FAA Requirements Engineering Management Handbook

5. Develop the Functional Architecture

SAnToS Laboratory
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

Functional Architecture: Goals

What are we trying to achieve with this step in the requirements engineering process?

- Start to organize the system functions identified when we created our use cases
  - What functions should be grouped together?
  - What are the dependencies between the groups (what data flows between them?)

One of the outputs of previous step in the REMH…

While creating the use cases, a preliminary list of System Functions should be assembled. These will be used as input to the activity described in section 2.5. Such a preliminary list of System Functions for the Isolette Thermostat is shown in table 6.

Table 6. Preliminary Set of Isolette Thermostat Functions

Turn the thermostat on and off	Indicate the thermostat status
Set the Desired Temperature	Turn heat source on and off
Display the current temperature	

Functional Architecture: Goals

What is a "functional architecture"?

- A functional architecture is an organization of the system's functions into groups (which may eventually become subsystems/modules, etc.) where
  - the functions in each group are cohesive/similar – they work on the same data, the same set of monitored/controlled variables
  - the primary flows of information (dependences) between groups are captured
- A functional architecture tends to be conceptual and does not necessarily precisely specify interfaces, mapping to implementation strategies, etc.
- A functional architecture can form the input to subsequent steps in the design process which refine the architecture to meet implementation or safety goals.

Functional Architecture: Goals

What are we trying to achieve with this step in the requirements engineering process?

- Enhance readability
  - Easier to understand requirements if they are grouped according to related functions
- Improve robustness in the presence of change
  - Requirements pertaining to the same function are likely to change together
- Organization should reflect project's needs
  - Readability vs. findability
- Nest functional groups as necessary
- Provide traceability of requirements
  - Lower level requirements related to subfunction G can be easily traced to high level requirements in the function group F of which G is a member

Functional Architecture: Artifacts

What artifacts should we produce as a result of this step?

- List of the logically independent functions of the system (potentially nested)
- Data flow / dependency diagrams representing these functions and flows of data/information between them
- High-Level requirement lists

5 Develop the Functional Architecture

5 Develop the Functional Architecture: To enhance readability of the requirements and to make them robust in the face of change, the requirements are organized into functions that are logically related with minimal dependencies between the functions. To allow the requirements to scale to large systems, functions are broken down into smaller functions.

5.1 Organize System Functions into groups that are closely related and likely to change together.

5.2 Use data flow diagrams to graphically depict System Functions and their dependencies.

5.3 Minimize dependencies between functions by ensuring that the information shared between functions represent stable, high-level concepts from the problem domain that are unlikely to change. Push volatile dependencies as far down in the function hierarchy as possible.

5.4 Define the type, range, precision, and units of all internal variables introduced to specify data dependencies between functions.

5.5 Organize large requirements specifications into multiple levels by nesting functions and data dependencies.

5.6 If providing high-level requirements, define only what can be stated at that level of the hierarchy. Do not use terms defined at lower levels of the hierarchy.

5.7 If providing high-level requirements for functions, **avoid incorporating rationale** into the requirement in an attempt to specify details more appropriately specified in a lower-level function.

5.1 Organize System Functions into Related Groups

Getting started…

- Group logically similar functions that are likely to change together
  - Begin with system functions identified in use case (operational concept) construction in Section 3

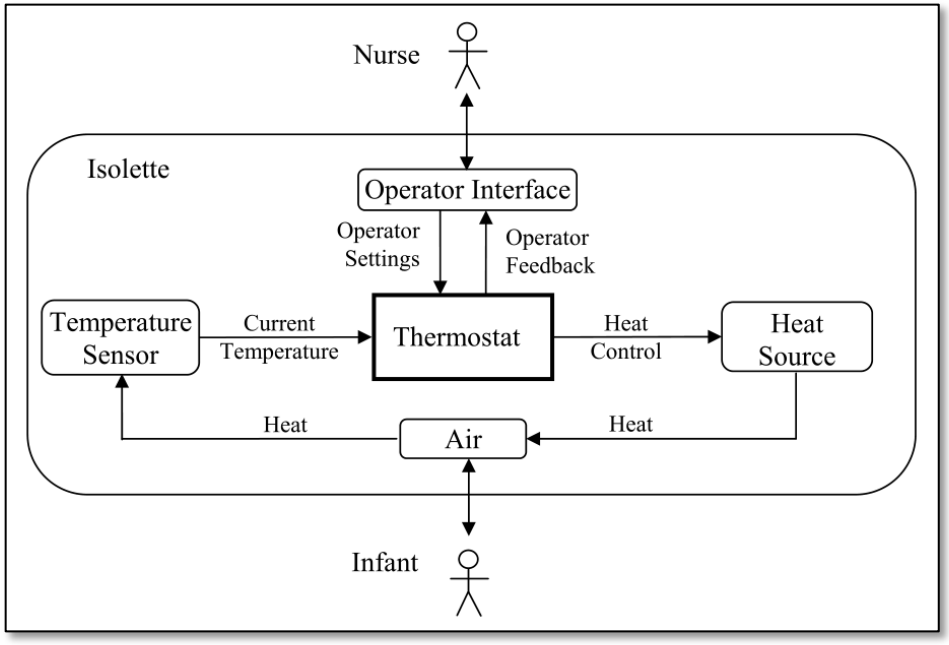
5.1 Organize System Functions into Related Groups

Isolette Example

Table 6. Preliminary Set of Isolette Thermostat Functions

Turn the thermostat on and off	Indicate the thermostat status
Set the Desired Temperature	Turn heat source on and off
Display the current temperature	

- Which functions interact with the same external entity?
- Which functions work on related monitored/ controlled variables?



5.1 Organize System Functions into Related Groups

Isolette Example

Table 6. Preliminary Set of Isolette Thermostat Functions

Turn the thermostat on and off	Indicate the thermostat status
Set the Desired Temperature	Turn heat source on and off
Display the current temperature	

- Operator facing functions (power switch, temperature display, etc.) can be grouped into "Manage Operator Interface"
- Turning heat source on/off could be allocated to "Manage Heat Source"

5.2 Use Data Flow Diagrams

Interactions between function groups

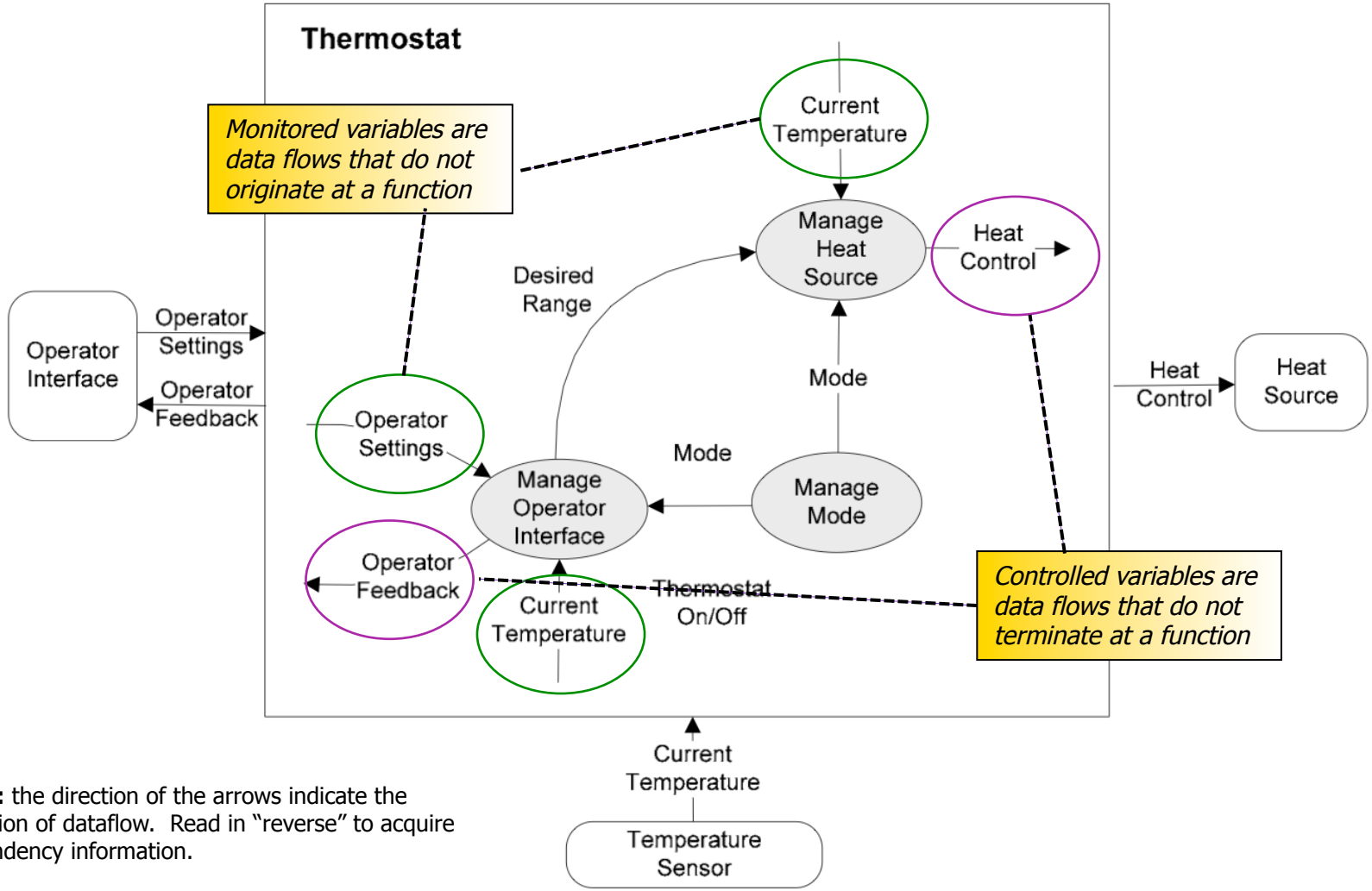
- Dependency diagrams are a simple way of showing dependencies of system functions
- Monitored variables are…
  - arrows that do not originate from a function
- Controlled variables are…
  - arrows that do not terminate at a function

Data Flow Diagram Example

Classroom Exercise – Draw the Dataflow Diagram for Isolette example

Data Flow Diagram Example

Classroom Exercise – Draw the Dataflow Diagram for Isolette example



Note: the direction of the arrows indicate the direction of dataflow. Read in "reverse" to acquire dependency information.

5.3 Minimize Dependencies Between Functions

Evaluate decomposition by examining dependences

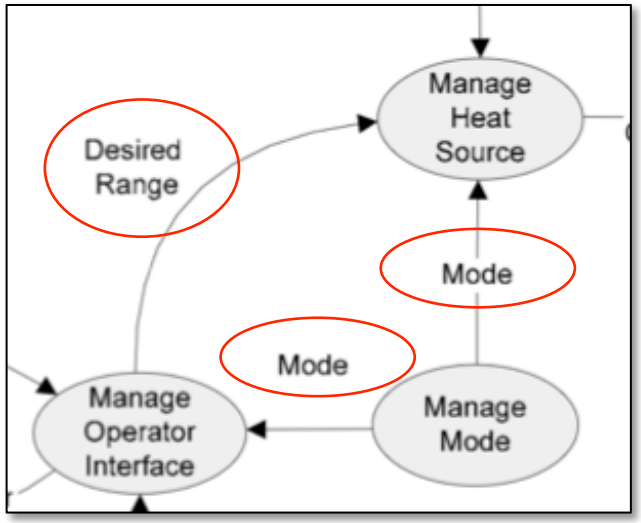
- High-level dependencies (arrows in dependency diagrams) should be as stable as possible
  - The more volatile a dependency, the further it should be pushed down the hierarchy
- This requires fewer changes when the dependency changes

Need example!

5.4 Define Internal Variables

Data flows between functions correspond to "internal variables" or communication links between components

- Internal variables can either be defined:
  - In a single location in a dependency diagram, or
  - Within the specification of the function that the variable is local to
- Remember to define type, range, precision and units of all internal variables



5.5 Nest Functions and Data Dependencies

General advice (which happens to be hard to illustrate using the Isolette example because it is so small)

- Large systems may become unmanageable due to an excessive number of functions
- Nesting functions allows diagrams to remain readable
- Dependencies can be similarly "bundled" where a collection can be represented by a single arrow
- The hierarchy is best preserved if this decomposition is done recursively

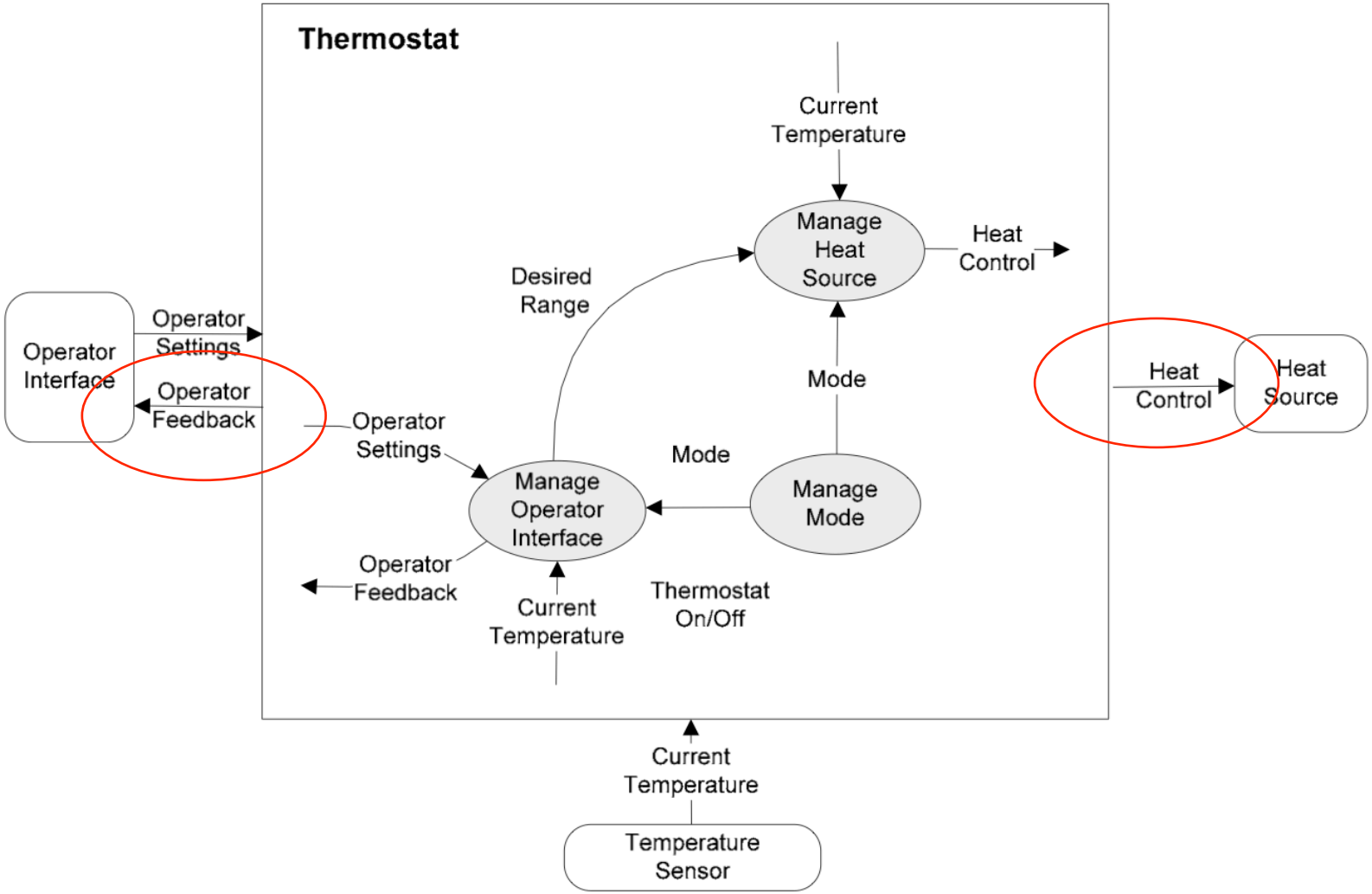
5.6 Provide High-Level Requirements

From dependency diagrams to requirements

- Dependency diagrams can implicitly define high-level requirements
  - If desired, these can be made explicit, but care should be taken to avoid details as this negates the benefits of decomposition
- Avoid using terms defined at lower-levels of the hierarchy
  - This will make lower-level requirements ambiguous or duplicitous
  - Prevents the requirements from being self-contained

High-Level Requirements for the Thermostat

Let's consider some of the most basic control aspects…



High-Level Requirements for the Thermostat

High-level requirements related to control…

REQ-1	The Thermostat Function shall set the value of the Heat Control. Rationale. A primary function of the Thermostat is to turn the Heat Control on and off to maintain the Current Temperature in the Isolette within the Desired Temperature Range.
REQ-2	The Thermostat Function shall set the value of the Operator Feedback. Rationale. The Thermostat provides the Operator Feedback to the Operator Interface that it will use it to report the overall status of the Thermostat to the Operator.

Figure 4. High-Level Requirements for the Thermostat Function

High-Level Requirements for the Thermostat

Note: It is tempting to be more specific in the requirements statements at this point…

Consider the following…

> REQ-1—The Thermostat shall set the value of the Heat Control to maintain the Current Temperature in the Isolette within the Desired Temperature Range.

Two problems…

- It is only True when the Thermostat is in its NORMAL mode of operation.
  - During the INIT or FAILED mode, the Heat Control is set to the value of off.
- The system modes are not defined until the next lower level of refinement, so specifying precisely how the Heat Control is to be set requires using terms and definitions that are not available at this level.

5.7 Do Not Incorporate Rationale into Requirement Statements

Rationale is important, but keep it separate from the requirements statement

- Incorporating rationale confuses what with why
- If necessary, provide rationale but set it off from the requirement text visually
- Note also that lower-level functions can often more appropriately specify rationale

Summary

This step from the REMH begins our organization of requirements statement and architecture

- Collect initial information about functions from use cases
- Organize functions into logically related groups
- Nest functional requirements hierarchically
- Use dependence diagrams to guide further refinement
- Minimize dependence, especially if it involves volatility
- Keep requirements specific to their level of the hierarchy

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.