**Garvin’s Quality Dimensions**

* **David Garvin suggests** that quality should be considered by taking a multidimensional viewpoint
* Although Garvin’s **eight dimensions** of quality developed for0020when software quality is considered:

**Performance quality**.

* Software deliver all content, functions, and features that are specified as part of the requirements from end user

**Feature quality**.

* Does the software provide features that as per wish of user

**Reliability.**

* Does the software deliver all features and capability without Failure?
* Is it available when it is needed? .
* Does it deliver functionality that error-free?

**Conformance.**

* Does the software conform to local and external software standards that are relevant to the application?
* Does it conform to de fact design and coding conventions?

**Durability**.

* Can the software be maintained or corrected without unintended side effects?
* Will changes cause the error rate or reliability to degrade with time?

**Serviceability**.

* Can the software be maintained (changed) or corrected (debugged) in an acceptably short time period?
* Can support staff acquire all information they need to make changes or correct defects?

**Aesthetics**.

* an aesthetic entity has a certain **elegance, a unique flow**, and an obvious “presence” that are hard to quantify

**Perception**.

* In some situations, you have a set of prejudices that will influence

your perception of quality.

**For example**, if you are introduced to a software

product that was built by a vendor who has produced **poor quality in**

**the past**, your initial perception of the current software

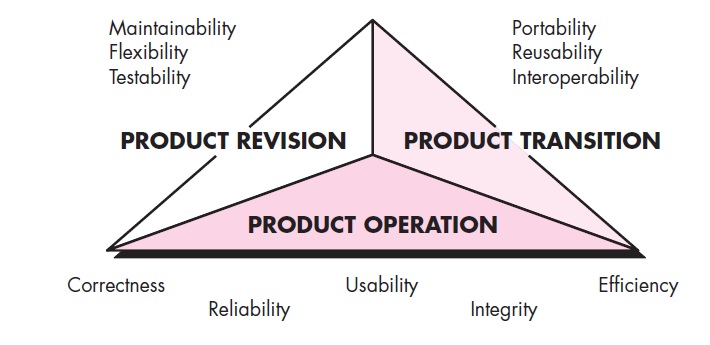
product quality might be negatively.

Similarly, if a vendor has an excellent reputation, you may perceive quality, even when it does not really exist

**McCall’s Quality Factors**

**McCall and his colleagues provide the following Quality Factors:**

* ***Correctness****.* The extent to which a program satisfies its specification and fulfills thecustomer’s mission objectives.
* ***Reliability.***The extent to which a program can be expected to perform its intended function with required precision
* ***Efficiency****.* The amount of computing resources and code required by a program to perform its function.
* ***Integrity.***Extent to which access to software or data by unauthorized persons can be controlled.
* ***Usability****.* Effort required to learn, operate, prepare input for, and interpret output of a Program
* ***Maintainability****.* Effort required to locate and fix an error in a program. .
* ***Flexibility****.* Effort required to modify an operational program.
* ***Testability****.* Effort required to test a program to ensure that it performs its intended function.
* ***Portability****.* Effort required to transfer the program from one hardware and/or software system environment to another.
* ***Reusability****.* Extent to which a program [or parts of a program] can be reused in other applications—related to the packaging and scope of the functions that the program performs.
* ***Interoperability.***Effort required to couple one system to another.



**ISO 9126 QUALITY FACTORS**

The ISO 9126 standard was developed in an attempt to identify the key quality attributes for computer software.

**The standard identifies six key quality attributes**:

**Functionality.** The degree to which the software satisfies stated needs as

indicated by the following sub attributes: suitability, accuracy, interoperability, compliance, and security.

**Reliability.** The amount of time that the software is available for use as indicated by the following sub attributes: maturity, fault tolerance, recoverability

**Usability.** The degree to which the software is easy to use as indicated by

the following subattributes: understandability, learnability, operability.

**Efficiency.** The degree to which the software makes optimal use of system

resources as indicated by the following subattributes: time behavior, resource

behavior.

**Maintainability.** The ease with which repair may be made to the software as

indicated by the following subattributes: analyzability, changeability, stability, testability.

**Portability.** The ease with which the software can be transposed from one

environment to another as indicated by the following subattributes: adaptability, installability, conformance, replaceability.

**ACHIEVING SOFTWARE QUALITY**

Software quality doesn’t just appear. It is the result of good project management and solid software engineering practice.

Management and practice are applied within the context of four broad activities

**Software Engineering Methods**

* To build high-quality software**, need to understand the problem to be**

**solved.**

* Software engineering methods can lead to a reasonably complete understanding of the problem and a comprehensive design that establishes a solid foundation for the construction activity.
* By applying methods probability of getting high-quality software will increase

**Project Management Techniques**

* There is vital impact on quality of software because of poor management decisions
* Impact of poor Project management techniques involves
* **Impact on delivery dates,**
* **Irregular schedule**
* **risk planning will be affected**
* The project plan should include explicit techniques for quality and

change management.

**Quality Control**

* Quality control encompasses a set of software engineering actions that help to ensure that each work product meets its quality goals.
* **Models are reviewed to ensure that they are complete and consistent.**
* **Code may be inspected in order to uncover and correct errors before testing commences**.
* A **series of testing steps is applied to uncover errors in processing logic**, data manipulation, and interface communication.
* A combination of measurement and feedback process when any of these work products fail to meet quality goals.

**Quality Assurance**

* Quality assurance establishes the infrastructure that supports useful various software engineering methods, rational project management, and quality control actions—
* **All related methods are pivotal if it intend to build high-quality software**.
* **The quality assurance consists of a set of auditing and reporting functions that assess the effectiveness and completeness of quality control actions**.
* **The goal of quality assurance is to provide management and technical staff with the data necessary to be informed about product quality**,
* if the data provided through quality assurance identifies problems, it is management’s responsibility to address the problems and apply the necessary resources to resolve quality issues

**Software Quality Assurance**

BACKGROUND ISSUES

* Definition- Software quality assurance is a “planned and systematic pattern of actions that are required to ensure high quality in software”
* **Q**uality **c**ontrol **and assurance are essential activities** for any business that produces products
* **Prior to the twentieth century, quality control was the**

**sole responsibility of the craftsperson who built a product. As time passed and mass production, quality control became an activity performed by people other than the ones who built the product**.

* The **first formal quality assurance and control function** was introduced **at Bell Labs in 1916** and spread rapidly throughout world.
* During the 1940s, more formal approaches to quality control were suggested. These relied on measurement and continuous process improvement as key elements of quality management.
* **Today, every company has mechanisms to ensure quality in its products**.
* The history of quality assurance in software development parallels the history of quality in hardware manufacturing.
* **During the early days of computing quality was the sole responsibility of the programmer**.
* Standards for quality assurance for software were introduced in military contract software development during 1970s and have spread rapidly into software development in world

**ELEMENTS OF SOFTWARE QUALITY ASSURANCE**

**Standards.**

* The IEEE, ISO, and other standards organizations have produced

a broad array of software engineering standards and related documents.

* **Standards may be adopted voluntarily** by a software engineering

organization or imposed by the customer or other stakeholders.

* **The job of SQA is to ensure that standards that have been adopted are followed and that all work products conform to them**.

**Reviews and audits.**

* Technical reviews are a quality control activity

performed by software engineers. Their **intent is to uncover errors**.

* Audits are a type of review performed by SQA personnel with the intent of ensuring that quality guidelines are being followed for software engineering work.

**Testing.**

* Software **testing, is A quality control function that has primary goal to find errors**.
* **The job of SQA is to ensure that testing is properly planned and efficiently conducted so that it has the highest likelihood of achieving its primary goal**.

**Error/defect collection and analysis.**

* The only way to improve is to measure how you’re doing.
* **SQA collects and analyzes error and defect data to better understand how errors are introduced and what software engineering activities are best suited to eliminating them**.

**Change management.**

* Change is one of the **most disruptive** aspects of any software project.
* If it is **not properly managed**, change can **lead to confusion**,

and confusion almost always leads to poor quality.

**Education.**

* **Every software organization** wants to improve its software

engineering practices.

* A key contributor to improvement is education of software

engineers, their managers, and other stakeholders.

* The **SQA organization takes the lead in software process improvement** and is a key proponent and sponsor of educational programs.

**Vendor management.**

* Three categories of software are acquired from external software vendors—
* ***shrink-wrapped packages***,
* a ***tailored shell***that provides a basic skeletal structure that is custom tailored to the needs of a purchaser, and
* ***contracted software***that is custom designed and constructed from specifications provided by the customer organization.
* The job of the SQA organization is to ensure that high-quality software results by suggesting specific quality practices that the vendor should follow when possible, and incorporating quality mandates as part of any contract with an external vendor.

**Security management.**

* With the increase in **cyber crime** and new government regulations regarding privacy,
* every software organization should institute policies that protect data at all levels, establish firewall protection for WebApps, and ensure that software has not been tampered with internally.
* **SQA ensures that appropriate process** and technology are used **to achieve software security.**

**Safety.**

* SQA may be responsible for assessing the impact of software failure and for initiating those steps required **to reduce risk**.

**Risk management.**

* Although the analysis and mitigation of risk is the concern of software engineers,
* The SQA organization ensures that risk management activities are properly conducted and that risk-related contingency plans have been established.

**SQA TASKS, GOALS, AND METRICS**

SQA Tasks

**SQA Tasks** **prepares an SQA plan for a project.**

* The plan is developed as part of project planning and is reviewed by all stakeholders.
* Quality assurance actions performed by the software engineering team and the SQA group are governed by the plan.
* The plan identifies
* evaluations to be performed,
* audits and reviews to be conducted,
* standards that are applicable to the project,
* procedures for error reporting and tracking,
* work products that are produced by the SQA group, and
* feedback that will be provided to the software team.

**Participates in the development of the project’s software process**

**description.** The software team selects a process for the work to be

performed. The SQA group reviews the process description for compliance

with organizational policy, internal software standards, externally

imposed standards (e.g., ISO-9001), and other parts of the software

project plan

**Reviews software engineering activities to verify compliance with the**

**defined software process.** The SQA group identifies, documents, and tracks

deviations from the process and verifies that corrections have been made.

**Audits designated software work products to verify compliance with**

**those defined as part of the software process.** The SQA group reviews

selected work products; identifies, documents, and tracks deviations; verifies

that corrections have been made; and periodically reports the results of its

work to the project manager.

**Ensures that deviations in software work and work products are**

**documented and handled according to a documented procedure.**

Deviations may be encountered in the project plan, process description,

applicable standards, or software engineering work products.

**Records any noncompliance and reports to senior management.**

Noncompliance items are tracked until they are resolved.

In addition to these actions, the SQA group coordinates the control and management

SQA goals and metrics

**Requirements quality.**

* The **correctness, completeness, and consistency** have a influence on the quality of work products that follow.
* SQA must ensure that the software team has properly reviewed the requirements model to achieve a high level of quality.

**Design quality.**

* Every element of the design model should be assessed by the software team to ensure that it exhibits high quality
* SQA looks for attributes of the design that are indicators of quality.

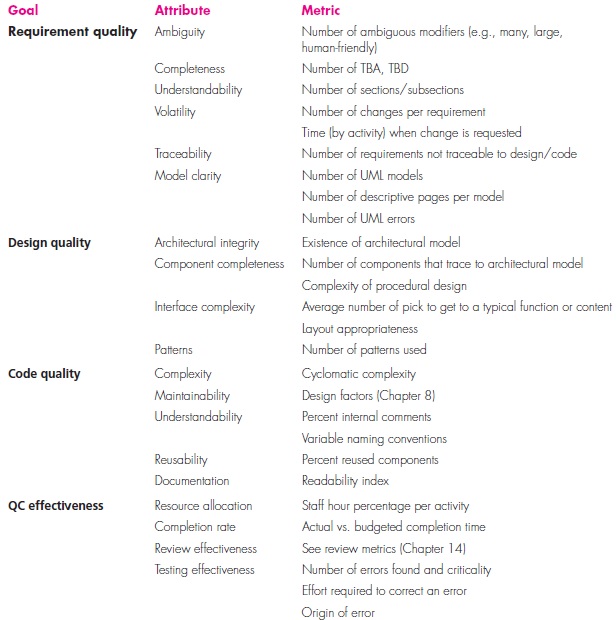
**Code quality.**

* Source code and related work products must conform local coding standards and exhibit characteristics that will facilitate maintainability.
* SQA should isolate those attributes that allow a reasonable analysis of the quality of code.

**Quality control effectiveness.**

* A software team should apply limited resources in a way that has the highest likelihood of achieving a high-quality result.
* SQA analyzes the allocation of resources for reviews and testing to

assess whether they are being allocated in the most effective manner.

****

**FORMAL APPROACHES TO SQA**

* software quality is everyone’s job and that it can be achieved through
* competent software engineering practice
* application of technical reviews,
* a multi-tiered testing strategy,
* better control of software work products and the changes made to them, and
* the application of accepted software engineering standards.
* Software Quality can be defined in terms of quality attributes and measured using a variety of indices and metrics.
* Over the past three decades, a small, segment of the software engineering community demanding more formal approach to software quality assurance
* A computer program is a mathematical object. A rigorous syntax and semantics can be defined for every programming language, and a

rigorous approach to the specification of software requirements is

available.

* If the requirements model and the programming language can be

represented in a rigorous manner, it should be possible to apply mathematic proof of correctness to demonstrate that a program conforms exactly to its specifications.

* Attempts to prove programs correct are not new. Dijkstra and Linger,

Mills, and Witt [Lin79], advocated proofs of program correctness and

tied these to the use of structured programming concepts

**STATISTICAL SOFTWARE QUALITY ASSURANCE**

* Statistical quality assurance reflects a **growing trend** throughout industry to become more quantitative about quality.
* Software, statistical quality assurance implies the following steps:
* Information about software errors and defects is collected and categorized.
* An attempt is made to trace each error and defect
* Using the Pareto principle (80 percent of the defects can be traced to 20 percent), isolate the 20 percent (the *vital few*).
* Once the vital few causes have been identified, move to correct the problems that have caused the errors and defects.

A Generic Example

* To illustrate the use of statistical methods for software engineering work, assume that a software engineering organization collects information on errors and defects for a period of one year.
* Some of the errors are uncovered as software is being developed.

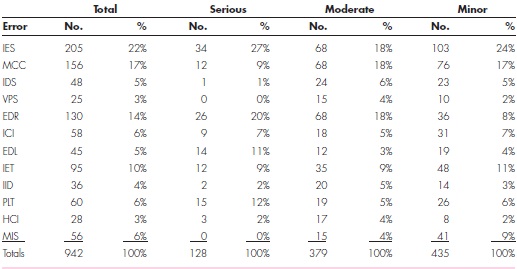
Others (defects) are encountered after the software has been released to

Its end users.

Although many different problems are uncovered, tracked

the following causes:

* Incomplete or erroneous specifications (IES)
* Misinterpretation of customer communication (MCC)
* Intentional deviation from specifications (IDS)
* Violation of programming standards (VPS)
* Error in data representation (EDR)
* Inconsistent component interface (ICI)
* Error in design logic (EDL)
* Incomplete or erroneous testing (IET)
* Inaccurate or incomplete documentation (IID)
* Error in programming language translation of design (PLT)
* Ambiguous or inconsistent human/computer interface (HCI)
* Miscellaneous (MIS)

****

Six Sigma for Software Engineering

* *Six Sigma* is the most widely used strategy for statistical quality assurance in industry
* Six Sigma strategy
* is a rigorous and disciplined methodology
* uses data and statistical analysis to measure and improve a company’s operational performance
* company’s operational performance measures by identifying and eliminating defects’ in manufacturing and service-related processes
* The term Six Sigma is derived from six standard deviations—

defects per million occurrences

* Used for extremely high quality

**The Six Sigma methodology defines three core steps**:

* ***Define***customer requirements and project goals via well defined

methods of customer communication.

* ***Measure***the existing process and its output to determine current

quality performance (collect defect metrics).

* ***Analyze***defect metrics and determine the vital few causes.

**If an existing software process is in place, but improvement is required, Six Sigma suggests two additional steps:**

* ***Improve***the process by eliminating the root causes of defects.
* ***Control***the process to ensure that future work does not reintroduce the

causes of defects.

These core and additional steps are sometimes referred to as the **DMAIC** method. (define, measure, analyze, improve, and control)

**If an organization is developing a software process (rather than improving an existing process), the core steps are augmented as follows:**

* ***Design***the process to avoid the root causes of defects and to meet customer requirements.
* ***Verify***that the process model will, in fact, avoid defects and meet customer requirements.

This variation is sometimes called the **DMADV** (define, measure, analyze, design, and verify) method.

**SOFTWARE RELIABILITY**

* *Software reliability* is defined in statistical terms as “the probability of failure-free operation of a computer program in a specified environment for a specified time”
* **One failure** can be corrected within seconds, while another requires weeks or even months to correct.
* the correction of one failure may in fact result in the introduction of other errors that ultimately result in other failures.

Measures of Reliability and Availability

* Early work in software reliability deals with hardware reliability theory to the prediction of software reliability.
* Most hardware-related reliability models are predicated on failure due to wear rather than failure due to design defects.
* In hardware, failures due to physical wear (the effects of temperature,

corrosion, shock) are more likely than a design-related failure. the

opposite is true for software

* All software failures because of design or implementation problems; but wear does not enter into the picture.
* If we consider a computer-based system, a simple measure of reliability is
* **MTBF : *meantime- between-failure***
* **MTTF :*mean-time-to-failure***
* **MTTR:*mean-time-to repair*,**
* MTBF is a far more useful measure than other quality-related software metrics
* an end user is concerned with failures, not with the total defect count.
* MTBF can be problematic for two reasons:

1. it projects a time span between failures, but does not provide

us with a projected failure rate, and

1. MTBF can be misinterpreted to mean average life span but

this is *not* what it implies.

* An alternative measure of reliability is ***failures-in-time* (FIT)—**a statistical measure of how many failures a component will have over one billion hours of operation.

EX: 1 FIT is equivalent to one failure in every billion hours of

operation.

* ***Software availability***a reliability measure the probability that a program is operating according to requirements

Ex: Availability \_ \_ 100%

Software Safety

* *Software safety* is a **SQA activity, focuses on the identification and assessment of hazard**s that may affect software negatively and cause an entire system to fail.
* If hazards can be identified early in the software process, that will either eliminate or control potential hazards.
* A **modeling** and **analysis process** is conducted as part of software safety.
* Initially, hazards are identified and categorized by criticality and risk.
* Once these system-level hazards are identified, there is starting to assign severity and probability of occurrence.
* To be effective, software must be analyzed in the context of the entire
* If and only if a set of external environmental conditions is met,

the improper position of the mechanical device will cause a disastrous

failure.

* Analysis techniques such as **fault tree analysis, real-time logic, and Petri net models can be used to predict** causes of hazards.
* Once hazards are identified and analyzed, safety-related requirements can be specified for the software.
* the specification can contain a list of undesirable events and the desired system responses to these events.
* Software safety examines the ways in which failures result in conditions that can lead to a mishap.
* That is, failures are not considered in a vacuum, but are evaluated in the context of an entire computer-based system and its environment.

**THE ISO 9000 QUALITY STANDARDS**

* A quality assurance system may be defined as the organizational structure, responsibilities, procedures, processes, and resources for implementing quality management
* Quality assurance systems are created to help organizations ensure

their products and services satisfy customer expectations by meeting

their specifi -cations.

* These systems cover a wide variety of activities encompassing a product’s entire life cycle including planning, controlling, measuring, testing and reporting, and improving quality levels throughout the development and manufacturing process.
* ISO 9000 describes quality assurance elements in generic terms that can be applied to any business regardless of the products or services offered.
* To become registered to one of the quality assurance system models contained in ISO 9000, a company’s quality system and operations are scrutinized by thirdparty auditors for compliance to the standard and for effective operation.
* Upon successful registration, a company is issued a certificate from a registration body represented by the auditors.
* Semi annual surveillance audits ensure continued compliance to the standard.
* The requirements delineated by ISO 9001:2008 address topics such as
* management responsibility,
* quality system,
* contract review,
* design control,
* document and data control,
* product identification and traceability,
* process control,
* inspection and testing,
* corrective and preventive action,
* control of quality records,
* internal quality audits,
* training,
* servicing, and
* statistical techniques.
* In order for a **software organization to become registered to ISO 9001:2008**, **it must establish policies and procedures to address each of the requirements just noted** and thenbe able to demonstrate that these policies and procedures are being followed.

**THE SQA PLAN**

* The SQA Plan provides a road map for instituting software quality assurance.
* It developed by the SQA group or by the software team if an SQA group does not exist the plan serves as a template for SQA activities that are instituted for each software project
* A standard for SQA plans has been published by the IEEE [IEE93].
* The standard recommends a structure that identifies:
* the purpose and scope of the plan,
* a description of all software engineering work products

(e.g., models, source code) that fall within the SQA,

* all applicable standards and practices that are applied during the software process,
* SQA actions and tasks (including reviews and audits) and

their placement throughout the software process,

* the tools and methods that support SQA actions and tasks,
* software configuration management procedures,
* methods for assembling, safeguarding, and maintaining all

SQA-related records, and

* organizational roles and responsibilities relative to product

quality

**REENGINEERING**

**Re-engineering**

***“Re-engineering is the examination and alteration of a system to reconstitute it in a new form***.”

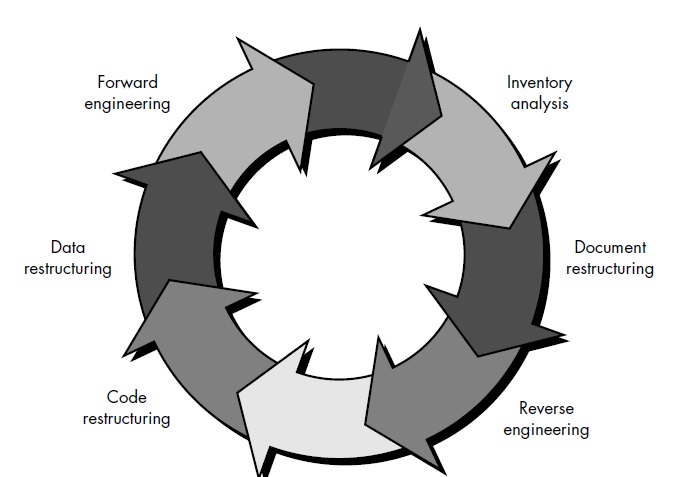
* This process encompasses a combination of sub-processes like reverse engineering, forward engineering, reconstructing etc

**Objectives of Re-engineering:**

* To describe a cost-effective option for system evolution.
* To describe the activities involved in the software maintenance process.
* To distinguish between software and data re-engineering and to explain the problems of data re-engineering

**SOFTWARE REENGINEERING**

* Reengineering is a rebuilding activity,



**FIG**: RE-ENGINEERING

**Steps involved in Re-engineering:**

* **Inventory analysis. .**
* Every software organization should have an inventory of all applications.
* The inventory can be nothing more than a spread sheet model containing information that provides a detailed description of every active application.
* the inventory should be revisited on a regular cycle.
* The status of applications change as a function of time,
* **Document restructuring.**

Weak documentation is the trademark of many legacy systems. Solution over this

1. *Creating documentation is far too time consuming.*

If a program is static- Documentations is limited

If a program is mot static life, -Documentations

undergo significant changes

1. *Documentation must be updated, but we have limited*

*resources.*

1. *The system is business critical and must be fully redocumented.*

* **Reverse engineering.** 
  + - The term *reverse engineering* has its origins in the hardware world.
    - A company disassembles a competitive hardware product in an effort to understand its competitor's design and manufacturing "secrets."
    - These secrets could be easily understood if the competitor's design and manufacturing specifications were obtained.
    - In essence, successful reverse engineering derives one or more design and manufacturing specifications for a product by examining actual specimen of the product.
* Therefore, reverse engineering for software is the process of analyzing a program in an effort to create a representation of the program at a higher level of abstraction than source code.
* Reverse engineering is a process of design recovery.
* Reverse engineering tools extract data, architectural, and procedural design information from an existing program

**Code restructuring.**

* Some legacy systems have a relatively solid program architecture,
* But individual modules were coded in a way that makes them difficult to understand, test, and maintain.
* In such cases, the code within the suspect modules can be restructured.
* To accomplish this activity, the source code is analyzed using a restructuring tool.
* Violations of structured programming constructs are noted and code is then restructured
* The resultant restructured code is reviewed and tested to ensure that no anomalies have been introduced

**Data restructuring**

* A program with weak data architecture will be difficult to adapt and enhance.
* Data restructuring is a full-scale reengineering activity.
* In most cases, data restructuring begins with a reverse engineering activity.
* Data objects and attributes are identified, and existing data structures are reviewed for quality.

**Forward engineering**

* Forward engineering not only recovers design information from existing software
* Also forward engineering uses design information to alter or reconstitute the existing system in an effort to improve its overall quality.

**Reverse Engineering**

Reverse engineering can extract design information from source code,

but the abstraction level, the completeness of the documentation,

the degree to which tools and a human analyst work together, and the

directionality of the process are highly variable.

The *abstraction level* of a reverse engineering process and the tools used to effect

it refers to the sophistication of the design information that can be extracted from

source code. Ideally, the abstraction level should be as high as possible. That is, the

reverse engineering process should be capable of deriving procedural design representations

(a low-level abstraction), program and data structure information

(a somewhat higher level of abstraction), object models, data and/or control flow

models (a relatively high level of abstraction), and entity relationship models (a high

level of abstraction). As the abstraction level increases, you are provided with information

that will allow easier understanding of the program.

The *completeness* of a reverse engineering process refers to the level of detail that

is provided at an abstraction level. In most cases, the completeness decreases as the

abstraction level increases. For example, given a source code listing, it is relatively

easy to develop a complete procedural design representation. Simple architectural

design representations may also be derived, but it is far more difficult to develop a

complete set of UML diagrams or models.

Completeness improves in direct proportion to the amount of analysis performed

by the person doing reverse engineering. *Interactivity* refers to the degree to which

the human is “integrated” with automated tools to create an effective reverse engineering

process. In most cases, as the abstraction level increases, interactivity must

increase or completeness will suffer.

If the *directionality* of the reverse engineering process is one-way, all information

extracted from the source code is provided to the software engineer who can