วิศวกรรมซอฟต์แวร์

Software Engineering

สมเกียรติ วังศิริพิทักษ์

somkiat.wa@kmitl.ac.th

ห้อง 518 หรือ ห้อง 506 (MIV Lab)

PART II

Quality Management

Software Quality Assurance

Software Quality Assurance

What is it?

- To understand what 'software quality' is, you have to

  (1) explicitly define what is meant when you say "software quality,"

  (2) create a set of activities that will *help ensure* that every software engineering work product exhibits *high quality*,

  (3) perform quality control and assurance activities on every software project,

  (4) use metrics to develop strategies for improving your software process

  and, as a consequence, the quality of the end product.

Software Quality Assurance

Who does it?

- Everyone involved in the software engineering process is responsible for quality.

Why is it important?

- You can do it right, or you can do it over again. ทำซ้ำแล้วจะถูก
- If a software team stresses quality in all software engineering activities, it reduces the amount of rework that it must do.
- That results in lower costs, and more importantly, improved time to market.

Software Quality Assurance

What are the steps?

- Before software quality assurance (SQA) activities can be initiated, it is important to define **software quality** at several **different levels** of abstraction.
  1 2
- Once you understand what quality is, a software team must identify a set of **SQA activities** that will filter errors out of work products before they are passed on.
  3

Software Quality Assurance

What is the work product?

- A Software Quality Assurance Plan is created to define a software team's SQA strategy.
  - During modeling and coding, the primary SQA work product is the **output of technical reviews**.
  - During testing, **test plans** and **procedures** are produced.
  - **Other work products** associated with **process improvement** may also be generated.

How do I ensure that I've done it right?

- Find errors **before** they become defects!
- That is, work to improve your *defect removal* **efficiency**, thereby reducing the amount of rework that your software team must perform.

Software Quality Assurance

- Software quality assurance (**SQA**) encompasses :

(1) an SQA process,

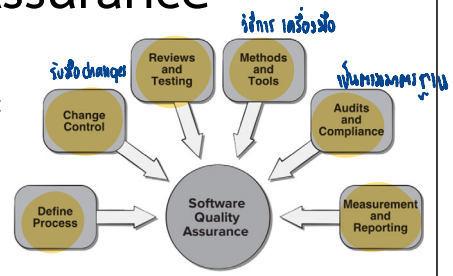
(2) specific quality assurance and quality control tasks (including technical reviews and a multitiered testing strategy),

(3) effective software engineering practice (**methods** and **tools**),

(4) control of all software work products and the changes made to them

(5) a procedure to ensure **compliance** with software development standards (when applicable), and

(6) **measurement** and **reporting** mechanisms.



Elements of Software Quality Assurance

SQA ต้องดูให้มั่นใจว่า ทีม+รพ นิยามตามมาตรฐาน ต้องสอดส่อง

- **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 for software engineers.
  - The 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.

  ทำตามหรือไม่

Elements of Software Quality Assurance

- **Testing**. Software testing is a quality control function that has one 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.

  รวบรวมและวิเคราะห์ deu คนไหนทำเองเร่อนั่นบ่อย → แล้วเราสรุปเพื่อแก้ปัญหา
  - 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.

Elements of Software Quality Assurance

- **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.
  - SQA ensures that adequate change management practices have been instituted.
- **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.

Elements of Software Quality Assurance

เอาพวกคนอื่น

- **Vendor management**. Three categories of software are acquired from external software vendors—
  shrink-wrapped packages (e.g., Microsoft Office),
  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. SQA ต้องตามเจ้าของ vendor
  - 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.

Elements of Software Quality Assurance

- **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 mobile apps, and ensure that software has not been tampered with internally.
  - SQA ensures that appropriate process and technology are used to achieve software security.
- **Safety**. Because software is almost always a pivotal component of human-rated systems (e.g., automotive or aircraft applications), the impact of hidden defects can be catastrophic.
  - SQA may be responsible for assessing the impact of software failure and for initiating those steps required to reduce risk.

Elements of Software Quality Assurance

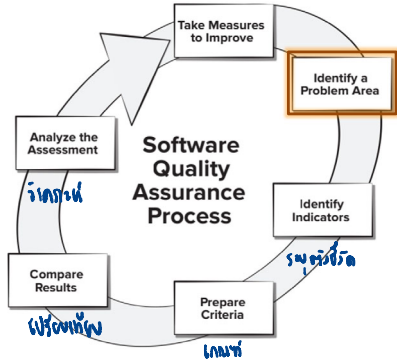
ค.เสี่ยงที่อาจเกิดขึ้น

- **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.

- In addition to each of these concerns and activities, SQA works to ensure that software **support activities** (e.g., maintenance, help lines, documentation, and manuals) are conducted or produced with quality as a dominant concern.

หาจุดบกพร่อง ปรับปรุง เสนอแนวทางแก้ไข

SQA Tasks, Goals, and Metrics



- Modern software QA is often data driven, as shown in the figure below.
  - The product stakeholders **define goals** and **quality measures**, problem areas are identified, indicators are measured, and a determination is made as to whether or not process changes are needed.
  - Software engineers address **quality** (and perform quality control activities) by applying solid technical methods and measures, conducting technical reviews, and performing well-planned software testing.

SQA Tasks

- The charter of the SQA group is to assist the software team in achieving a high quality end product.
- The Software Engineering Institute recommends a set of **SQA activities** that address

  quality assurance planning,
  oversight,
  record keeping,
  analysis, and
  reporting.

อิสระ: ต้องแยกออก (dev เป็น SQA ไม่ได้ งานจะเยอะ ความต่อรองน้อยลง)

- These **activities** are performed (or facilitated) by an independent SQA group that: →

SQA Tasks

- **Prepares an SQA plan for a project**.
  - The plan identifies
    - evaluations to be performed
    - audits and reviews to be performed
    - standards that are applicable to the project
    - procedures for error reporting and tracking
    - documents to be produced by the SQA group
    - amount of feedback provided to the software project team

เข้าประชุม

- **Participates in the development of the project's software process description**.
  - 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.

SQA Tasks

- Reviews software engineering activities to verify *compliance with* the defined software *process*.
  - 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.
  - reviews selected work products; identifies, documents, and tracks deviations; verifies that corrections have been made
  - periodically reports the results of its work to the project manager.

SQA Tasks

- Ensures that deviations in software work and work products are documented and handled according to a documented procedure.
- Records any noncompliance and reports to senior management.
  - Noncompliance items are tracked until they are resolved.

Goals, Attributes, and Metrics

- The SQA activities are performed to achieve a set of **pragmatic goals**:
- **Requirements quality**. The correctness, completeness, and consistency of the requirements model will have a strong influence on the quality of all work products that follow.
- **Design quality**. Every element of the design model should be assessed by the software team to ensure that it exhibits high quality and that the design itself conforms to requirements.
- **Code quality**. Source code and related work products (e.g., other descriptive information) must conform to local coding standards and exhibit characteristics that will facilitate maintainability.
- **Quality control effectiveness**. A software team should apply limited resources in a way that has the highest likelihood of achieving a high quality result.

Software quality goals, attributes, and metrics

Goal	Attribute	Metric
Requirement quality	Ambiguity	Number of ambiguous modifiers (e.g., many, large, human-friendly)
	Completeness	Number of TBA, TBD
	Understandability	Number of sections/subsections
	Volatility	Number of changes per requirement
		Time (by activity) when change is requested
	Traceability	Number of requirements not traceable to design/code
	Model clarity	Number of UML models
		Number of descriptive pages per model
		Number of UML errors
Design quality	Architectural integrity	Existence of architectural model
	Component completeness	Number of components that trace to architectural model
	Interface complexity	Complexity of procedural design
		Average number of pick to get to a typical function or content
		Layout appropriateness
	Patterns	Number of patterns used

Code quality	Complexity	Cyclomatic complexity
	Maintainability	Design factors
	Understandability	Percent internal comments
		Variable naming conventions
		Percent reused components
	Reusability	Percent reused component
	Documentation	Readability index
QC effectiveness	Resource allocation	Staff hour percentage per activity
	Completion rate	Actual vs. budgeted completion time
	Review effectiveness	See review metrics
	Testing effectiveness	Number of errors found and criticality
		Effort required to correct an error
		Origin of error

Software quality goals, attributes, and metrics

Statistical Software Quality Assurance

- For software, statistical quality assurance implies the following **steps**:

1. Information about software errors and defects is collected and categorized.
2. An attempt is made to trace each error and defect to its underlying cause (e.g., nonconformance to specifications, design error, violation of standards, poor communication with the customer).
3. Using the Pareto principle (80 percent of the defects can be traced to 20 percent of all possible causes), isolate the 20 percent (the *vital few*).
4. Once the vital few causes have been identified, move to correct the problems that have caused the errors and defects.

Statistical Software Quality Assurance

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 1 year.
  - Some of the errors are uncovered as software is being developed.
  - Other defects are encountered after the software has been released to its end users.
- Although hundreds of different problems are uncovered, all can be tracked to one (or more) of the following **causes**:

Statistical Software Quality Assurance

A Generic Example

- 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)

Statistical Software Quality Assurance

A Generic Example

- To apply statistical SQA, a table is built.
- The table indicates that **IES**, **MCC**, and **EDR** are the **vital few causes** that account for **53 percent of all errors.**
- **IES**, **EDR**, **PLT**, and **EDL** would be selected as the vital few causes **if only serious errors** are considered.
- Once the vital few causes are determined, the SE organization can begin corrective action.

	Total		Serious		Moderate		Minor	
Error	No.	%	No.	%	No.	%	No.	%
IES	205	22%	34	27%	68	18%	103	24%
MCC	156	17%	12	9%	68	18%	76	17%
IDS	48	5%	1	1%	24	6%	23	5%
VPS	25	3%	0	0%	15	4%	10	2%
EDR	130	14%	26	20%	68	18%	36	8%
ICI	58	6%	9	7%	18	5%	31	7%
EDL	45	5%	14	11%	12	3%	19	4%
IET	95	10%	12	9%	35	9%	48	11%
IID	36	4%	2	2%	20	5%	14	3%
PLT	60	6%	15	12%	19	5%	26	6%
HCI	28	3%	3	2%	17	4%	8	2%
MIS	56	6%	0	0%	15	4%	41	9%
Totals	942	100%	128	100%	379	100%	435	100%

Statistical Software Quality Assurance

A Generic Example

- For example, to correct MCC (Misinterpretation of customer communication), you might implement requirements gathering techniques to improve the quality of customer communication and specifications.
- To improve EDR (Error in data representation), you might acquire tools for data modeling and perform more stringent data design reviews.
- Corrective action focuses primarily on the vital few.
- As the vital few causes are corrected, **new candidates pop** to the top of the stack.

	Total		Serious		Moderate		Minor	
Error	No.	%	No.	%	No.	%	No.	%
IES	205	22%	34	27%	68	18%	103	24%
MCC	156	17%	12	9%	68	18%	76	17%
IDS	48	5%	1	1%	24	6%	23	5%
VPS	25	3%	0	0%	15	4%	10	2%
EDR	130	14%	26	20%	68	18%	36	8%
ICI	58	6%	9	7%	18	5%	31	7%
EDL	45	5%	14	11%	12	3%	19	4%
IET	95	10%	12	9%	35	9%	48	11%
IID	36	4%	2	2%	20	5%	14	3%
PLT	60	6%	15	12%	19	5%	26	6%
HCI	28	3%	3	2%	17	4%	8	2%
MIS	56	6%	0	0%	15	4%	41	9%
Totals	942	100%	128	100%	379	100%	435	100%

Six Sigma for Software Engineering

- the **Six Sigma** strategy "is a business-management strategy designed to improve the quality of process outputs by minimizing variation and causes of defects in processes."
- It is a subset of the Total Quality Management (TQM) methodology with a heavy focus on statistical applications used to reduce costs and improve quality.
- The term Six Sigma is derived from **six standard deviations**—3.4 instances (defects) per million occurrences—implying an extremely high-quality standard.

Six Sigma for Software Engineering

- The Six Sigma methodology defines three core steps:
  - **Define** customer requirements and deliverables 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 improvement is required for an existing software process, 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** (define, measure, analyze, improve, and control) **method**.

Six Sigma for Software Engineering

- 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
    (1) avoid the root causes of defects and
    (2) 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**.
- A comprehensive discussion of Six Sigma is best left to resources dedicated to the subject.

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".
- To illustrate, program X is estimated to have a reliability of 0.999 over eight elapsed processing hours.
  - In other words, if program X were to be executed 1000 times and require a total of 8 hours of elapsed processing time (execution time), it is likely to operate correctly (without failure) 999 times.

Measures of Reliability and Availability

- If we consider a computer-based system, a simple measure of **reliability** is *mean time between failure* (**MTBF**):

$$MTBF = MTTF + MTTR$$

where the acronyms MTTF and MTTR are *mean time to failure* and *mean time to repair*, respectively.

- **MTBF** and related measures are **based on CPU time**, not wall clock time.

- Although *debugging* (and related corrections) *may be required* following a failure, in many cases the **software** will **work properly after a restart with no other change**.

Measures of Reliability and Availability

- Many researchers argue that MTBF is a far more useful measure than other quality-related software metrics (e.g., the total defect count)
- However, 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

(2) MTBF can be misinterpreted to mean average life span even though 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 1 billion hours of operation.*
- Therefore, 1 FIT is equivalent to one failure in every billion hours of operation.

Measures of Reliability and Availability

- **Software availability** is the probability that a program is operating according to requirements at a given point in time:

  Availability = (MTTF / (MTTF + MTTR)) x 100%

- The MTBF reliability measure is equally sensitive to MTTF and MTTR.
- The availability measure is somewhat more sensitive to MTTR, an *indirect measure of the maintainability* of software.
- Of course, some aspects of availability have *nothing to do with failure*. For example, scheduling downtime (for support functions) causes the software to be unavailable.

Software Safety

- **Software safety** is a software quality assurance activity that focuses on the identification and assessment of potential hazards that may affect software negatively and cause an entire system to fail.
- If hazards can be identified early in the software process,

  → software design features can be specified that will either eliminate **or** control potential hazards.

The ISO 9000 Quality Standards

จะปรับปรุงเรื่อยๆ

- ISO 9000 describes quality assurance elements in generic terms that can be applied to any business regardless of the products or services offered.
- The requirements delineated by **ISO 9001:2015** 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.

  มีค.น่าเชื่อถือ, มั่นใจได้ว่าจะออกมาเป็น รป ที่มีมาตรฐาน

- For a software organization to become registered to ISO 9001:2015, it must establish policies and procedures to address each of the requirements just noted (and others) and then be able to demonstrate that these policies and procedures are being followed.