



# SE 4352

## Software Architecture and Design

Fall 2018

Module 11

# Barriers to Achieving Quality

- Common reasons for failure to meet high levels of quality include:
  - Misunderstanding of the importance of quality attributes
  - Inadequate languages for expressing and specifying quality requirements
  - Inadequate modeling methods and notations for expressing solutions that address specific quality attributes
  - Difficulty in designing for quality attributes
  - Lack of documented design and architecture patterns for addressing quality attributes
  - Quality control as an afterthought in most projects

# What are quality attributes?





# Functionality

- It is the ability of the system or application to satisfy the purpose for which it was designed.
- It is related to validity, correctness, interoperability, and security.
- It drives the initial decomposition of the system.
- It is the basis upon which all other quality attributes are specified



# Interoperability

- It is the quality of a system that enables it to work with other systems.
- It includes the quality to work with other systems not yet known.



# Security

- It is the ability to enforce authorization, authentication, and deliberate denial of service attacks.
- Security requirements can affect the functional decomposition of the system.



# Performance (Efficiency)

- It represents the responsiveness of the system (e.g., the time required to respond to events or the number of events processed in some period of time).
- Some aspects of performance are architectural (the distribution of computations across components) and some are not (choices of algorithms).
- Performance has been a driving factor in systems architecture and often compromises the achievement of other quality attributes.



# Resource Efficiency

- One aspect of performance is the efficient utilization of resources.
- The decomposition of a system into layers may impede the system's ability to satisfy performance requirements.





# Modifiability

- It is the ability to grow an architecture over time.
- A modifiable architecture can have new features added without requiring architectural rework



# Availability and Reliability

- Availability is an attribute that measures the proportion of time the system is up and running.
- Reliability is an attribute that measures the system's ability to continue operating over time.
- Both are partially addressed by the architecture.



# Recoverability

- It is the ability of a system to pick up where it left off after a shutdown or crash.
- This might involve launching diagnostic programs to check the integrity of the system before the actual system launches.



# Usability

- This typically refers to the usability with respect to the end user.
- It can also address other system users such as system maintainers, operators, etc.
- It is measured using scenarios.

# Portability

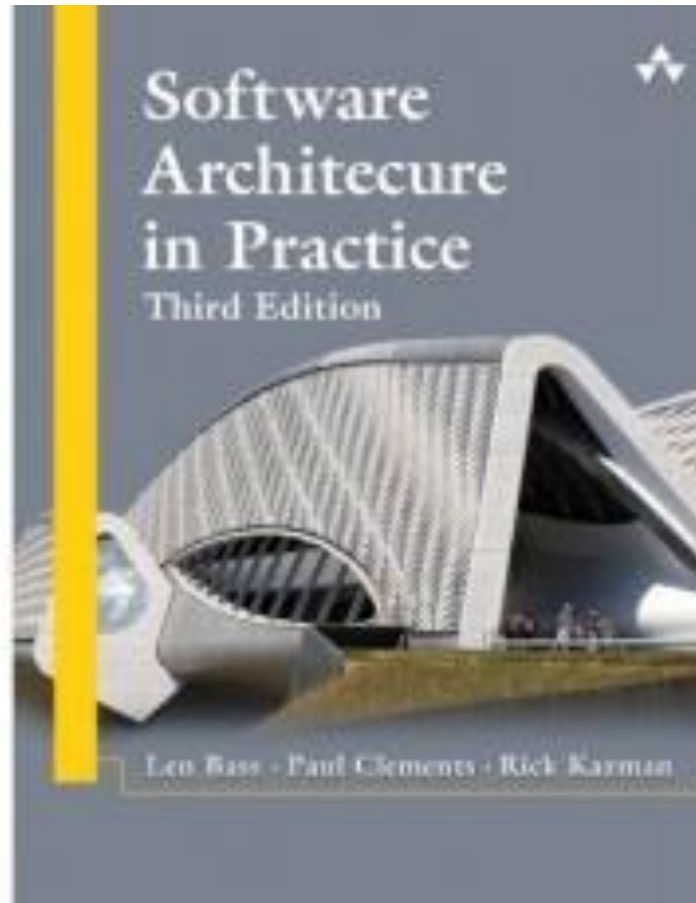
- It is the ability to reuse a component in a different application or operating environment.
- It can be considered as a special kind of modifiability.
- Portability related attributes are:
  - Adaptability
  - Installability
  - Conformance
  - Replaceability



# Portability (Cont'd)

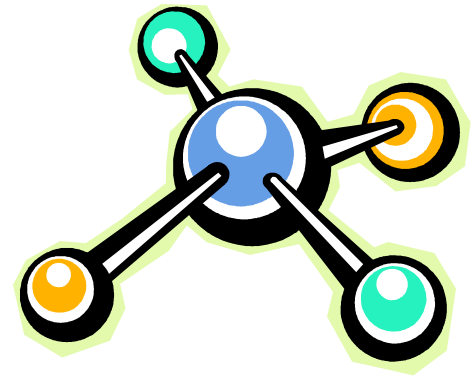
- Extensibility is another word for portability.
- One way to satisfy the extensibility requirement is to produce a set of platform-independent architectural models. This forces the designer to consider the architecture from a platform-independent point of view.

# Chapter 14



# Modeling Architectures to Enable Quality Attribute Analysis

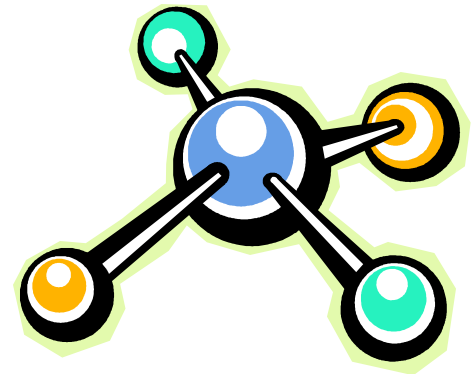
- Ch 2 Importance of SA: Analysis of the architecture enables early prediction of a system's qualities



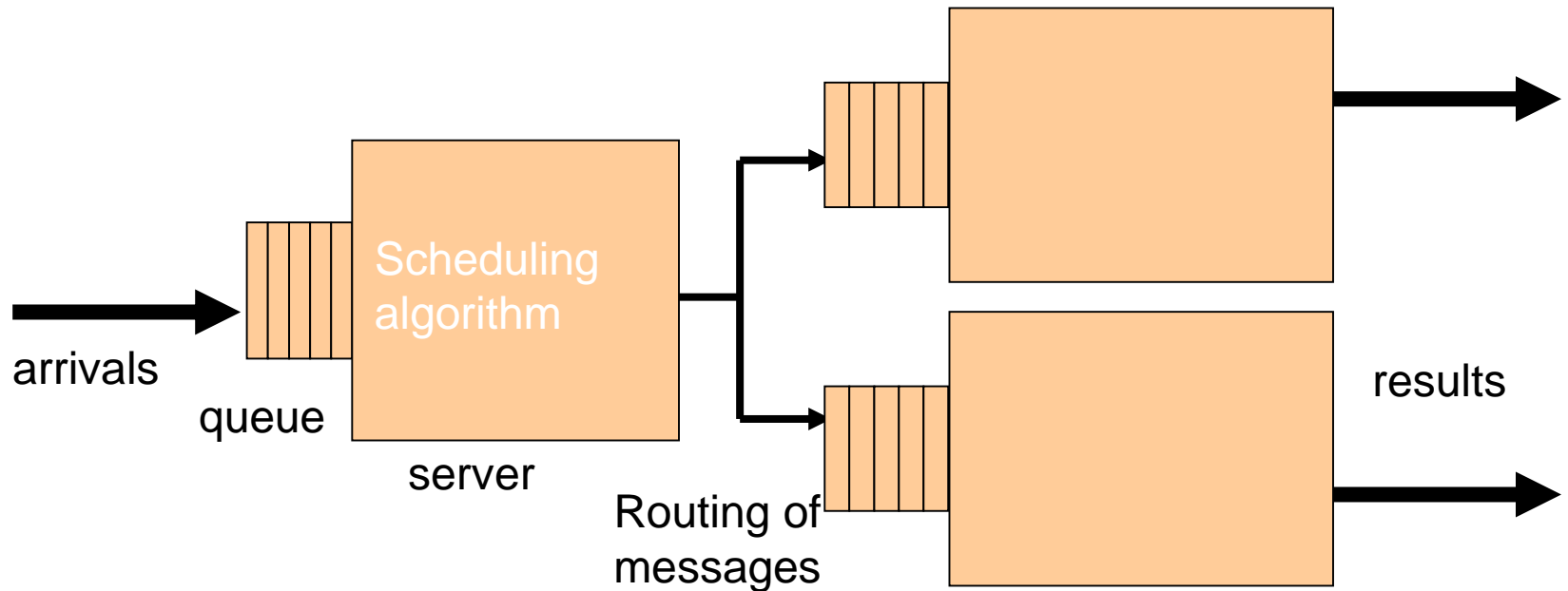


# Modeling Architectures to Enable Quality Attribute Analysis

- Some quality attributes, most notably performance and availability, have well-understood, time-tested analytic models that can be used to assist in an analysis.
- By *analytic model*, we mean one that supports quantitative analysis.

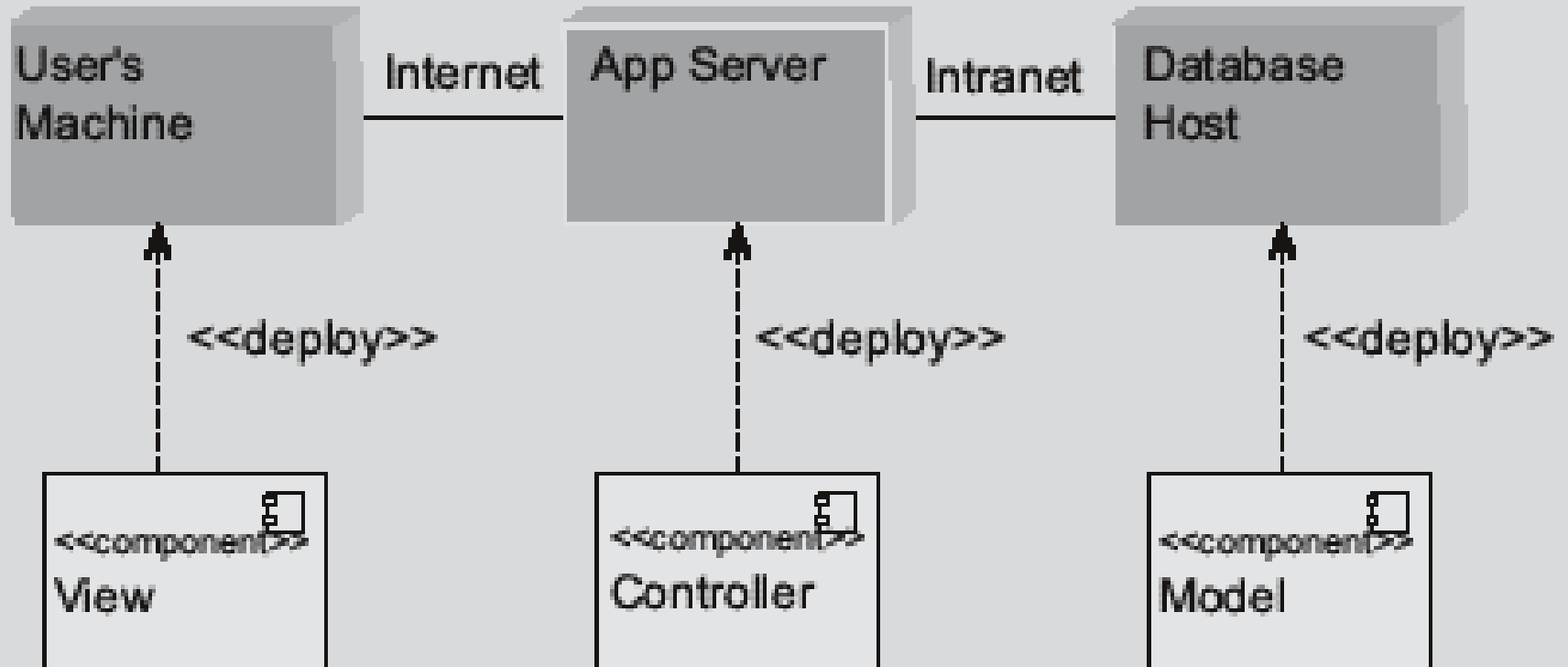


# Performance Models

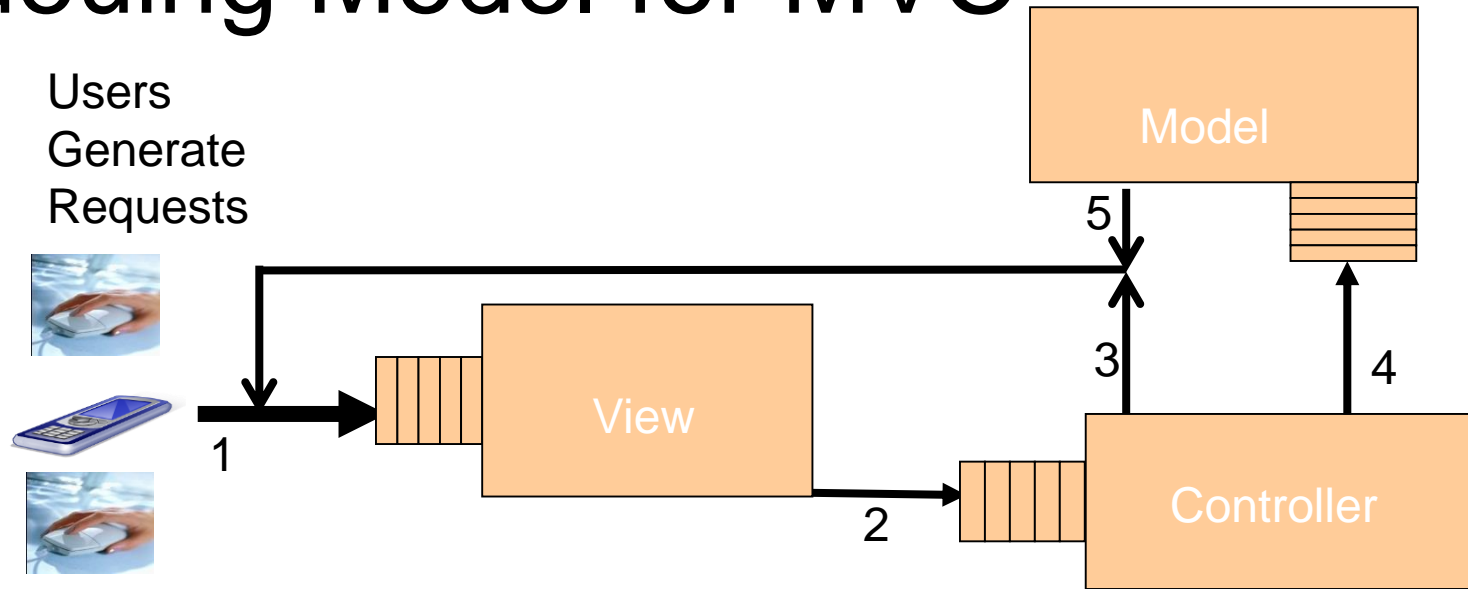


- Parameters: arrival rate of events, chosen queuing discipline, chosen scheduling algorithm, service time for events, network topology, network bandwidth, routing algorithm chosen

# Allocation Model for MVC



# Queuing Model for MVC



1. Arrivals
2. View sends requests to Controller
3. Actions returned to View
4. Actions returned to model
5. Model sends actions to View

# Parameters

- To solve a queuing model for MVC performance, the following parameters must be known or estimated:
  - The frequency of arrivals from outside the system
  - The queuing discipline used at the view queue
  - The time to process a message within the view
  - The number and size of messages that the view sends to the controller
  - The bandwidth of the network that connects the view and the controller
  - The queuing discipline used by the controller
  - The time to process a message within the controller
  - The number and size of messages that the controller sends back to the view
  - The bandwidth of the network from the controller to the view
  - The number and size of messages that the controller sends to the model
  - The queuing discipline used by the model
  - The time to process a message within the model
  - The number and size of messages the model sends to the view
  - The bandwidth of the network connecting the model and the view



# Cost/benefit of Performance Modeling

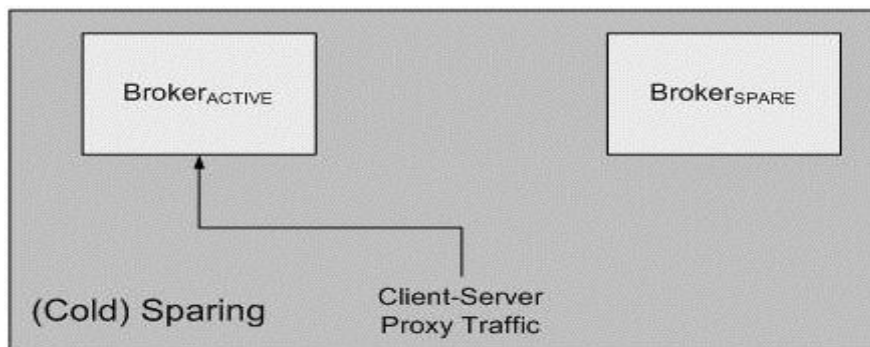
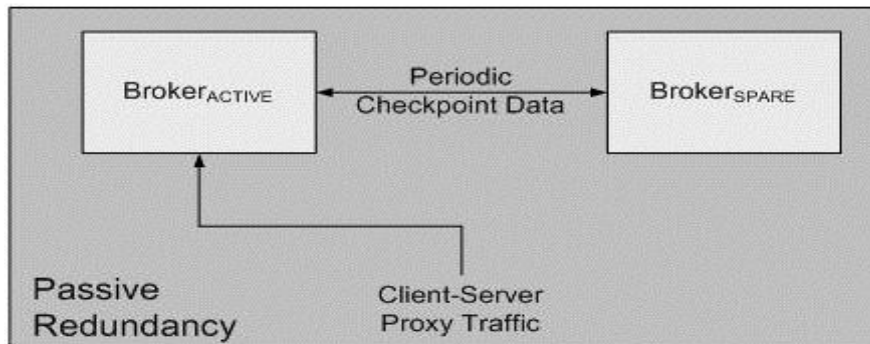
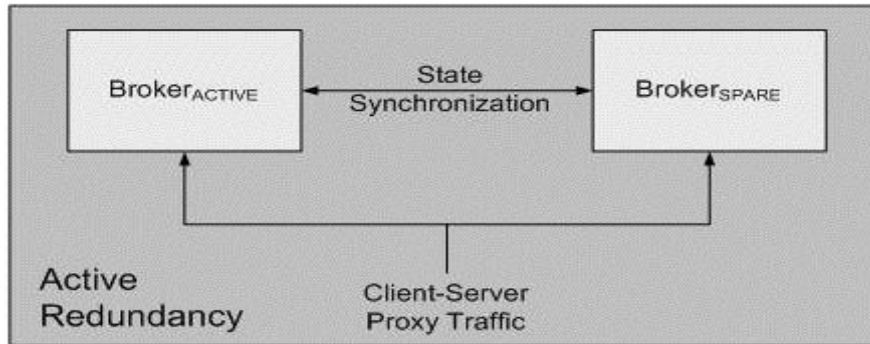
- Cost: determining the parameters previously mentioned
- Benefit: estimate of the latency
- The more accurately the parameters can be estimated, the better the prediction of latency.
- This is worthwhile when latency is important and questionable.
- This is not worthwhile when it is obvious there is sufficient capacity to satisfy the demand.



# Availability Modeling

- Another quality attribute with a well-understood analytic framework is availability.
- Modeling an architecture for availability—or to put it more carefully, modeling an architecture to determine the availability of a system based on that architecture—is a matter of determining the failure rates and the recovery times of the components.
- Just as for performance, to model an architecture for availability, we need an architecture to analyze.
- Suppose we want to increase the availability of a system that uses the **Broker pattern**, by applying redundancy tactics.

# Making Broker More Available



Key:

process:



message:







# Availability Modeling

- Three different tactics for increasing the availability of the broker are:
  - active redundancy (hot spare)
  - passive redundancy (warm spare)
  - spare (cold spare).



# Applying Probabilities to Tactics

- Using probabilities to model different tactics
  - When two events A and B are independent, the probability that A or B will occur is the sum of the probability of each event:
  - When two events A and B are independent, the probability of both occurring is
  - When two events A and B are dependent, the probability of both occurring is

# Passive Redundancy

- Assume
  - failure of a component (primary or backup) is independent of the failure of its counterpart
  - assume failure probability of both is the same:  $P(F)$
- Then probability that both will fail is:  $1 - P(F)^2$
- Can also estimate probability of failure given other tactics.
- Then given a cost of implementing appropriate tactic we can do cost/benefit analysis

# Calculated Availability for an Availability-Enhanced Broker

Function	Failure Severity	MTBF (Hours)	MMTR (Seconds)		
			Active Redundancy (Hot Spare)	Passive Redundancy (Warm Spare)	Spare (Cold Spare)
Hardware	1	250,000	5	5	300
	2	50,000	30	30	30
Software	1	50,000	5	5	300
	2	10,000	30	30	30
Availability			0.9999998	0.999990	0.9994

# Maturity of Quality Attribute Models

Quality Attribute	Intellectual Basis	Maturity / Gaps
Availability	Markov models; Statistical models	Moderate maturity in the hardware reliability domain, less mature in the software domain. Requires models that speak to state recovery and for which failure percentages can be attributed to software.
Interoperability	Conceptual framework	Low maturity; models require substantial human interpretation and input.
Modifiability	Coupling and cohesion metrics; Cost models	Substantial research in academia; still requires more empirical support in real-world environments.
Performance	Queuing theory; Real time scheduling theory	High maturity; requires considerable education and training to use properly.
Security	No architectural models	
Testability	Component Interaction Metrics	Low maturity; little empirical validation.
Usability	No architectural models	




# Quality Attribute Checklists

- A quality attribute checklist provides a means of:
  - Checking requirements. Do the requirements capture all of the nuances of a particular quality attribute?
  - Auditing. Does the design satisfy all of the aspects necessary for a certification process.


# Security Checklists

- Security checklists are common.
  - Vendors who accept credit cards should conform to the PCI (Personal Credit Information) standard
  - Electricity producers have security checklists to prevent attacks on critical infrastructure
- Checklists have both:
  - *Product requirements*. E.g. the PCI checklist states that the security code on the back of the credit card should never be stored.
  - *Process requirements*. E.g. patches should be applied promptly and there should be someone who has the organizational responsibility to ensure that they are.

# Security Checklists Example



Sponsored by  
DHS/NCIC/US-CERT



NIST  
National Institute of  
Standards and Technology

## National Vulnerability Database

Automating vulnerability management, security measurement, and compliance checking

Vulnerabilities

Checklists

800-53/800-53A

Product Dictionary

Impact Metrics

Data Feeds

Statistics

FAQs

Home

SCAP

SCAP Validated Tools

SCAP Events

About

Contact

Vendor Comments

Visualizations

### Mission and Overview

NVD is the U.S. government repository of standards based vulnerability management data. This data enables automation of vulnerability management, security measurement, and compliance (e.g. FISMA).

### Resource Status

NVD contains:

- 79931 [CVE Vulnerabilities](#)
- 376 [Checklists](#)
- 249 [US-CERT Alerts](#)
- 4458 [US-CERT Vuln Notes](#)
- 10286 [OVAL Queries](#)
- 115382 [CPE Names](#)

Last updated: 11/2/2016 1:39:22 PM  
CVE Publication rate: 21.97

### Email List

NVD provides four mailing lists to the public. For information and subscription instructions please visit [NVD Mailing Lists](#)

### Workload Index

Vulnerability Workload Index: 11.04

### About Us

NVD is a product of the NIST Computer Security

## National Checklist Program Repository

The National Checklist Program (NCP), defined by the [NIST SP 800-70 Rev. 3](#), is the U.S. government repository of publicly available security checklists (or benchmarks) that provide detailed low level guidance on setting the security configuration of operating systems and applications. NCP is migrating its repository of checklists to conform to the Security Content Automation Protocol (SCAP). SCAP enables standards based security tools to automatically perform configuration checking using NCP checklists. For more information relating to the NCP please visit the [information page](#) or the [glossary of terms](#).

Search for Checklist using the fields below. The keyword search will search across the name, and summary.

Tier: Any.....

Type: Any.....

Target Product: Any.....

Category: Any.....

Authority: Any.....


Keyword:

SearchReset

There are **376** matching records. Displaying matches **1** through **20**.

12345678910 >>>

Tier	Target Product	Product Category	Authority	Last Modified Date	Checklist Name (Version)	Resources
IV	Microsoft Internet Explorer 7	Web Browser	USGCB/TIS	01/25/2016	<a href="#">USGCB Internet Explorer 7 (2.1.x.1)</a>	<ul style="list-style-type: none"><li>GPOs - USGCB IE7 GPOs</li><li>Prose - This is the human readable version of the USGCB settings.</li><li>SCAP 1.2 Content - USGCB Internet Explorer 7 SCAP Content using OVAL version 5.10</li></ul>
IV	Microsoft Internet Explorer 8	Web Browser	USGCB/TIS	04/28/2015	<a href="#">USGCB Internet Explorer 8 (1.3.x.1)</a>	<ul style="list-style-type: none"><li>SCAP 1.2 Content - USGCB Internet Explorer 8 SCAP Content using OVAL version 5.10</li><li>GPOs - USGCB IE8 GPOs</li><li>Prose - This is the human readable version of the USGCB settings.</li></ul>
IV	Microsoft Windows 7 Microsoft Windows 7 x86 (32-bit) Microsoft Windows 7 x64 (64-bit)	Operating System	USGCB/TIS	01/25/2016	<a href="#">USGCB Windows 7 (2.0.x.1)</a>	<ul style="list-style-type: none"><li>GPOs - USGCB Windows 7 GPOs</li><li>Prose - This is the human readable version of the USGCB settings.</li><li>SCAP 1.2 Content - Windows 7 Oval 5.10</li></ul>
IV	Microsoft Windows 7 Microsoft Windows 7 32-bit (X86) Microsoft Windows 7 64-bit (X64)	Operating System	USGCB/TIS	04/28/2015	<a href="#">USGCB Windows 7 Firewall (1.3.x.1)</a>	<ul style="list-style-type: none"><li>SCAP 1.2 Content - Windows 7 Firewall Oval 5.10</li><li>GPOs - USGCB Windows 7 Firewall GPOs</li><li>Prose - This is the human readable version of the USGCB settings.</li></ul>
IV	Microsoft Windows Vista	Operating System	USGCB/TIS	04/28/2015	<a href="#">USGCB Windows Vista (3.0.x.1)</a>	<ul style="list-style-type: none"><li>SCAP 1.2 Content - USGCB Windows Vista using OVAL version 5.10</li><li>GPOs - USGCB Windows Vista GPOs</li><li>Prose - This is the human readable version of the USGCB settings.</li></ul>





# Security Checklists Example

## Checklist Details for USGCB Internet Explorer 7 2.1.x.1 [\(Checklist Revisions\)](#)

### SCAP 1.2 Content:

- [Download SCAP 1.2 Content - USGCB Internet Explorer 7 SCAP Content using OVAL version 5.10](#)
  - Technology Infrastructure Subcommittee (TIS)

### SCAP Expression Information:

SCAP	XCCDF	OVAL	CCE	CVE	CVSS	CPE	OCIL
Expressed	Expressed	Expressed	Expressed	Expressed	Expressed	Expressed	Expressed
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

[Machine-Readable CCE to 800-53 Data Stream](#)

### Supporting Resources:

- [Download Prose - This is the human readable version of the USGCB settings.](#)
  - NIST, Computer Security Division
- [Download GPOs - USGCB IE7 GPOs](#)
  - NIST, Computer Security Division

### Target Product:

Target Product	CPE Name	Product Category
Microsoft Internet Explorer 7	cpe:/a:microsoft:internet_explorer:7 <a href="#">(View CVEs)</a>	• Web Browser

### Checklist Summary:

The purpose of the United States Government Configuration Baseline (USGCB) initiative is to create security configuration baselines for Information Technology products widely deployed across the federal agencies. The USGCB baseline evolved from the Federal Desktop Core Configuration mandate. The USGCB is a Federal government-wide initiative that provides guidance to agencies on what should be done to improve and maintain an effective configuration settings focusing primarily on security. This checklist represents the USGCB guidance for Microsoft Internet Explorer 7.

### Checklist Role:

- Web Browser

### Known Issues:

Spreadsheet containing known issues can be found at [http://usgcb.nist.gov/usgcb/microsoft\\_content.html](http://usgcb.nist.gov/usgcb/microsoft_content.html), under the "Documentation" column.

### Target Audience:

US Federal Agencies. Do not attempt to implement any of the settings without first testing them in a non-operational environment. These recommendations have only been tested on Windows 7 Ultimate 32-bit, Windows 7 Ultimate 64-bit, Windows 7 Enterprise x86, and Windows 7 Enterprise x64. These settings may be applicable to other Windows systems and service packs; however, NIST has not tested other Windows based systems with these settings. Please see the National Checklist Program (NCP) website for configuration guides related to other Windows Based systems and applications. The draft download packages contain recommended security settings; they are not meant to replace well-structured policy or sound judgment. Furthermore, these recommendations do not address site-specific configuration issues. Care must be taken when implementing these settings to address local operational and policy concerns. These recommendations were developed at the National Institute of Standards and Technology, which collaborated with DoD and Microsoft to produce the Windows 7, Windows 7 Firewall, Internet Explorer 8 USGCB. Pursuant to title 17 Section 105 of the United States Code, these recommendations are not subject to copyright protection and are in the public domain. NIST assumes no responsibility whatsoever for their use by other parties, and makes no guarantees, expressed or implied, about their quality, reliability, or any other characteristic. We would appreciate acknowledgement if the recommendations are used.

## Checklist Highlights

<b>Checklist Name:</b>	USGCB Internet Explorer 7
<b>Checklist ID:</b>	170
<b>Version:</b>	2.1.x.1
<b>Tier:</b>	IV
<b>Type:</b>	Compliance
<b>Review Status:</b>	Final
<b>Authority:</b>	Governmental Authority: USGCB/TIS
<b>Original Publication Date:</b>	06/19/2008
<b>Checklist Group:</b>	<a href="#">View</a>



# Quality Attribute Checklists

- Auditor uses for certification
- Architect uses for regulation – can treat checklists as a list of requirements, understood and prioritized
- Safety, Usability checklists are also used
- Considering various kinds of analysis for a system leads to early architectural decisions



# Quality Attribute Checklists

- How are they generated?

- Time consuming activity

- Undertaken by multiple individuals

- Refined, evolved over time

- Domain specialists, quality attribute specialists, architects

# Quality Attribute Checklists

## Example

### Quality Attributes Checklist

This section of the SRS describes any desired attributes or characteristics required of the system that don't provide a function or capability. The following checklist serves as the criteria for quality attributes:

#### Requirements

- 1. Does the section cover the expected system response time?
- 2. Does the section cover the expected system reliability?
- 3. Does the section cover allowances for maintainability and enhancement?
- 4. Does the section cover system restraints?
- 5. Does the section cover system documentation?
- 6. Does the section cover portability to other operating systems?
- 7. Does the section cover system usability requirements?
- 8. Does the section cover testability and deployment?



# Thought Experiments

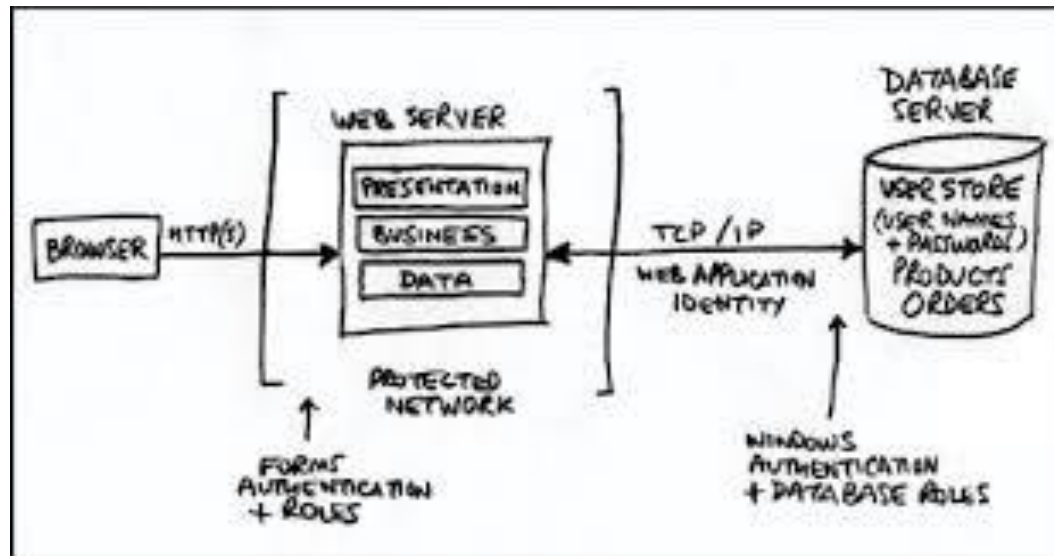
- A thought experiment is mentally or verbally working through a particular scenario.
  - Commonly done by the architect during design to explore alternatives.
  - Also done during evaluation/documentation to convince third parties of the wisdom of particular design choices



# Thought Experiments

- Discussions that developers and architects have on a daily basis in their offices, meetings, lunches, whiteboards, hallways
- Find problems, or verify nonexistence of problems
- Level of formality depends on the context
  - Two people – private discussion, whiteboard
  - Third person – legend added
  - Results included in documentation as design rationale – more details captured

# Thought Experiments



# Thought Experiment Steps

- Enumerate the steps of a use case
- At each step, ask {yourself, the architect}
  - What mechanism is being implemented to support the achievement of which particular quality requirement?
  - Does this mechanism hinder the achievement of other quality attribute requirements?
- Record problems for later deeper analysis or prototype building
- May consider several quality attribute requirements simultaneously





# Back-of-the-Envelope Analysis

- Analysis does not need to be precise or detailed.
- Best data available, past experiences, guesses
- Rough analysis serves for many purposes. E.g. “the volume of traffic generated by this source should be well within the bounds handled by modern infrastructure”
- Only do deeper analysis for questionable areas or important requirements.

# Thought Experiments

- Analysis Paralysis


If architects are being very thorough, problems are significant, stuck in endless meetings

- Time spent on performing a particular thought experiment should be bounded
- Time Boxing - set deadline on the length of discussion
- Not spending more than the cost of the analysis – cost estimation

# Thought Experiments

Consider, for example, the following thought experiment: imagine defining a Web page in a rich text document editor, such as Microsoft Word or Open Office. Now, save this document as HTML and imagine the challenge of maintaining and modifying this HTML document over time. Alternatively, one may just try to edit the Web page in the document editor, saving a new HTML document every time a revised Web page must be published. Although the aesthetics of this process are questionable, it should at least get the job done: you will have succeeded in publishing the information you wished to on the Web.

Other vocabulary differences exist because architecture models and code express distinct ideas. Consider a thought experiment where you express your architecture model in UML and you also automatically generate a UML model of your source code. When you compare these two UML models, you will find differences. For example, your source code model does not express the component types or instances found in your architecture model. While method call connectors and event bus connectors both show up in your architecture model, only method calls are seen in your source code model. The architecture model and source code have different vocabulary because each expresses ideas the other does not.



# Experiments, Simulations, and Prototypes

- Experiments, simulations, prototypes  
exploration of tradeoffs, alternative ways to analyze
- Experiments often included in development phase
- Analysis of the results consumes time
- Cost, schedule for design and analysis of experiments  
should be included in project schedules
- Measured rather than estimated

# Simulations

- Event based simulators exist that can be used to simulate behavior of system
  - Must create the simulation
  - Must have a variety of different loads and responses to check for
- Web conference system
  - Client simulator – tens of thousands of clients are simultaneously interacting with a conferencing server
  - Instrumentation - measuring loads in server and database

# Experiments, Simulations, and Prototypes

- Many tools can help perform experiments to determine behavior of a design
  - Request generators can create synthetic loads to test scalability
  - Monitors can perform non-intrusive resource usage detection.
- These depend on having a partial or prototype implementation.
  - Prototype alternatives for the most important decisions
  - If possible, implement prototype in a fashion so that some of it can be re-used.
  - Fault injection tools can induce faults to determine response of system under failure conditions.



# Analysis During Requirements and Design

- Different types of analysis are done at different stages of the life cycle
- Requirements:
  - Analytic models/back of the envelope analysis can help capacity planning
  - Checklists can help ensure capture correct set of requirements
- Design:
  - Prototypes can help explore design options
  - Analytic models or simulation can help understand potential bottlenecks
  - Checklists can help determine if used a correct mechanism



# Analysis During Implementation or Fielding

- Experiments and synthetic load tests can be used during the implementation process or after fielding
- Monitors can be used after fielding to determine actual behavior and find bottlenecks.





# Analysis at Different Stages of the Lifecycle

Life-Cycle Stage	Form of Analysis	Cost	Confidence
Requirements	Experience-based analogy	Low	Low-High
Requirements	Back-of-the-envelope	Low	Low-Medium
Architecture	Thought experiment	Low	Low-Medium
Architecture	Checklist	Low	Medium
Architecture	Analytic Model	Low-Medium	Medium
Architecture	Simulation	Medium	Medium
Architecture	Prototype	Medium	Medium-High
Implementation	Experiment	Medium-High	Medium-High
Fielded System	Instrumentation	Medium-High	High

# Summary

- Analysis is always a cost/benefit activity
  - Cost is measure of creating and executing the analysis models and tools
  - Benefit depends on
    - Accuracy of analysis
    - Importance of what is being analyzed
- Analysis can be done through
  - Models for some attributes
  - Measurement
  - Thought experiments
  - Simulations
  - Prototypes

- 
- 
- Enumerate responsibilities or actions that need to be carried out for providing a “cancel” operation in a user’s interface or computer system

