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
FernUniversität, 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)

- **FORMAL METHODS**
  - Semi-Formal Methods
  - Review and Analysis
  - Traceability Analysis

- **ALGORITHM ANALYSIS** (13)
  - Control Analysis (8)
  - Data Analysis (12)
  - Fault/Failure Analysis (11)
  - Inspection (14)

## Implementation Phase Methods (125)

- **STATIC TESTING METHODS** (58)
  - Structural Testing (8)
  - Active Interface Testing (6)
  - Competency Testing (3)
  - Execution Testing (5)
  - Performance Testing (4)
  - Stress Testing (5)
  - Realistic Testing (8)

- **DYNAMIC TESTING METHODS** (67)
  - Functional Testing (5)
  - Domain Testing (4)
  - Special Input Testing
  - Random Testing (4)
  - General Testing (10)

- **ERROR-INTRODUCING TESTING** (3)
  - Domain Testing (4)
  - Special Input Testing
  - Random Testing (4)
  - General Testing (10)

- **FUNCTIONAL TESTING** (5)
  - Domain Testing (4)
  - Special Input Testing
  - Random Testing (4)
  - General Testing (10)

- **STRESS TESTING** (5)
  - Domain Testing (4)
  - Special Input Testing
  - Random Testing (4)
  - General Testing (10)

- **PERFORMANCE TESTING** (4)
  - Special Input Testing
  - Random Testing (4)
  - General Testing (10)

- **EXECUTION TESTING** (5)
  - Special Input Testing
  - Random Testing (4)
  - General Testing (10)

- **COMPETENCY TESTING** (3)
  - Special Input Testing
  - Random Testing (4)
  - General Testing (10)

- **ACTIVE INTERFACE TESTING** (6)
  - Special Input Testing
  - Random Testing (4)
  - General Testing (10)

- **STRUCTURAL TESTING** (8)
  - Special Input Testing
  - Random Testing (4)
  - General Testing (10)

- **ERROR-INTRODUCING TESTING** (3)
  - Special Input Testing
  - Random Testing (4)
  - General Testing (10)
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

Formalizing the requirements (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

INPUTS

LOGIC
ALGORITHM
PROCESS

OUTPUT

?
Logic Based GRA

EXTERNAL INPUT/OUTPUT

K

INTERNAL INPUT/OUTPUT

N

PHYSICAL CONNECTIVITY RELATIONSHIPS

LOGICAL (BOOLEAN) CONNECTIVITY RELATIONSHIPS

BOUNDARY OF SUB-FUNCTION

User/operator inputs

Sensor outputs

Outputs of another function/module

Any hardware action

INPUT 1

INPUT 2

INPUT 3

INPUT n

OUTPUT
Dealing with Complexity
A Case Study 1: Control System Requirement

Collection and Identification of FURs

- TRAC-M Control System Software Requirement Specification
- TRAC-M Control System Software Design Specification
- TRAC-M Control Procedure Theory Manual APPENDIX N

FURs - Functional User Requirements -
### CNSYS 4.1 Vessel 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:**
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 a 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:**
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 steam line valve area to achieve the pressure set point.

**Inputs:**
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
### 3. 4.1 Built in Control System Data Structure

#### 3.4.2 Level Controller - Control Block Type #(8);

23, 26, 56, 59, 11, 54, 59, 56

#### 3.4.3 Pressure Controller—5 Control Block Type#(5) :

23, 26, 56, 59, 56

#### 3.4.4 Flow Controller- Control Block Type # (5):

23, 26, 56, 59, 56

---

**CNSYS 4.1**

Vessel Water Level Control System

**CNSYS 4.2**

Core Flow Controller

**CNYSYS 4.3**

Pressure Controller
Pressure Control Procedure

Software Requirement Specifications

INPUTS (USER)
SSC Inputs
C1 (Pa) is user desired pressure value. C1 should be within the range between 6.5E+6 Pa to 7.5E+6 Pa.

INPUTS (SYSTEM)
X1 (Pa) provides the controller with the valve upstream pressure.

ALGORITHMS
INT (23)
LAG (26)
SUBTC (56)
WSUM (59)

OUTPUT
X out=f (X1, C1,)

User Defined Unit
-Name Label Data

Variable Values
Component Action Table
Valve open area ratio

Control Blocks Data

Signal Variables Parameters

VALVE COMPONENT

User Guide

Theory Manuel
APPENDIX N

Software Design Specifications

User Guide
Lesson Learned: Limited Assurance using Formal Review Methods

<table>
<thead>
<tr>
<th>REVIEW &amp; INSPECTION “Criteria”</th>
<th>Measurement “Rules and Procedure”</th>
<th>GRA Analysis Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>External Input</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>External Output</td>
<td></td>
</tr>
<tr>
<td>Algorithms-</td>
<td>Sub-Process</td>
<td>Logic</td>
</tr>
<tr>
<td>-</td>
<td>Read</td>
<td>Internal Input</td>
</tr>
<tr>
<td>-</td>
<td>Write</td>
<td>Internal Output</td>
</tr>
<tr>
<td>Interfaces</td>
<td>Software Boundary</td>
<td>Data Flow Sequence</td>
</tr>
<tr>
<td>Time</td>
<td>Triggering</td>
<td></td>
</tr>
</tbody>
</table>
A Case Study (2)
Integration of System/Software Specifications

Generic Westinghouse Reactor Protection System Requirements
High Water Level Trip

- Presurizer High Water Level Trip
  - Initiating Trip
    - Requirement 79
  - Monitoring Trip
    - Requirement 35
    - Requirement 37
    - Requirement 38
INPUT/OUTPUT RELATION BETWEEN REQUIREMENTS

User/system Requirements

Actuator

Monitor

Trip Function

Req.79

Req.38

Req.35

Req.37

Req.87

Req.86
IFIP World Congress 2002
Stream 7: Distributed and Parallel Embedded Systems
Integrated System/Software Specifications

Integration of Level 1 and Level 2

Actuator

Monitor

Req.38
Req.35
Req.37

Req.79

Req.87

Req.86

High Level
Set-point

Display Logic [TRIP]

Req.38

Actuator

Monitor

Display Logic [LEVEL]

Req.35

Display Logic [P-7]

Req.37

Integration of Level 1 Integration of Level 1

and Level 2 and Level 2

Integrated System/Software Specifications

©

28
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.