TPC3 Workshop Subtopic #3

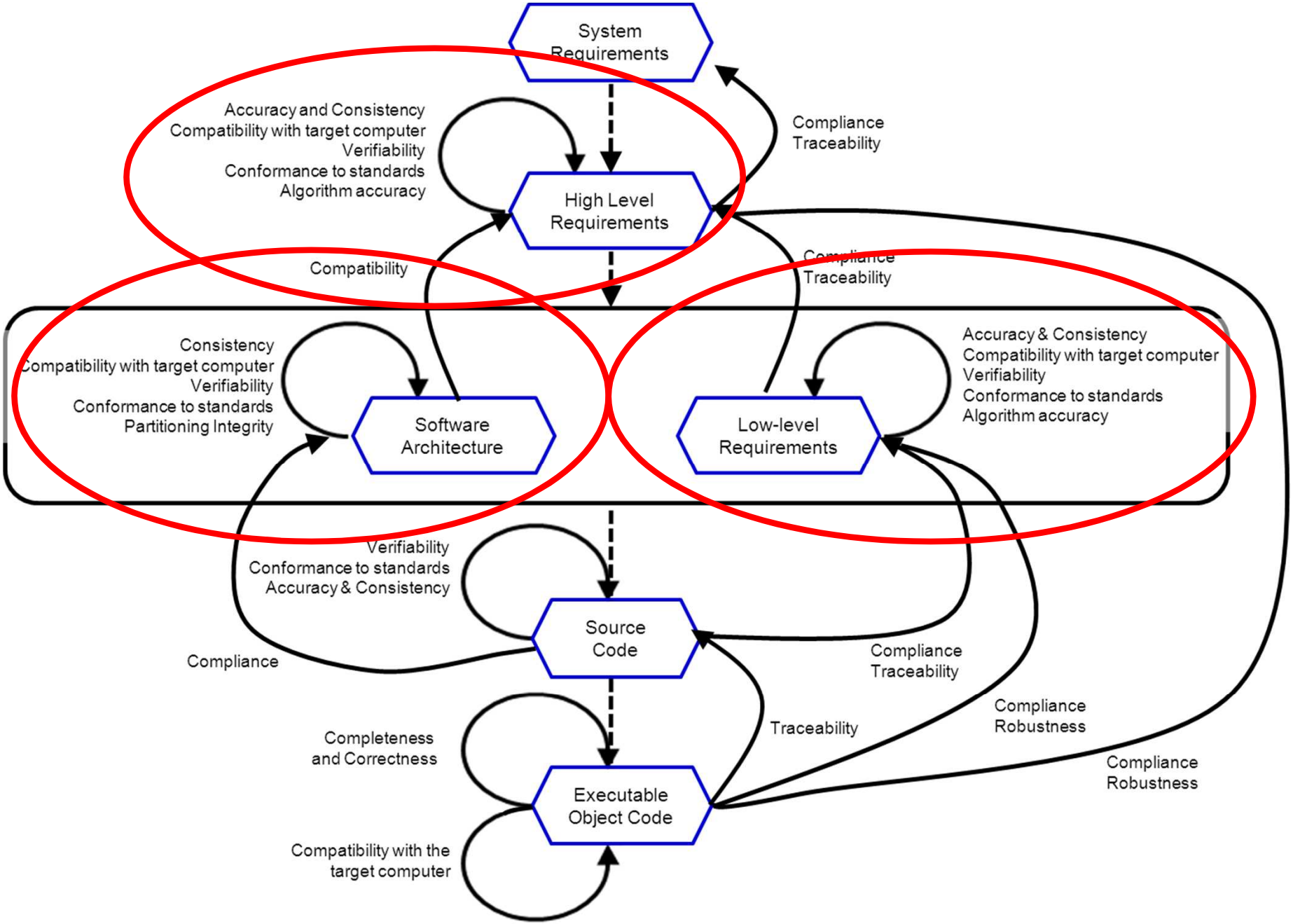
Jeff Joyce

Critical Systems Labs

December 9, 2014

Terms of Reference

1. Specific guidance for the use of TP to verify requirements with respect to completeness, consistency, traceability (up and down).
2. Specific guidance of the use of TP to verify properties of the system/software architecture, e.g., temporal and spatial partitioning. For example, what are the kinds of theorems that should be proved to verify that a particular application on a platform can't modify (or access) the data of another application.
3. Specific guidance that limits the extent to which assumptions about the environment can be introduced. (Without such restrictions, the worst case consequence is the "false implies everything" problem if the assumptions turn out to be contradictions).
4. Practical uses of TP at the architecture/detailed design level, i.e., proving that a protocol is deadlock free or freedom from specific problems such as priority inversion.
5. Specific guidance on the formulation of verification results for the verification of requirements and the verification of architectural properties.


System Requirements
Accuracy and Consistency
Compatibility with target computer
Verifiability
Conformance to standards
Algorithm accuracy
Compliance
Traceability
High Level Requirements
Compatibility
Compliance
Traceability
Consistency
Compatibility with target computer
Verifiability
Conformance to standards
Partitioning Integrity
Software Architecture
Low-level Requirements
Accuracy & Consistency
Compatibility with target computer
Verifiability
Conformance to standards
Algorithm accuracy
Verifiability
Conformance to standards
Accuracy & Consistency
Source Code
Compliance
Compliance
Traceability
Traceability
Compliance
Robustness
Completeness
and Correctness
Executable Object Code
Compliance
Robustness
Compatibility with the
target computer

1. Specific guidance for the use of TP to verify requirements with respect to completeness, consistency, traceability (up and down).
2. Specific guidance of the use of TP to verify properties of the system/software architecture, e.g., temporal and spatial partitioning. For example, what are the kinds of theorems that should be proved to verify that a particular application on a platform can't modify (or access) the data of another application.
3. Specific guidance that limits the extent to which assumptions about the environment can be introduced. (Without such restrictions, the worst case consequence is the "false implies everything" problem if the assumptions turn out to be contradictions).
4. Practical uses of TP at the architecture/detailed design level, i.e., proving that a protocol is deadlock free or freedom from specific problems such as priority inversion.
5. Specific guidance on the formulation of verification results for the verification of requirements and the verification of architectural properties.

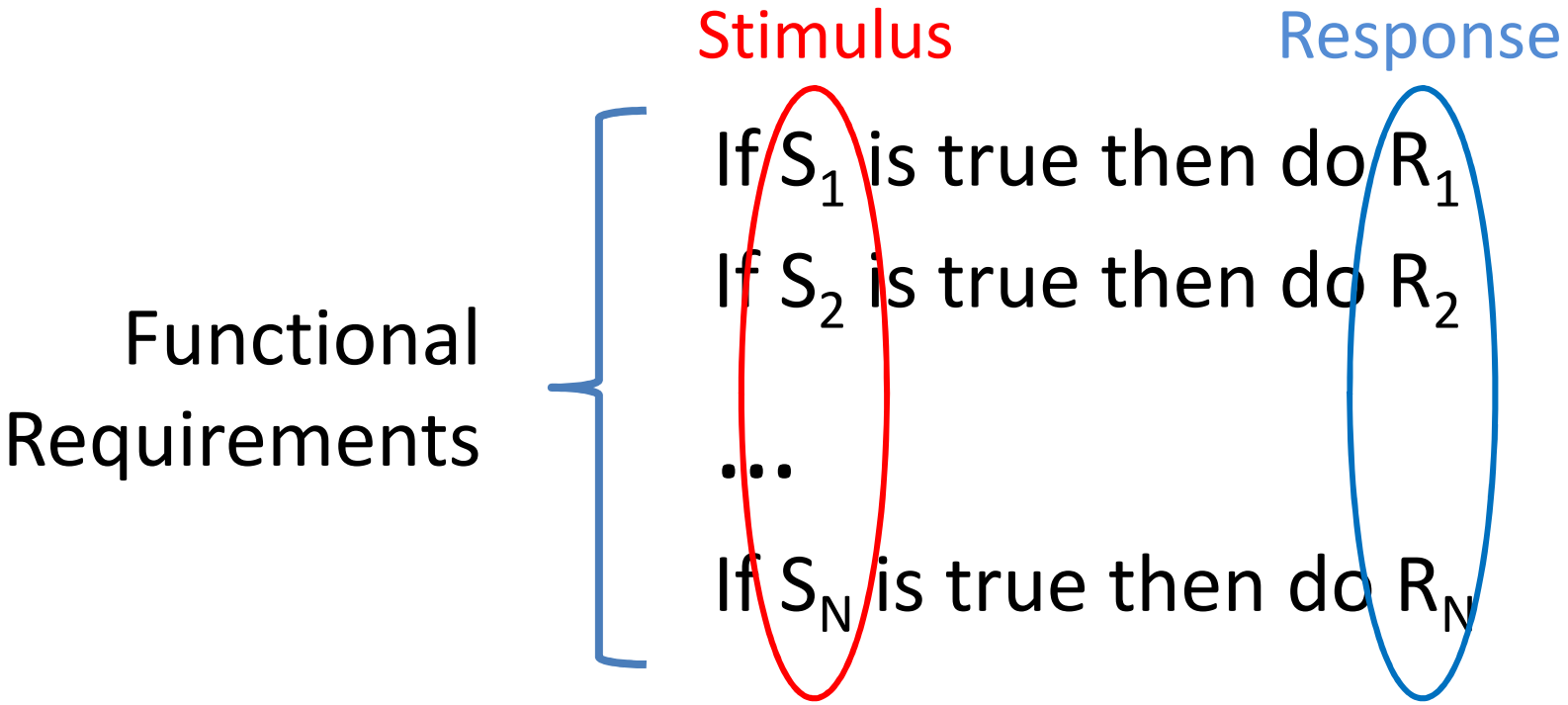
RTCA DO 178C FM Supplement

The Formal Methods supplement for RTCA DO 178C/ Eurocae ED-12C explicitly mentions the possibility of using formal analysis to demonstrate properties of both requirements specifications and the software architecture

For example:

> FM 6.3.2: "If the high-level requirements and low-level requirements are formally modeled, then formal analysis can be used to show compliance."
>
> FM 6.3.3: "If the software architecture is formally modeled, some aspects of partitioning integrity can be verified by formal analysis."

- How can TP used to verify properties of specifications and architectures?
  - Completeness
  - Consistency
  - Temporal/spatial partitioning
  - Absence of priority inversion
  - Deadlock-free?
- Is it possible to describe generic templates (or patterns) for how such properties should be verified?
- Should such templates be “prescribed” by the strawman?



1. If the system is in the armed state and fire condition is detected or the alarm button is being pressed by the operator, then turn on the building alarm.

2. If the alarm button is not being pressed by the operator, then turn off the building alarm.

1. Specific guidance for the use of TP to verify requirements with respect to completeness, consistency, traceability (up and down).
2. Specific guidance of the use of TP to verify properties of the system/software architecture, e.g., temporal and spatial partitioning. For example, what are the kinds of theorems that should be proved to verify that a particular application on a platform can't modify (or access) the data of another application.
3. Specific guidance that limits the extent to which assumptions about the environment can be introduced. (Without such restrictions, the worst case consequence is the "false implies everything" problem if the assumptions turn out to be contradictions).
4. Practical uses of TP at the architecture/detailed design level, i.e., proving that a protocol is deadlock free or freedom from specific problems such as priority inversion.
5. Specific guidance on the formulation of verification results for the verification of requirements and the verification of architectural properties.

NGV Example – Some Assumptions

1. The sensor detects the completion of a full rotation of the nose gear wheel with a maximum error of 1 cm, i.e., there is a maximum variation of +/- 1 centimeter in the location of a fixed position on the perimeter of the wheel between the times when a “click” is signaled to the computer.

2. The two global variables, NGClickTimeand NGRotations, are updated no more than 2 milliseconds after the sensor detects the completion of a full rotation of the nose gear wheel.

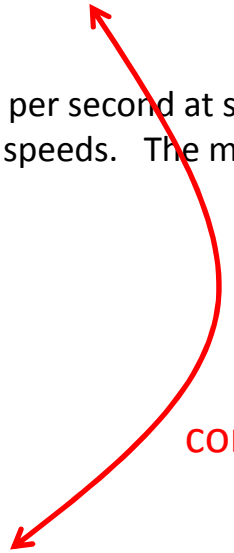
3. The maximum change in velocity (to be tolerated by this function) is 20 meters per second at speeds above 150 km/hr and no more than 10 meters per second per second at lower speeds. The maximum tolerable jerk (i.e., derivative of acceleration) is 3 metersper second cubed.

4. The wheel diameter is between 12 and 50 inches.

5. No other part of the software is capable of modifying the values of estimatedGroundVelocityIsAvailableand estimatedGroundVelocity.

6. The aircraft is moving at least 3km/hr.

7. Once invoked, this update function runs to completion, or at least the global variables NGClickTime and NGRotations will not be modified by any other part of the software between the time when this function is entered and when its execution is completed.



consistent?