Software Quality Engineering

Week 2

Engr. Muhammad Umer Haroon

Inspecting SRS using Fault Checklist Method

Inspection of Software Requirements Document

Outline

- Fault Checklist Technique
- Checklist Method
  - General Faults
  - Omission Faults
  - Commission Faults
  - Other Faults
- Fault Form
  - Fields
  - Example

Fault Checklist Technique

Question: How does one detect fault?

Answer:

1. By reading the document
2. By understanding what the document describes
3. By answering the questions in the fault checklist

Checklist Method

- General Faults
- Omission Faults
- Commission Faults
- Other Faults

Checklist Method:
General Faults

- Are the goals of system defined?
- Are the requirements clear and unambiguous?
- Is a functional overview of system provided?
- Is an overview of operational modes provided?
- If assumptions that affect implementation have been made, are they stated?
- Have the requirements been stated in the terms of inputs, outputs, and processing for each function?
- Are all functions, devices, constraints traced to requirements and vice versa?
- Are the required attributes, assumptions and constraints of the system completely listed?

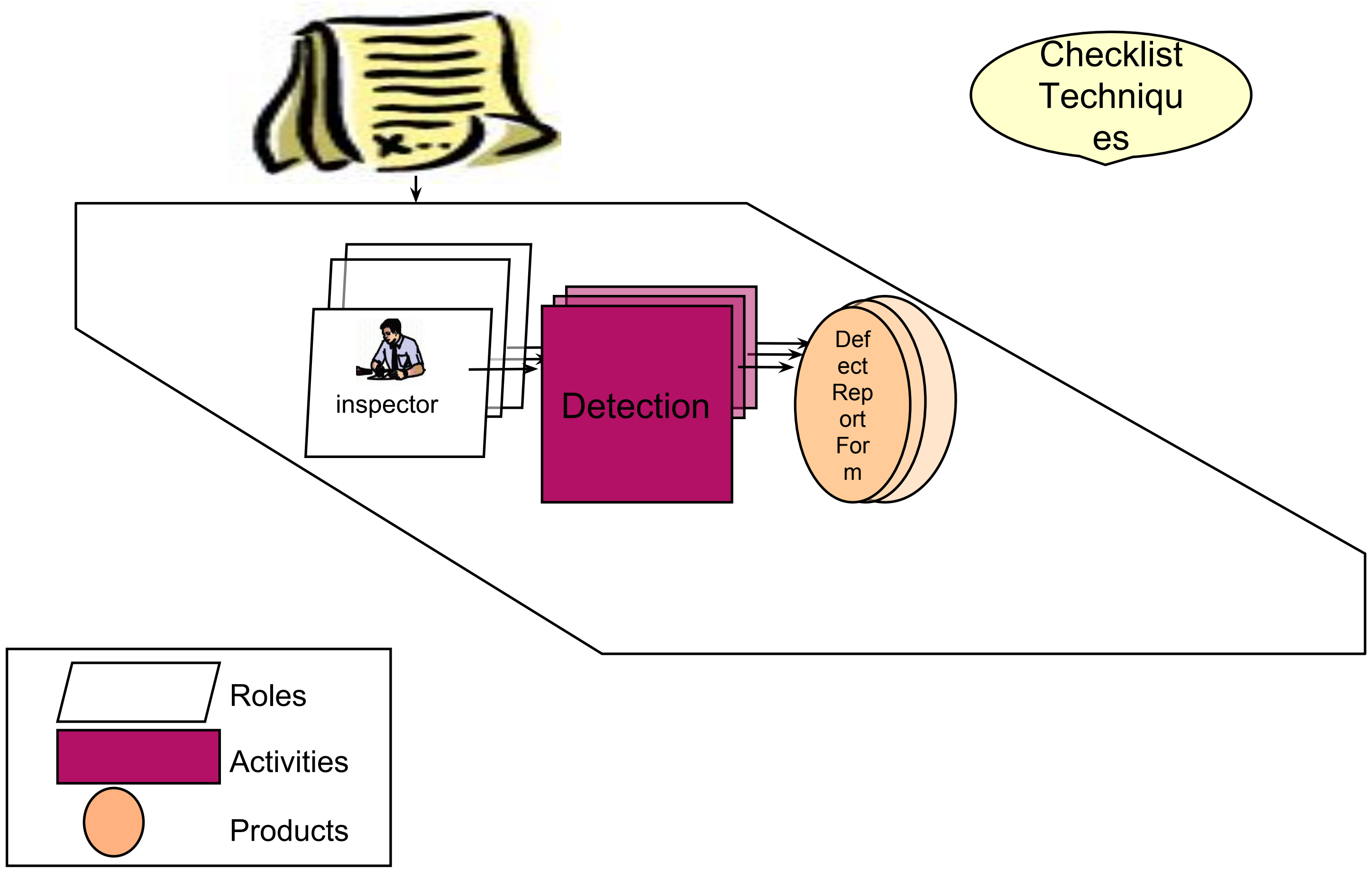
Checklist Method: Omission Faults

- Missing Functionality
  - Are the desired functions sufficient to meet the system objectives?
  - Are all inputs to a function sufficient to perform the required function?
  - Are undesired events considered and their required responses specified?
  - Are the initial and special states considered (e.g., system initiation, abnormal termination)?
- Missing Performance
  - Can the system be tested, demonstrated, analyzed or inspected to show that it satisfies the requirements?
- Missing Interface
  - Are the inputs and outputs for all interfaces sufficient?
  - Are the interface requirements between hardware, software, personnel and procedures included?
- Missing Environment
  - Have the functionality of hardware or software interacting with the system been properly specified?

Checklist Method:
Commission Faults

- **Ambiguous Information**
  - Are the individual requirements stated so that they are discrete, unambiguous, and testable?
  - Are all mode transitions specified deterministically?
- **Inconsistent Information**
  - Are the requirements mutually consistent?
  - Are the functional requirements consistent with the overview?
  - Are the functional requirements consistent with the actual operating system?
- **Inconsistent and Extra Functionality**
  - Are all desired functions necessary to meet the system objectives?
  - Are all inputs to a function necessary to perform the required function?
  - Are the inputs and outputs for all interfaces necessary?
  - Are all the outputs produced by a function used by another function or transferred across an external interface?
- **Wrong Section**
  - Are all the requirements, interfaces, constraints, etc. listed in the appropriate sections?
- **Other Faults**
  - *If you find additional faults, not related to specific questions on the checklist, that do not fall in any of the existing categories, classify it as Other (O).*

Study Run


Checklist Techniques
inspector
Detection
Defect Report Form
Roles
Activities
Products

Fault Form

Name: Start time:

Fault #	Page #	Requirement #	Fault Class	Description	Time found	Importance level	Probability of causing failure	Break

End time:

Fault Form:
Fields

- Fault #- serial identification number (e.g., 1,2,3 etc)

- Page #- maps to the page number in a SRS document where that fault is present (e.g., 3,5,6 etc)

- Requirement #- maps to a particular requirement number where a fault is found (e.g., FR2.1, FR3 etc)

Fault Form: Fields

- Fault class- describes the classification of a fault. A fault is classified in following classes using fault checklist
  - General (G)
  - Missing Functionality (MF)
  - Missing Performance (MP)
  - Missing Interface (MI)
  - Missing Environment (ME)
  - Ambiguous Information (AI)
  - Inconsistent Information (II)
  - Incorrect or Extra Functionality (EF)
  - Wrong Section (WS)
  - Other (O)
- Description- provides a brief but clear description of the fault in the requirements document. This description should be clear enough for a developer to understand and fix it without having to talk to you
- Time found- it is the time when a particular fault was found

Fault Form: Fields

- Importance level- this is the scale of importance of a particular requirement fault found during inspection and has to classified as per following scale:
  - 0: not important, designer should easily see the problem
  - 1: problem, if a failure occurs it should be easy to find and fix (e.g. change to 1 module)
  - 2: important, if a failure occurs, it could be hard to find and fix (e.g. change to few modules)
  - 3: very important, if a failure occurs, it could be very hard to find and fix (e.g., change to several modules and their dependencies)
  - 4: if a failure occurs, it could cause a redesign

Fault Form: Fields

- Probability of causing failure- describes the probability scale that a particular fault can cause system failure using following scale:
  - 0: will not cause fault of failure, regardless whether it is caught by the designer
  - 1: will not cause fault or failure, because it will be caught by designer
  - 2: could cause a failure, but will most likely be caught by designer
  - 3: would cause a failure, will most likely not be caught by designer
- Break: describes the time breaks during the inspection

Fault Form:
Example

Fault #	Page #	Requirement #	Fault Class	Description	Time found	Importance level	Probability of causing failure	Break
1	3	FR2	AI	…………… ………………… ………………… …	9:30 AM	3	2	
2	5	FR3.5.6	IF	…………… …………… ……………	10:00 AM	1	2	Break: 10 AM
3	12	FR 5.2	MI	…………… …………… ……………	1 PM	2	1	Resume 12 PM
4	14	FR 5.3.2	MP	…………… …………… ……………	2 PM	0	0	