UNIT III

CREATING AN ARCHITECTURE-I

3.1 FUNCTIONALITY AND ARCHITECTURE

Functionality: It is the ability of the system to do the work for which it was intended.

A task requires that many or most of the system's elements work in a coordinated manner to complete the job, just as framers, electricians, plumbers, drywall hangers, painters, and finish carpenters all come together to cooperatively build a house.

Software architecture constrains its allocation to structure when *other* quality attributes are important.

3.2 ARCHITECTURE AND QUALITY ATTRIBUTES

Achieving quality attributes must be considered throughout design, implementation, and deployment. No quality attribute is entirely dependent on design, nor is it entirely dependent on implementation or deployment. For example:

Usability involves both architectural and non-architectural aspects.

Modifiability is determined by how functionality is divided (architectural) and by coding techniques within a module (non-architectural).

Performance involves both architectural and non-architectural dependencies

The message of this section is twofold:

- Architecture is critical to the realization of many qualities of interest in a system, and these qualities should be designed in and can be evaluated at the architectural level
- Architecture, by itself, is unable to achieve qualities. It provides the foundation for achieving quality.

3.3 SYSTEM QUALITY ATTRIBUTES

QUALITY ATTRIBUTE SCENARIOS

A quality attribute scenario is a quality-attribute-specific requirement. It consists of six parts.

Source of stimulus*.*This is some entity (a human, a computer system, or any other actuator) thatgenerated the stimulus.

Stimulus*.*The stimulus is a condition that needs to be considered when it arrives at a system.

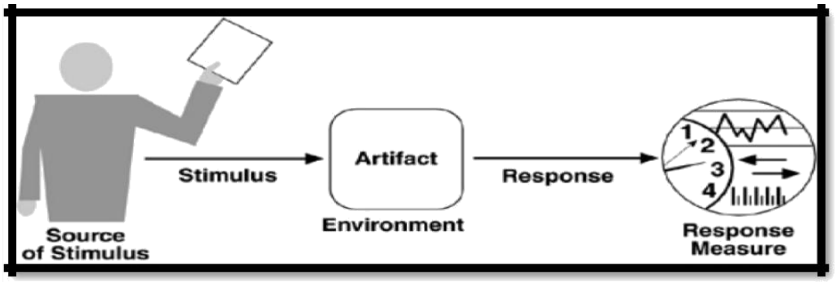
Environment. The stimulus occurs within certain conditions. The system may be in an overloadcondition or may be running when the stimulus occurs, or some other condition may be true.

Artifact*.*Some artifact is stimulated. This may be the whole system or some pieces of it.

Response*.*The response is the activity undertaken after the arrival of the stimulus.

Response measure. When the response occurs, it should be measurable in some fashion so that the requirement can be tested.

Figure 3.1 shows the parts of a quality attribute scenario.



QUALITY ATTRIBUTE SCENARIOS IN PRACTICE

AVAILABILITY SCENARIO

Availability is concerned with system failure and its associated consequencesFailures are usually a result of system errors that are derived from faults in the system. It is typically defines as

$$\alpha = \frac{\text{mean time to failure}}{\text{mean time to failure} + \text{mean time to repair}}$$

Source of stimulus.We differentiate between internal and external indications of faults or failure since thedesired system response may be different. In our example, the unexpected message arrives from outside the system.

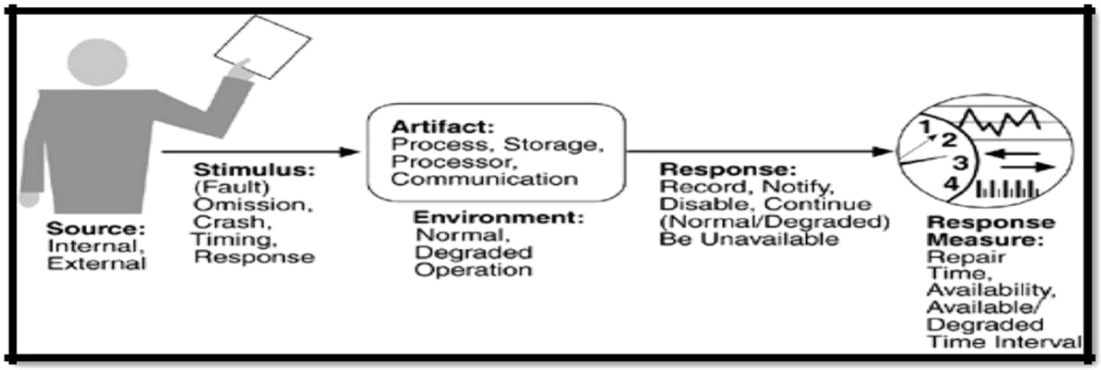
Stimulus. A fault of one of the following classes occurs.

omission. A component fails to respond 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.


Source: Internal, External
Stimulus: (Fault) Omission, Crash, Timing, Response
Artifact: Process, Storage, Processor, Communication
Environment: Normal, Degraded Operation
Response: Record, Notify, Disable, Continue (Normal/Degraded) Be Unavailable
1
2
3
4
Response Measure: Repair Time, Availability, Available/ Degraded Time Interval

Artifact. This specifies the resource that is required to be highly available, such as a processor, communication channel, process, or storage.

Environment.The state of the system when the fault or failure occurs may also affect the desired systemresponse. For example, if the system has already seen some faults and is operating in other than normal mode, it may be desirable to shut it down totally. However, if this is the first fault observed, some degradation of response time or function may be preferred. In our example, the system is operating normally.

Response. There are a number of possible reactions to a system failure. These include logging the failure,notifying selected users or other systems, switching to a degraded mode with either less capacity or less function, shutting down external systems, or becoming unavailable during repair. In our example, the system should notify the operator of the unexpected message and continue to operate normally.

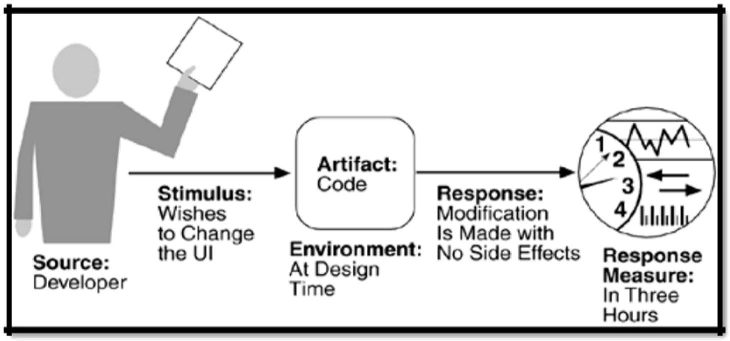
Response measure. The response measure can specify an availability percentage, or it can specify a time torepair, times during which the system must be available, or the duration for which the system must be available.

MODIFIABILITY SCENARIO

Modifiability is about the cost of change. It brings up two concerns.

What can change (the artifact)?

When is the change made and who makes it (the environment)?



Source of stimulus.This portion specifies who makes the changes—the developer, a system administrator, oran end user. Clearly, there must be machinery in place to allow the system administrator or end user to modify a system, but this is a common occurrence. In Figure 4.4, the modification is to be made by the developer.

Stimulus. This portion specifies the changes to be made. A change can be the addition of a function, themodification of an existing function, or the deletion of a function. It can also be made to the qualities of the system—making it more responsive, increasing its availability, and so forth.

The capacity of the system may also change. Increasing the number of simultaneous users is a frequent requirement. In our example, the stimulus is a request to make a modification, which can be to the function, quality, or capacity.

Artifact. This portion specifies what is to be changed—the functionality of a system, its platform, its userinterface, its environment, or another system with which it interoperates. In Figure 4.4, the modification is to the user interface.

Environment. This portion specifies when the change can be made—design time, compile time, build time,initiation time, or runtime. In our example, the modification is to occur at design time.

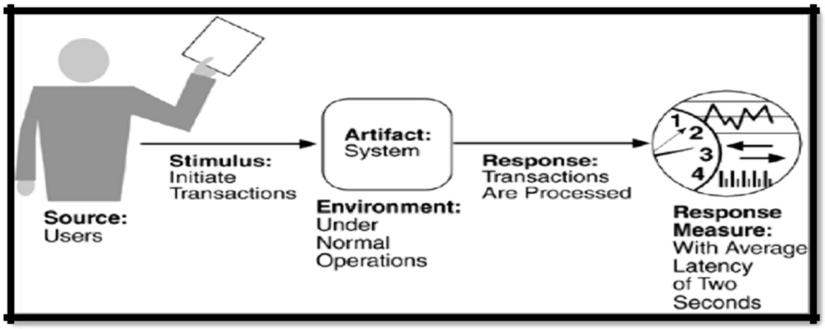
Response. Whoever makes the change must understand how to make it, and then make it, test it and deploy it.

In our example, the modification is made with no side effects.

Response measure.All of the possible responses take time and cost money, and so time and cost are the mostdesirable measures. Time is not always possible to predict, however, and so less ideal measures are frequently used, such as the extent of the change (number of modules affected). In our example, the time to perform the modification should be less than three hours.

PERFORMANCE SCENARIO:

Performance is about timing. Events (interrupts, messages, requests from users, or the passage of time) occur, and the system must respond to them. There are a variety of characterizations of event arrival and the response but basically performance is concerned with how long it takes the system to respond when an event occurs.



Source of stimulus.The stimuli arrive either from external (possibly multiple) or internal sources. In ourexample, the source of the stimulus is a collection of users.

Stimulus. The stimuli are the event arrivals. The arrival pattern can be characterized as periodic, stochastic, orsporadic. In our example, the stimulus is the stochastic initiation of 1,000 transactions per minute.

Artifact. The artifact is always the system's services, as it is in our example.

Environment. The system can be in various operational modes, such as normal, emergency, or overload. In ourexample, the system is in normal mode.

Response. The system must process the arriving events. This may cause a change in the system environment(e.g., from normal to overload mode). In our example, the transactions are processed.

Response measure .The response measures are the time it takes to process the arriving events (latency or adeadline by which the event must be processed), the variation in this time (jitter), the number of events that can be processed within a particular time interval (throughput), or a characterization of the events that cannot be processed (miss rate, data loss). In our example, the transactions should be processed with an average latency of two seconds.

SECURITY SCENARIO

Security is a measure of the system's ability to resist unauthorized usage while still providing its services to legitimate users. An attempt to breach security is called an attack and can take a number of forms. It may be an unauthorized attempt to access data or services or to modify data, or it may be intended to deny services to legitimate users.

Security can be characterized as a system providing non-repudiation, confidentiality, integrity, assurance, availability, and auditing. For each term, we provide a definition and an example.

Non-repudiation is the property that a transaction (access to or modification of data or services)cannot be denied by any of the parties to it. This means you cannot deny that you ordered that item over the Internet if, in fact, you did.

Confidentiality is the property that data or services are protected from unauthorized access. Thismeans that a hacker cannot access your income tax returns on a government computer.

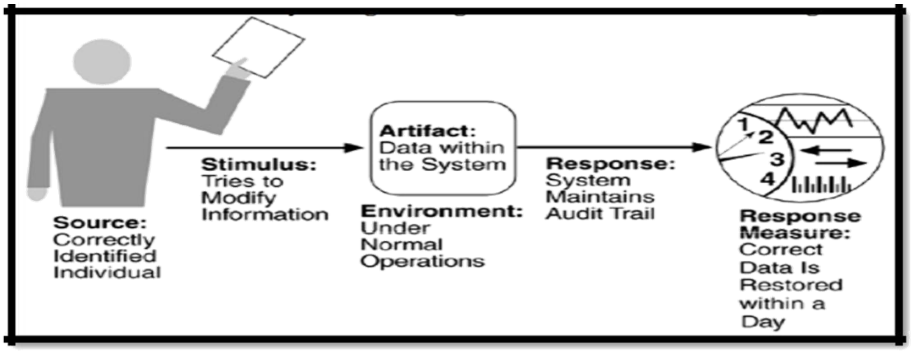
Integrity is the property that data or services are being delivered as intended. This means that yourgrade has not been changed since your instructor assigned it.

Assurance is the property that the parties to a transaction are who they purport to be. This means that,when a customer sends a credit card number to an Internet merchant, the merchant is who the customer thinks they are.

Availability is the property that the system will be available for legitimate use. This means that adenial-of-service attack won't prevent your ordering *this* book.

Auditing is the property that the system tracks activities within it at levels sufficient to reconstructthem. This means that, if you transfer money out of one account to another account, in Switzerland, the system will maintain a record of that transfer.

Each of these security categories gives rise to a collection of general scenarios.


Source: Correctly Identified Individual
Stimulus: Tries to Modify Information
Artifact: Data within the System
Environment: Under Normal Operations
Response: System Maintains Audit Trail
Response Measure: Correct Data Is Restored within a Day
1
2
3
4

Source of stimulus.The source of the attack may be either a human or another system. It may have beenpreviously identified (either correctly or incorrectly) or may be currently unknown.

Stimulus. The stimulus is an attack or an attempt to break security. We characterize this as an unauthorizedperson or system trying to display information, change and/or delete information, access services of the system, or reduce availability of system services. In Figure, the stimulus is an attempt to modify data.

Artifact.The target of the attack can be either the services of the system or the data within it. In our example,the target is data within the system.

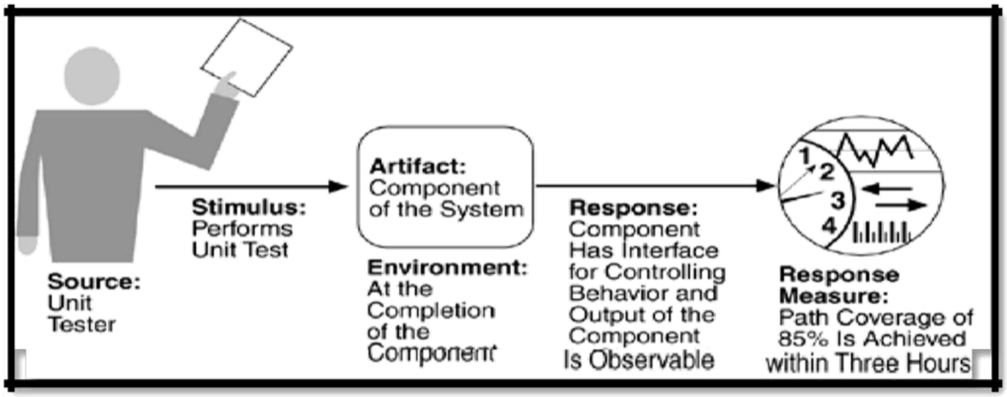
Environment. The attack can come when the system is either online or offline, either connected to ordisconnected from a network, either behind a firewall or open to the network.

Response.Using services without authorization or preventing legitimate users from using services is a differentgoal from seeing sensitive data or modifying it. Thus, the system must authorize legitimate users and grant them access to data and services, at the same time rejecting unauthorized users, denying them access, and reporting unauthorized access

Response measure.Measures of a system's response include the difficulty of mounting various attacks and thedifficulty of recovering from and surviving attacks. In our example, the audit trail allows the accounts from which money was embezzled to be restored to their original state.

TESTABILITY SCENARIO:

Software testability refers to the ease with which software can be made to demonstrate its faults through testing. In particular, testability refers to the probability, assuming that the software has at least one fault that it will fail on its *next* test execution. Testing is done by various developers, testers, verifiers, or users and is the last step of various parts of the software life cycle. Portions of the code, the design, or the complete system may be tested.


Source:
Unit Tester
Stimulus:
Performs Unit Test
Artifact:
Component of the System
Environment:
At the Completion of the Component
Response:
Component Has Interface for Controlling Behavior and Output of the Component Is Observable
1
2
3
4
Response Measure:
Path Coverage of 85% Is Achieved within Three Hours

Source of stimulus.The testing is performed by unit testers, integration testers, system testers, or the client. Atest of the design may be performed by other developers or by an external group. In our example, the testing is performed by a tester.

Stimulus. The stimulus for the testing is that a milestone in the development process is met. This might be thecompletion of an analysis or design increment, the completion of a coding increment such as a class, the completed integration of a subsystem, or the completion of the whole system. In our example, the testing is triggered by the completion of a unit of code.

Artifact.A design, a piece of code, or the whole system is the artifact being tested. In our example, a unit of codeis to be tested.

Environment. The test can happen at design time, at development time, at compile time, or at deployment time.

In Figure, the test occurs during development.

Response.Since testability is related to observability and controllability, the desired response is that thesystem can be controlled to perform the desired tests and that the response to each test can be observed. In our example, the unit can be controlled and its responses captured.

Response measure.Response measures are the percentage of statements that have been executed in some test,the length of the longest test chain (a measure of the difficulty of performing the tests), and estimates of the probability of finding additional faults. In Figure, the measurement is percentage coverage of executable statements.

USABILITY SCENARIO

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. It can be broken down into the following areas:

Learning system features.If the user is unfamiliar with a particular system or a particular aspect of it, what canthe system do to make the task of learning easier?

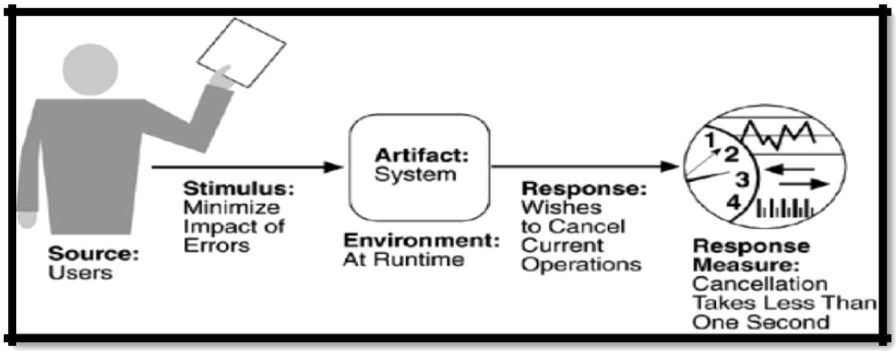
Using a system efficiently.What can the system do to make the user more efficient in its operation?

Minimizing the impact of errors. What can the system do so that a user error has minimal impact?

Adapting the system to user needs.How can the user (or the system itself) adapt to make the user's task easier?

Increasing confidence and satisfaction.What does the system do to give the user confidence that the correctaction is being taken?

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. The portions of the usability general scenarios are:



Source of stimulus.The end user is always the source of the stimulus.

Stimulus.The stimulus is that the end user wishes to use a system efficiently, learn to use the system, minimizethe impact of errors, adapt the system, or feel comfortable with the system. In our example, the user wishes to cancel an operation, which is an example of minimizing the impact of errors.

Artifact.The artifact is always the system.

Environment.The user actions with which usability is concerned always occur at runtime or at systemconfiguration time. In Figure, the cancellation occurs at runtime.

Response.The system should either provide the user with the features needed or anticipate the user's needs. Inour example, the cancellation occurs as the user wishes and the system is restored to its prior state.

Response measure.The response is measured by task time, number of errors, number of problems solved,user satisfaction, gain of user knowledge, ratio of successful operations to total operations, or amount of time/data lost when an error occurs. In Figure, the cancellation should occur in less than one second.

COMMUNICATING CONCEPTS USING GENERAL SCENARIOS

One of the uses of general scenarios is to enable stakeholders to communicate. Table 4.7 gives the stimuli possible for each of the attributes and shows a number of different concepts.

Some stimuli occur during runtime and others occur before. The problem for the architect is to understand which of these stimuli represent the same occurrence, which are aggregates of other stimuli, and which are independent.

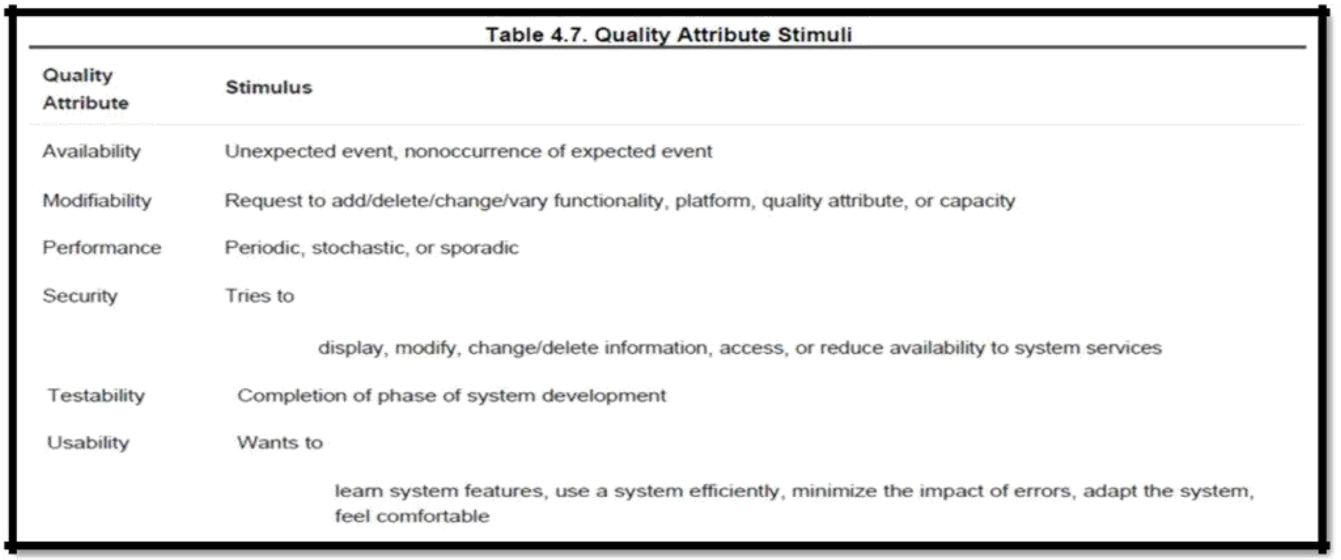


Table 4.7. Quality Attribute Stimuli

Quality Attribute	Stimulus
Availability	Unexpected event, nonoccurrence of expected event
Modifiability	Request to add/delete/change/vary functionality, platform, quality attribute, or capacity
Performance	Periodic, stochastic, or sporadic
Security	Tries to display, modify, change/delete information, access, or reduce availability to system services
Testability	Completion of phase of system development
Usability	Wants to learn system features, use a system efficiently, minimize the impact of errors, adapt the system, feel comfortable

3.5 OTHER SYSTEM QUALITY ATTRIBUTES

SCALABILITY

PORTABILITY

3.6 BUSINESS QUALITIES

Time to market.

If there is competitive pressure or a short window of opportunity for a system or product, development time becomes important. This in turn leads to pressure to buy or otherwise re-use existing elements.

Cost and benefit.

The development effort will naturally have a budget that must not be exceeded. Different architectures will yield different development costs. For instance, an architecture that relies on technology (or expertise with a technology) not resident in the developing organization will be more expensive to realize than one that takes advantage of assets already inhouse. An architecture that is highly flexible will typically be more costly to build than one that is rigid (although it will be less costly to maintain and modify).

Projected lifetime of the system

If the system is intended to have a long lifetime, modifiability, scalability, and portability become important. On the other hand, a modifiable, extensible product is more likely to survive longer in the marketplace, extending its lifetime.

Targeted market.

For general-purpose (mass-market) software, the platforms on which a system runs as well as its feature set will determine the size of the potential market. Thus, portability and functionality are key to market share. Other qualities, such as performance, reliability, and usability also play a role.

Rollout schedule.

If a product is to be introduced as base functionality with many features released later, the flexibility and customizability of the architecture are important. Particularly, the system must be constructed with ease of expansion and contraction in mind.

Integration with legacy systems.

If the new system has to *integrate* with existing systems, care must be taken to define appropriate integration mechanisms. This property is clearly of marketing importance but has substantial architectural implications.

3.7 ARCHITECTURE QUALITIES

Conceptual integrity is the underlying theme or vision that unifies the design of the system at all levels.The architecture should do similar things in similar ways.

Correctness* and *completeness are essential for the architecture to allow for all of the system'srequirements and runtime resource constraints to be met.

Buildability allows the system to be completed by the available team in a timely manner and to be opento certain changes as development progresses.

3.8 INTRODUCING TACTICS

A **tactic** is a design decision that influences the control of a quality attribute response.

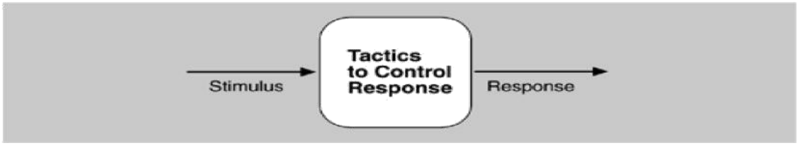
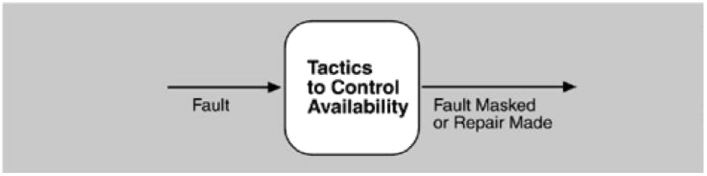


Figure 3.8 Tactics are intended to control responses to stimuli.

- *Tactics can refine other tactics*. For each quality attribute that we discuss, we organize the tactics as a hierarchy.
- *Patterns package tactics.* A pattern that supports availability will likely use both a redundancy tactic and a synchronization tactic.

3.9 AVAILABILITY TACTICS



The above figure depicts goal of availability tactics. All approaches to maintaining availability involve some type of redundancy, some type of health monitoring to detect a failure, and some type of recovery when a failure is detected. In some cases, the monitoring or recovery is automatic and in others it is manual.

FAULT DETECTION

- ***Ping/echo***.One component issues a ping and expects to receive back an echo, within a predefined time,from the component under scrutiny. This can be used within a group of components mutually responsible for one task
- ***Heartbeat (dead man timer).*** In this case one component emits a heartbeat message periodically andnother component listens for it. If the heartbeat fails, the originating component is assumed to have failed and a fault correction component is notified. The heartbeat can also carry data.
- ***Exceptions***.The exception handler typically executes in the same process that introduced the exception.

FAULT RECOVERY

- ***Voting.*** Processes running on redundant processors each take equivalent input and compute a simpleoutput value that is sent to a voter. If the voter detects deviant behavior from a single processor, it fails it.
- ***Active redundancy (hot restart).*** All redundant components respond to events in parallel. Theresponse from only one component is used (usually the first to respond), and the rest are discarded. Active redundancy is often used in a client/server configuration, such as database management systems, where quick responses are necessary even when a fault occurs
- ***Passive redundancy (warm restart/dual redundancy/triple redundancy)***.One component (theprimary) responds to events and informs the other components (the standbys) of state updates they must make. When a fault occurs, the system must first ensure that the backup state is sufficiently fresh before resuming services.

- ***Spare*** A standby spare computing platform is configured to replace many different failed components. It must be rebooted to the appropriate software configuration and have its state initialized when a failure occurs.
- ***Shadow operation**.* A previously failed component may be run in "shadow mode" for a short time to make sure that it mimics the behavior of the working components before restoring it to service.
- ***State resynchronization**.*The passive and active redundancy tactics require the component being restored to have its state upgraded before its return to service.
- ***Checkpoint/rollback**.*A checkpoint is a recording of a consistent state created either periodically or inresponse to specific events. Sometimes a system fails in an unusual manner, with a detectably inconsistent state. In this case, the system should be restored using a previous checkpoint of a consistent state and a log of the transactions that occurred since the snapshot was taken.

FAULT PREVENTION

- **R**_**emoval from service.**_This tactic removes a component of the system from operation to undergo someactivities to prevent anticipated failures.
- ***Transactions**.*A transaction is the bundling of several sequential steps such that the entire bundle Can be undone at once. Transactions are used to prevent any data from being affected if one step in a process fails and also to prevent collisions among several simultaneous threads accessing the same data.

***Process monitor**.* Once a fault in a process has been detected, a monitoring process can delete the non performing process and create a new instance of it, initialized to some appropriate state as in the spare tactic.

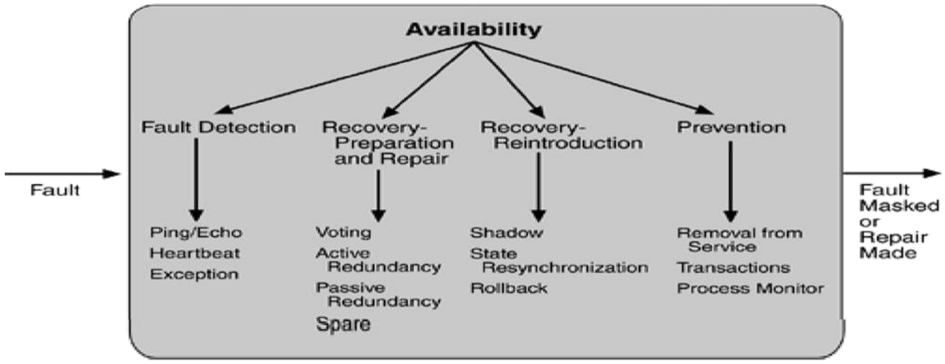
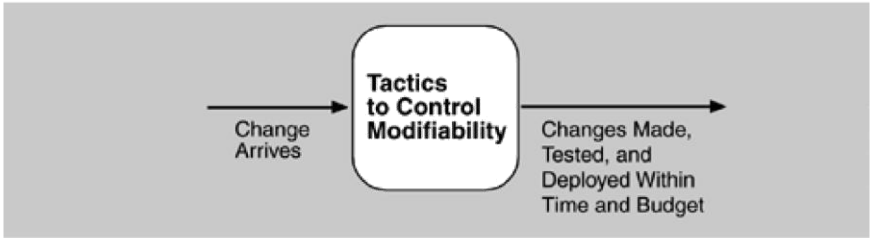


Figure 3.9. Summary of availability tactics

3.10 MODIFIABILITY TACTICS

The figure below represent goal of modifiability tactics.


Change Arrives
Tactics to Control Modifiability
Changes Made, Tested, and Deployed Within Time and Budget

LOCALIZE MODIFICATIONS

- ***Maintain semantic coherence***. Semantic coherence refers to the relationships among Responsibilities in a module. The goal is to ensure that all of these responsibilities work together without excessive reliance on other modules.
- ***Anticipate expected changes***. Considering the set of envisioned changes provides a way to evaluate a particular assignment of responsibilities. In reality this tactic is difficult to use by itself since it is not possible to anticipate all changes.
- ***Generalize the module***. Making a module more general allows it to compute a broader range Of functions based on input
- ***Limit possible options***. Modifications, especially within a product line, may be far ranging and Hence affect many modules. Restricting the possible options will reduce the effect of these modifications

PREVENT RIPPLE EFFECTS

We begin our discussion of the ripple effect by discussing the various types of dependencies that one module can have on another. We identify eight types:

- Syntax of *data service.*
- Semantics of *data service.*
- Sequence of *data control.*
- *Identity of an interface of A*
- *Location of A (runtime).*
- *Quality of service/data provided by A.*
- *Existence of A*
- *Resource behaviour of A.*

With this understanding of dependency types, we can now discuss tactics available to the architect for preventing the ripple effect for certain types.

- ***Hide information***. Information hiding is the decomposition of the responsibilities for an entity Into smaller pieces and choosing which information to make private and which to make public. The goal is to isolate changes within one module and prevent changes from propagating to others
- ***Maintain existing interfaces*** it is difficult to mask dependencies on quality of data or quality of service,resource usage, or resource ownership. Interface stability can also be achieved by separating the interface from the implementation. This allows the creation of abstract interfaces that mask variations.
- ***Restrict communication paths***. This will reduce the ripple effect since data production/consumptionintroduces dependencies that cause ripples.
- ***Use an intermediary*** If B has any type of dependency on A other than semantic, it is possible to insert an intermediary between B and A that manages activities associated with the dependency.

DEFER BINDING TIME

Many tactics are intended to have impact at loadtime or runtime, such as the following.

- ***Runtime registration*** supports plug-and-play operation at the cost of additional overhead *to* Manage the registration.
- ***Configuration files*** are intended to set parameters at startup.
- ***Polymorphism*** allows late binding of method calls.
- ***Component replacement*** allows load time binding.
- ***Adherence to defined protocols*** allows runtime binding of independent processes.

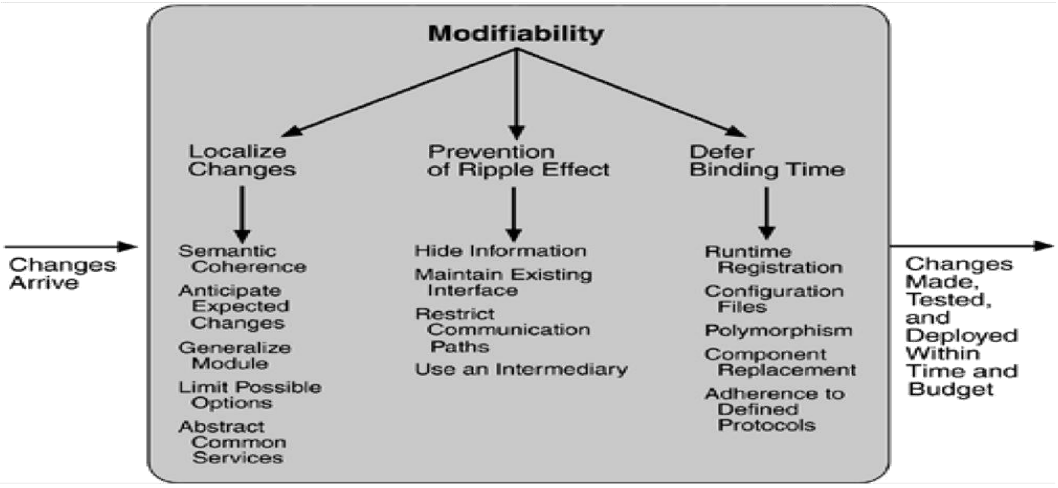
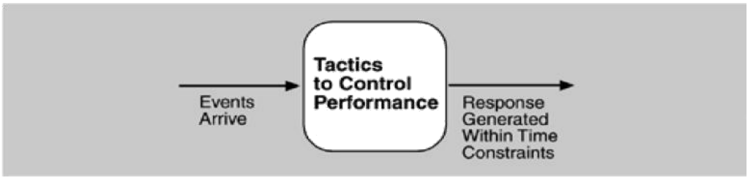


Figure 3.10. Summary of modifiability tactics

3.11 PERFORMANCE TACTICS

Performance tactics control the time within which a response is generated. Goal of performance tactics is shown below:



After an event arrives, either the system is processing on that event or the processing is blocked for some reason. This leads to the two basic contributors to the response time: resource consumption and blocked time.

Resource consumption. For example, a message is generated by one component, is placed on the network, and arrives at another component. Each of these phases contributes to the overall latency of the processing of that event.

Blocked time. A computation can be blocked from using a resource because of contention for it, because the resource is unavailable, or because the computation depends on the result of other computations that are not yet available.

Contention for resources

Availability of resources

Dependency on other computation.

RESOURCE DEMAND

One tactic for reducing latency is to reduce the resources required for processing an event stream.

- ***Increase computational efficiency***. One step in the processing of an event or a message is applying some algorithm. Improving the algorithms used in critical areas will decrease latency. This tactic is usually applied to the processor but is also effective when applied to other resources such as a disk.
- ***Reduce computational overhead.*** If there is no request for a resource, processing needs are reduced. The use of intermediaries increases the resources consumed in processing an event stream, and so removing them improves latency.

Another tactic for reducing latency is to reduce the number of events processed. This can be done in one of two fashions.

- ***Manage event rate.*** If it is possible to reduce the sampling frequency at which environmental Variables are monitored, demand can be reduced.

- ***Control frequency of sampling.*** If there is no control over the arrival of externally generated events, queued requests can be sampled at a lower frequency, possibly resulting in the loss of requests.

Other tactics for reducing or managing demand involve controlling the use of resources.

- ***Bound execution times***. Place a limit on how much execution time is used to respond to an event. Sometimes this makes sense and sometimes it does not.
- ***Bound queue sizes.*** This controls the maximum number of queued arrivals and consequently there sources used to process the arrivals.

RESOURCE MANAGEMENT

- ***Introduce concurrency***. If requests can be processed in parallel, the blocked time can be reduced
- ***Maintain multiple copies of either data or computations***. Clients in a client-server pattern are replicas of the computation. The purpose of replicas is to reduce the contention that would occur if all computations took place on a central server.

Increase available resources.Faster processors, additional processors, additional memory, and fasternetworks all have the potential for reducing latency.

RESOURCE ARBITRATION

- ***First-in/First-out***. FIFO queues treat all requests for resources as equals and satisfy them in turn.

Fixed-priority scheduling. Fixed-priority scheduling assigns each source of resource requests A particular priority and assigns the resources in that priority order.

Three common prioritization strategies are

- *semantic importance*. Each stream is assigned a priority statically according to some domaincharacteristic of the task that generates it.
- *deadline monotonic*. Deadline monotonic is a static priority assignment that assigns higher priority to streams with shorter deadlines.
- *rate monotonic.*Rate monotonic is a static priority assignment for periodic streams that assigns higher priority to streams with shorter periods.
- *round robin*. Round robin is a scheduling strategy that orders the requests and then, at everyassignment possibility, assigns the resource to the next request in that order.
- *earliest deadline first*. Earliest deadline first assigns priorities based on the pending requests withthe earliest deadline.

- ***Static scheduling***. A cyclic executive schedule is a scheduling strategy where the pre-emption pointsand the sequence of assignment to the resource are determined offline.

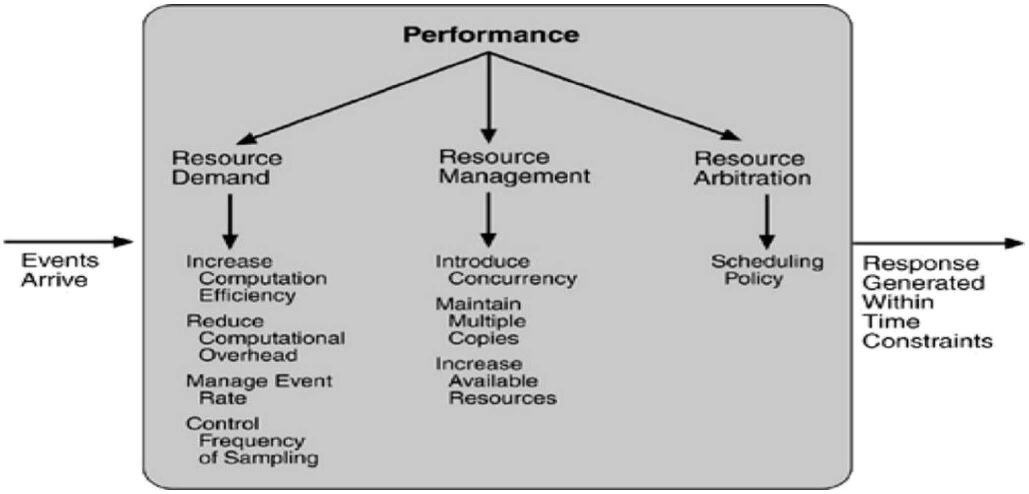
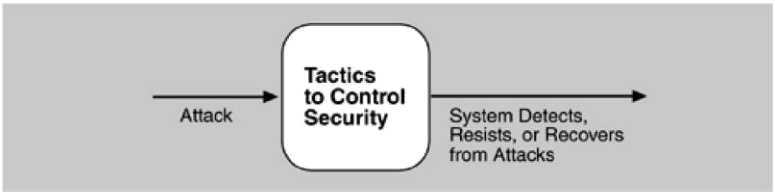

Performance
Resource Demand
Resource Management
Resource Arbitration
Events Arrive
Increase Computation Efficiency
Reduce Computational Overhead
Manage Event Rate
Control Frequency of Sampling
Introduce Concurrency
Maintain Multiple Copies
Increase Available Resources
Scheduling Policy
Response Generated Within Time Constraints

Figure 3.11. Summary of performance tactics

3.12 SECURITY TACTICS

Tactics for achieving security can be divided into those concerned with resisting attacks, those concerned with detecting attacks, and those concerned with recovering from attacks.


Attack
Tactics to Control Security
System Detects, Resists, or Recovers from Attacks

RESISTING ATTACKS

- ***Authenticate users.*** Authentication is ensuring that a user or remote computer is actually who It purports to be.

  Passwords, one-time passwords, digital certificates, and biometric identifications provide authentication.
- ***Authorize users.*** Authorization is ensuring that an authenticated user has the rights to access andmodify either data or services. Access control can be by user or by user class.
- ***Maintain data confidentiality***.Data should be protected from unauthorized access. Confidentiality isusually achieved by applying some form of encryption to data and to communication links. Encryption provides extra protection to persistently maintained data beyond that available from authorization.
- ***Maintain integrity.*** Data should be delivered as intended. It can have redundant information encodedin it, such as checksums or hash results, which can be encrypted either along with or independently from the original data.

Limit exposure. Attacks typically depend on exploiting a single weakness to attack all data and serviceson a host. The architect can design the allocation of services to hosts so that limited services are available on each host.

- ***Limit access.*** Firewalls restrict access based on message source or destination port. Messages fromunknown sources may be a form of an attack. It is not always possible to limit access to known sources.

DETECTING ATTACKS

- The detection of an attack is usually through an *intrusion detection* system.
- Such systems work by comparing network traffic patterns to a database.
- In the case of misuse detection, the traffic pattern is compared to historic patterns of known attacks.
- In the case of anomaly detection, the traffic pattern is compared to a historical baseline of itself. Frequently, the packets must be filtered in order to make comparisons.
- Filtering can be on the basis of protocol, TCP flags, payload sizes, source or destination address, or port number.
- Intrusion detectors must have some sort of sensor to detect attacks, managers to do sensor fusion, databases for storing events for later analysis, tools for offline reporting and analysis, and a control console so that the analyst can modify intrusion detection actions.

RECOVERING FROM ATTACKS

- Tactics involved in recovering from an attack can be divided into those concerned with restoring state and those concerned with attacker identification (for either preventive or punitive purposes).
- The tactics used in restoring the system or data to a correct state overlap with those used for availability since they are both concerned with recovering a consistent state from an inconsistent state. One difference is that special attention is paid to maintaining redundant copies of system administrative data such as passwords, access control lists, domain name services, and user profile data.
- The tactic for identifying an attacker is to *maintain an audit trail*
- **An** audit trail is a copy of each transaction applied to the data in the system together with identifying information.

- Audit information can be used to trace the actions of an attacker, support nonrepudiation (it provides evidence that a particular request was made), and support system recovery.
- Audit trails are often attack targets themselves and therefore should be maintained in a trusted fashion.

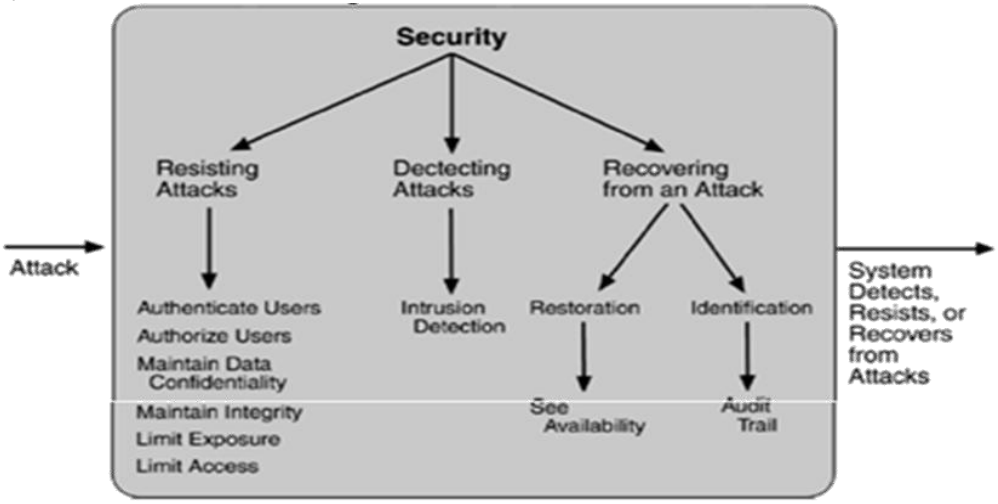
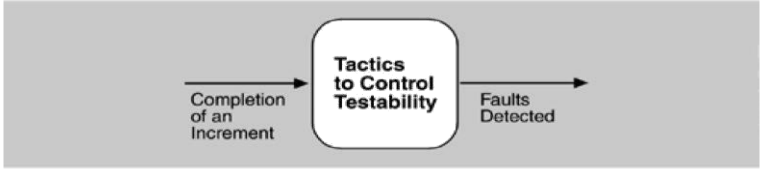


Figure 5.9. Summary of tactics for security

3.13 TESTABILITY TACTICS

The goal of tactics for testability is to allow for easier testing when an increment of software development is completed. Figure 5.10 displays the use of tactics for testability.



INPUT/OUTPUT

- ***Record/playback***.Record/playback refers to both capturing information crossing an interface andusing it as input into the test harness. The information crossing an interface during normal operation is saved in some repository. Recording this information allows test input for one of the components to be generated and test output for later comparison to be saved.
- ***Separate interface from implementation.*** Separating the interface from the implementation allowssubstitution of implementations for various testing purposes. Stubbing implementations allows the remainder of the system to be tested in the absence of the component being stubbed.

- ***Specialize access routes/interfaces.*** Having specialized testing interfaces allows the capturing orspecification of variable values for a component through a test harness as well as independently from its normal execution. Specialized access routes and interfaces should be kept separate from the access routes and interfaces for required functionality.

INTERNAL MONITORING

- ***Built-in monitors****.*The component can maintain state, performance load, capacity, security, or otherinformation accessible through an interface. This interface can be a permanent interface of the component or it can be introduced temporarily. A common technique is to record events when monitoring states have been activated. Monitoring states can actually increase the testing effort since tests may have to be repeated with the monitoring turned off. Increased visibility into the activities of the component usually more than outweigh the cost of the additional testing.

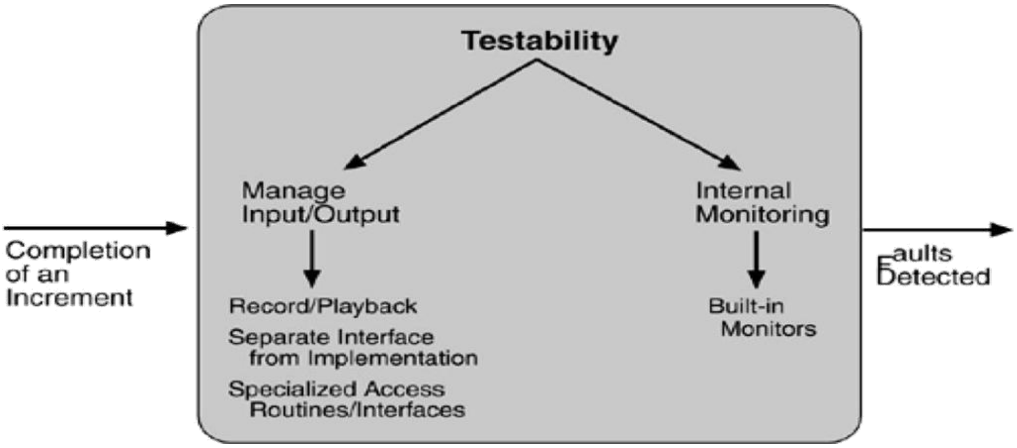
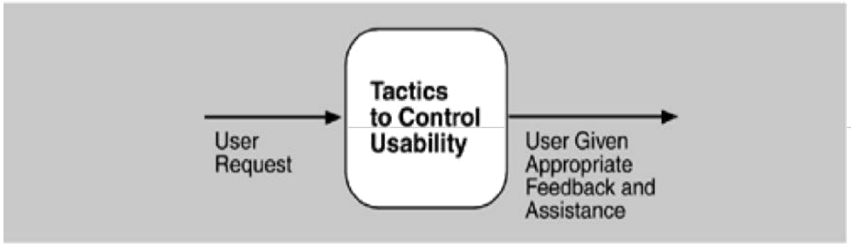

Testability
Manage Input/Output
Internal Monitoring
Completion of an Increment
Faults Detected
Record/Playback
Separate Interface from Implementation
Specialized Access Routines/Interfaces
Built-in Monitors

Figure 3.13. Summary of testability tactics

3.14 USABILITY TACTICS


User Request
Tactics to Control Usability
User Given Appropriate Feedback and Assistance

RUNTIME TACTICS

- ***Maintain a model of the task.*** In this case, the model maintained is that of the task. The task model isused to determine context so the system can have some idea of what the user is attempting and provide various kinds of assistance. For example, knowing that sentences usually start with capital letters would allow an application to correct a lower-case letter in that position.

  Maintain a model of the user.In this case, the model maintained is of the user. It determines the user'sknowledge of the system, the user's behavior in terms of expected response time, and other aspects specific to a user or a class of users. For example, maintaining a user model allows the system to pace scrolling so that pages do not fly past faster than they can be read.

  Maintain a model of the system.In this case, the model maintained is that of the system. It determinesthe expected system behavior so that appropriate feedback can be given to the user. The system model predicts items such as the time needed to complete current activity.

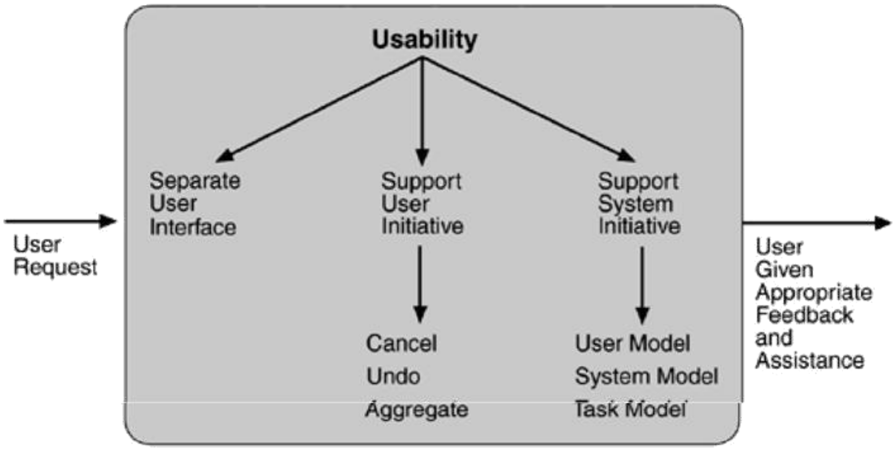


Figure 3.14. Summary of runtime usability tactics

DESIGN-TIME TACTICS

- ***Separate the user interface from the rest of the application***. Localizing expected changes is therationale for semantic coherence. Since the user interface is expected to change frequently both during the development and after deployment, maintaining the user interface code separately will localize changes to it. The software architecture patterns developed to implement this tactic and to support the modification of the user interface are:
  - Model-View-Controller
  - Presentation-Abstraction-Control
  - Arch/Slink