A Semi-Formal Method to Verify Correctness of Functional Requirements Specification of Complex Embedded Systems

Nihal Kececi
University of Maryland Department of Material and Nuclear Engineering,
Reliability Engineering Program, College Park, MD 20742, U.S.A.
nkececi@eng.umd.edu

Wolfgang A. Halang
FernUniversitat, Fachbereich Elektrotechnik und Informationstechnik,
58084 Hagen, Germany
wolfgang.halang@fernuni-hagen.de

Alain Abran
École de Technologie Supérieure ETS
1100 Notre-Dame Ouest, Montréal, Québec Canada H3C 1K3
aabran@ele.etsmtl.ca

Purpose

- The primary purpose of this work is to develop a methodology for "translating" functional user requirements into a graphic form.

- The approach (GRA) provides communication language in two directions:
  - For user/system engineer: building functional specifications
  - For software developer: verifying functional requirements

Functional specifications are important

- Several studies have shown that about 50% of software faults can be traced back to requirements

- During the integration testing of Voyager and Galileo spacecraft,
  - 197 faults were characterized as the cause of catastrophic failure
    - 3 were coding errors
    - 194 were traced back to a problem in the specifications.

Why?

- Experience has shown that some of the reasons why more errors tend to occur in the requirements phase are as follows:

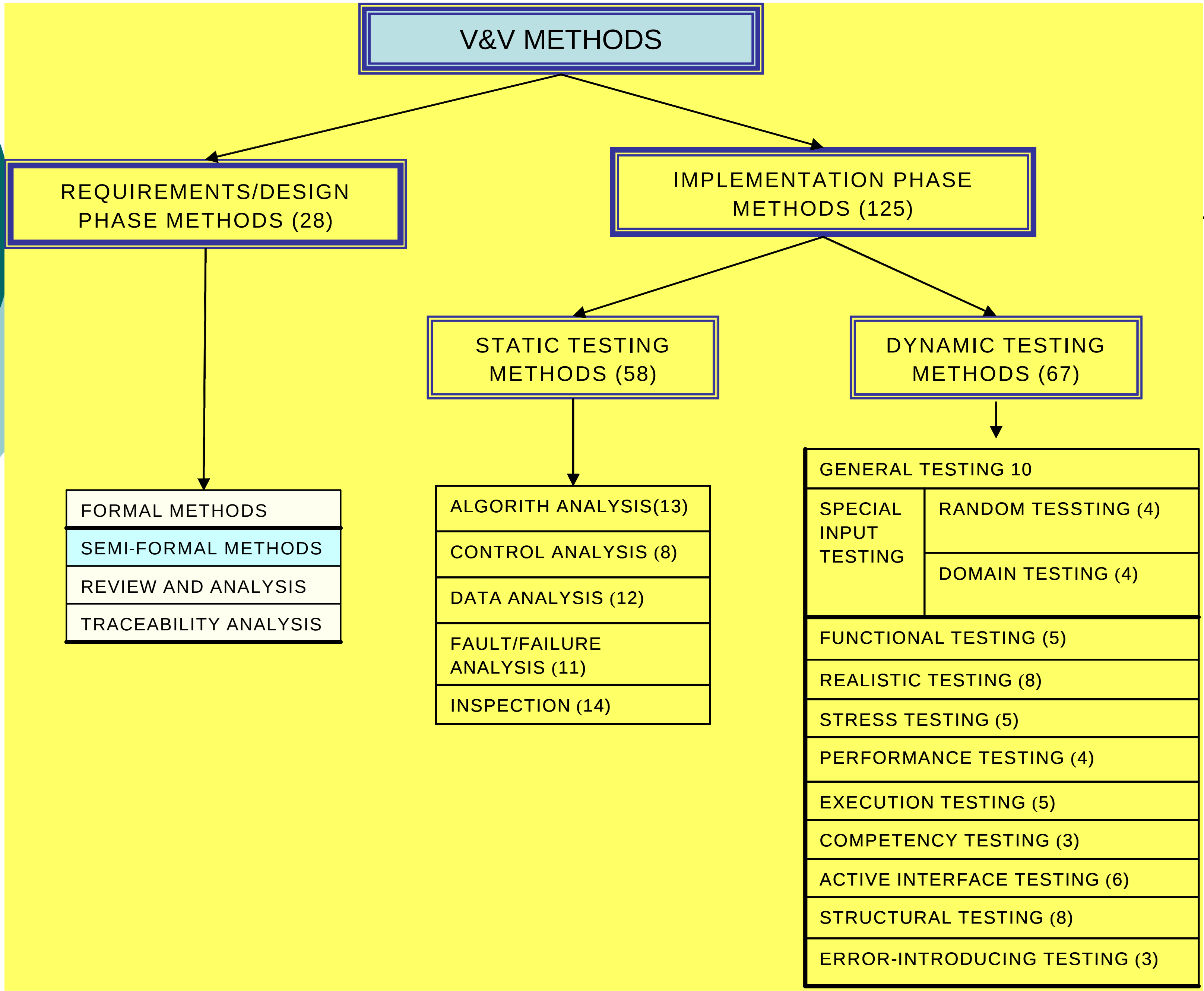
  - Misunderstanding / Misinterpretation of requirements.

  - Incomplete requirements: customer usually can not describe exactly what the software is supposed to do.

  - Software requirements written in natural language by the customer may be ambiguous, inconsistent and/or incomplete.

Requirements Specification

- In general, there are two types of specification relevant to software system development.
- The first is the statement of the user's view, in documents referred to as a requirement specification. These documents must be clearly validated by users since only they know what they want.
- The second specification is drawn up from the software developer's view, and as such it is a technical document, which restates the requirements in a form meaningful to software developers.


V&V METHODS
REQUIREMENTS/DESIGN PHASE METHODS (28)
IMPLEMENTATION PHASE METHODS (125)
STATIC TESTING METHODS (58)
DYNAMIC TESTING METHODS (67)
FORMAL METHODS
SEMI-FORMAL METHODS
REVIEW AND ANALYSIS
TRACEABILITY ANALYSIS
ALGORITH ANALYSIS(13)
CONTROL ANALYSIS (8)
DATA ANALYSIS (12)
FAULT/FAILURE ANALYSIS (11)
INSPECTION (14)
GENERAL TESTING 10
SPECIAL INPUT TESTING
RANDOM TESSTING (4)
DOMAIN TESTING (4)
FUNCTIONAL TESTING (5)
REALISTIC TESTING (8)
STRESS TESTING (5)
PERFORMANCE TESTING (4)
EXECUTION TESTING (5)
COMPETENCY TESTING (3)
ACTIVE INTERFACE TESTING (6)
STRUCTURAL TESTING (8)
ERROR-INTRODUCING TESTING (3)

Requirement Analysis Techniques

- **Formal methods** mathematical verification of requirements (8):

  Based on translation of requirements into mathematical form .

- **Semi-formal methods** requirement language analysis (11):

  Based on an expression of requirement specifications in a special requirement language.

Requirement Analysis Techniques

- Informal method reviews and analysis (7):
  - They are based on review of the requirement specifications according to a pre-established set of criteria and a detailed checklist and procedures by specialized person.
- Requirement tracability(2):
  - They are based on matching of unique requirement elements to design elements and then to the elements of implementation

Problems

<u>Formalizing the requirements</u> (in total or in part) presents a new viewpoint

- But formalization itself cannot guarantee to detect system error, nor can it prove that the software requirement specification is correct
- Mathematical verification of requirements does not seem to greatly simplify development.

What is meant by “CORRECT”

- Program matches the specification.
- However the specification itself may not be correct!

- Correctness is concerned with whether the software meets user or system requirements.

Graphical Requirement Analysis GRA

- GRA is a modeling technique for complex embedded systems specifications
- It is designed from core concept of
  - Functional modeling
  - Object-oriented design
  - Hierarchical model
  - Cosmic Functional Size Measurement
  - Success-failure paradigm

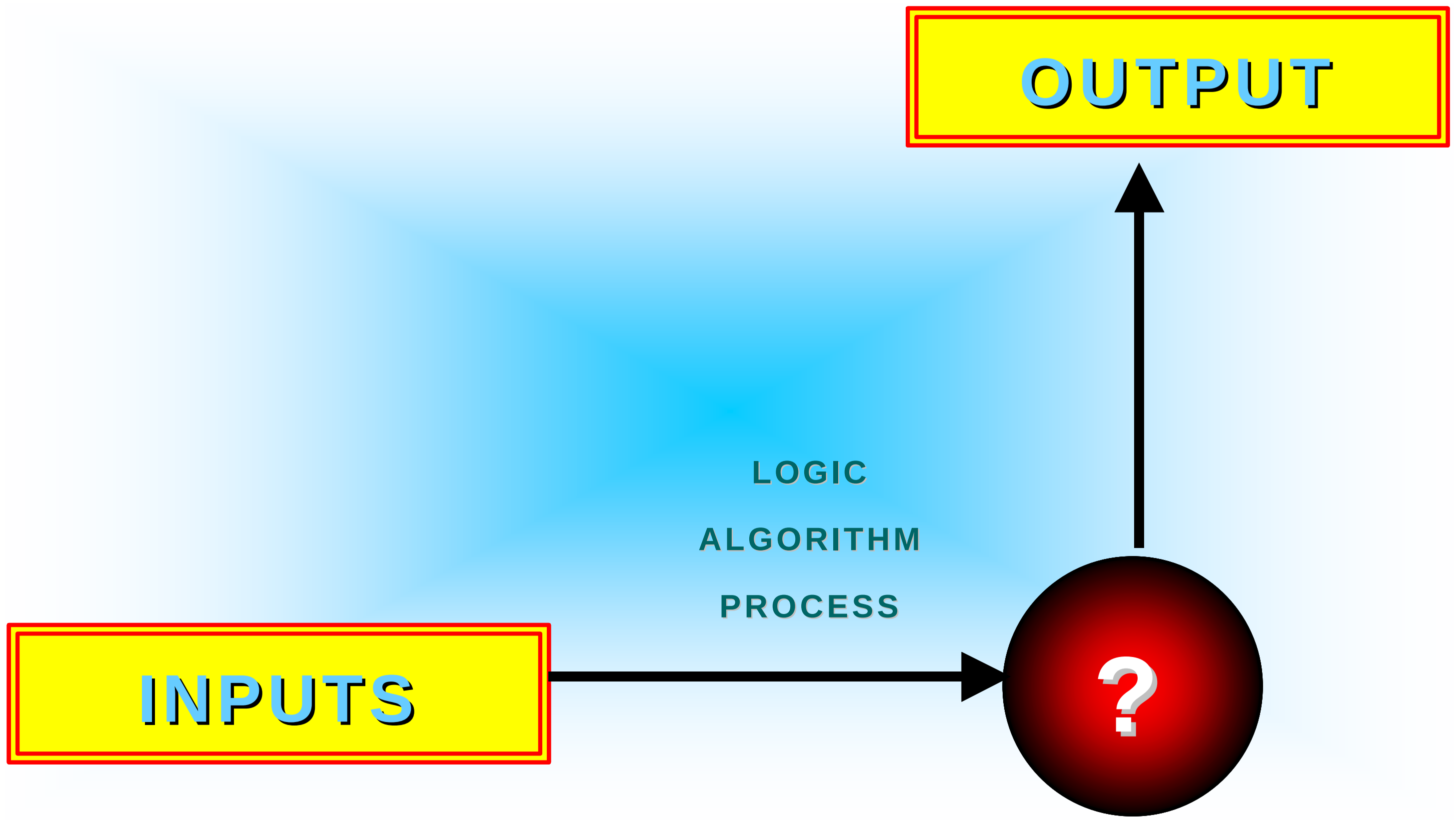
Basic Characteristics of a Function

- A software module that performs a specific action is invoked by the appearance of its name in an expression, may receive input value, and return a single value.

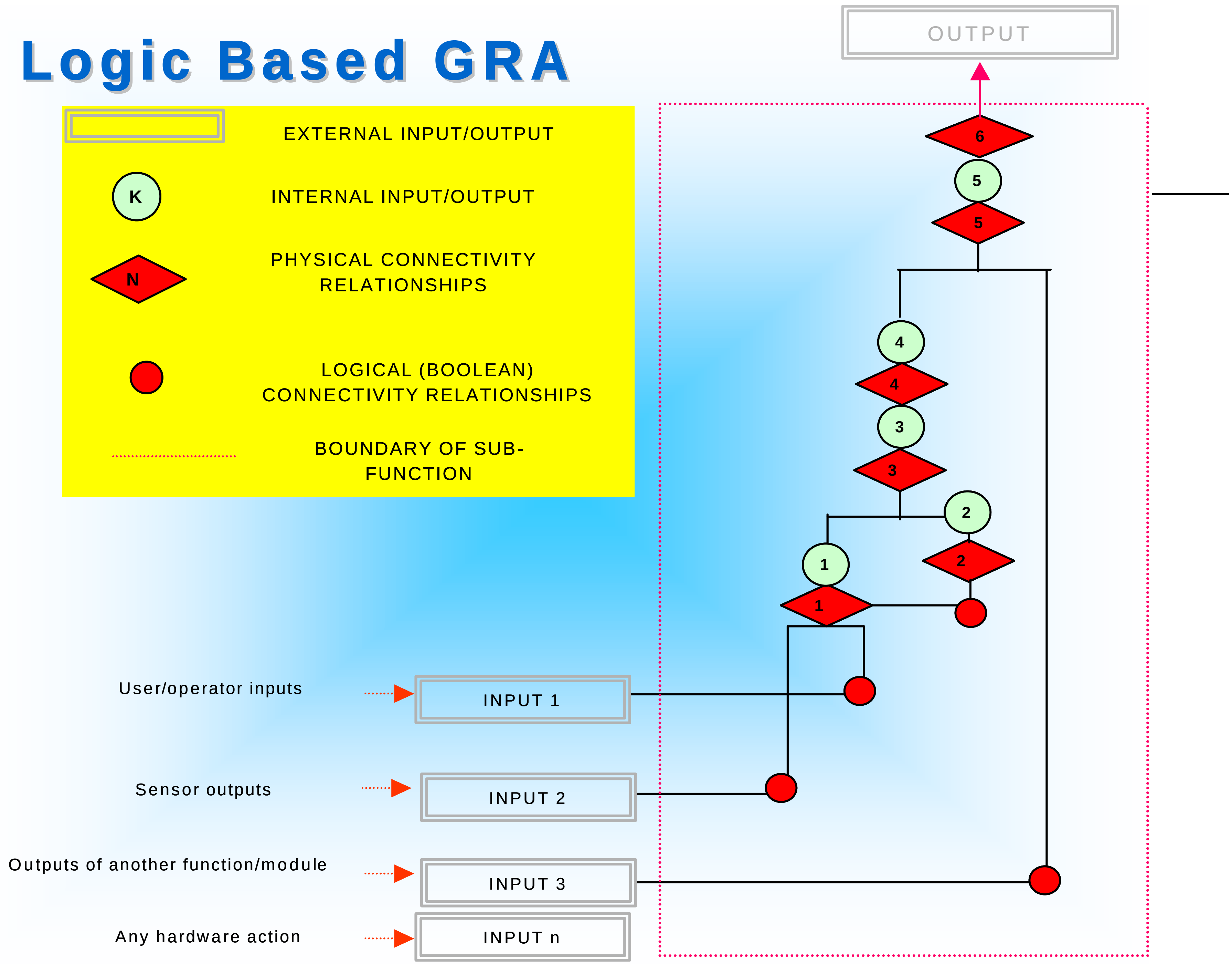
- When a function is decomposed, sub-functions can be identified".

Source: [IEEE610.12]

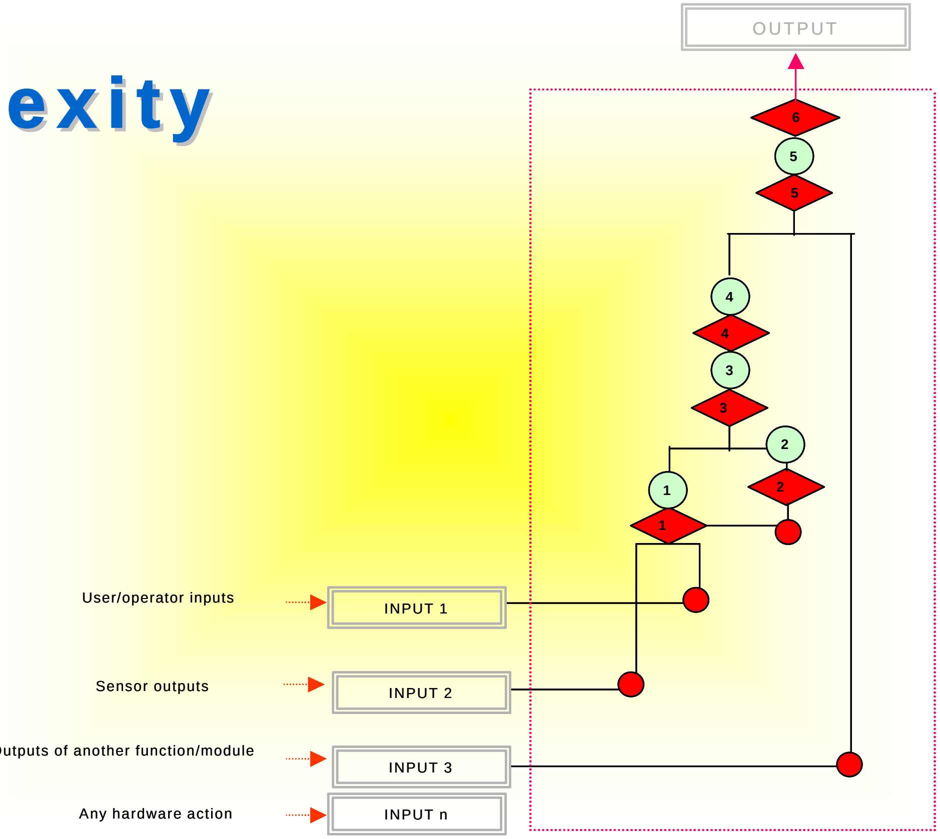
A Function

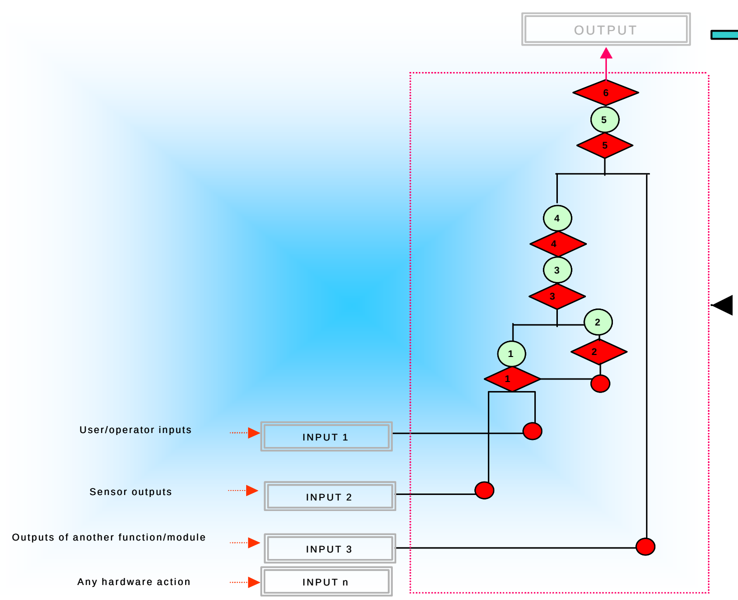


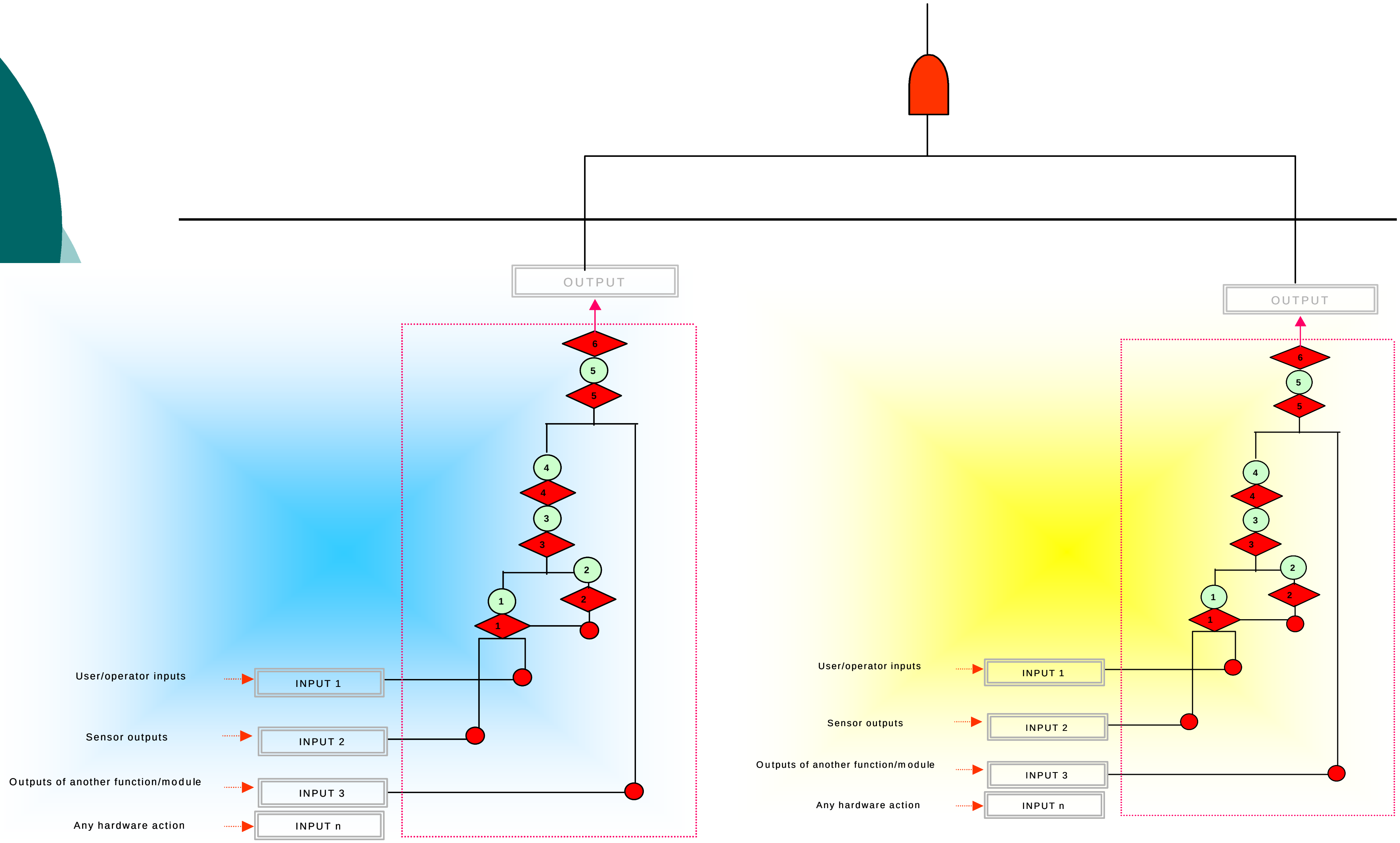
Logic Based GRA



Dealing with Complexity



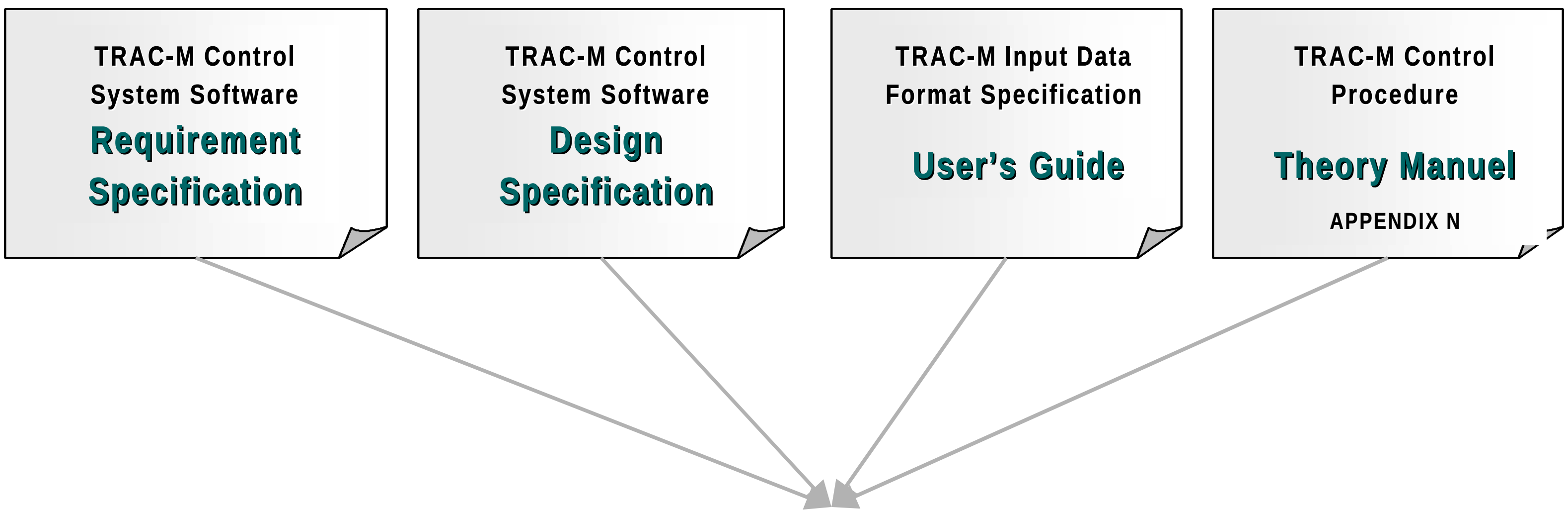



OUTPUT
6
5
5
4
4
3
3
2
2
1
1
User/operator inputs
INPUT 1
Sensor outputs
INPUT 2
Outputs of another function/module
INPUT 3
Any hardware action
INPUT n
OUTPUT
6
5
5
4
4
3
3
2
2
1
1
User/operator inputs
INPUT 1
Sensor outputs
INPUT 2
Outputs of another function/module
INPUT 3
Any hardware action
INPUT n

A Case Study 1:

Control System Requirement

Collection and Identification of FURs



FURs-Functional User Requirements-

Formal Review and Inspection

Software Functional User Requirements

CNSYS 4.1
Vessel Water Level Control System

CNSYS 4.2
Core Flow Controller

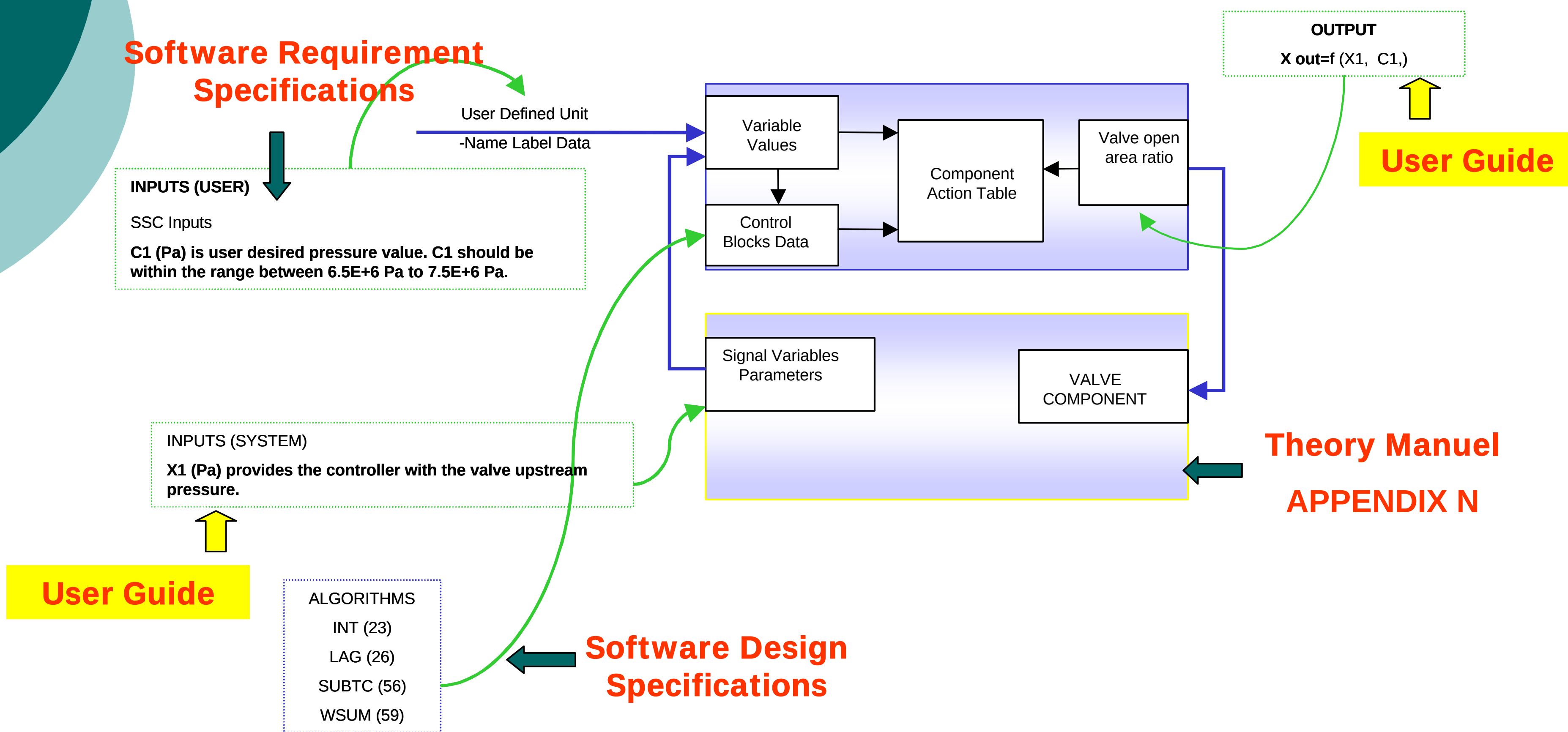
CNYSYS 4.3
Pressure Controller

CNSYS 4.1 Water Level Control System	**Function:** The level controller will be provided with the dynamic level position inside a given component along with the current feed-water line and steam line mass flow rate.	INPUTS (6inputs)	1. Water Level set point (m) 2. Initial feed-water flow rate (kg/s) 3. ID of the Component (and the zone ID if it is vessel) in which the collapsed water level will be calculated. 4. ID of the component in which the feed-waterline mass flow rate will be detected and the location ID 5. ID of the Component in which the steam line mass flow rate will be detected and the Location ID 6. ID of the **FILL**, which provides the feed-water line mass flow rate.	**Output:** **Mass flow rate** of the FILL component, which provides the inlet flow of the feed-water system
CNSYS 4.2 Flow Controller System	**Function**: It detects the flow rate from **CHAN** or **JETP** component and calculates the appropriate pump motor torque.	INPUTS (4 inputs)	1. Mass flow set point (kg/s) 2. Initial rated re-circulation pump torque 3. ID number of the component in which the flow rate will be detected. In addition, the user is also required to identify the flow detection location within the component. 4. ID number of the **PUMP** component in which the motor torque is to be controlled.	**Output:** Re-circulation pump motor **Torque**
CNSYS 4.3 Pressure Controller System	**Function:** It detects the main steam line pressure and adjust the main stem line valve area to achieve the pressure set point	INPUT (4	1. Pressure Set point 2. Time zero valve area fraction 3. ID number of the component in which the steam line pressure is to be detected and the cell number. 4. ID number of the **VALVE** component in which the valve area is to be controlled	**Output:** Main steam line **valve area**

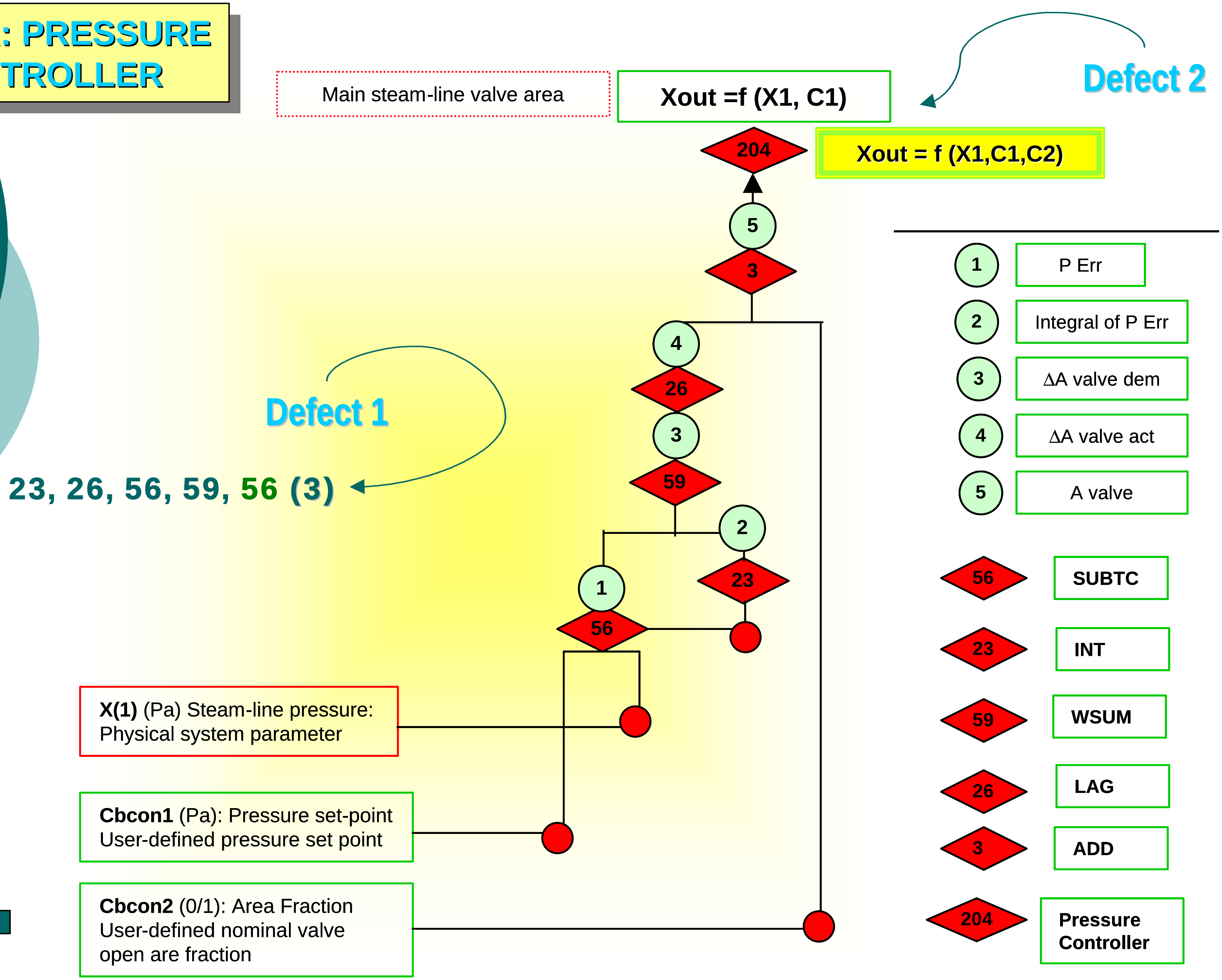
Formal Review and Inspection Software Design Specification

CNSYS 4.1 Vessel Water Level Control System / CNSYS 4.2 Core Flow Controller / CNYSYS 4.3 Pressure Controller	
	3. 4.1 Built in Control System Data Structure
CNSYS 4.1 Vessel Water Level Control System	**3.4.2 Level Controller – Control Block Type #(8);** **23, 26, 56, 59, 11, 54, 59, 56**
CNSYS 4.2 Core Flow Controller	**3. 4.4 Flow Controller- Control Block Type # (5):** **23,26,56,59,56**
CNYSYS 4.3 Pressure Controller	**3. 4.3 Pressure Controller—5 Control Block Type#(5) :** **23, 26, 56, 59, 56**

Pressure Control Procedure



GRA: PRESSURE CONTROLLER



Lesson Learned: Limited Assurance using Formal Review Methods

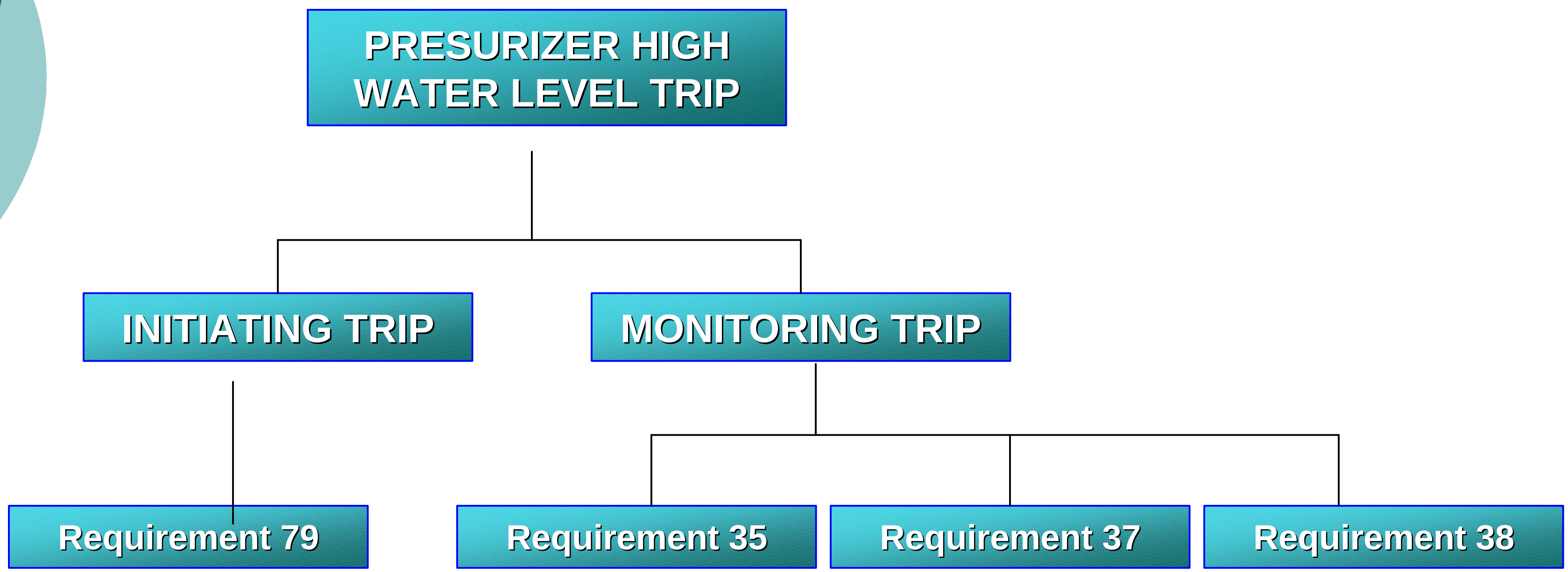
REVIEW & INSPECTION “Criteria”	Measurement “Rules and Procedure”	GRA Analysis Framework
Inputs	External Input	
Output	External Output	
Algorithms-	Sub-Process	Logic
-	Read	Internal Input
-	Write	Internal Output
Interfaces	Software Boundary	
Time	Triggering	Data Flow Sequence t

A Case Study (2)

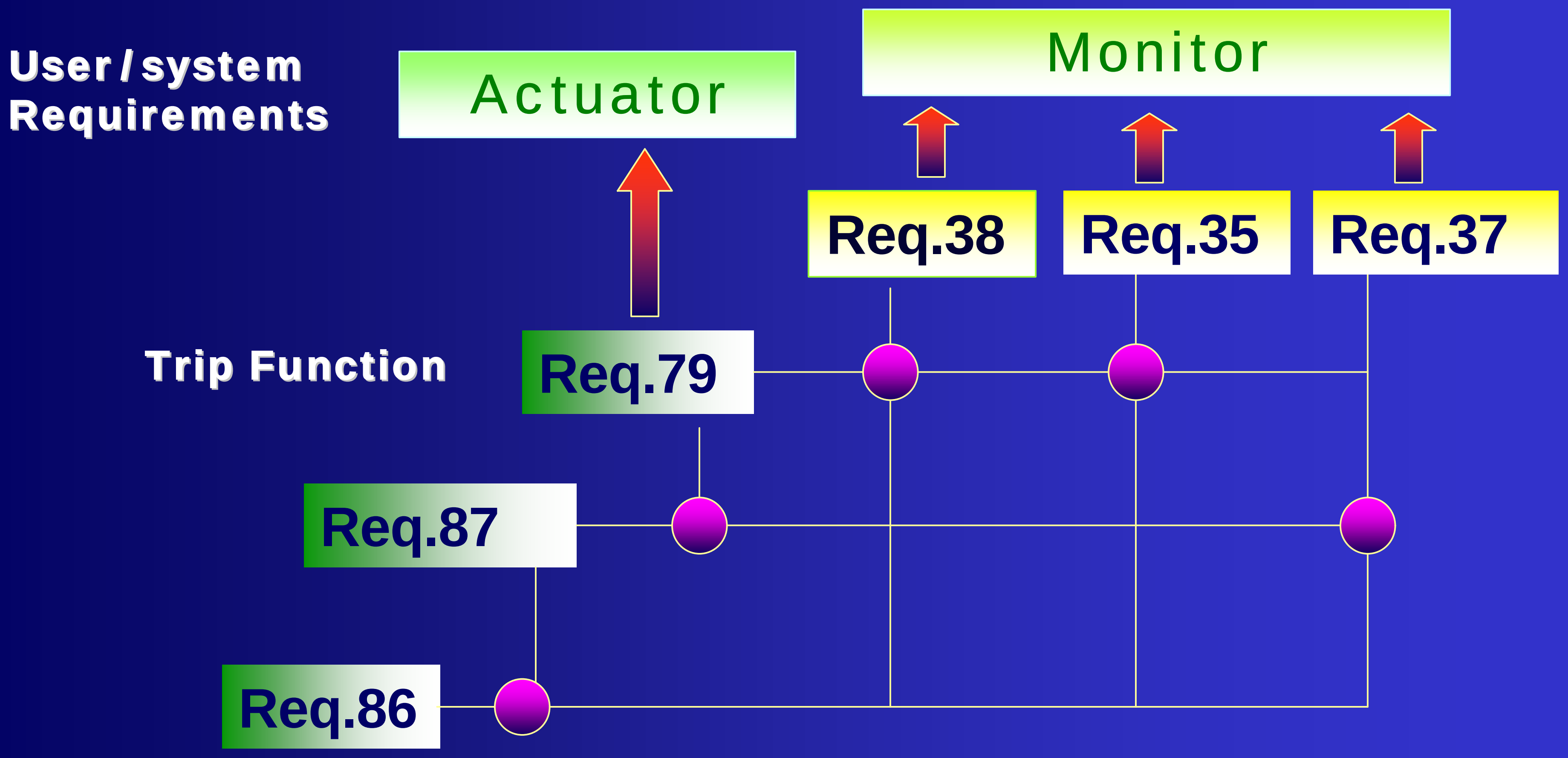
Integration of System/Software Specifications

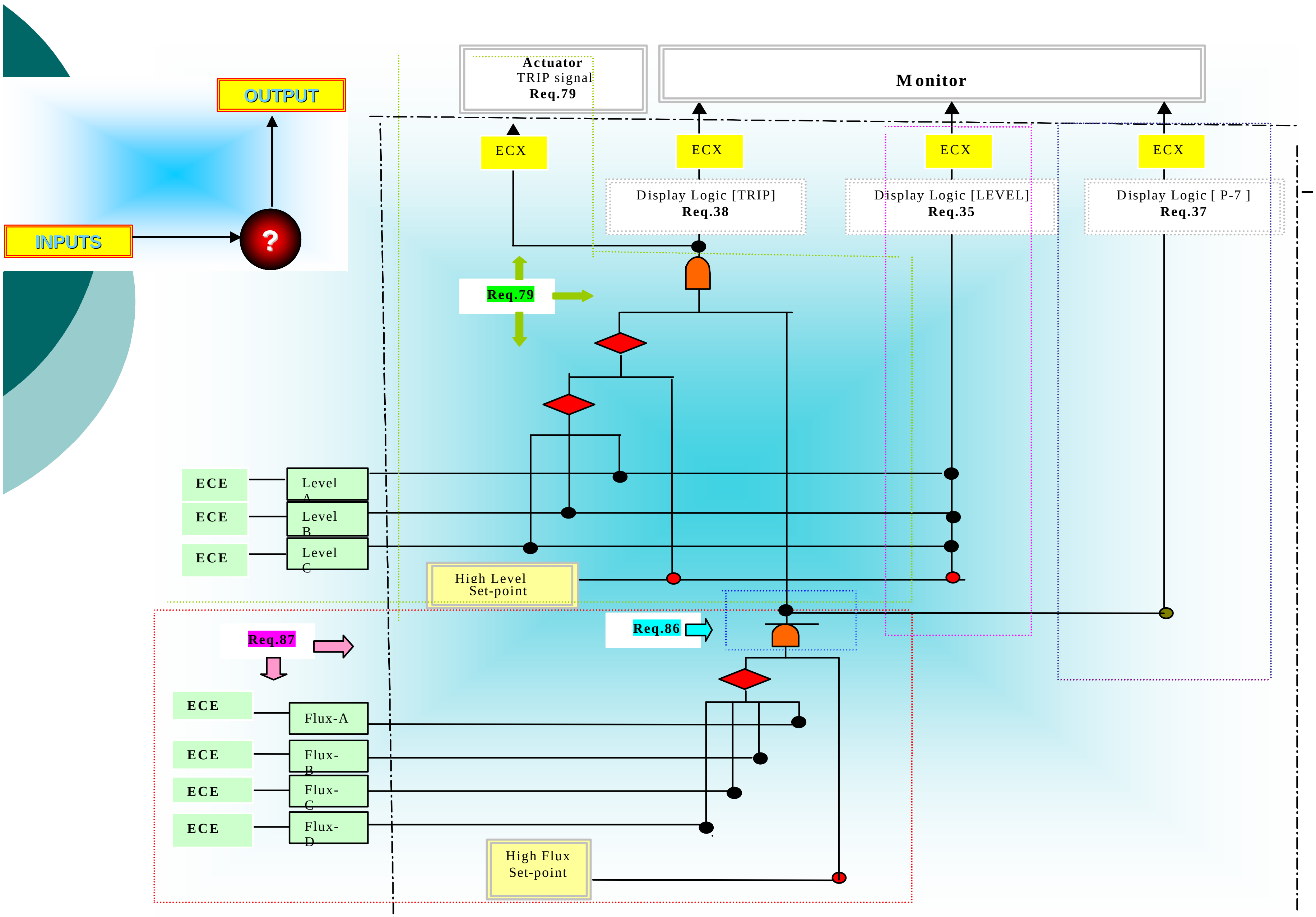
Generic Westinghouse Reactor Protection System Requirements

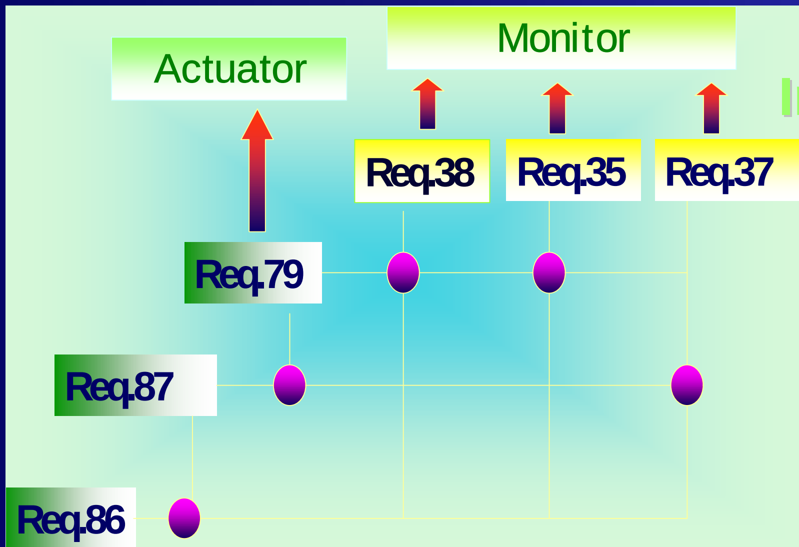
High Water Level Trip



INPUT/OUTPUT RELATION BETWEEN REQUIREMENTS

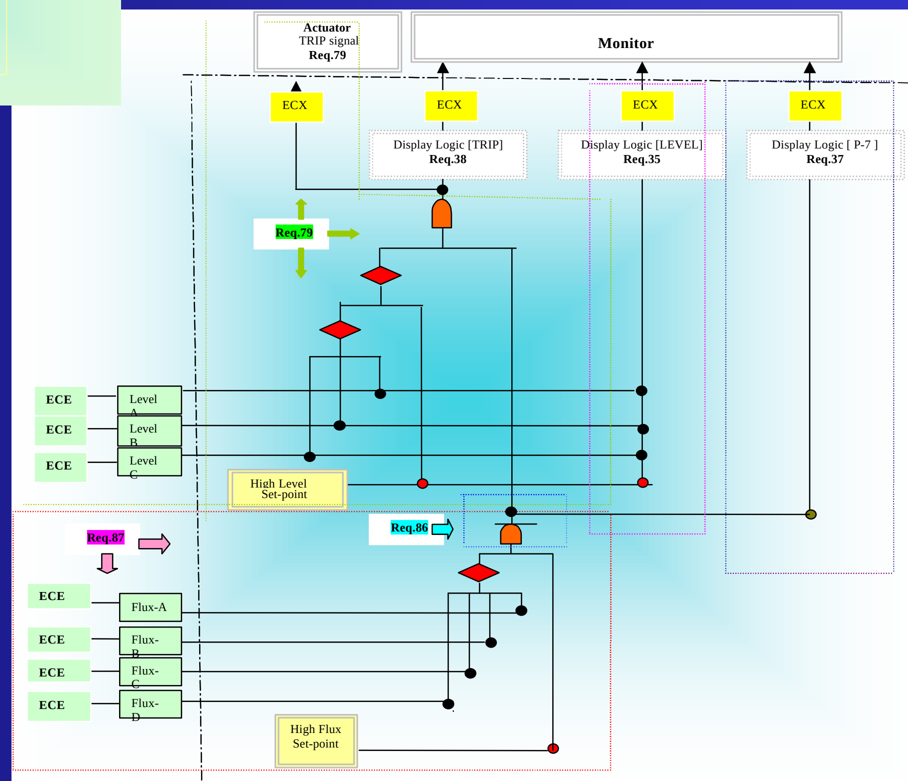



OUTPUT
INPUTS
?
Actuator
TRIP signal
Req.79
Monitor
ECX
ECX
ECX
ECX
Display Logic [TRIP]
Req.38
Display Logic [LEVEL]
Req.35
Display Logic [P-7]
Req.37
Req.79
ECE
ECE
ECE
Level A
Level B
Level C
High Level Set-point
Req.86
Req.87
ECE
ECE
ECE
ECE
Flux-A
Flux-B
Flux-C
Flux-D
High Flux Set-point



Integrated System/Software Specifications

Integration of Level 1 and Level 2



Conclusion

- **The model provides**
  - efficient and accurate way to specify functional requirements.
  - a mapping from inputs to outputs into a multi-level detailed system and software functionality.
  - a means by which to verify clarity and its presence/absence of functionality.

Conclusion

- **The model helps to**
  - **identify the interconnections between modules, functional blocks, functions and sub-functions**
  - **identify the sub-processes and boundary/layer as well as inputs/outputs/reads/writes for the measuring functional size of a software module.**

Conclusion

- The model can be used as a measure of
  - Completeness
  - Consistency
  - Ambiguity
  - Traceability

Conclusion

- **The model can be used for**
  - **defining and modeling embedded system requirements**
  - **verifying functional correctness of embedded systems**

Future Work

- To design smart test cases using GRA functional framework such as
  - Scenario based test cases,
  - Simulating functional specifications based on the Probabilistic Risk Assessment (PRA) report of a critical system.