Quality Attributes

Parinya Ekparinya

Parinya.Ekparinya@gmail.com

Software Architecture and Design

2021 Semester 1

No matter the source, all requirements encompass the following categories:

1. Functional requirements
2. Quality attribute requirements
3. Constraints

Functionality

- Functionality is the ability of the system to do the work for which it was intended.
- Functionality does not determine architecture.
- Functionality is achieved by assigning responsibilities to architectural elements, resulting in one of the most basic of architectural structures.

- Intuitively
  - Functionality is what a product does.
  - Quality is how well it does it.
- The most important qualities are:
  - Availability
  - Interoperability
  - Modifiability
  - Performance
  - Security
  - Testability
  - Usability

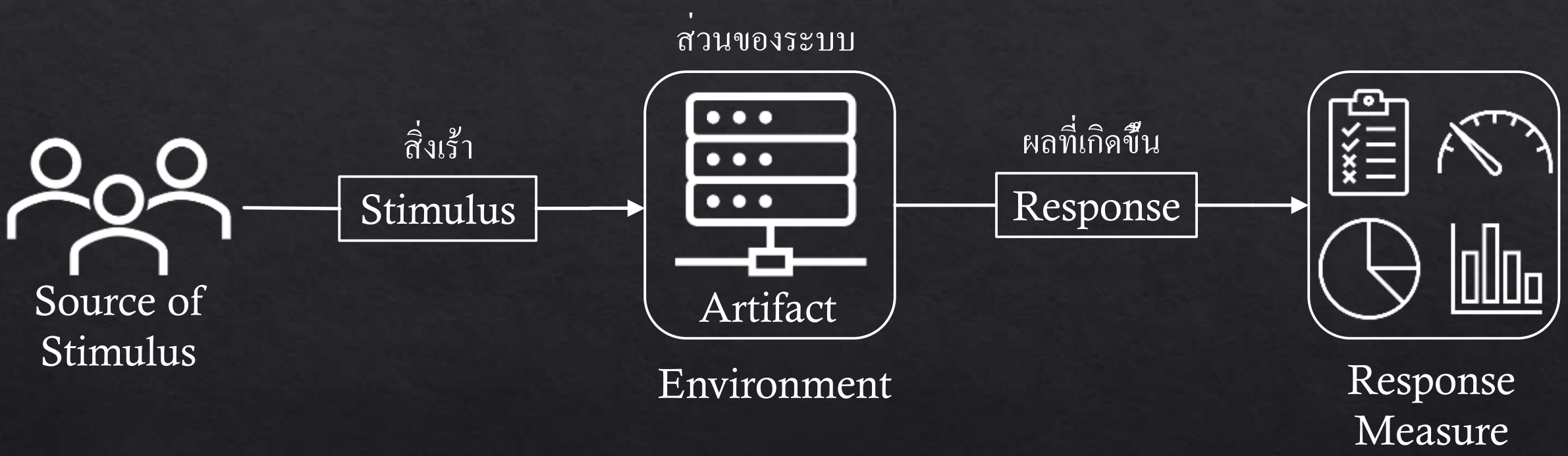
- These are sometimes called “non-functional requirements.” We don’t like that term!

คุณสมบัติที่วัดได้ ตรวจสอบได้

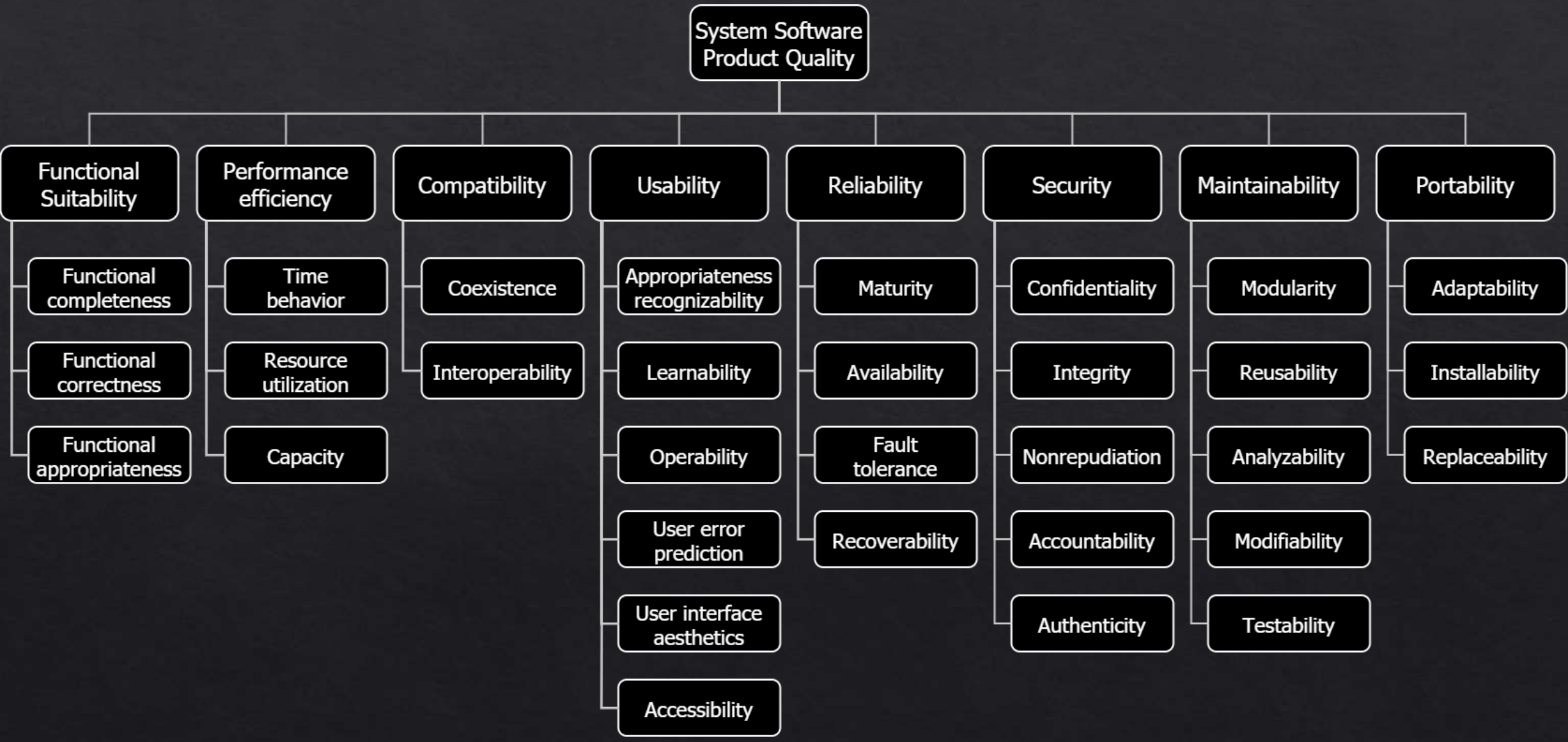
Quality Attributes

- A quality attribute is a measurable or testable property of a system that is used to indicate how well the system satisfies the needs of its stakeholders.
- While functionality describes what the system does, quality describes how well the system does its function.
- Quality attribute scenarios is used to characterizing quality attributes.

Quality Attribute Scenario



ISO/IEC FCD 25010 Product Quality Standard



Availability

Availability

- A failure is the deviation of the system from its specification, where the deviation is externally visible.
- A failure's cause is called a fault.
- A fault can be either internal or external to the system under consideration.
- Faults can be prevented, tolerated, removed, or forecast.
- Availability refers to the ability of a system to mask or repair faults such that the cumulative service outage period does not exceed a required value over a specified time interval.

Availability General Scenario (1)

Source of stimulus	Internal/external: people, hardware, software, physical infrastructure, physical environment
Stimulus	Fault: omission, crash, incorrect timing, incorrect response
Artifacts	Processors, communication channels, persistent storage, processes
Environment	Normal operation, startup, shutdown, repair mode, degraded operation, overloaded operation

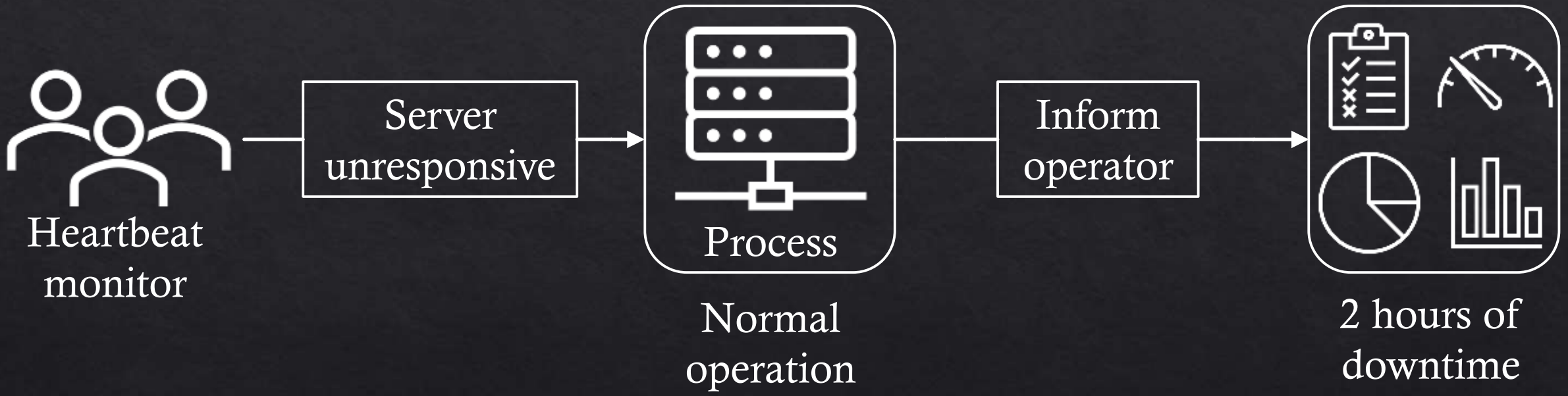
Availability General Scenario (2)

Response	Prevent the fault from becoming a failure Detect the fault: • Log the fault • Notify appropriate entities (people or systems) Recover from the fault: • Disable source of events causing the fault • Be temporarily unavailable while repair is being affected • Fix or mask the fault/failure or contain the damage it causes • Operate in a degraded mode while repair is being affected

Availability General Scenario (3)

Response measure	• Time or time interval when the system must be available • Availability percentage (e.g., 99.999%) • Time to detect the fault • Time to repair the fault • Time or time interval in which system can be in degraded mode • Proportion (e.g., 99%) or rate (e.g., up to 100 per second) of a certain class of faults that the system prevents, or handles without failing

Sample Availability Scenario



Availability Calculation

- Typically, the availability of a system can be measured as the proportion of time it has provided the specified services within required bounds over a specified time interval.
- There is a well-known expression used to derive steady-state availability:

$$\frac{MTBF}{(MTBF + MTTR)}$$

- MTBF refers to the mean time between failures
- MTTR refers to the mean time to repair

System Availability Requirements

Availability	Downtime/90 days	Downtime/365 days
90%	9 days	36 days, 12 hours
95%	4 days, 12 hours	18 days, 6 hours
99%	21 hours, 36 minutes	3 days, 15 hours, 36 minutes
99.9%	2 hours, 9 minutes, 36 seconds	8 hours, 45 minutes, 36 seconds
99.99%	12 minutes, 58 seconds	52 minutes, 34 seconds
99.999%	1 minutes, 18 seconds	5 minutes, 15 seconds
99.9999%	~8 seconds	~32 seconds

Tactics for Availability

Detect faults	Recover from faults		Prevent faults
	Preparation and Repair	Reintroduction	
• Ping/Echo • Monitor • Heartbeat • Timestamp • Sanity Checking • Condition Monitoring • Voting • Exception Detection • Self-Test	• Active redundancy (hot spare) • Passive redundancy (warm spare) • Spare (cold spare) • Exception handling • Rollback • Software upgrade • Retry • Ignore fault behavior • Degradation • Reconfiguration	• Shadow • State re-synchronization • Escalating restart • Non-stop forwarding (NSF)	• Removal from service • Transactions • Predictive model • Exception prevention • Increase competence

ความสามารถในการทำงานร่วมกัน

Interoperability

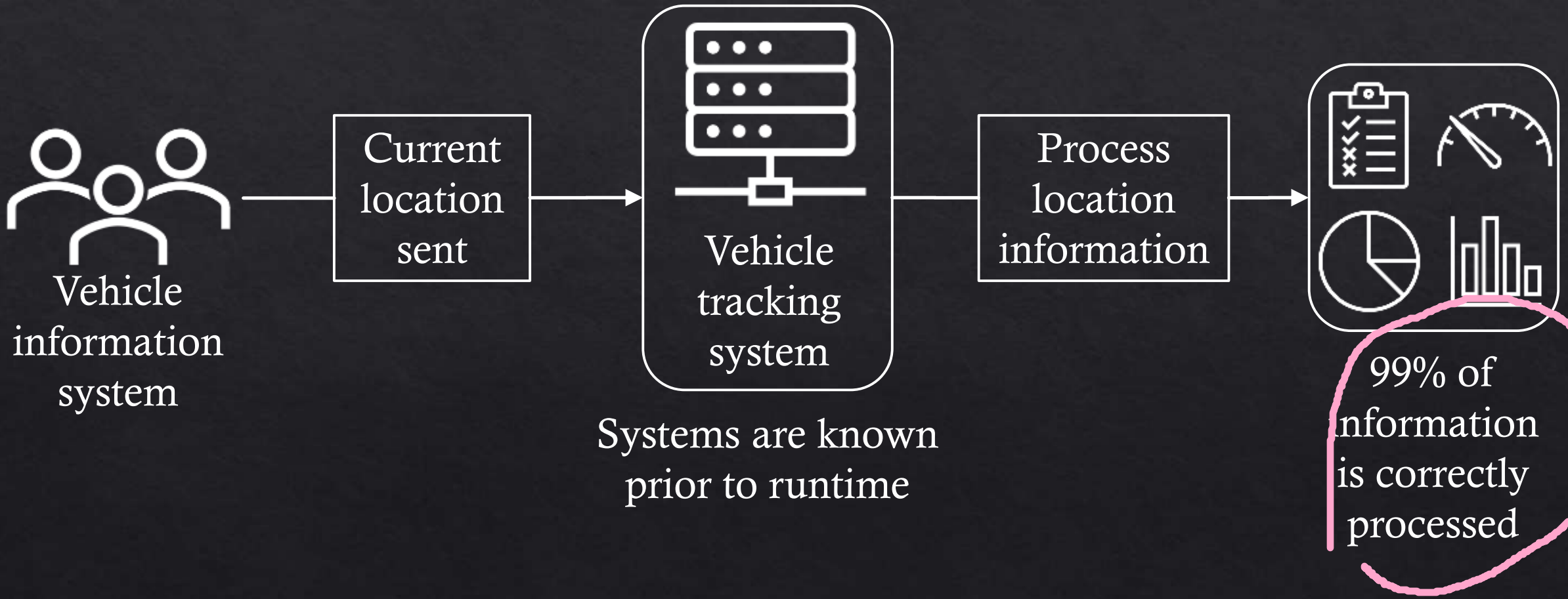
Interoperability

- Interoperability is about the degree to which two or more systems can usefully exchange meaningful information via interfaces in a particular context.
- The definition includes both:
  - the ability to exchange data (syntactic interoperability)
  - the ability to correctly interpret the data being exchanged (semantic interoperability).
- The external systems with which a system will interoperate can either be known or unknown prior to runtime.

Interoperability General Scenario

Source of stimulus	A system that initiates a request.
Stimulus	A request to exchange information among systems.
Artifacts	The systems that wish to interoperate.
Environment	The systems that wish to interoperate are discovered at runtime or are known prior to runtime.
Response	The request to interoperate results in the exchange of information. • The information is understood by the receiving party both syntactically and semantically. • Alternatively, the request is rejected, and appropriate entities are notified.
Response measure	The percentage of information exchanges correctly processed, or the percentage of information exchanges correctly rejected.

Sample Interoperability Scenario



Tactics for Interoperability

Locate	Manage Interfaces
• Discover Service ไม่รู้เซฟปลายทางมาก่อน	• Orchestrate • Tailor Interface

- Discover service – locate a service through various searching techniques.
- Orchestrate – a tactic that uses a control mechanism to coordinate and manage sequence of particular services.
- Tailor interface – a tactic that adds or removes capabilities, such as translation or smoothing data, to an interface.

ความสามารถในการเปลี่ยนแปลง

Modifiability

Modifiability

- A system's modifiability refers to its receptiveness to change.

  McGovern, J., Tyagi, S., Stevens, M., & Mathew, S. (2003). Java web services architecture. Elsevier.

- Change happens:
  - to add new features, to change or even retire old ones.
  - to fix defects, tighten security, or improve performance.
  - to enhance the user’s experience. Changes happen to embrace new technology, new platforms, new protocols, new standards.
  - to make systems work together, even if they were never designed to do so.

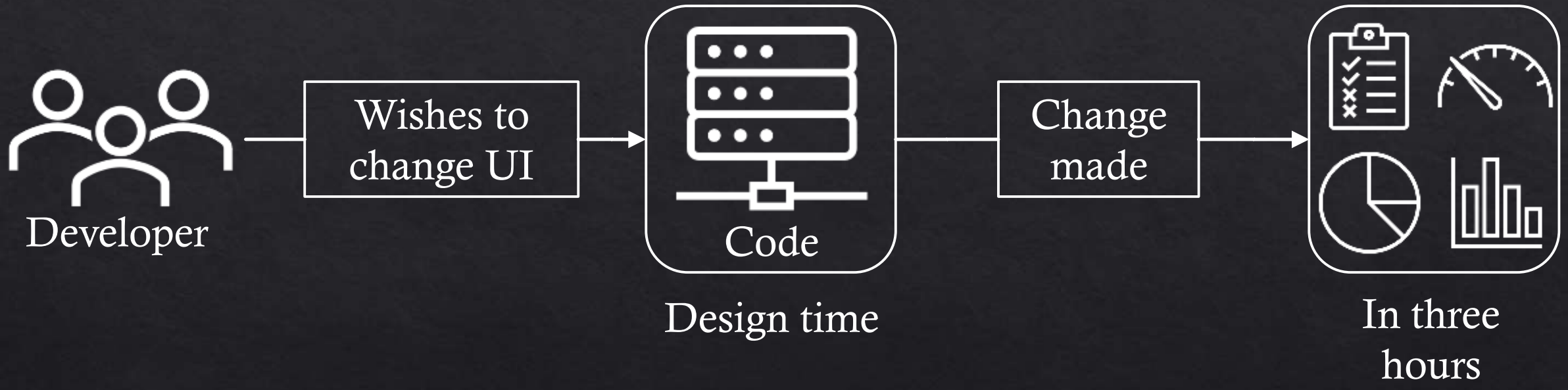
Modifiability General Scenario (1)

Source of stimulus	End user, developer, system administrator
Stimulus	A directive to add/delete/modify functionality, or change a quality attribute, capacity, or technology
Artifacts	Code, data, interfaces, components, resources, configurations, …
Environment	Runtime, compile time, build time, initiation time, design time

Modifiability General Scenario (2)

Response	One or more of the following: • Make modification • Test modification • Deploy modification
Response measure	Cost in terms of the following: • Number, size, complexity of affected artifacts Effort • Calendar time • Money (direct outlay or opportunity cost) • Extent to which this modification affects other functions or quality attributes • New defects introduced

Sample Modifiability Scenario



Tactics for Modifiability

Reduce size of a module	Increase cohesion	Reduce coupling	Defer binding
• Split module	• Increase sematic coherence	• Encapsulate • Use an intermediary • Restrict dependencies • Refactor • Abstract common services	

Cohesion

- Cohesion measures how strongly the responsibilities of a module are related.
- The cohesion of a module is the probability that a change scenario that affects a responsibility will also affect other (different) responsibilities.
- The higher the cohesion, the lower the probability that a given change will affect multiple responsibilities. ดี
- If module A has a low cohesion, then cohesion can be improved by removing responsibilities unaffected by anticipated changes.

Coupling

- Modules have responsibilities. When a change causes a module to be modified, its responsibilities are changed in some way.
- Generally, a change that affects one module is easier and less expensive than if it changes more than one module.
- However, if two modules' responsibilities overlap in some way, then a single change may well affect them both.
- We can measure this overlap by measuring the probability that a modification to one module will propagate to the other. This is called coupling, and high coupling is an enemy of modifiability.

เยอะไม่ดี

Binding time of modification

- We need to be concerned with when in the software development life cycle a change occurs.
- If we ignore the cost of preparing the architecture for the modification, we prefer that a change is bound as late as possible.
- An architecture that is suitably equipped to accommodate modifications late in the life cycle will, on average, cost less than an architecture that forces the same modification to be made earlier.
- Changes can only be successfully made (that is, quickly and at lowest cost) late in the life cycle if the architecture is suitably prepared to accommodate them.

Tactics for Defer Binding

- Compile time or build time:
  - Component replacement (for example, in a build script or makefile)
  - Compile-time parameterization
  - Aspects
- Deployment time:
  - Configuration-time binding
- Startup or initialization time:
  - Resource files
- Runtime:
  - Runtime registration
  - Dynamic lookup (e.g., for services)
  - Interpret parameters
  - Startup time binding
  - Name servers
  - Plug-ins
  - Publish-subscribe
  - Shared repositories
  - Polymorphism

Performance

Performance

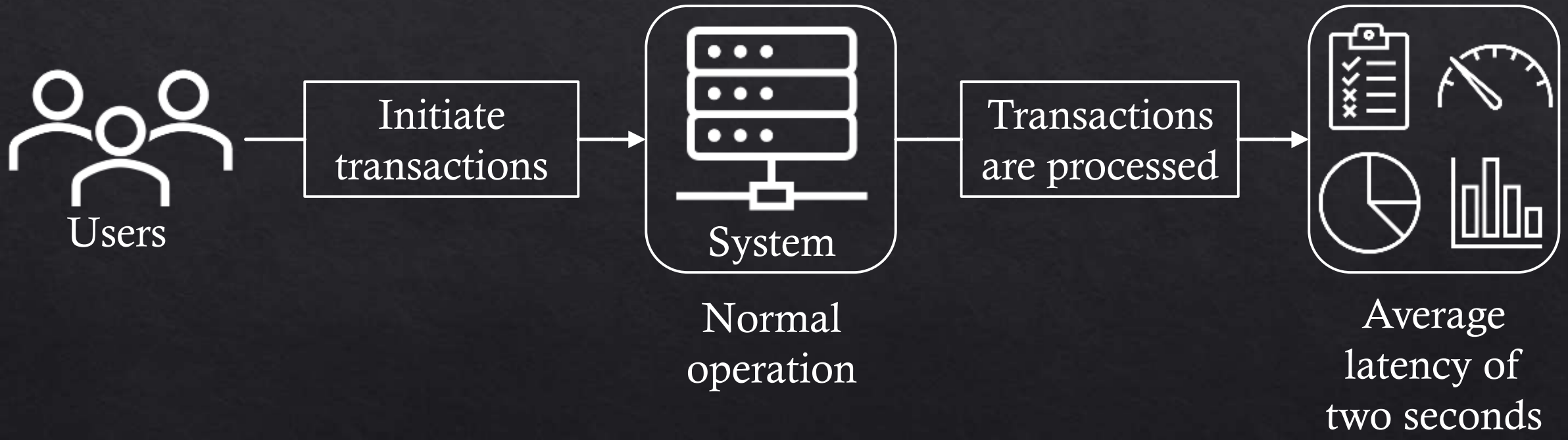
- Performance measures how effective is a software system with respect to time constraints and allocation of resources.

  Cortellessa V., Di Marco A., Inverardi P. (2011) What Is Software Performance?. In: Model-Based Software Performance Analysis. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-13621-4_1

Performance General Scenario

Source of stimulus	Internal or external to the system
Stimulus	Arrival of a periodic, sporadic, or stochastic event ความถี่ของ Request
Artifacts	System or one or more components in the system
Environment	Operational mode: normal, emergency, peak load, overload
Response	Process events, change level of service
Response measure	Latency, deadline, throughput, jitter, miss rate

Sample Performance Scenario



Tactics for Performance

Control resource demand	Manage resources
• Manage sampling rate • Limit event response • Prioritize events • Reduce overhead • Bound execution times • Increase resource efficiency	• Increase resources • Introduce concurrency • Maintain multiple copies of computations • Maintain multiple copies of data • Bound queue sizes • Schedule resources

Security

Security

- Security is a condition that results from the establishment and maintenance of protective measures that enable an organization to perform its mission or critical functions despite risks posed by threats to its use of systems.
- Protective measures may involve a combination of deterrence, avoidance, prevention, detection, recovery, and correction that should form part of the organization's risk management approach.

https://csrc.nist.gov/glossary/term/security

Security General Scenario (1)

Source of stimulus	Human or another system which may have been previously identified (either correctly or incorrectly) or may be currently unknown. A human attacker may be from outside the organization or from inside the organization.
Stimulus	Unauthorized attempt is made to display data, change or delete data, access system services, change the system's behavior, or reduce availability.
Artifacts	System services, data within the system, a component or resources of the system, data produced or consumed by the system
Environment	The system is either online or offline; either connected to or disconnected from a network; either behind a firewall or open to a network; fully operational, partially operational, or not operational.

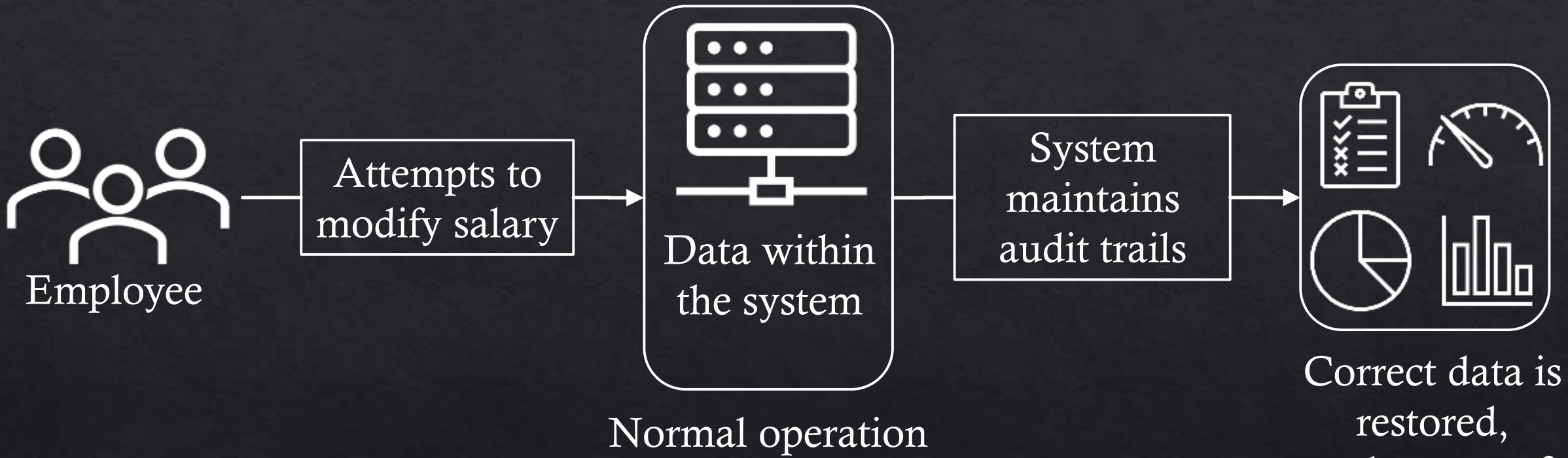
Security General Scenario (2)

Response	Transactions are carried out in a fashion such that • Data or services are protected from unauthorized access. • Data or services are not being manipulated without authorization. • Parties to a transaction are identified with assurance. • The parties to the transaction cannot repudiate their involvements. • The data, resources, and system services will be available for legitimate use. The system tracks activities within it by • Recording access or modification • Recording attempts to access data, resources, or services • Notifying appropriate entities (people or systems) when an apparent attack is occurring

Security General Scenario (3)

Response measure	One or more of the following: • How much of a system is compromised when a particular component or data value is compromised • How much time passed before an attack was detected • How many attacks were resisted • How long does it take to recover from a successful attack • How much data is vulnerable to a particular attack

Sample Security Scenario



Tactics for Security

Detect attacks	Resist attacks	React to attacks	Recover from attacks
• Detect intrusion • Detect service denial • Verify message integrity • Detect message delay	• Identity actors • Authenticate actors • Authorize actors • Limit access • Limit exposure • Encrypt data • Separate entities • Change default settings	• Revoke access • Lock computer • Inform actors	• Maintain audit trail • Restore

Testability

Testability

- Software testability refers to the ease with which software can be made to demonstrate its faults through (typically execution-based) testing.
- Specifically, testability refers to the probability, assuming that the software has at least one fault, that it will fail on its next test execution.
- Intuitively, a system is testable if it "gives up" its faults easily. If a fault is present in a system, then we want it to fail during testing as quickly as possible.

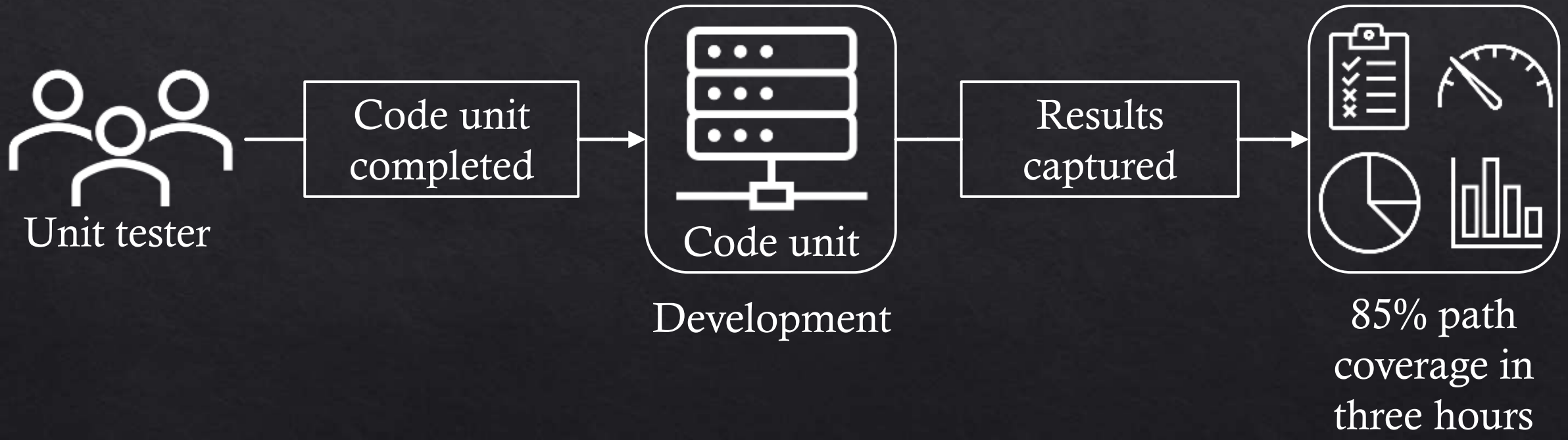
Testability General Scenario (1)

Source of stimulus	Unit testers, integration testers, system testers, acceptance testers, end users, either running tests manually or using automated testing tools
Stimulus	A set of tests is executed due to the completion of a coding increment such as a class layer or service, the completed integration of a subsystem, the complete implementation of the whole system, or the delivery of the system to the customer.
Artifacts	The portion of the system being tested
Environment	Design time, development time, compile time, integration time, deployment time, run time

Testability General Scenario (2)

Response	One or more of the following: • Execute test suite and capture results • Capture activity that resulted in the fault • Control and monitor the state of the system
Response measure	One or more of the following: • Effort to find a fault or class of faults • Effort to achieve a given percentage of state space coverage • Probability of fault being revealed by the next test • Time to perform tests • Effort to detect faults • Length of longest dependency chain in test • Length of time to prepare test environment • Reduction in risk exposure (size(loss) × prob(loss))

Sample Testability Scenario



Tactics for Testability

Control and observe system state	Limit complexity
• Specialized interfaces • Record/Playback • Localize state storage • Abstract data sources • Sandbox • Executable assertions	• Limit structural complexity • Limit nondeterminism

Usability

Usability

- Usability is concerned with how easy it is for the user to accomplish a desired task and the kind of user support the system provides.
- Usability comprises the following areas:
  - Learning system features
  - Using a system efficiently
  - Minimizing the impact of errors
  - Adapting the system to user needs
  - Increasing confidence and satisfaction

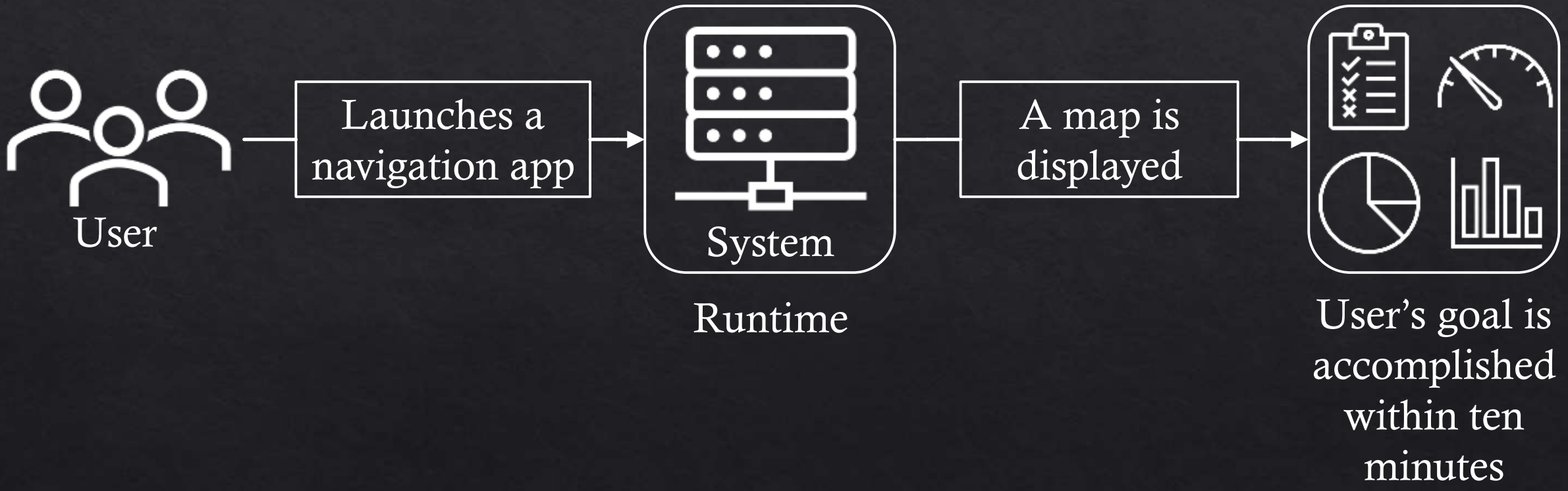
Usability General Scenario (1)

Source of stimulus	End user, possibly in a specialized role
Stimulus	End user tries to use a system efficiently, learn to use the system, minimize the impact of errors, adapt the system, or configure the system.
Artifacts	System or the specific portion of the system with which the user is interacting
Environment	Runtime or configuration time
Response	The system should either provide the user with the features needed or anticipate the user's needs.

Usability General Scenario (2)

Response measure	One or more of the following: • Task time • Number of errors • Number of tasks accomplished • User satisfaction • Gain of user knowledge • Ratio of successful operations to total operations • Amount of time or data lost when an error occurs

Sample Usability Scenario



Tactics for Usability

Support user initiative	Support system initiative
• Cancel • Undo • Pause/Resume • Aggregate	• Maintain task model • Maintain user model • Maintain system model

Support System Initiative

- When the system takes the initiative, it must rely on a model of the user, the task being undertaken by the user, or the system state itself.
- Each model requires various types of input to accomplish its initiative.
- The support system initiative tactics are those that identify the models the system uses to predict either its own behavior or the user's intention.

Summary

- Quality Attribute Scenario
- Availability
- Interoperability
- Modifiability
- Performance
- Security
- Testability
- Usability