



# Chapter 15 Software Hazard and Requirements Analysis

Ravindranath Jaglal

26<sup>th</sup> April 2012



# Contents

- Process Considerations
- Requirements Specification Components
- Completeness in Requirements Specifications
- Completeness Criteria for Requirements Analysis
- Constraint Analysis



# Process Considerations

- Trace identify system hazards to the software-hardware interface.
- Translate the identified software related hazards into requirements and constraints on software behavior
- To accomplish the above task a top down hazard analysis is carried out.



# Process Considerations

- Prove the consistency of the software safety constraints with the software requirements specifications.
- Demonstrate the completeness of the software requirements with respect to system safety properties.
- Most current software hazard and requirements analyses are done in an ad hoc manner.
- Currently there is no validated analysis for this step.



# Requirements Specification Components

- A basic function or objective.
  - Example: Airborne collision avoidance system
- Constraints on operating conditions.
  - Are the range of conditions within which the system may operate.
- Prioritized quality goals to help make tradeoff decisions.
  - Goals and constraints often conflict.
  - Goals may not be completely achievable.



# Completeness in Requirements Specifications

- The desired software behavior must have been specified in sufficient detail to distinguish it from any undesired program that might be designed.
- The requirement specification must simply be complete enough that it specifies safe behavior in all circumstances in which the system is to operate.



# Completeness Criteria for Requirements Analysis

- Human-Computer Interface Criteria
- State Completeness
- Input and Output Variable Completeness
- Trigger Event Completeness
- Output Specification Completeness
- Output to Trigger Event Relationships
- Specification of Transitions Between States



# Human-Computer Interface Criteria

- Specification of events to be queued
- Specification of the type and number of queues to be provided
- Ordering scheme within the queue
- Operator notification mechanism for items inserted in the queue
- Operator review and disposal commands for queue entries
- Queue entry deletion





# State Completeness

- The system and software must start in safe state.
- The internal software model of the process must be updated to reflect the actual process state at initial startup and after temporary shutdown.
- All system and local variables must be properly initialized upon startup, including clocks.
- Paths from fail safe states must be specified.
- There must be response specified for the arrival of an input in any state, including indeterminate state.



# Input and Output Variable Completeness

- All information from the sensors should be used somewhere in the specification.
- Legal output values that are never produced should be checked for potential specification incompleteness.
  - Example: Specification for only **open** and not the **close**



# Trigger Event Completeness

- Robustness Criteria
  - Every state must have a behavior defined for every possible input.
  - The logical OR of the conditions on every transition out of any state must be logically complete expression.
  - Every State must have a software behavior (transition) defined in case there is no input for a given period of time



# Trigger Event Completeness

- Non-determinism
  - The behavior of the state machine should be deterministic.
- Value and Timing Assumptions
  - The times of inputs and outputs are as important as the values.



# Trigger Event Completeness

- **Essential Value Assumptions**
  - All incoming values should be checked and a response specified in the event of an out-of-range or unexpected values.
- **Essential Timing Assumptions**
  - All inputs must be fully bounded in time and the proper behavior specified in case the limits are violated or an expected input does not arrive.



# Trigger Event Completeness

- Essential Timing Assumptions
  - A minimum and maximum load assumptions must be specified for every interrupt-signaled event whose arrival rate is not limited by another type of event.
  - A minimum arrival rate check by the software should be required for each physically distinct communication path.
  - The response to excessive inputs must be specified.



# Output Specification Completeness

- Safety critical outputs should be checked for reasonableness and for hazardous values and timing.
- For the largest interval in which both input and output loads are assumed and specified, the absorption rate of the output environment must equal or exceed the input arrival rate.



# Output Specification Completeness

- Contingency action must be specified when the output absorption rate limit will be exceeded
- Update timing requirements or other solutions to potential overload problems, such as operator event queue, need to be specified.





# Output Specification Completeness

- All inputs used in specifying output events must be properly limited in the time they can be used.
- Incomplete hazardous action sequences should have a finite time specified after which the software should be required to cancel the sequence automatically and inform the operator.



# Output Specification Completeness

- **Latency**
  - A latency factor must be included when an output is triggered by an interval of time without a specified input and the upper bound on the interval is not a simple, observable event.
  - Contingency action may need to be specified to handle events that occur within the latency period.



# Output to Trigger Event Relationship

- Basic feedback loops, as defined by the process control function, must be included in the software requirements. That is, there should be an input that the software can use to detect the effect of any output on the process.



# Specification of Transitions Between States

- All specified states must be reachable from the initial state.
- Desired recurrent behavior must be part of at least one cycle.
- States should not inhibit the production of later required outputs.
- Output commands should be reversible.
- Soft and hard failure modes should be eliminated for all hazard-reducing outputs. Hazard increasing outputs should have both soft and hard failure modes.



# Constraint Analysis

- Transitions must satisfy software system safety requirements and constraints.
- Constraint analysis on the software requirements specification includes a reachability analysis to determine whether the software, as specified, could reach the identified hazardous states.

Questions?

