SOFTWARE HAZARD AND REQUIREMENTS ANALYSIS

A REQUIREMENTS SAFETY CHECKLIST FOR SOFTWARE

D'Angelo R. Woods
October 24, 2013

WHY HAZARD ANALYSIS IS IMPORTANT

OVERVIEW

- System Theory Model
- Process Considerations
- Requirements Specifications
  - Three components
- Completeness in Requirements Specifications
  - What is completeness?
  - Contemporary Example
- Completeness Criteria for Requirements Analysis
- Final Thoughts: A Lesson on Safety

SUMMARY

- The goal of Chapter 15 is to describe the completeness and safety criteria for software requirement specifications. In other words, we want to use real-world and theoretical experience to develop a safety checklist to better manage and control potential safety hazards

PROCESS CONSIDERATIONS

Process Considerations

SYSTEMS THEORY MODEL

Computers contribute to system hazards by controlling the actions of other components either directly or indirectly

Trace identified system hazards to the software-hardware interface

- Top-down versus Bottom-up analysis

Translate the identical software-related hazards into requirements and constraints

- Specifications should be readable and reviewable

Demonstrate the completeness of the software requirements with respect to system safety properties

REQUIREMENTS SPECIFICATIONS

AUTOMATED DOOR CONTROL SYSTEM FOR A TRAIN

Automated Door Control System

THREE COMPONENTS

Basic Function or Objective

Identify Hazards/Constraints on Operating Conditions

Prioritize Quality Goals to help Make Tradeoff Decisions

Automated Door Control System

BASIC FUNCTION/OBJECTIVE

The basic functions of the Automated Door Control System is to

- Open Doors
- Stop Opening Doors
- Close Doors
- Stop Closing Doors

Automated Door Control System

IDENTIFY HAZARDS AND CONSTRAINTS

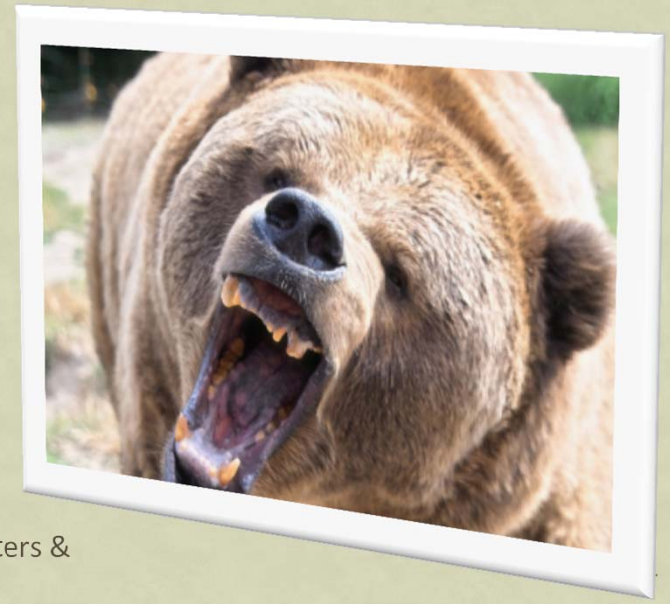
- Potential Accidents
  - Person is hit by a closing door
  - Person falls out of the train
  - Person trapped inside of train during an emergency

- Hazards
  - Doors close on a person in the doorway
  - Doors open when the train is not in a station or is not aligned with a station platform
  - Passengers/Staff are unable to exit train during an emergency

Automated Door Control System

PRIORITIZE GOALS AND IDENTIFY CONFLICTS

- Potential Conflicts
  - There is a fire—train is in motion—open command is received. What do we do?
  - Train arrives at platform—ferocious beast outside of doors—open command is received. What do we do?
  - Can you think of others?
  - Solutions may not involve ADCS at all



COMPLETENESS IN REQUIREMENTS SPECIFICATIONS

Completeness in Requirements Specifications

WHAT IS COMPLETENESS?

- Aim is to reduce or eliminate ambiguity
- Designer should be able to distinguish between safe and unsafe behavior
- Kernel requirements: “Core” requirements derived from knowledge of the system’s intent
- System-hazard or top-down analysis to detect incompleteness

Completeness in Requirements Specifications

AUTOMATED TRAIN DOOR EXAMPLE

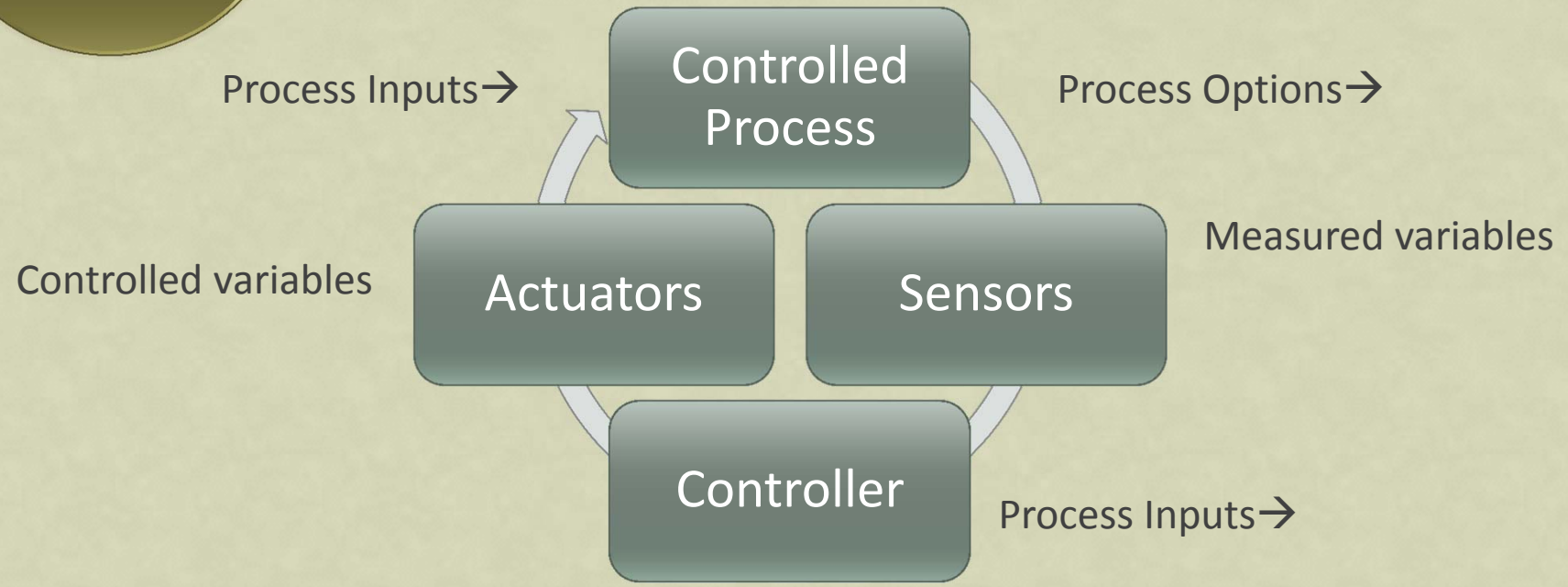
Control Action	Train Motion	Emergency	Train Position	Hazardous Control Action?		
				Provided any time?	Provided too early?	Provided too late?
Door Open	Train is moving	None	Any	Yes	Yes	Yes
Door Open	Train is moving	Exists	Any	Yes	Yes	Yes
Door Open	Train is stopped	Exists	Any	No	No	Yes
Door Open	Train is stopped	None	Not aligned	Yes	Yes	Yes
Door Open	Train is stopped	None	Aligned	No	No	No

COMPLETENESS CRITERIA FOR REQUIREMENTS ANALYSIS


Completeness Criteria for Requirements Analysis
Human Component Interface
State Completeness
Input and Output Variable Completeness
Trigger Event Completeness
Robustness Criteria
Nondeterminism/Value Timing Assumptions

Completeness Criteria for Requirements Analysis

BLACKBOX BEHAVIOR



- State Machine Model: States and Transitions
- Accidents occur when the internal model does not accurately reflect the state of the controlled process
- Must consider time lags and measurement error

Completeness Criteria for Requirements Analysis

HUMAN-COMPUTER INTERFACE CRITERIA

- Three Questions to Answer for Data Displayed to Humans
  - What events cause this item to be displayed?
  - Can and should the display of this item ever be updated once it is displayed?
    - What events should cause the upgrade?
  - What events should cause this data display to disappear?
- Failure to specify answers to these questions leads to incompleteness and thus increase potential for [safety] hazards

Completeness Criteria for Requirements Analysis

STATE COMPLETENESS

The internal software model of the process must be updated to reflect the actual process state at initial startup and after temporary shutdown

- Startup after the software has been temporarily off-line but the process has continued under manual control
- Systems where ordering of incoming data is important must include requirements to handle pre-startup inputs in order to avoid errors
- Message serialization and time stamps are useful methods

Completeness Criteria for Requirements Analysis

STAGES OF COMPLETENESS

- State Completeness
  - Specify some finite limit for how long the program waits for an input before trying various alternatives
  - Software response to the arrival of an input in any state, including unexpected inputs
- Input and Output Variable Completeness
  - All information from the sensors should be used somewhere in the specification
- Trigger Event Completeness
  - Document environmental assumptions and check at runtime

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 form a logically complete expression (tautology)
- Every state must have a software transition defined in case there is a timeout, or no input for a given period of time
- If there is a trigger condition for a state to handle inputs within a range, there will be some transition defined to handle data out of that range

Completeness Criteria for Requirements Analysis

NONDETERMINISM

Determinism means that only one possible transition is applicable at a time

$X > 0$
$X < 2$

Human operator able to rely on consistent behavior

CONCLUSION

- Safety Hazard Analysis is an important process and is required in order to identify potential hazards and develop system constraints to address these hazards prior to operation
- June 2009 incident may have been avoided with the proper safety hazard analysis
- While not all possibilities of potential hazards may be detectable (bottom-up analysis), a top-down analysis should reveal most

A LESSON ON SAFETY

WORKS CITED

- "Hazard Analysis." *Wikipedia*. Wikimedia Foundation, 13 Oct. 2013. Web. 24 Oct. 2013. <http://en.wikipedia.org/wiki/Hazard_analysis>.
- Thomas, J., S.M., and N. G. Leveson, PhD. "Performing Hazard Analysis on Complex, Software-and Human-Intensive Systems." *Journal of System Safety* (2011): 1-9. Web. 22 Oct. 2013. <http://www.system-safety.org/conferences/2011/papers/Performing%20Hazard%20Analysis%20on%20Complex,%20Software-%20and%20Human-Intensive%20Systems.pdf>.
- "June 2009 Washington Metro Train Collision." *Wikipedia*. Wikimedia Foundation, 18 Oct. 2013. Web. 22 Oct. 2013. <http://en.wikipedia.org/wiki June_2009_Washington_Metro_train_collision>.