FAA Requirements Engineering Management Handbook

9. Define the Software Requirements

Kansas State University

Steps in the REMH

1. Develop the System Overview
2. Identify the System Boundary
3. Develop the Operational Concepts
4. Identify the Environmental Assumptions
5. Develop the Functional Architecture
6. Revise the Architecture to Meet Implementation Constraints
7. Identify System Modes
8. Develop the Detailed Behavior and Performance Requirements
9. **Define the Software Requirements**
10. Allocate System Requirements to Subsystems
11. Provide Rationale

Software Requirements: Goals

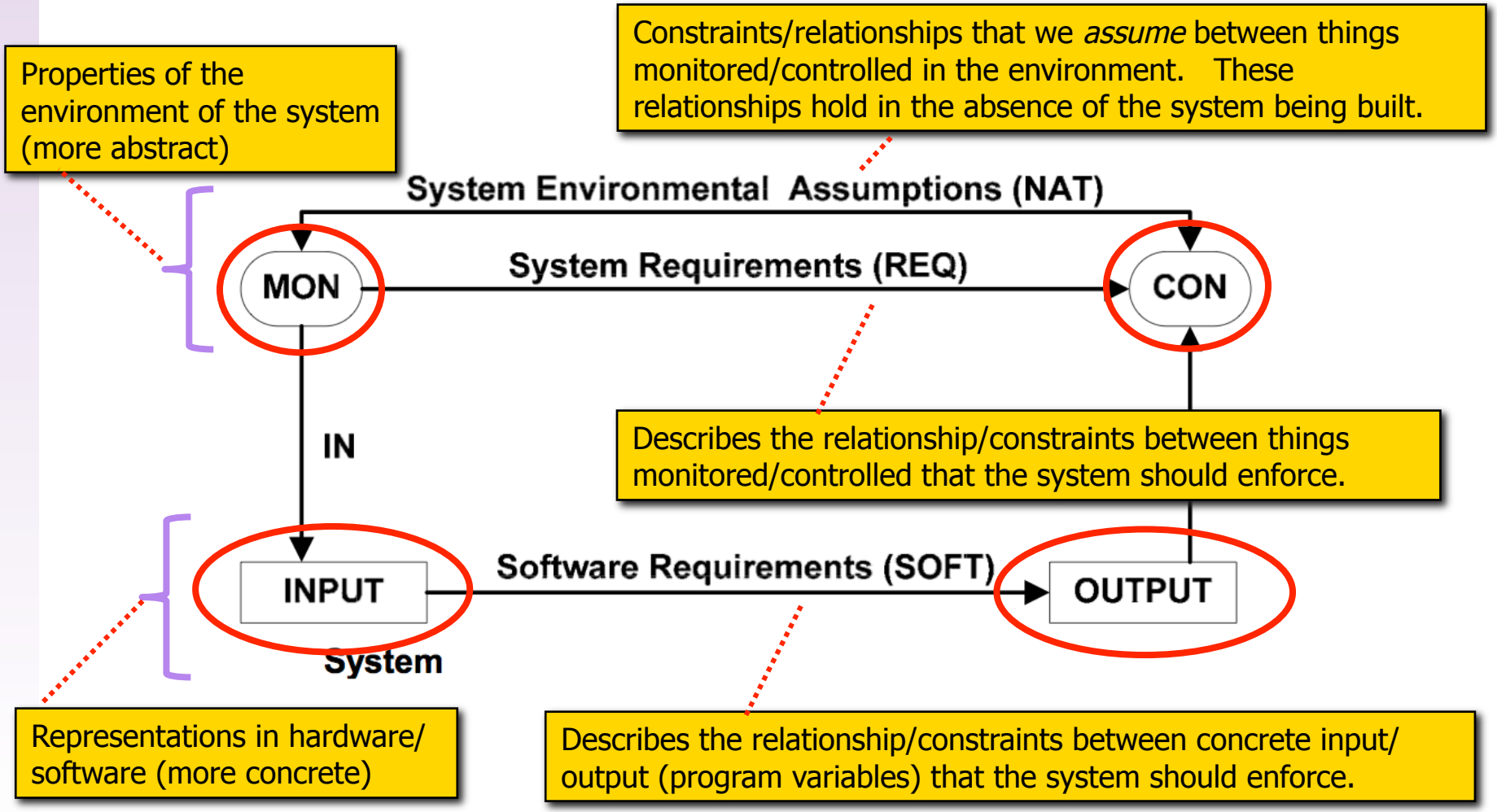
- Produce a complete and consistent set of detailed software behavioral and performance requirements.
- Use the four-variable model to separate system and software functional requirements

Software Requirements: Artifacts

- Set of software behavior goals
- Set of software performance goals

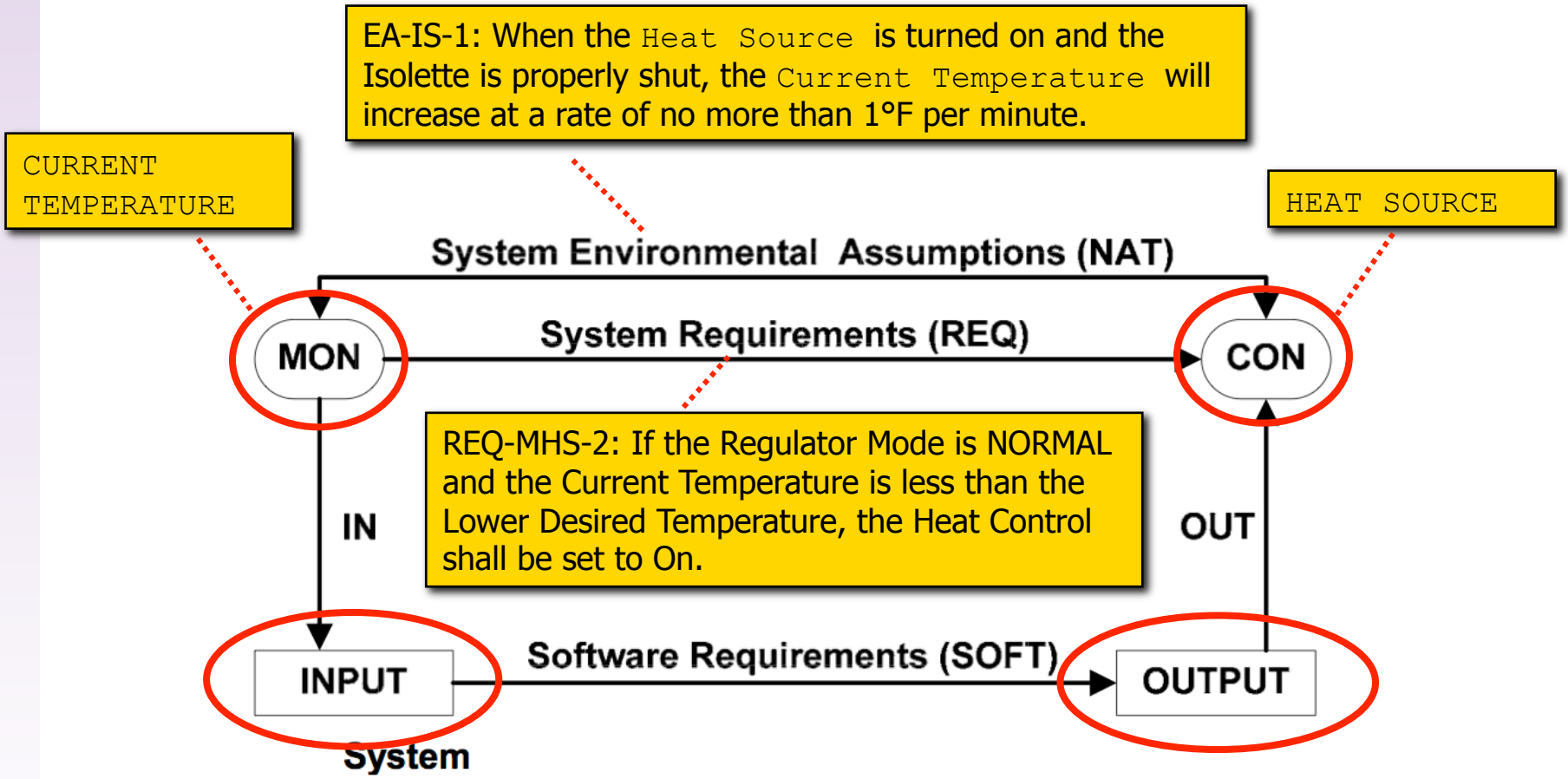
Four Variable Model

The FAA software requirement process is based on the four variable model described by Parnas and Madey



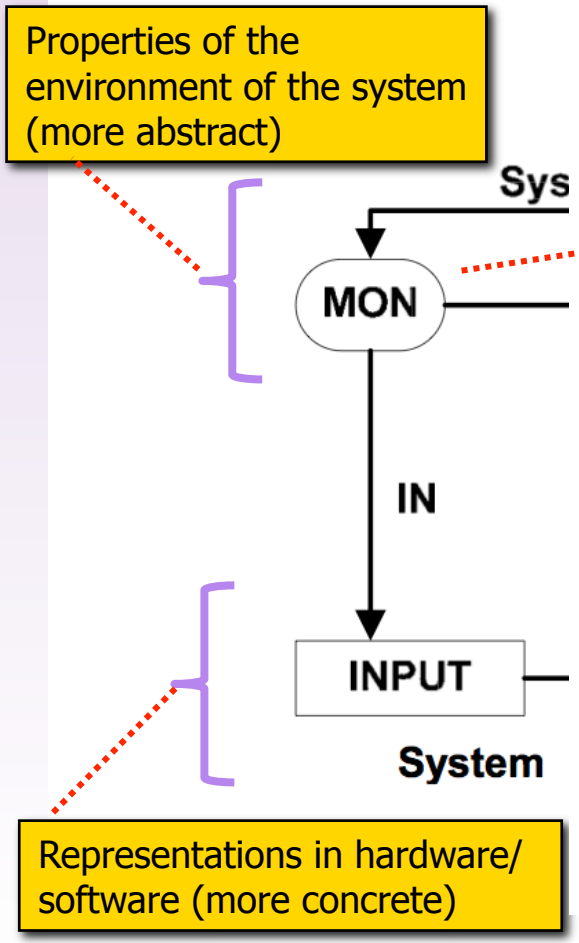
Four Variable Model

Examples from the Isolette



Four Variable Model

The FAA software requirement process is based on the four variable model described by Parnas and Madey



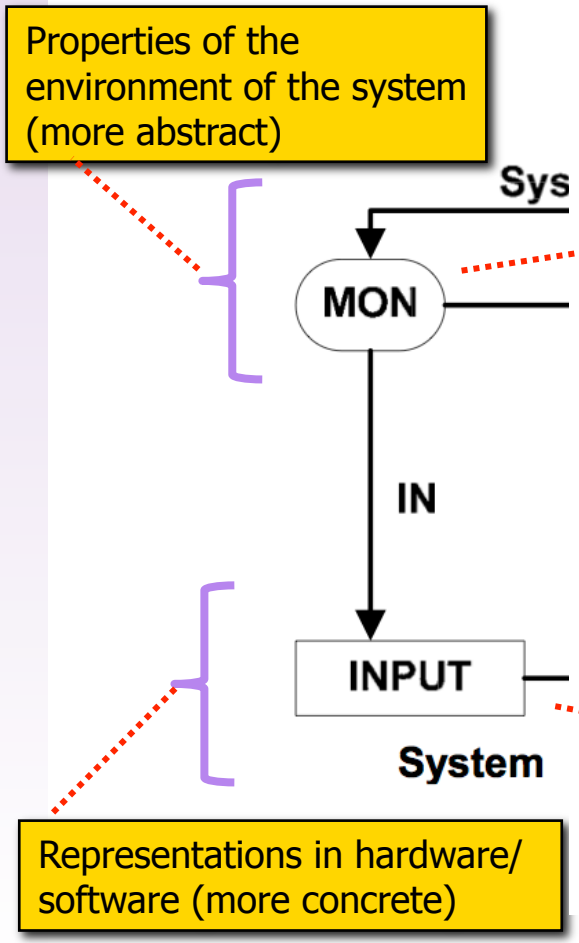
Name	Type	Range	Units
Current Temperature	Real	[68.0..105.0]	°F
	Status	•Invalid, Valid	

Table 17. Input Variable Curr Temp In

Description	Current Temperature of the Isolette in °F multiplied by 256
Data Representation	16-bit unsigned integer
Values	[0..65,535]
Location	Memory words X'0048' and X'0049'
Ideal Value	Current Temperature multiplied by 256
Latency	0.20 millisecond
Accuracy	±20 (i.e., ±0.08°F)

Four Variable Model

The FAA software requirement process is based on the four variable model described by Parnas and Madey



Name	Type	Range	Units
Current Temperature	Real	[68.0..105.0]	°F
	Status	•Invalid, Valid	

Table 18. INPUT Variable Curr Temp Status In

Description	Indication of whether Curr Temp In is valid
Data Representation	8-bit unsigned integer
Values	[0 .. 255]
Initial Value	255
Location	Memory word X'0046'
Ideal Value	No error in sensing Curr Temp In → 0, Error in sensing Curr Temp In → 1 to 255
Latency	0.20 msec
Accuracy	N/A

Four Variable Model

Summary of terms / concepts

- MON – Quantities the system will monitor
- CON – Quantities the system will control
- NAT – Environmental assumptions: relationships maintained by the environment
- REQ – System requirements: relationships maintained by the system
- INPUT – Quantities read by the software
- OUTPUT – Quantities written by the software
- IN – Relationship between one or more monitored variables and a software input
- OUT – Relationship between a controlled variable and one or more software output
- SOFT – The software itself: the mapping from the software's inputs to its outputs

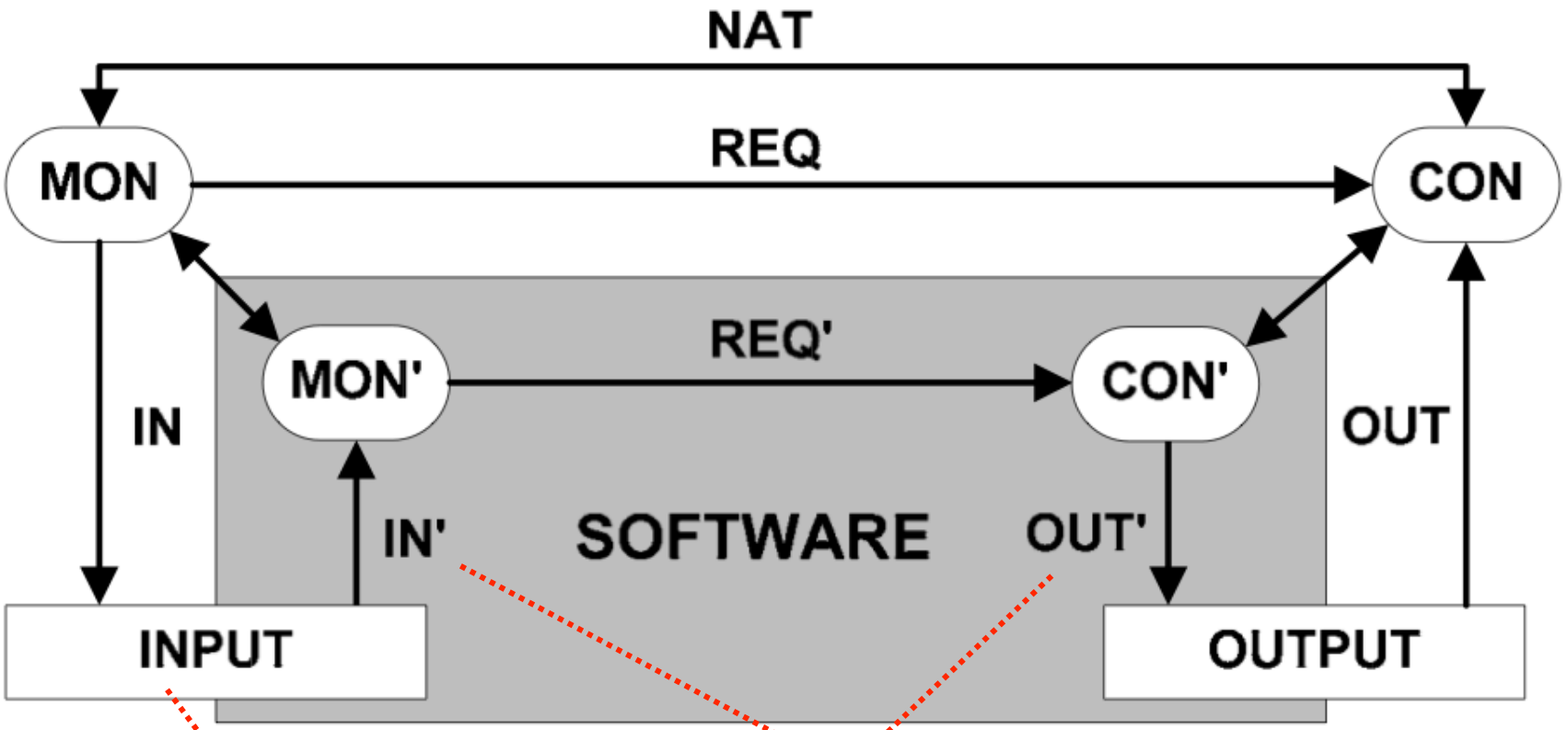
Four Variable Model

Importance

- The Four Variable Model methodology by Parnas and Madey for documenting/organizing system is widely used in industry
- Evolved out of efforts to specify requirements for the A-7 aircraft
- Extended by the Software Productivity Consortium in the Consortium Requirements Engineering (CoRE) methodology
- The CoRE methodology was later used to specify the avionics system of the C-130J aircraft
- *Many follow-on works and references…*

Extended Four Variable Model

Miller and Tribble proposed an extension that aims to better align the software requirements (REQ') with system requirements (REQ)

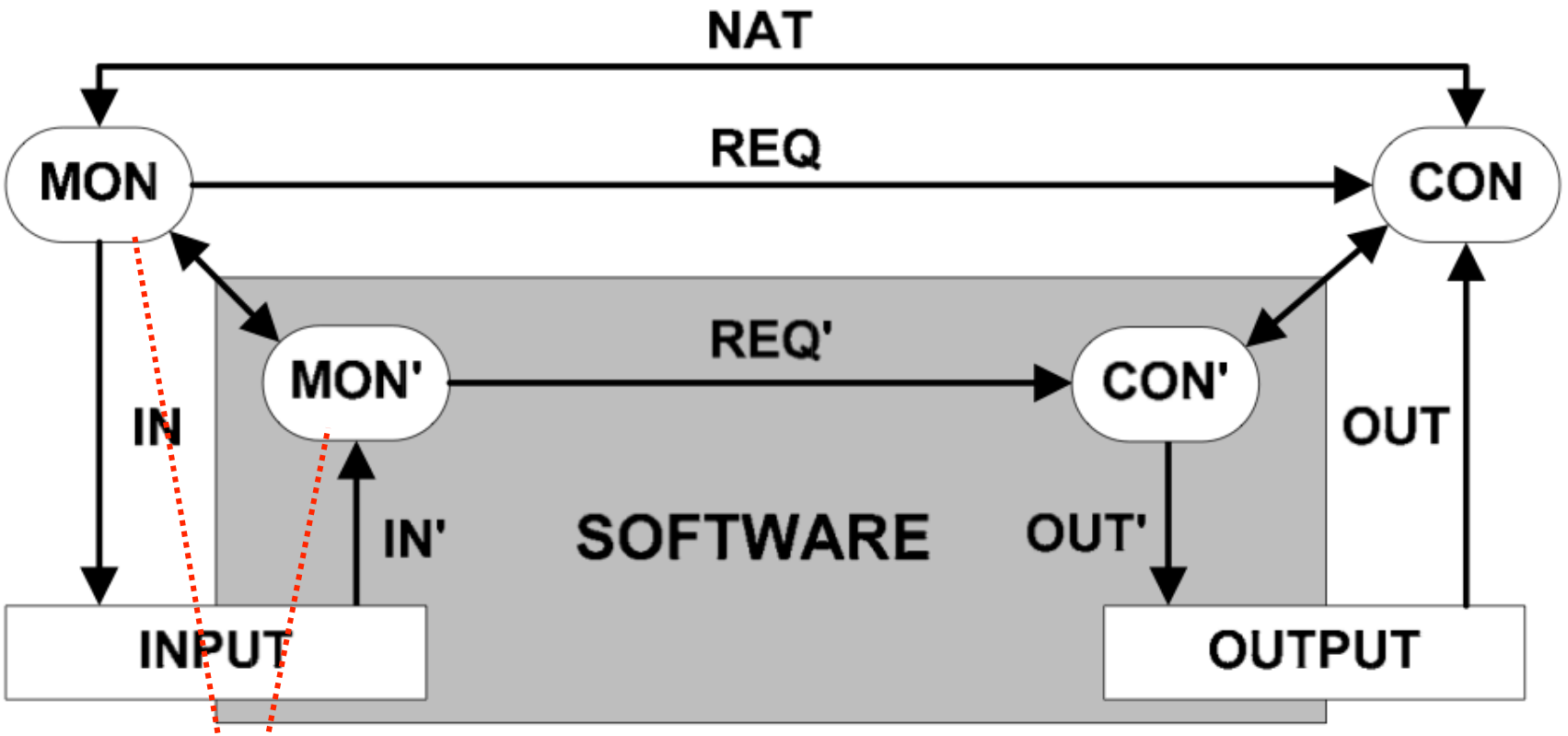


The concrete representation of INPUT may be too low-level for a natural statement of the software requirements.

New IN' / OUT' relations roughly correspond to device driver code that relates low-level hardware-oriented representations to higher-level program variables MON'/CON' more closely aligned with MON/CON

Extended Four Variable Model

We expect some differences between the monitored variables (MON) and their software representations (MON')



For example, we expect there to be latencies between the ideal "real world value" (MON) and its representation in software (MON')

Extended Four Variable Model

Summary of terms / concepts

- IN’ – The inverse of IN: Defines how to recreate an image of MON (termed MON’) from INPUT
- OUT’ – The inverse of OUT: Defines how to recreate OUTPUT from an image of CON (termed CON’)
- REQ’ – Software requirements: relationships maintained by the software

- This extension makes the relationship between system and software requirements direct and straightforward.
- MON’ and CON’ will contain small differences in value from actual values, and have non-zero latencies.

9 Define the Software Requirements

9 Define the Software Requirements: With careful structuring, the software requirements and their architecture can map directly to the system requirements and their architecture. This recommended practice describes how to define the software requirements as a straightforward extension of the system requirements.

9.1 For each input the software must read**, provide a description of anything the software developer must know to access and correctly interpret the input**. This may include an input description, the data format, the range of values it may assume, its location, and any protocols to follow when accessing it.

9.2 For each input the software must read**, provide a specification of its accuracy**, where accuracy refers to the amount that its value may deviate from its ideal value.

9.3 For each input the software must read**, provide a specification of its latency**, where latency refers to the maximum time that its value may lag behind the true value of the monitored variable or variables it represents.

9.4 For each monitored variable, specify how to recreate an image of the monitored variable in software from the input variables.

9.5 For each monitored variable, specify how to recreate its status attribute from the input variables.

9.6 If design choices must be made when recreating a monitored variable in software that may affect the externally visible system behavior or system safety, **flag** those **decisions as derived software requirements** to be reviewed by the safety assessment process.

9.7 For each output the software must set, provide a description of anything the software developer must know to access and correctly set its value. This may include an output description, the data format, the range of values it may assume, its location, and any protocols to follow when accessing it.

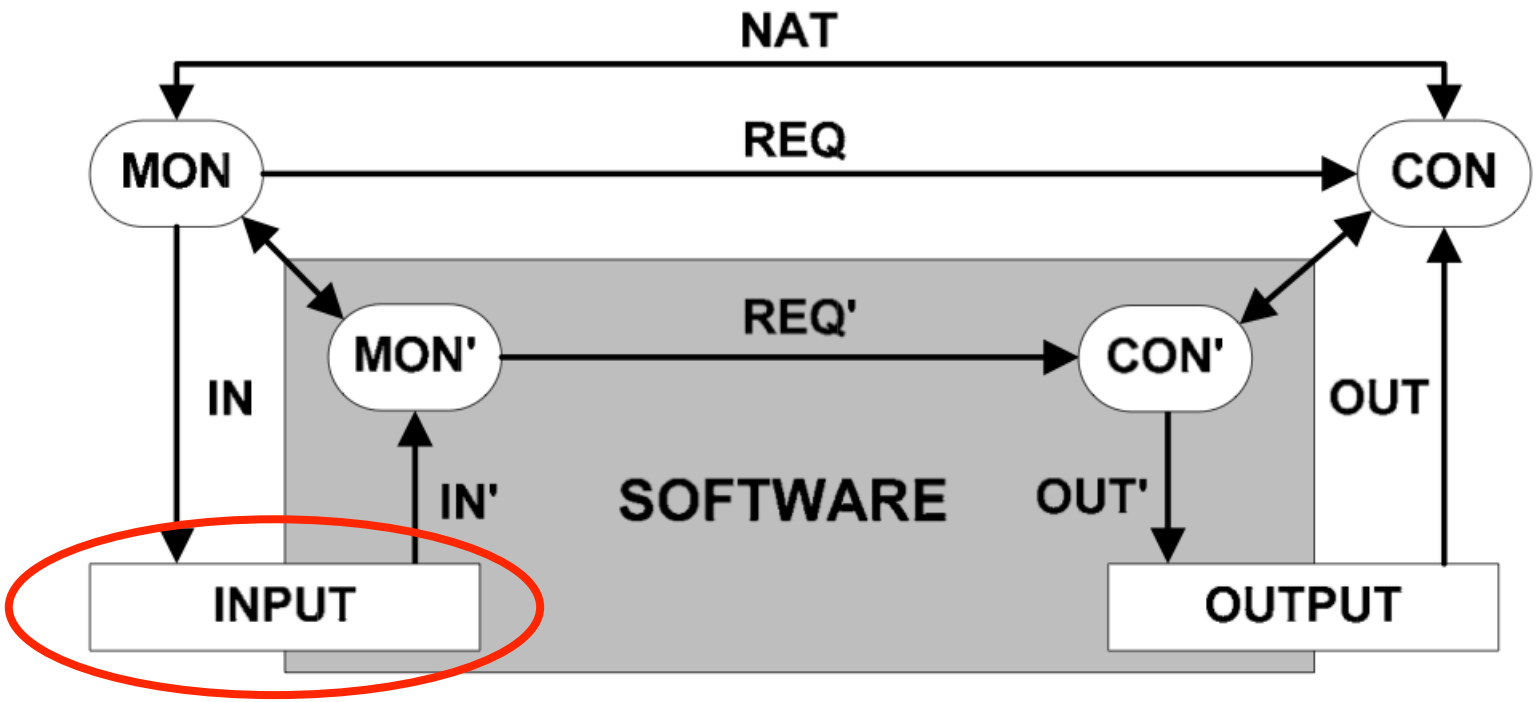
9.8 For each output variable the software must set, provide a specification of its latency, where latency refers to the maximum allowed time from when the output variable is set until it changes the controlled variable value.

9.9 For each output the software must set, provide a specification of its accuracy, where accuracy refers to the amount the affected control variable can diverge from its ideal value.

9.10 For each controlled variable, specify how to set the output variables values based on the value of the controlled variable image in software.

9.11 For each controlled variable, confirm that the latency and accuracy specified in the system requirements can be met given the latency and accuracy of the input and output variables and the computation time of the software.

9.1 Specify the Input Variables



- Describe anything the software developer must know to access and correctly interpret input
  - Includes input description, data format, range of values, location, protocols, etc.

9.1 Specify the Input Variables

- Describe anything the software developer must know to access and correctly interpret input
  - Includes input description, data format, range of values, location, protocols, etc.

Example

Table 17. Input Variable Curr Temp In

Description	Current Temperature of the Isolette in °F multiplied by 256
Data Representation	16-bit unsigned integer
Values	[0..65,535]
Location	Memory words X'0048' and X'0049'
Ideal Value	Current Temperature multiplied by 256
Latency	0.20 millisecond
Accuracy	±20 (i.e., ±0.08°F)

9.1 Specify the Input Variables

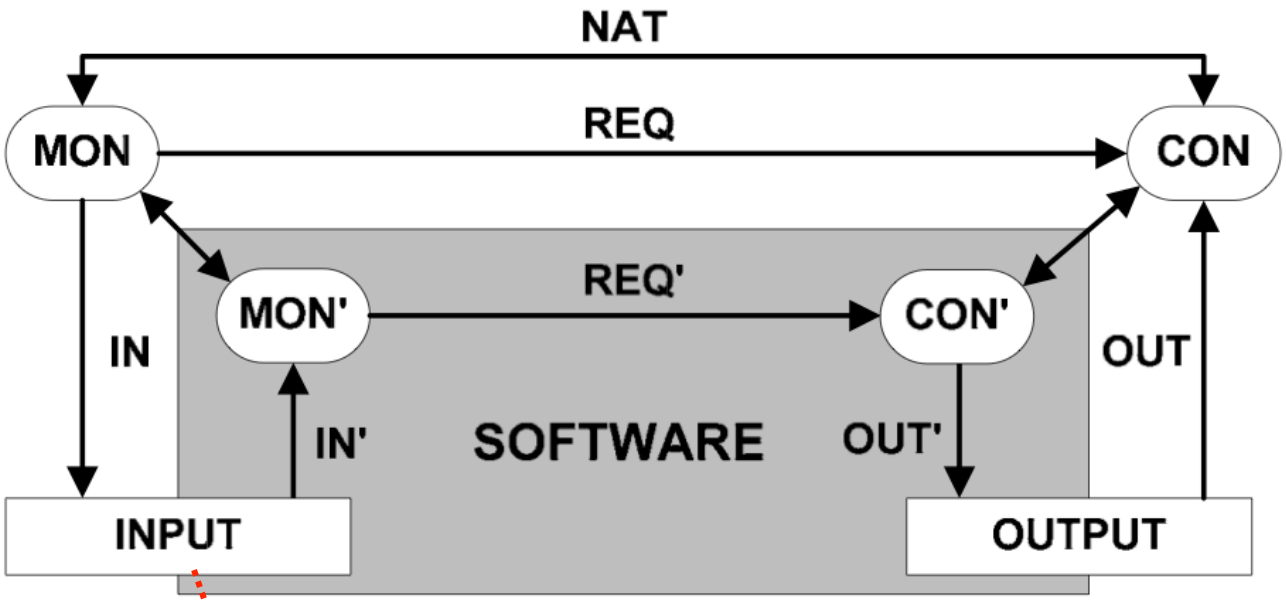


Table 17. Input Variable Curr Temp In

Description	Current Temperature of the Isolette in °F multiplied by 256
Data Representation	16-bit unsigned integer
Values	[0..65,535]
Location	Memory words X'0048' and X'0049'
Ideal Value	Current Temperature multiplied by 256
Latency	0.20 millisecond
Accuracy	±20 (i.e., ±0.08°F)

9.1 Specify the Input Variables

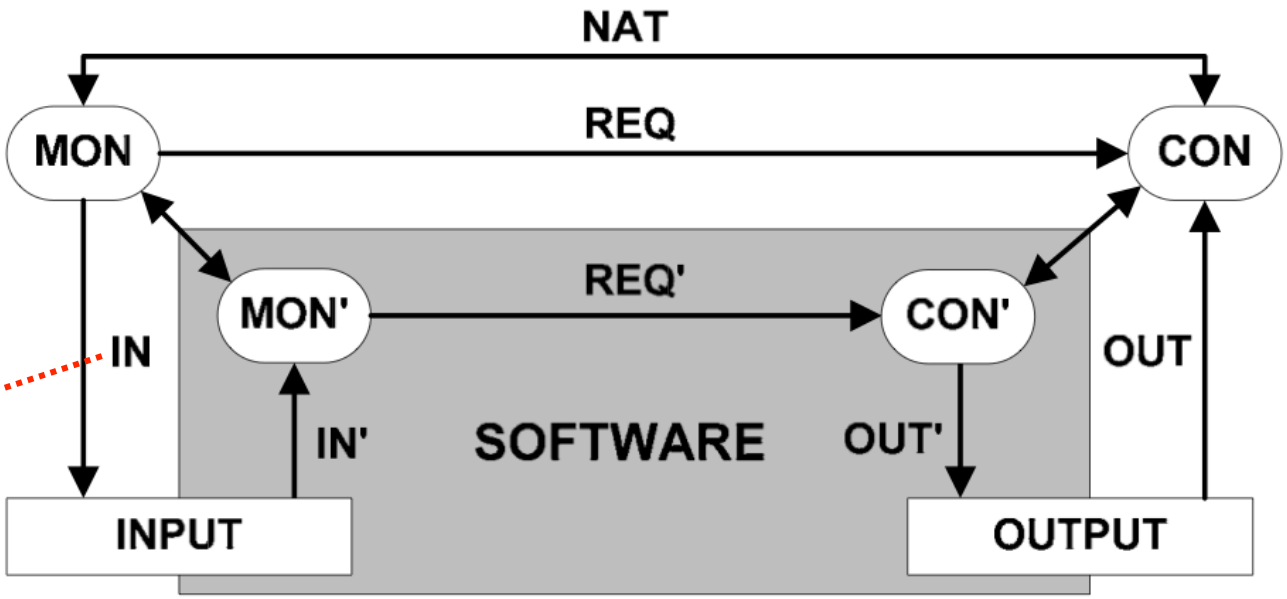


Table 17. Input Variable Curr Temp In

Description	Current Temperature of the Isolette in °F multiplied by 256
Data Representation	16-bit unsigned integer
Values	[0..65,535]
Location	Memory words X'0048' and X'0049'
Ideal Value	Current Temperature multiplied by 256
Latency	0.20 millisecond
Accuracy	±20 (i.e., ±0.08°F)

9.1 Specify the Input Variables

- Describe anything the software developer must know to access and correctly interpret input
  - Includes input description, data format, range of values, location, protocols, etc.

Example

Table 18. INPUT Variable Curr Temp Status In

Description	Indication of whether Curr Temp In is valid
Data Representation	8-bit unsigned integer
Values	[0 .. 255]
Initial Value	255
Location	Memory word X'0046'
Ideal Value	No error in sensing Curr Temp In → 0, Error in sensing Curr Temp In → 1 to 255
Latency	0.20 msec
Accuracy	N/A

Recall that we want to initialize status to *invalid*.

9.2 Specify Input Variable Accuracy

Address the ways in which the software representation (MON') can differ from the ideal (MON)

- Specify the accuracy for each input variable
  - Accuracy refers to the amount that its value may deviate from the ideal value.
- This variance is introduced by hardware (not the software itself)

Table 17. Input Variable Curr Temp In

Description	Current Temperature of the Isolette in °F multiplied by 256
Data Representation	16-bit unsigned integer
Values	[0..65,535]
Location	Memory words X'0048' and X'0049'
Ideal Value	Current Temperature multiplied by 256
Latency	0.20 millisecond
Accuracy	±20 (i.e., ±0.08°F)

9.3 Specify Input Variable Latency

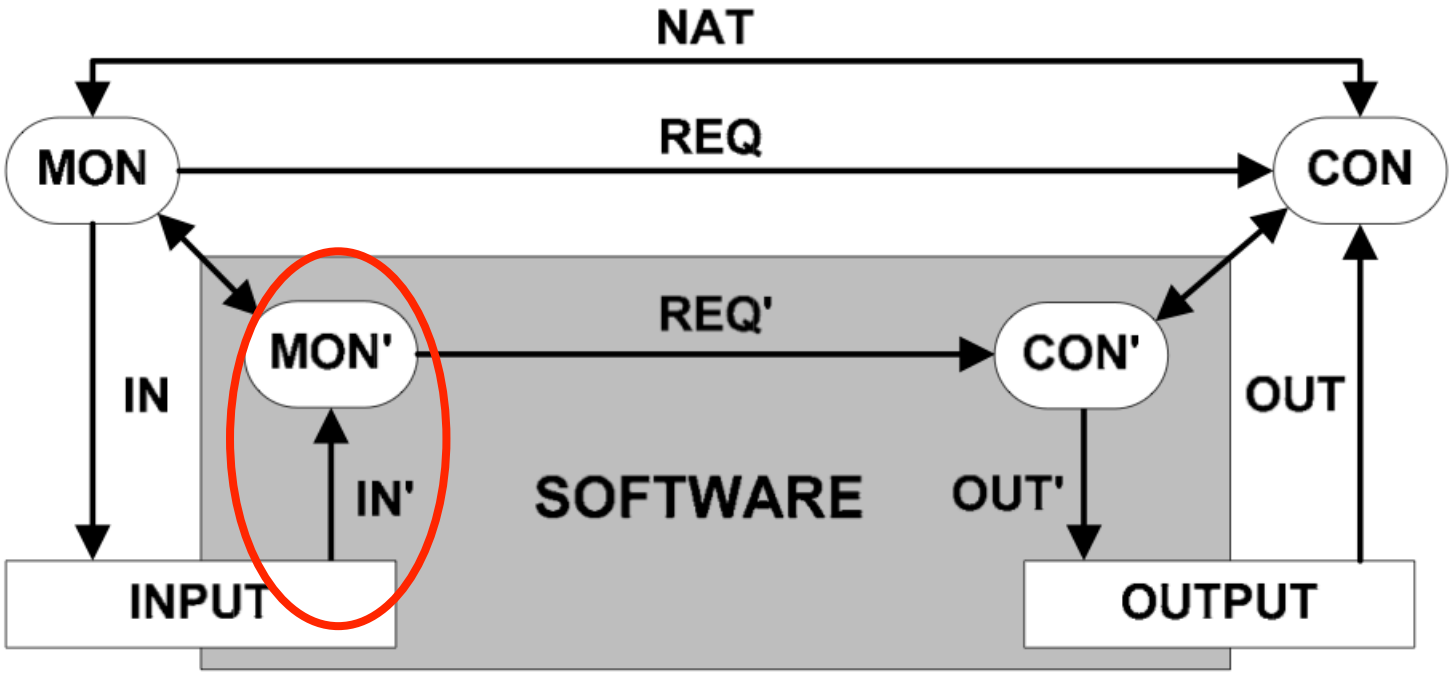
Address the ways in which the software representation (MON') can differ from the ideal (MON)

- Specify the latency for each input variable
  - Latency refers to the maximum time that a value may lag behind the actual value of the monitored variable or set of variables.
- This variance is introduced by hardware.

Table 17. Input Variable Curr Temp In

Description	Current Temperature of the Isolette in °F multiplied by 256
Data Representation	16-bit unsigned integer
Values	[0..65,535]
Location	Memory words X'0048' and X'0049'
Ideal Value	Current Temperature multiplied by 256
Latency	0.20 millisecond
Accuracy	±20 (i.e., ±0.08°F)

9.4 Specify IN' for Monitored Variables



- Specify how to recreate an image of the monitored variable in software from the input variables
  - This can be a simple one-to-one mapping
  - It can also be a complex many-to-one function

9.4 Specify IN' for Monitored Variables

- Specify how to recreate an image of the monitored variable in software from the input variables
  - This can be a simple one-to-one mapping
  - It can also be a complex many-to-one function

Example

Table 19. IN' Relation for Value of Current Temperature'

	Curr Temp In < 17,408	17,408 ≤ Curr Temp In ≤ 26,880	Curr Temp In > 26,880
Value =	68.0	Curr Temp In/256.0	110.0

Note that we are "saturating" the computed value of IN' at 68.0 and 110.0 (typo in document, should be 105.0) to ensure that we always stay within the specified range of the monitored variable.

Name	Type	Range	Units	Physical Interpretation
Current Temperature	Real	[68.0..105.0]	°F	Current air temperature inside Isolette
	Status	•Invalid, Valid		

9.5 Specify Status for Monitored Variables

- Specify how to recreate the status of the monitored variable in software
  - Indicates the "health" of the variable

Example

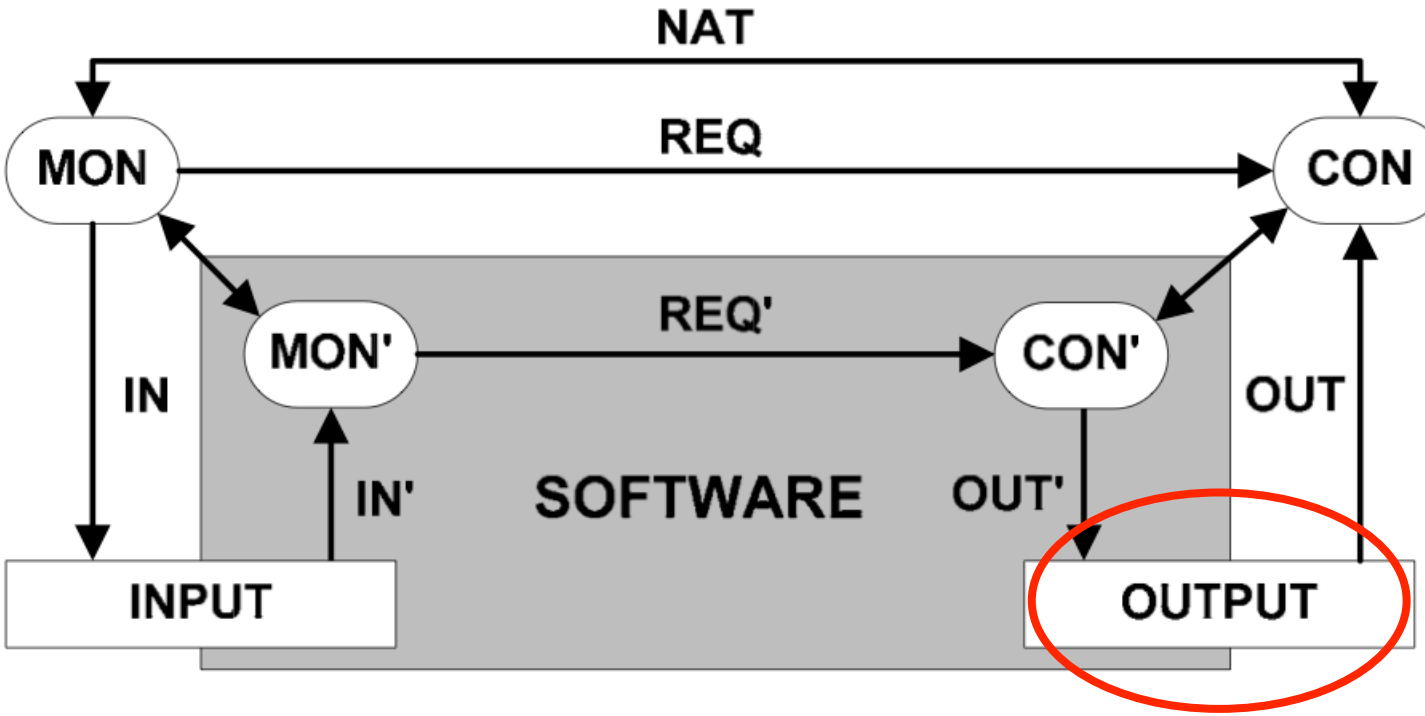
Table 20. IN' Relation for Status of Current Temperature'

		Curr Temp Status In = 0	
	Curr Temp Status In ≠ 0	Curr Temp In < 17,408 OR Curr Temp In > 26,880	17,408 ≤ Curr Temp In AND Curr Temp In ≤ 26,880
Status =	Invalid	Invalid	Valid

9.6 Flag Design Decisions as Derived Requirements

- The choice to saturate the value of Current Temperature' is an example of a derived requirement, as defined in RTCA DO-178B [27] and DO-248B [28],
  - i.e., a requirement that is not directly traceable to a higher-level requirement
- Indicated by the fact that other design choices could be made that would affect the externally visible system behavior, while still meeting the system requirements.
  - For example, the range shown in Table 20 could be expanded from 17,000 to 27,000.
  - This would reduce the risk of putting the thermostat into a failed state due to noise in the value of Curr Temp In, but opens the possibility of operating with a dangerously high temperature.
- Decisions such as these that affect the externally visible system behavior should be flagged as a derived software requirement to ensure they are provided to the system safety assessment process, as called out in DO-178B.

9.7 Specify the Output Variables



- Describe anything the software developer must know to access and correctly set output values
  - Includes output description, data format, range of values, location, protocols, etc.

9.7 Specify the Output Variables

- Describe anything the software developer must know to access and correctly set output values
  - Includes output description, data format, range of values, location, protocols, etc.

Example

Table 21. OUTPUT Variable Heat Control OUT

Description	Command to turn heat source on or off
Data Representation	8-bit unsigned integer
Values	[0 .. 255]
Location	Memory word X'0060'
Ideal Value	0 → heat control = off 2 to 255 → heat control = on
Latency	0.60 msec
Accuracy	N/A

9.7 Specify the Output Variables

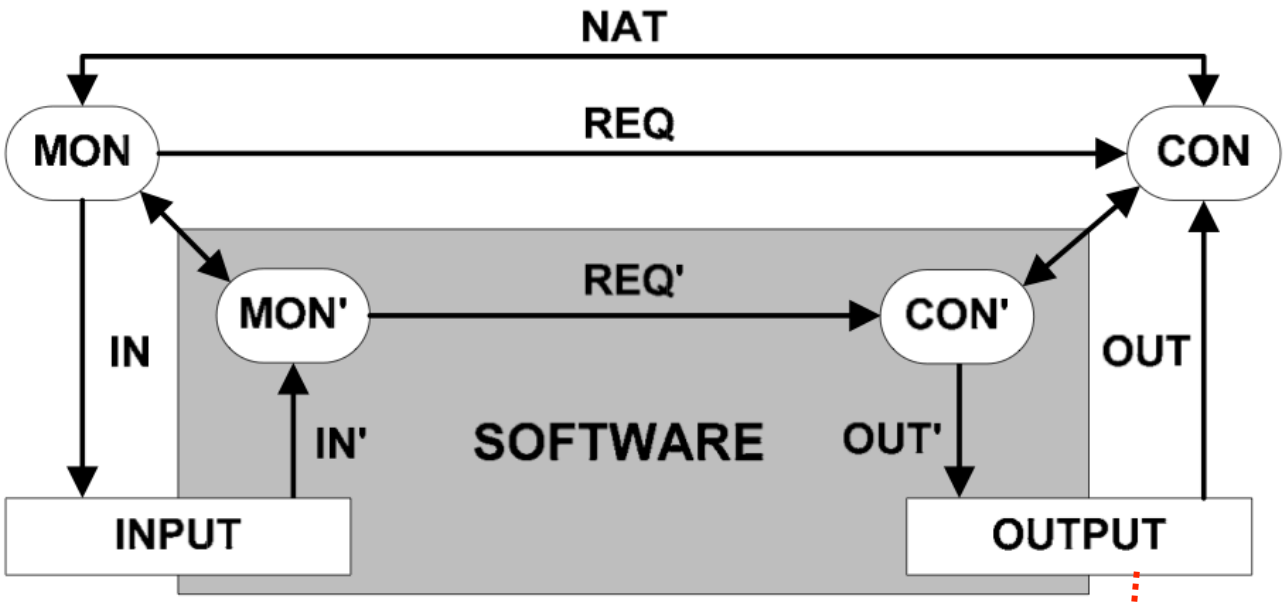


Table 21. OUTPUT Variable Heat Control OUT

Description	Command to turn heat source on or off
Data Representation	8-bit unsigned integer
Values	[0 .. 255]
Location	Memory word X'0060'
Ideal Value	0 → heat control = off 2 to 255 → heat control = on
Latency	0.60 msec
Accuracy	N/A

9.7 Specify the Output Variables

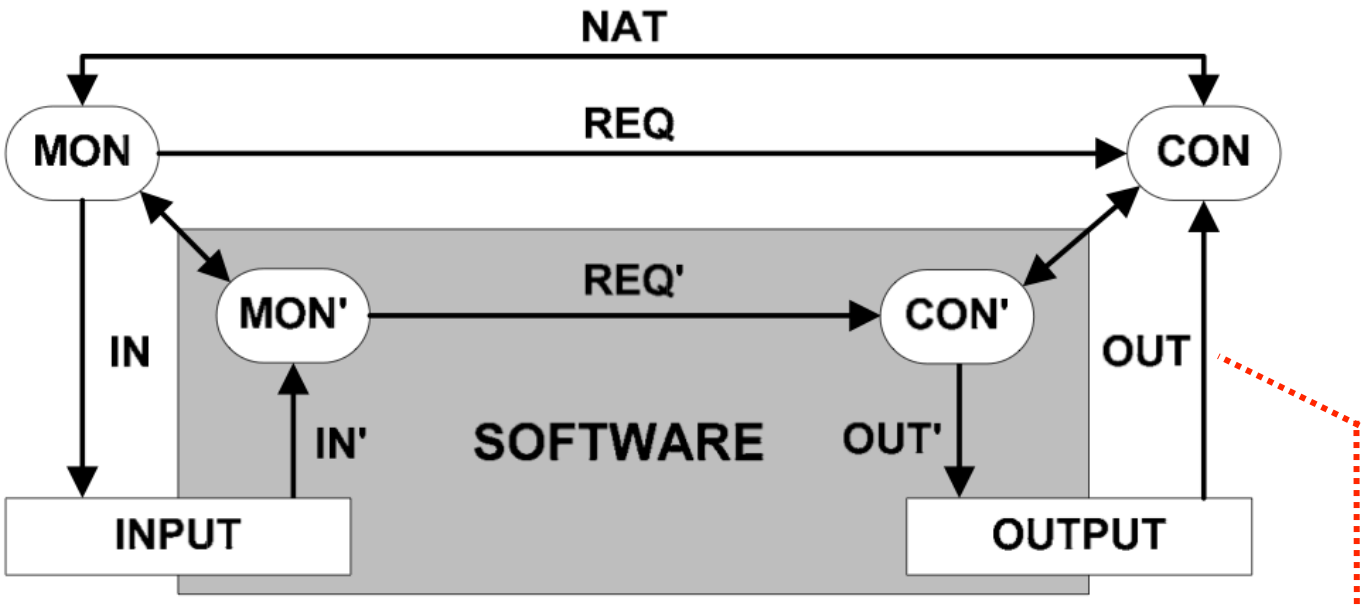


Table 21. OUTPUT Variable Heat Control OUT

Description	Command to turn heat source on or off
Data Representation	8-bit unsigned integer
Values	[0 .. 255]
Location	Memory word X'0060'
Ideal Value	0 → heat control = off 2 to 255 → heat control = on
Latency	0.60 msec
Accuracy	N/A

9.8 Specify Output Variable Latency

Address the ways in which the software representation (CON') can differ from the ideal (CON)

- Specify the latency for each output variable.
  - Latency refers to the maximum allowed time from when the output variable is set until it changes the controlled variable value.

Table 21. OUTPUT Variable Heat Control OUT

Description	Command to turn heat source on or off
Data Representation	8-bit unsigned integer
Values	[0 .. 255]
Location	Memory word X'0060'
Ideal Value	0 → heat control = off 2 to 255 → heat control = on
Latency	0.60 msec
Accuracy	N/A

9.9 Specify Output Variable Accuracy

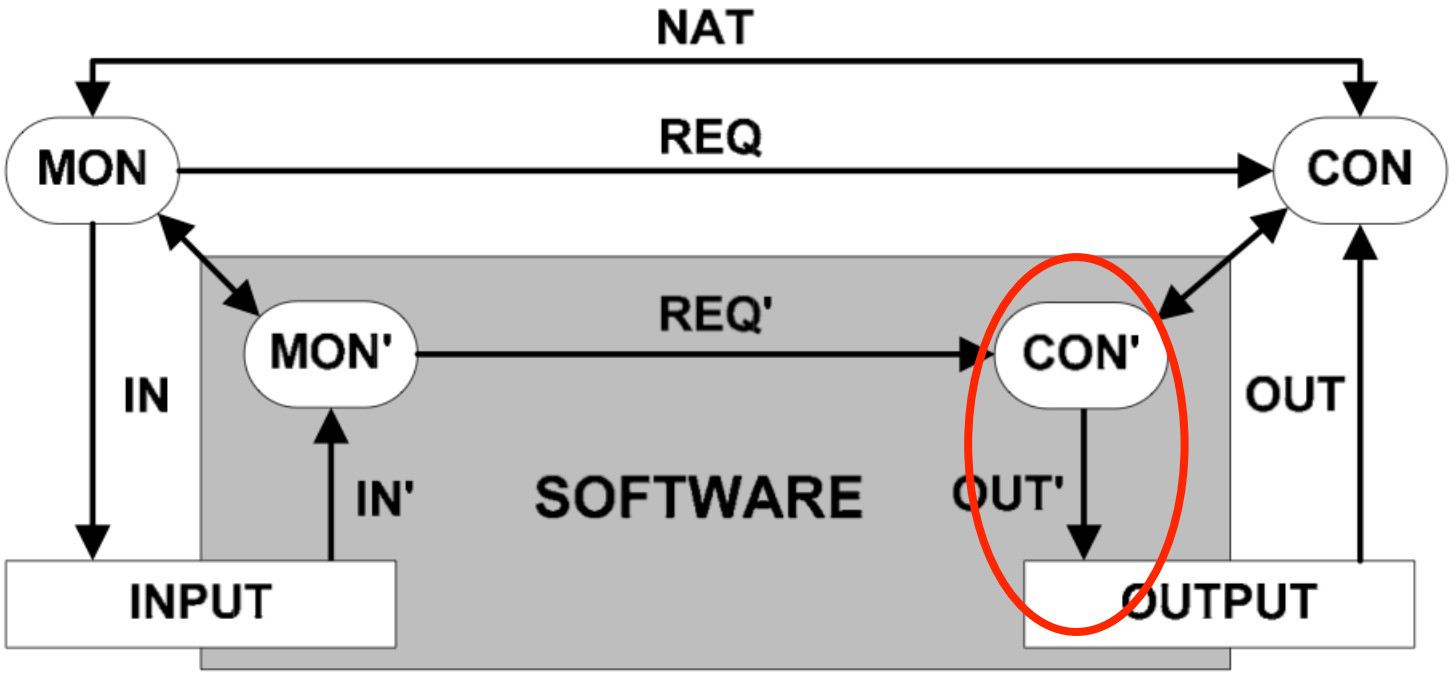
Address the ways in which the software representation (CON') can differ from the ideal (CON)

- Specify the accuracy for each output variable
  - Accuracy refers to the maximum amount that the affected control variable can diverge from its ideal value.
  - Note that discrete outputs will have an accuracy of N/A.

Table 21. OUTPUT Variable Heat Control OUT

Description	Command to turn heat source on or off
Data Representation	8-bit unsigned integer
Values	[0 .. 255]
Location	Memory word X'0060'
Ideal Value	0 → heat control = off 2 to 255 → heat control = on
Latency	0.60 msec
Accuracy	N/A

9.10 Specify OUT' for Monitored Variables



- Specify how to set each output variable's value(s) based on the value of the controlled variable image
  - This can be a simple one-to-one mapping
  - It can also (potentially) be a one-to-many derivation

9.10 Specify OUT' for Monitored Variables

- Specify how to set each output variable's value(s) based on the value of the controlled variable image
  - This can be a simple one-to-one mapping
  - It can also (potentially) be a one-to-many derivation

Example

Table 22. OUT' Relation for Heat Control

	Heat Control' = Off	Heat Control' = On
Value =	0	[1..255]

Assessment

- With the definition of the INPUT and OUTPUT variables and the IN' and OUT' relationships, the software requirements specification is essentially complete.
- The software developer now knows how to recreate the monitored variable images in software and how to set the output variables based on the images of the controlled variables in the software.
- The other information the software developer needs to know is how to change the controlled variable image in software when the monitored variable image in software change (i.e., REQ').
- However, the ideal value function for REQ' is identical to the ideal value function defined for the detailed system requirements (i.e., the REQ relation) defined in Section 2.8.

9.11 Confirm Overall Latency and Accuracy

- Add all latencies along a variable's possible derivations
  - Verify they do not exceed the maximum allowable values
- Add maximum acceptable inaccuracies along a variable's possible derivations

Summary

- For each input and output, determine the maximum allowable latency and inaccuracy
- Determine how to find images of monitored variables from inputs, and use images of controlled variables to control outputs.
- Flag new design decisions for later safety review.

For You To Do

Acknowledgements

- The material in this lecture is based almost entirely on
  - *FAA* DOT/FAA/AR-08/32*, Requirements Engineering Management Handbook.* David L. Lempia & Steven P. Miller.

High and Low Level Software Requirements

- DO-178B specifies that software requirements should be organized into high- and low-level requirements.
  - High-Level Requirements: “Software requirements developed from analysis of system requirements, safety-related requirements, and system architecture”
  - Low-Level Requirements: “Software requirements derived from high-level requirements, derived requirements, and design constraints from which source code can be directly implemented without further information.”

High and Low Level Software Requirements

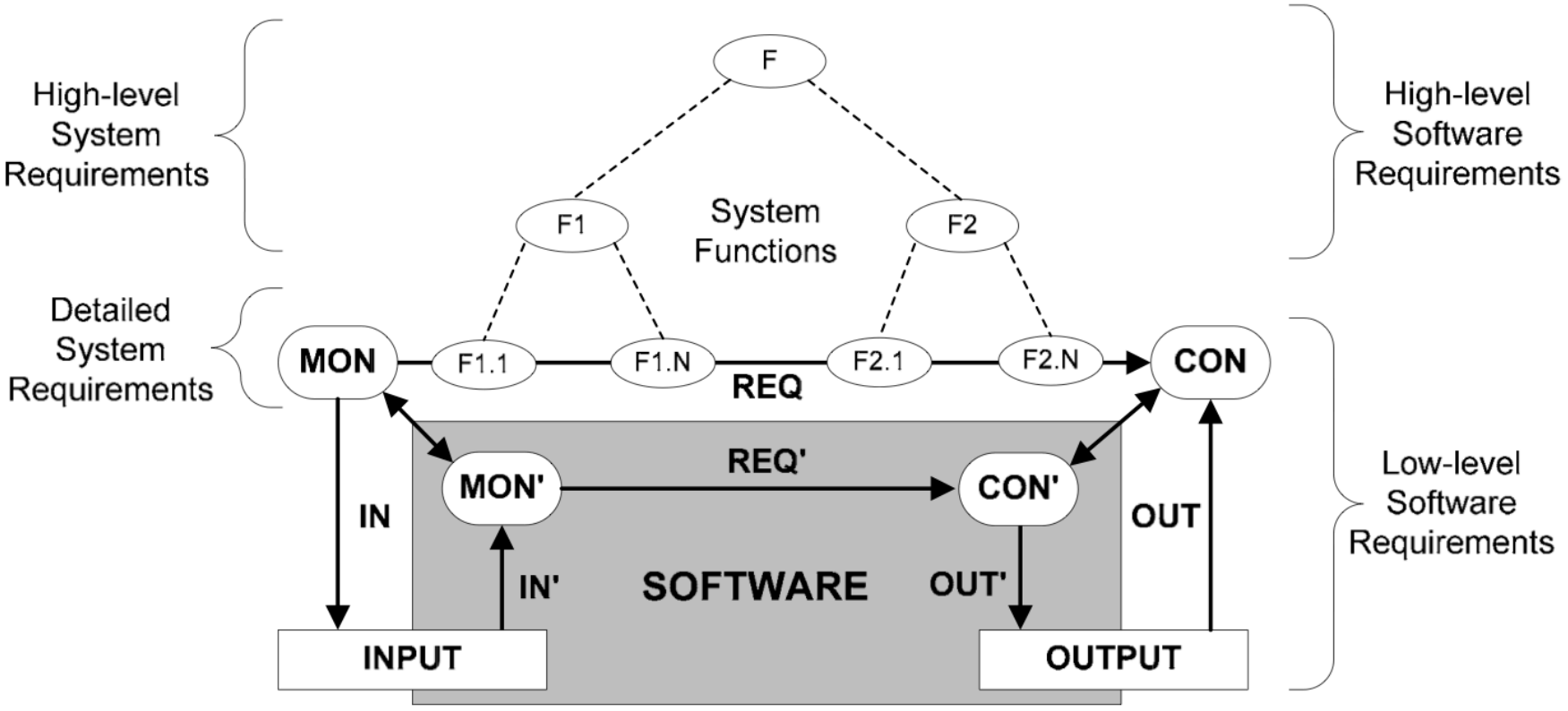


Figure 13. High- and Low-Level Software Requirements