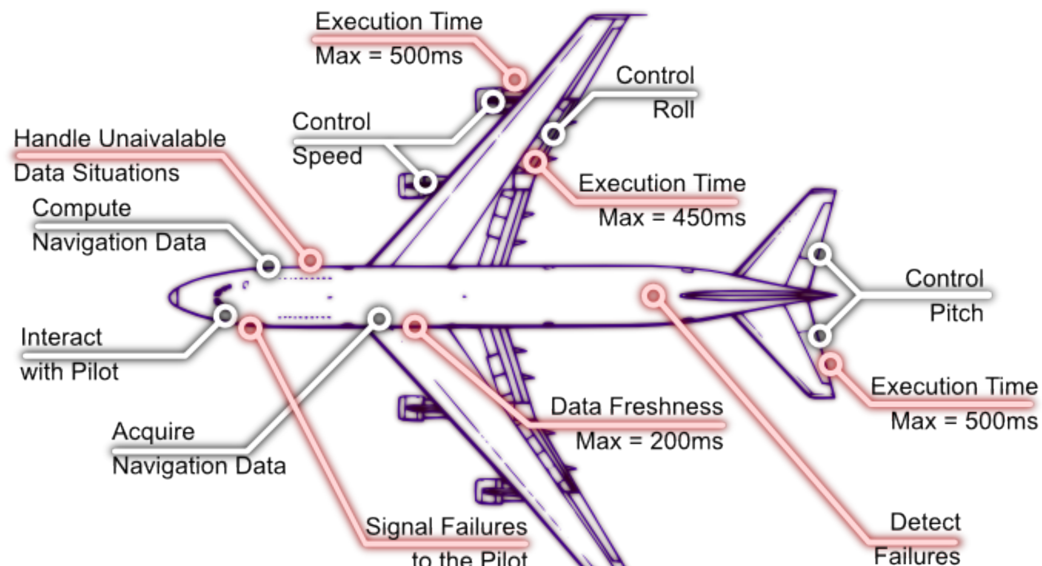
- Risk analysis ➔ Design Assurance Level (DAL) = A
- Definition of the requirements

Availability

Requirement	
Req1. The execution time of the heading mode must not exceed 650 ms.	It ensures that the frequency of computation of the heading mode does not lead to an unexpected behavior of the plane.
Req2. The freshness of the navigation data used by the application must be less than 200 ms.	The use of an outdated navi- ... lead to erro-
Req3. The ADIRU [1] must be replicated twice to tolerate at least one crash failure.	It ensures the availability of navigation data, despite the loss of a sensor.
Req4. Any malfunctioning or lost sensor must be signaled to the pilot, with identification of the probable cause.	Decisions taken by the pilot are based on information about the sensors' state.
Req5. A navigation mode must be deactivated if the required data is unavailable.	...ppropriate data, a ... mode cannot safely ... plane.
Req6. Information related to the activation/deactivation of navigation modes must be logged.	Application monitoring is used for maintenance.

Reliability

Safety



Certification Process

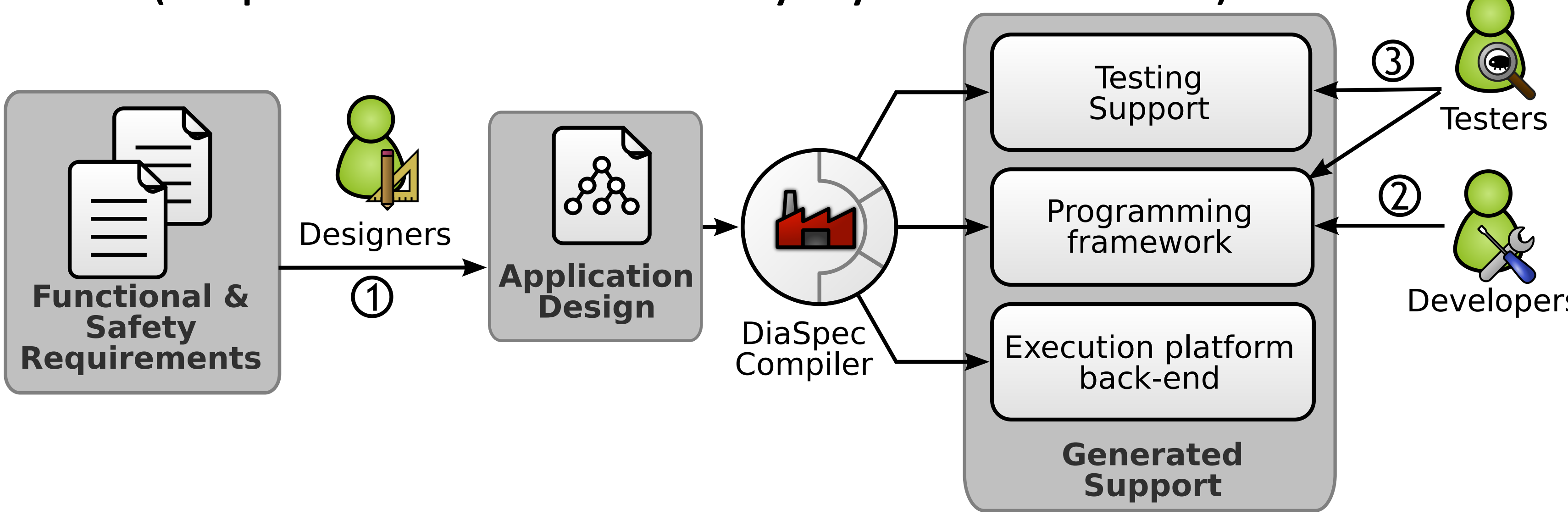
- Automotive or medical
  - No governmental certification authority
  - Own certification processes based on safety standards (*e.g.*, ISO 26262 for automotive systems)
- Avionics
  - Governmental certification authority (*e.g.*, FAA)
  - Strict certification process (*e.g.*, DO 178 B for software)
  - Peer review sessions and traceability documents (human intensive and potentially error prone)

Towards a Design-driven Approach

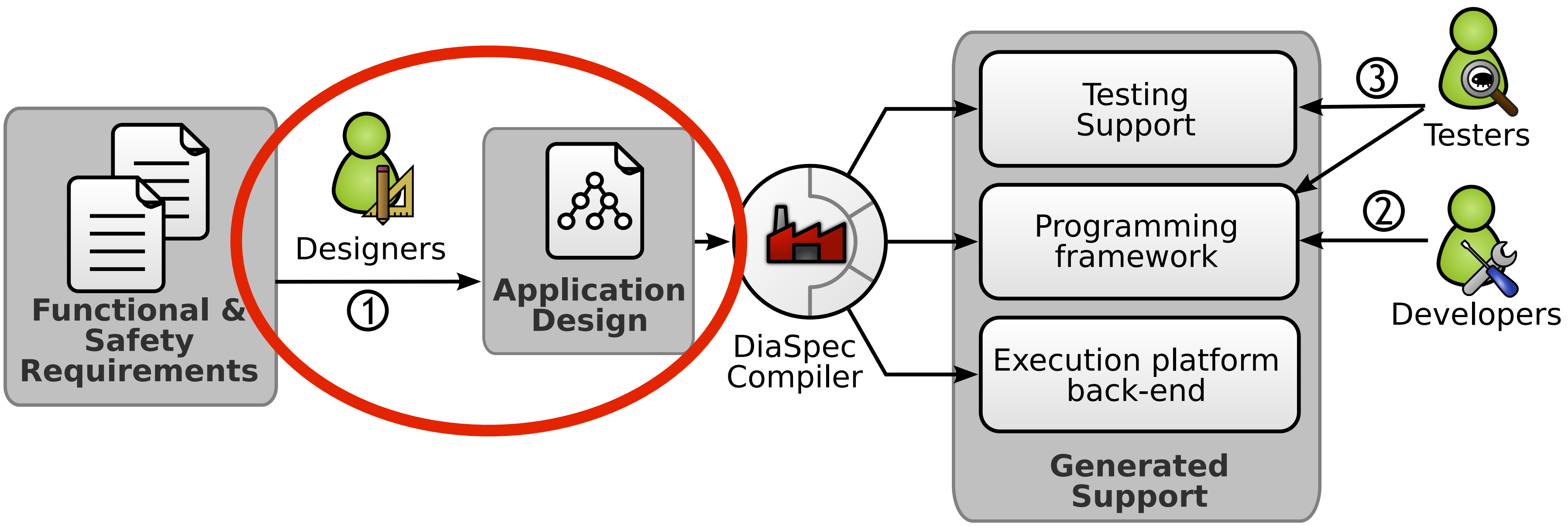
- Design artifact
  - Early verification of the requirements
  - Development guidance
  - Basis for tracing documents
- General-purpose approaches (*e.g.*, MDE)
  - Coherence between the multiple views of the system
  - Conformance across the development stages
- Need for a design framework that
  - Supports the development process
  - Guides the certification process in a **systematic** manner

Design Framework

- DiaSuite: a development tool-suite based on a specific design paradigm
  - A design language dedicated to this design paradigm [ICSE'11]
  - Covers both functional and non-functional aspects of an application [OOPSLA'10, FASE'11]
  - Generation of dedicated development support (requirement traceability by construction)

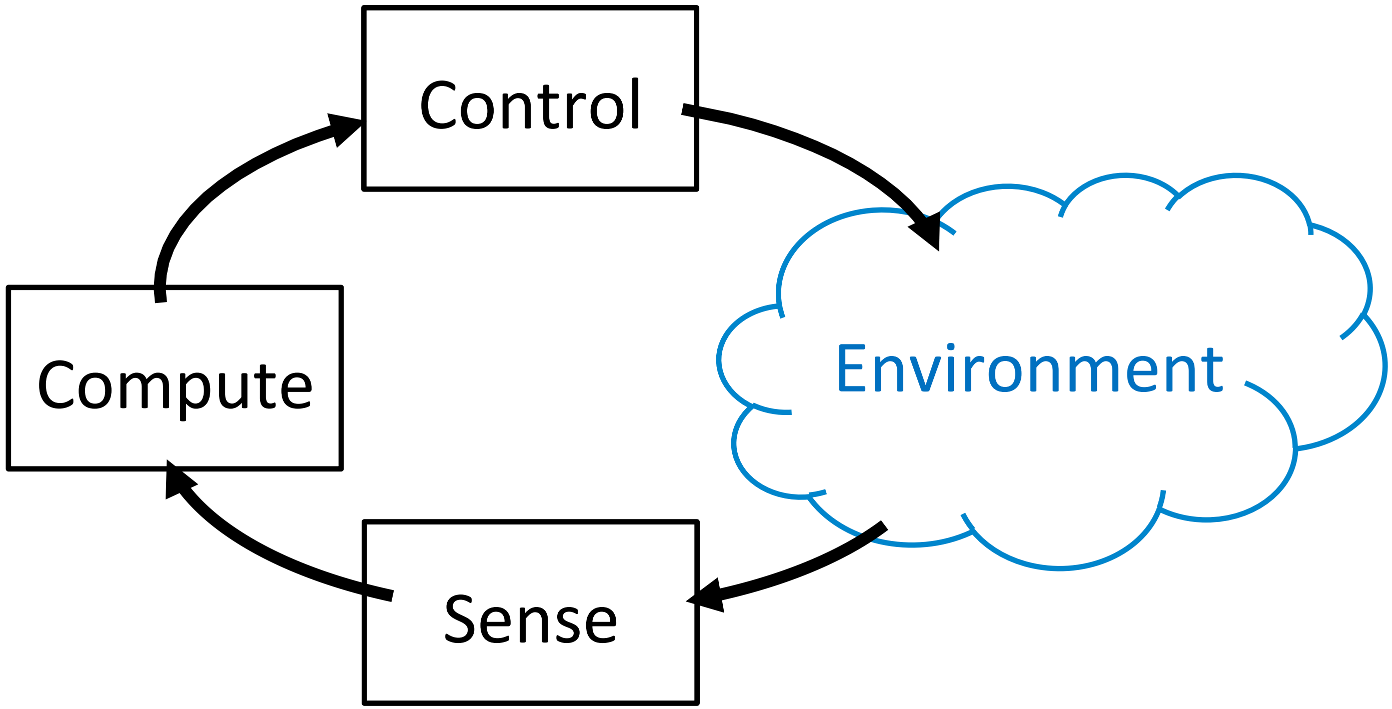


Design



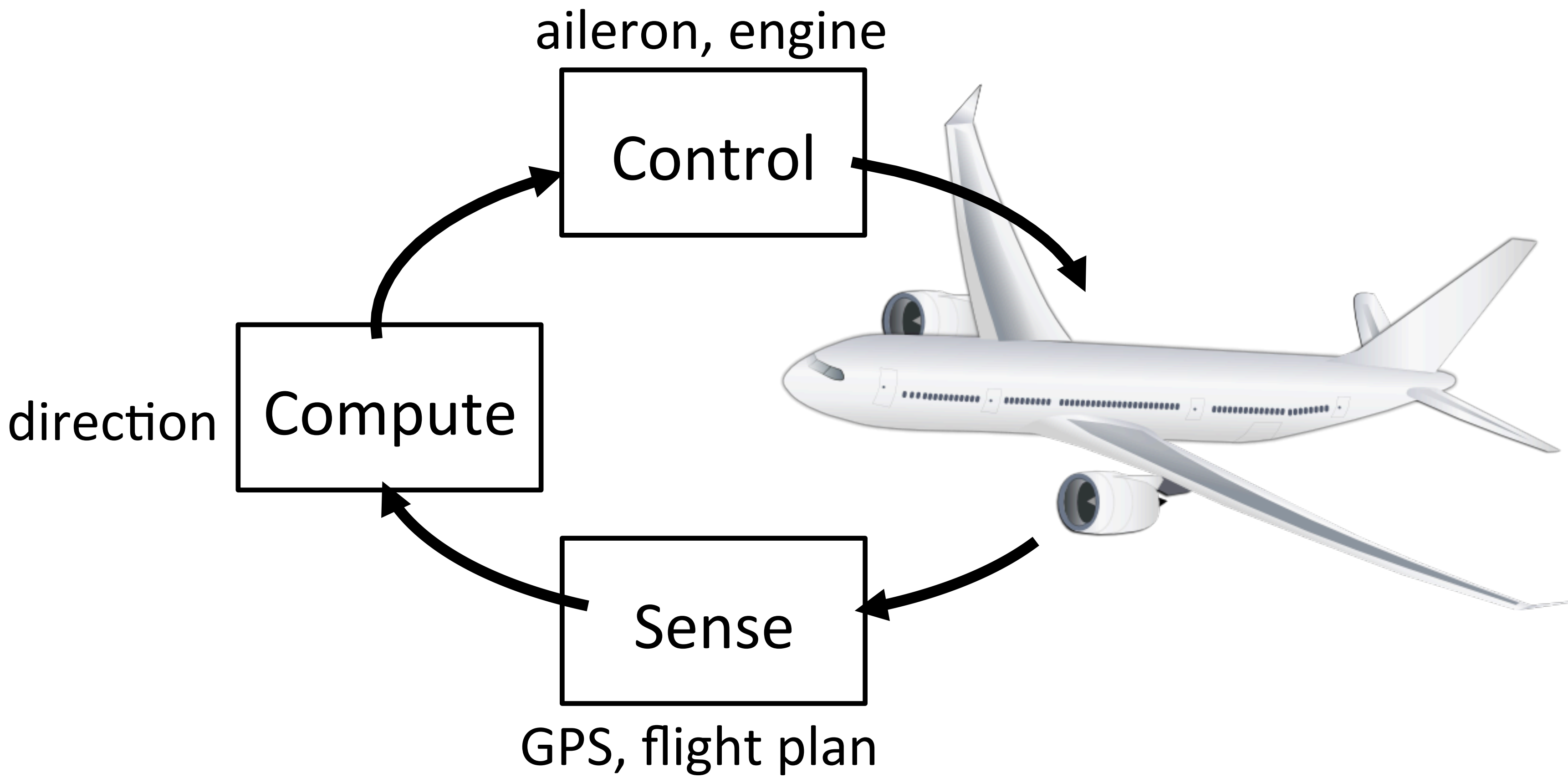
Functional &
Safety
Requirements
Designers
①
Application
Design
DiaSpec
Compiler
Testing
Support
Programming
framework
Execution platform
back-end
Generated
Support
③
Testers
②
Developers

Sense/Compute/Control (SCC) Software System

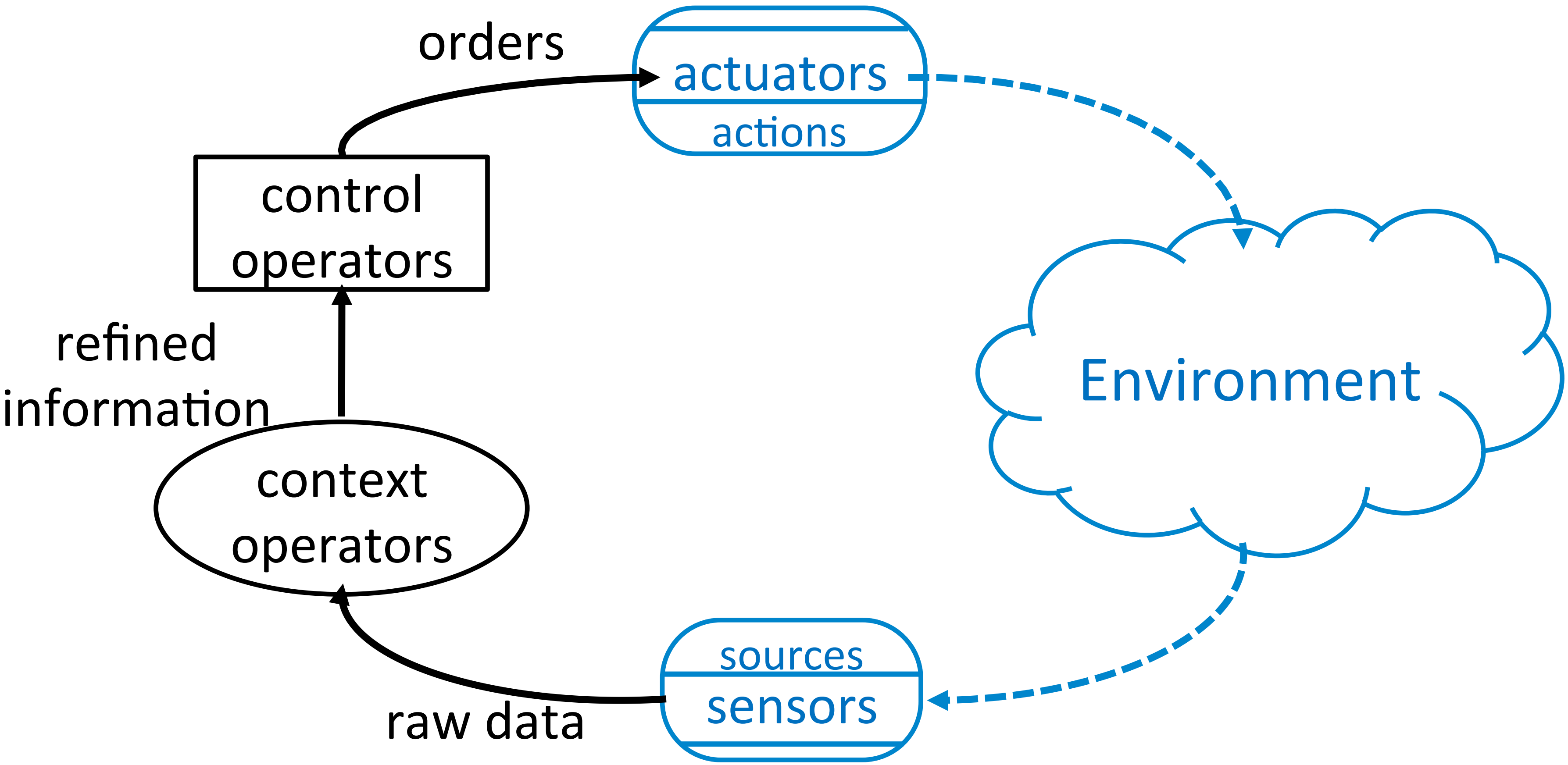


[Taylor et al., Software Architecture: Foundations, Theory, and Practice, 2009]

Sense/Compute/Control (SCC) Software System

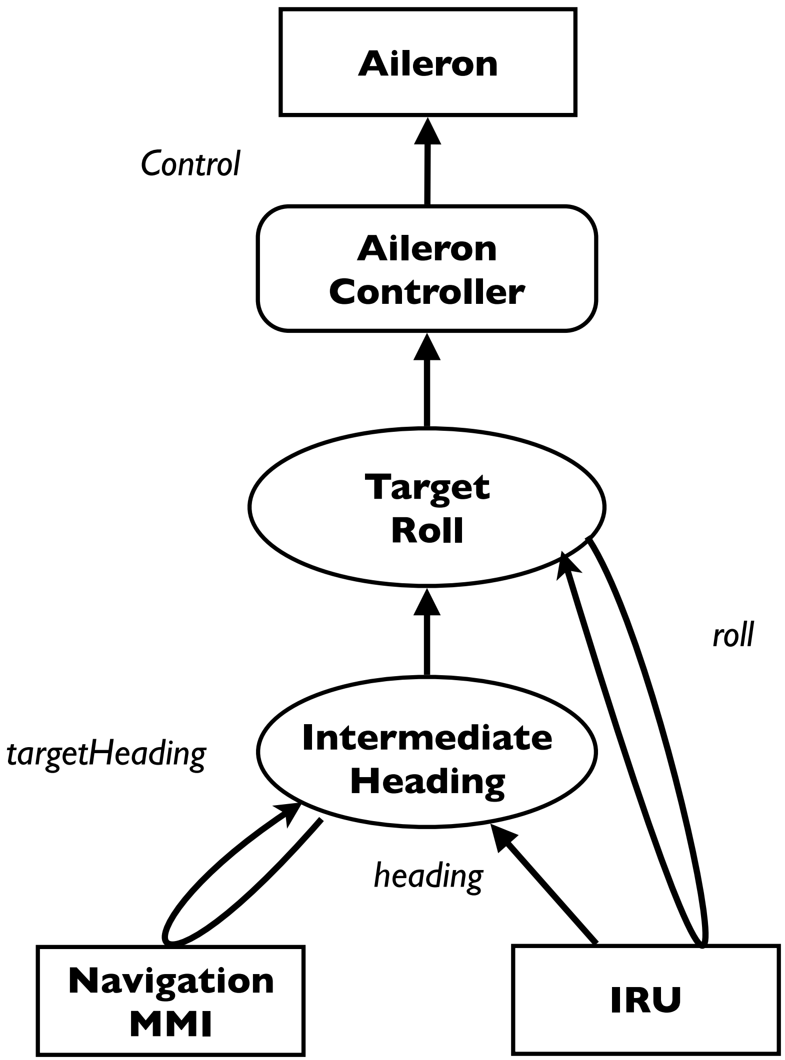


The SCC Architectural Style



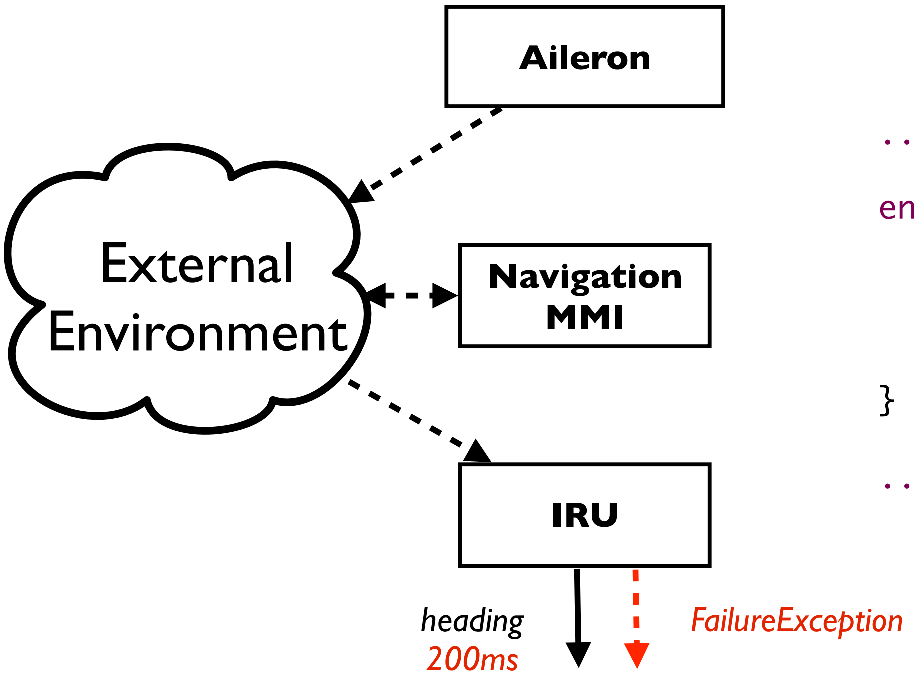
Functional Design

The Heading Mode




```
entity Aileron {
      action Control;
}

entity NavigationMMI {
      source targetHeading as Float;
}

entity IRU {
      source heading as Float;
      source roll as Float;
      action Deactivate;
}

context IntermediateHeading as Float {
      when provided heading from IRU;
      get targetHeading from NavigationMMI;
      always publish;
}

...
```

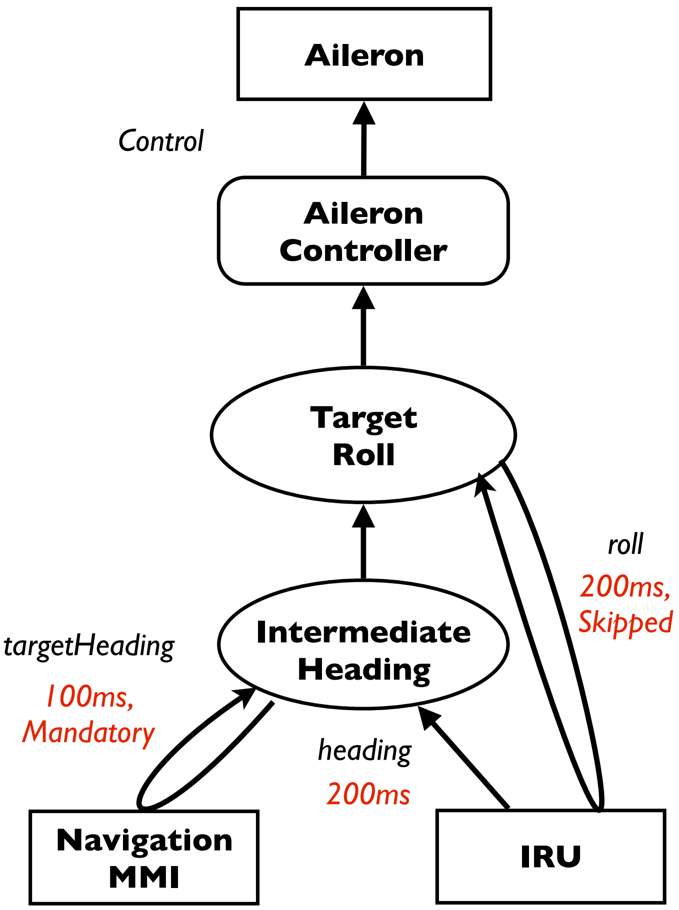
Non-functional Design



From Req. 2 and Req. 3

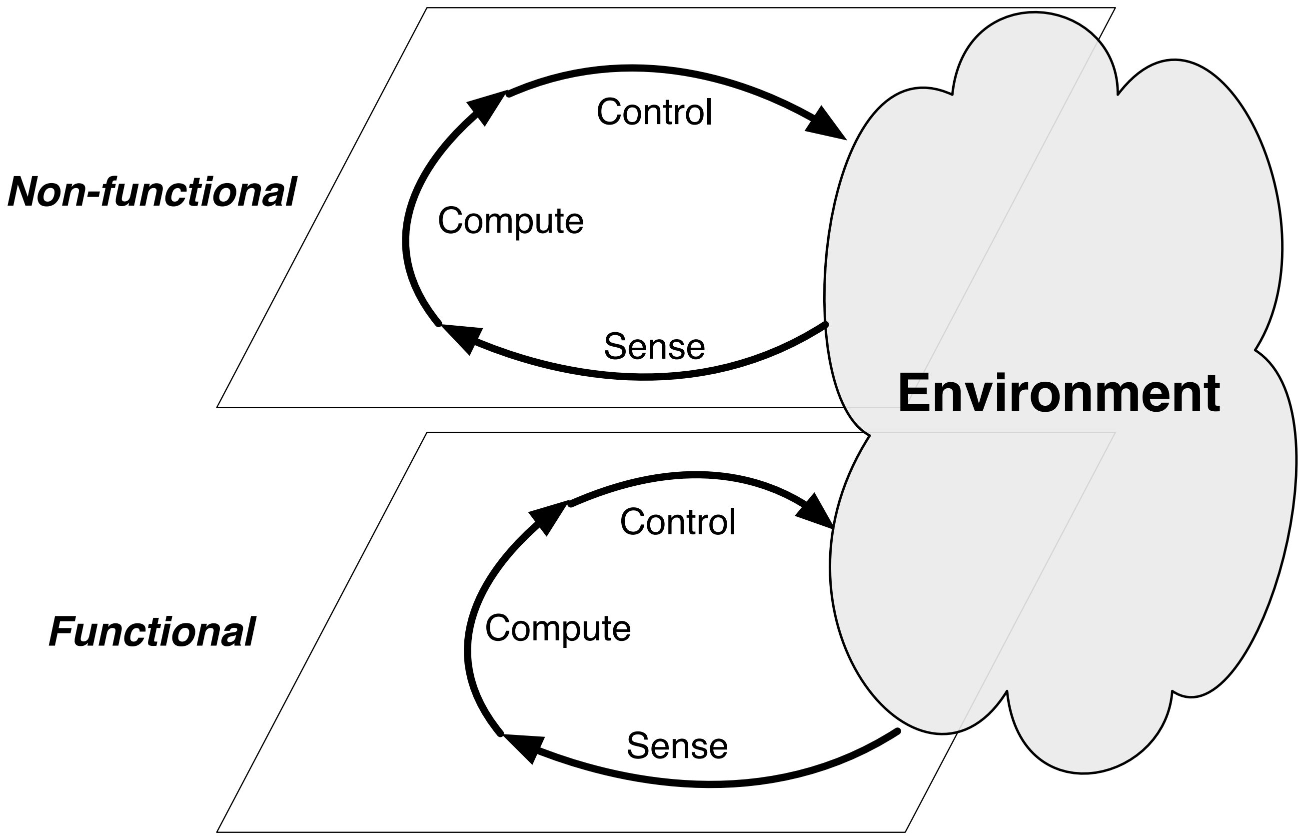
```
...

entity IRU {
    source heading as Float [frequency 200 ms];
    source roll as Float;
    action Deactivate;
    raises FailureException;
}

...
```

Non-functional Design

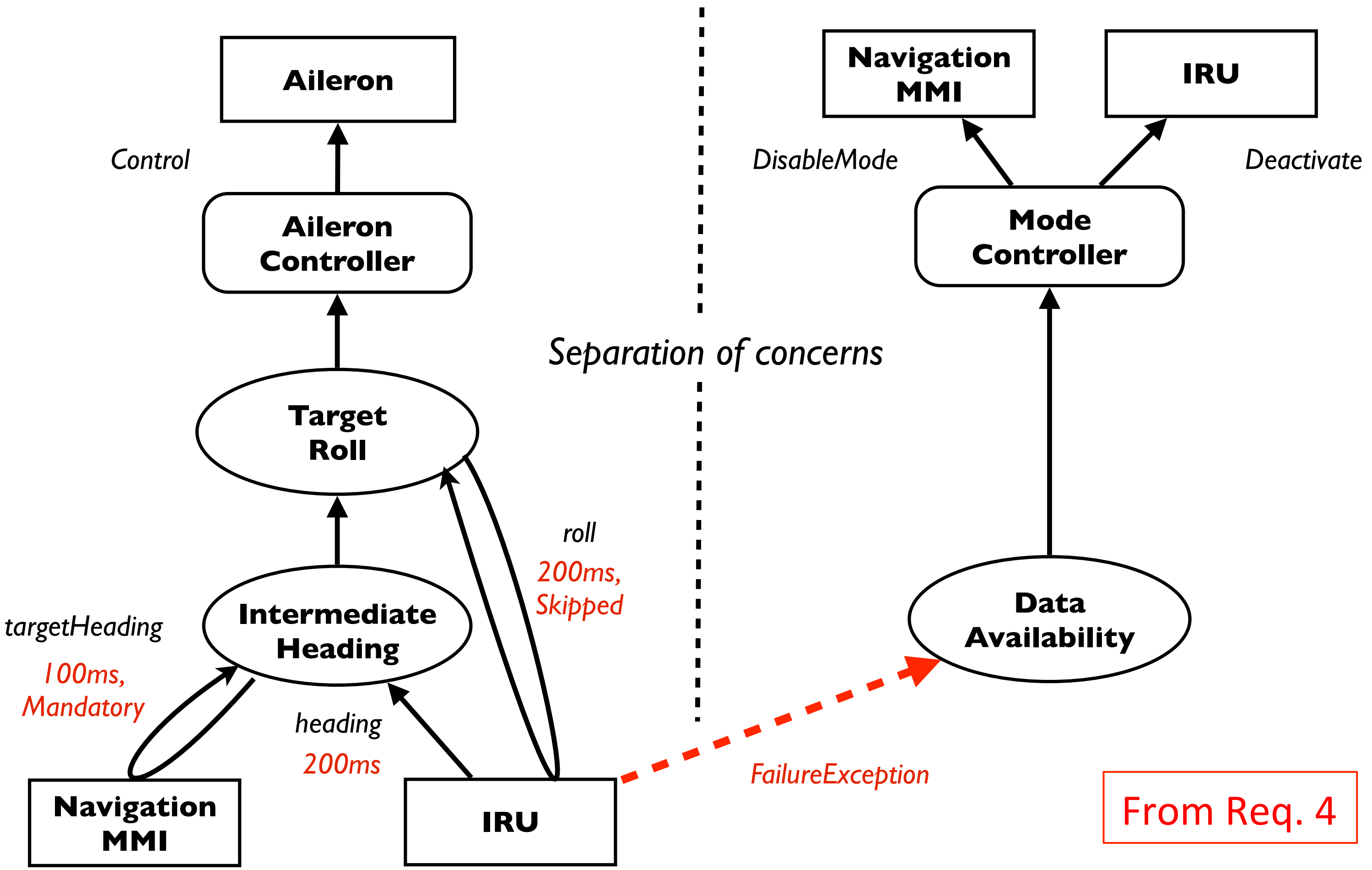



```
...

context IntermediateHeading as Float {
      when provided heading from IRU;
      get targetHeading from NavigationMMI
             in 100 ms [skipped catch];
      always publish;
}

context TargetRoll as Float {
      when provided IntermediateHeading;
      get roll from IRU
             in 200 ms [mandatory catch];
      always publish;
}

...
```

Layered View of the SCC Paradigm



Multi-layer Design



From Req. 4

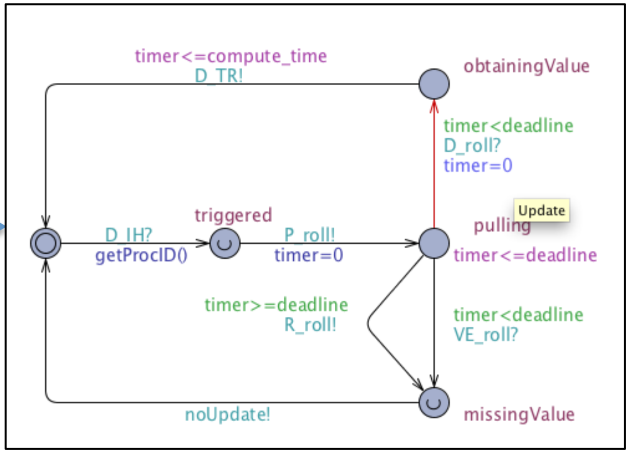
Verification at Design Time

- Early verification of the requirements
  - Functional (behavioral invariants)
  - Non-functional (time-related properties, error handling)
- Generation of a formal model from the application design
- Traceability by construction (generation of a dedicated programming framework)

Verification Support

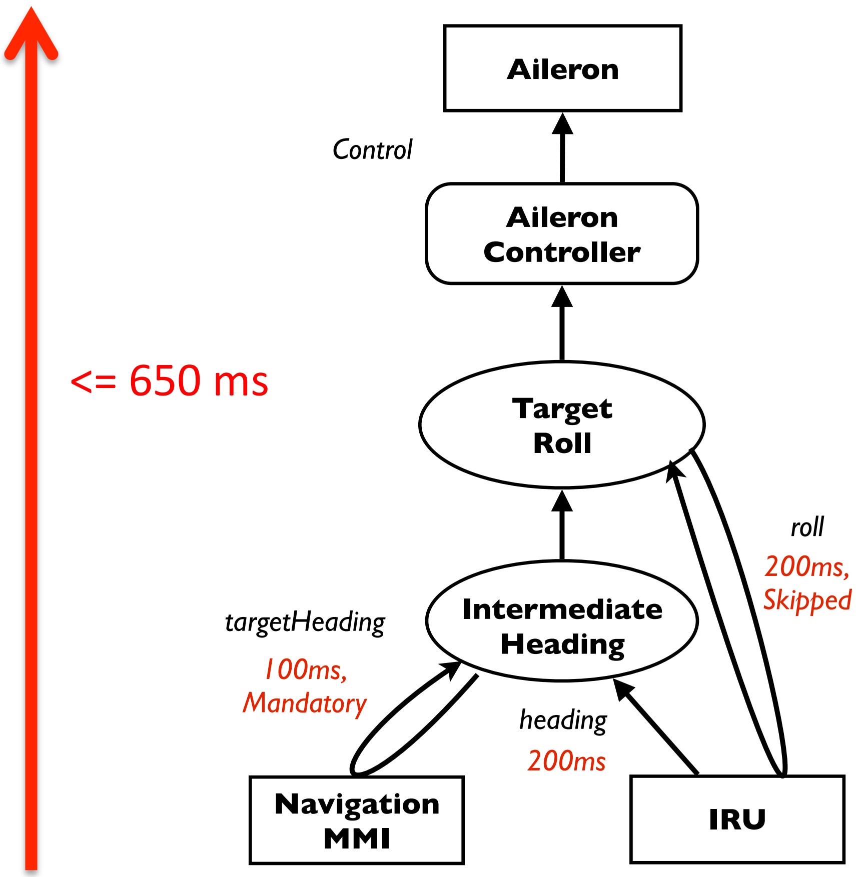
- Translation of the design into a network of timed automata
  - Automatic generation of a UPPAAL model
  - Each component = a timed automata

```
context TargetRoll as Float {
      when provided IntermediateHeading;
      get roll from IRU in 200 ms[mandatory catch]
      always publish;
}
```



- Translation of the high-level requirements into temporal properties (by hand)
- Verification of the properties using the UPPAAL model checker

Time-related Properties

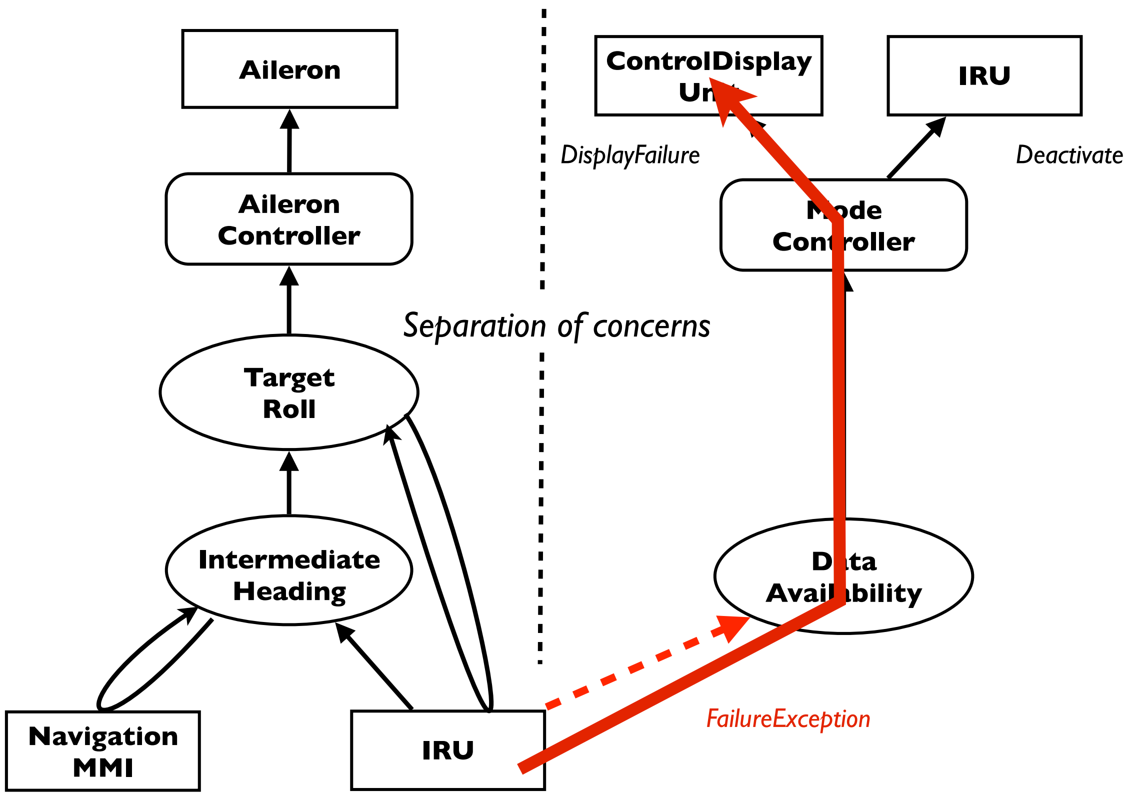


Requirement	Description
[illegible] the heading mode must not exceed 650 ms.	It ensures that the frequency of computation of the heading mode does not lead to an unexpected behavior of the plane.
Req2. The freshness of the navigation data used by the application must be less than 200 ms.	The use of an outdated navigation data can lead to erroneous decisions.
Req3. The ADIRU [1] must be replicated twice to tolerate at least one crash failure.	It ensures the availability of navigation data, despite the loss of a sensor.
Req4. Any malfunctioning or lost sensor must be signaled to the pilot, with identification of the probable cause.	Decisions taken by the pilot are based on information about the sensors' state.
Req5. A navigation mode must be deactivated if the required data is unavailable.	Without appropriate data, a navigation mode cannot safely control the plane.
Req6. Information related to the activation/deactivation of navigation modes must be logged.	Application monitoring is used for maintenance.

- Generation of an observer automata model to ease the verification of time-related properties

A[] (!(heading.FailureDetected && roll.FailureDetected) imply ! observer.Timeout)

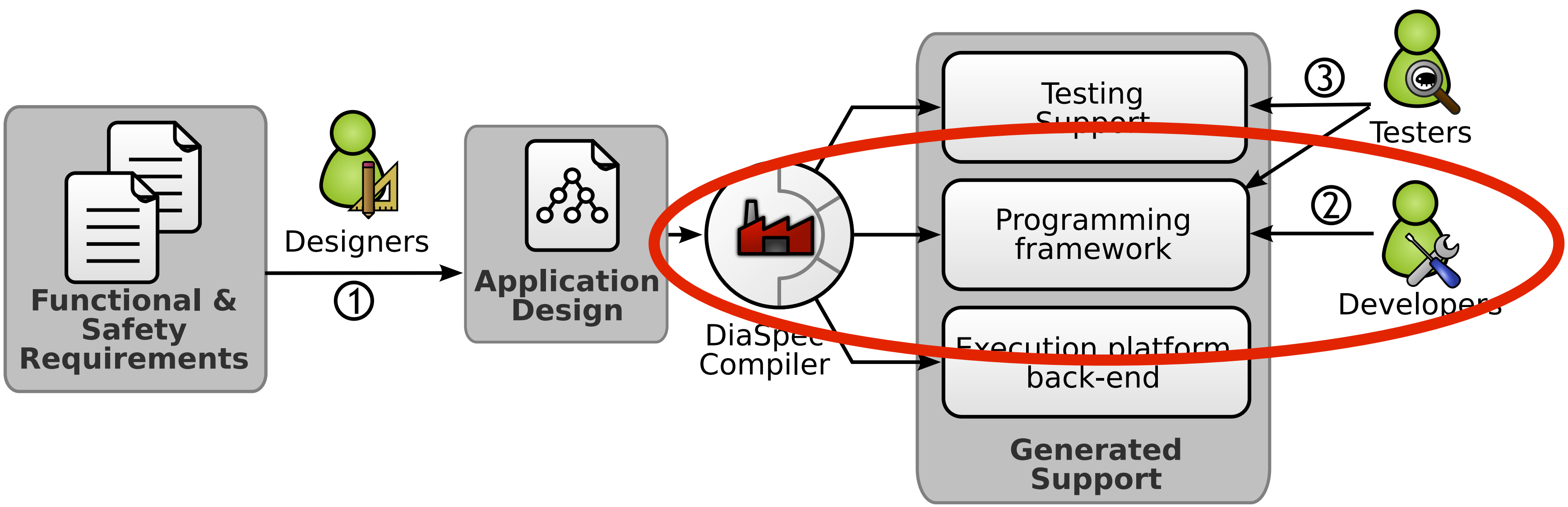
Error-handling Properties



Requirement	Description
Req1. The execution time of the heading mode must not exceed 650 ms.	It ensures that the frequency of computation of the heading mode does not lead to an unexpected behavior of the plane.
Req2. The freshness of the navigation data used by the application must be less than 200 ms.	The use of an outdated navigation data can lead to erroneous decisions.
Req3. The ADIRU[1] must be replicated twice to tolerate at least one crash failure.	It ensures the availability of navigation data, despite the loss of a sensor.
Req4. Any malfunctioning or lost sensor must be signaled to the pilot, with identification	Decisions taken by the pilot are based on information about the sensors' state.
Req5. A navigation mode must be deactivated if the required data is unavailable.	Without appropriate data, a navigation mode cannot safely control the plane.
Req6. Information related to the activation/deactivation of navigation modes must be logged.	Application monitoring is used for maintenance.

A[] (IRU.failureException → A<> controlDisplayUnit.DisplayFailure

Implementation



Programming Support

```
context IntermediateHeading as Float {
    when provided heading from IRU;
    get targetHeading from NavigationMMI;
    always publish;
}
```

Generation

```
public abstract class AbstractIntermediateHeading {
  [ ... ]

  public abstract Float onHeadingFromIRU(
       Float heading,
       Binding binding);
   ...
}
```

Implementation

```
public class IntermediateHeading extends AbstractIntermediateHeading {
  private PIDController controller;

  @Override
  public Float onHeadingFromIRU(Float heading, Binding binding,
       GetContext getContext) {
    NavigationMMI mmi = binding.navigationMMI();
    Float targetHeading = mmi.getTargetHeading();
    return controller.update(Config.PERIOD, targetHeading, heading.value(), 0);
  }
}
```

Error-handling Support

```
context IntermediateHeading as Float {
    when provided heading from IRU;
    get targetHeading from NavigationMMI in
200ms [mandatory catch]
    always publish;
}
```

Generation

```
public abstract class AbstractIntermediateHeading {

  public abstract Float onHeadingFromIRU(
       Float heading,
       Binding binding)

  public abstract Float getTargetHeading(...,
                 IRUHeading continuation);

}
```

Implementation

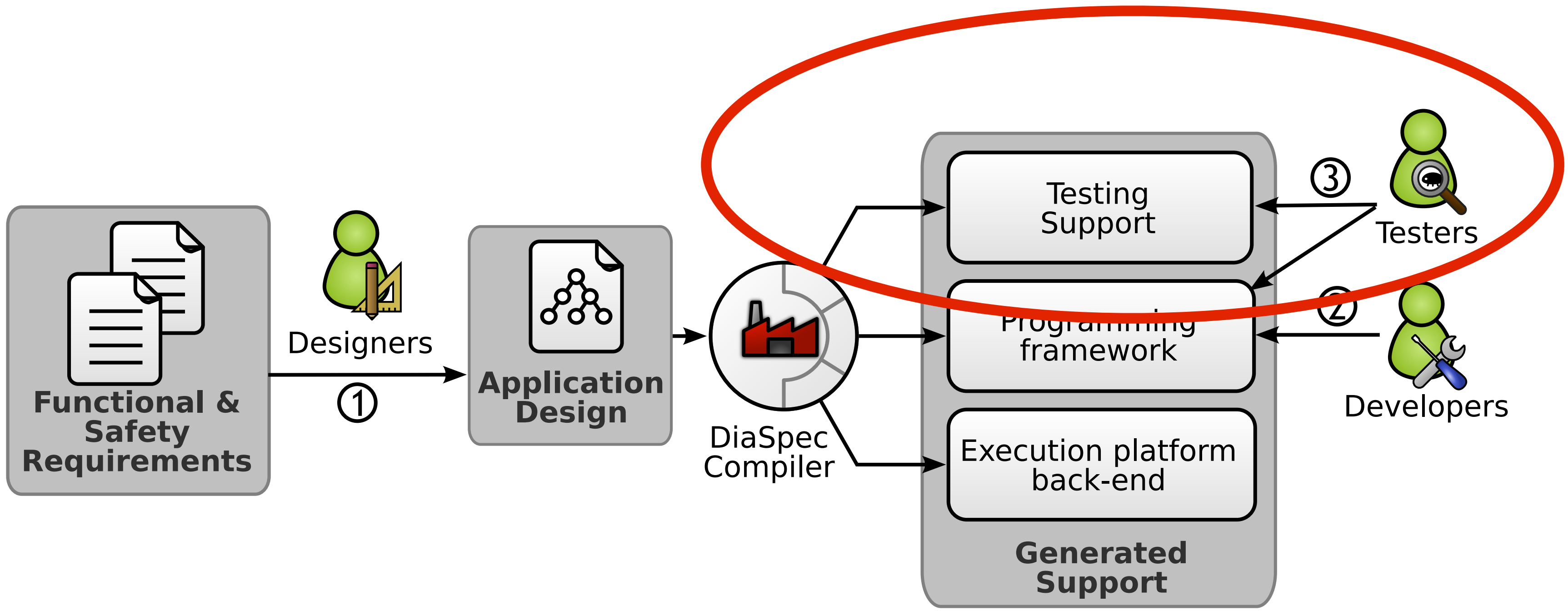
```
public class IntermediateHeading extends AbstractIntermediateHeading {
  @Override
  public Float onHeadingFromIRU(Float heading, Binding binding) {
    NavigationMMI mmi = binding.navigationMMI();
    Float targetTargetHeading = mmi.getTargetHeading(
      new TargetHeadingContinuation() {
        public Float onError() {
          return DEFAULT_VALUE;}
      }
    );
    return ...
  }
}
```

Mandatory error handling

Timing-Constraints Monitoring Support

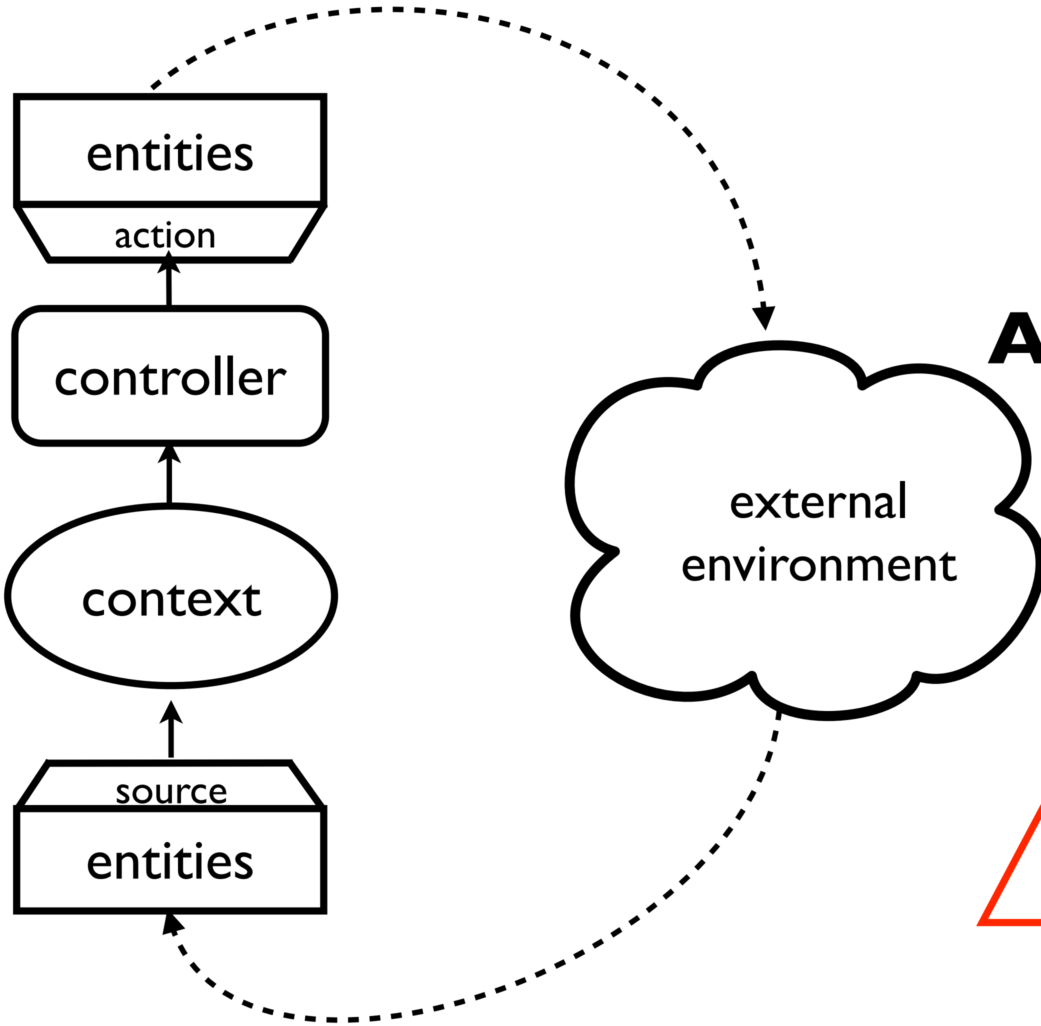
- Generation of monitors dedicated to timing-constraints verification [FASE'11]
  - Transparent for the developer
  - Traceability of the time-related requirements (needed for the certification process)
- Implementation of handlers for the violation of the timing constraints
  - Testing
  - Runtime

Testing



Functional &
Safety
Requirements
Designers
①
Application
Design
DiaSpec
Compiler
Testing
Support
Programming
framework
Execution platform
back-end
Generated
Support
③
Testers
②
Developers

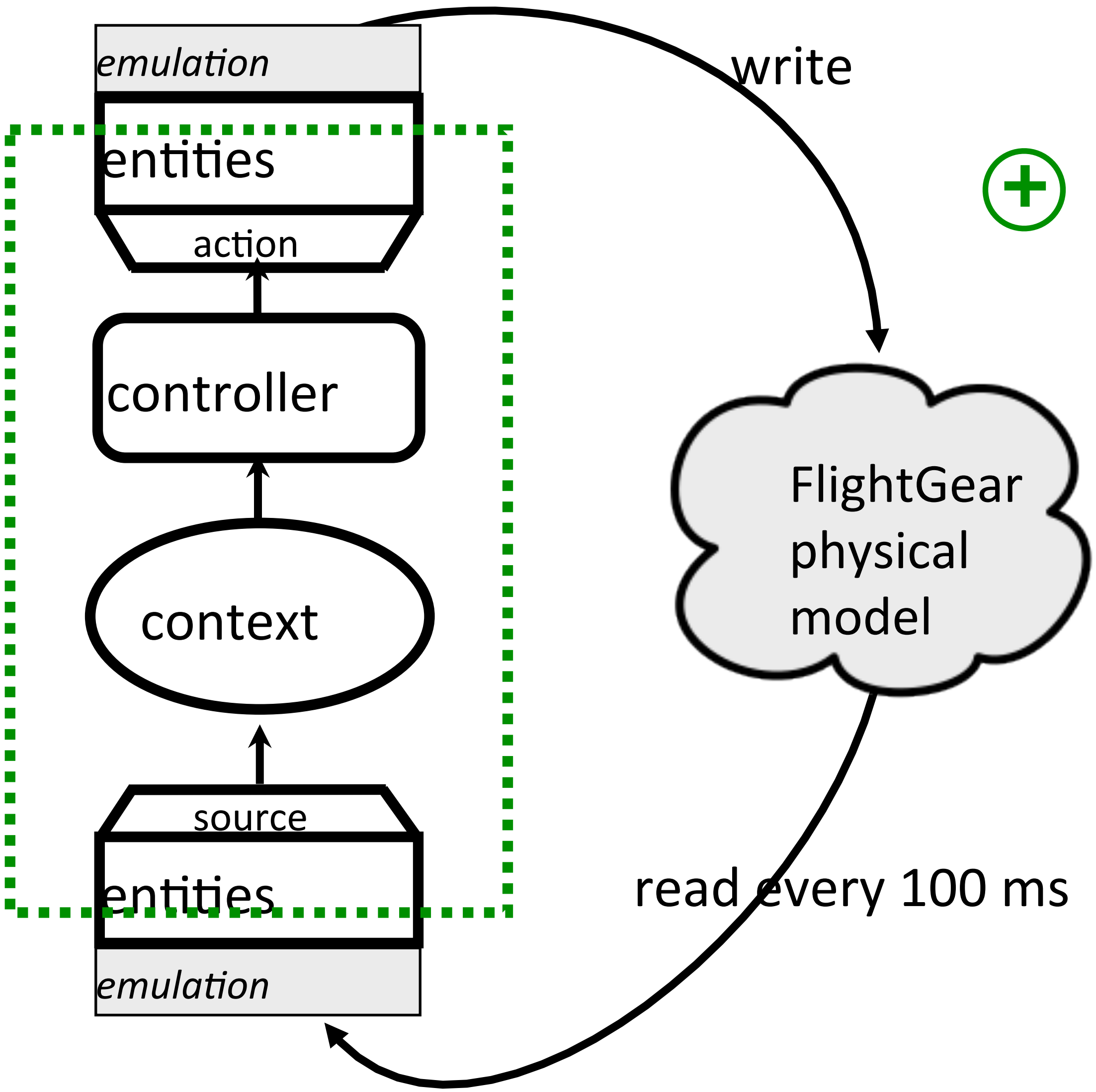
Testing



Environment
=
Aircraft physical system

How to test
the flight guidance
application?

Simulation-based Testing Support

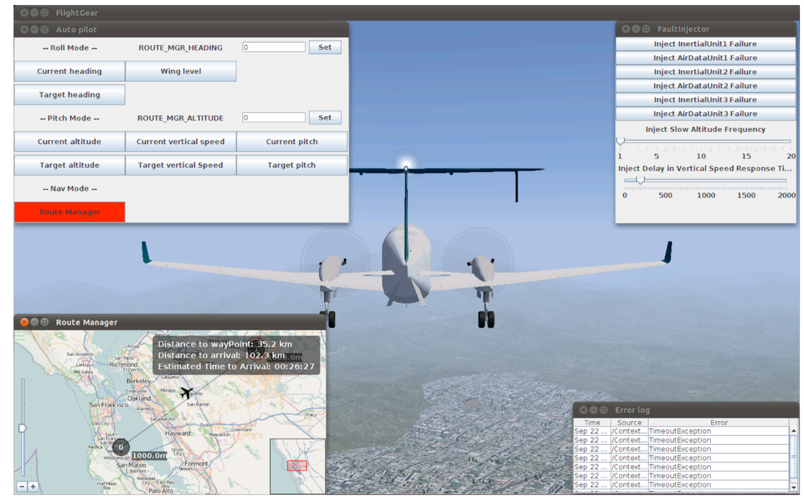


Physical system simulation

⊕ **Simulation without any modification of the application logic**

Simulation-based Testing Support

- Functional aspects
  - Simulated external environment
  - Validation of the functional implementation using mock-up entities
- Non-functional aspects
  - Fault injection
  - Verification of the exceptional component behavior




```java
public class SimulatedIRU extends AbstractIRU
                          implements SimulatorListener {

  public SimulatedIRU(FGModel model) {
    model.addListener(this);
  }

  public void simulationUpdated(FGModel model) {
    publishPosition(model.getInertialPosition());
  }
}
```

Conclusion

- A **design language** to describe both the functional and non-functional aspects of a dependable SCC application
- A **generative approach** that leverages the design to provide
  - development support (high-level programming framework, automatic traceability of the requirements)
  - validation support (formal model for the early verification of the requirements, simulation-based testing)

http://diasuite.inria.fr