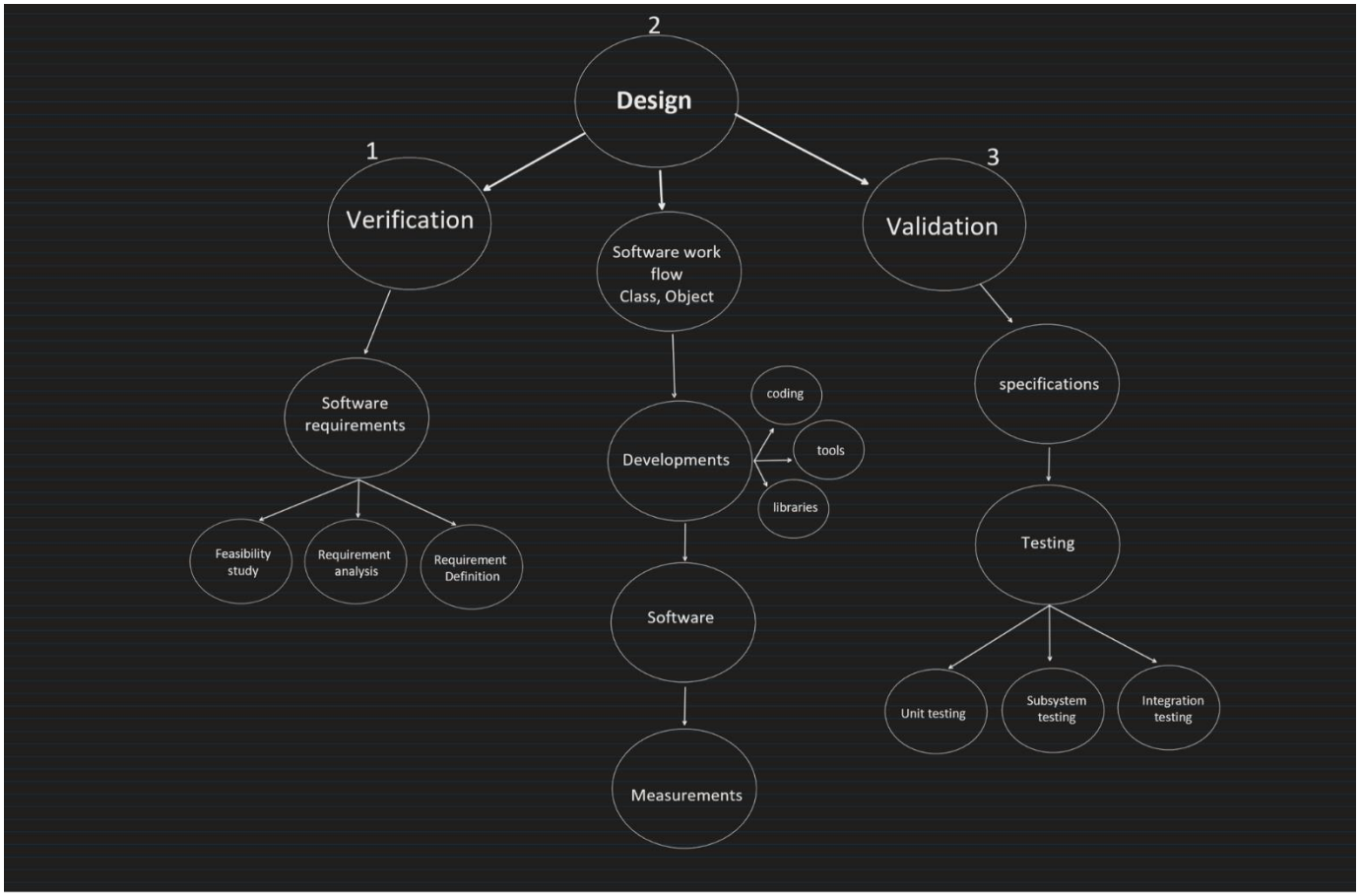
Software Engineering 2 and Software Quality assurance project. External document.

Ravshanbek Musaev, 32034926, Software Engineering group B.



1. Verification. Quality planning.

1.1. Software requirements.

1.1.a. Feasibility study – actual process.

Input:

- Client specification and system specification.
- Budget – define budget based on required quality level.
- Scope of project – to define population from it.
- Audience – users and experts.
- List of operational and technical issues – questionnaire is formulated to identify these issues.
  Questionnaire:
  - Quantitative analysis – users (operational).
  - Qualitative analysis – experts (technical).

Output (decision if the project potentially successful/profitable or not):

- Specification level 1 – Client specification.
  - Components: functions, features, and budget.
    - Functions:
      - Sequential – goes in one sequence.

        - Procedural – not in sequence, has many branches, more expensive to implement.
- Agreed budget document.
- Joint report – align operational and technical issues. Represents actual study. Define backend and frontend parts.
- List of keywords and recommendations – to digitalize process. Obtained from experts' conclusion.

Between Joint report and list of keywords is process of reduction.

Definition of LOQ – Level of Quality. Agree and confirm with client the chose of Quality Level.

1.1.b. Requirements analysis – simplify the process.

Goal seeking analysis. Goal is quality.

Input (from Feasibility study):

- Goal from specification.
- Defined budget.
- Defined Level of Quality.
- Software quality:
    - Software behavior – max load / machine size = threshold.
    - System quality = user I/O + measurement.

Output:

- List of users – types of users who will use software product.
- List of activities – actions available for users in software product.
- Measurements – monitoring user activity, map activities with users.

Placement of indicators and measurement to define threshold and detect errors.

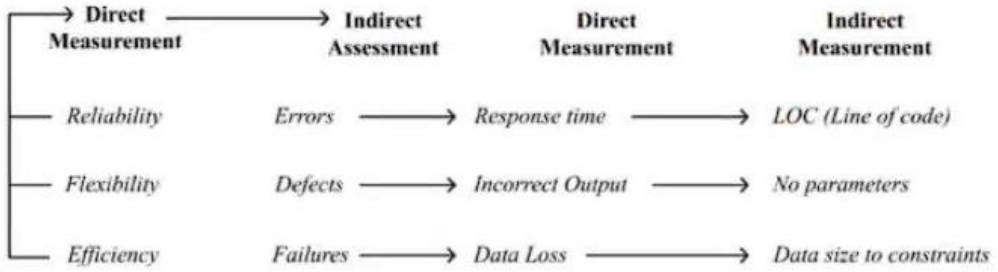
Quality control in Requirements analysis.

Level of Quality = Quality Planning (verify) + Quality Control (validate).
Quality planning -> requirement analysis -> verification with standard rules -> parameters -> metrics -> presume threshold to be measured -> code error -> solve errors.

Functions are measured by:

- Soft operation – by session time.
- Hard operation – by threshold.

To enforce control assessment is used. Direct assessment -> indirect assessment -> direct measurement -> indirect measurement. Assessment and measurement together = indicators of quality.



Direct assessment represented by:

- Reliability
- Security
- Efficiency

Indirect assessment represented by:

- Response time
- Incorrect output
- Data loss

Software measurement is divided into 2 categories:

I. Functional oriented metrics (to report the job based on presumed threshold). Direct assessment -> indirect assessment.

To find if function is working properly firstly, we should set normal behavior.
Reliability = consistency, completeness, accuracy which are interlinked with response time, data loss, wrong output.
To connect indirect assessment to direct measurement we should put metrics in way so every set of metrics indicate and show exact error (response time, wrong output or data loss). Connection between them is requirements and specifications.

II. Software quality metrics. Direct measurement -> indirect measurement

Direct measurements:

- LOC - line on code
- Execution speed
- Memory size
- Defect per time period (report)

Direct measurement represented by threshold of response time, threshold of incorrect output, threshold of data loss.

Indirect measurements:

- Efficiency
- Accessibility
- Availability

Indirect measurement represented by error, defect, failure.

Metrics report: delayed response, incorrect output, and data loss.

Direct indicator of assessment is result of action. Indirect indicator – cause of an action.

Software quality = software product quality (1) + software development quality (2)

1. Lifecycle, standard, procedure
2. Quality analysis method - goal seeking analysis

3. Quality = user input + measurement
3.1. Retrieval of measurement = behavioral and technical
3.2. Retrieval of input = users + activities
3.3. Retrieval of quality = software behavior
4. Software behavior = max load / machine capacity -> threshold
5. Software runtime = threshold and test case sniffer (User Input + Output + Measurement)

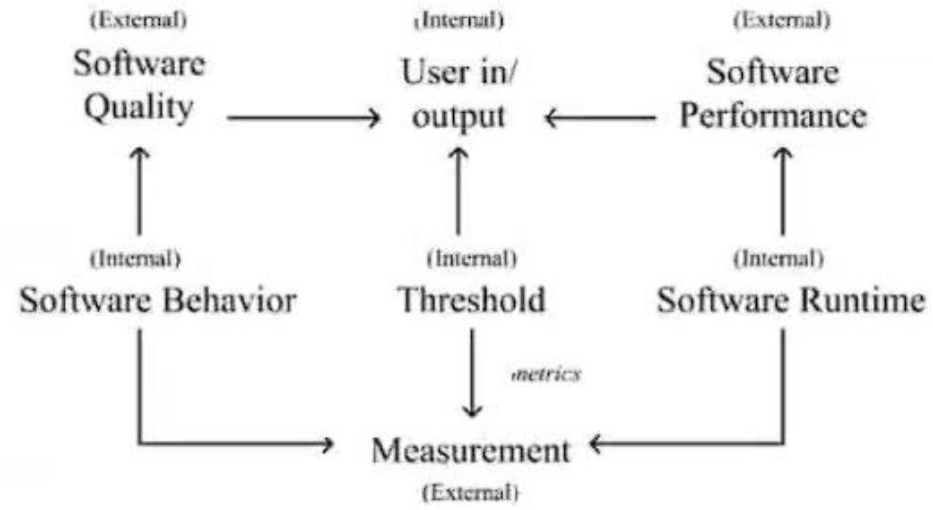
Development objective = product outcome.

1.1.c. Requirements definition – digitalize process.

- Workflow. Class – users, features, items. Map users to features, create separate data structures for all.
- Data structure – choose data structure, map it to functions. It includes table of objects attributes.
- Metadata – information about users and activities, identifies features of users / activities.
- Staging to digitalized form – mapping classes and objects to functions using attributes (derived from metadata).

Output:

- Functional and non-functional requirements.
  - All functional requirements have non-functional counterparts.
  - All non-functional requirements have metrics.
- Specification level 2 – System specification.
  - Components: software (sequential or procedural), hardware, defined level of quality to fit client's specification.



1 layer. Software quality -> input output -> Performance
2 layer. Normal behavior -> Threshold -> Software runtime
3 layer. Measurement.

Software quality = user input + measurement.
Software quality -> Software behavior.
Software behavior = max load ÷ machine size = threshold (bits/second).

2. Design.

2.1. Workflow, class object.

- Design of software product, how it works, diagrams of classes and objects.
- Identify objects -> define classes based on objects -> coupling and cohesion.
- Identify main scenario -> divide to sub scenarios.
- UML represents workflow.

2.2. Developments.

2.2.a. Programming

- Programming language(s), programming style.

2.2.b. Coding.

- Process of implementing the design into product by code.

2.2.c. Tools.

- What tools have been used in design implementation.

2.2.d. Libraries.

- Storage of information.

2.3. Output.

- Prototype of product with set up measurements (measurable software product).

3. Validation. Quality control.

3.1. Specification.

- Check if the software product works properly.

3.2. Testing. Test plan.

3.2.a. Unit testing.

- Testing functions or units of software product separately.

3.2.b. Subsystem testing.

- Testing groups of units together in sub scenarios.

3.2.c. Integration testing.

- Testing the connection between software parts and if the software works as a whole.

4. Source Code Management (SCM).

Documentation to store and retrieve data.

- To archive and serve as the only entry/exit point while manipulating development files.
- Keep record of versions in the development and further exploitation of software product.
- Primary portfolio design of development documents.
- Optimization of disc storage.

Main SCM documents.

- Verification stage – formal version of SRS with standards to run each phase of development.
- Design stage – UML Entity relation model, not coding yet.
- Development stage – Code standards (for programming and coding).
- Validation stage – Exit report including test cases.

Functions of SCM.

- Version control – most important function. Keeps record of the versions during development and maintenance, including any changes, errors occurred, error solutions.
- Access management – managing how the source files are shared, who has access to them.
- Locking – prevention of access to the files of anyone except one person at once.
- Merging – merging the work on file of several people and updating code.
- Labelling – label all checkpoint during development.

Iterative refinement.

- Max 10% error rate is a standard in industry. Error rate = Lines of code (LOC) / number of Parameters.
- Process of repeating the procedure.
- Requirements -> Design -> Development -> Testing (evaluation) loop, which gives output of documents SRS -> UML -> coding documentation -> Exit report and errors evaluation.
- Requirements come from client. Client's specification is developer's requirements.
- Initial version of software product undergoes series of trials with error detection, fixing and testing, until the error rate is less or equal to 10%.
- Procedures.
    - Data.
    - Descriptive analytics.
    - Predictive analytics.
    - Prescriptive analytics.
    - Make decision.
    - Take actions.

Improvement and reengineering.

Fixing and improving software product, constant process.

- Process improvement.
    - Initiated by measurements when case sniffers detect threshold exceeding.
    - Identifying what points need improvement, based on measurements.
    - Current version "As is" (v.1) -> identify experimental version "To be" -> establish measures -> follow process -> measure performance -> implement improvements by solving problems -> updated version "As is" (v.2).
- Process reengineering.

- Identify points to be improved.
- Measurement -> improve -> reengineering.
- Set project scope -> Study competition -> Create new processes -> Implement a solution.
- Main factors of successful software product.
    - Flexibility
    - Scalability
    - Reliability
    - Availability
    - Performance