Subsystem Hazard Analysis (SSHA)

- Examine subsystems to determine how their
  - Normal performance
  - Operational degradation
  - Functional failure
  - Unintended function
  - Inadvertent function (proper function but at wrong time or in wrong order)

  could contribute to system hazards.
- Determine how to satisfy design constraints in subsystem design.
- Validate the subsystem design satisfies safety design constraints and does not introduce previously unidentified hazardous system behavior.

Software Hazard Analysis

- A form of subsystem hazard analysis.
- Validate that specified software blackbox behavior (requirements) satisfies the system safety design constraints.
- Check specified software behavior satisfies general software system safety design criteria.
- Trace requirements into code.
- Must perform on ALL software, including COTS.

Requirements Topics

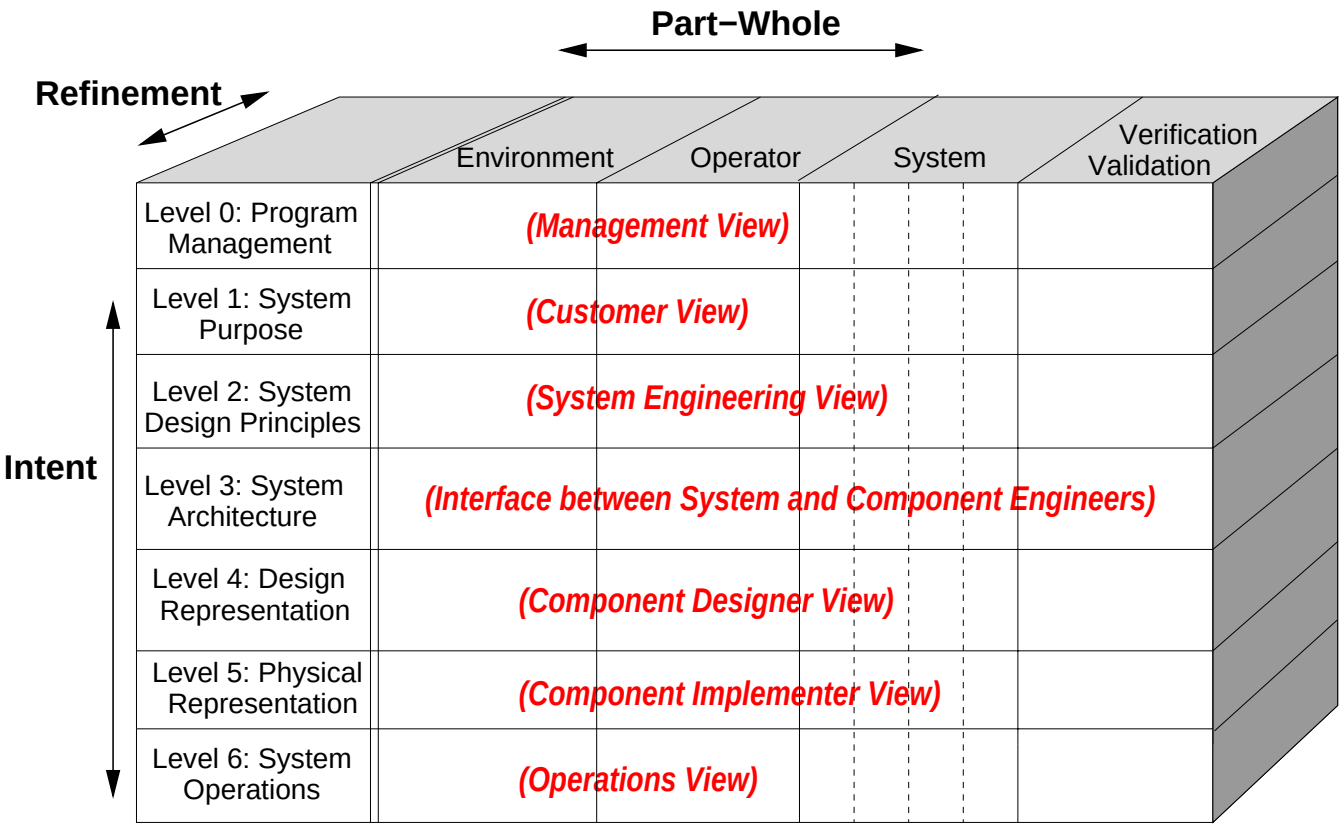
- Intent Specifications
- State Machine Models
- Completeness Criteria for Requirements
- Analyzing Requirements

Intent Specifications

- Based on
  - Research in how experts solve problems
  - Basic principles of system engineering
- Goals
  - Enhance communication and review
  - Capture domain knowledge and design rationale
  - Integrate safety information into decision−making environment
  - Provide traceability from requirements to design to code
    - For verification and validation
    - To support change and upgrade process

Intent Specifications (2)

- Differs in structure, not content from other specifications
  - Hierarchy of models
  - Describe system from different viewpoints
  - Traceability between models (levels)
- 7 levels:
  - Represent different views of system (not refinement)
    - From different perspectives
    - To support different types of reasoning



	Environment	Operator	System and components	V&V
Level 0 Prog. Mgmt.	Project management plans, status information, safety plan, etc.			
Level 1 System Purpose	Assumptions Constraints	Responsibilities Requirements I/F requirements	System goals, high−level requirements, design constraints, limitations	Preliminary Hazard Analysis, Reviews
Level 2 System Principles	External interfaces	Task analyses Task allocation Controls, displays	Logic principles, control laws, functional decomposition and allocation	Validation plan and results, System Hazard Analysis
Level 3 System Architecture	Environment models	Operator Task models HCI models	Blackbox functional models Interface specifications	Analysis plans and results, Subsystem Hazard Analysis
Level 4 Design Rep.		HCI design	Software and hardware design specs	Test plans and results
Level 5 Physical Rep.		GUI design, physical controls design	Software code, hardware assembly instructions	Test plans and results
Level 6 Operations	Audit procedures	Operator manuals Maintenance Training materials	Error reports, change requests, etc.	Performance monitoring and audits

Requirements Validation

- Requirements are source of most operational errors and almost all the software contributions to accidents.
- Much of software safety effort therefore should focus on requirements.
- Problem is dealing with complexity

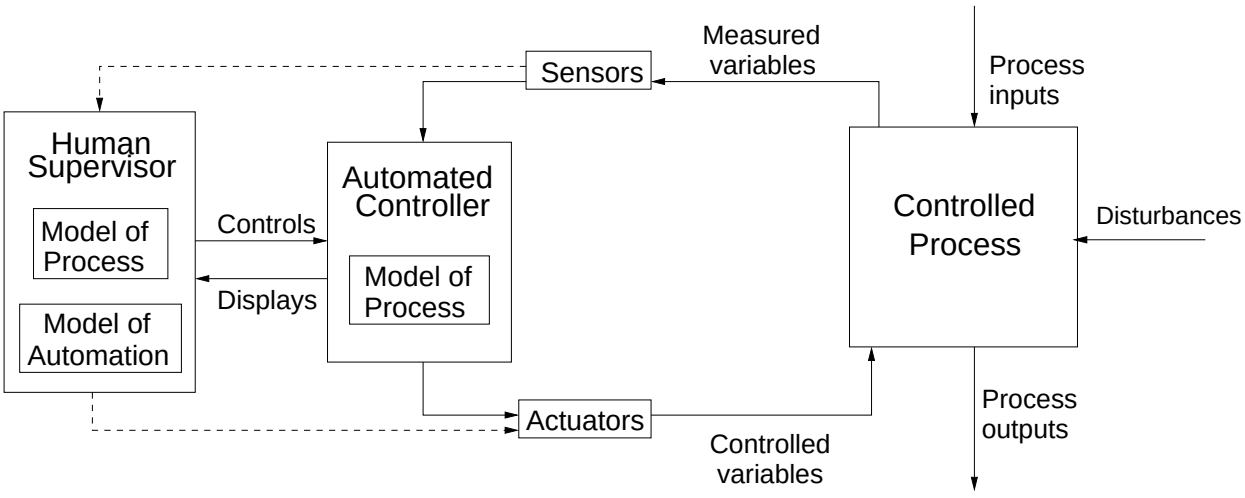
    1) Use blackbox models to separate external behavior from complexity of internal design to accomplish the behavior.

    2) Use abstraction and metamodels to handle large number of discrete states required to describe software behavior.

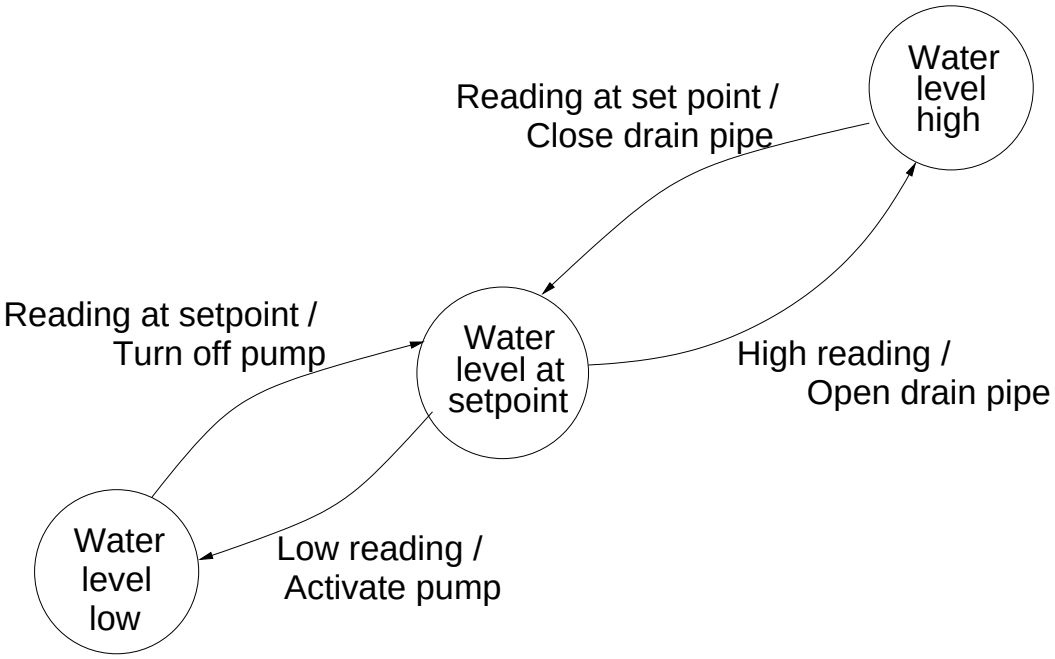
        - Do not have continuous math to assist us
        - But new types of state machine modeling languages drastically reduce number of states and transitions modeler needs to describe.

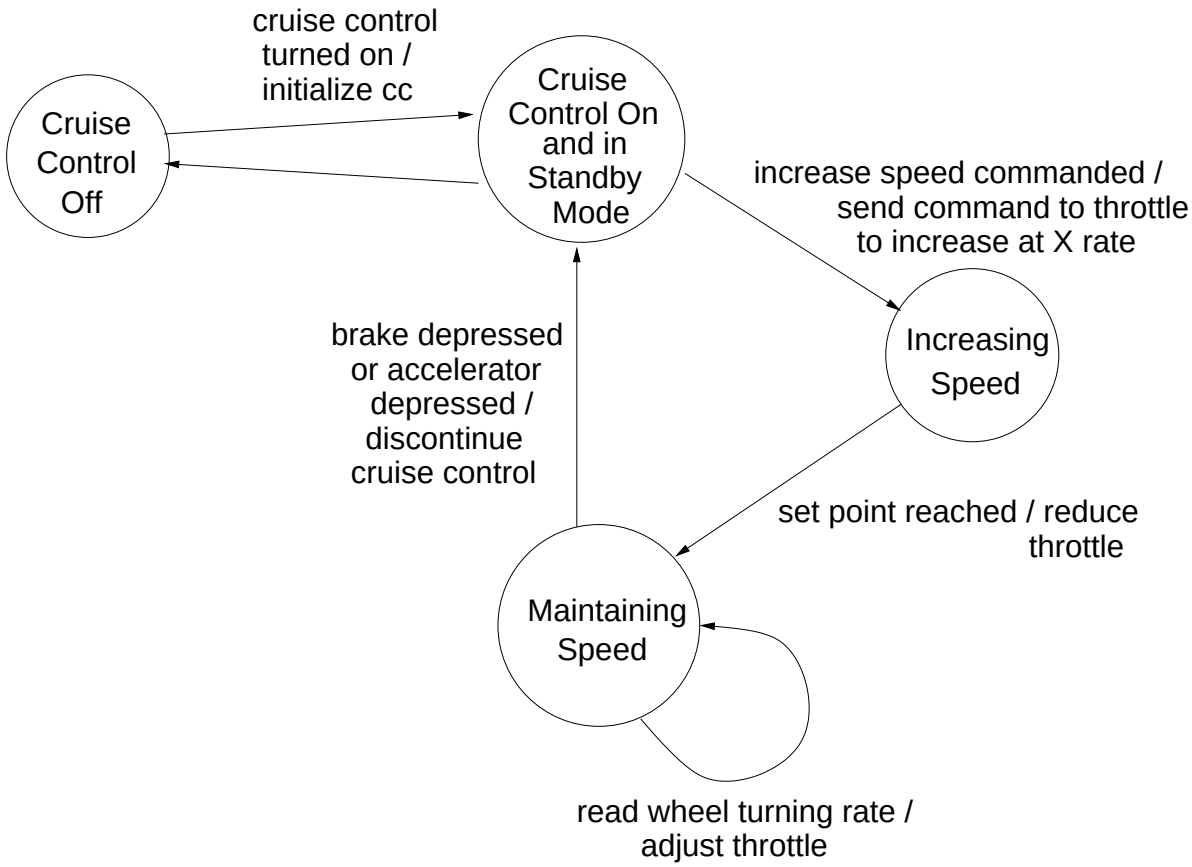
Requirements as State Machine Models

Define required blackbox behavior of software in terms of a state machine model of the process (plant).



Example of a State Machine Model





Requirements Completeness

- Most software−related accidents involve software requirements deficiencies.
- Accidents often result from unhandled and unspecified cases.
- We have defined a set of criteria to determine whether a requirements specification is complete.
- Derived from accidents and basic engineering principles.
- Validated (at JPL) and used on industrial projects.

Completeness:	Requirements are sufficient to distinguish the desired behavior of the software from that of any other undesired program that might be designed.

Blackbox specifications

Start from a "blackbox" statement of software behavior:

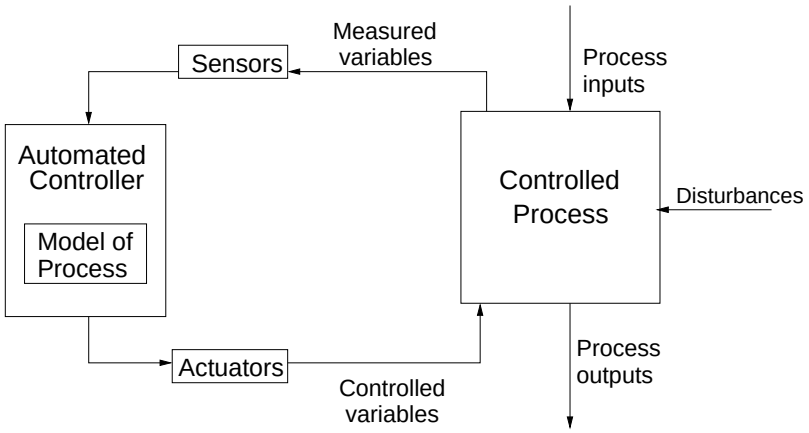
- Includes externally visible behavior only
  - Internal design decisions are not included.
  - Simplifies the specification and review by system experts
- In essence, the specification is the input to output function computed by the component, i.e., the transfer function.
- Specify:
  - Outputs
  - Triggers for outputs (externally observable conditions or events)

    (not just inputs, e.g., may want to trigger an output on the passage of time)

Requirements Completeness Criteria

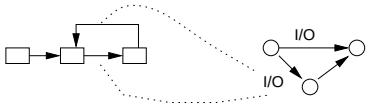
Completeness criteria define what needs to be specified about triggers, outputs, and the relationship between them.

- Defined in terms of a state machine model of the process (plant) model.
- Required blackbox behavior specified using the process model.



Requirements Completeness Criteria (2)

- How were criteria derived?
  - Mapped the parts of a control loop to a state machine



  - Defined completeness for each part of state machine
    - States, inputs, outputs, transitions
    - Mathematical completeness
  - Added basic engineering principles (e.g., feedback)
  -

Requirements Completeness Criteria (3)

About 60 criteria in all including human−computer interaction.

(won't go through them all — they are in the book)

Startup, shutdown	Robustness
Mode transitions	Data age
Inputs and outputs	Latency
Value and timing	Feedback
Load and capacity	Reversibility
Environment capacity	Preemption
Failure states and transitions	Path Robustness
Human−computer interface	

Startup and State Completeness

Many accidents involve off−nominal processing modes, including startup and shutdown and handling unexpected inputs.

Examples of completeness criteria in this category:

- The internal software model of the process must be updated to reflect the actual process state at initial startup and after temporary shutdown.
- The maximum time the computer waits before the first input must be specified.
- There must be a response specified for the arrival of an input in any state, including indeterminate states.

Input and Output Variable Completeness

At blackbox interface, only time and value observable to software.

So triggers and outputs must be defined only as constants or as the value of observable events or conditions.

Criteria:

- 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.

Trigger Event Completeness

- Behavior of computer defined with respect to assumptions about the behavior of the other parts of the system.
- A robust system will detect and respond appropriately to violations of these assumptions (such as unexpected inputs).
- Therefore, robustness of software built from specification will depend on completeness of specification of environmental assumptions.
  - There should be no observable events that leave the program's behavior indeterminate.
- Why need to document and check all assumptions?

Formal Robustness Criteria

- To be robust, the events that trigger state changes must satisfy the following:
  1. Every state must have a behavior (transition) defined for possible input.
  2. The logical OR of the conditions on every transition out of every state must form a tautology.

     $x < 5$

     $x \geqslant 5$
  3. Every state must have a software behavior (transition) defined in case there is no input for a given period of time (a timeout).
- Together these criteria guarantee handing input that are within range, out of range, and missing.

Nondeterminism Criterion

- The behavior of the requirements should be deterministic (only one possible transition out of a state is applicable at any time).

  $X > 0$

  $X < 2$

- We (and others) have tools to check specifications based on state machines for robustness, consistency, and nondeterminism.

NOTE: This type of mathematical completeness is NOT enough.

e.g., *"true"* is a mathematically complete, consistent, and deterministic specification.

Failure States and Transition Criteria

Need to completely specify:

- Off−nominal states and transitions
- Performance degradation
- Communication with operator about fail−safe behavior
- Partial shutdown and restart
- Hysteresis in transitions between off−nominal and nominal

Most accidents occur while in off−nominal processing modes.

Value and Timing Assumptions

Examples:

- All inputs should be checked and a response specified in the event of an out−of−range or unexpected value.
- All inputs must be fully bounded in time and the proper behavior specified in case the limits are violated.
- Minimum and maximum load assumptions ...
- A minimum−arrival−rate check should be required for each physically distinct communication path.

  Software should have the capability to query its environment with respect to inactivity over a given communication path.
- Response to excessive inputs (violations of load assumptions) must be specified.

Environment Capacity Constraints

Examples:

- 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.

- Contingency action must be specified when the output absorption rate limit will be exceeded.

Human−Computer Interface Criteria

- For every data item displayable to a human, must specify:
  1. What events cause this item to be displayed?
  2. What events cause item to be updated?
     If so, what events should cause the update?
  3. What events should cause the display to disappear?
- For queues need to specify:
  1. Events to be queued
  2. Type and number of queues to be provided (alert and routine)
  3. Ordering scheme within queue (priority vs. time of arrival)
  4. Operator notification mechanism for items inserted in the queue.
  5. Operator review and disposal commands for queue entries.
  6. Queue entry deletion.

Data Age Criteria

- All inputs used in specifying output events must be properly limited in the time they can be used.
- Output commands that may not be able to be executed immediately must be limited in the time they are valid.
- Incomplete hazardous action sequences (transactions) should have a finite time specified after which the software should be required to cancel the sequence automatically and inform the operator.
- Revocation of partially completed transactions may require:
    1. Specification of multiple times and conditions under which varying automatic cancellation or postponement actions are taken without operator confirmation.
    2. Specification of operator warnings to be issued in case of such revocation.

Latency Criteria

- Latency is the time interval during which receipt of new information cannot change an output even though it arrives prior to output.
    - Influenced by hardware and software design (e.g., interrupt vs. polling)
    - Cannot be eliminated completely.
    - Acceptable length determined by controlled process.
- Subtle problems when considering latency of HCI data.
  (see book for criteria)

Feedback Criteria

Basic feedback loops, as defined by the process control function, must be included in the requirements along with appropriate checks to detect internal or external failures or errors.

Examples:

- There should be an input that the software can use to detect the effect of any output on the process.
- Every output to which a detectable input is expected must have associated with it:
    1. A requirement to handle the normal response
    2. Requirements to handle a response that is missing, too late, too early, or has an unexpected value.

Path Criteria

- Paths between states are uniquely defined by the sequence of trigger events along the path.
- Transitions between modes are especially hazardous and susceptible to incomplete specification.

REACHABILITY

- Required states must be reachable from initial state.
- Hazardous states must not be reachable.
- Complete reachability analysis often impractical, but may be able to reduce search by focusing on a few properties or using backward search.
- Sometimes what is not practical in general case is practical in specific cases.

Path Criteria (2)

RECURRENT BEHAVIOR

- Most process control software is cyclic. May have some non−cyclic states (mode change, shutdown)
- Required sequences of events must be specified in and limited by transitions in a cycle.
- *Inhibiting state:* State from which output cannot be generated.
- There should be no states that inhibit later required outputs.

REVERSIBILITY

PREEMPTION

Path Criteria (3)

PATH ROBUSTNESS

Soft failure mode: The loss of ability to receive input X could inhibit the production of output Y

Hard failure mode: The loss of ability to receive input X will inhibit the production of output Y

- Soft and hard failure modes should be eliminated for all hazard reducing outputs.
- Hazard increasing outputs should have both soft and hard failure modes.
- Multiple paths should be provided for state changes that maintain safety.
- Multiple inputs or triggers should be required for paths from safe to hazardous states.

SpecTRM

Specification Tools and Requirements Methodology

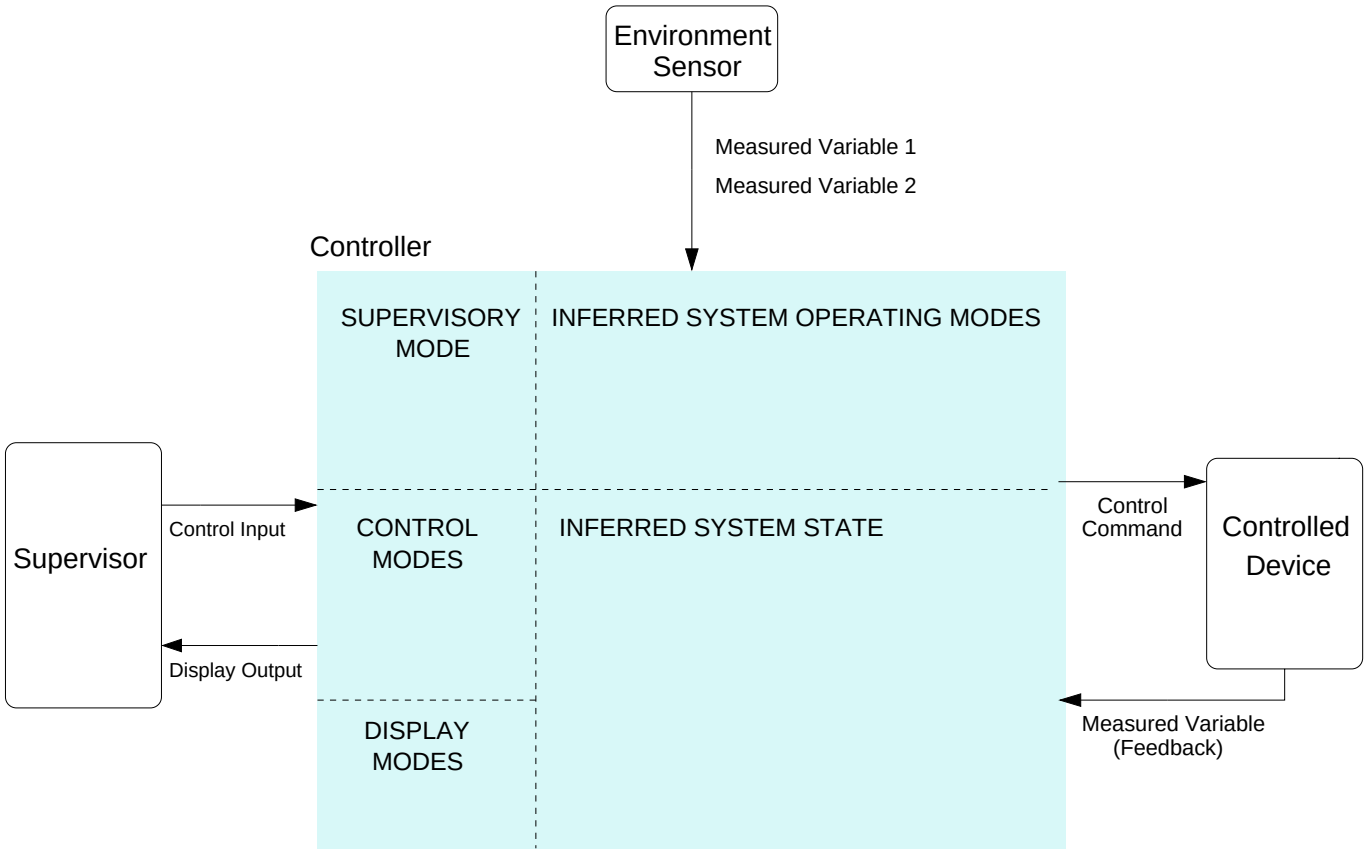
A set of integrated tools to assist in building complex control systems.

Level 3 Specification (modeling) language goals

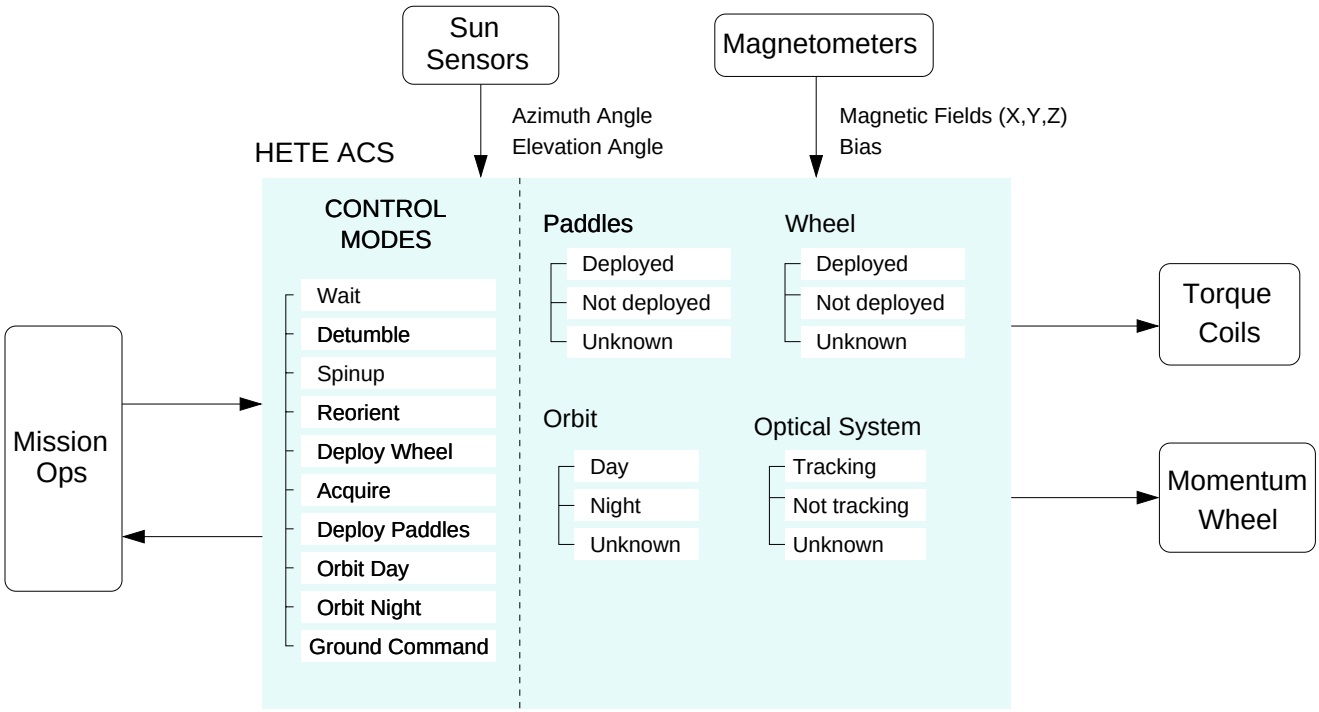
- Readable and reviewable
  - Minimize semantic distance
  - Minimal (blackbox)
  - Easy to learn
  - Unambiguous and simple semantics
- Complete
  - Can specify everything need to specify
- Analyzable
  - Executable
  - Formal (mathematical) foundation
  - Includes human actions
  - Assists in finding incompleteness

SpecTRM−RL

- Combined requirements specification and modeling language
- A state machine with a domain−specific notation on top of it.
  - Includes a task modeling language
  - Can add other notations and visualizations of state machine
- Enforces or includes most of completeness criteria
- Supports specifying systems in terms of modes
  - Control modes
  - Operational modes
  - Supervisory modes
  - Display modes



SpecTRM Model of HETE Attitude Control System



Control Mode

ACS Mode (2)

= Detumble (Mode 1)

The purpose of detumble mode is to minimize the magnitude of body momuntum vector in the X−Z plane. As soon as the magnitude falls below a threshold,e software should transition to spinup mode. The mode delay provides hysteresis in the mode transitions to prevent the software from jumping between modes too rapidly.

In detumble mode, the wheel actuator shall be controlled such that the wheel maintains the velocity it had upon entering the mode, and the magnetic moment along the Y axis shall be controlled to minimize the angular velocity about the X and Z axes.

				OR			
Control Mode	Wait	T					
	Detumble		T	T			
	Spinup				T	T	
	Ground Control						T
State Values	Time since entered wait >= 10 sec	T					
	Time since entered detumble < 100 sec		T	F			
	xz momentum error > xz momentum error threshold			T	T	T	
	Time since entered spinup >= 100 sec				T	T	
	Paddles in−state deployed				F		
	Optical system in−state tracking					F	
	Time since entered ground control >= 10 sec						T

Requirements Analysis

- Model Execution, Animation, and Visualization
- Completeness
- State Machine Hazard Analysis (backwards reachability)
- Human Task Analysis
- Test Coverage Analysis and Test Case Generation
- Automatic code generation

Completeness Analysis

- Automated consistency checker
- Automated completeness (robustness) checker
- Components of modeling language to assist in checking additional criteria
- API to allow checking by other formal analysis tools

Output Command

Name

Destination:
Acceptable Values:
- **Units:**
- **Granularity:**
- **Exception Handling:**
- **Hazardous Values:**

Timing Behavior:
- **Initiation Delay:**
- **Completion Deadline:**
- **Output Capacity Assumptions:**
  - **Load:**
  - **Min time between outputs:**
  - **Max time between outputs:**
- **Hazardous timing behavior:**
- **Exception−Handling:**

Feedback Information:
- **Variables:**
- **Values:**
- **Relationship:**
- **Min. time (latency):**
- **Max. time:**
- **Exception Handling:**

Reversed By:
Comments:
References:

DEFINITION

= ...

Input Value

DA1 Alt Signal

Source: Digital Altimeter 1
Type: integer
Possible Values (Expected Range): −20..2500
Exception−Handling: Values below −20 are treated as −20 and values above 2500 as 2500
Units: feet AGL
Granularity: 1 foot
Arrival Rate (Load): one per second average
Min−Time−Between−Inputs: 100 milliseconds
Max−Time−Between−Inputs: none
Obsolescence: 2 seconds
Exception−Handling: Assumes value Obsolete
Description:
Comments:
Appears in: Altitude

DEFINITION

= New Data for DA1 Alt Signal

DA1 Alt Signal was Received	T

= Previous Value of DA1 Alt Signal

DA1 Alt Signal was Received	F
Time Since DA1 Slt Signal was last received > 2 seconds	F

= Obsolete

DA1 Alt Signal was Received	F	*	*
Time Since DA1 Alt Signal was last received > 2 seconds	T	*	*
System Start	*	T	*
DA1 Alt Signal was Never Received	*	*	T

Model Execution and Animation

- SpecTRM−RL models are executable.
- Model execution is animated
- Results of execution could be input into a graphical visualization
- Inputs can come from another model or simulator and output can go into another model or simulator.

Executable Specifications as Prototypes

- Easily changed
- At end, have specification to use
- Can be reused (product families)
- Can be more easily reviewed
- If formal, can be analyzed
- Can be used in hardware−in−the−loop or operator−in−the−loop simulations

Operator Task Models

- To ensure safe and efficient operations, must look at the interaction between the human controllers and the computer.
- Use same underlying formal modeling language.
- Designed a visual representation more appropriate for the task modeling.
- Can be executed and analyzed along with other parts of the model.

Interactive Visualizations

- Pictures or diagrams allowing direct manipulation to learn more about the thing represented
- Useful to understand specification or results of analysis or simulation
- Part of a potential "CATIA" for the logical parts of systems
- Goal is to enhance intellectual manageability of complex system design, review, maintenance, and operation.

  To extend human cognitive limits
- Experimental results show extremely useful in reading and reviewing specifications of complex systems.
- We are trying to provide a theoretical foundation for designing interactive visualizations.