2.1 EVALUATION PROCESS

A general evaluation objective is to appraise or assess the significance. It helps to understand the state-of-practice and confirm the underlying theories. When an area is not well understood we need to explore it, describe the current state of things, predict the future and explain why things happen. General evaluation objectives include general understanding of the current practice, confirming the theories or conventional knowledge also exploring when a domain is not well understood and describing the current state of things. Evaluation helps in predicting the future and explaining why the things or sequences are taking place.

In software engineering, for evaluation measures we need to understand the software process and the product. The qualities of process and product are defined, measured and validated. We have to evaluate and confirm the successes and failures. The information feedback for project control can also be conducted. Experience helps to learn to predict and plan the software process. Software process needs to evaluate technology and improve software development.

The software engineering evaluation objectives include understanding the software process and product and defining, measuring and validating qualities of process and product. It also includes evaluating and confirming successes and failures. Information feedback for project control is also included. Evaluation is done on the basis of learning from the experience, that is, one learns to predict from proper and
precise planning. The technology needs to be evaluated and there should be some improvement in the software development.

Software process evaluation involves analyzing the activities carried out in order to produce appropriate software. The eventual goal of process evaluation is to improve software production. Development process evaluation is based on the proposition that the quality of the software product is determined by the quality of its development process. [40]

Software process evaluation methods have introduced innovative concepts that have changed the way in which software production activities are performed. Process evaluation is used to help you create infrastructure that supports your project or organization, and to evaluate how effectively that process functions. It will also help you assess short-term changes in skills, attitudes, and knowledge of the participants. Because changes in circumstances may require you to make adjustments in your plan along the way to meet your goals, you and your program officer may review your process evaluation during the term of your grant. [41]

As per ISO 9126: 1991, 5.3, [39] the evaluation process consists of three stages and it may be applied in every appropriate phase of the life cycle for each component of the software product:

2.1.1 Quality requirement definition

The purpose of this initial/first stage is to specify the requirements in terms of quality characteristics and the probable sub characteristics. Requirements state the demand of the environment for the software product under consideration, and they must be defined prior to the development. Software product is decomposed into major
components and the requirements derived from the overall product may differ for the different components.

**ISO/IEC 9126** Software engineering: For the evaluation of software quality, product quality is an international standard. The fundamental objective of this standard is to deal with the preconceived notion that can adversely affect the delivery and perception of a software development project. These include changing the priorities after the start of a project. ISO/IEC 9126 tries to develop a common understanding of the project's objectives and goals. [51]

The standard is divided into four parts:

- quality model
- external metrics
- internal metrics
- quality in use metrics.

**Quality Model:** The quality model of the standard, ISO/IEC 9126-1, [50, 52] classifies software quality in a structured set of characteristics and sub-characteristics as follows:

**Functionality** – It is a set of attributes that depends on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs.

- Suitability
- Accuracy
- Interoperability
- Security
- Functionality Compliance
Reliability – It is a set of attributes that depends on the capability of software to maintain its level of performance under stated conditions for a stated period of time.

- Maturity
- Fault Tolerance
- Recoverability
- Reliability Compliance

Usability – It is a set of attributes that depends on the effort needed for use, and on the individual assessment, by a stated or implied set of users.

- Understandability
- Learnability
- Operability
- Attractiveness
- Usability Compliance

Efficiency – It is a set of attributes that depends on the relationship between the level of performance of the software and the amount of resources used, under the stated conditions.

- Time Behaviour
- Resource Utilisation
- Efficiency Compliance

Maintainability – It is a set of attributes that depends on the effort needed to make specified modifications.

- Analyzability
- Changeability
- Stability
- Testability
- Maintainability Compliance

**Portability** – It is a set of attributes that bear on the ability of software to be transferred from one environment to another.

- Adaptability
- Installability
- Co-Existence
- Replaceability
- Portability Compliance

Each quality sub-characteristic is further divided into attributes. An attribute is an entity, which can be verified or measured in the software product. They are not defined in the standard, as they vary between different software products.

This standard provides a framework for organizations to define a quality model for a software product. It is up to each organization to precisely specify the task of its own model. For example, by specifying target values for quality metrics and evaluate the degree of presence of quality attributes.

**Internal Metrics:** Internal metrics are those that do not rely on the software execution.

**External Metrics:** External metrics are applicable to running a software.

**Quality in Use Metrics:** Quality in use metrics is only available when the final product is used in real conditions. Internal quality determines the external quality and external quality determines quality in use. This standard starts from the GE model for
describing software quality, presented in 1977 by McCall et al., [53] which is organized around three types of Quality Characteristics:

- Factors (To specify): These describe the external view of the software, as viewed by the users.

- Criteria (To build): These describe the internal view of the software, as seen by the developer.

- Metrics (To control): These are defined and used to provide a scale and method for measurement.

ISO/IEC 9126 distinguishes between a defect and nonconformity. Defect is the nonfulfilment of intended usage requirements, whereas nonconformity is the nonfulfilment of specified requirements. A similar distinction is made between validation and verification, known as V&V in the testing trade.

McCall model was used in 1992 as the basis for an international standard. ISO 9126 (ISO 1991) is called Software Product Evaluation.

2.1.2 Evaluation preparation

The second stage is that of evaluation preparation. It involves the selection of appropriate metrics, a rating level definition and the definition of assessment criteria. Metrics, in ISO 9126, has quantifiable measures mapped on to scales. The rating levels definition determines which ranges of values on those scales count as satisfactory or unsatisfactory. No general levels for rating are possible; they must be defined for each specific evaluation. The assessment criteria definition involves
preparing a procedure for summarising the results of the evaluation. It takes management criteria specific to a particular environment as input, which may influence the relative importance of different quality characteristics and sub-characteristics. It is specific to the particular evaluation. [54]

The purpose of the second stage is to prepare the basis for evaluation. This stage consists of three components:

- **Quality metrics selection**: The way in which quality characteristics have been defined does not allow conducting their direct measurement. There is a need to establish some metrics that can correlate to the characteristics of the software product. Every quantifiable feature of software and every quantifiable interaction of software with its environment that correlates with a characteristic can be established as a metric. Metrics can differ depending on the environment and the phases of the development process in which they are used. Metrics used in the development process should be correlated to the user perspective, because the metrics from the user's view are essential.

- **Rating levels definition**: Quantifiable features can be measured quantitatively using quality metrics. The result, the measured value, must be interpreted as a rated value, i.e. divided into ranges corresponding to the different degrees of satisfaction of the requirements. Since quality refers to given needs, no general levels for rating are possible. They must be defined for each specific evaluation.

- **Assessment criteria definition**: To assess the quality of the product, the results of evaluation of different characteristics must be summarized. The evaluator has to prepare a procedure for this, using, for instance, decision tables or weighted averages. The procedure can also include other aspects such
as time and cost that contribute to the assessment of quality of a software product.

2.1.3 Evaluation procedure

The final stage is the evaluation procedure. It is refined into three steps: measurement, rating and assessment. In measurement, the selected metrics are applied to the software product and values on the scales of the metrics are obtained. Subsequently, for each measured value, the rating level is determined. Assessment is the final step of the software evaluation process, where a set of rated levels are summarised. The result is a summary of the quality of the software product. The summarised quality is then compared with other aspects such as time and cost, and the final managerial decision is taken, based on the managerial criteria. [54,55]

This last step of the Evaluation Process Model is refined into three steps, namely measurement, rating and assessment.

- **Measurement**: For measurement, the selected metrics are applied to the software product. The result is values on the scales of the metrics.

- **Rating**: In the rating step, the rating level is determined for a measured value.

- **Assessment**: Assessment is the final step of the software evaluation process where a set of rated levels is summarized. The result is a statement of the quality of the software product. Then the summarized quality is compared with the other aspects such as time and cost. Finally, managerial decision will be made based on the managerial criteria. The result is a managerial decision on the acceptance or rejection, or on the release or no-release of the software product.
This international standard provides guidance for the practical implementation of software product evaluation when several parties need to understand, accept and trust evaluation results. It may be used to apply the concepts described in ISO/IEC 9126. It defines and describes the activities required to analyze evaluation requirements, to specify, design and perform evaluation actions and to conclude the evaluation of any kind of software product.

This standard can be used by:

- Testing laboratory evaluators - for providing software product evaluation services
- Software suppliers - for planning evaluation of their products, including evaluation to be carried out by independent testing services
- Software buyers - when requesting evaluation information from a supplier or testing service
- Software users - for evaluating products, including the use of test reports provided by testing laboratories
- Certification bodies - for establishing new schemes for software products.

The evaluation process is described in ISO 14598-5.2. Evaluation takes into account various documents that can be considered as a part of the software product (e.g. software requirements and design, test and validation reports, etc.) Evaluator should use a library of evaluation modules that define evaluation methods. The evaluation leads to the production by the evaluator of an evaluation report.

**Evaluation process characteristics**

The standard defines the main characteristics of the evaluation process (repeatability, reproducibility, impartiality, objectivity) and describes the specific steps and relationships describing its activities:
• **Analysis of evaluation requirements**: analysis of the requirements in order to identify the actual requirements for the evaluation

• **Evaluation specification**: produced on the basis of the evaluation requirements and the description of the software product provided

• **Evaluation design and definition of the evaluation plan**: on the basis of the evaluation specification, the components to be evaluated and the evaluation methods proposed by the evaluator

• **Execution of the evaluation plan**: it comprises of inspecting, modeling, measuring, and testing components of the product according to the evaluation plan. These actions can be performed using software tools (generally provided by the evaluator). The actions performed by the evaluator are recorded and the results obtained are put in the draft evaluation report

• **Evaluation conclusion**: it consists of the delivery of the evaluation report and the disposal of the product components evaluated by the evaluator.

**Evaluation process inputs**

The requester provides the following input to the evaluation process:

• an initial version of the evaluation requirements
• a description of the software product to be evaluated which identifies the components of the product which are submitted to the evaluation
• the product
The evaluator provides the following input to the evaluation process:

- pre-defined evaluation specifications
- evaluation methods
- evaluation tools

**Evaluation process outputs**

During the evaluation process, the following information is produced:

- **Evaluation requirements**: they describe the objectives of the evaluation; specially the critical requirements

- **Evaluation specification**: they define all the analysis and measurement to be performed and all components of the product

- **Evaluation plan**: they describe operational procedures needed to implement the evaluation specification; that is all the methods and tools to be used in the evaluation are described

- **Evaluation records**: they consist of the evaluation plan and details of the actions performed by the evaluator while executing the evaluation plan. The records are kept by the evaluator

- **Evaluation report**: they contain evaluation requirements, the evaluation specification, results from measurements and analysis performed and any other information necessary to be able to repeat or reproduce the evaluation; the evaluation report is delivered to the requester.
Figure 2.1. The Evaluation Process

2.2 EXISTING EVALUATION METHODS OF SOFTWARE PROCESS MODELS

In general, evaluation objectives include understanding the way work is carried out, exploring an area that is not well understood, describing the current state, predicting the future and explaining why things happen. The evaluation objectives in software engineering include understanding the software process and product. Defining, measuring and validating qualities of process and product. Evaluating and confirming the success and failures. Incorporating the information feedback for project control, evaluating technology and improving the software development. [42,45,46]
Various approaches to perform the software evaluation include the following:

- Methods for model definition
- Definition of measurements or metrics
- Methods for data capture
- Methods for analysis
- Managing validity treats
- Research methods designed for objective evaluation

Goals of software evaluation include, increasing the quality, making compliance with the budget and eventually the software process improvement. It includes the software engineering management by determining satisfaction of the requirements, reviewing and evaluating the performance. The software engineering process that defines or uses process assessment models and methods is also taken into account and it also defines or uses process and product measurement (that is, size, structure, quality). Another parameter incorporated is the tools and methods that include software process improvement, evaluates the effect of introduction or use, feature analysis and tools benchmarking.

2.2.1 Methods for model definition

There are many statistical tools for model validation, but the primary tool for most of the process modeling applications is **graphical residual analysis**. Different types of
plots of the residuals from a fitted model provide information on the adequacy of different aspects of the model. Numerical methods for model validation, such as the $R^2$ statistic, are also useful, but usually to a lesser degree than graphical methods.

Graphical methods have an advantage over numerical methods for model validation because they readily exhibit a broad range of complex aspects of the relationship between the model and the data. Numerical methods for model validation tend to focus on a particular aspect of the relationship between the model and the data and normally try to compress that information into a single description or test result. [47]

Normally, it takes more than one metric to understand, evaluate or control a software product, process, service or project. A Kiviat chart, also called a polar chart, radar chart or spider chart, can show a summary view of a set of metrics. Other statistical techniques are, Run Charts, Histograms, Pareto Analysis, Scatter Diagrams, Regression Analysis, Pie Charts, Radar/Kiviat charts, etc.

Some Applications of Statistics in Software Engineering are as follows: [49,92]

**Requirements**: Specify performance goals that can be measured statistically, e.g., no more than 50 total field faults and zero critical faults with 90% confidence.

**Design**: Pareto analysis to identify fault-prone modules. Use of design of experiments in making design decisions empirically.

**Coding**: Statistical control charts applied to inspections.

**Testing**: Coverage metrics provides attributes. Design of experiments useful in creating test suites. Statistical usage testing is based on specified operational profile. Reliability models can be applied.
Six Sigma provides continuous process improvement and attempts to reduce the natural variation in processes. Typically, Six Sigma programs use the “Seven Tools of Quality”. The Seven Tools of Quality are described as follows:

**Check Sheet**: To count occurrences of problems.

**Histogram**: To identify central tendencies and any skewing to one side or the other.

**Pareto Chart**: To identify the 20% of the modules which yield 80% of the issues.

**Cause and Effect Diagram**: For identifying assignable causes.

**Scatter Diagram**: For identifying correlation and suggesting causation.

**Control Chart**: For identifying processes that are out of control.

**Graph**: For visually displaying data, e.g., in a pie chart.

2.2.2 Definition of measurements or metrics

There are many different types of customers, who add complexity to the program because each customer may have different information requirements. Customers may include: [44]

- **Functional Management**: they are interested in applying control to the software development process, reducing the risks and maximizing returns on the investment
• **Project Management**: they are interested in being able to accurately predict and control the project size, effort, resources, budgets and the schedules. They control the projects and communicating facts to their management.

• **Software Engineers / Programmers**: these are people who actually work on the software development. They are interested in making informed decisions about their work and the work products. They are responsible for collecting a significant amount of data required for the metrics program.

• **Test Managers / Testers**: these are people who are responsible for performing the verification and validation activities. They are interested in finding as many new defects as possible in the time allocated for testing and in confirming that software works as specified. These people are also responsible for collecting a significant amount of required data.

• **Specialists**: these are individuals performing specialized functions (e.g., Marketing, Software Quality Assurance, Process Engineering, Software Configuration Management, Audits and Assessments, Customer Technical Assistance etc.). They are interested in quantitative information, upon which, they can base their decisions, findings and recommendations.

• **Customers / Users**: they are interested in on-time delivery of high quality software products and in reducing the over-all cost of ownership. According to **Watts Humphrey**, there are four major roles for software measurement: [43]

  - **Control**: Control type metrics are used to monitor our software processes, products and services and identify areas where corrective or management action is required.
- **Evaluate**: Evaluate type metrics are used in the decision-making process to study products, processes or services in order to establish baselines and to determine if established standards, goals and acceptance criteria are being met.

- **Understand**: As part of research studies, understand type metrics can be gathered to learn about software processes, products and services.

- **Predict**: Predict type metrics are used to estimate the values of base or derived measures in the future.

These different types of roles are used to explore and establish decision criteria for different types of measures. Measurement plays a critical role in effective and efficient software development, and provides the scientific basis for software engineering that makes it a true engineering discipline. The software quality engineering metrics and models include quality planning, process improvement and quality control, in-process quality management, product engineering (design and code complexity), reliability estimation and projection, object-oriented metrics, availability metrics, and analysis of customer satisfaction data.

2.2.3 Methods for data capture

Process modeling is one of the most important tasks that a software process development teams perform. Teams composed of domain experts and software engineers perform modeling which takes significant effort and time. We can use the data extracted from the actual events that took place to determine the process models. On applying, process discovery algorithms we can compare the results by the actual process and the process definitions. The discrepancies between the actual flow and the process definitions, also the weaknesses and strong aspects of the algorithms can be compared. [57]
Process data capture is the activity of obtaining information about an existing software process. Process analysis is the manipulation of that information for the purpose of problem identification. Capture and analysis are key elements in any strategy for software process improvement.

The aim of the process mining is to use process discovery to construct a process model as an abstract representation of event logs. The objective is to build a model (e.g., a Petri net) that can reproduce the logs under consideration, and does not allow different behaviours compared with those shown in the logs. The theory of regions can be used to transform a state-based model into a Petri net that exactly imitates the behavior given as an input. There are different Petri net based discovery algorithms from both the area of process mining and the theory of regions. The classical α-algorithm is unable to rediscover various nets, while the region-based approach, which can mine them correctly, is too complex. [58]

Process mining (PM) had emerged as a scientific discipline around 1990 when techniques like the Alpha algorithm were used to extract process models (usually represented as Petri nets) from event logs. Today, there are more than 100 process-mining algorithms that are capable of discovering process models that also include concurrency, e.g., genetic process discovery techniques, heuristic mining algorithms, region-based mining algorithms, and fuzzy mining algorithms. The field of process mining combines ideas, techniques, and methods from both the data mining field and the process modeling and process analysis fields. [60]

Process mining is a process management technique that allows the analysis of business processes based on event logs. The purpose is to extract knowledge from event logs recorded by an information system. Process mining aims at improving this by providing techniques and tools for discovering process, control, data,
organizational, and social structures from event logs. Process mining techniques are used when no formal description of the process can be obtained by other ways, or when the quality of an existing documentation is questionable.

There are three classes of process mining techniques. This classification is based on whether there is an *a priori model* and how it is used. In statistics, *a priori* knowledge is prior knowledge about a population rather than that estimated by recent observation. A *statistical population* is a set of entities concerning which statistical inferences are to be drawn, often based on a random sample taken from the population. *Population* is also used to refer to a set of potential measurements or values, including not only the cases actually observed but also those that are potentially observable.

- **Discovery**: In this class there is no, *a priori* model, i.e., based on an event log some model is constructed. For example, using the *alpha algorithm*, a process model can be discovered based on low-level events. There exist many techniques which automatically construct process models (e.g., in terms of a *Petri net*) based some event log. [62, 63, 64, 65, 66] Recently, process-mining research also started to target the other perspectives (e.g., data, resources, time, etc.).

- **Conformance**: In this class there is an *a priori* model. This model is compared with the event log and discrepancies between the log and the model are analyzed. Conformance checking may be used to detect deviations to enhance the model. For each choice the event log is consulted to see which information is typically available when the choice is made. Then classical data mining techniques are used to see which data elements influence the choice. As a result, a decision tree is generated for each choice in the process.
• **Extension**: There is an *a priori* model. This model is extended with a new aspect or perspective, that is, the aim is not to check conformance but to enhance the model. An example is the extension of a process model with performance data. [59, 61]

2.2.4 Methods for analysis

Aim of data analysis is to highlight useful information by inspecting, cleaning, transforming, and modeling data, suggesting conclusions, and supporting decision-making. Data analysis can be done by using different approaches, encompassing different techniques. Fundamental methods of statistical analysis are available from probability and statistical distributions, through basic concepts of statistical inference, to a collection of methods of analysis that are useful for scientific research.

Analyses of data from experiments, product testing, simulation, surveys, and statistical process and quality control must be appropriately performed for determining results and to draw conclusion. Statistical analysis and modeling involves the appropriate application of statistical analysis techniques, each requiring certain assumptions be met, in order to perform hypothesis tests, interpret the data, and reach to valid conclusions. To prove the reliability, results from experimentation or testing must be obtained following the established statistical procedures, including experimental design and the appropriate use of statistical analysis and modeling techniques. These results can then be reproduced, within sampling error, by repeating the experiment.

Statistical analysis and modeling requires careful selection of analytical techniques, verification of assumptions, and verification of data. Descriptive statistics, graphs, and relational plots of the data should be examined first, to evaluate the authenticity of the data, identify possible outliers and assumption violations, and form preliminary
ideas on variable relationships for modeling. The many different statistical analysis and modeling techniques have different goals and are appropriate for different types of data. [68]

Data analysis techniques include univariate analysis (such as, analysis of single-variable distributions), bivariate analysis, and multivariate analysis. [72] Data mining is data analysis technique that focuses on modeling and knowledge discovery for predictive rather than purely descriptive purposes. Data analysis is closely linked to data visualization and data dissemination. The term data analysis is sometimes used as a synonym for data modeling. [67,74,75]

Data Mining is an analytic process designed to explore data, usually large amounts of data related to business or market, in search of consistent patterns and systematic relationships between variables, and then to validate the findings by applying the detected patterns to new subsets of data. The ultimate goal of data mining is prediction. Predictive data mining is the most common type of data mining and has the most direct business applications. The process of data mining consists of three stages:

- the initial exploration
- model building or pattern identification with validation/verification
- deployment (i.e., the application of the model to new data in order to generate predictions) [69]

Data mining is the extraction of hidden predictive information from large databases. It is a powerful new technology that can help companies focus on the most important information in their data warehouses. Data mining tools predict future trends and behaviors, allowing businesses to make proactive, knowledge-driven decisions. The
automated, prospective analyses offered by data mining is better than the analyses provided by retrospective tools of decision support systems.

Most companies already collect and refine massive quantities of data. Data mining techniques can be implemented rapidly on existing software and hardware platforms to enhance the value of existing information resources, and can be integrated with new products and systems as they are brought on-line. When implemented on high performance client/server or parallel processing computers, data mining tools can analyze massive databases to answer questions like, which clients are most likely to respond to the next promotional mailing, and why.

The most commonly used techniques in data mining are:

- **Artificial neural networks**: Non-linear predictive models that learn through training and resemble biological neural networks in structure

- **Decision trees**: Tree-shaped structures that represent sets of decisions. These decisions generate rules for the classification of a dataset. Specific decision tree methods include Classification and Regression Trees (CART) and Chi Square Automatic Interaction Detection (CHAI)

- **Genetic algorithms**: Optimization techniques that use processes such as genetic combination, mutation, and natural selection in a design based on the concepts of evolution

- **Nearest neighbor method**: A technique that classifies each record in a dataset based on a combination of the classes of the k record(s) most similar to it in a historical dataset (where k ≥ 1). Sometimes called the k-nearest neighbor technique
- **Rule induction**: The extraction of useful if-then rules from data based on statistical significance [70]

Data modeling in software engineering is the process of creating a data model by applying formal data model descriptions using data modeling techniques. It is a method used to define and analyze data requirements needed to support the business processes of an organization. The data requirements are recorded as a conceptual data model with associated data definitions. Actual implementation of the conceptual model is called a logical data model. To implement one conceptual data model may require multiple logical data models. Data modeling defines not just data elements, but their structures and relationships between them. [72] Data modeling techniques and methodologies are used to model data in a standard, consistent, predictable manner in order to manage it as a resource. The use of data modeling standards is strongly recommended for all projects requiring a standard means of defining and analyzing data within an organization, e.g., using data modeling:

- to manage data as a resource
- for the integration of information systems
- for designing databases/data warehouses (aka data repositories)

Data modeling may be performed during various types of projects and in multiple phases of projects. Data models are progressive and there is no final data model for a business or application. [71]
2.2.5 Managing validity treats

Verification and Validation (V&V) is the process of checking that a software system meets the specifications and that it fulfils the intended purpose. Thereby ensuring that the program being developed meets its specification and delivers the functionality as expected by the customer.Normally it is the part of the software testing process of a project. It is also known as software quality control.

Validation is intended to show that the software produced is as per the demand of the customer. It checks that the product design satisfies the intended usage (it is a high-level checking, indicating that we built the right product). It is done through dynamic testing. It ensures that the product actually meets the user's requirement, and that the
specifications were correct, while verification is ensuring that the product has been built according to the requirements and design specifications. It also assures that we built the right thing. Validation confirms that the product will fulfill its intended use.

Boehm (Boehm, 1979) expressed the difference between validation and verification:
Validation: Are we building the right product?
Verification: Are we building the product right?

According to the Capability Maturity Model (CMMI-SW v1.1),

Verification: It is the process of evaluating software to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase. [IEEE-STD-610].

Validation: It is the process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements. [IEEE-STD-610]

From testing perspective:

- Fault implies wrong or missing function in the code.
- Failure implies the manifestation of a fault during execution.
- Malfunction implies according to its specification the system does not meet its specified functionality.

Within the modeling and simulation community, the definitions of validation, verification and accreditation are similar:
• Validation is the process of determining the degree to which a model, simulation, or federation of models and simulations, and their associated data are accurate representations of the real world from the perspective of the intended use(s)

• Accreditation is the formal certification that a model or simulation is acceptable to be used for a specific purpose

• Verification is the process of determining that a computer model, simulation, or federation of models and simulations implementations and their associated data accurately represents the developer's conceptual description and specifications [77]

Verification is a Quality control process that is used to evaluate whether or not a product, service, or system complies with regulations, specifications, or conditions imposed at the start of a development phase. Verification can be in development, scale-up, or production. This is often an internal process.

Validation is a Quality assurance process of establishing evidence that provides a high degree of assurance that a product, service, or system accomplishes its intended requirements. This often involves acceptance of fitness for purpose with end users and other product stakeholders. [79]

In software engineering, the syntactic methods are techniques for developing correct software programs. These techniques detect and prevent certain kinds of defects (bugs) by examining the structure of the code being produced at its syntactic level rather than at the semantic level.
When formal methods prove to be too expensive with the costs of modeling, execution and development, than the benefits gained by preventing the possible failures, then we use an abstract dependency graph, which is created from the system under consideration. An abstract dependency graph is a directed graph, a graph of vertices connected by one-way edges. The vertices and edges of the graph represent the inputs and outputs of functions or components of the system. By inspecting the created abstract dependency graph, the developer can detect syntactic anomalies in the system. Anomalies are not always defects; but they normally provide clues to finding defects in a system. Therefore, anomalies in a system help the developer in finding defects. [76]

2.2.6 Research methods designed for objective evaluation

For any research it is necessary to evaluate the accuracy of the study with regard to the applied methods of data gathering and the analysis of the material. According to Kirk & Miller, to make sure that the quality of the research is high, the research design is carefully planned and followed along the study. To make the quality of the research available for the reader to evaluate, each phase of the research must be as thoroughly reported as possible throughout the report [80]. In addition to this, it is also important for the researcher to critically evaluate the quality of the research as well. This kind of evaluation can be conducted using construct validity, internal validity, external validity and reliability as per Linconln & Cuba 1985, Tsoukas 1989, Yin 1989. [82,83]

Quantitative research is based upon formulating the research hypotheses and verifying them empirically on a specific set of data. This method is deductive and particularistic. Scientific hypotheses are value-free, that is, the researcher's own values, biases, and subjective preferences have no place in the quantitative approach.
Researchers can view the communication process as concrete and tangible and can analyze it without contacting actual people involved in communication.

The strengths of the quantitative method include:

- Stating the research problem in very specific and set terms
- Clearly and precisely specifying both the independent and the dependent variables under investigation
- Following firmly the original set of research goals, arriving at more objective conclusions, testing hypothesis, determining the issues of causality
- Achieving high levels of reliability of gathered data due to controlled observations, laboratory experiments, mass surveys, or other form of research manipulations
- Eliminating or minimizing subjectivity of judgment
- Allowing for longitudinal measures of subsequent performance of research subjects

The weaknesses of the quantitative method include:

- Failure to provide the researcher with information on the context of the situation where the studied phenomenon occurs
- Inability to control the environment where the respondents provide the answers to the questions in the survey
- Limited outcomes to only those outlined in the original research proposal due to closed type questions and the structured format
- Not encouraging the evolving and continuous investigation of a research phenomenon
Qualitative research shares the theoretical assumptions of the interpretative paradigm, which is based on the notion that social reality is created and sustained through the subjective experience of people involved in communication. Qualitative researchers are concerned in their research with attempting to accurately describe, decode, and interpret the meanings of phenomena occurring in their normal social contexts. The researchers operating within the framework of the interpretative paradigm are focused on investigating the complexity, authenticity, contextualization, shared subjectivity of the researcher and the researched, and minimization of illusion.

The strengths of the qualitative method include:

- Obtaining a more realistic feel of the world that cannot be experienced in the numerical data and statistical analysis used in quantitative research
- Flexible ways to perform data collection, subsequent analysis, and interpretation of collected information
- Provide a holistic view of the phenomena under investigation
- Ability to interact with the research subjects in their own language and on their own terms
- Descriptive capability based on primary and unstructured data

The weaknesses of the qualitative method include:

- Departing from the original objectives of the research in response to the changing nature of the context
- Arriving to different conclusions based on the same information depending on the personal characteristics of the researcher
- Inability to investigate causality between different research phenomena
• Difficulty in explaining the difference in the quality and quantity of information obtained from different respondents and arriving at different, non-consistent conclusions
• Requiring a high level of experience from the researcher to obtain the targeted information from the respondent
• Lacking consistency and reliability because the researcher can employ different probing techniques and the respondent can choose to tell some particular stories and ignore others. [81]

**Construct validity** means establishing correct operational measures for the concepts under study. The assessment involves subjective judgement, as construct validity is a conceptual and not an empirical issue, according to Wigdor & Garner. This is tackled by using multiple sources of evidence while collecting the data. In this study, various types of empirical data are gathered. Construct validity was advanced using data triangulation as per Denzin by using different types of empirical material (e.g. interviews, workshop transcripts, company documents including suggested procedures) and contrasting them with each other when categorizing the various types of material. Main stakeholders can review the case study reports to increase the construct validity.

**Yin** states that **internal validity** is more essential when establishing explanations and causal relations, and it is not much important in case of descriptive and explanatory research. It aims at creating the understanding of the research phenomena. Internal validity is a measure that ensures that a researcher’s experiment design closely follows the principle of cause and effect. Internal validity only shows that we have evidence to suggest that a program or study had some effect on the observations and results. Internal validity does not mention about whether the results were what we expected, or whether generalization is possible.
**External validity** is about establishing the domain to which a study’s findings can be generalized as per *Yin 1989*. External validity also indicates the compatibility between theoretical conclusions and empirical data according to *Grönfors*. This refers specially to the use of multiple case studies when the study aims at making generalizations of the research findings that can be applied to a wider group of situations.

The main criterion of external validity is the process of generalization, and whether results obtained from a small group, can be extended to make predictions about the entire population. In 1966, Campbell and Stanley proposed the commonly accepted definition of external validity, “External validity asks the question of generalizability: To what populations, settings, treatment variables and measurement variables can this effect be generalized?”

Case studies do not aim towards making generalizations, rather they relate to comparative and correlation studies. In case studies, the point is more on making particularizations, which mean that we take a certain case and study it thoroughly. It is concerned as to how it is different or similar to others and assess that what it is and what it does. Thus, there is emphasis on uniqueness, and this uniqueness implies the differences that the case has compared to others as per *Stake*. With respect to the external validity, this research draws analytical generalizations. In addition to this, some generalizations especially concerning the theoretical elaboration of the concept and its various perspectives are drawn.

**Reliability**, refers to the concept of repeating the case study with the same results, according to *Yin*. It is done by documenting and reporting the empirical research process thoroughly. Aim is to describe the progression of the research process precisely to increase the reliability of the research.
Reliability is the extent to which an experiment, test, or any measuring procedure yields the same result on repeated trials. Without the agreement of independent observers able to replicate research procedures, or the ability to use research tools and procedures that yield consistent measurements, researchers would be unable to satisfactorily draw conclusions, formulate theories, or make claims about the generalizability of their research. In addition to its important role in research, reliability is critical for many parts of our lives, including manufacturing, medicine, and sports. Reliability is such an important concept that it has been defined in terms of its application to a wide range of activities. For researchers, four key types of reliability are: [83,84,85,86]

- Equivalency Reliability
- Stability Reliability
- Internal Consistency
- Interrater Reliability

2.3 SOLVING A MODEL

A formal model of a system is a mathematical model of it, at some chosen level of abstraction. Its purpose is to permit precise understanding, specification, and analysis of the system.

Formal methods are mathematically based techniques for the specification, development and verification of software and hardware systems. The use of formal methods for software and hardware design helps in performing appropriate mathematical analysis and contributes to the reliability and robustness of a design.
The high cost of using formal methods means that they are usually used in the development of high-integrity systems, where safety or security is of utmost importance.

A Formal Model with mathematical foundation can be mathematically solved, that is, quantitative results that characterize the behavior of the real system are calculated. Reasons for replacing the observation of the real system by a mathematical calculation of the system behavior include the following:

- the behavior of the real system may be too fast or slow to allow observation
- the behavior being examined might destroy or seriously damage the irreplaceable or expensive real system or its environment
- current technology does not allow direct observation of the behavior, but only the result is seen
- the real system does not exist

Characteristics of high quality software include the following attributes:

- high quality software is intuitive and easy to use
- it is efficient
- high quality software always produce the results and does not crash

The needs for precision in the specification of software are:

Software components can be reused is a primary motivation of object-oriented programming. To reuse a previously written software component, a software engineer
must have a precise description of its behavior, as a minor misconception of the function of a component that is unapparent at the beginning may cause serious errors that are difficult and expensive to correct later in the process.

Typical software development phases:

Software development models commonly subdivide the process into phases similar to the following:

- requirements analysis: determine user needs
- specification: describe precisely the role of the software
- design: devise overall organization of the software
- implementation: formulate the algorithms and programs
- verification: certify that the programs meet the specification
- maintenance: perform ongoing changes and corrections after software is in use

The role of formal methods

Formal methods are intended to systematize and introduce thoroughness into all the phases of software development. This helps us to avoid overlooking the critical issues, to provide a standard means to record various assumptions and decisions, and form a basis for consistency among the related activities. By providing precise and explicit description mechanisms, formal methods facilitate the understanding required to combine the various phases of software development.

There is a visible tendency to merge discrete mathematics and formal methods for software engineering (Denvir, Ince, and Woodcock & Loomes). But applying discrete mathematics to software engineering does not assure relevant formal
methods. The prime concern of software engineering is the creation of high quality software systems. [87,88,89]

Leveson states that formal methods are: [90]

- application of discrete mathematics to software engineering
- involves modeling and analysis
- with an underlying mathematically-precise notation

Wing states that formal methods are: [90]

- Use of a formal language
  Which are set of strings over some well-defined alphabet, with rules for distinguishing which strings belong to the language
- Formal reasoning about formulae in the language
  For example, formal proofs use axioms and proof rules to demonstrate that some formula is in the language

2.3.1 Analytical methods

To solve a model analytically, a complete mathematical description of the model is required. According to Hiller and Liebermann, numerous modeling formalisms are available to support the mathematical description, for example Markov Chains or Queuing Nets. Based on state probabilities the characteristic values for models can be exactly calculated or approximated without involving large amounts of memory or computation power.

Brendel and Schäfer state that for analytical solution, a Conceptual Model must be transformed into a Formal Model, for which well-defined solution methods are available. The limited expressiveness of the formalism and the restricted choice of
stochastic input models may force the modeler to abstract and idealize in a manner that prohibits the inclusion of relevant aspects of the real system in the Formal Model.[91]

Every type of analysis has a specific purpose, and they are applied to the project as required. For example, a support technique called complexity analysis is useful for determining whether the design or implementation is too complex to develop correctly, to test, or to maintain. The results of complexity analysis may be used in developing test cases. Defect-finding techniques, for instance control flow analysis, may be used to support another activity. For software with algorithms, algorithmic analysis is important, mainly when an incorrect algorithm could cause a disastrous result.

A software engineer generally applies analytical techniques to different parts of the product. Some techniques are tool-driven while others are manual. Some techniques find the defects directly, but are usually used to support other techniques. Some also include various assessments as part of overall quality analysis. Examples of such techniques include complexity analysis, control flow analysis, and algorithmic analysis.

More formal types of analytical techniques are known as formal methods. They are used to verify software requirements and designs. And have mostly been used in the verification of crucial parts of critical systems, such as specific security and safety requirements.[93,94]

2.3.2 Computer based simulation

If an analytical solution of the model is not feasible due to the limitations of the modeling formalism or model complexity, or not desirable, one may approximate the
behavior of the real system by executing the model over time, and subsequently or interactively draw conclusions about reality from the observed dynamic behavior of the model.

Computer simulation is the technique of representing the real world by a computer program. Simulation should imitate the internal processes and not simply the results of the thing being simulated. Here simulation is defined in analogy to IEEE 610.3 1989 [95]

Thomas and Milligan state that computer simulation has the following two key features:

1. There is a computer model of a real or theoretical system that contains information on how the system behaves
2. Experimentation can take place, that is, changing the input to the model affects the output

As a numerical model of a system, presented for a learner to manipulate and explore, simulations can provide a rich learning experience for the student. These model can be a powerful resource for teaching: providing access to environments which may otherwise be too dangerous, or impractical due to size or time constraints; and facilitating visualisation of dynamic or complex behavior. [96]

There are numerous distinct simulation methods and techniques available. The selection of a particular simulation method depends on the type of the model and the intended use of the simulation results.

Simulation is the execution of a model that behaves similar to the real system when provided with a set of controlled inputs over time. For computer-based simulation, a
computer calculates the behavior of the symbolic model. Model behavior reflects the
dynamic change of the model output over time, as a function of model input, and,
where applicable, the internal model state or random elements.

The concepts considered for computer-based simulation are listed as below:

- **Quasi-continuous vs. discrete simulation**: Continuous simulation requires a
continuous model (with a continuous state space) and continuous advancement
of the time. If the model’s state space is discrete, or the time advances in
discrete time steps, the simulation is called *discrete simulation*. Theoretically,
on a digital computer no continuous simulation is possible, but if the steps of
time advancement and the discretization of the state space are sufficiently
small, continuous simulation may be approximated (*quasicontinuous*). We
can have both, quasi-continuous and discrete advancement of time in one
simulation together.

- **Time-stepped vs. discrete event simulation**: If in a simulation, the time
advances in time intervals i.e. steps of equal length, this simulation is called
*time-stepped*. Time-stepped simulation is scaled to the real time, and supports
interaction between a user, physical models, or other real systems and the
model. In *discrete event simulation*, simulation time jumps from event to
event. Events are typically triggered by conditions and may be scheduled and
caused from outside the model or take place internally as a consequence of
interactions among the objects or submodels. Time advancement may be
either time-stepped or based on event occurrence for separate submodels
contained in a model.

A discrete event simulation (DES) manages events in time. Logic-test and
fault-tree simulations are of this type. In this type of simulation, the simulator
maintains a queue of events sorted by the simulated time they should occur.
The simulator reads the queue and triggers new events as each event is processed. It is not important to execute the simulation in real time. In this simulation it is more important to be able to access the data produced by the simulation, to discover logic defects in the design, or the sequence of events.

- **Deterministic vs. stochastic simulation**: This simulation depends on whether a stochastic or deterministic model is used. Whenever a deterministic model is executed, the simulation results depend entirely on the initial state of the model and the input data. Random decisions that must be made when executing a stochastic model rely on the random numbers that are drawn during each simulation run. Depending on the random numbers, simulation results in stochastic simulation can vary with each run, even if the input data does not change. Stochastic models use random number generators to model chance or random events. To get statistically significant results, experimentation with a stochastic model must be repeated with a sufficient number of times. [98]