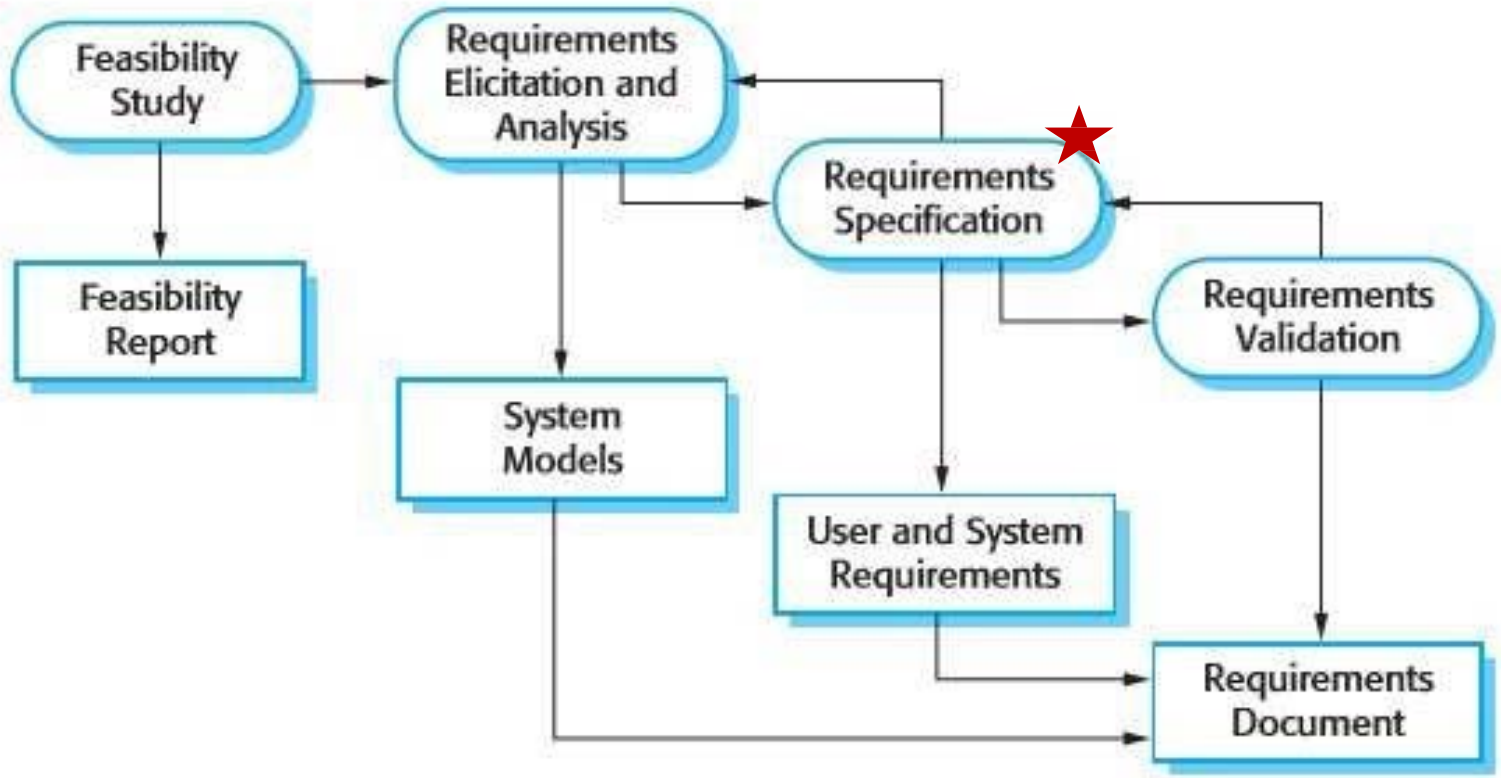
8. Quality Attributes

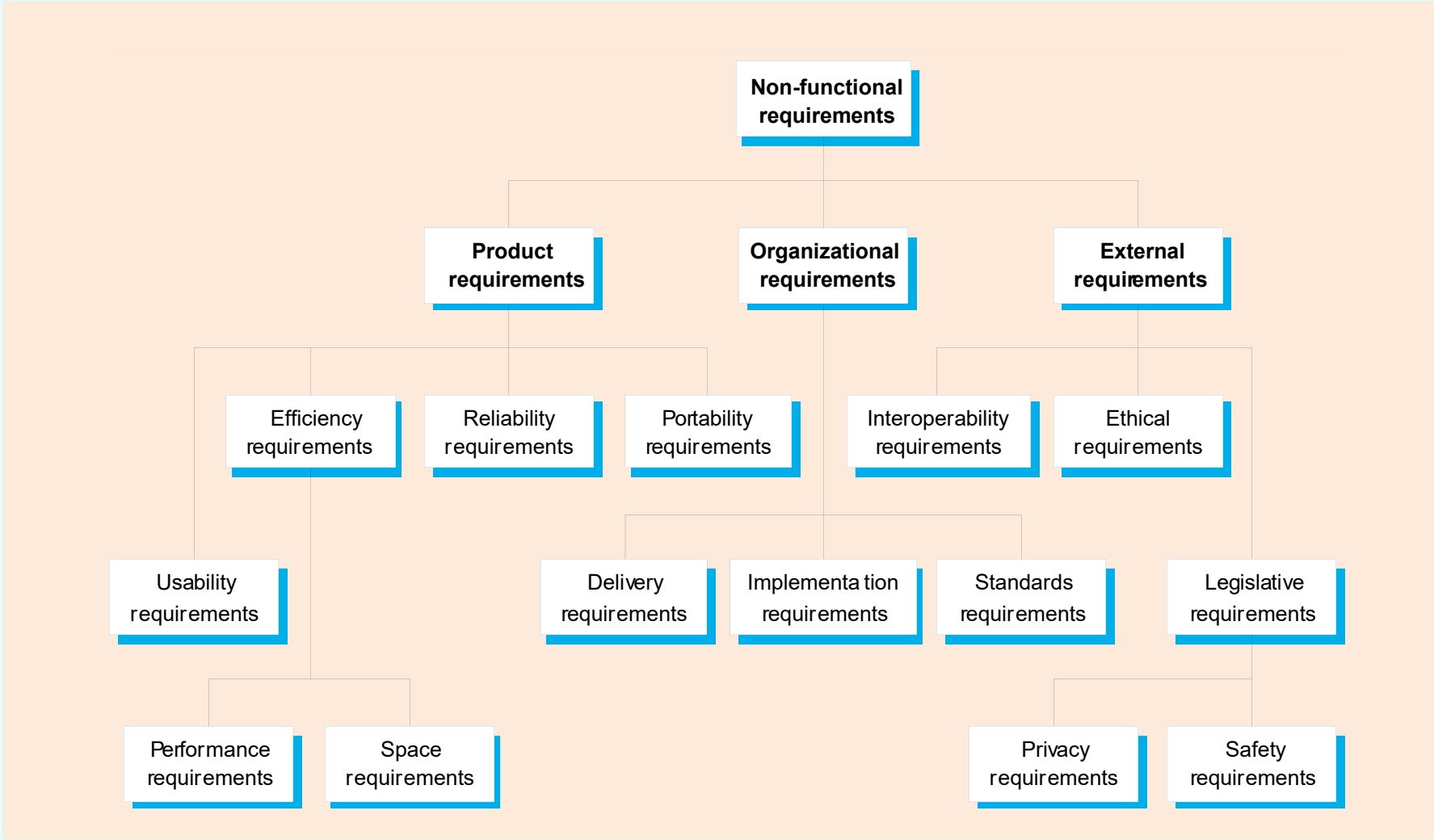
Requirements Engineering Process



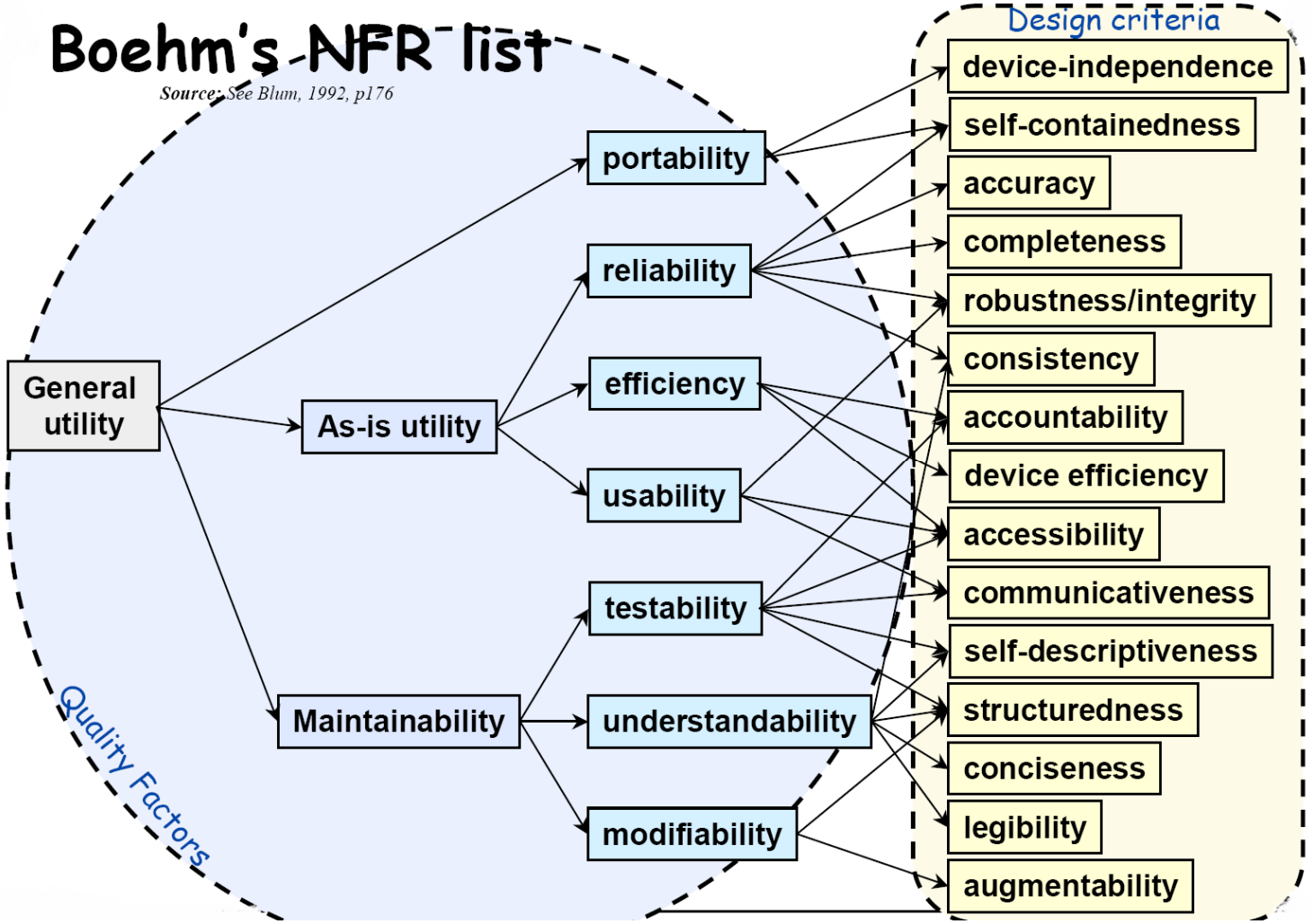
Non-Functional Requirements

- IEEE 9126 / 25010
  - "*A Software requirement that described not what the software will do, but how the software will do it, for example, software performance requirements, software external interface requirements, design constraints, and software quality attributes.*"
- Sommerville
  - "*Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.*"
- Wikipedia
  - "*An requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions. The plan for implementing functional requirements is detailed in the system design. The plan for implementing non-functional requirements is detailed in the system architecture, because they are usually architecturally significant requirements.*"

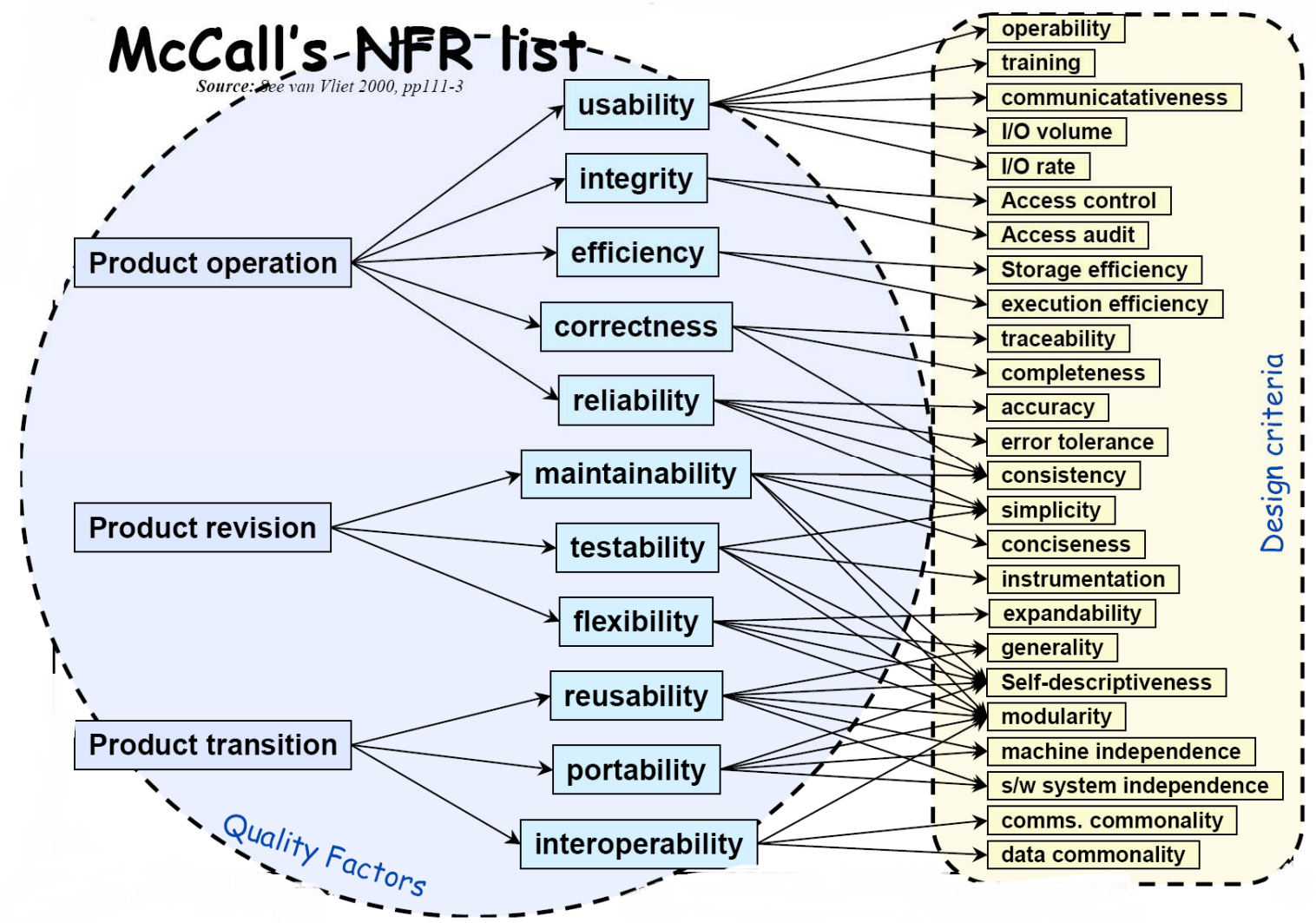
3 Types of Non-Functional Requirements



Boehm's NFR


Boehm's NFR list
Source: See Blum, 1992, p176
General utility
As-is utility
Maintainability
portability
reliability
efficiency
usability
testability
understandability
modifiability
Quality Factors
Design criteria
device-independence
self-containedness
accuracy
completeness
robustness/integrity
consistency
accountability
device efficiency
accessibility
communicativeness
self-descriptiveness
structuredness
conciseness
legibility
augmentability

McCall's NFR


McCall's NFR list
Source: See van Vliet 2000, pp111-3
Product operation
Product revision
Product transition
usability
integrity
efficiency
correctness
reliability
maintainability
testability
flexibility
reusability
portability
interoperability
Quality Factors
operability
training
communicatativeness
I/O volume
I/O rate
Access control
Access audit
Storage efficiency
execution efficiency
traceability
completeness
accuracy
error tolerance
consistency
simplicity
conciseness
instrumentation
expandability
generality
Self-descriptiveness
modularity
machine independence
s/w system independence
comms. commonality
data commonality
Design criteria

Quality Attributes

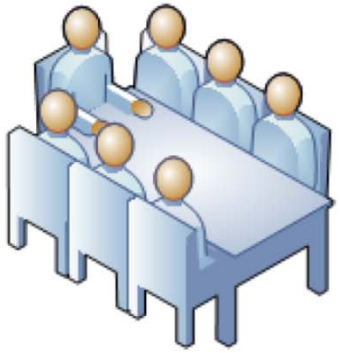
- Measurable or testable properties of a system
  - Used to indicate how well the system satisfies the needs of its stakeholders
    - Availability, configurability, modifiability, performance, reliability, reusability, security, portability, maintainability, efficiency, usability
  - Emergent properties : not a measure of software in isolation
    - Measures the relationship between software and its application domain
    - Cannot measure this until you place the software into its environment
      - Quality will be different in different environments

- Software quality is all about **fitness to purpose** of **stakeholders**.
  - *"Does it do what is needed?"*
  - *"Does it do it in the way that its users need it to?"*
  - *"Does it do it reliably enough? fast enough? safely enough? securely enough?"*
  - *"Will it be affordable? will it be ready when its users need it?"*
  - *"Can it be changed as the needs change?"*

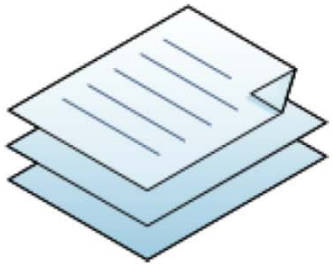
Quality Attributes : Taxonomies

- **-ilities**
  - understandability, usability, modifiability, interoperability, reliability, portability, maintainability, scalability, configurability, customizability, adaptability, variability, volatility, traceability, …
- **-ities**
  - security, simplicity, clarity, ubiquity, integrity, modularity, …
- **-ness**
  - user-friendliness, robustness, timeliness, responsiveness, correctness, completeness, conciseness, cohesiveness, …
- **others**
  - performance, efficiency, accuracy, precision, cost, development time, low coupling, …

Stakeholders and Quality Attributes



Lead To



Stakeholder Needs

Quality Attribute Requirements

"Increase market share" ----------> Modifiability, Usability

"Maintain a quality reputation" ----------> Performance, Usability, Availability

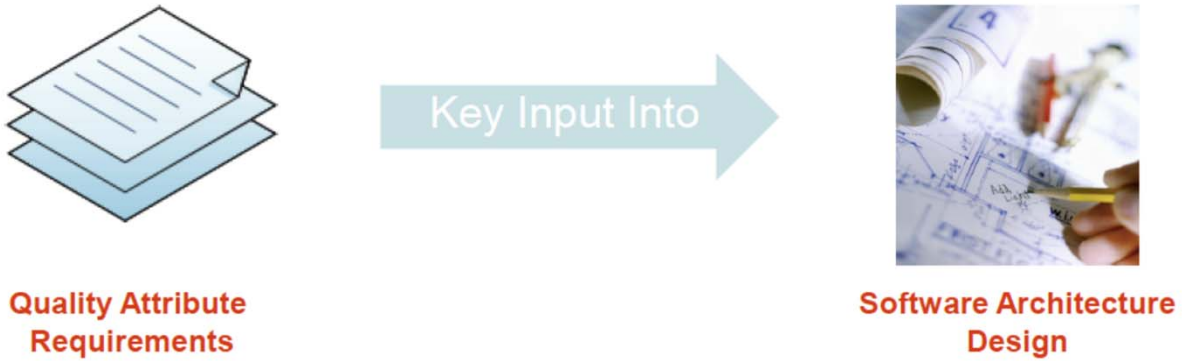
"Introduce new capabilities seamlessly" ----------> Performance, Availability, Modifiability

"Provide a programmer-friendly framework" ----------> Modifiability

"Integrate with other systems easily" ----------> Interoperability, Portability, Modifiability

Quality Attributes and Architecture

- The degree to which a system satisfies quality attribute requirements is directly dependent on architectural structure.



- **Architects** need to have a solid understanding of the quality attribute requirements for a system, when they are designing the system's software architecture.

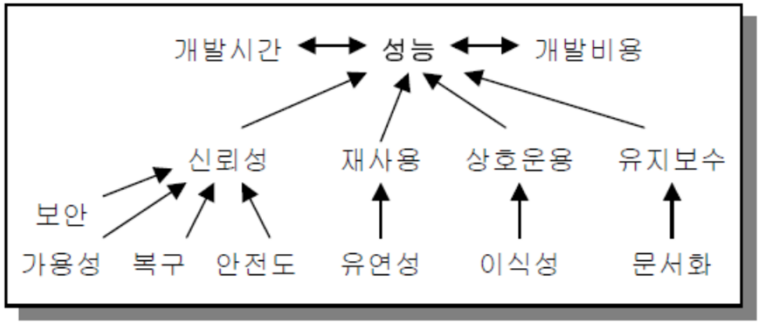
Problematic Features of Quality Attribute

- **Non-Operational requirements**
  - "The system must be easy to use."
  - "The system must have high performance."
  - "The system must be portable."

- **Debating the quality attribute to which a system behavior belongs**
  - "The system must process 10,000 messages per second."

- **Vocabulary variations**
  - Everyone knows what "high performance" means, but different each others.

- **Various inter-dependency among quality attributes**
  - Trade-off
  - No 100% satisfied



Quality Requirements: Examples

- 응용 프로그램을 위한 프로세서 용량과 RAM 중에서 20%는 최대 부하시점에서도 사용되지 않아야 한다
- 감사접속 권한을 가진 자만이 고객 거래자료를 볼 수 있다
- 사용자가 파일을 저장하기 전 편집기에 에러가 발생하면 편집 중이던 모든 변경내용을 에러발생 5분 전 까지로 복구한다
- 메뉴 파일의 모든 기능은Ctrl+다른 키를 사용하는 단축키가 정의되어야 한다
- 함수 호출은 3 단계 이상 중첩되지 않는다
- 모듈의 Cyclomatic Complexity는 20을 넘지 않는다
- 온도관리 주기는 0.8초 이내에서 수행한다
- 모든 웹 페이지는 10Mbps LAN 접속에서 5초 이내로 다운로드 한다

Quality Requirements: Example

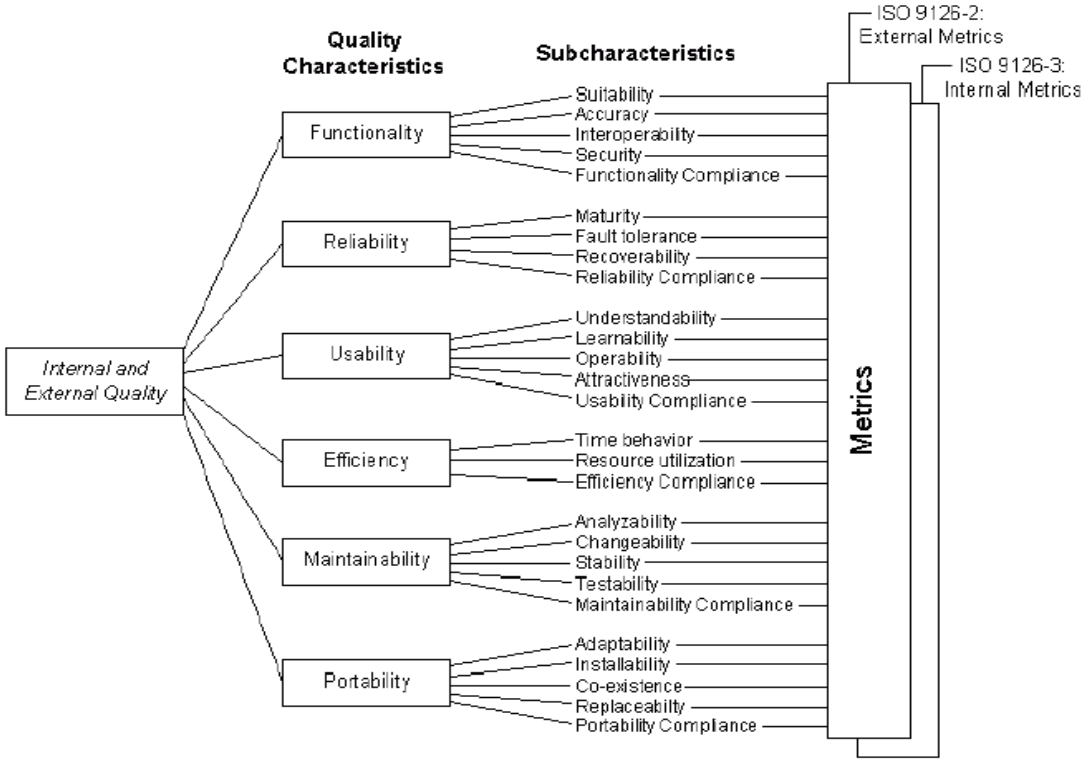
- MHC-PMS (Mental Health Care Patient Management System)
  - Product requirements
    - The MHC-PMS shall be available to all clinics during normal working hours (Mon–Fri, 0830–17.30). Downtime within normal working hours shall not exceed five seconds in any one day. → Availability
    - The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized. → Usability
  - Organizational requirements
    - Users of the MHC-PMS system shall authenticate themselves using their health authority identity card. → Security
  - External requirements
    - The system shall implement patient privacy provisions as set out in HStan-03-2006-priv. → Standard compliance

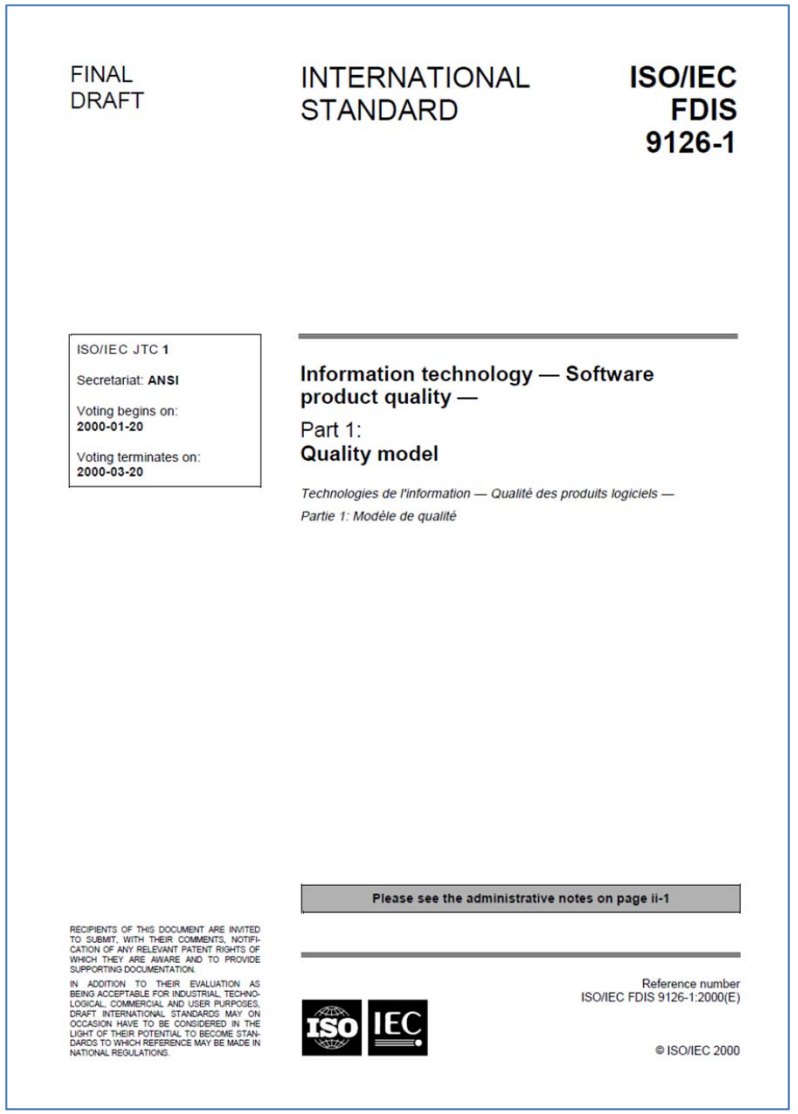
Two Categories of Quality Attributes

- Bredemeyer Consulting 2001

	사용자 관점	제품 관점
대 상	• 소비자/고객 관점	• 생산자 /기업 관점
품질 특성	Run-time 품질 • 목적 만족을 위한 성질, 성능 • 시스템 행위에 대한 사용자 평가	Development-time 품질 • 개발 조직의 산출물 속성 만족 • 프로세스 산출물에 대한 내부 평가
품질 요소	Usability, Correctness, Safety, Reliability, Availability, Performance,	Localizability, Modifiability, Extensibility, Evolvability, Composability, Reusability
품질 Source	• User Objectives, Values, Concerns • Competitive Analysis of Features	•개발 조직의 Objectives, Values, Concerns, 제약사항 • 경쟁업체 및 산업 동향

ISO/IEC 9126





FINAL DRAFT

INTERNATIONAL STANDARD

ISO/IEC FDIS 9126-1

ISO/IEC JTC 1

Secretariat: ANSI

Voting begins on: 2000-01-20

Voting terminates on: 2000-03-20

Information technology — Software product quality —

Part 1:
Quality model

Technologies de l'information — Qualité des produits logiciels — Partie 1: Modèle de qualité

Please see the administrative notes on page ii-1

RECIPIENTS OF THIS DOCUMENT ARE INVITED TO SUBMIT, WITH THEIR COMMENTS, NOTIFICATION OF ANY RELEVANT PATENT RIGHTS OF WHICH THEY ARE AWARE AND TO PROVIDE SUPPORTING DOCUMENTATION.

IN ADDITION TO THEIR EVALUATION AS BEING ACCEPTABLE FOR INDUSTRIAL, TECHNOLOGICAL, COMMERCIAL AND USER PURPOSES, DRAFT INTERNATIONAL STANDARDS MAY ON OCCASION HAVE TO BE CONSIDERED IN THE LIGHT OF THEIR POTENTIAL TO BECOME STANDARDS TO WHICH REFERENCE MAY BE MADE IN NATIONAL REGULATIONS.

Reference number
ISO/IEC FDIS 9126-1:2000(E)

ISO 9126-1 : Information Technology - Software Product Quality - Part 1: Quality Model

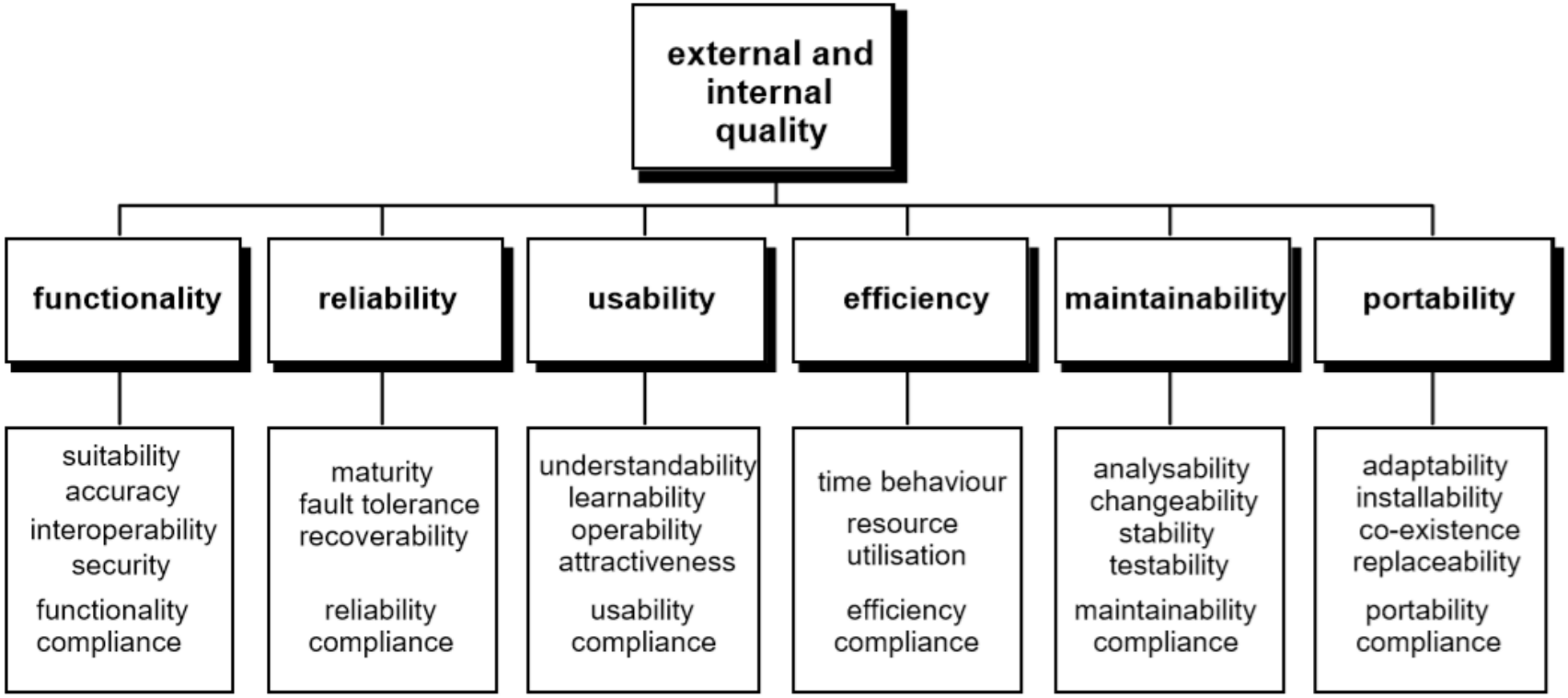


Figure 4 – Quality model for external and internal quality

Microsoft Application Architecture Guide

- **Quality attributes** are the overall factors that affect run- time behavior, system design, and user experience.
  - They represent areas of concern that have the potential for application wide impact across layers and tiers
  - When designing applications to meet any of the quality attributes requirements, it is necessary to consider the potential impact on other

- 4 Categories of Quality Attributes
  - **Design Qualities** : Conceptual Integrity ,Maintainability ,Reusability
  - **Run-time Qualities** : Availability, Interoperability, Manageability, Performance, Reliability, Scalability, Security
  - **System Qualities** : Supportability, Testability
  - **User Qualities** : Usability

Microsoft Quality Attributes – Design Qualities

- **Conceptual Integrity**
  - defines the consistency and coherence of the overall design.
  - includes the way that components or modules are designed, as well as factors such as coding style and variable naming.

- **Maintainability**
  - the ability of the system to undergo changes with a degree of ease.
  - these changes could impact components, services, features, and interfaces when adding or changing the functionality, fixing errors, and meeting new business requirements.

- **Reusability**
  - defines the capability for components and subsystems to be suitable for use in other applications and in other scenarios.
  - Reusability minimizes the duplication of components and also the implementation time.

Microsoft Quality Attributes – Runtime Qualities

- **Availability**
  - defines the proportion of time that the system is functional and working.
  - It can be measured as a percentage of the total system downtime over a predefined period.
  - Availability will be affected by system errors, infrastructure problems, malicious attacks, and system load.

- **Interoperability**
  - the ability of a system or different systems to operate successfully by communicating and exchanging information with other external systems written and run by external parties.
  - An interoperable system makes it easier to exchange and reuse information internally as well as externally.

- **Manageability**
  - defines how easy it is for system administrators to manage the application, usually through sufficient and useful instrumentation exposed for use in monitoring systems and for debugging and performance tuning.

Microsoft Quality Attributes – Runtime Qualities

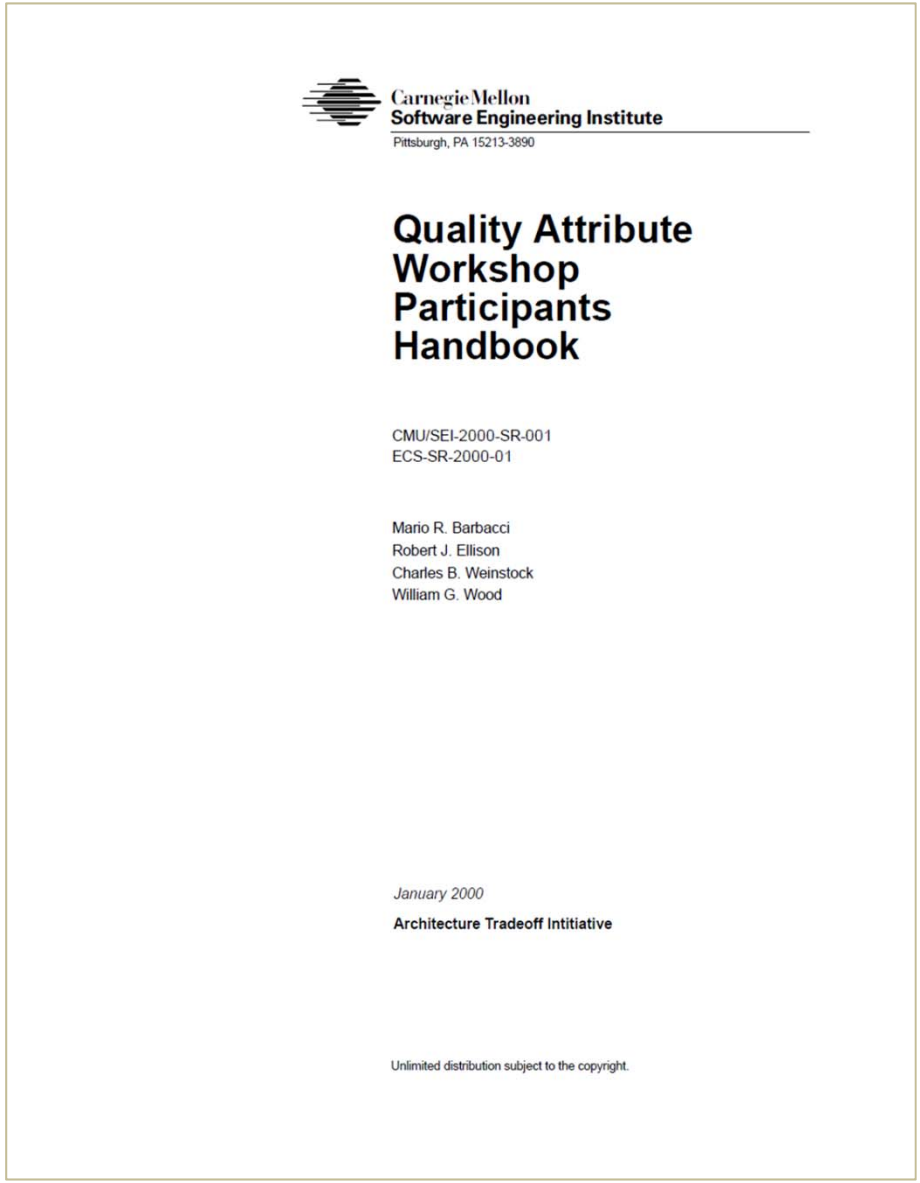
- **Performance**
  - an indication of the responsiveness of a system to execute any action within a given time interval.
  - It can be measured in terms of latency or throughput.
- **Reliability**
  - the ability of a system to remain operational over time.
  - measured as the probability that a system will not fail to perform its intended functions over a specified time interval.
- **Scalability**
  - ability of a system to either handle increases in load without impact on the performance of the system, or the ability to be readily enlarged.
- **Security**
  - the capability of a system to prevent malicious or accidental actions outside of the designed usage, and to prevent disclosure or loss of information.

Microsoft Quality Attributes – User Qualities

- **Usability**
  - defines how well the application meets the requirements of the user and consumer by being intuitive, easy to localize and globalize, providing good access for disabled users, and resulting in a good overall user experience.

CMU SEI Quality Attributes

- Dependability
- Security
- Modifiability
- Interoperability
- Performance



Carnegie Mellon
Software Engineering Institute
Pittsburgh, PA 15213-3890

Quality Attribute Workshop Participants Handbook

CMU/SEI-2000-SR-001
ECS-SR-2000-01

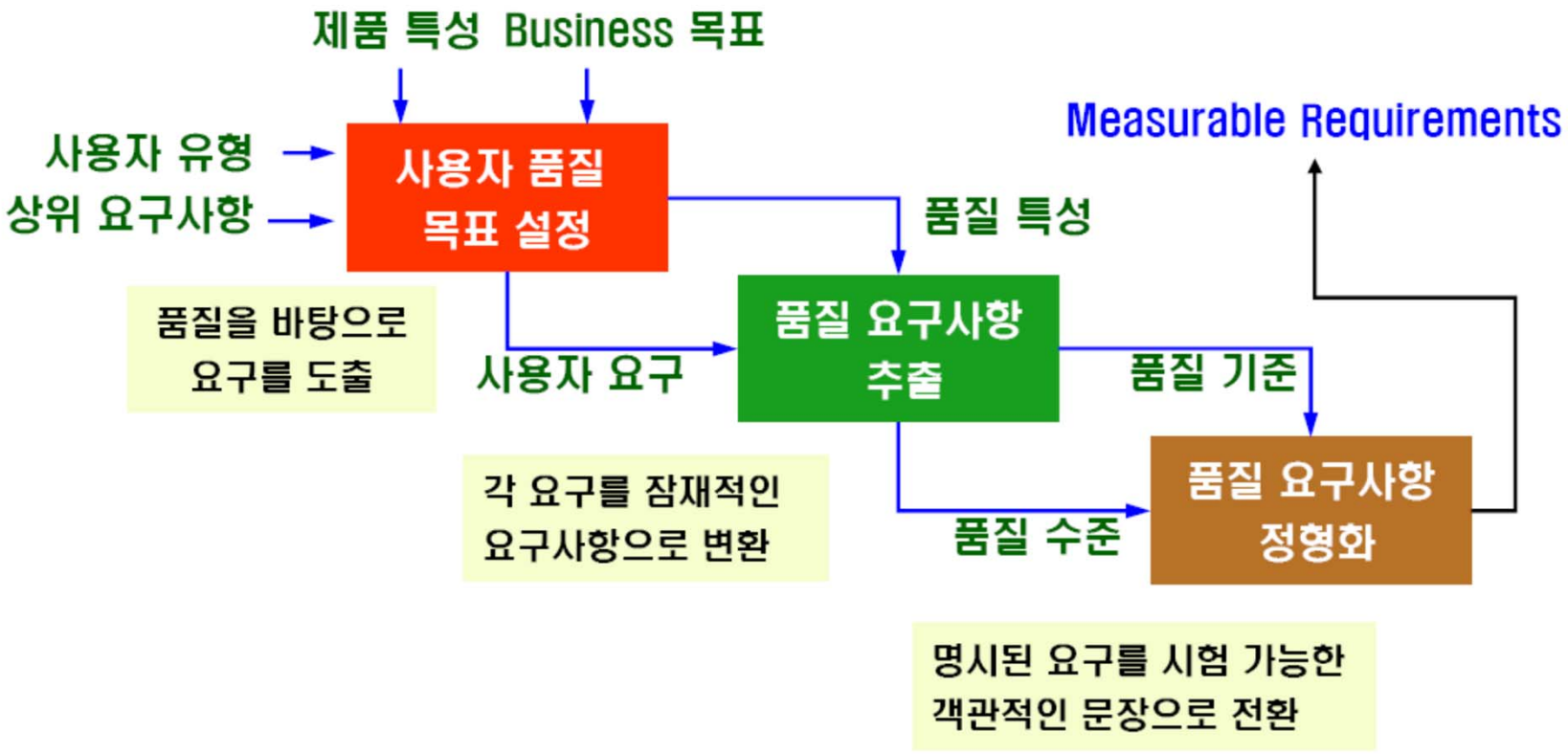
Mario R. Barbacci
Robert J. Ellison
Charles B. Weinstock
William G. Wood

January 2000

Architecture Tradeoff Intitiative

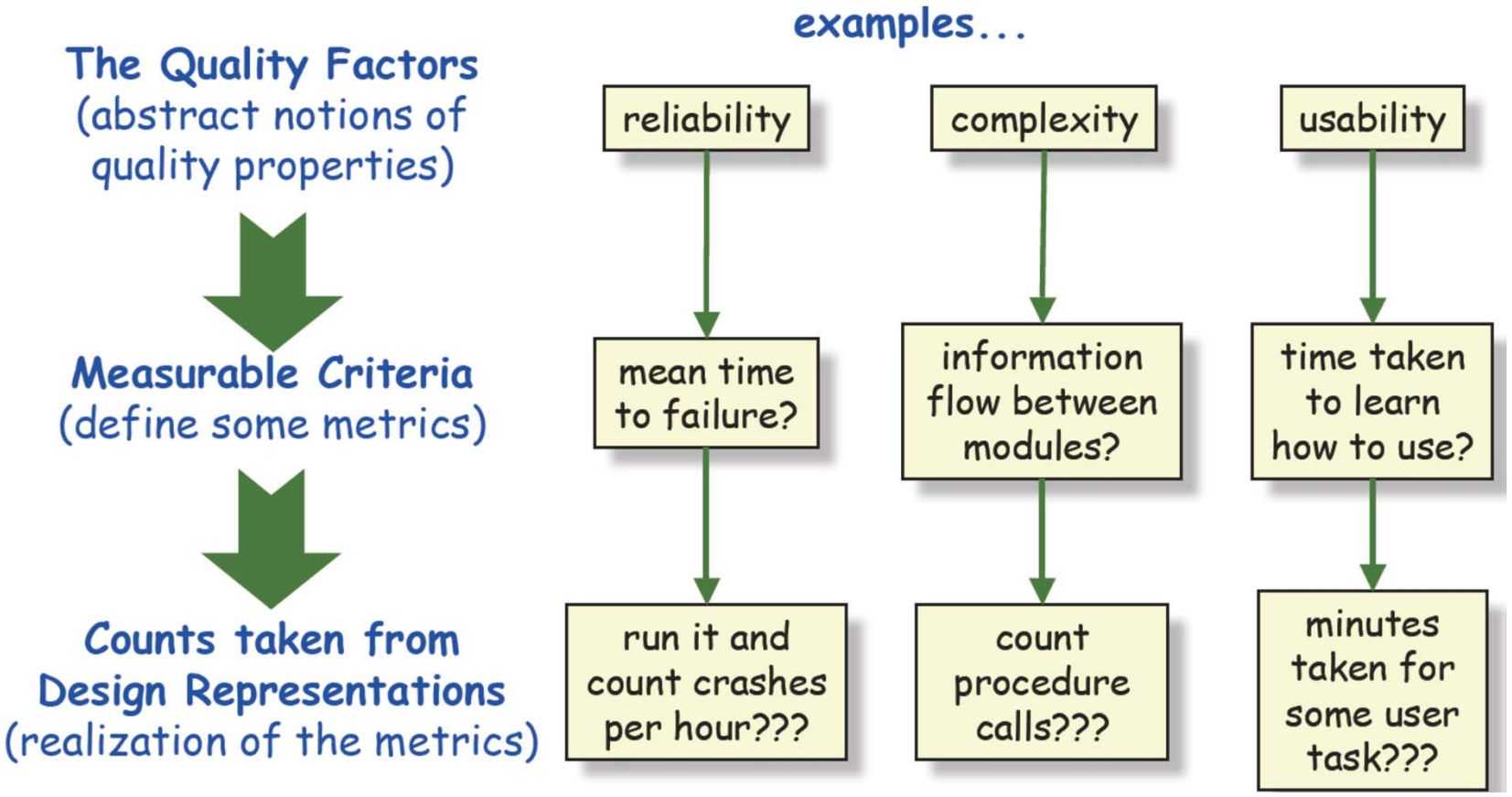
Unlimited distribution subject to the copyright.

A Process Defining Quality Requirements



Making All Requirements Measurable

- Turn vague ideas about quality into **measurables** or **verifiable**

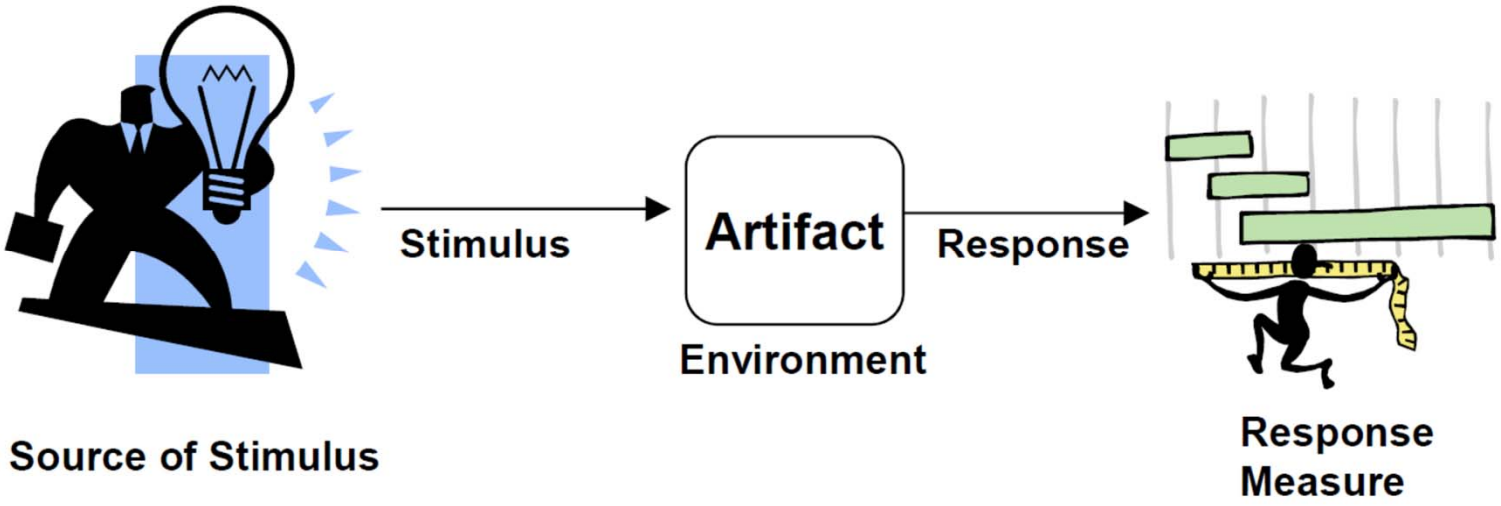


Quality Metric & Measure

Quality Factor	Metric & Measures
Speed	- Processed transactions/second - User/event response time - Screen refresh time
Size	- Mbytes - Number of ROM chips
Ease of Use	- Training time - Number of help frames
Reliability	- Mean time to failure - Probability of unavailability - Rate of failure occurrence - Availability
Robustness	- Time to restart after failure - Percentage of events causing failure - Probability of data corruption on failure
Portability	- Percentage of target dependent statements - Number of target systems
Maintainability	- Volume of data recorded in operation - Number of failures estimated - Correction time / software size

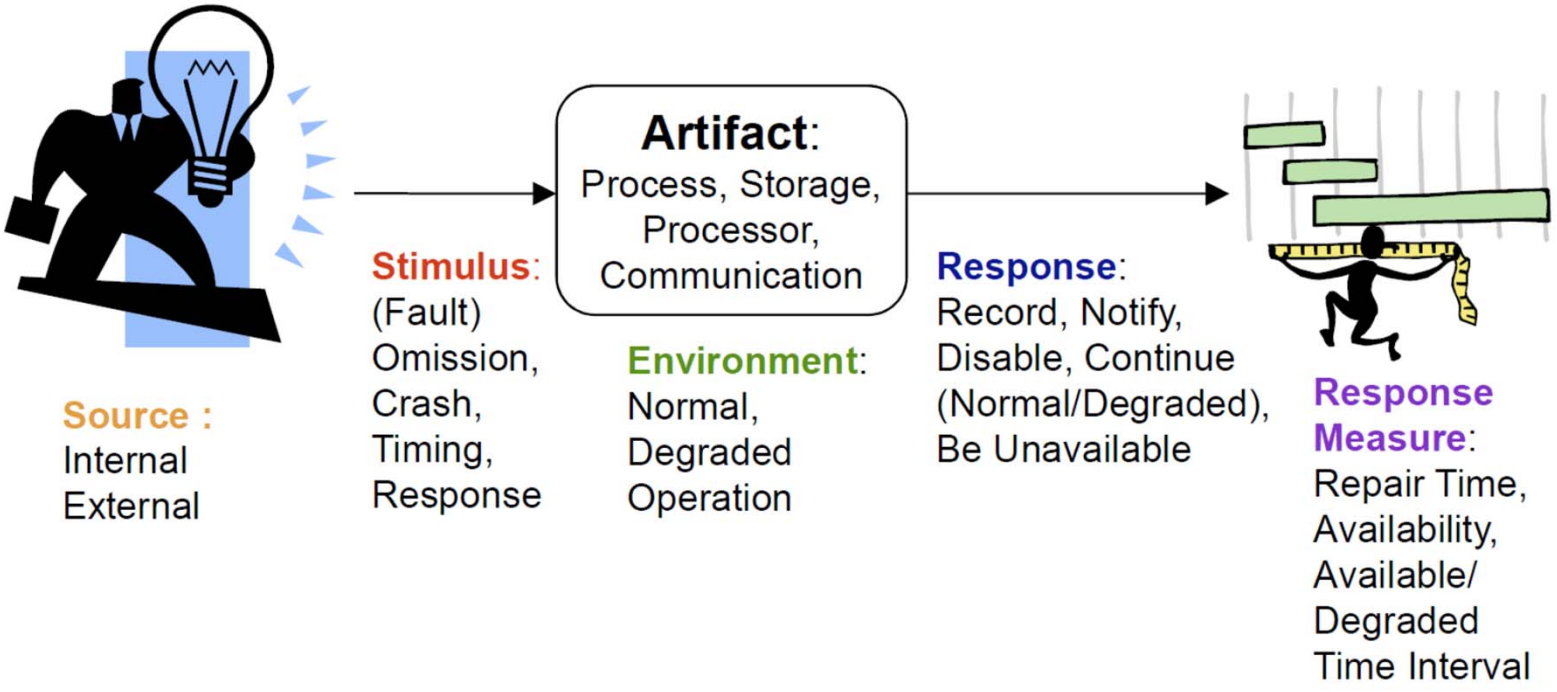
Quality Attribute Scenarios

- **QAS (Quality Attribute Scenario)** is an effective way of identifying and specifying quality-attribute-specific requirements.
  - Specific to the particular system under considerations
  - Instantiated from the attribute characterizations of general scenarios



A QAS Example for Availability

- "An **unanticipated external message** is received by a process during **normal operation**. **The process informs the operator of the receipt of the message and continues to operated with no downtime.**"



Quality Attribute Tree : Example

No.	Category	Response Measure	품질속성 요구사항
1	Performance	시스템 동작 Hz 수	1 Core System에서 4 Core System을 지원하는 system으로 변경될 경우, 정상적인 Operation을 수행하는 데 있어서 소모되는 전력이 1 Core System의 경우와 비교하여 35%의 Hz수를 출력하여야 한다.
2		Communication 횟수, Communication 데이터	정상적인 Operation을 수행하는 데 있어서 Core 간 Communication 으로 인해 발생되는 Overhead 증가분은 10% 이내이어야 한다.
3		사용된 Memory 용량	정상적인 Operation을 수행하는 데 있어서 memory 사용량 증가 정도는 70 % 이내이어야 한다.
4		수행 중 각 Core들의 Idle Time	정상적인 Operation을 수행하는 데 있어서 전체 system의 운영 시간 중 각 Core들이 Idle 상태에 머무르는 시간은 15% 이내 이어야 한다.
5		Data Frames/Second	정상적인 Operation하에서 Video Data Decoding 성능은 기존 1 Core System의 3배 정도인 2.5 Data Frames/Second를 만족시켜야 한다.
6		화면의 가로 세로 픽셀 수	정상적인 Operation하에서 Video Decoder가 지원 가능한 출력 화면의 크기는 기존 1 Core System의 경우와 마찬가지로 1280X720 픽셀까지이다 .
7		Bit rate	정상적인 Operation하에서 Video Decoder가 출력하는 화면의 화질을 측정하는 Bit rate는 기존 1 Core System의 경우와 마찬가지로 20Mbps를 지원 가능하여야 한다.
8	Modifiability	수정 컴포넌트 비율	4 Core System에서 향후 그 이상의 개수로 Core 숫자가 늘어날 경우, 4 Core System 기반의 Architecture 상에서 변경되는 Component와 Connector의 비율은 Parallel Node의 숫자에 해당되는 Instance 개수 증가를 제외하고 50% 미만이어야 한다.
9	Functionality	시스템 기능 가용률 :전체 기능 개수 대비 가용 기능 비율	1 Core System에서 4 Core System을 지원하는 system으로 변경될 경우, 정상적인 Operation하에서 Video Decoder가 제공하던 기능 중 가용한 기능은 100%를 만족시켜야 한다.
10		해당사항 없음	Video Decoder의 input data stream 중에 error가 있는 input frame이 입력될 경우, Video Decoder는 해당 frame의 decoding을 수행하지 않고 pass한 후 다음 frame을 읽어 들여야 한다.
11	Portability	시스템 기능 가용률 :전체 기능 개수 대비 가용 기능 비율	Cache memory size 가 32K인 Device에서 16K인 Device로 변경될 경우, 정상적인 Operation하에서 Video Decoder가 제공하던 기능 중 가용한 기능은 100%를 만족시켜야 한다.

The QAS Template

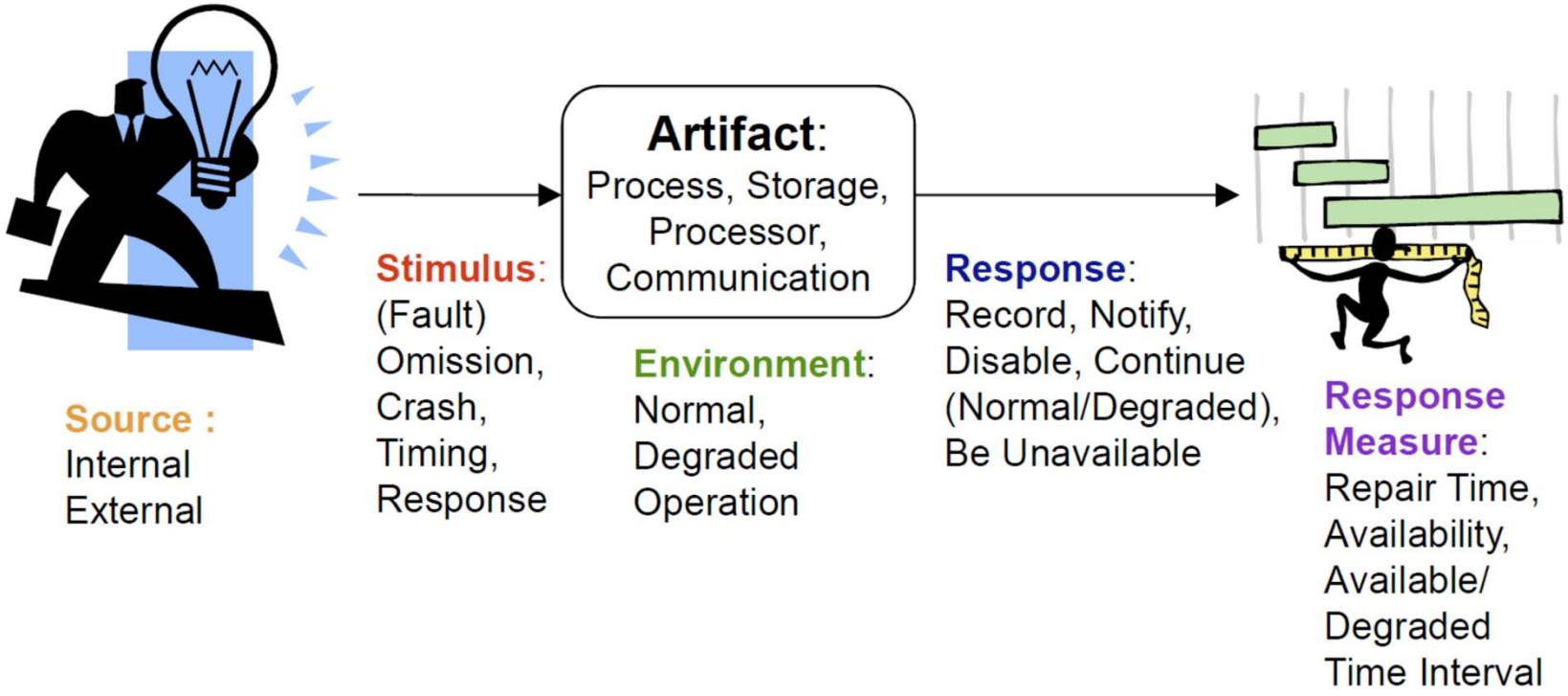
Requirement-ID	*QA_###*
Category	관련된 ***Quality Attribute***가 무엇인지 기술함 ***(*** 예***: Performance, Reliability, Security*** 등***)***
Source	***Stimulus***를 발생시키는 주체가 무엇인지 기술함
Stimulus	시스템에 입력되는 내외부 자극이 무엇인지 기술함
Artifacts	***Stimulus***의 영향을 받는 시스템의 내부 모듈, 컴포넌트, 혹은 시스템 전체
Environment	해당 ***Stimulus*** 발생시 시스템의 환경***(***운영모드일 수도 있고, 그 외 다양한 상황 가능***)***
Response	기술된 ***Environment***에서 ***Artifact***이 ***Stimulus***를 받아들인 후 취하는 ***Action***이 무엇인지 설명함
Response Measure	위의 ***Response***의 정도를 측정하는 단위가 무엇인지 기술함 ***(***예***:*** 초당 데이터 처리량, 반응시간, 시간일 경우 ***hour, minute, second*** 여부 등***)***
Priority	***Quality Attribute Tree*** 상에서의 우선순위
Description	위에 기술된 ***Source***부터 ***Response Measure***까지의 내용을 하나의 문장으로 요약해서 기술함

QAS – Availability (Reliability)

- **Source of Stimulus**
  - Internal or External
- **Stimulus: a fault one of the following classes**
  - Omission: a component fails to response to an input
  - Crash: the component repeatedly suffers omission faults
  - Timing: a component responds but the response is early or late
  - Response: a component responds with an incorrect value
- **Artifact**
  - Processor, communication channel, process, or storage
- **Environment**
  - Normal operation, degraded mode
- **Response**
  - Logging the failure,
  - notifying selected users or other systems,
  - switching to a degraded mode,
  - shutting down external systems,
  - becoming unavailability during repair.
- **Response Measure**
  - An availability percentage
  - A time to repair
  - Time interval in which the system must be available
  - Time interval in which system can be in degraded mode

QAS Example – Availability (Reliability)

- "An **unanticipated external message** is received by a process during **normal operation**. **The process informs the operator of the receipt of the message and continues to operated with no downtime.**"

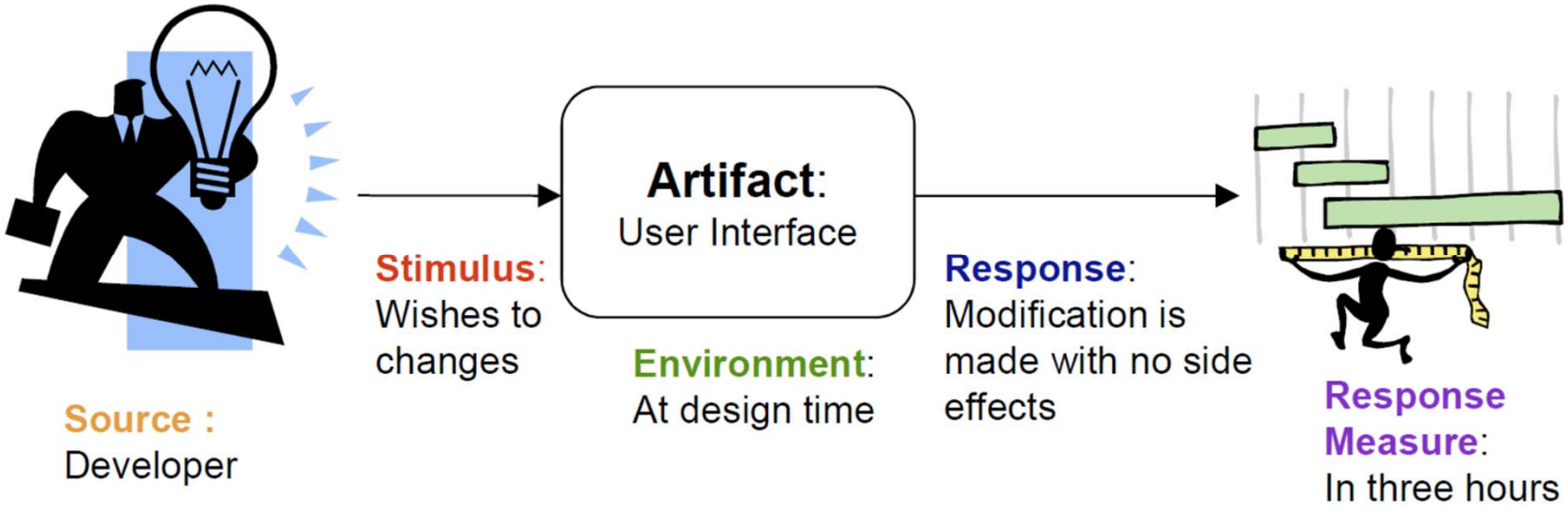


QAS – Modifiability (Adaptability)

- **Source of Stimulus: who makes the changes**
  - Developer, end user, system administrator
- **Stimulus: the changes to be made**
  - Addition/deletion/modification of a function, quality attribute, capacity
- **Artifact: what is to be changed**
  - The functionality of a system
  - its platform
  - its user interface
  - its environment
  - systems with which it interoperates
- **Environment: when the change can be made**
  - Design time, compile time, build time, initiation time, or run time
- **Response**
  - Locates places in architecture to be modified; Makes modification without affecting other functionality; tests modification; deploys modification
- **Response Measure**
  - Cost in terms of number of elements affects, effort, money; extent to which this affects other functions or quality attributes

QAS Example – Modifiability (Adaptability)

- "A **developer** **wishes to change** **the user interface to make a screen's background color blue. This change will be made to the code** **at design time**. **It will take less than three hours to make and test the change** and **no side effect changes** **will occur in the behavior**."

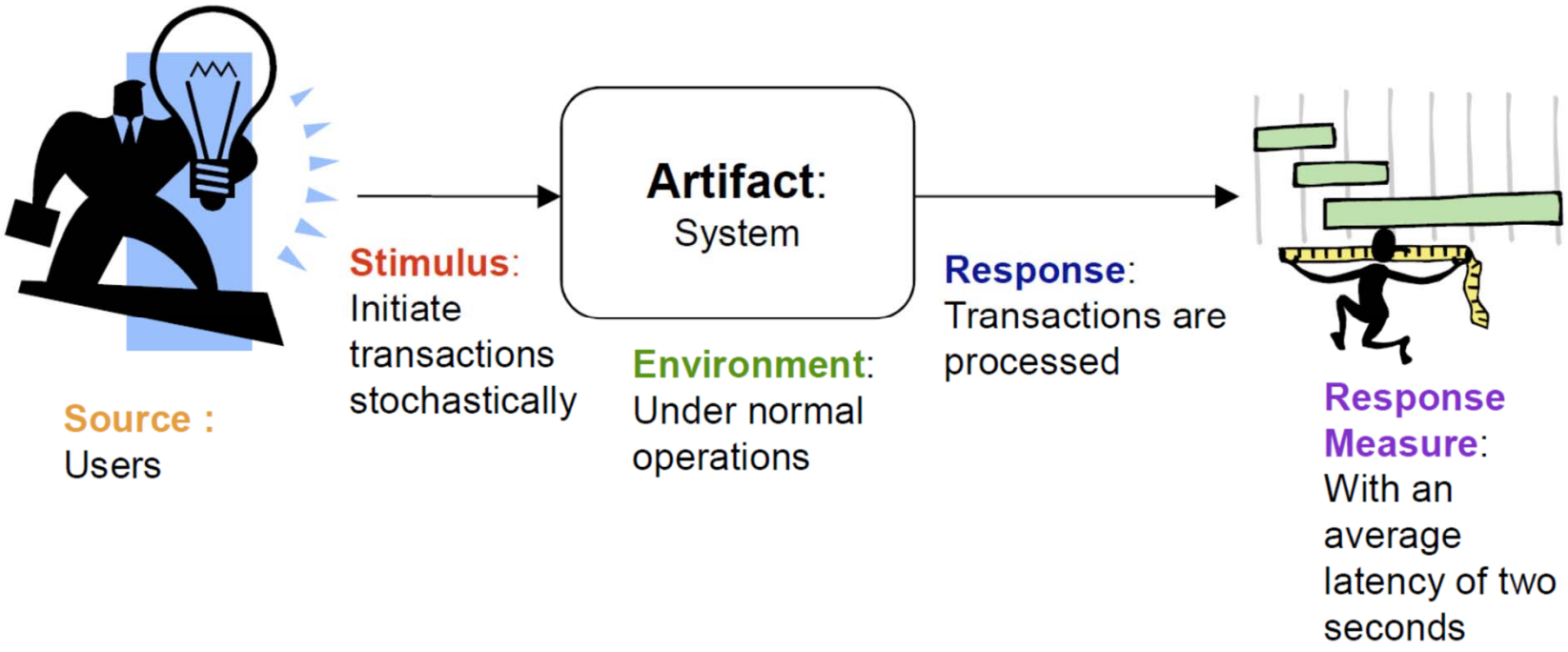


QAS – Performance

- **Source**
  - One of a number of independent sources, possible from within system
- **Stimulus**
  - Periodic events arrive, sporadic events arrive, stochastic events arrive
- **Artifact**
  - system's service
- **Environment**
  - Normal mode
  - Overloaded mode
- **Response**
  - Processes stimuli
  - Changes level of service
- **Response Measure**
  - Latency (Response Time)
  - Throughput

QAS Example – Performance

- "**Users initiate 1,000 transactions per minute stochastically under normal operations**, and **these transactions are processed with an average latency of two seconds**."

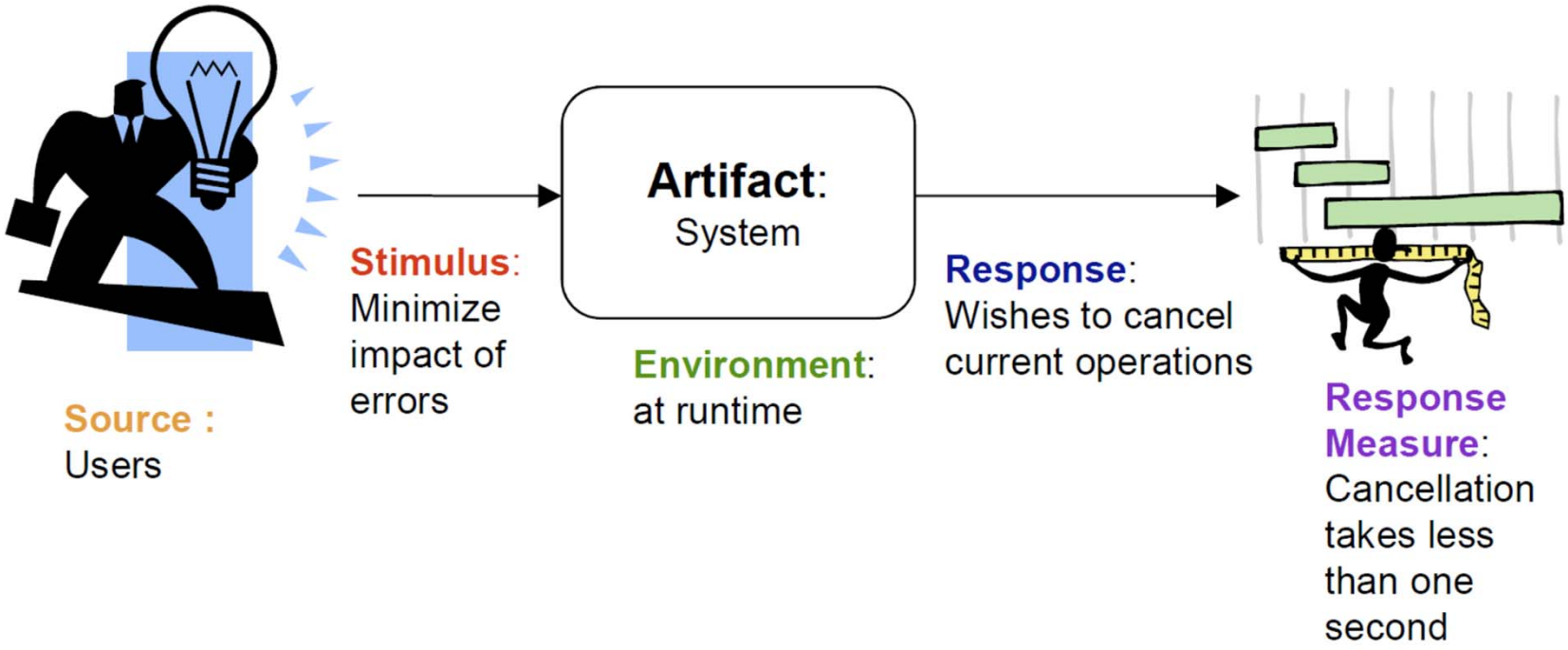


QAS - Usability

- **Source**
  - End user
- **Stimulus**
  - want to
    - learn system features; use system efficiently; minimize impact of errors; adapt system; feel comfortable
- **Artifact**
  - system
- **Environment**
  - At runtime or configure time
- **Response**
  - System provides one or more of the following responses
  - To support “learn system features”
    - Help system is sensitive to context; interface is familiar to user; interface is usable in an unfamiliar context
  - To support “use system efficiently”
    - Aggregation of data and/or commands; re-use of already entered data and/or commands; support for efficient navigation within a screen
  - To minimize impact of errors
    - Undo, cancel, recover from failure; recognize and correct user error; retrieve forgotten password; verify system resources
- **Response Measure**
  - Task time, number of errors, number of problems solved, user satisfaction, gain of user knowledge, ratio of successful operations to total operations, amount of time/data lost

QAS Example – Usability

- "A user wanting to minimize the impact of an error, wishes to cancel a system operation at runtime; cancellation takes place in less than one second."



Quality Requirements and Architecture Evaluation

- Quality requirements gives important information such as
  - "*Is the architecture suitable for the system for which it was devised?*"
  - "*Which of two competing architectures is most suitable for the system at hand?*"

- **An architecture is suitable if**,
  - The system that results from it **will meet its quality goals**.
    - A system is modifiable or not wrt. a specific kind of change.
    - A system is secure or not wrt. a specific kind of threat.
    - A system is reliable or not wrt. a specific kind of fault occurrence.
    - A system performs well or not wrt. specific performance criteria.
    - An architecture is buildable or not wrt. specific time and budget constraints.

- Questioning techniques for **architecture evaluation**
  - Rely on thought experiments to check architecture suitability
  - Scenario-based style: **ATAM** **(Architecture Tradeoff Analysis Method)**
  - Checklist-based style

Quality Attribute Workshop (QAW)

- **Quality Attribute Workshop (QAW)**
  - Facilitated method
    - System-centric
    - Used before the software architecture has been created
  - Engages system stakeholders early in the life-cycle
  - Reveals the driving quality attribute requirements of a software-intensive system
    - Scenario-based

- Outputs of a QAW
  - **Quality attribute requirements** for the system, documented as refined and prioritized **QAS**.
  - The quality attribute scenarios can then be used as the basis for designing the software architecture for the system.

Motivations of QAW

- **Design with Assurance**
  - The SEI Quality Attribute Workshop (QAW) provides the means to identify important quality attributes, derived from business and mission goals, before there is a software architecture.
  - Clarifying quality attribute concerns early provides architects with more insight into what is important and why, in turn improving their ability to create architectures that better meet organization needs.

- **Scenario-Based QAW**
  - In the QAW, an external team (such as SEI facilitators) holds meetings among stakeholders during which scenarios representing the quality attribute requirements are generated, prioritized, and refined.
  - From these discussions, system designers gain insight concerning stakeholder assumptions that may not have been expressed during elicitation of goals which quality attributes are pulling the architecture in different directions, informing subsequent tradeoff decisions.

The QAW Steps

1. QAW Introduction
2. Business/Mission Presentation
3. Architectural Plan Presentation
4. Identification of Architectural Drivers
5. Scenario Brainstorming
6. Scenario Consolidation
7. Scenario Prioritization
8. Scenario Refinement

The QAW Steps in Detail

1. **QAW Presentation and Introduction**
   - QAW facilitators describe the motivation for the QAW and explain each step of the method.
2. **Business/Mission Presentation**
   - A stakeholder presents the business and/or programmatic drivers for the system.
3. **Architectural Plan Presentation**
   - A technical stakeholder presents the system architectural plans as they stand with respect to early documents, such as high-level system descriptions, context drawings, or other artifacts that describe the system's technical details.
4. **Identification of Architectural Drivers**
   - Architectural drivers often include high-level requirements, business/mission concerns, and various quality attributes.
   - During this step, the facilitators and stakeholders reach a consensus about which drivers are key to the system.
5. **Scenario Brainstorming**
   - Stakeholders generate real-world scenarios for the system. Scenarios comprise a related stimulus, an environmental condition, and a response.
   - Facilitators ensure that at least one scenario addresses each of the architectural drivers identified in Step 4.
6. **Scenario Consolidation**
   - Scenarios that are similar in content are consolidated.
7. **Scenario Prioritization**
   - Stakeholders prioritize the scenarios through a voting process.
8. **Scenario Refinement**
   - For the top four or five scenarios, the following are described: the business/mission goals that are affected by those scenarios, the relevant quality attributes associated with those scenarios

Mini-QAW

1. Mini-QAW Introduction
2. Introduction to Quality Attributes, Quality Attributes Taxonomy
3. Scenario Brainstorming
   - "Walk the System Properties Web" activity
4. Raw Scenario Prioritization
   - Dot voting
5. Scenario Refinement
   - While time remains
6. Review Results with Stakeholders

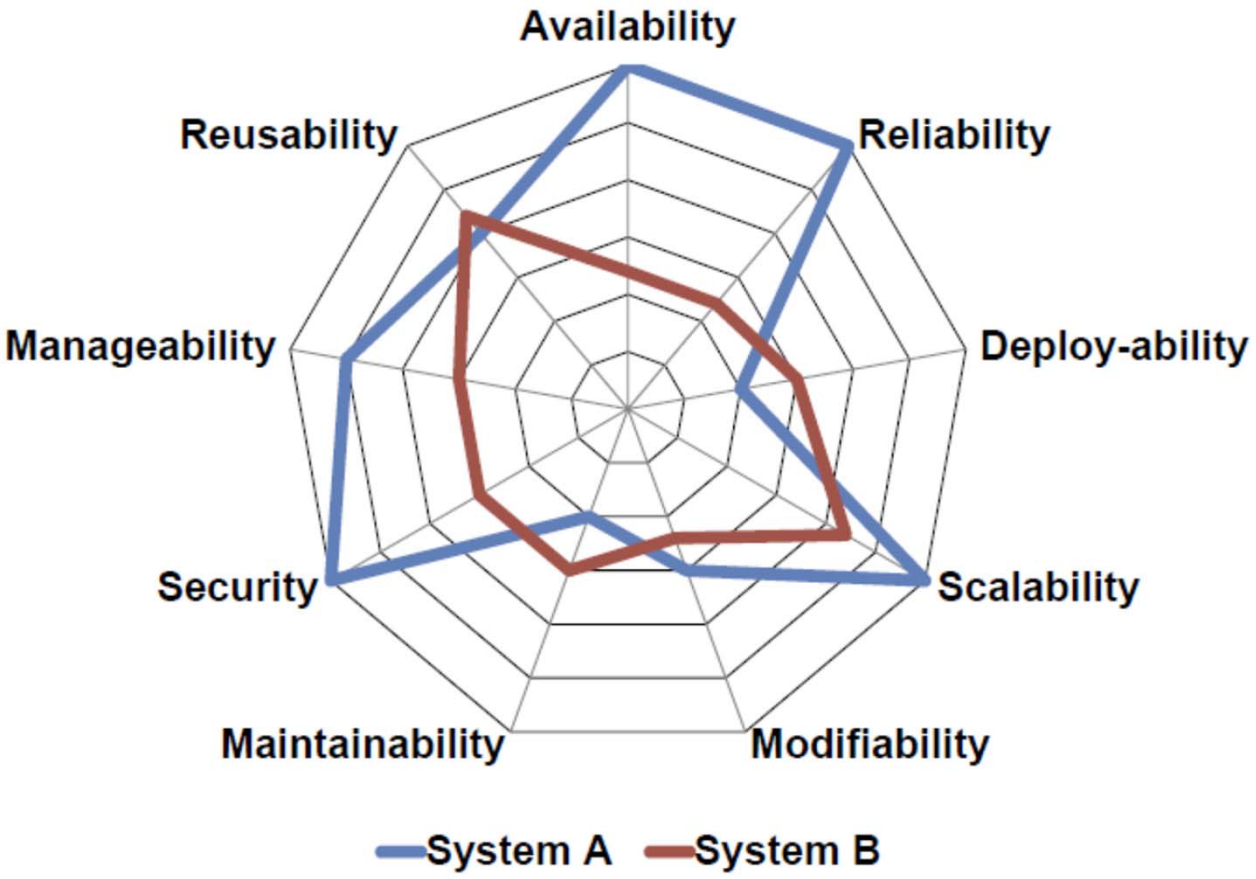
1. Mini-QAW Introduction

- Take into account specific roles of stakeholders.
  - For example, “Smart Home System” has 10 different stakeholders and goals.

Stakeholders	주요 역할	희망사항 - Goal
CEO		
개발자		
입주민1 (싱글)		
입주민2 (4인 가족)		
관리사무소		
AS센터 (출장방문수리)		
119 / 112		
정보통신부 / 시청		
한국전력공사		
인터넷 Provider		

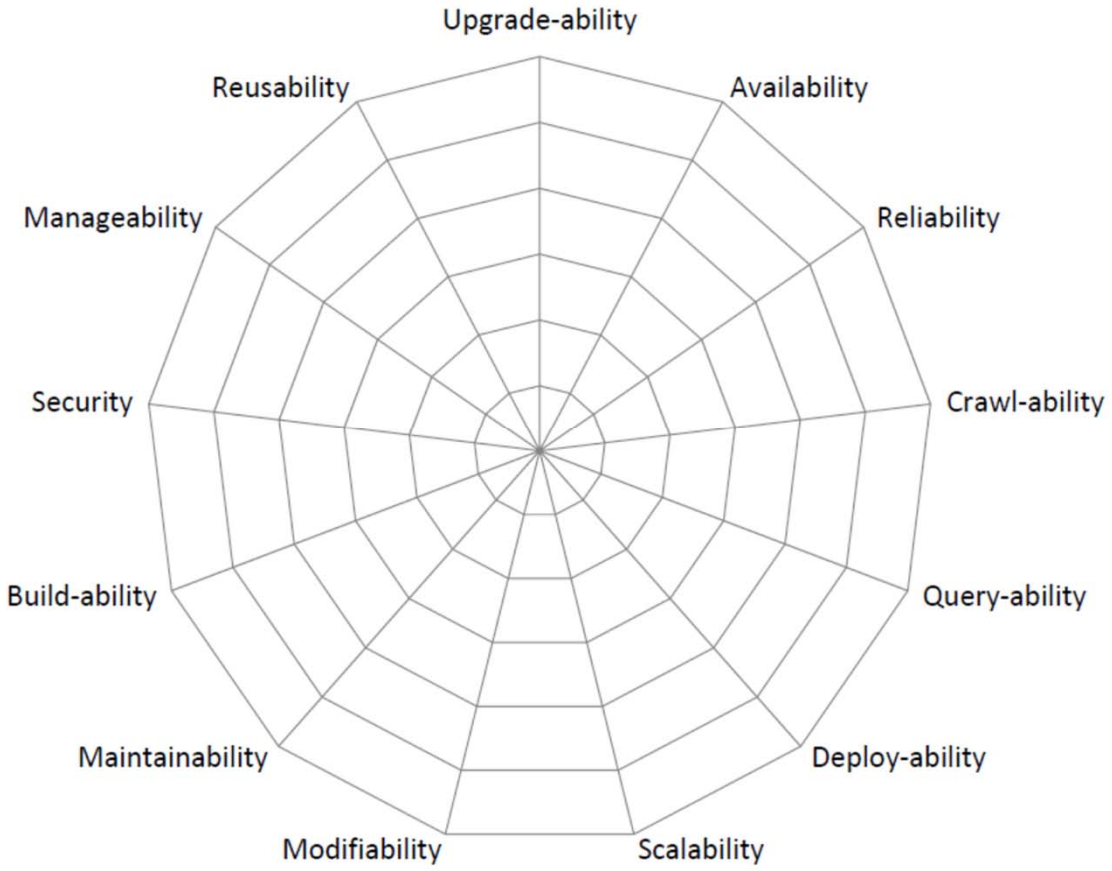
2. Introduction to Quality Attributes, Quality Attributes Taxonomy

- Same properties for different systems



System Properties Web

- Define your own system properties web
  - Select appropriate quality factors for your system under consideration.

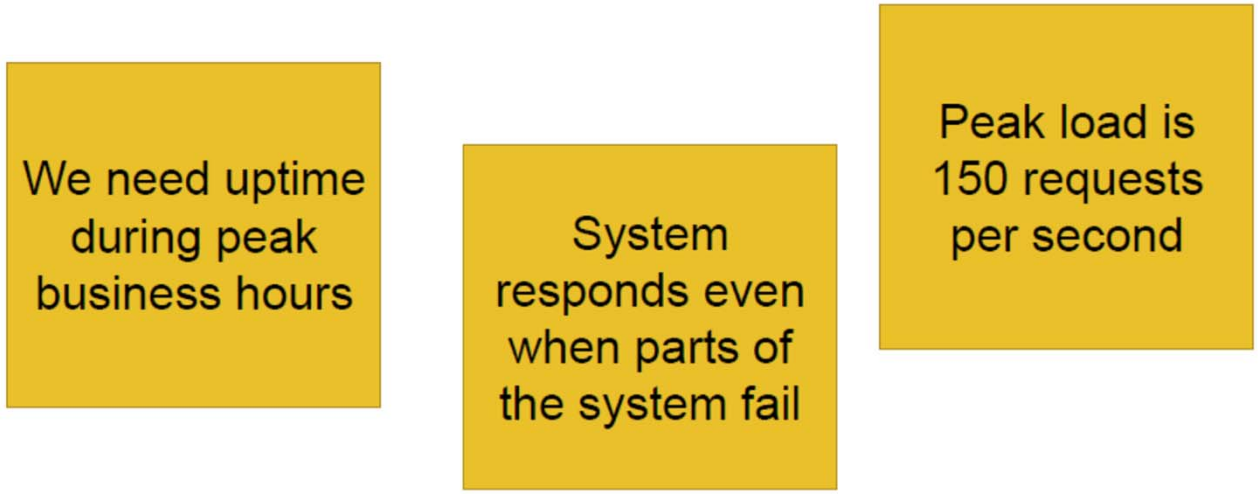


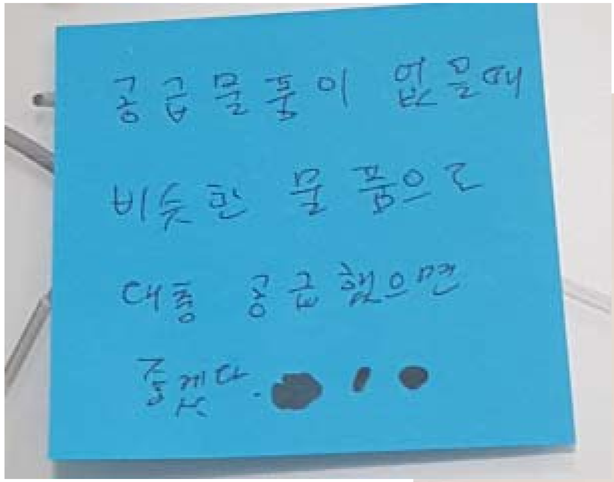
3. Scenario Brainstorming

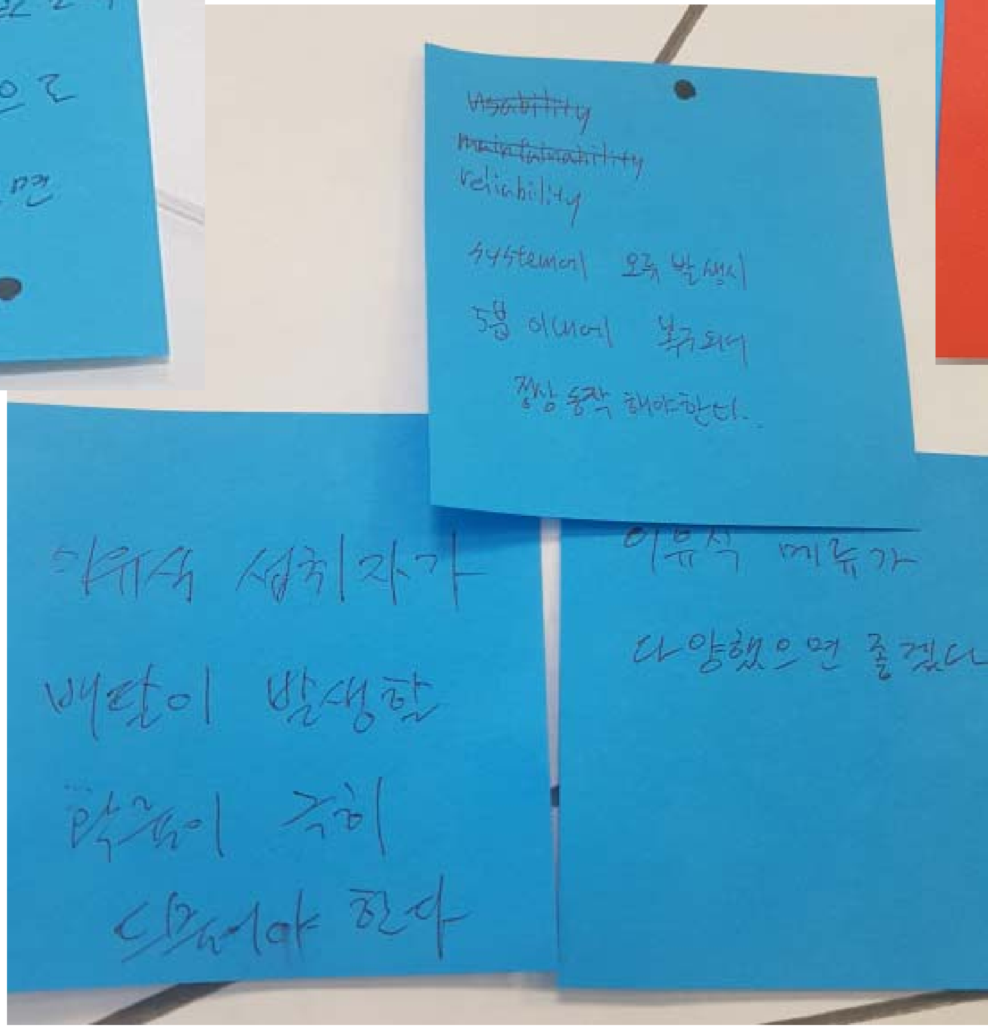
- Objective: Identify raw quality attribute scenarios
- Timing: 30 minutes to 2-3 hours
- Steps:
  1. Start with a Scenario on the web, ask “Is this Quality attribute relevant to your system?”
  2. If Yes, spend 5 minutes brainstorming scenarios / concerns on that scenario.
  3. Write raw scenarios on stickies and put on web
  4. After 5 minutes, move to next scenario

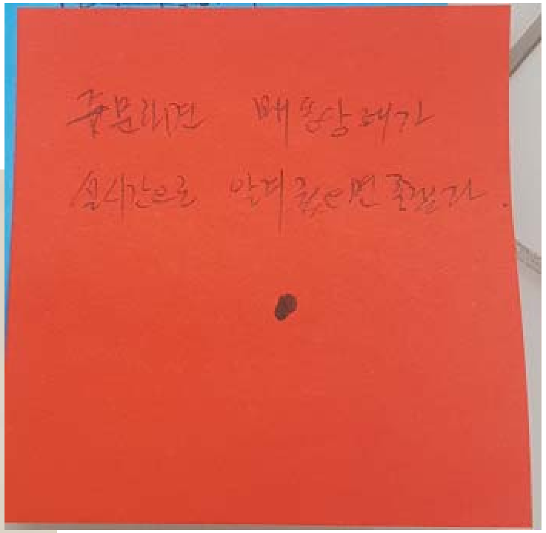
Raw Quality Attribute Scenario

- Informally describes a stakeholder's concern and concrete instances of quality attributes

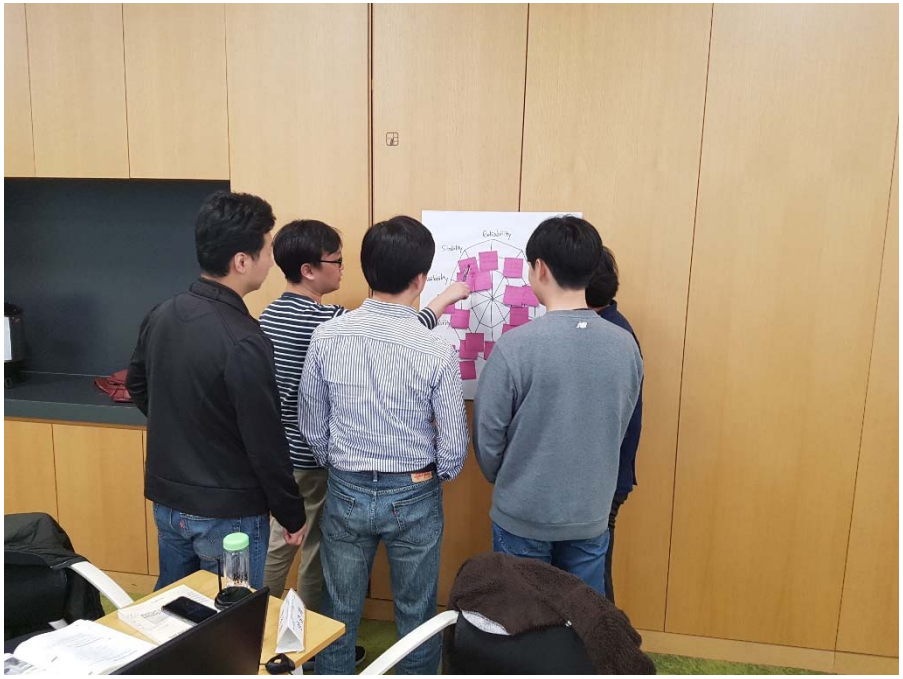


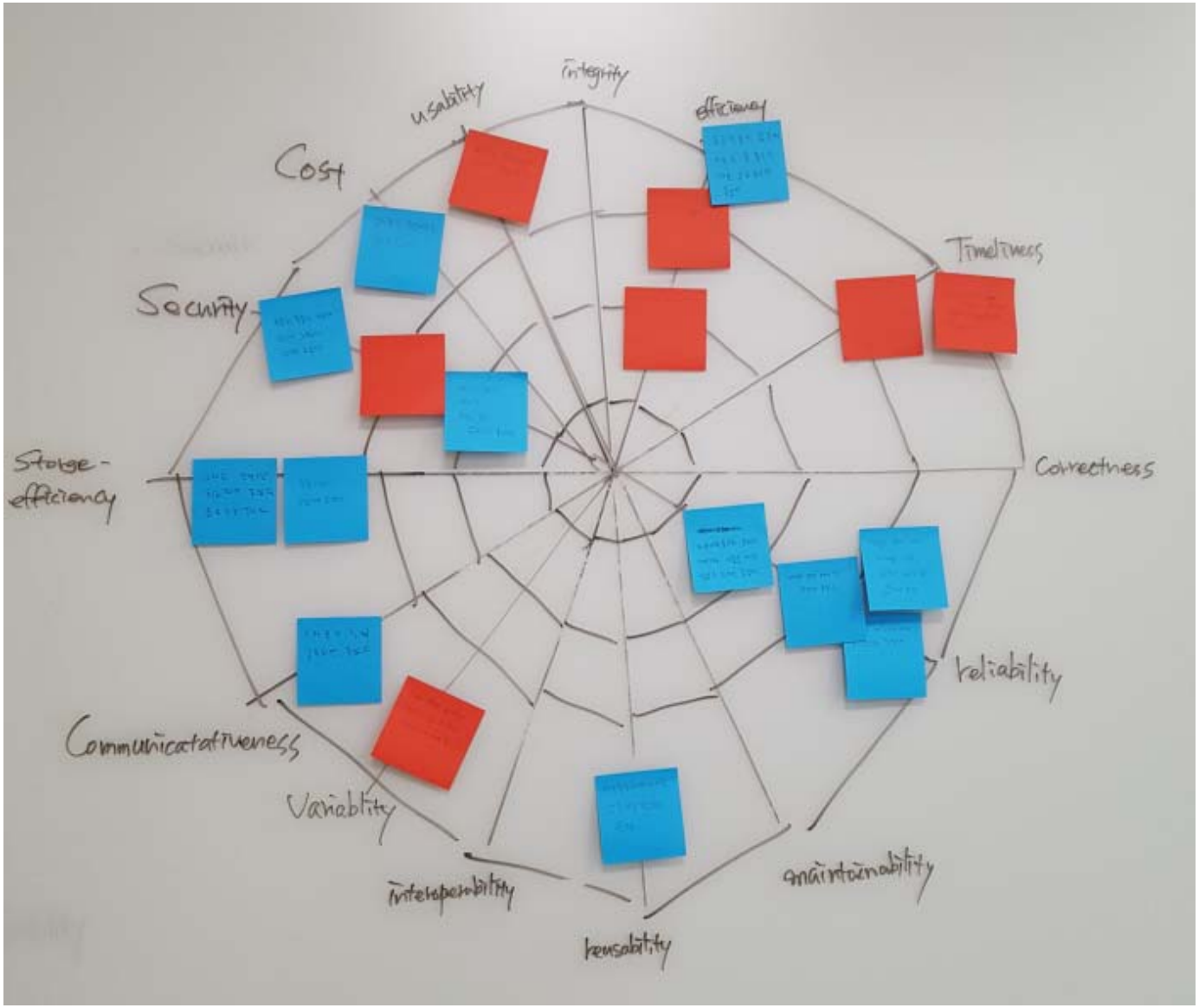

공급물품이 없을때
비슷한 물품으로
대충 공급했으면
좋겠다.


usability
maintainability
reliability
system에 오류 발생시
5분 이내에 복구되어
정상 동작 해야한다.
이유식 섭취자가
배탈이 발생할
드려야 한다
이유식 메뉴가
다양했으면 좋겠다


주문하면 배송상태가
실시간으로 알려줬으면 좋겠다.

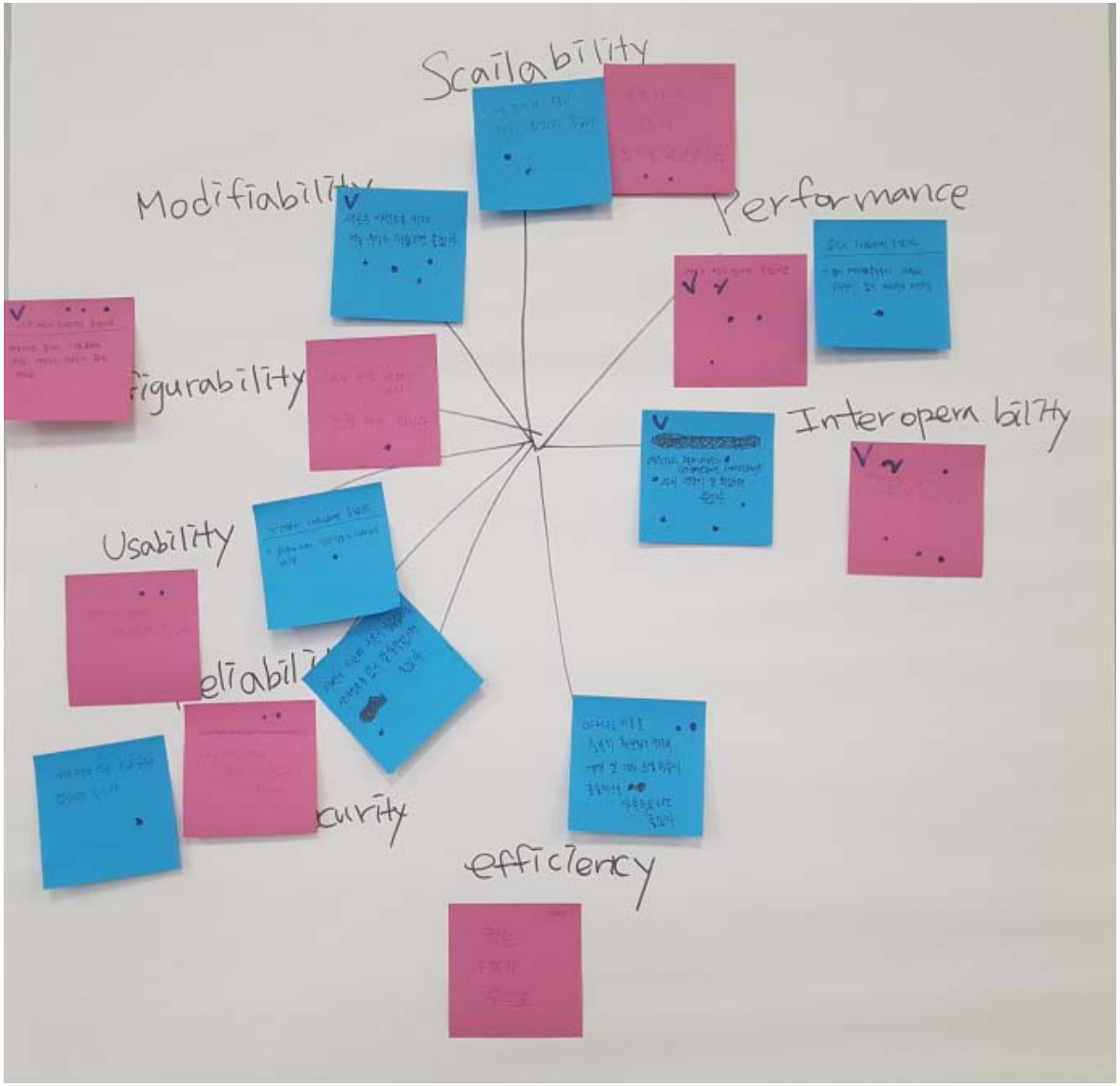
“Walk the System Properties Web” Activity




usability
integrity
efficiency
Cost
Security
Timeliness
Storage-
efficiency
Correctness
Reliability
Communicativeness
Variablity
interoperability
reusability
maintainability

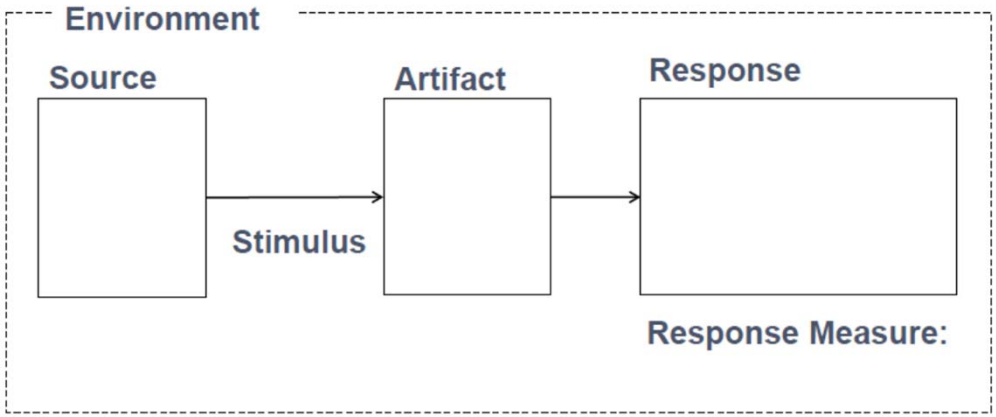
4. Raw Scenario Prioritization

- Objective : Identify Highest Priority Scenarios usingdot voting
- Timing : 5 minutes
- Steps:
  - Dot Voting:
    - Each stakeholder gets n / 3 + 1 dots for scenarios where n = # scenarios
    - 2 votes to choose “top quality attribute”


Scailability
Performance
Interopern bility
Usability
efficiency

5. Scenario Refinement

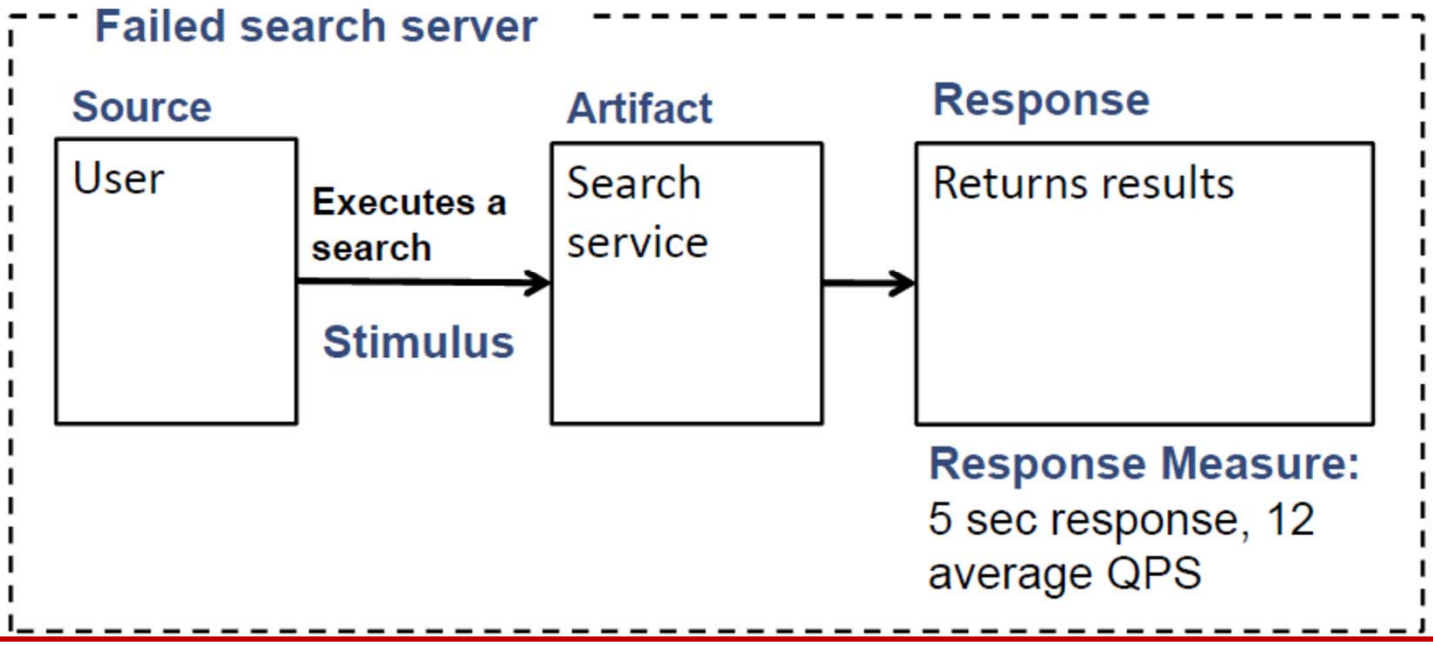
- Objective : Generate Quality Attribute Scenarios based on raw notes
- Timing : 30 - 60 minutes
- Steps :
  1. Start with high priority scenario
  2. Fill out the worksheet, identifying the components of a quality attribute scenario
  3. Complete and present to stakeholders



Availability

Example

Raw Scenario: In the event of hardware failure, search service is expected to return results during normal working hours for US services representatives.



Refined Scenario: In the event of hardware failure, search service is expected to return results within 5 sec, in 12 average QPA (Queries Per Sec)

Mini-QAW vs. Traditional QAW

Mini-QA

- Routine or well understood systems/problems
- Required to minimize upfront costs
- Limited experience with traditional QAW
- Relatively short overall schedule

Traditional QAW

- Higher risk projects
- System or problem is new to team
- Stakeholders prefers traditional methods
- Experienced facilitators available

Exercise 5: Mini-QAW

- Perform the **Mini-QAW** for your "**인증과제**"
  - Follow the steps of Mini-QAW
  - Refine 5 QASs

- **The Mini-QAW steps :**
  1. Mini-QAW Introduction
     - Assign 10 different roles of stakeholders to all team members
     - Define/share the overall context/boundary of the system under consideration (SUC)
  2. Introduction to Quality Attributes, Quality Attributes Taxonomy
     - Select about 10 quality factors relevant to the SUC
  3. Scenario Brainstorming
     - Identifying raw quality attribute scenarios
     - "Walk the System Properties Web" activity
  4. Raw Scenario Prioritization
     - Dot voting to select 5 scenarios
  5. Scenario Refinement
     - Generate 5 well-refined QASs
  6. Review Results with Stakeholders