Faculty of Computers and Information
BSc. Software Engineering Program

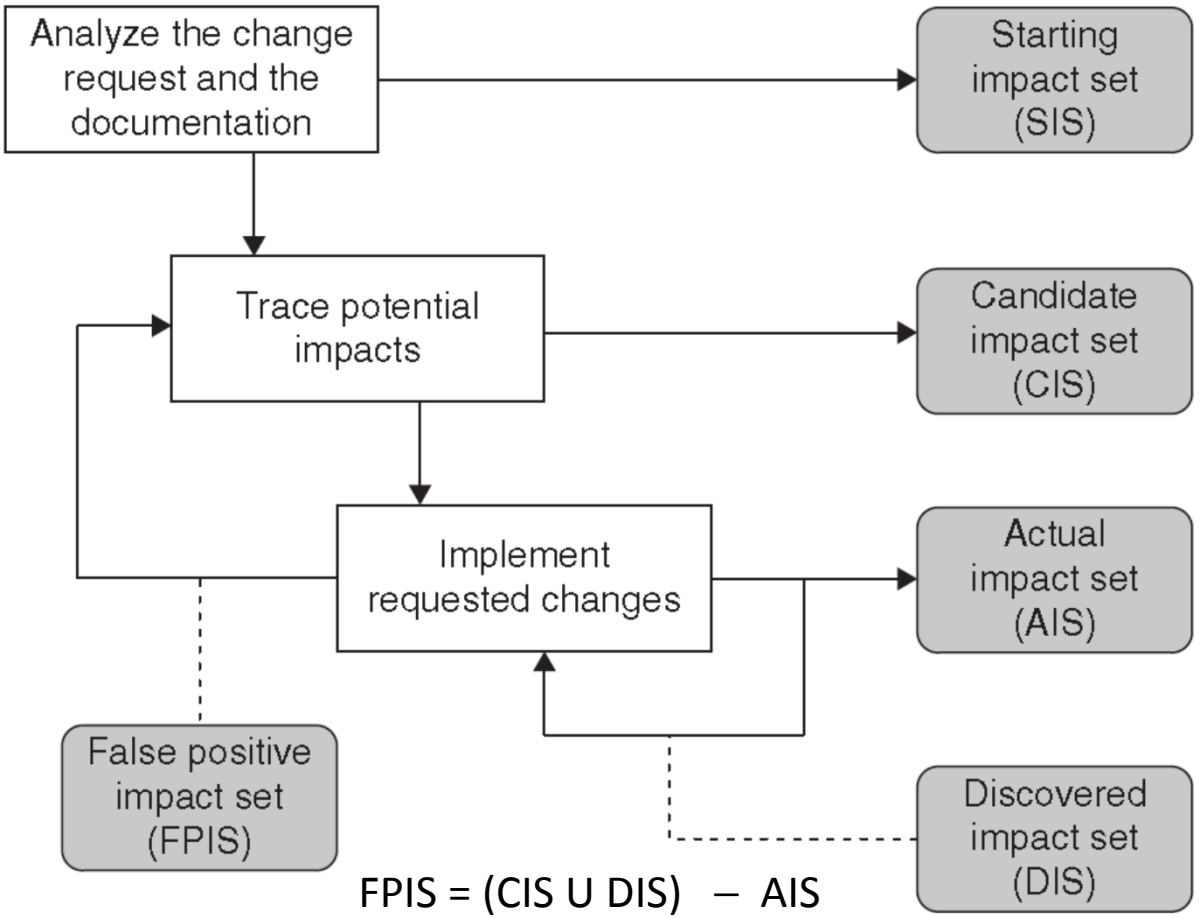
Dr. Lamia Abo Zaid

د. لمياء أبوزيد

Software Evolution : TOC

1. Introduction to Software Evolution
2. Taxonomy of Software Maintenance and Evolution
3. Evolution and Maintenance Models
4. Reuse and Domain Engineering
5. Program Comprehension
6. Impact Analysis
7. Refactoring
8. Reengineering
9. Legacy Information Systems

Impact Analysis Process



The initial set of objects (or components) presumed to be impacted by a software CR

The set of objects (or components) estimated to be impacted according to a certain impact analysis approach

The set of objects (or components) actually changed as a result of performing a CR

the set of new objects (or components), not contained in CIS, discovered to be impacted while implementing a CR.

FPIS = (CIS U DIS) – AIS

Impact Analysis Process

- In the process of impact analysis it is important to minimize the differences between AIS and CIS, by eliminating false positives and identifying true impacts.
- to evaluate the impact analysis process two traditional information retrieval metrics: recall and precision are used:
  - **Recall:** represents the fraction of actual impacts contained in CIS, and it is computed as the ratio of $|CIS \cap AIS|/|AIS|$.
    - The value of recall is 1 when DIS is empty.
  - **Precision:** represents the fraction of candidate impacts that are actually impacted, and it is computed as the ratio of $|CIS \cap AIS| / |CIS|$.
    - For an empty FPIS set, the value of precision is 1

Impact Analysis Process - Adequacy

- Adequacy of an impact analysis approach is the ability of the approach to identify all the affected elements to be modified. Ideally, AIS $\subseteq$ CIS.
- Adequacy is repressed in terms of a performance metric called inclusiveness, as follows:

$$Inclusiveness = \begin{cases} 1 & \text{if AIS} \subseteq \text{CIS} \\ 0 & \text{otherwise} \end{cases}$$

- An inadequate approach is in fact useless, as it provides the maintenance engineer with incorrect information.

Impact Analysis Process - Effectiveness

- Effectiveness is the ability of an impact analysis technique to generate results that actually benefit the maintenance tasks
- Effectiveness is expressed in terms of three fine grained characteristics:
  1. Ripple-sensitivity
  2. Sharpness
  3. Adherence

Impact Analysis Process - Effectiveness

1. Ripple-sensitivity

- implies producing results that are influenced by ripple effect.
- DISO (directly impacted set of objects) is the set of objects that are directly affected by the change
- IISO (indirectly impacted set of objects) is the set of objects that are indirectly impacted by the change is denoted by The cardinality of IISO is an indicator of ripple effect.
- Amplification is used as a measure of Ripple-sensitivity.

$$Amplification = \frac{|\ IISO\ |}{|\ DISO\ |} \longrightarrow 1$$

where $|\ .\ |$ denotes the cardinality operator.

Impact Analysis Process - Effectiveness

2. Sharpness

- Sharpness is the ability of an impact analysis approach to avoid having to include objects in the CIS that are not needed to be changed.
- Sharpness is expressed by means of Change Rate as defined below:

  ChangeRate = | CIS | / | System |

- ChangeRate falls in the range from 0 to 1.
- For Sharpness to be high ➔ Change Rate $\ll$ 1.

Impact Analysis Process - Effectiveness

- **3. Adherence**
- **Adherence** is the ability of the approach to produce a CIS which is as close to AIS as possible.
- A small difference between CIS and AIS means that a small number of candidate objects fail to be included in the actual modification set.
- Adherence is expressed by S-Ratio as follows:

  S-Ratio = | AIS | / | CIS |

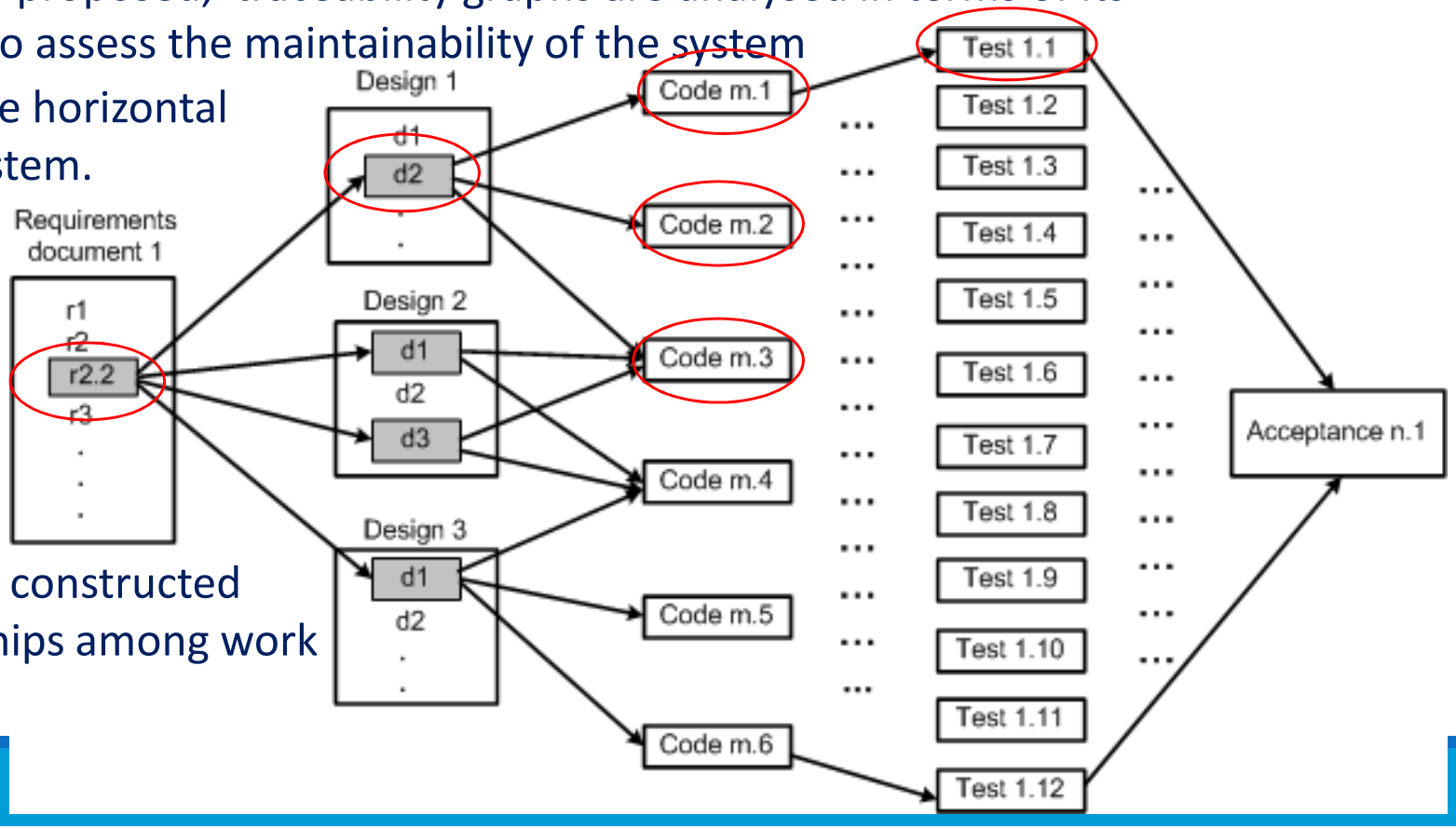
- S-Ratio takes on values in the range from 0 to 1 , Ideally *S-Ratio*= 1.

Identifying the Starting Impact Set (SIS)

- Impact analysis begins with identifying the SIS (Starting Impact Set).
- The CR specification, documentation, and source code are analyzed to find the SIS.
- It takes more efforts to map a new CR's "concepts" onto source code components (or objects).
- In the "concept assignment problem," one discovers human oriented concepts and assigns them to their realization.
- It is difficult to fully automate the concept assignment problem because programs and concepts do not occur at identical levels of abstractions, thereby necessitating human interactions

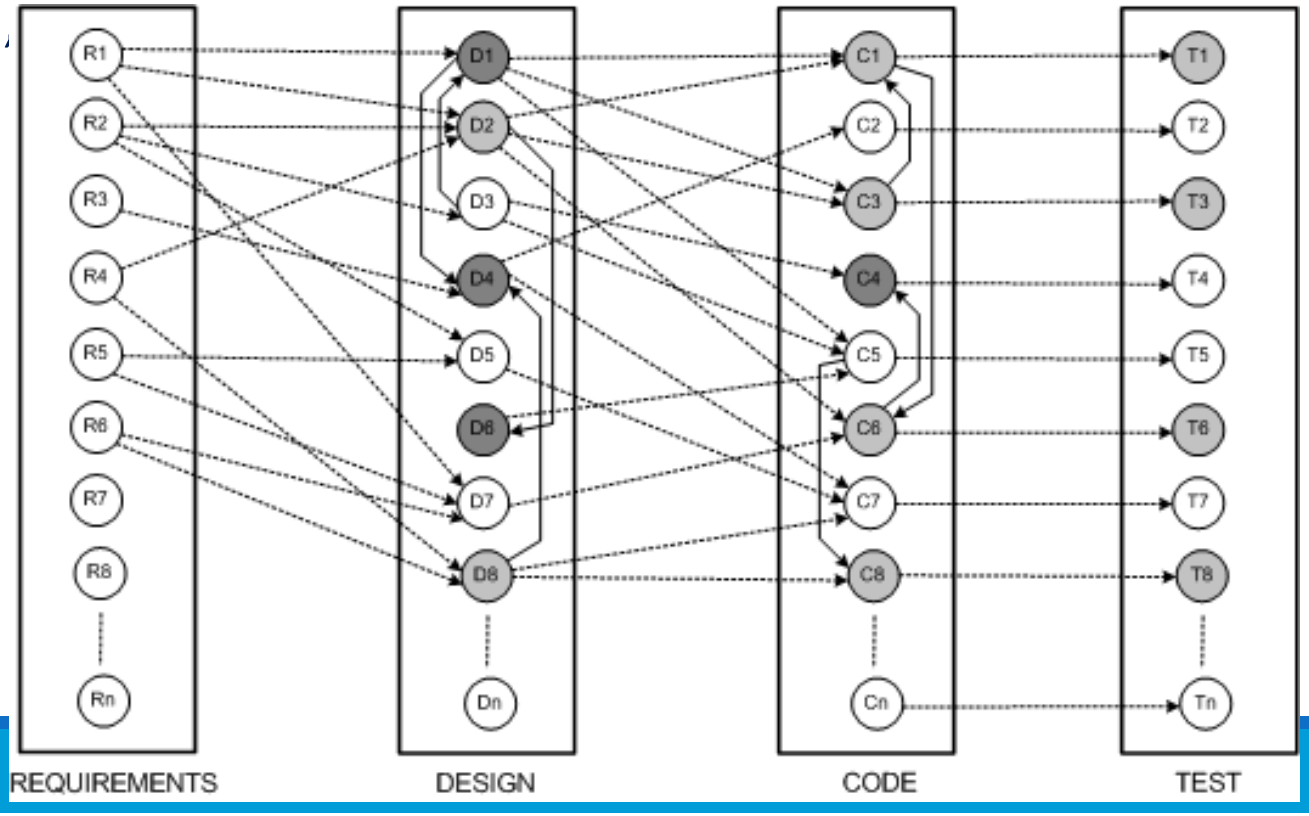
Analysis of Traceability Graph

- whenever change is proposed, traceability graphs are analysed in terms of its complexity and size to assess the maintainability of the system
- The graph shows the horizontal traceability of the system.
- The graph that is so constructed reveals the relationships among work products.



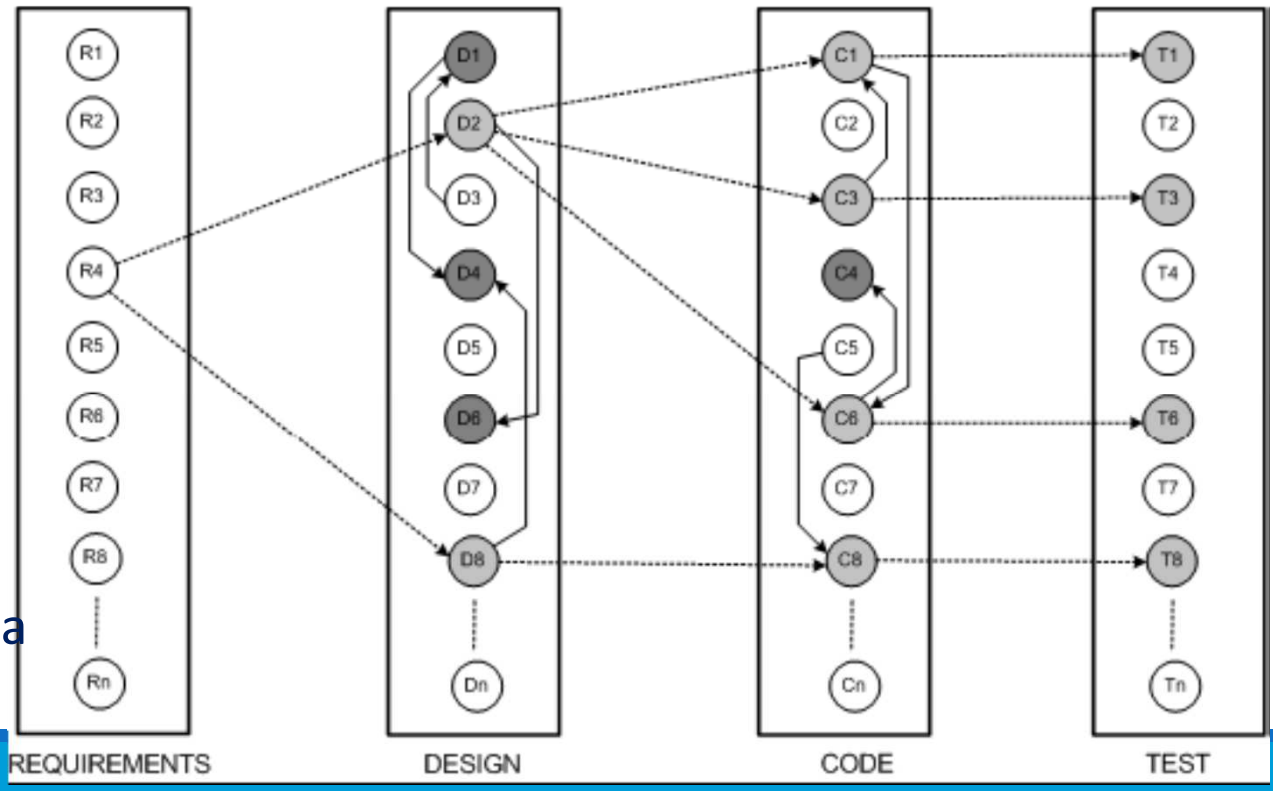
Analysis of Traceability Graph

- The graph has four categories of nodes: requirements, design, code, and test.
- The edges within a silo represent vertical traceability for the kind of work product represented by the silo.



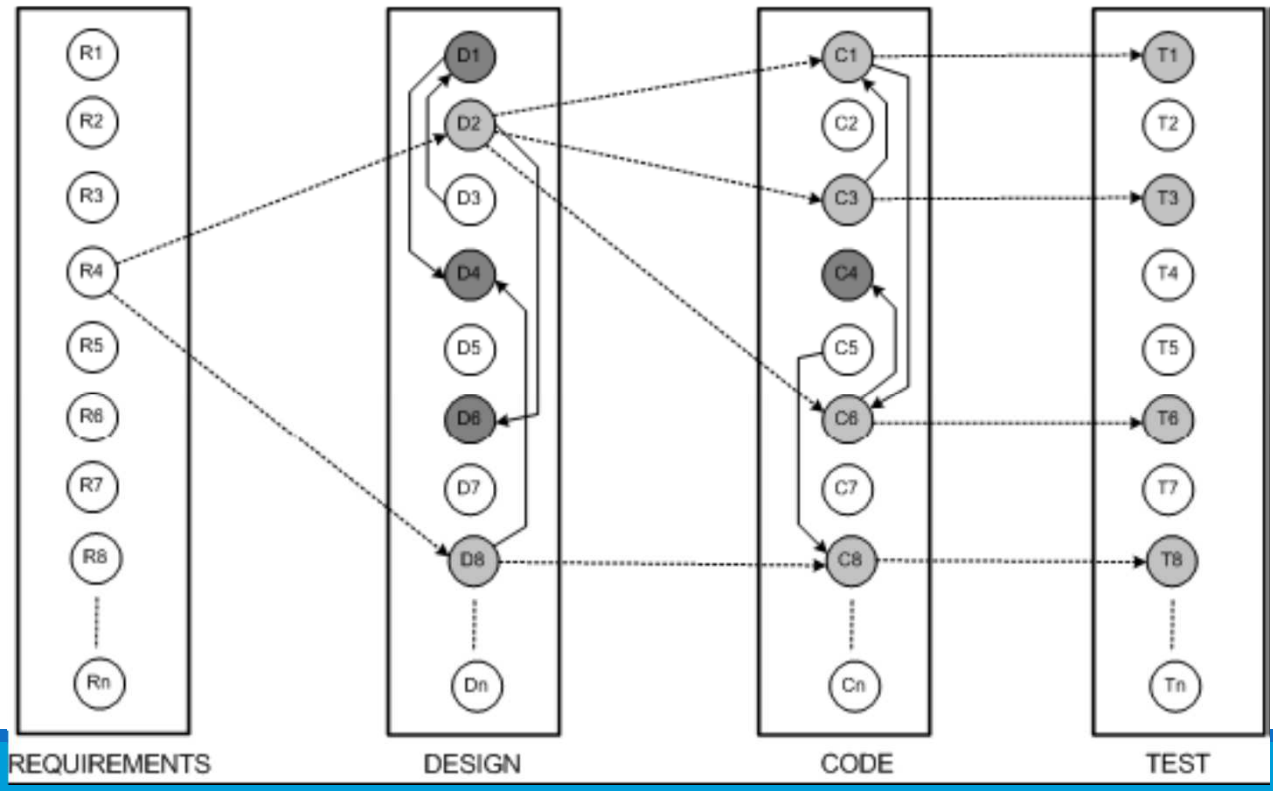
Analysis of Traceability Graph

- Changes made to requirement object "R4," can be traced in terms of horizontal traceability and vertical traceability
- The horizontally traced objects are shown as lightly shaded circles
- The vertically traced objects are shown as darkly shaded circles.
- A common measure of complexity of a graph is the Cyclomatic complexity.



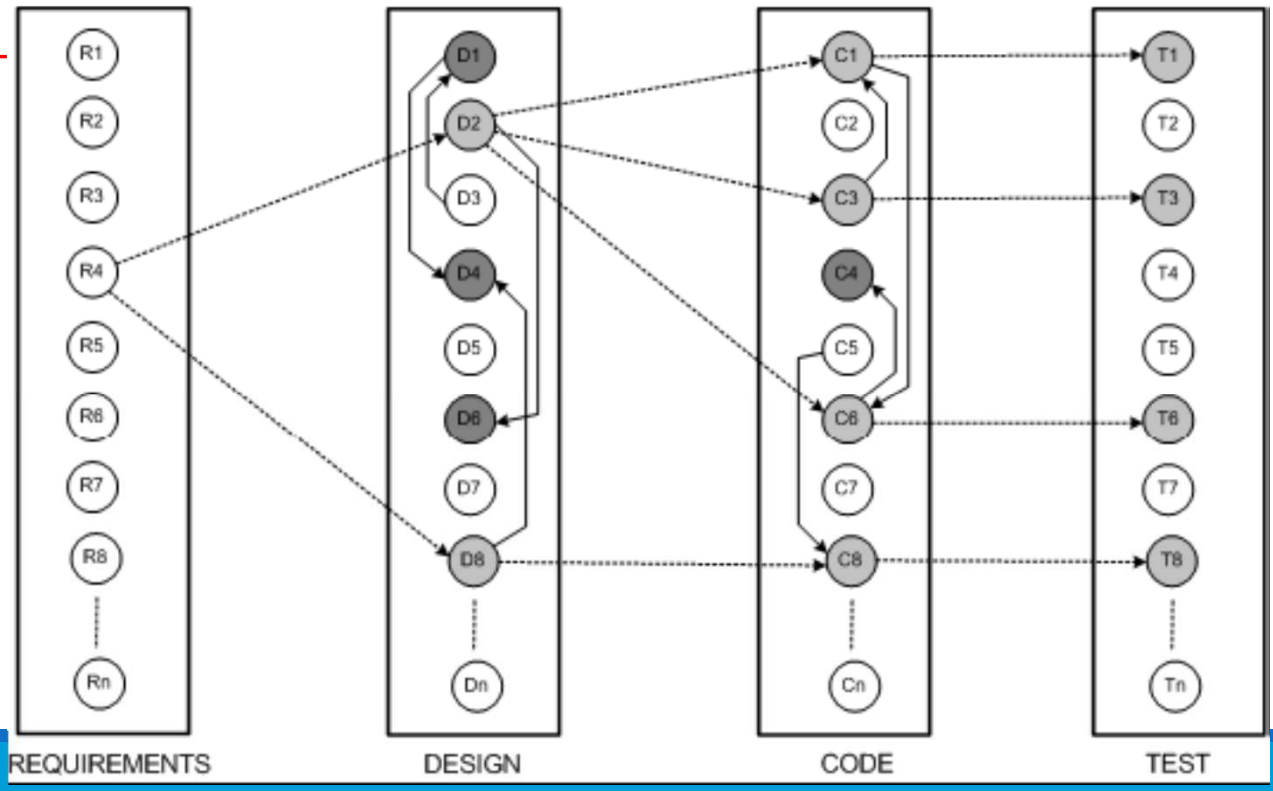
Analysis of Traceability Graph

- For a node i its in-degree **in(i)** counts the number of edges for which i is the destination node, **in(i)** denotes the number of nodes having a direct impact on i.
- The out-degree **out(i)** of node i is the number of edges for which i is the source. out(i) is a measure of the number of nodes which are likely to be modified if node i was changed.



Analysis of Traceability Graph

- node count is a measure of size
- To minimize the impact of a change, out-degrees of nodes need to be made small.
- For nodes with large out-degrees, one may partition the nodes to uniformly allocate dependencies across multiple nodes
- Low in-degrees of nodes are an indication of a good design



Analysis of Traceability Graph

- To understand changes in horizontal traceability, it is necessary to understand:
  - the relationships among the work products.
  - how work products relate to the process as a whole.
- There exist three graphs:
  1. Relating requirements to software design.
  2. Relating software design to source code.
  3. Relating source code to tests.
- If a proposed change results in increased size or complexity of the relationship between a pair of work products, the resulting system will be more difficult to maintain