
# Chapter Seven

# Ensuring Your Requirements Are correct:

# Requirement validation Techniques

# Chapter Outline

- *Testing Early and Often*
- *Use Case Scenario Testing*

# Requirements Validation

- *Validation denotes checking whether inputs, performed activities, and* ***created outputs*** *(requirements artifacts) of the requirements engineering core activities* ***fulfill defined quality criteria.***
- *Validation is performed by involving relevant stakeholders, other requirement sources (standards, laws, etc.) as well as external reviewers, if necessary.*

## Quality Criteria

- ***Completeness:*** *The requirement must contain all relevant information (template).*
- ***Consistency:*** *The requirements must be compatible with each other.*
- ***Adequacy:*** *The requirements must address the actual needs of the system.*
- ***Unambiguity:*** *Every requirement must be described in a way that precludes different interpretations.*
- ***Comprehensibility:*** *The requirements must be understandable by the stakeholders*

# *Cntd…*

- ***Importance:*** *Each requirement must indicate how essential it is for the success of the project.*
- ***Measurability:*** *- The requirement must be formulated at a level of precision that enables to evaluate its satisfaction.*
- ***Necessity:*** *The requirements must all contribute to the satisfaction of the project goals.*
- ***Viability:*** *All requirements can be implemented with the available technology, human resources and budget.*
- ***Traceability:*** *The context in which a requirement was created should be easy to retrieve.*
- *In System development* ***"the earlier the error is discovered the cheaper it is correct"***

# Principles of Validation

## The 6 Principles of Validation

1. ***Involving the Right Stakeholders:*** *Ensure that relevant company-internal as well as relevant external stakeholders participate in validation. Pay attention to the reviewers' independence and appoint external, independent stakeholders, if necessary.*

2. ***Defect Detection vs. Defect Correction:*** *Separate defect detection from the correction of the detected defects.*

3. ***Leveraging Multiple Independent Views:*** *Whenever possible, try to obtain independent views that can b e integrated during requirements validation in order to detect defects more reliably.*

4. ***Use of Appropriate Documentation Formats:*** *Consider changing the documentation format of the requirements into a format that matches the validation goal and the preference s of the stakeholders who actually perform the validation.*

# *Cntd…*

*5.**Creation of Development Artifacts during Validation:** If your validation approach generates results, try to support defect detection by creating development artifacts such as architectural artifacts, test artifacts, user manuals, or goals and scenarios during validation.*

*6. **Repeated Validation:** Establish guidelines that clearly determine when or under what conditions an already released requirements artifact has to b e validated again*

# Validation Techniques

- *Validation Techniques*
  - *Inspections*
  - *Desk-Checks*
  - *Walkthroughs*
  - *Prototypes*
- ***Inspection:*** *an organized examination process of the requirements*

**Involved roles:**

- Organizer
- Moderator
- Author
- Inspectors
- Minute-taker

**Benefit:** Detailed checking of the artefacts

**Critical Success Factors:**

- Commitment of the organization
- Size and complexity of the inspected artefacts
- Number and experience of the inspectors

**Effort:** Medium-High

# Cntd…

- *Desk-Checks*
  - *The author of a requirement artifact distributes the artifact to a set of stakeholders.*
  - *The stakeholders check the artifact individually.*
  - *The stakeholders report the identified defects to the author.*
  - *The collected issues are discussed in a group session (optional)*

**Critical Success Factors:**

- Commitment of the participants
- Coverage of all the aspects
- Not recommended for critical artefacts

**Benefit:** Obtain feedback from individual reviewers

**Effort:** Medium

# Cntd…

- ***Walkthrough***

❖ *A walkthrough does not have formally defined procedure and does not require a differentiated role assignment.*

- *Checking early whether an idea is feasible or not.*
- *Obtaining the opinion and suggestions of other people.*
- *Checking the approval of others and reaching agreement.*

**Critical Success Factors:**

- Involving stakeholders from different contexts
- Comprehensible presentation of the artefact

**Benefit:** Validation of ideas and sketches

**Effort:** Medium-Low

# *Prototypes*

- *A prototype allows the stakeholders to try out the requirements for the system and experience*
- *them thereby.*
  - *Develop the prototype (tool support).*
  - *Training of the stakeholders.*
  - *Observation of prototype usage.*
  - *Collect issues*

**Critical Success Factors:**

- Effort
- Level of detail of the prototype
- Quality of the review

**Benefit:**

- Highly effective defect detection
- Proof of feasibility

**Effort:** Very High-High

# What is Early Testing: Test Early, Test Often BUT How?

## What is Early Testing?

- ***Software testing*** *should start early in the Software Development Life Cycle.*
- *This helps to capture and eliminate defects in the early stages of SDLC i.e requirement gathering and design phases.*
- ***An early start*** *to testing helps to reduce the number of defects and ultimately the rework cost in the end.*
- *The various aspects of Early Testing which would help the Managers and Leads while developing or devising the Testing Strategy document in SDLC.*
- *Adoption of Early Test will immensely* ***result in the successful delivery of a Quality Product.***

# Principles of Testing

What is Testing?
Why Testing?
What to Test?
How to Test?

When to start testing in a software release
When should testing start in a project?
When to start testing and when to stop testing?
Why testing should start early in SDLC?
What is early testing in software development?

# *Cntd…*

- *For a given **Software** or **System** or **Product release** in SDLC, there are various well-defined **methodologies or strategies** for most of the following Principles of Testing.*
- *So, the following questions should answer?*
  - *What is Testing?*
  - *Why Testing?*
  - *What to Test?*
  - *How to Test?*
- *For easy understanding of the audience, we have included all the 'grey area' questions under one umbrella called **Early Testing**.*

# Cntd….

## Why Testing Early in SDLC?

**Some events and activities which are a part of testing.**

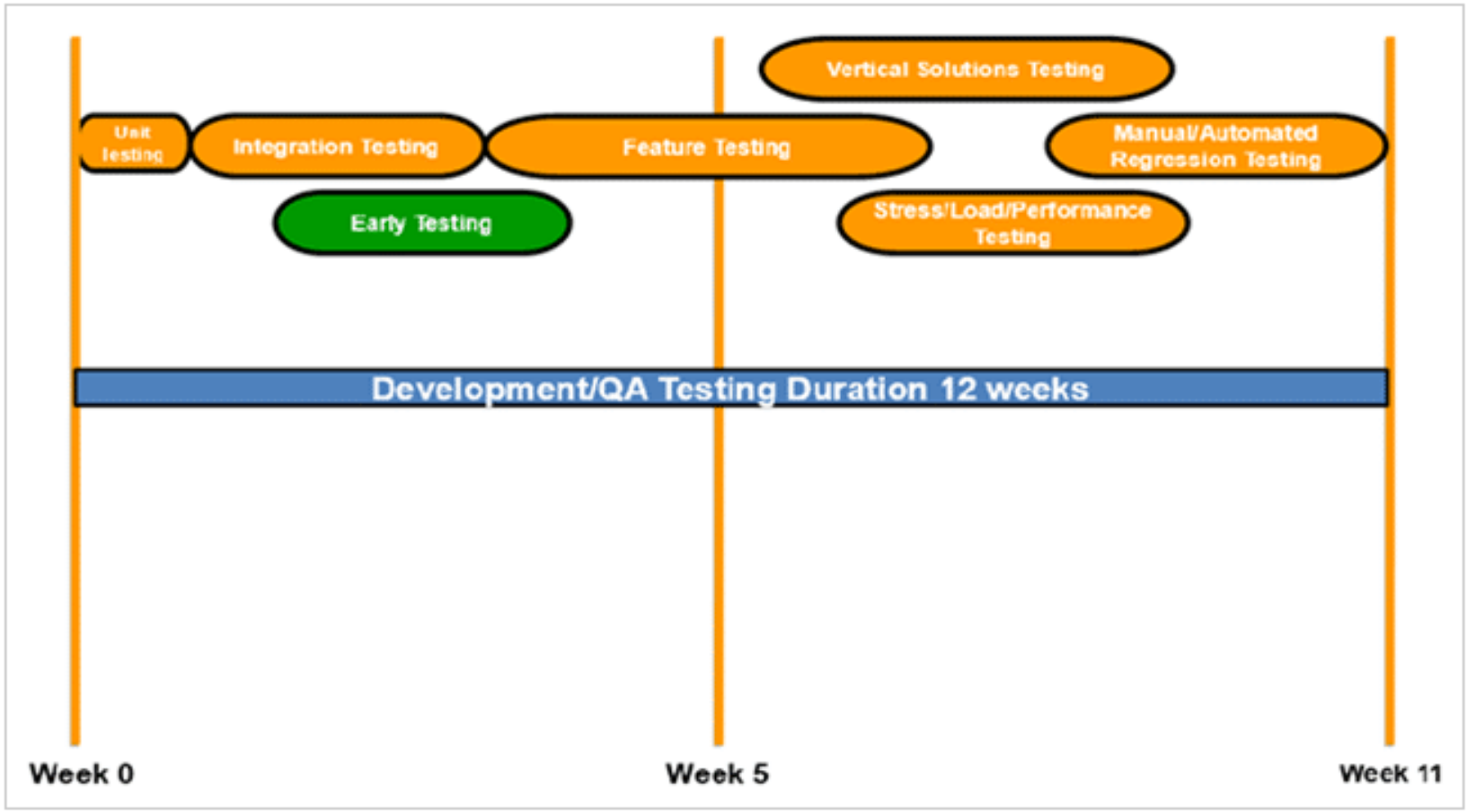
- The Program Management Team assigns a Program Manager (PM) to a given Software Release or a Project.
- The PM in collaboration with all the stakeholders including Marketing, Development, QA (Quality Assurance )and Support teams comes up with a Release Schedule

## Software Release Testing Schedule

- Most of the organizations still follow the **traditional Time Based Release (TBR)** models where the Software or Product releases are planned for quarterly or half-yearly or yearly delivery.
- **Predominantly,** the Waterfall model is used for executing such Software releases.

# Cntd…

**Figure 2 – Typical Quarterly Release Testing Schedule (Not overall Project or Release Schedule)**




*Impact of Critical or High Severity Defects*

# Cntd…

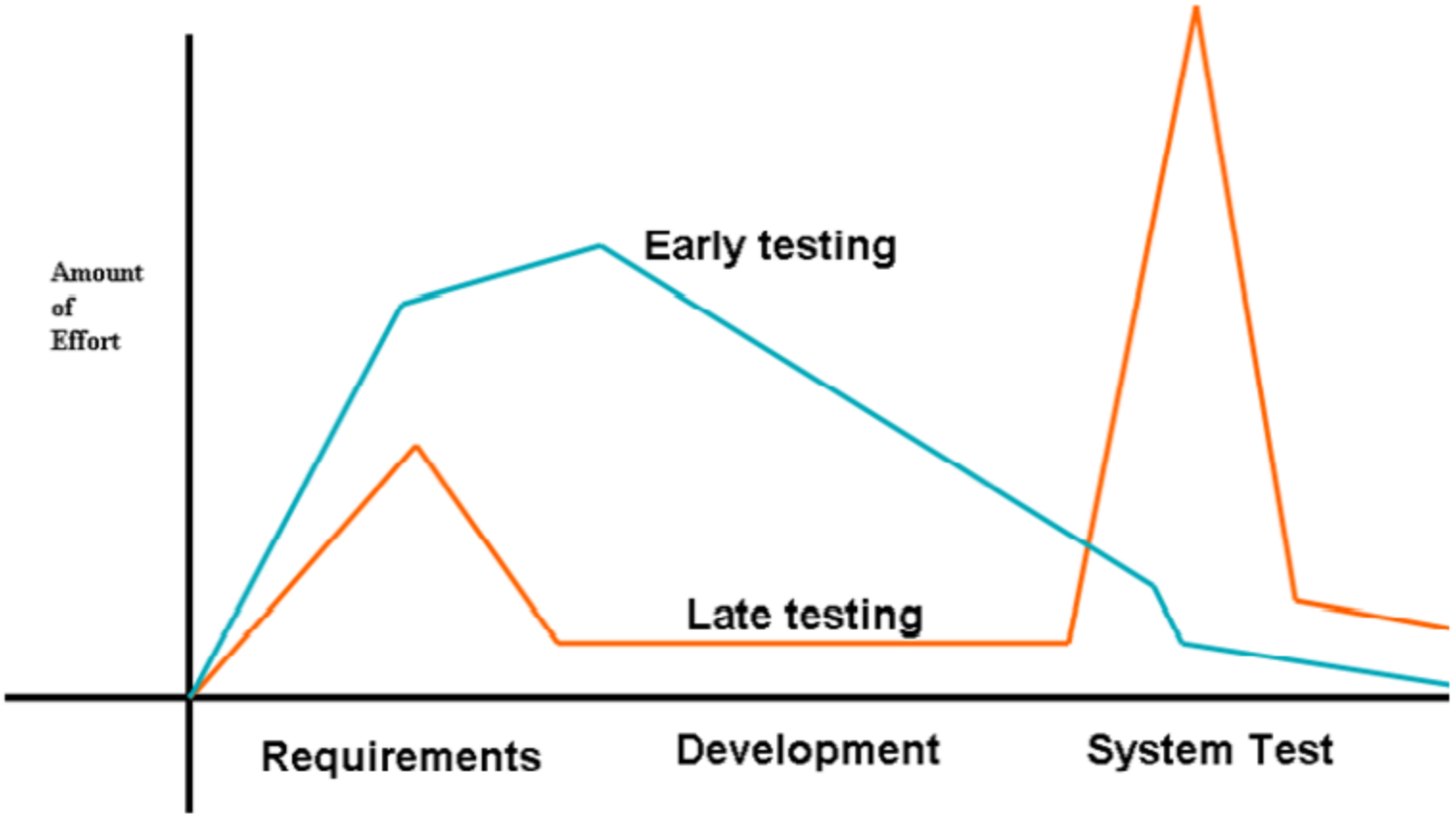



*Figure: testing effect profile*

# *Cntd…*

- ***Mainly, during the course of Testing, it is expected that***
  - ✓ *Critical or high severity defects be identified and logged by Testers.*
  - ✓ *Developers will need to fix those defects.*
  - ✓ *Subsequently, testers will need to verify the fixes.*
- ***Secondly,*** *it is widely acknowledged by many Product and Software Engineering organizations that* ***fixing and verifying high severity or critical bugs*** *at a very large number is*
  - ✓ *Time-consuming*
  - ✓ *Resource hogging (human + machine)*
  - ✓ *Prone to collateral(security), fixing critical bugs mostly touch a large part of the code including the intersection areas.*

# *Cntd…*

- ***Lastly,*** *if a large number of critical bugs are found during the end of a given release*
- *Then, one or more of* ***the following negative developments take place.***
  - ✓ *High probability of Testing cycle being extended.*
  - ✓ *High probability of release deadline being missed.*
  - ✓ *A particular feature having a large number of defects, pulled out from that particular release.*
  - ✓ *Customer commitments are being missed.*
- ***How about the other Defects?*** *There are medium and low-priority defects that will be identified and logged by the Testers.*
- ***It is a well-known fact*** *that no amount of Testing can extract every defect that a Software Product or System has.*
- *Meaning, practically,* ***neither there is an end to testing nor the product is defect-free.***

# Cntd…

- However, from the '**Serviceability' point of view in a Competitive and Time To Market (TTM) model,** there is a need to break the typical mindset to unearth maximum defects early in a Release cycle, **especially identification of critical** and **high severity defects**.
- Any or all of the above will have **a negative impact** on the Organization's business.
- In this context, **adopting 'Early Testing' as a separate Test activity will be beneficial for the overall management of SDLC for a given Project or Release**.

**The Scope of Early Testing Effort:** **Testing Early as a new activity to be tracked exclusively during the course of Testing execution,** it is recommended to practice the **scope of the testing effort** as explained below.

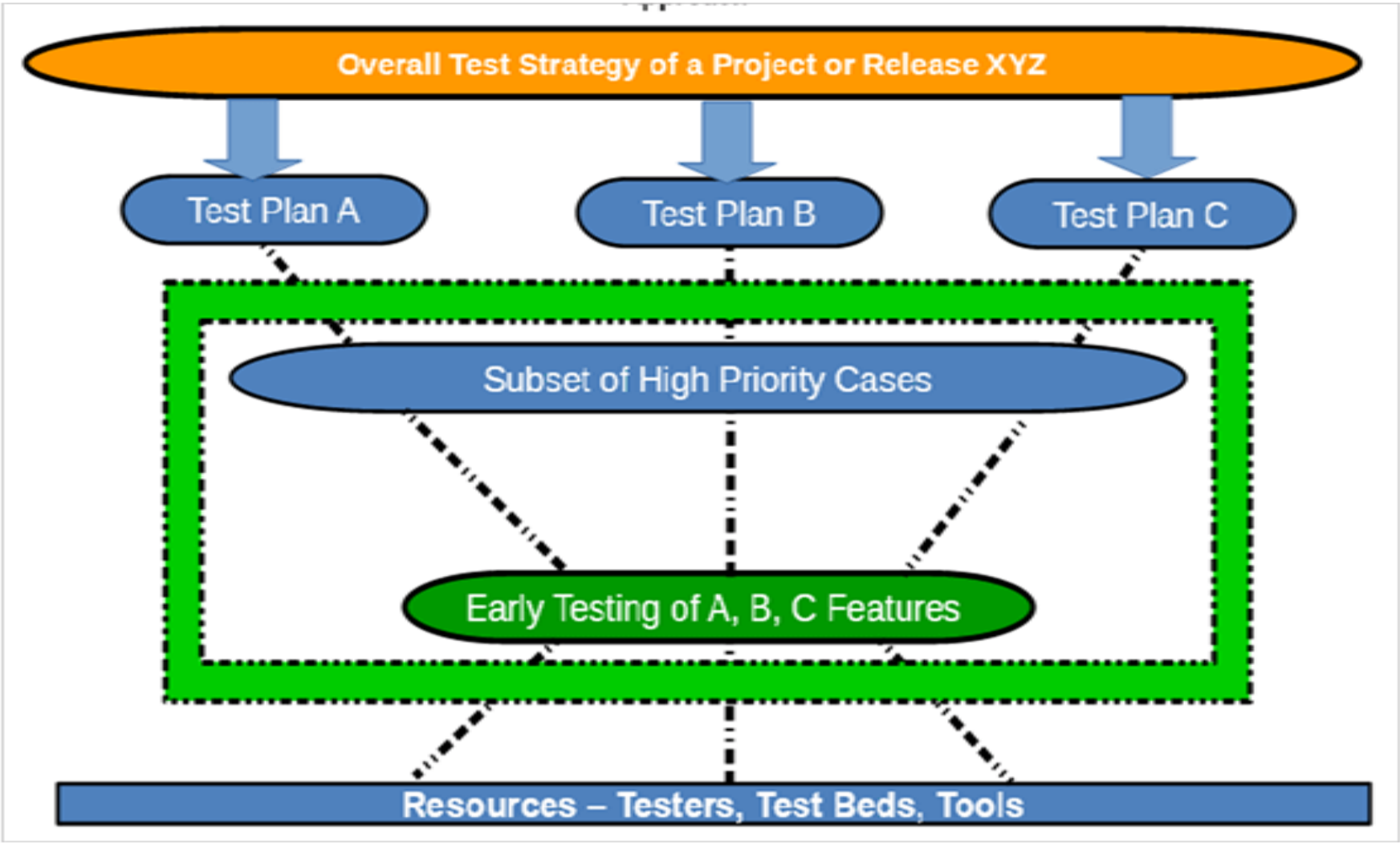
# Cntd…

*Assumption:*

- ***The entire Project or Software Release schedule*** *is approved and made available to all the stakeholders.*
- ***Overall Test Strategy document*** *is developed, reviewed, and approved by all the stakeholders.*
- *High, Medium, Low* ***priority features to be tested*** *are well documented.*
- ***Test Plans and Test cases*** *for all the Features are developed, reviewed, and approved by all the stakeholders.*
- *All Test Plans and Test Cases are uploaded to a central repository for tracking testing execution.*
- ***All human resources, infrastructure equipment, and tools are available for setting up the testbed(s) and executing Test plans.***

# Cntd…

**Figure 4 – Overall approach to the scope of Testing Early**



# Cntd…

*Conclusion*

- *Customers or end-users buy or adopt serviceability products or a system or solutions.*
- *Validating a software that is running on such system or products for its serviceability is the primary requirement.*
- *Key components of Principles of Testing like Why to Test? What is Testing? What to Test? How to Test? are mostly well defined and understood.*
- *However, there are some lasting questions that keep supporting up in the mind of the Readers, Testers, Leads, and Managers on concepts like Early Testing.*
- *Adoption of Early Testing as an integral activity of the overall Testing Schedule for any given Software Project or a Release immensely benefits the Organization to deliver a robust qualified Product or a System.*
- *The importance of Early Testing in your career? Feel free to share your thoughts and experiences in the comments section below!!*

# Use Case Scenario Testing

## Generating Test Cases from Use Cases

- The verification of the correct implementation of use eases is a vital task in software development and quality assurance.
- Nowadays, **use cases are a widely used technique to define the functional requirements of software systems**.
- There **are two main gaps in the generation** of test cases from use cases: lack of automatism and absence of empirical evaluation.
- **The automatism of scenarios analysis** written in natural language (first gap) has been resolved using language patterns and regular expressions for extracting information from use case templates.
- **The main goal of empirical evaluation** and its original contribution is the execution of cases studies to measure and evaluate the effectiveness of **scenario analysis technique.**

# *Cntd…*

- ***The scenario analysis technique*** *is a common technique for generating test cases from use cases.*
- ***It identifies the scenarios from a use case and generates test cases from them.***
- ***A use case is mainly defined by natural language and it is mainly composed of steps.***
- *In Thus, the first task is to translate the behavior of a use case into a more formal model.*
- ***An UML Activity diagram has been chosen to define the behavior of a use case.***
- ***An Activity diagram allows*** *indicating if an action is performed by the system or by an external action; it includes different execution flows.*
- ***It does not need to expose*** *information about the implementation of the system or its external interfaces.*

# *Cntd…*

Table 1. Use case example

```
<useCase id="Search link by description">
  <description> A use case searches a set of links by their description and shows the results. </description>
  <mainSequence>
      <step id="1"> The visitor asks the system for searching links by description.    </step>
      <step id="2"> The system asks for the description. </step>
      <step id="3"> The visitor introduces de description. </step>
      <step id="4"> The system searches for links which match up with the description introduced by the visitor. </step>
      <step id="5"> The system shows the found results. </step>
 </mainSequence>
 <alternativeSteps>
    <astep id="3.1"> At any time, the visitor may cancel the search, then the use case ends.  </astep>
    <astep id="4.1"> If the visitor introduces an empty description, then the system
                     searches for all the stored links and step 5 is repeated.   </astep>
  </alternativeSteps>
  <errorSteps>
    <estep id="4.2"> If the system finds any error performing the search, then an error
                     message is shown and this use case ends.   </estep>
    <estep id="4.3"> If the result is empty, then the system shows a message and this use case ends.   </estep>
  </errorSteps>
</useCase>
```

# Cntd…

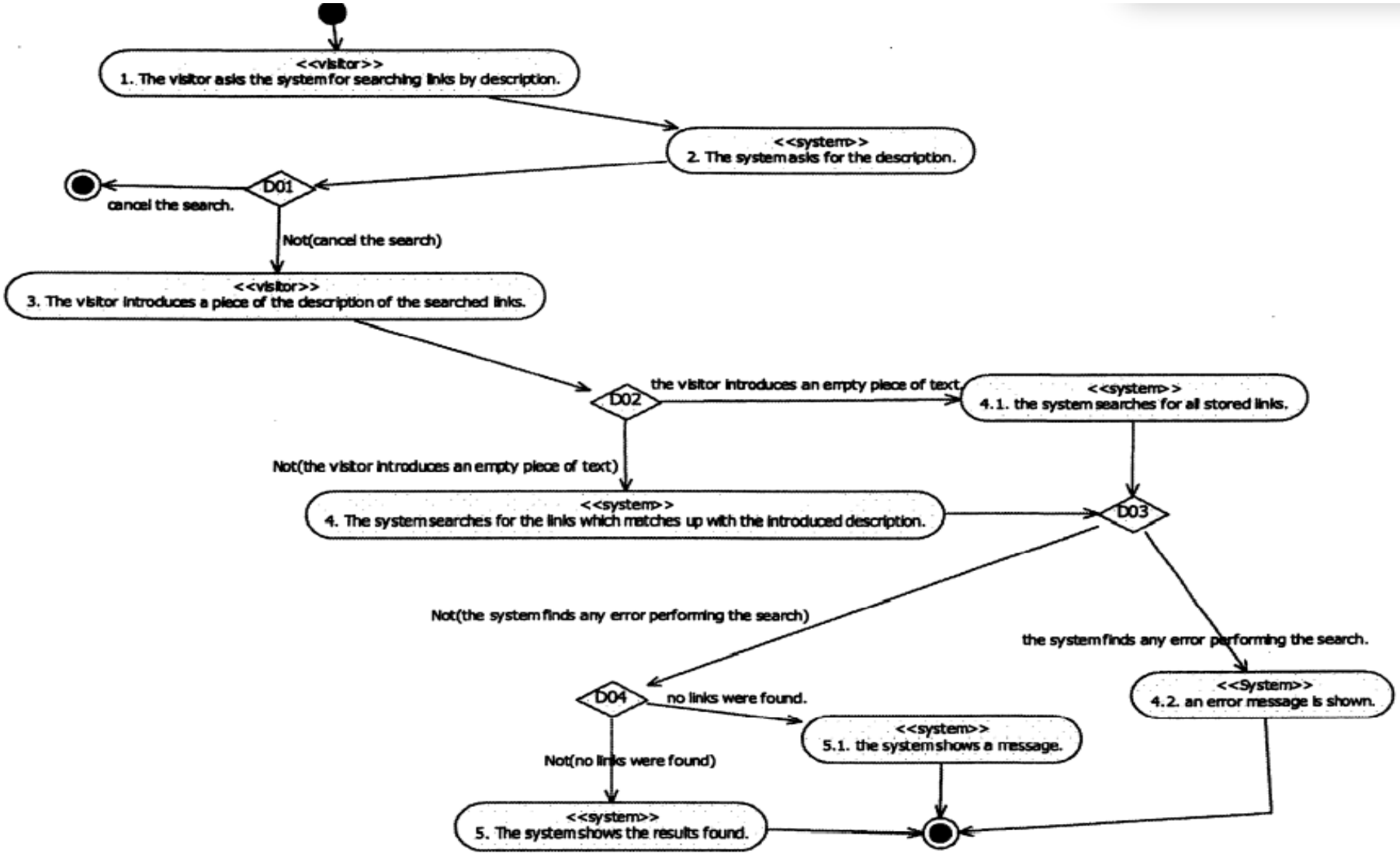



Fig 1. Activity diagram automatically generated.

# *Cntd…*

- *Then, use case scenarios are derived from the activity diagram.*
- *If the activity diagram has not got any loops, as in figure 1, the all-scenarios criterion selects the paths that go through all output object-flow edges from decision nodes at least once.*
- *If the activity diagram has **got some loops**, the all-scenarios criterion selects the paths that go through all output object flow edges from decision nodes and all combinations among loops at least once. .*
- *The numbers in each test case indicates the activities and decisions traversed from the activity diagram.*

# *Cntd…*

**Table 2. Paths and a use case scenario.**

```
Use case: Search link by description
The All-Scenarios Criterion
Test cases (Tc): 7

1: 1, 2, D01, End.
2: 1, 2, D01, 3, D02, 4.1, D03, 4.2, End.
3: 1, 2, D01, 3, D02, 4.1, D03, D04, 5.1, End.
4: 1, 2, D01, 3, D02, 4.1, D03, D04, 5, End.
5: 1, 2, D01, 3, D02, 4, D03, 4.2, End.
6: 1, 2, D01, 3, D02, 4, D03, D04, 5.1, End.
7: 1, 2, D01, 3, D02, 4, D03, D04, 5, End.
```

```
Use case scenario 1:

1: The visitor asks the system for searching
links by description.
2: The system asks for the description.
D01: The visitor cancels the search then the use
case ends.
End.
```

# Cntd…

- ***Test case generation,*** *Test cases were generated over the activity diagram generated from each use case Scenario.*

| *Use cases* | AllNodes | AllScenarios | AllTransitions |
|---|---|---|---|
| *Add new list* | 3 | 10 | 5 |
| *Search links* | 6 | 7 | 5 |
| *List recent links* | 3 | 3 | 3 |
| *View details of a link* | 1 | 3 | 3 |
| **Total:** | 13 | 23 | 16 |

*Table. Test cases*

- *Preform the testing activates and then recorded and report the Test case result*

*End of chapter Seven*

*Any question???*