Unit I
Workstations, Servers and Services

Table of Contents

1. Workstations

1.1 Workstation Architecture

- One of the most fundamental functions of an IT department is to manage the fleet of workstations used by the organization.
- Workstations are computers used by people. Whether it is a desktop PC in an office or a laptop computer carried from place to place, workstations are the computers people use to get work done.
- The experiences people have when using their workstations are determined by design decisions.
- Issues important for the customers are:
    - Locality: Is the workstation available where and when it is needed? A workstation is not useful if it isn't where the user needs it.
    - Reliability: Is the workstation reliable or does it lock up or crash frequently? Reliability starts with good hardware but increasingly depends on software and firmware updates being received in a timely manner.
    - Productivity: Can the customer work with minimal obstruction?
    - User agency: Do the users have agency or control over their environment?
    - Currentness: How much lag time is there between when new features ship and when they are installed and ready to use on the machine?

1.1.1 Fungibility

- Workstations should be a fungible resource: Any one unit should be able to substitute for any other.
- It should be possible for anyone with an account to log into any workstation.
- The ability to log into any workstation improves access locality.
- Computers break. If a person is tied to a particular machine, he or she is unable to work until the computer is fixed.
- When the software on a machine is corrupted, it is convenient to simply wipe and reload the machine from scratch.

1.1.2 Hardware

- Selecting the physical hardware is the most fundamental choice one can make.
- There are laptops and desktops, mobile devices and tablets, and other physical form factors to choose from.
- There is also the decision between physical workstations and virtual workstations, or bring your own device (BYOD) strategy is employed.

1.1.3 Operating System

- The operating system is the software layer between the applications and the hardware.
- It is the mediator that enables many applications to share the same computer.

1.2 Workstation Hardware Strategies

1.2.1 Physical Workstations

- In an environment where physical hardware is used for people's workstations, whether the user has a laptop or desktop is one of the most fundamental choices to be made.
- Other considerations include vendor selection and product line selection.
- **Laptop vs Desktop**
    - Desktops are generally more expandable.
    - They have slots for add-on cards and video processor cards, and sockets for additional memory.
    - They have internal space for additional storage.
    - Laptops are generally more expensive than comparable desktop models.
    - A laptop's mobility requires it to be more rugged because people drop, bang, and otherwise mistreat these machines.
- **Vendor Selection**
    - Organizations may standardize on one vendor, select a fixed set of vendors, or have, essentially, a policy of supporting all vendors.
    - Standardizing on one vendor makes support easier and makes it easier to qualify for bulk purchase discounts.
    - Having multiple vendors permits price competition but incurs a bigger support cost.
    - However, the additional support cost may be offset by playing one vendor off another to get the best price.
    - Before starting a vendor selection process, document your criteria, evaluation, and rating methods.
- **Product Line Selection**
    - Most vendors have multiple product lines.
    - Either, emphasize on lowest initial cost or purchase price OR emphasize lowest total cost of ownership (TCO) OR Emphasize performance.
        - **Lowest Initial Cost**
            - The lowest initial purchase price is achieved by sacrificing the features that would make the machine less expensive in the long term or would make managing many machines less expensive.
        - **Total Cost of Ownership**
            - A product line that emphasizes TCO does so by focusing on issues that would save money when an organization is maintaining a large fleet of machines.
            - These product lines are often called the "enterprise line" or "business line."
        - **Performance**
            - The product line that emphasizes performance is often called the "engineering" or "power" line.
            - It includes features required by engineering applications such as computer-aided design (CAD) and computer-aided engineering (CAE), which require high-end graphics, vector processing, and heavy-duty computation.

1.2.2 Virtual Desktop Infrastructure

- The point of VDI is to turn workstations into a centrally managed network service to reduce administrative and hardware costs.
- With VDI, the user's applications run on a virtual machine (VM) in a VM server cluster elsewhere on the network.
- The benefits: savings due to reduced hardware costs, along with increased ease of management.
    - **Reduced Costs**
        - The thin client hardware is inexpensive since it is designed for the single purpose of speaking a remote access protocol such as Windows Terminal Services, VMWare Horizon, or Citrix.
    - **Ease of Maintenance**
        - The VM server infrastructure can be upgraded more cost-effectively because it does not require the physical labor of visiting each thin client.
    - **Persistent or Non-Persistent**
        - With non-persistent VDIs, the virtual desktop is created from scratch using a standard golden image each time it is needed.
        - Non-persistent VDIs do not permit customization.
        - With persistent VDIs, there is a one-to-one mapping between users and virtual desktops, and each user's desktop is created from a separate disk image.
        - The user's settings are saved at the end of each session, and appear each time at login, just like with a physical workstation.

1.2.3 Bring Your Own Device (BYOD)

- Bring your own device (BYOD) is a service model where users supply their own hardware device rather than use one provided by their employer.
- Strategies:
    - In **BYOD-only** strategy, the IT organization supplies no hardware and requires users to supply their own device. BYOD-only users are usually a specific subset of users, such as people involved in a particular program or short-term project.
    - In a **BYOD mixed model** strategy, the IT organization supplies workstations but also permits BYOD. This is often a productivity enhancement, or a way to provide mobile access.
    - A **BYOD-lite** strategy enables a small number of specific applications to be accessed, possibly only from registered devices made by approved vendors.

1.3 OS Installation

- The OS installation process erases any existing operating system and installs a new one, bringing the OS to Evard's "clean" state.
- Installation is best achieved through automation. A fully automated installation process assures that each machine is configured correctly.
- Doing it manually is prone to errors and, to be blunt, is pretty boring.
- After installation, many subsystems and components need to be configured.
- The specific settings may change over time, so we need a way to both make the initial configuration changes and update them.

2. Servers

- The term “server” is overloaded and ambiguous. It means different things in different contexts.
- The phrase “web server” might refer to a host being used to provide a web site (a machine) or the software that implements the HTTP protocol.

2.1 Hardware Strategies

The three most common strategies are:

- **All eggs in one basket**: One machine used for many purposes.
- **Beautiful snowflakes**: Many machines, each uniquely configured
- **Buy in bulk, allocate fractions**: large machines partitioned into many smaller virtual machines using virtualization or containers

There are variations and alternatives:

- **Grid computing**: Many machines managed one as unit
- **Blade servers**: A hardware architecture that places many machines in one chassis
- **Cloud-based computing services**: Renting use of someone else’s server
- **Software as a service (SaaS)**: Web-hosted applications
- **Server appliances**: Purpose-built devices, each providing a different service

2.2 Hardware Features

- **Workstation VS Servers**
    - Workstations and servers are fundamentally different.
    - The differences stem from the ways that each is used, which gives us a different set of requirements for each.
    - Server hardware is different from workstation hardware.
    - Server operating systems are different from workstation operating systems.
    - Management of servers are done differently, from patching to configuration.
    - Server Hardware Design has:
        - **More CPU Performance**
        - **High Performance I/O**
        - **Expandability:** Servers usually have more physical space inside for hard drives and more slots for cards and CPUs
        - **Upgrade Options:** Servers are designed for growth. Have ability to add CPUs or replace individual CPUs with faster ones.
        - **Rack Mountable**
        - **Front and rear access**
        - **High availability**
        - **Remote Management**

- **Server Reliability**
    - All devices fail. We need to accept that fact, and prepare for failure.
    - As servers need to be more reliable, and have higher uptime, buy server hardware with additional features for reliability and data integrity.
    - In addition, servers should be housed in a restricted-access, climate-controlled environment, with protected power—i.e., a computer room or datacenter.
    - **Levels of Redundancy**: Servers often have internal redundancy such that one part can fail and the system keeps running.

    - **Data Integrity:** Hard disks and SSDs eventually wear out and die. A plan to deal with this eventuality is a must. Not having a plan is assuming our storage systems will last forever.
    - **Hot-Swap Components:** Server hardware has many redundant components, and these components should be hot-swappable. Hot-swap refers to the ability to add, remove, and replace a component while the system is running.
    - **Servers should be in Computer Room:** Servers should be installed in an environment with proper power, fire protection, networking, temperature and humidity control, and physical security.
- **Remotely Managing Servers**
    - Servers need to be maintained remotely.
    - Remote management: It should be possible to do all system administration tasks involving the machine from a remote location, except physical labour such as adding and removing physical hardware.
    - It should be possible to remotely access the machine's console and, optionally, have remote control of the power switch.
    - The benefit of permitting you to operate a system's console when the machine is in a bad state or otherwise disconnected from the network.
    - Important as servers are often housed in machine rooms located around the world.
    - Requiring hands-on access for daily tasks is inefficient.
    - Security implications must be considered when you have a remote console.
    - Considered authentication and privacy systems.
- **Selecting Vendors with Service Experience**
    - Some vendors have years of experience designing servers, and it shows.
    - They build hardware with the server-related features listed earlier, as well as include little extras that one can learn only from years of market experience.
    - Select vendors that are known for building reliable hardware. Some vendors cut corners by using consumer-grade parts; others use premium-quality parts.

2.3 Hardware Specifications

- **CPU**
    - The most fundamental decision in selecting a server is the CPU—in particular, the choice of a few high-speed cores versus many slow cores.
    - Depending on the architecture of the software that will run on the machine, this decision can determine whether your service is fast and efficient or underutilizes the capacity of the machine.
- **Memory**
    - Random access memory (RAM) is where running programs and their data are stored.
    - A server needs to have enough RAM to support the applications that are running, plus any RAM used by the OS and any support programs that are also running.
- **Network Interfaces**
    - Servers often have multiple network interface cards (NICs) because they are connected to different networks, or because they are ganged or grouped together for increased bandwidth or resiliency.
    - A server needs network bandwidth proportional to the number of clients accessing it.
    - A server with many clients needs more bandwidth than one with fewer clients, assuming all clients generate the same amount of bandwidth.

- **Power Supplies**
  - The second-most failure-prone component in a server is the power supply.
  - It is very common to have multiple redundant power supplies so that if one fails, the system can keep running.
  - Each power supply should also have a separate power cord
  - Each power supply should draw power from a different source: a separate circuit or uninterruptible power supply (UPS).
  - Generally, each power distribution unit (PDU) in a datacenter is its own circuit, so plugging each power cord into a different PDU assures two power sources.
- **Disks: Hardware vs Software RAID**
  - **Software RAID** is part of the operating system.
  - This setup provides RAID via device drivers.
  - The benefit of this approach is that it is inexpensive, because it is included in the price of the operating system.
  - The downside is that it is usually not as fast because it is running on the same CPUs as the rest of the system.
  - **Hardware RAID** is implemented in a hardware device, usually a card that is plugged into one of the expansion slots of the server.
  - The hard disks plug into this device, and it manages the disks as one or more RAID groups.
  - The server sees each RAID group as a hard disk. It has no idea that the hard disk presented to it is actually a bunch of disks in a RAID configuration.
  - Hardware RAID usually provides higher performance and more features than software RAID.
  - The typical hardware RAID controllers implement a wide range of RAID levels.
  - They usually have dedicated hardware that performs parity calculations.

3. Services

- Services are the applications customers need and use.
- Service means: all the parts that make an application work:
    - **Software**
    - **Hardware**
    - **Operations that bring it all together.**
- Service might be a single piece of software on a single machine, or many different software components across many machines that work in cooperation to provide a service.
- Service may have direct clients such as web-based application, or something invisible such as database.
- Network Administration is about providing services.
- Any workstations or devices are useful only if they are actively providing a service.
- A good service meets client's requirements, is reliable, and is maintainable.
- A running service requires maintenance and upkeep.
- Requirements changes over time; new updates must be deployed periodically to fix bugs and add features.
- Systems fail thus disaster recovery should be planned.
- **Services Make the Environment**
    - Utility of the design is based on the quantity and quality of services provided.
    - Small offices have few services as they rely on ISP for foundational services.
    - Large corporations have a rich environment of services, from the applications that run the company, to the foundational services and infrastructure on which they depend.
    - Basic Services: -
        - User visible applications.
        - Ex: Printer, Email, File Storage, VoIP or phone services.
    - Foundational Services: -
        - Creates the platform that other services rely on.
        - Ex: DNS, DHCP, Directory Services, Network Access, and Internet Gateways

3.1 Requirements

- Starting with a Kick-Off Meeting
    - To gather requirements, it is critical to start off with a f2f meeting.
    - Should have all the stakeholders present.
    - Can help get the agreement on:
        - The goal of new services
        - The timeline for completion
        - An approximate budget
- Gathering Written Requirements
    - Requirements are a list of what the service will be able to do.
    - Should list desired functionality, features, and capabilities.
    - Focus on the end goal.
    - Should not include engineering and implementation details.
    - This list guides all the other steps: Design, Engineering, Implementation, Testing
    - Customer Requirement
        - Features and functionality of the service to be built
    - Operational Requirement
        - Describe what is needed to make the system work and be maintainable.

        - Budget Requirements, Security Access Requirements, etc.
- Benefits:
    - **Transparency**: everyone can read the document
    - **Fewer gaps**: reduces errors caused by misheard or forgotten requests
    - **Fewer misunderstandings**: seeing the decision on writing verifies everyone agrees
    - **Buy in and approval**: provides an ability to get buy in and management approval in an accountable way
    - **Fixed scope**: provides a mechanism for handling additional feature requests
    - **Accountability:** two-way street for pointing out the missing or incomplete requests.
- **Customer Requirements**
    - Describing Features
        - Focus on "what" not "how"
        - Ex: "user should be able to send email" not "there should be a button on the left that the user clicks to send email"
    - Questions to Ask
        - How do customers intend to use the new service?
        - Why do they need this feature?
        - Where, when, and how will the system be accessed?
        - How critical is the service to them, and which level of availability and support do they need for the service?
- **Service Level Agreements**
    - Requirement documentation should include SLA, or some general specifications that can be used to build an SLA.
    - SLA includes the services that will be provided and the level of support they receive.
    - Categorizes problems by severity and commits to response times for each category.
- **Handling Difficult Requests**
    - Requirement gathering is a collaborative process with ultimate goal of finding a middle ground.
    - Ex: Clients asks that the service should be available 24/7 or have 100 percent uptime.
    - Unrealistic as it may seem, step back and take few deep breathes.
    - Explain the difference of amount between 99 percent and 99.9 percent availability.
    - Ensure that the service will be available when the user needs it according to the SLA.
- **Operational Requirements**
    - **System Observability**:
        - Key to manageable system is to make it observable. Visibility can be promoted by maintaining logs of evens, transitions, and operations.
    - **Remote and Central Management:**
        - Services should be remotely managed. Need to be on the console of a machine to do rudimentary maintenance and configuration will be time consuming.
    - **Scaling Up or Out:**
        - If service is successful, more people would use it. Over time it should be able to handle more users, more transactions, more capacity.
        - **Scale Up:** Getting a bigger system. To handle more, existing system is replaced with bigger and faster system.
        - **Scale Out:** The system is expanded by adding more replicas.
    - **Software Upgrades:**

        - A system must have a reasonable way to upgrade the software and firmware. It should be made possible to update or upgrade the service by placing it within the scheduled maintenance window.
    - **Environment Fit:**
        - If the service fits better in the existing IT environment, easier it is to adopt it. Integration issues are reduced and less training or skill development is required.
    - **Support Model**:
        - Key component in the success of a new service is support process. Define the support model in advanced.
    - **Service Requests:**
        - Most services will include service requests that people can raise. Depending on the service provided, these may be access or authorization requests, data changes, configuration changes, resource requests, and so on.
    - **Disaster Recovery:**
        - Failure happens, hardware fails, building loses power, internet access gets cut, and so on. The most basic disaster recovery requirement is to plan a way to perform backups and restores.

3.2 Planning and Engineering

- After the product is selected, we need to engineer how it will be configured and operated in our environment.
- Both the Service Architecture and the Local Engineering Plan needs to be defined and documented.
- Service Architecture is high level description whereas Local Engineering Plans is the specifics for a particular machine or site.
- Typically, Service Architecture is developed, and then the local engineering plan is created based on that architecture.
- **General Engineering Basics**
    - Follow vendor recommendations
    - Mirror your boot disk
        - Disks fail. Often the most difficult disk to replace is the boot disk, as it requires reinstalling all the software. Mirror the boot disk so that when a disk fails, the system will keep running and give you time to replace the failed disk
    - Run services on servers, not workstations
    - Run services from computer room, not offices:
        - Provides physical security, high-speed networks, reliable power, and cooling.
    - Restrict console access
    - Leverage the existing infrastructure
        - New services should be integrated into existing systems and processes. Existing monitoring, alerting and system management, and service request system can be used along with backup systems, and configuration and patch management system.
- **Simplicity**
    - When engineering a service, first consideration should be simplicity.
    - Strive to create the simplest solution that satisfies all the requirements.

- A simple service will be the easiest to maintain, easiest to expand, and easiest to integrate with other system.
- During engineering phase, create multiple design proposals. Consider each for its simplicity, manageability, cost, and so on.

- **Vendor Certified Designs**
  - Vendors often publish **certified engineering designs**, or **vendor best practices**.
  - Even then, some local decisions should be made, such as hardware vendor, model number, specific configurations, IP addresses, network switch connections, deployment racks, optional configuration parameters, etc. which are local engineering design.
- **Dependency Engineering**
  - A system is never more reliable than the most unreliable system it has dependency on.
  - The larger and more complex a system, the more dependencies it has.
  - **Primary Dependencies:**
    - core system reliability excluding any external dependencies.
  - **External Dependencies:**
    - many external dependencies that influence a service's reliability can be controlled. A dependency matrix can be used to list which subsystem depend on another subsystem.
    - Dependency matrix resembles a tree diagram, with each level relying on services in the levels below, and no cross dependencies.
    - When engineering new service, dependency matrix should be created to document dependencies. This can also be useful to find high-risk dependencies and mitigate them
  - **Dependency Alignment**
    - Dependencies can be realigned to smaller failure domains.
    - Failure Domain is all the services, locations, and users that are adversely affected when a particular component fails or is taken out of service.
    - Failure Domain can be categorized into:
    - Hardware Failure Domains
    - Service Failure Domains
    - Location Failure Domains
- **Decoupling Hostname from Service Name:**
  - consider an email service running on a machine called *mail*.
  - What do you do when it is time to replace the mail server with a new one?
  - You could call the new machine *newmail* or *mail2*, but then users would need to reconfigure their email clients, which would be a hassle.
  - You could call the new machine *mail* and rename the old machine to be *oldmail*, but now everyone would start referring to the new host, possibly before it is ready.
  - Design systems to use aliases for the service name.
  - Name the machines simply *server1*, *server2*, and so on.
  - Provide aliases such as mail, calendar, and so on.
  - These aliases can be changed to point at a new machine as needed.
- **Support**
  - During planning and engineering phase, consider how the service will be supported during its lifetime.

    - Support includes technical aspects such as detecting problems, patching, upgrading, performing backups, and scaling the service, and so on.
    - **Monitoring:**
        - It isn't support if it isn't monitored. A service should be monitored for availability, problems, performance, and capacity-planning.
        - It is necessary to document, what should be monitored and how.
    - **Support Model**
        - Support model needs to specify who supports the various components of the service, and what the OLAs, and SLAs for each component are.
    - **Service Request Model**
        - The local engineering plan should include a service request model (SRM).
        - SRM defines which requests users can make relating to this service, who can make requests, how they make those requests, which approvals are required, which change control policies apply, and which SLA is used for turning around those requests.
    - **Documentation**
        - Operational procedures must be defined.
        - These are different for every service, but generally include backups and restores, business continuity, or disaster recovery plans, tasks related to onboarding new users, disconnecting users who are leaving, performing periodic tasks, and anything else required to keep the service running.
- **Planning for Problems**
    - While a service launch is exciting and fulfilling, the problem with launching a new service is that something always goes wrong.
    - The road to launching a service is full of tricks and traps that catch us in the most non-obvious ways:
        - Users with a particular web browser can't access it,
        - the service doesn't work for remote users,
        - the production environment doesn't have enough space,
        - the documentation is confusing, and so on.

3.3 Service Launch

- Planning for Problems
- While a service launch is exciting and fulfilling, the problem with launching a new service is that something always goes wrong.
- The road to launching a service is full of tricks and traps that catch us in the most non-obvious ways:
    - Users with a particular web browser can't access it,
    - the service doesn't work for remote users,
    - the production environment doesn't have enough space,
    - the documentation is confusing, and so on.
- **The Six Step Launch Process**
    - Step 1: Define the Ready List
        - Define a list of tasks that must be complete before the launch may proceed
        - "Must have" feature set
        - "Would be nice" feature set
        - Bugs to be fixed

    - Step 2: Work the List
        - Schedule, work on, and mark as complete each item in the list
    - Step 3: Launch the Beta Service
        - Launch to one of many test areas where the assembled pieces are verified
        - Dev
        - Quality Assurance
        - User Acceptance Testing
        - Beta or Early Release
        - Production
    - Step 4: Launch the Production Service
        - Launch the service into production
    - Step 5: Capture the Lessons Learned
        - Capture the lessons learned for posterity and to educate the organization
        - Executive Summary
        - Timeline
        - Successes
        - Failures
        - Short Term Changes
        - Long Term Changes
    - Step 6: Repeat
        - Begin the process again
- **Launch Readiness Review**
    - **Launch Readiness Criteria (LCR)**
        - The LRC include basic tasks such as verifying that stakeholders have approved the launch,
        - there is a disaster recovery plan in place,
        - data backups are working and have been tested, and
        - various security audits have been completed.
        - The LRC also include items that come from the lessons learned during past launches.
    - **Sample Launch Criteria**
        - Service is monitored
        - Monitoring alerts are configured
        - Backups and restores are tested and working
        - Standard authentication, authorization, and access control have been tested and are working
        - SLA is defined
        - Service request model is in place
        - User documentation is complete
        - Operational documentation is complete
        - User training is complete
        - Load testing is complete
        - Ten-times response plan is developed and tested
    - **Organizational Learning**
        - Sometimes problems identified during a retrospective become candidates to add to the launch readiness checklist.
        - The LRC is how we capture learning in an organizationally beneficial way.
        - LRC are a way to improve the organization’s collective knowledge.

        - They communicate learning from one group to another so that people do not repeat one another's mistakes.
    - **LRC Maintenance**
        - As the LRC list grows, it can become a bureaucratic burden.
        - Prevent by being cautious about adding new items, and by removing obsolete items.
        - Each item on the list incurs a cost to the company. Whether the cost is just answering a question on the LRC form or the actual mitigation, it is work that someone must do for each launch.
    - **Common Launch Problems**
        - Processes Fail in Production
        - Unexpected Access Methods
        - Production Resources Unavailable
        - New Technology Failures
        - Lack of User Training
        - No backups

3.4 Disaster Recovery

- The most important aspect of disaster planning is understanding which services are the most critical to the company and what the time constraints are for restoring those services.
- Which disasters are likely to happen and how costly they would be should be considered to perform risk analysis and determine the budget for limiting the damage.
- A disaster plan should be built with consideration of those criteria.
- It should account for the time to get new equipment, retrieve the off-site backups, and rebuild the critical systems from scratch.
- Look for simple ways to limit damage, as well as more complex and expensive ways.
- Preparations that are automatic and become part of the infrastructure are most effective.
- The team members must be familiar with their individual roles and should practice disaster recovery a few times a year.
- **Risk Analysis:**
    - The first step in building a disaster-recovery plan is to perform a risk analysis.
    - A risk analysis involves determining which disasters the company is at risk of experiencing and what the chances are of those disasters occurring.
    - The analyst also determines the likely cost to the company for each disaster should it occur.
- **Legal Obligations:**
    - Commercial companies have legal obligations to their vendors, customers, and shareholders in terms of meeting contract demands.
    - Public companies have to abide by the laws of the stock markets on which they are traded. Some industries, such as banking, are governed by local laws relating to business continuity.
    - Universities have contractual obligations to their students.
    - Building codes and work-safety regulations also must be followed.
    - The legal department should be able to elaborate on these obligations.
- **Damage Limitation**:
    - Damage limitation is about reducing the cost of the disasters.

    - Most damage limitation efforts involve additional cost to the company and are subject to the cost/benefit analysis that is performed by the risk analysts.
    - Some damage limitation can come at little or no cost to the company through advance planning and good processes.
- **Preparation**:
    - Even with a reasonable amount of damage control in place, you may still experience a disaster.
    - Part of your disaster planning must include preparation for this eventuality.
    - Being prepared for a disaster means being able to restore the essential systems to working order in a timely manner, as defined by your legal, ethical, and fiduciary obligations.
    - Restoring services after a disaster can require rebuilding the necessary data and services on new equipment if the old equipment is not operational.
    - Arrange a source of replacement hardware in advance from companies that provide this service.
- **Security Disasters**:
    - A growing concern is security disasters.
    - Someone breaks into the corporate web site and changes the logo to something obscene.
    - Someone steals the database of credit card numbers from an e-commerce site. A virus deletes all the files it can access.
    - Unlike with natural disasters, no physical harm occurs with a security disaster, and the attack may not be from a physically local phenomenon.
    - A similar risk analysis can be performed to determine the kinds of measures required to protect data.
    - Architecture decisions, for example, always have a risk component. One can manage this risk in many ways—for example, by building barriers around the system or by monitoring the system so that it can be shut down quickly in the event of an attack.