Static Techniques

Compiled by - Nazmus Sakib Akash

Chapter III –Static Testing Techniques

-III/01 Reviews and the test process

-III/02 Review process

-III/03 Tool based static analysis

01. Reviews and the test process

Basic Approach

- Static testing techniques summarize various methods, that do not execute the component or the system that is being tested.

Static tests include:

- **Review** (Manual activity)
- **Static analysis** (Mostly tool based activity)

Static techniques have certain characteristics:

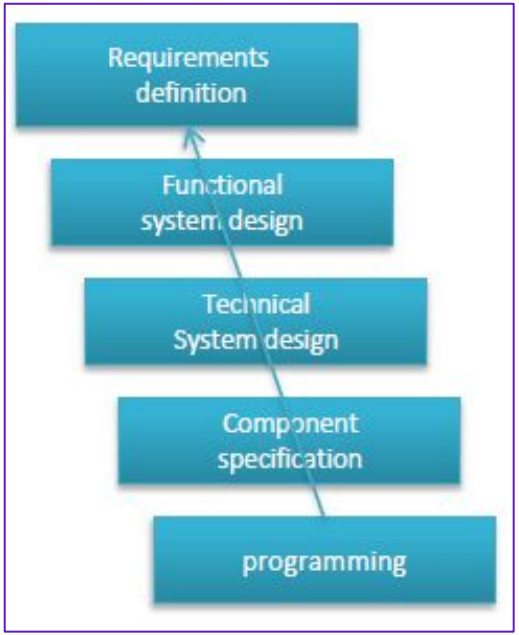
- Static tests find defects rather than failures
- Concepts are inspected as well, not only executable code
- Defects / deviations are found in an early phases, before they are implemented in the code
- Static tests might find defects not found in dynamic testing

High quality documents lead to high quality product.

- Even if the reviewed specifications do not contain any errors, interpreting the specification and creating design could be faulty

Review Objectives

- Review are done in order to improve product quality.
    - Review are used to verify the correct transition from one phase to the next phases, as defined in the left half of the V-model.
- Detecting errors early saves costs
- During reviews, the following error must be detected:
    - Errors in the specification
    - Errors in the design and architecture
    - Errors in the interface specifications
    - Deviations from agreed standards (e.g. programming guides)



Advantages and Drawbacks of Reviews

Advantages

- Lower costs and a relatively high saving potential.
- Errors in documentation are detected and corrected early.
- High quality documents improve the development process.
- Increase rate of communication/ exchange of know-how.

Drawbacks

- Stress may arise if the author is confronted directly.
- Experts involved in reviews need to attain specific product knowledge, good preparation is necessary.
- Considerable time investment (10% -15% of the overall budget)
- Moderator/ participants influence review quality directly.

02. Reviews process

Phases of a Review (I)

Planning phases

- Organizing the review, select and supplementary information.

Organizational preparation (and-kick-off)

- Hand out of review objects and supplementary information.

Individual preparation

- Reviewers inspect objects, note item in need of clarification.

Phases of a Review (II)

Review meeting

- Meeting of review members, reviewers present their results

Rework

- Author fixes any defects addressed by inspectors

Follow up

- The review meeting protocol is distributed to the manager, stating object under test, participants, roles, recommendation

Roles and responsibilities

(Project-) Manager

- Initiates the review, decides on participants and allocates resources

Moderator

- Runs the meeting / the discussion, mediates. summarizes

Author

- Exposes himself to the criticism, performs recommended changes.

Reviewer (also: inspector or checkers)

- Discover defects, deviations, problem areas etc

Scriber (also: recorder)

- Documents all issues, problems and open points that were identified

Types of Reviews (EEE 1028) /1

The basic process of a review –as outlined here –applies to the following variants of reviews

- **Inspection**, **walkthrough**, **technical review**, **informal review**
- These variants differ in a few aspects from the general outlined base practice

A further distinction of review is mode depending on the nature of the reviewed object: product or process

SW-development processor project process

- CMMI, IEC 12207, IPI are terms relating to process improvement
- Also called management review, these review do not directly interface with the testing process, they are not part of this lecture

Documents / products of the development process

- These reviews are addressed here.

Types of Reviews (EEE 1028) /2

Inspection: Key Characteristics

- Formal process based on rules, uses defined roles.
- Review inspect the object under review using checklists and metrics(e.g. problems per page)
- A trained, independent moderator is leading the review.
- Review-ability of the object is assessed prior to the review.
- Formal process for preparation, execution, documentation and follow up activities.

Types of Reviews (EEE 1028) /3

Inspection: Advantages and Drawbacks

- Well organized formal session with clear roles.
- Needs intensive preparation and clear roles.
- Moderator and scribe are necessary.
- Main purpose: finding defects using a structured method.

Types of Reviews (EEE 1028) /4

Walkthrough: Key Characteristics

- There is an optional pre-meeting preparation of the reviewers
- The meeting is led by the author, he explains the object under review
- A separate moderator is not necessary (the author moderates)
- During preparation by the author, the reviews try to locate deviations and / or problem areas.
- Example for using walkthroughs
  - Walkthroughs of documents
  - Walkthroughs of drafts for user interfaces
  - Walkthroughs of business process modeling data

Types of Reviews (EEE 1028) /5

Walkthrough: Key Characteristics

- Little effort in preparing a review session, but it is an open-end session.
- Session can be initiated on short term notice.
- Author has a great influence on the outcome: since she/ he moderates the review, there is a danger of domination by the author (critical points are not addressed in depth)
- Little control possible, since author is also in charge of any follow-up activities.

Types of Reviews (EEE 1028) /6

Technical Review: Key Characteristics

- Target of examination is a technical aspect of the object under review: is it fit for use?
- Experts are needed, preferably external experts.
- The meeting might take place without the author.
- The review is done using technical specifications and other documents.
- A unanimous vote of recommendation is given by the expert panel.
- Intensive preparation is needed

Types of Reviews (EEE 1028) /7

Informal Review: Key Characteristics

- Simplest form of reviews
- Often initiated by the author
- Only reviewers (one or more) will be involved
- No separate meeting necessary
- Results may be recorded in form of an action list
- Often performed in such a way that a colleague is asked to review a document
- Also called: peer review

Types of Reviews (EEE 1028) /8

Informal Review: Advantages and Drawbacks

- Easy to perform, even on short term notice.
- Cost effective.
- No protocol needed.

Success Factors of a Review (I)

- Reviews are to be performed goal oriented, i.e. deviation in the reviewed object should be started in an unbiased manner.

- The author of the reviewed object should be motivated in a positive manner by the review (“your document will be better still” instead of “your document is of poor quality”)

- Systematic usages of the introduced technique and templates.

- Using checklists will improve the efficiency of a review.

Success Factors of a Review (II)

- Sufficient budget is needed to perform proper review (10% to 15% of the overall development cost)

- Make use of the lesson learned effect, use feedback to implement a continuous improvement process

- Duration of review meeting (inspection): 2 hours maximum

Summary

- During static testing, the test object is not executed.
- Review can take place during the early phases of the development process, they complement/ extended the methods of dynamic testing
- **Phases of a review:**

  Planning-preparation –individual preparation –meeting –follow up
- **Roles and takes for the Review:**

  Manager –Moderator –Author –Reviewer –Scribe
- **Types of reviews:**

  Inspection –Walkthrough –Technical review –Informal review

03 –Tool Based Static Analysis

Terminology and Definitions

- Static analysis (Definition):

  Static analysis is the task of analyzing a test object (e.g. source code, script, requirement) without executing the test object.

- Possible aspect to be checked with static analysis are:
  - programming rules and standards
  - program design (control flow analysis)
  - use of data (data flow analysis)
  - complexity of the program structure (metrics, e.g. the cycloramic number)

General Aspects /1

- All test objects must have a formal structure

    - This is especially important when using testing tools.
    - Very often documents are not generated formally.
    - In practice, modeling, programming and scripting languages comply to the rule, some diagrams as well.
    - The tool-based static analysis of a programmed with less effort than an inspection.
    - Therefore, very often a static analysis is performed before a review takes place.

General Aspects /2

- Tools to be used are compilers and analyzing tools (analyzer)

- Compiler

    - detect syntax errors inside a program source code
    - create reference data of the program (e.g. cross reference list, call hierarchy, symbol table) of the program
    - check for consistency between types of variables
    - find undeclared variables and unreachable (dead) code

- Analyzer address additional aspects, such as

    - conventions and standards
    - metrics of complexity
    - object coupling

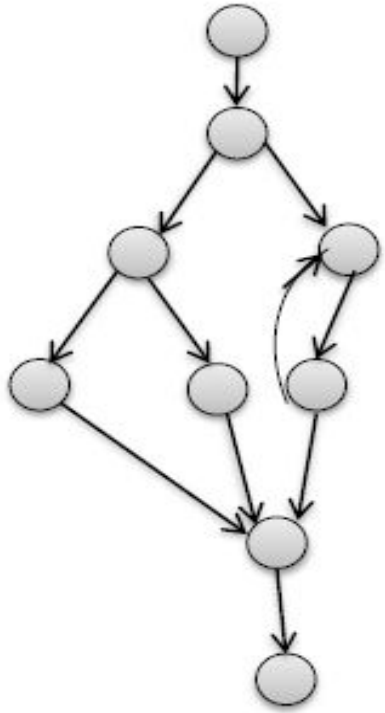
Control Flow Analysis /1

Aim

- To find defects caused by wrong construction of the program code (dead branches, dead code etc.

Method

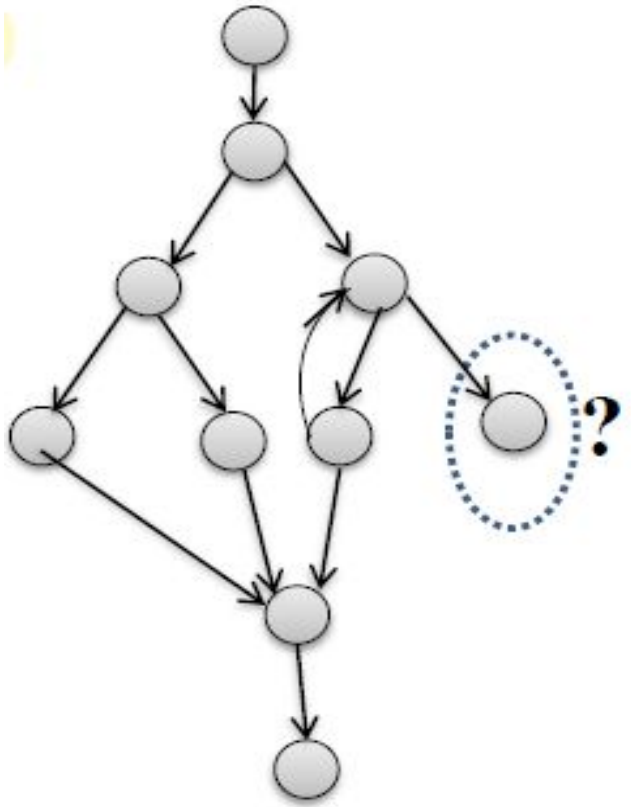
- Code structure is represented as a control flow graph.
- Directed graph
  - Nodes represent statements or sequences of statements.
  - Edges represent control flow transfer, as in decisions and loops.
  - Tools based construction.



Control Flow Analysis /2

- **Result**
  - Easy understandable overview of program code
  - A control flow graph is just a simplified version of flow chart
  - Anomalies* can easily be detected, defects stick out
    - loops exited by jumps
    - dead branches
    - multiple returns

* Anomalies: an irregularity of inconsistency



Data Flow Analysis /1

Aim:

- To discover data flow anomalies with help of control flow graph and sensible assumption of data flow sequences.

Benefits:

- Reliable detection of data flow anomalies.
- Exact location of errors can be determined easily.
- A good supplement for other testing methods.

Drawbacks:

- Limited to a narrow range of error types.

Data Flow Analysis /2 – Method

A variable x may have the following different stages during program execution:

- x is undefined (u): no value is assigned to x
- x is defined (d): a value is assigned to x (e.g. x = 1)
- x is referenced (r): a reference is taken, the value of x does not change (e.g. if (x>0 or a=b+x)

The data flow of a variable can be expressed as a sequence of the states: d, u, and r,
e.g. x-> u d r d r r u

If one of these sequence contain a sub sequence which does not make sense, a data flow anomaly is identified:

ur–anomaly: undefined value gets referenced

du–anomaly: defined value gets undefined before reading

dd–anomaly: defined value gets defined again before reading

Data Flow Analysis /3

For this example the values of two variables are exchanged via an auxiliary variable, if they are not stored by value.

```
void MinMax (int Min, int Max)
{
Int Help:
If (Min > Max)
{
Max = Help;
Max = Min;
Help = Min;
}
End MinMax
}
```

The data flow analysis shows:

- Variable Help is still undefined when it gets referenced: ur-anomaly
- Variable Max gets defined twice without any reference in between: dd-anomaly
- Defined value for Help gets undefined (program ends) without reference: du-anomaly

This example can be corrected easily:

```
void MinMax(intMin, intMax){
IntHelp:
        If (Min > Max){
                Max = Help;
                Max = Min;
                Help = Min;

        }

End MinMax

}
```

Metrics and Their Computation

Using Metrics, certain aspect of program quality can be measured

- The metric has only relevance for the measured aspect

The static complexity of the program can be measured

- Currently, there are about 100 different metrics available

Different metrics address different aspects of program complexity

- program size (e.g. Lines of Code -LOC)
- program control structure (e.g. cyclomatic number)
- data control structures (e.g. Halstead-Metric)

It is difficult to compare different metrics, even if they address the same attribute of the program !

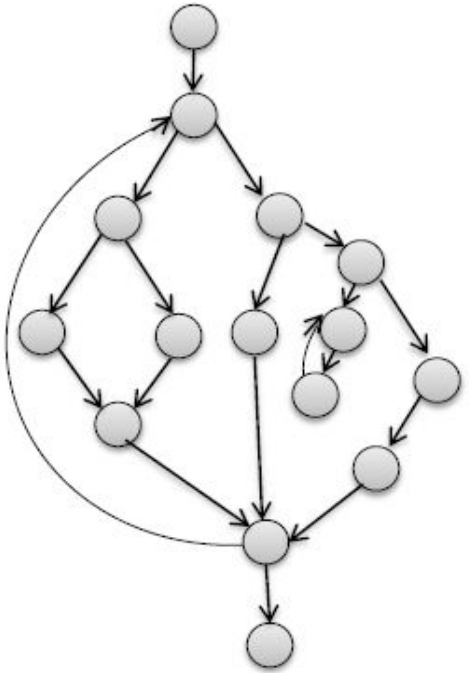
Metrics and Their Implementation /1

Cyclomatic Number v(G)

- Metric to measure the static complexity of a program based on its control flow graph
- Measure linear independent program paths, as an indication for testability and maintainability
- The cyclomatic number is made up of
  - Number of edges e
  - Number of nodes n
  - Number of inspected independent program parts p (mostly 1)

Values up to 10 are acceptable. Beyond this, code should be reworked/improved (best practice, McCabe)

v(G)=e-n+2p



Metrics and Their Implementation /2

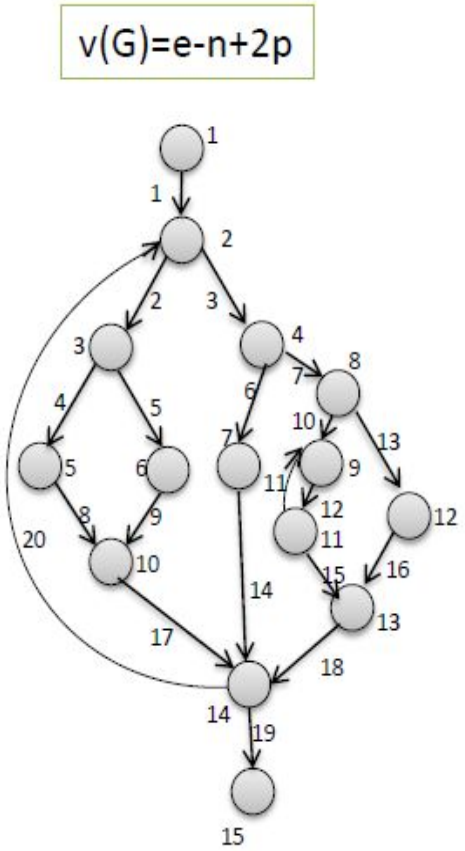
Example III/03-2 (Cyclomatic Number)

- the number to the right has
  - 1 program part: p = 1
  - 15 nodes: n = 15
  - 20 edges: e = 20

The number to a cyclomatic number of

v (G) = e-n+2p

v(G)=7



Metrics and Their Implementation /3

Cyclomatic Number (by McCabe) - Implication

- The cyclomatic number can be used as a target value for code reviews.

- The cyclomatic number can also be calculated as the number of independent decisions plus one. If both ways of calculation give different results, it may be due to
    - A superfluous branch
    - A missing branch

The cyclomatic number also gives as indication for the number of necessary test cases (to achieve decision coverage)