Assignment 1: Quality Attribute Analysis

刘鹏程

171250552

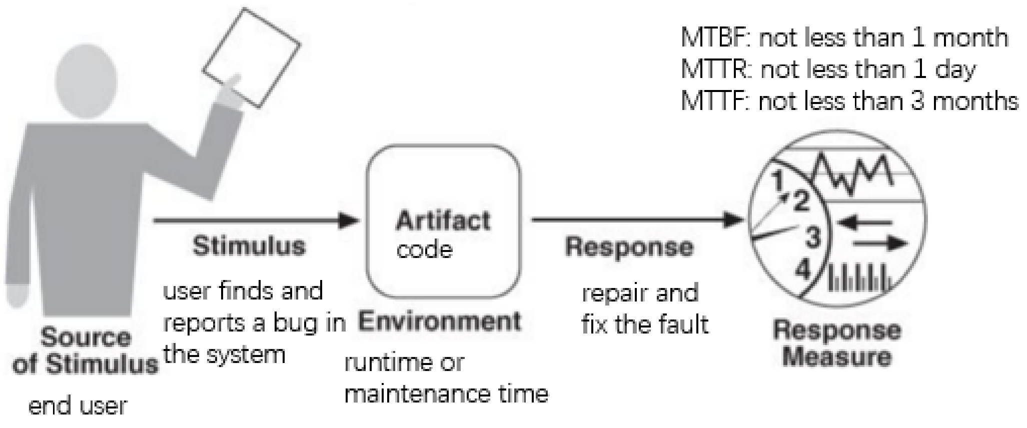
一、dependability vs. security

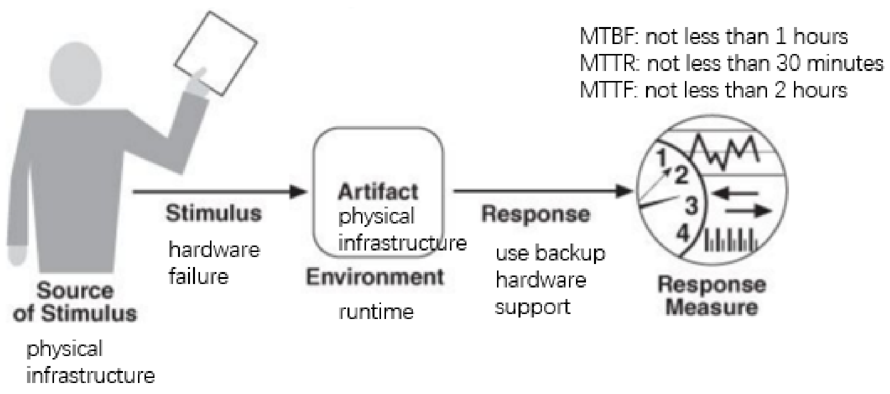
1、general scenario and concrete scenario

- **dependability** general scenario
    -

Portion of Scenario	Possible Values
Source	Large scale, complex systems with availability, safety and security requirements, physical infrastructure
Stimulus	fault: incorrect operation, crash, hidden bugs failure: hardware failure, components failure
Artifact	Code, data persistence, components, redundant storage, hardware support, application interfaces
Environment	Malicious operation, normal operation, startup, shutdown, repair mode, degraded operation, overloaded operation, running time
Response	prevent the fault from becoming the failure. i.e. log the fault; disable the events causing the fault; repair and fix the fault; bug detecting and fixing reduce the cost brought by the failure. i.e. use backup power/storage; temporarily stop related service; operate in a degraded mode while repairing
Response Measure	Time interval it takes to run normally again when facing a fault/failure MTBF: Mean Time Between Failures MTTR: Mean Time to Repair MTTF: Mean Time to Failure overall cost during the period of repairing and fixing

- **dependability** concrete scenario

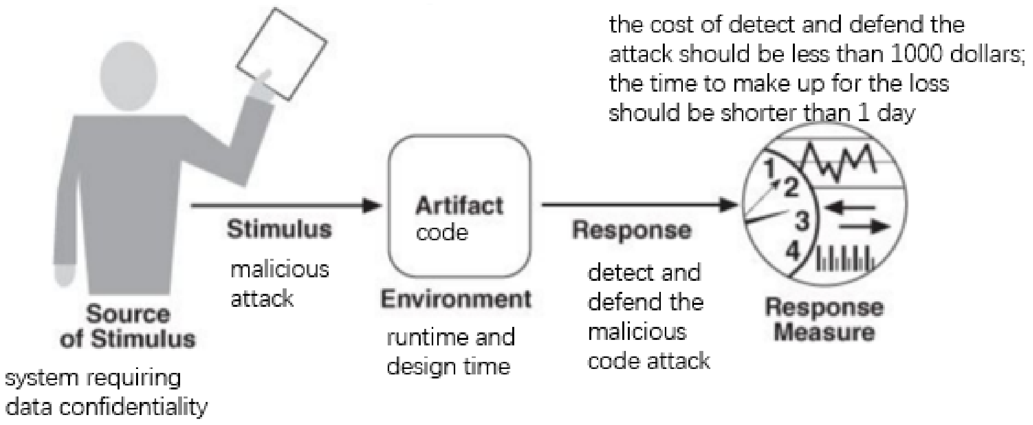

MTBF: not less than 1 month
MTTR: not less than 1 day
MTTF: not less than 3 months
Stimulus
user finds and reports a bug in the system
Artifact
code
Environment
runtime or maintenance time
Response
repair and fix the fault
Response Measure
Source of Stimulus
end user

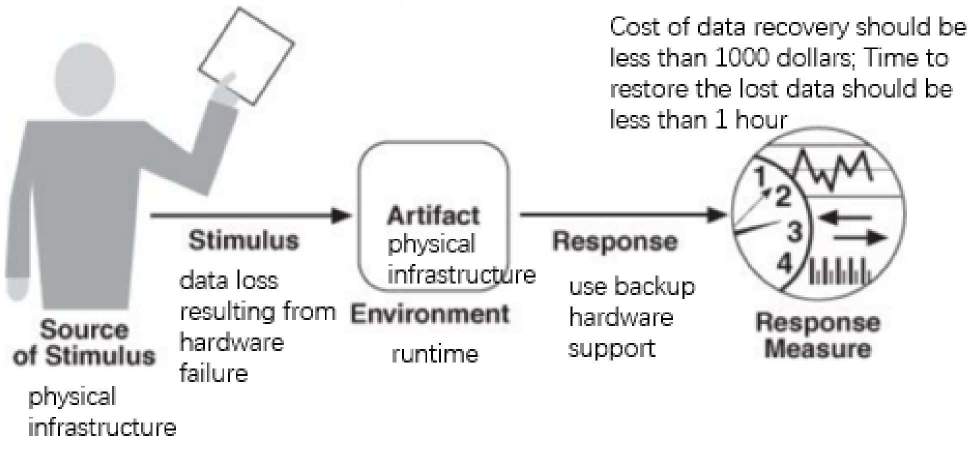


- **security** general scenario
    -

Portion of Scenario	Possible Values
Source	End user, system administrator, system requiring data confidentiality
Stimulus	Malicious code/attack, identity verification, request permission, authentication and authorization
Artifact	Code, data, security application interfaces, data encryption
Environment	Runtime, startup, maintenance time, interaction time
Response	Detect and defend the malicious code attack Reply to users' or external entities' request for permission Prepare for accidents that may cause damage and loss to data Test for system defect which may cause data leakage
Response Measure	Time to complete the verification process Cost of data recovery Time and cost to detect and defend the malicious attack Time and cost to make up for the defect of security mechanism

- **security** concrete scenario


the cost of detect and defend the attack should be less than 1000 dollars; the time to make up for the loss should be shorter than 1 day
Source of Stimulus
system requiring data confidentiality
Stimulus
malicious attack
Artifact
code
Environment
runtime and design time
Response
detect and defend the malicious code attack
Response Measure

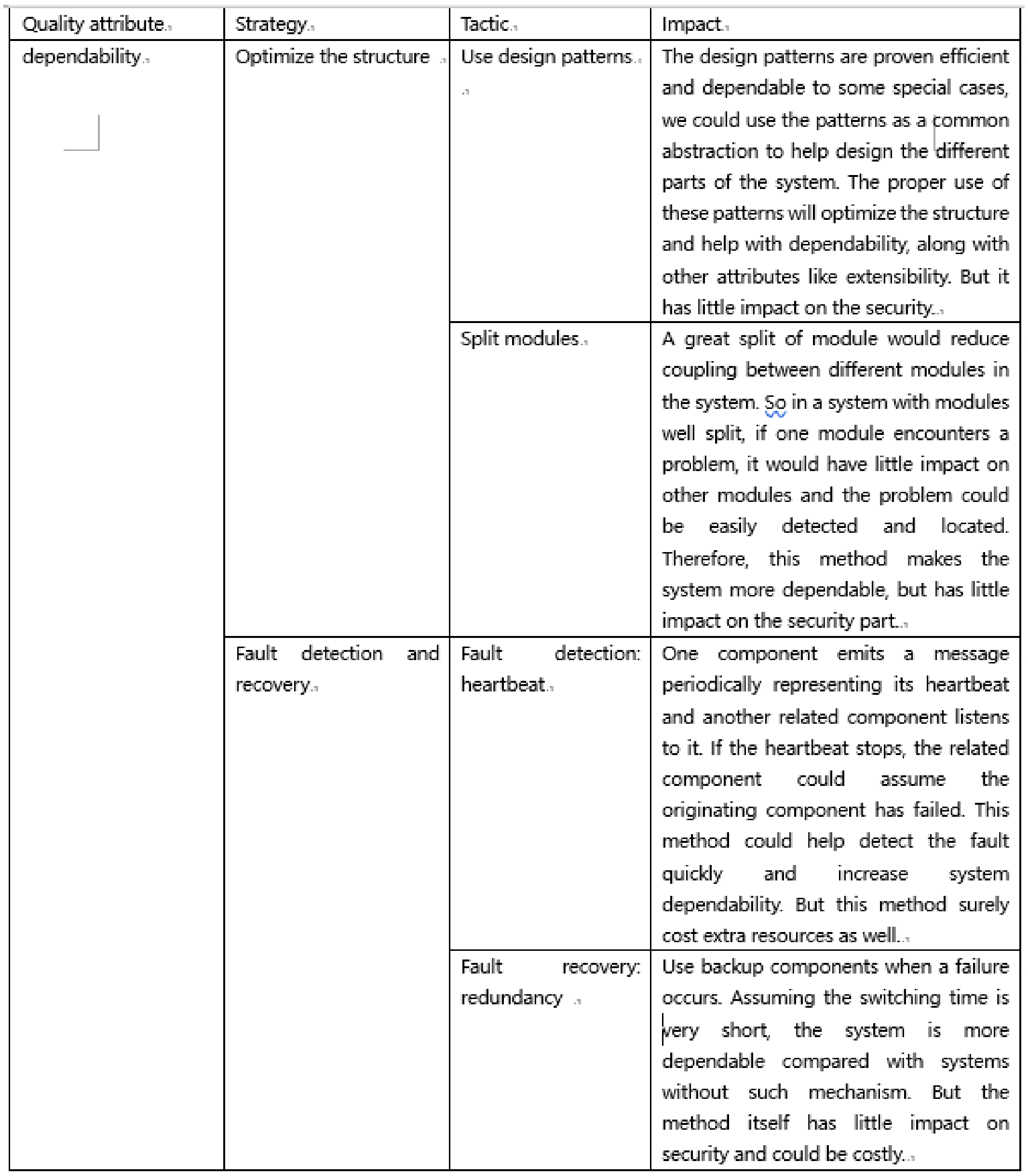

Cost of data recovery should be less than 1000 dollars; Time to restore the lost data should be less than 1 hour
Source of Stimulus
physical infrastructure
Stimulus
data loss resulting from hardware failure
Artifact
physical infrastructure
Environment
runtime
Response
use backup hardware support
Response Measure

2、relationships and differences

- definitions
    - **dependability** in software engineering means the ability to provide services that can defensibly be trusted within a time-period. This may also encompass mechanisms designed to increase and maintain the dependability of a system or software. It reflects the extent of the user's confidence that it will operate as users expect.
    - **security** in software engineering means the ability to continue to function correctly under malicious attack or other hacker risks
- relationships
    - **dependability** covers the related systems attributes of reliability, availability and **security**
    - dependability and security both try to make the system run as usual under malicious attack or other hacker risks. e.g. when some malicious code request to access some top secret database, a dependable system or a secure system will act correspondingly to block such requests. Thus both of them share similar features such as privacy protection. On the other hand, if a system is unable to protect its data from being attacked, it is insecure and cannot be depended on in many cases.
- differences
    - some special dependable systems don't have to satisfy the strict security attribute. e.g. some systems that are very simple may be available only within a small group. There is no need to satisfy the strict security rules because no information about this system has to be secure. This system can gain its dependability by satisfying other related factors.
    - a system that only satisfies the security attribute is far from being dependable. e.g. an accidental hardware failure may make a highly secure system become available, which may lead to the system losing its dependability.

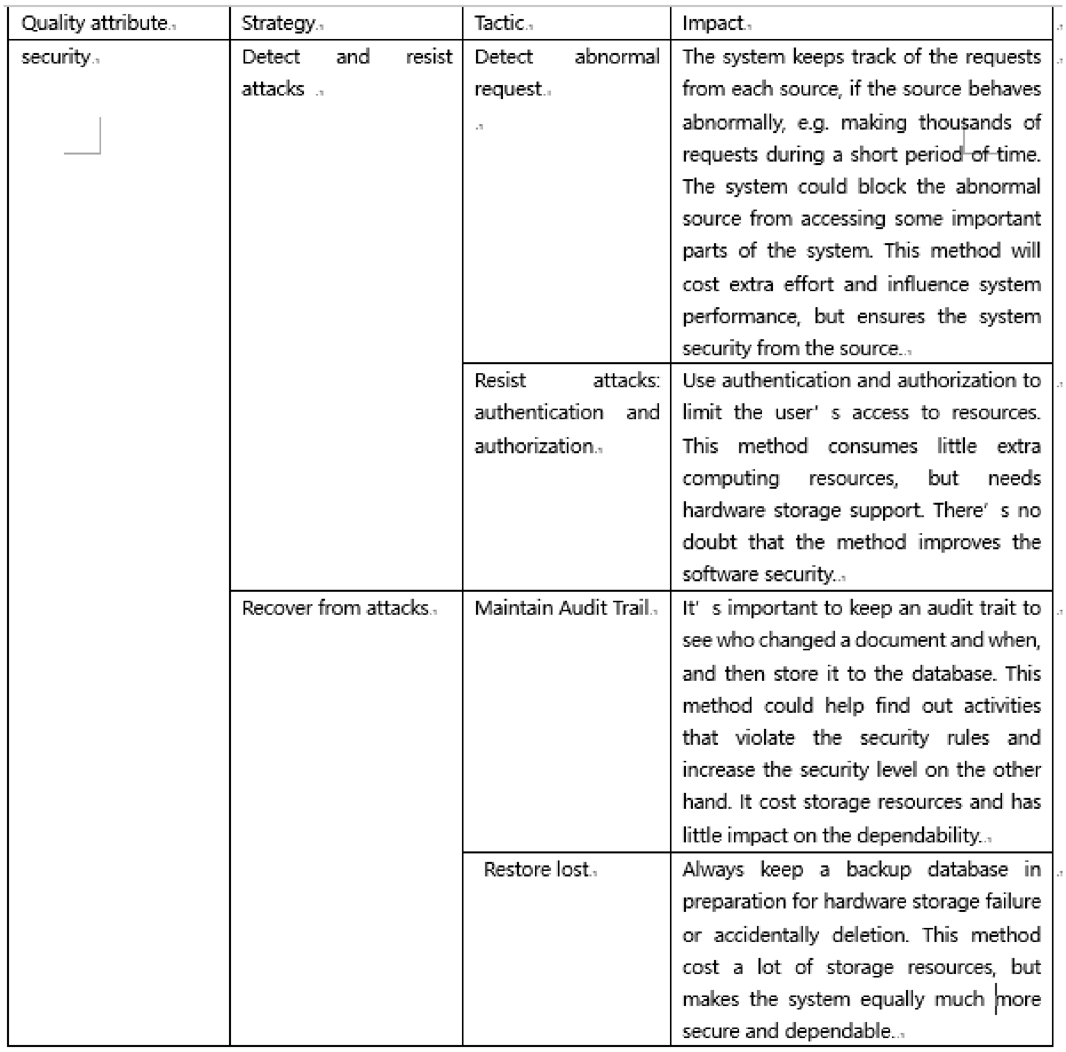
3、strategies and tactics

- dependability



Quality attribute	Strategy	Tactic	Impact
dependability	Optimize the structure	Use design patterns	The design patterns are proven efficient and dependable to some special cases, we could use the patterns as a common abstraction to help design the different parts of the system. The proper use of these patterns will optimize the structure and help with dependability, along with other attributes like extensibility. But it has little impact on the security.
		Split modules	A great split of module would reduce coupling between different modules in the system. So in a system with modules well split, if one module encounters a problem, it would have little impact on other modules and the problem could be easily detected and located. Therefore, this method makes the system more dependable, but has little impact on the security part.
	Fault detection and recovery	Fault detection: heartbeat	One component emits a message periodically representing its heartbeat and another related component listens to it. If the heartbeat stops, the related component could assume the originating component has failed. This method could help detect the fault quickly and increase system dependability. But this method surely cost extra resources as well.
		Fault recovery: redundancy	Use backup components when a failure occurs. Assuming the switching time is very short, the system is more dependable compared with systems without such mechanism. But the method itself has little impact on security and could be costly.

- security



Quality attribute	Strategy	Tactic	Impact
security	Detect and resist attacks	Detect abnormal request.	The system keeps track of the requests from each source, if the source behaves abnormally, e.g. making thousands of requests during a short period of time. The system could block the abnormal source from accessing some important parts of the system. This method will cost extra effort and influence system performance, but ensures the system security from the source.
		Resist attacks: authentication and authorization.	Use authentication and authorization to limit the user' s access to resources. This method consumes little extra computing resources, but needs hardware storage support. There' s no doubt that the method improves the software security.
	Recover from attacks	Maintain Audit Trail	It' s important to keep an audit trait to see who changed a document and when, and then store it to the database. This method could help find out activities that violate the security rules and increase the security level on the other hand. It cost storage resources and has little impact on the dependability.
		Restore lost	Always keep a backup database in preparation for hardware storage failure or accidentally deletion. This method cost a lot of storage resources, but makes the system equally much more secure and dependable.

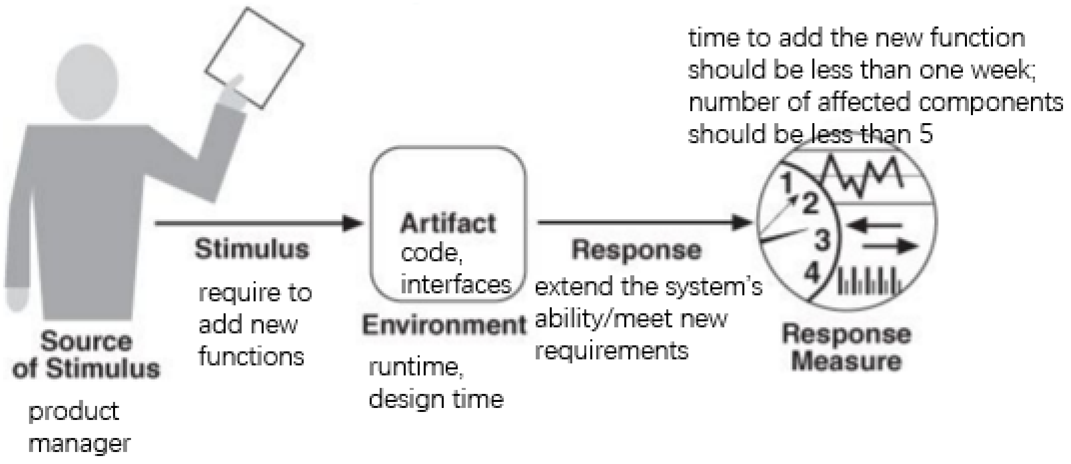
二、sustainability vs. scalability

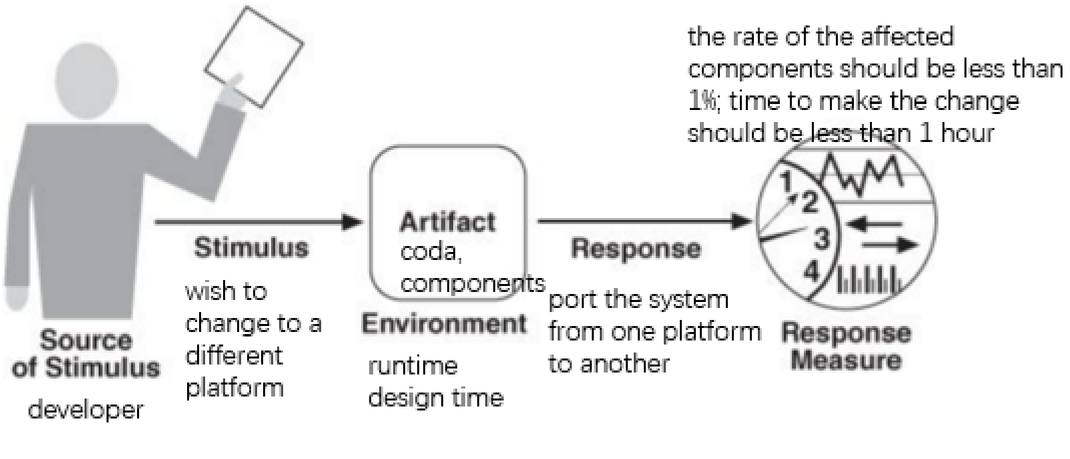
1、general scenario and concrete scenario

- **sustainability** general scenario

Portion of Scenario	Possible Values
Source	a system with the ability to be modified based on stakeholders' changing requirements, end users, developer, system administrator, product manager
Stimulus	Activities related to software evolution, changes of product needs
Artifact	Code, data, interfaces, components, resources, configurations...
Environment	runtime, compile time, build time, initiation time, design time
Response	Extend the system's ability or meet new requirements Locate and fix an error in the system Port the system from one platform to another
Response Measure	Time and cost to sustain a software system on an average day Number, size, rate, complexity of affected artifacts when sustaining the system Complexity of new changes or requirements introduced Time and money to add new components or making changes to the system

- **sustainability** concrete scenario

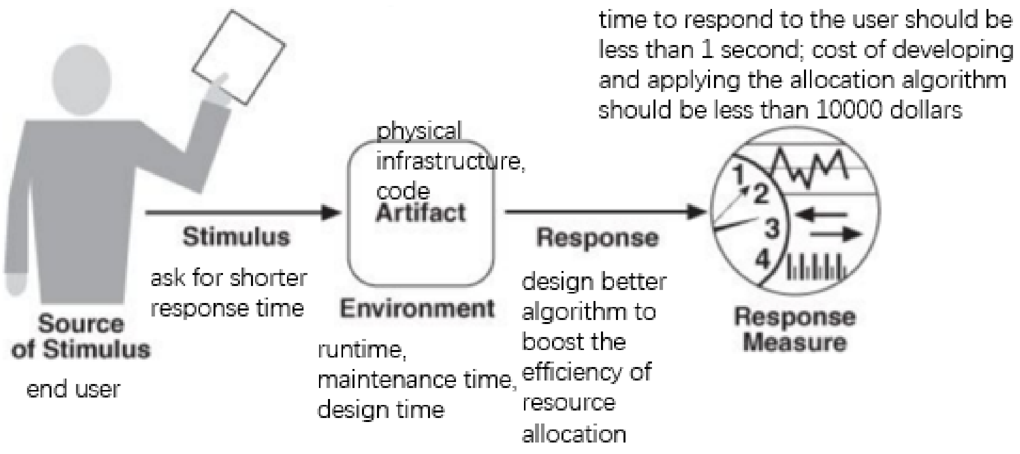

time to add the new function should be less than one week; number of affected components should be less than 5
Source of Stimulus
product manager
Stimulus
require to add new functions
Artifact
code, interfaces
Environment
runtime, design time
Response
extend the system's ability/meet new requirements
1
2
3
4
Response Measure

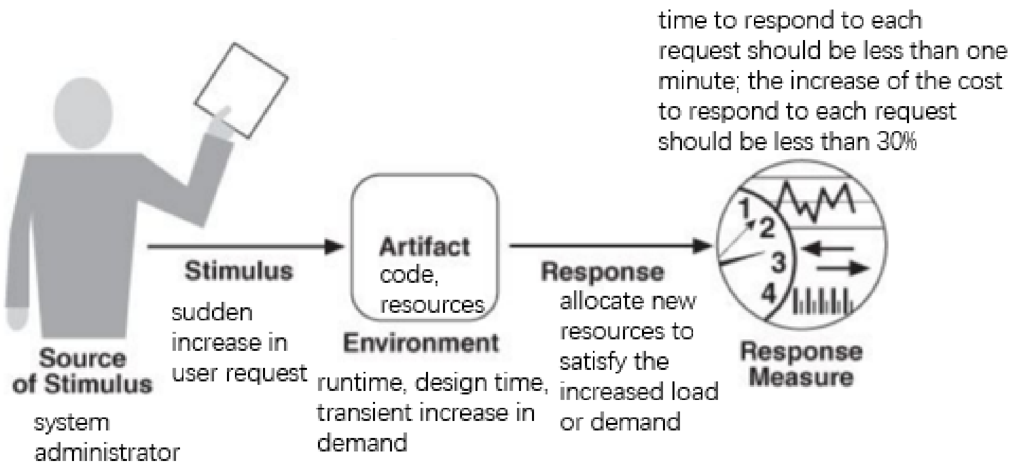

the rate of the affected components should be less than 1%; time to make the change should be less than 1 hour
Source of Stimulus
developer
Stimulus
wish to change to a different platform
Artifact
coda, components
Environment
runtime design time
Response
port the system from one platform to another
1
2
3
4
Response Measure

- **scalability** general scenario

Portion of Scenario	Possible Values
Source	A system with the ability to meet the increase of resource demand or amount of work, end users, developer, system administrator
Stimulus	Increase in demand on a system resource such as processing ability, I/O, storage
Artifact	Code, data, interfaces, components, resources, configurations...
Environment	Increase in load/demand is transient Increase in load/demand is permanent runtime, design time
Response	Provides new resources to satisfy the increased load or demand Free and reallocate old resources for the new increased load or demand
Response Measure	Time and cost to provide new resources to the increased load or demand Time and cost to free and reallocate old resources to the increased load or demand

- **scalability** concrete scenario


time to respond to the user should be less than 1 second; cost of developing and applying the allocation algorithm should be less than 10000 dollars
physical infrastructure, code
Artifact
Stimulus
ask for shorter response time
Response
design better algorithm to boost the efficiency of resource allocation
Environment
runtime, maintenance time, design time
Source of Stimulus
end user
Response Measure

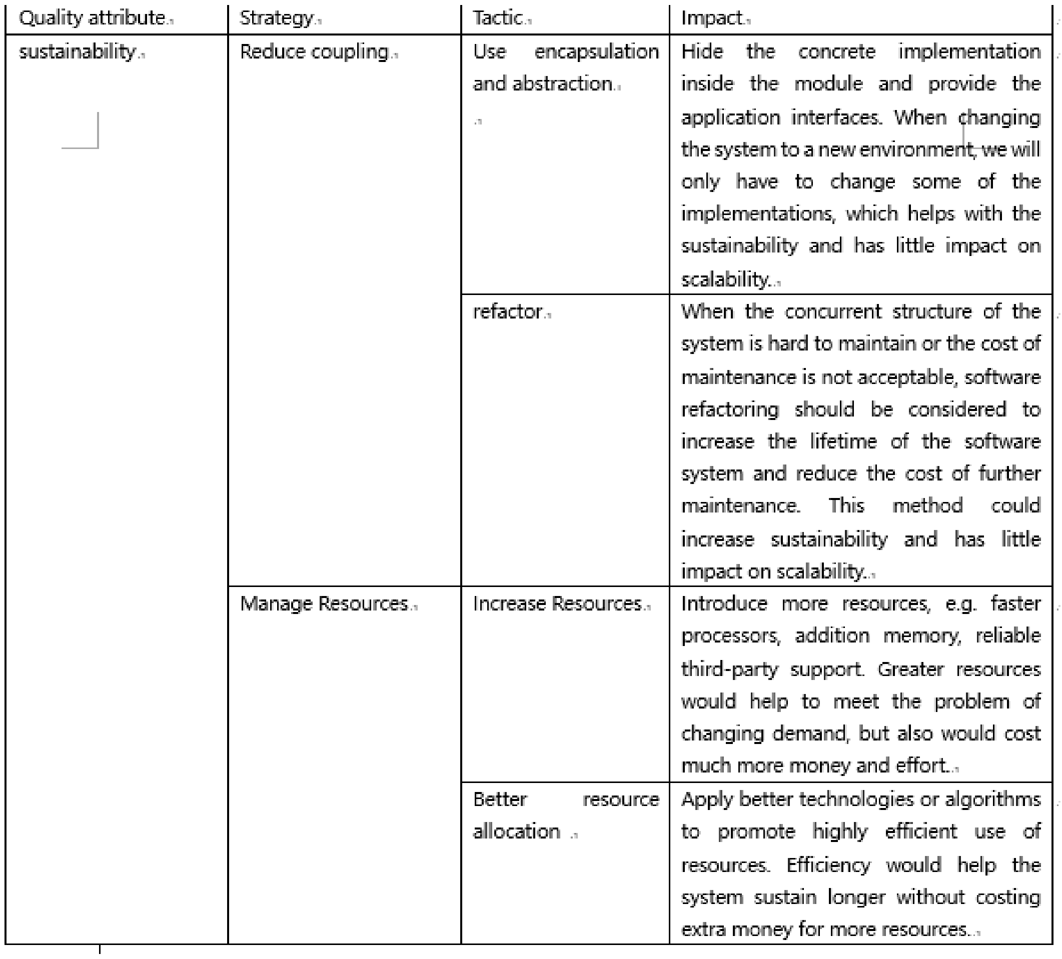

time to respond to each request should be less than one minute; the increase of the cost to respond to each request should be less than 30%
Artifact
code, resources
Stimulus
sudden increase in user request
Response
allocate new resources to satisfy the increased load or demand
Environment
runtime, design time, transient increase in demand
Source of Stimulus
system administrator
Response Measure

2、relationships and differences

- definitions
  - **sustainability** in software engineering means the ability to modify a software system based on stakeholders' changing requirements. It is also described as a mode of software development in which resource use aims to meet product software needs while ensuring the sustainability of natural systems and the environment
  - **scalability** in software engineering means the ability to handle a growing amount of work/demand/load by adding resources to the system. in other words, it means the extent to which software can accommodate horizontal or vertical growth.
- relationships
  - the software **sustainability** can be defined as a measure of a systems' extensibility, interoperability, maintainability, portability, reusability, **scalability**, and usability.
  - sustainability and scalability work together to make the system work normally through a long period of time. Sustainability needs some features of scalability to tackle some strict and unnormal conditions. Scalability ensures that the system can sustain longer and stronger.
- differences
  - suppose that we are developing a ticket reservation system called TRS. TRS has to run normally throughout the whole year except for some special conditions such as Spring Festival travel rush in China. Besides, we hope this system will provide more services in the future, like proposing a travel plan for the visitors, which may consume extra computing and storage effort.
  - if TRS system is scalable, it should be designed to meet the most serious situation such as millions of request happening at the same time. However, scalability alone wouldn't help to add new functions to the software system.
  - if TRS system is sustainable, it should be quickly adjusted to new environment and perform normally as usual. In other words, sustainability makes the software system live longer and healthier. Certainly, the system will also be designed to encounter the problems as peak time request so that it will continue to sustain.

3、strategies and tactics

- sustainability

- 

Quality attribute	Strategy	Tactic	Impact
sustainability	Reduce coupling	Use encapsulation and abstraction	Hide the concrete implementation inside the module and provide the application interfaces. When changing the system to a new environment, we will only have to change some of the implementations, which helps with the sustainability and has little impact on scalability.
		refactor	When the concurrent structure of the system is hard to maintain or the cost of maintenance is not acceptable, software refactoring should be considered to increase the lifetime of the software system and reduce the cost of further maintenance. This method could increase sustainability and has little impact on scalability.
	Manage Resources	Increase Resources	Introduce more resources, e.g. faster processors, addition memory, reliable third-party support. Greater resources would help to meet the problem of changing demand, but also would cost much more money and effort.
		Better resource allocation	Apply better technologies or algorithms to promote highly efficient use of resources. Efficiency would help the system sustain longer without costing extra money for more resources.

- scalability

Quality attribute	Strategy	Tactic	Impact
Scalability	Resource competition	Prioritize Event	With limited resources, complete the most important events first in order to make full use of the system scalability. However, the neglect of some less important events may contribute to some potential risks that may damage the sustainability of the system.
		Bound Execution Times.	Set a fixed time to handle one request. This setting could make all the requests receive the resources equally, which helps the system deal with an increase in demand. But this tactic would lead to catastrophe if one important request were blocked out before receiving enough resources, which decreases sustainability.
	Application distribution	Report back and send.	When the user sends a request, the distributed system may choose one node to report back to the user. If this node fails, the system would automatically arrange another node to handle the previous request without notifying the user. This method could meet a sharp increase of resource demand, and also make the system more sustainable.
		broadcast	If some parts of the distributed system go wrong, they will broadcast their state to the whole system for better resource allocation. This method helps both scalability for increased demand and sustainability for further improvement.