Building a Software Requirements Specification and Design for an Avionics System

An Experience Report

Andrés Paz and Ghizlane El Boussaidi

École de Technologie Supérieure
Montreal, Canada

33rd ACM/SIGAPP Symposium on Applied Computing
Requirements Engineering Track - 11th Edition

Le génie pour l'industrie

April 9 - 13, 2018
Pau, France

Outline

- Context, motivation and related work
- Methodology for requirements specification and design
- Case study
- Lessons learned, challenges and issues
- Conclusions and future work

Outline

- Context, motivation and related work
- Methodology for requirements specification and design
- Case study
- Lessons learned, challenges and issues
- Conclusions and future work

DO-178C

- DO-178C is a guideline identifying the set of *best practices* for the development of airworthy software.
- Certification of compliance is mandatory and evidence-based.
  - All compliance claims must be backed by evidence artifacts (*a.k.a.* data items).
- Supplemental document DO-331 and DO-332 modify the guideline to support model-based and object-oriented developments, respectively.

Motivation

- Avionics case studies are rarely publicly available.
- Available descriptions of avionics systems rarely talk about the software.
- Avionics systems as described in the literature do not allow for their use as benchmarks for avionics software development approaches targeting certification with DO-178C.

Related Work

Study	Case study... Representative of industrial needs	Openly available	Covers requirements specification and design according with DO-178C	Supported by methodology
Leveson *et al.*, 1994	✓		Compliance with regulation is not discussed	
Zoughbi *et al.*, 2011	✓		According with DO-178B	
Wu *et al.*, 2015	✓		Only software architecture	
White *et al.*, 2012	✓		Requirements and design have shortcomings	
Schamai *et al.*, 2015	✓		Compliance with regulation is not discussed	
Boniol *et al.*, 2014	✓	✓	Compliance with regulation is not discussed	
Ours (based on Boniol *et al.*, 2014)	✓	✓	✓	✓

Outline

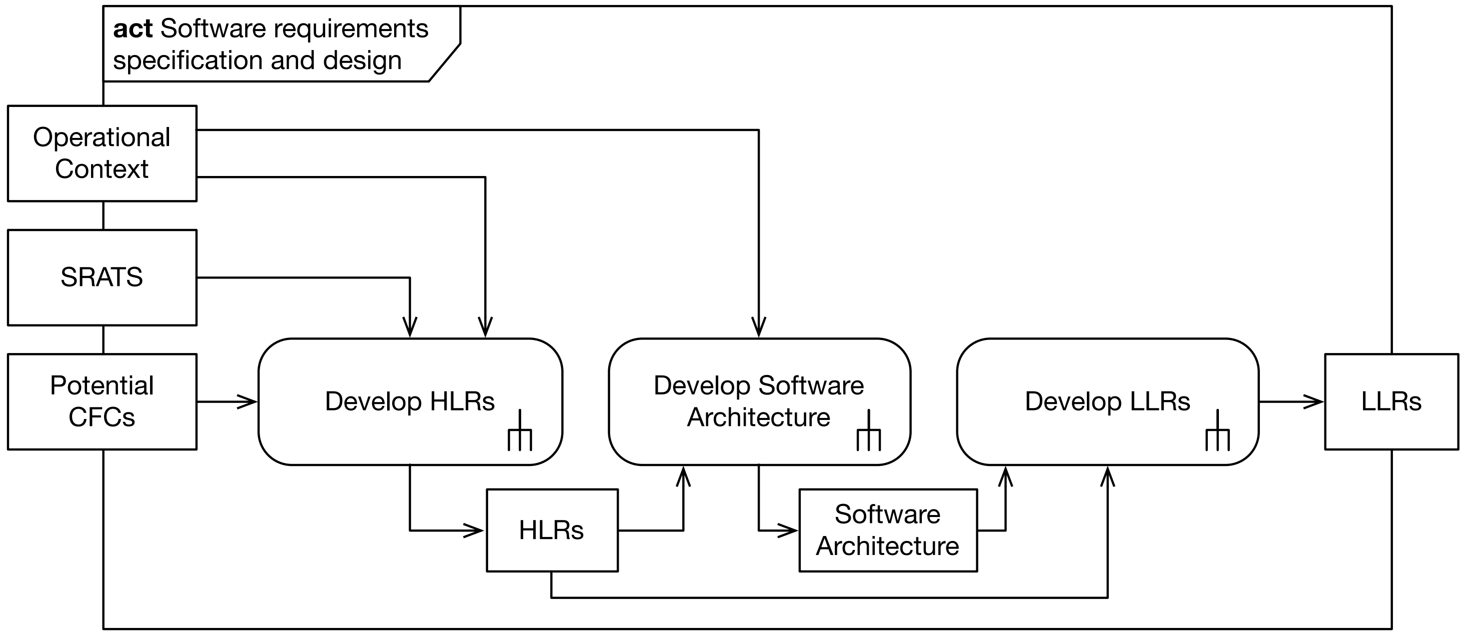
- Context, motivation and related work
- **Methodology for requirements specification and design**
- Case study
- Lessons learned, challenges and issues
- Conclusions and future work

Methodology

- No methodology for building software requirements specifications and design following DO-178C has been reported in the literature.
- Our methodology encompasses the general flow for requirements specification and design defined in DO-178C.
- As a means for quality assurance several industrial experts validated both the methodology and its outputs for the case study.

Methodology

- Exhibits the sequential nature promoted by DO-178C.
- 3 major activities: *Develop HLRs*, *Develop Software Architecture* and *Develop LLRs*.
- Actions within the activities are organized to form iterative and incremental cycles.



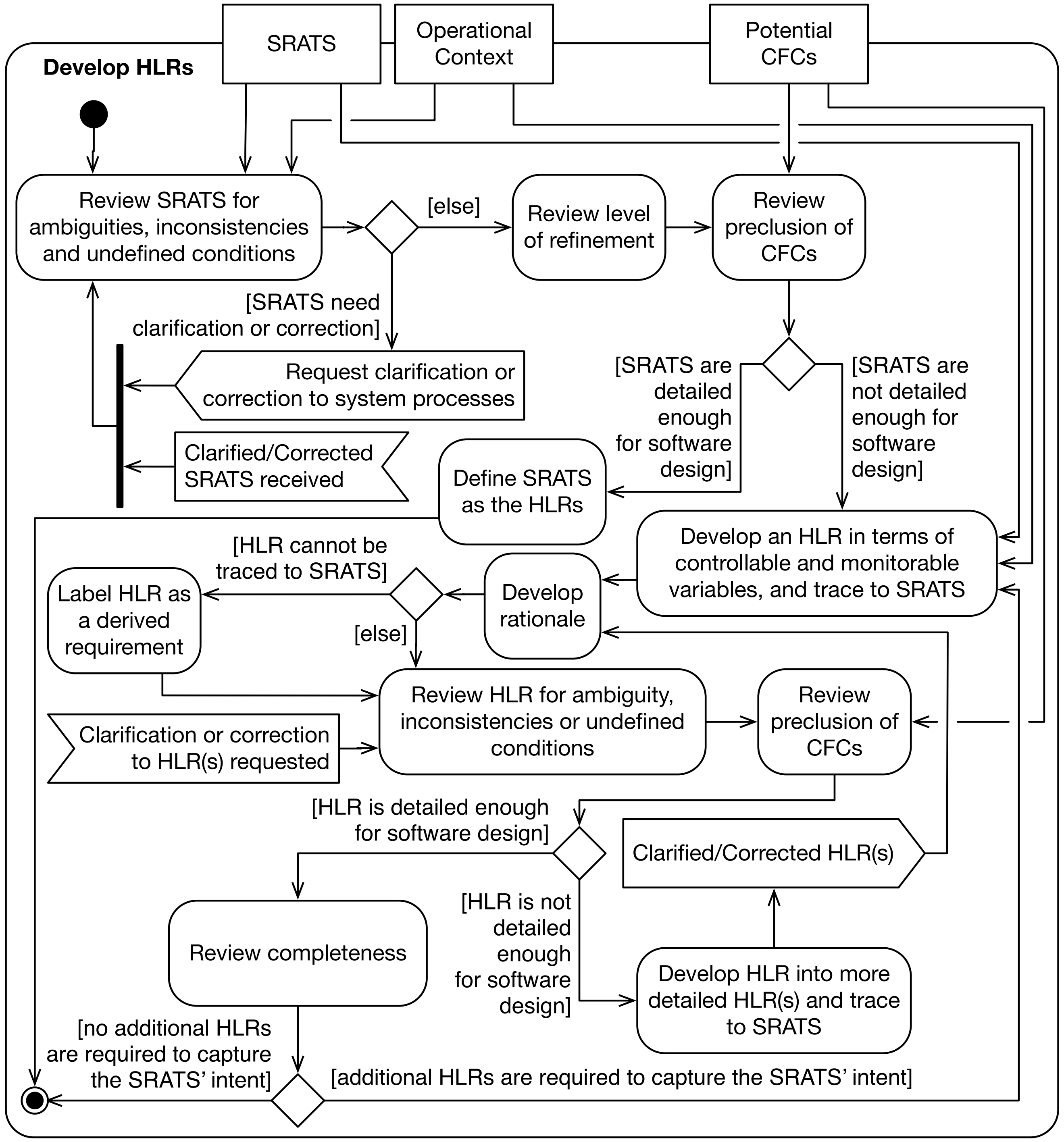
SRATS: System Requirements Allocated To Software

CFC: Contribution to Failure Condition

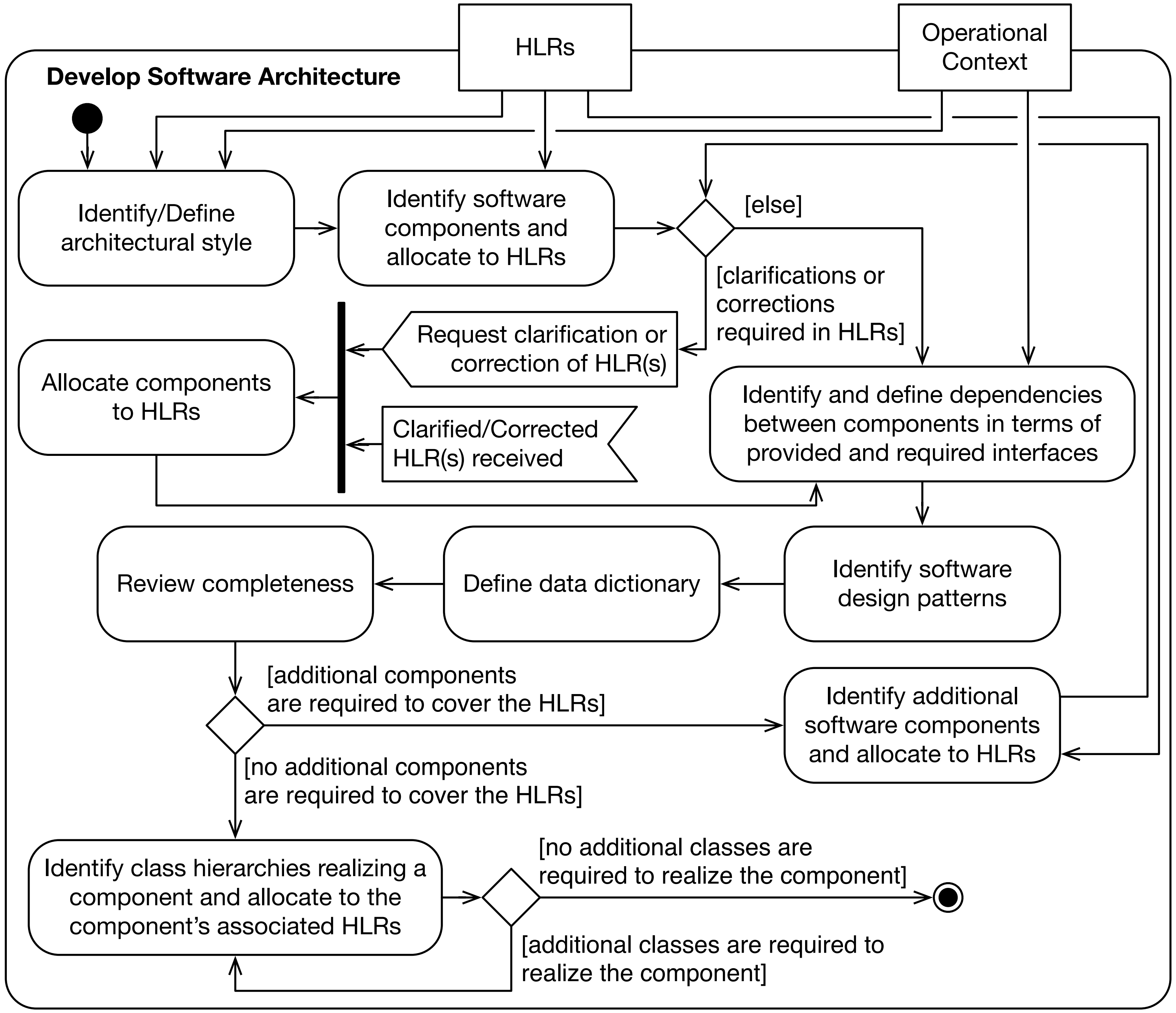
HLR: High-Level Requirement

LLR: Low-Level Requirement

Develop HLRs


Develop HLRs
SRATS
Operational Context
Potential CFCs
Review SRATS for ambiguities, inconsistencies and undefined conditions
[else]
Review level of refinement
Review preclusion of CFCs
[SRATS need clarification or correction]
Request clarification or correction to system processes
Clarified/Corrected SRATS received
[SRATS are detailed enough for software design]
[SRATS are not detailed enough for software design]
Define SRATS as the HLRs
Develop an HLR in terms of controllable and monitorable variables, and trace to SRATS
[HLR cannot be traced to SRATS]
Label HLR as a derived requirement
Develop rationale
[else]
Review HLR for ambiguity, inconsistencies or undefined conditions
Review preclusion of CFCs
Clarification or correction to HLR(s) requested
[HLR is detailed enough for software design]
Clarified/Corrected HLR(s)
[HLR is not detailed enough for software design]
Review completeness
Develop HLR into more detailed HLR(s) and trace to SRATS
[no additional HLRs are required to capture the SRATS' intent]
[additional HLRs are required to capture the SRATS' intent]

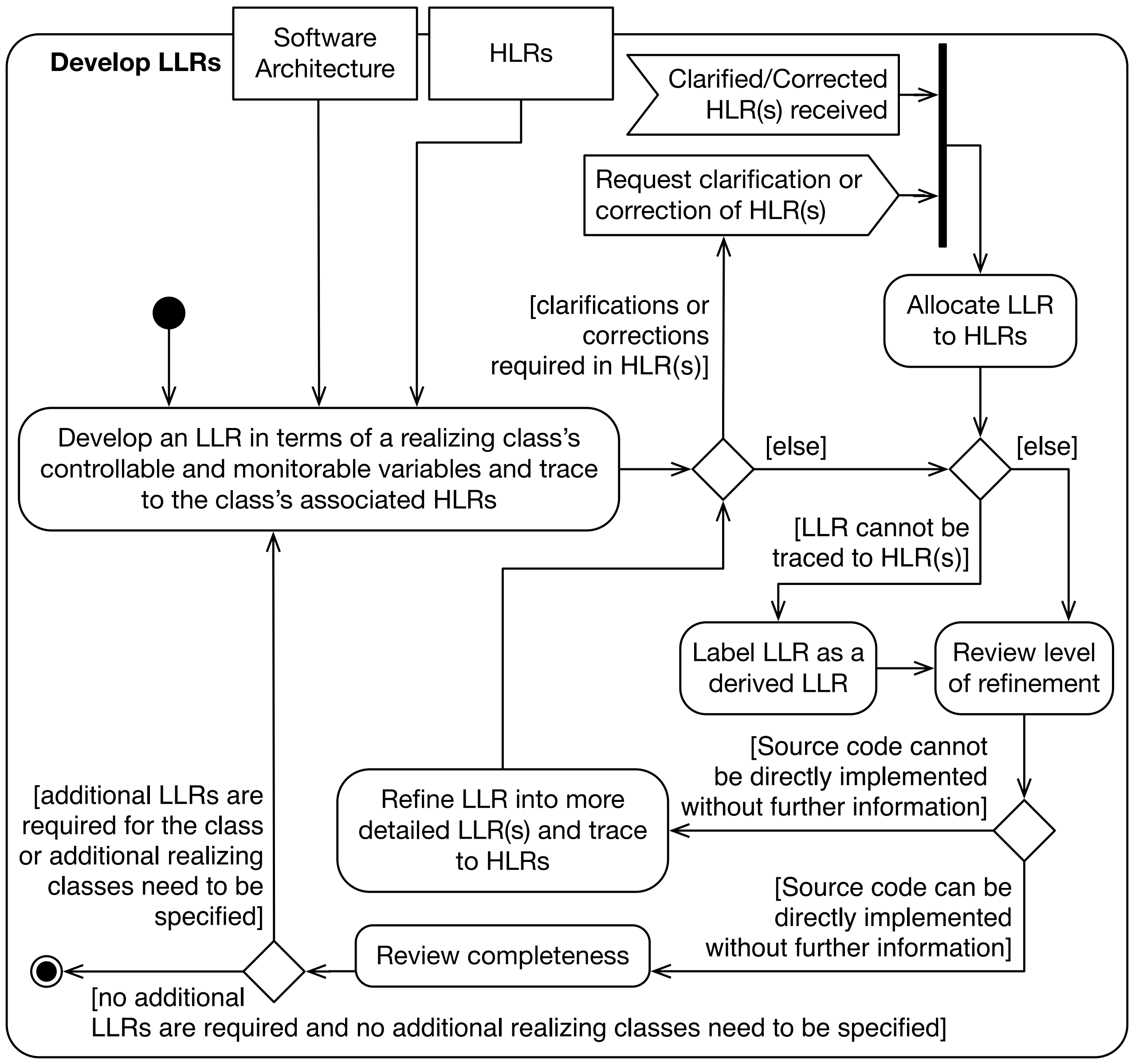
Develop Software Architecture



HLRs
Operational Context
Develop Software Architecture
Identify/Define architectural style
Identify software components and allocate to HLRs
[else]
[clarifications or corrections required in HLRs]
Request clarification or correction of HLR(s)
Allocate components to HLRs
Clarified/Corrected HLR(s) received
Identify and define dependencies between components in terms of provided and required interfaces
Review completeness
Define data dictionary
Identify software design patterns
[additional components are required to cover the HLRs]
Identify additional software components and allocate to HLRs
[no additional components are required to cover the HLRs]
Identify class hierarchies realizing a component and allocate to the component's associated HLRs
[no additional classes are required to realize the component]
[additional classes are required to realize the component]

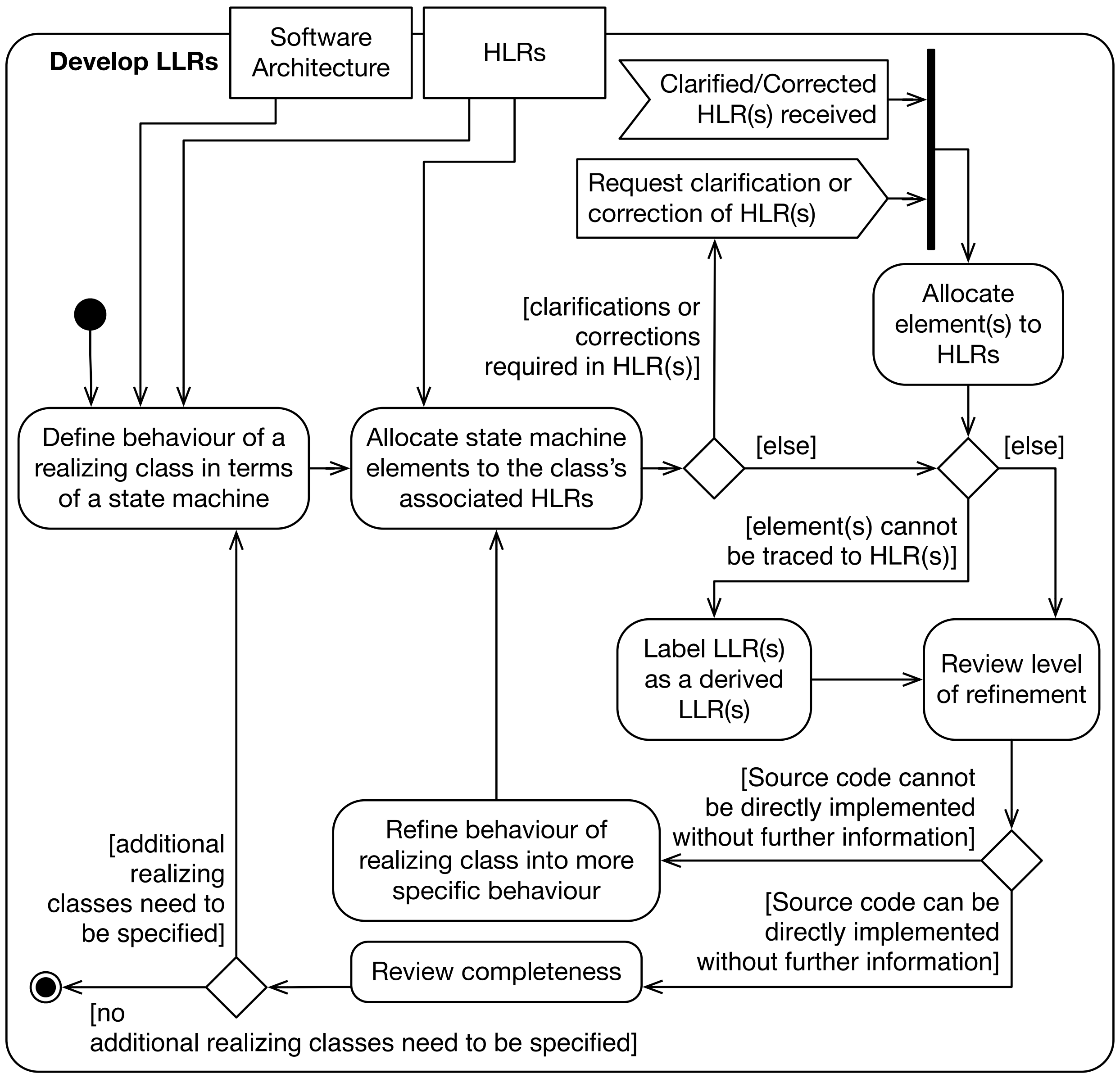
Develop LLRs

For textual LLRs



Develop LLRs
Software Architecture
HLRs
Clarified/Corrected HLR(s) received
Request clarification or correction of HLR(s)
Allocate LLR to HLRs
[clarifications or corrections required in HLR(s)]
Develop an LLR in terms of a realizing class's controllable and monitorable variables and trace to the class's associated HLRs
[else]
[else]
[LLR cannot be traced to HLR(s)]
Label LLR as a derived LLR
Review level of refinement
[Source code cannot be directly implemented without further information]
Refine LLR into more detailed LLR(s) and trace to HLRs
[Source code can be directly implemented without further information]
[additional LLRs are required for the class or additional realizing classes need to be specified]
Review completeness
[no additional LLRs are required and no additional realizing classes need to be specified]

For model-based LLRs



Develop LLRs
Software Architecture
HLRs
Clarified/Corrected HLR(s) received
Request clarification or correction of HLR(s)
Allocate element(s) to HLRs
[clarifications or corrections required in HLR(s)]
Define behaviour of a realizing class in terms of a state machine
Allocate state machine elements to the class's associated HLRs
[else]
[else]
[element(s) cannot be traced to HLR(s)]
Label LLR(s) as a derived LLR(s)
Review level of refinement
[Source code cannot be directly implemented without further information]
Refine behaviour of realizing class into more specific behaviour
[Source code can be directly implemented without further information]
[additional realizing classes need to be specified]
Review completeness
[no additional realizing classes need to be specified]

Outline

- Context, motivation and related work
- Methodology for requirements specification and design
- Case study
- Lessons learned, challenges and issues
- Conclusions and future work

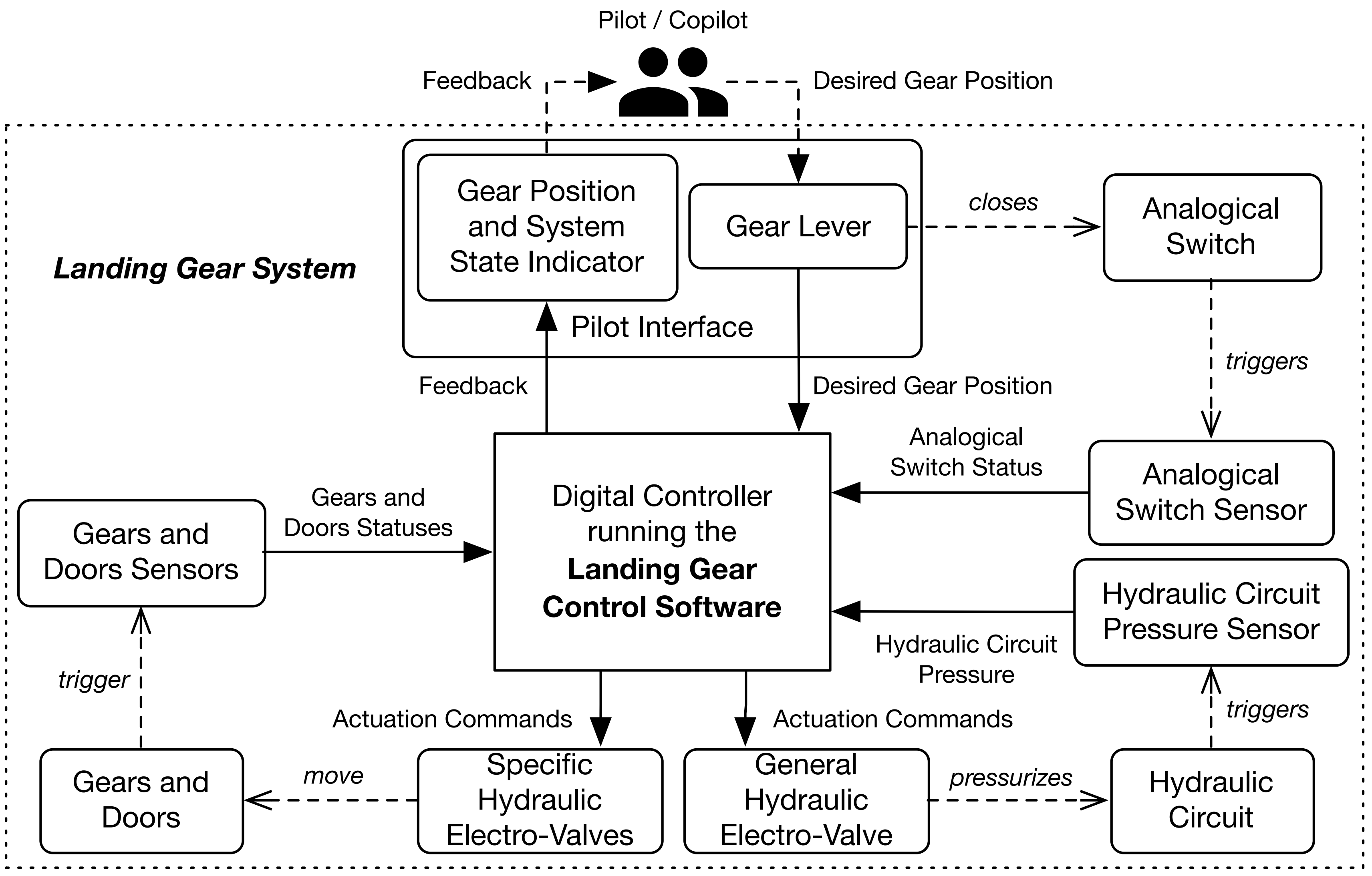
Case Study Overview

- We adapted the case study developed by Boniol et al. for the ABZ 2014 Conference.

- Our case study addresses the development of the software controlling the operation of a retractable landing gear in a tricycle configuration.
  - Landing Gear Control Software (LGCS)

- We organized the case study in several chapters corresponding to DO-178C data items:
  - Plan for Software Aspects of Certification (PSAC)
  - Software Development Standards (contains requirements, design and code standards)
  - Requirements Data (contains SRATS and HLRs)
  - Design Description (contains Software architecture and LLRs)

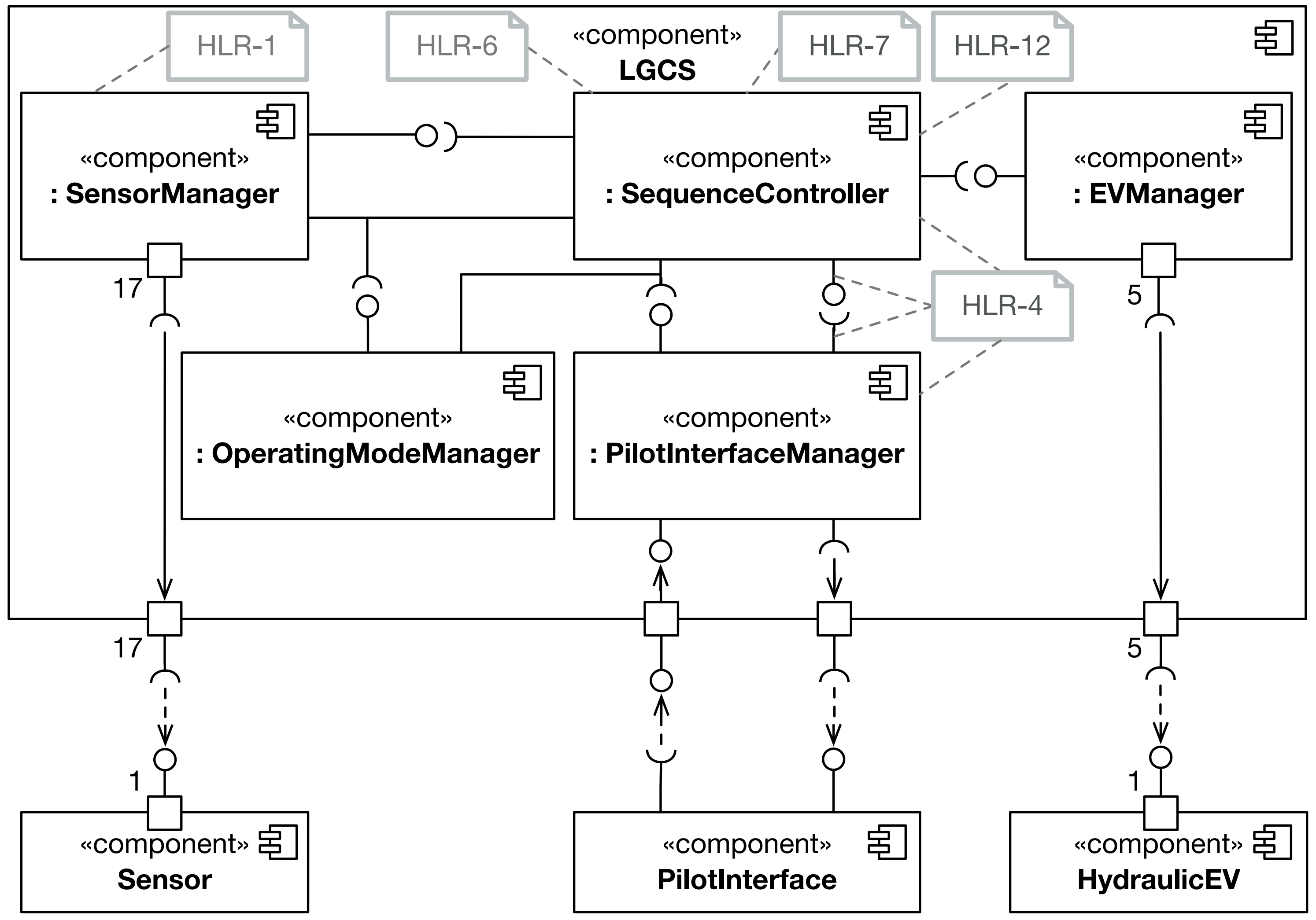
Operational Context



HLRs

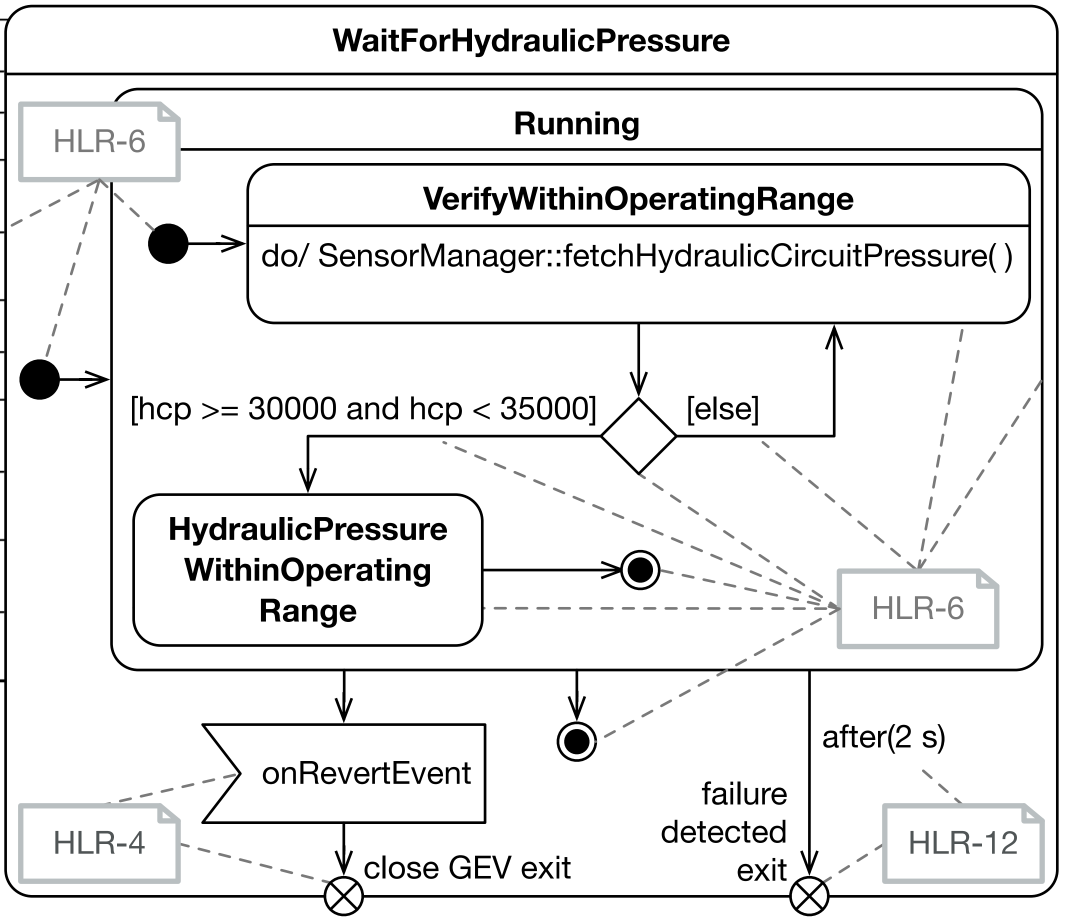
ID	Description	Rationale	Traces	Precluded CFCs
HLR-1	When the LGCS receives data from one of the LGS sensors, the LGCS shall process the three Readings associated to the sensor data based on the following rules… (the rules are not shown due to length constraints).	A sensor should not be trusted if it cannot accurately describe the status of its associated system entity. For redundancy purposes, all the sensors perform three independent readings describing simultaneously the same situation. Thus, voting strategy is followed to determine the overall value and validity of the sensor. At least two of the three readings must be equal.	SRATS-1 LLR-1 LLR-2 LLR-3 LLR-4 LLR-5 …	CFC-2
HLR-4	When the LGCS is currently executing a retraction sequence and a Down value is received for the Desired Gear Position, the LGCS shall halt the current retraction sequence and revert all the actions that were executed. Likewise, when the LGCS is currently executing an extension sequence and an Up value is received for the Desired Gear Position, the LGCS shall halt the current extension sequence and revert all the actions that were executed.	A gear motion can be canceled by the pilot or copilot at any step of an ongoing sequence. Any action executed of the ongoing sequence has to be reverted.	SRATS-4 LLR-35	
HLR-6	Once the overall value of the Hydraulic Circuit Pressure is greater than or equal to 30,000 kPa and less than 35,000 kPa after the General EV Actuation Command is set to Open, the LGCS can set to Open the necessary specific EV.	This is the specified oil pressure needed in the hydraulic circuit for operating the specific electro-valves.	SRATS-6 LLR-14 LLR-43 LLR-44 LLR-45	
HLR-7	Once at least 0.2 seconds have elapsed since the General EV Actuation Command was set to Open, the LGCS can set to Open the Door Opening EV Actuation Command.	This is the specified minimum wait time between stimulations of the general electro-valve and the specific electro-valves due to oil pressure differential produced by the change in fluid velocity across the hydraulic circuit.	SRATS-7 LLR-14 LLR-46	
HLR-12	Once 2 seconds have elapsed since the General EV Actuation Command was set to Open and the overall value of the Hydraulic Circuit Pressure is still less than 30,000 kPa, the LGCS shall detect a failure of the general hydraulic electro-valve and halt the currently executing sequence.	The hydraulic circuit should be pressurized in less than 2 seconds after the general electro-valve has been stimulated. The hydraulic circuit being unpressurized after this time is an indication of its failure. If a failure is detected the system should not be not be trusted to perform correctly and, therefore, the LGCS shall halt its operation.	SRATS-12 LLR-44 LLR-56 LLR-57	CFC-3

Software Architecture



LLRs

ID	Description	Traces	Component
LLR-14	If the PressurizationEvent and the DelayDOEVActuationTimeoutEvent are raised, the Door Opening EV Actuation Command shall be set to Open and the waitForDoorsOpen method shall be activated.	HLR-2 HLR-3 HLR-6 HLR-7	SequenceController
LLR-35	If the waitForHydraulicPressure method is active and the RevertEvent is raised, all the actions that were previously executed shall be reverted.	HLR-4	SequenceController
LLR-43	If the General EV Actuation Command is set to Open, the waitForHydraulicPressure method shall be activated.	HLR-6	SequenceController
LLR-44	If the waitForHydraulicPressure method is active and the overall value of the Hydraulic Circuit Pressure monitorable variable is less than 30,000 kPa, the waitForHydraulicPressure method shall remain active until the PressurizationTimeoutEvent is raised.	HLR- HLR-	
LLR-45	If the waitForHydraulicPressure method is active and the overall value of the Hydraulic Circuit Pressure is greater than or equal to 30,000 kPa and less than 35,000 kPa, the waitForHydraulicPressure method shall end and the PressurizationEvent shall be raised.	HLR-	
LLR-46	If the General EV Actuation Command was set to Open, the DelayDOEVActuation method shall be activated.	HLR-	
LLR-56	If the waitForHydraulicPressure method is active and 2 seconds have elapsed since the General EV Actuation Command was set to Open, the PressurizationTimeoutEvent shall be raised.	HLR-	
LLR-57	If the waitForHydraulicPressure method is active, the PressurizationTimeoutEvent is raised and the overall value of the Hydraulic Circuit Pressure monitorable variables is less than 30,000 kPa, the FailureEvent shall be raised.	HLR-	


WaitForHydraulicPressure
Running
HLR-6
VerifyWithinOperatingRange
do/ SensorManager::fetchHydraulicCircuitPressure( )
[hcp >= 30000 and hcp < 35000]
[else]
HydraulicPressure WithinOperating Range
HLR-6
onRevertEvent
after(2 s)
failure detected exit
HLR-4
close GEV exit
HLR-12

Outline

- Context, motivation and related work
- Methodology for requirements specification and design
- Case study
- **Lessons learned, challenges and issues**
- Conclusions and future work

Lessons Learned, Challenges and Issues

- Quality and granularity of SRATS.
- Requirements specification language.
- Granularity of LLRs.
- Bi-directional traceability in model-based LLRs.

Lessons Learned, Challenges and Issues

- Quality and granularity of SRATS:
  - Descriptions in Boniol *et al.* 2014 were found to be inconsistent and ambiguous.
  - SRATS had to be corrected, clarified and uniquely identified before developing HLRs.
  - SRATS can be very detailed as to be considered HLRs without further development.
  - SRATS have to be specified as clearly and as detailed as possible.

Lessons Learned, Challenges and Issues

- Requirements specification language for HLRs:
  - HLRs are routinely specified in natural language by the industry.
    - Not suitable for supporting requirements-based analyses and verification due to inherent ambiguities.
  - A form of controlled natural language following FAA guidelines was used for the specification of HLRs in the case study.
    - Model-based HLRs may help with comprehensibility and analyzability of HLRs.
    - DO-331 enables model-based HLRs. → Part of future work.
  - A heavy reliance on review actions in the Develop HLRs activity is imperative to output HLRs at an acceptable quality.

Lessons Learned, Challenges and Issues

- Requirements specification language for LLRs:
  - Developing design models for the LGCS with UML was difficult.
  - The UML specification contains lots of vaguenesses and inconsistencies and cannot be taken on its own for model development under DO-178C, DO-331 and DO-332.
    - All notational and semantical unclarities must be removed to comply with regulations.
  - UML state machine notation is not suitable for representing complex trigger conditions and trigger actions.
    - Tabular notations have been suggested in the literature but are non-standardized constructs.

Lessons Learned, Challenges and Issues

- Granularity of LLRs:
  - Developing LLRs with an appropriate granularity was challenging due to intertwined conditions the LGCS has to respect at any given moment.
    - LLRs are expected to be very detailed so source code can be implemented without needing more information.
  - Use of pseudocode or action languages (*e.g.* Alf) is discouraged because black-box verification may not be possible.
    - DO-178C requires a clear separation between LLRs and code.
    - Clear separation between LLRs and code enables black-box verification.

Lessons Learned, Challenges and Issues

- Bi-directional traces in model-based LLRs:
  - Establishing bi-directional traces in UML is an issue.
  - Backwards traceability was achieved with UML comments.
  - Forwards traceability is challenging because uniquely identifying an LLR in UML is not trivial.
  - The same issue may appear with model-based HLRs.

Outline

- Context, motivation and related work
- Methodology for requirements specification and design
- Case study
- Lessons learned, challenges and issues
- Conclusions and future work

Conclusions

- Contributions:
  - A methodology for requirements specification and design following DO-178C guideline and supplements.
  - A detailed requirements specification and design of an avionics software that:
    - Is representative of complexity and constraints found in the industry.
    - Is compliant with DO-178C, and the DO-331 and DO-332 supplements.
    - Can serve as a benchmark specification for avionics software development approaches.

Future Work

- Extension of the methodology to include software verification and validation.
- Development of model-based HLRs for the LGCS compliant with DO-331.
- Incorporate different modelling mechanisms, *e.g.*, Simulink.
- Specify other requirements, *e.g.*, those regarding deployment.

Thank you.

Summary:

- Methodology for requirements specification and design following DO-178C guideline and supplements
- Landing Gear Control Software (LGCS) requirements specification and design
- Lessons learned, challenges and issues:
  - Quality and granularity of SRATS
  - Requirements specification language (for HLRs and LLRs)
  - Granularity of LLRs
  - Bi-directional traces in model-based LLRs