2025 CISSP Mentor Program

SESSION 12 pt. 1

John Kennedy, Sec +, CISSP

INTRODUCTION

Agenda

- Welcome
- Reminders
- Introduction
- Chapter 20 - Software Development Security

FRSECURE CISSP MENTOR PROGRAM LIVE STREAM

THANK YOU!

Quick housekeeping reminder.

- The online/live chat that's provided while live streaming on YouTube is for constructive, respectful, and relevant (about course content) discussion **ONLY**.
- At **NO TIME** is the online chat permitted to be used for disrespectful, offensive, obscene, indecent, or profane remarks or content.
- Please do not chat about controversial subjects, and please **NO DISCUSSION OF POLITICS OR RELIGION**.
- Failure to abide by the rules may result in disabling chat for you.
- **DO NOT share or post copywritten materials. (pdf of book)**

GETTING GOING…

Managing Risk!

Study Tips:

- Study in small amounts frequently (20-30 min)
- Flash card and practice test apps help
- Take naps after heavy topics (aka Security Models)
- Write things down, say them out loud, explain them to others
- Use the Discord Channels
- Exercise or get fresh air in between study sessions

Let’s get going!

2025 CISSP MENTOR PROGRAM

SCHEDULE

[Our plan]

Class Number	Date	Topic	Lead Mentor
1	4/23/25	Session 1 – CISSP Mentor Program Introduction	Evan
2	4/30/25	Session 2 – Chapter 1 & 2 (pg. 1–114)	Evan
3	5/7/25	Session 3 – Chapter 3, 4, & 5 (pg. 121–221)	Christophe
4	5/14/25	Session 4 – Chapter 6 & 7 (pg. 227-311)	Evan
5	5/21/25	Session 5 – Chapter 8 & 10 (pg. 317-353, 443-483)	Christophe
6	5/28/25	Session 6 – Chapter 9 (pg. 359-435)	Brad
7	6/4/25	Session 7 – Chapter 11 (pg. 491-574)	Evan
8	6/11/25	Session 8 – Chapter 12 & 13 (pg. 581-674)	John
9	6/18/25	Session 9 – Chapter 14 & 15 (pg. 681-764)	Jake
10	6/25/25	Session 10 – Chapter 16 & 17 (pg. 769-862)	Brad
11	7/2/25	Session 11 – Chapter 18 & 19 (pg. 869-945)	Evan
12	7/9/25	Session 12 – Chapter 20 & 21 (pg. 951-1048)	John/Jake
13	7/16/25	Session 13 – Practice Tests & Final Prep	All
14	7/23/25	Session 14 – Practice Tests & Final Prep	All

AGENDA – SESSION 8

Chapter 20

Chapter 20 - Software Development Security

THE CISSP EXAM TOPICS COVERED IN THIS CHAPTER INCLUDE:

- Domain 3.0: Security Architecture and Engineering
  - 3.5 Assess and mitigate the vulnerabilities of security architectures, designs, and solution elements
    - 3.5.3 Database systems
- Domain 8.0: Software Development Security
  - 8.1 Understand and integrate security in the Software Development Life Cycle (SDLC)
    - 8.1.1 Development methodologies (e.g., Agile, Waterfall, DevOps, DevSecOps, Scaled Agile Framework)
    - 8.1.2 Maturity models (e.g., Capability Maturity Model (CMM), Software Assurance Maturity Model (SAMM))
    - 8.1.3 Operation and maintenance
    - 8.1.4 Change management
    - 8.1.5 Integrated Product Team
  - 8.2 Identify and apply security controls in software development ecosystems
    - 8.2.1 Programming languages
    - 8.2.2 Libraries
    - 8.2.3 Tool sets
    - 8.2.4 Integrated Development Environment

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

2025 CISSP MENTOR PROGRAM

AGENDA – SESSION 8

Chapter 20

Chapter 20 - Software Development Security

THE CISSP EXAM TOPICS COVERED IN THIS CHAPTER INCLUDE:

- Domain 8.0: Software Development Security (cont.)
    - 8.2.5 Runtime
    - 8.2.6 Continuous Integration and Continuous Delivery (CI/ CD)
    - 8.2.7 Software Configuration Management (CM)
    - 8.2.8 Code repositories
  - 8.3 Assess the effectiveness of software security
    - 8.3.1 Auditing and logging of changes
  - 8.4 Assess security impact of acquired software
    - 8.4.1 Commercial-off-the-shelf (COTS)
    - 8.4.2 Open source
    - 8.4.3 Third-party
  - 8.5 Define and apply secure coding guidelines and standards
    - 8.5.2 Security of application programming interfaces (API)
    - 8.5.3 Secure coding practices
    - 8.5.4 Software-defined security

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Joke break

What's a hacker’s favorite season?

Phish-ing season.

Source: ChatGPT

2025 CISSP MENTOR PROGRAM

CHAPTER 20

Software Development Security

Software development is a complex and challenging task undertaken by developers with many different skill levels and varying levels of security awareness. Applications created and modified by these developers often work with sensitive data and interact with members of the general public. That means that applications can present significant risks to enterprise security, and information security professionals must understand these risks, balance them with business requirements, and implement appropriate risk mitigation mechanisms.

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Introducing Systems Development Controls

Why Controls Are Critical in Custom Software:

- Custom software ≠ Secure software
- Vulnerabilities may include:
  - 🕳 Backdoors
  - 💥 Buffer overflows
  - 🧑‍💻 Poor coding practices

Mitigation Strategy:

- Integrate security **across the Software Development Life Cycle (SDLC)**
- Use **organized, methodical processes**
- Align functional and **security requirements**

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Software Development

Embed Security Early

- Design with **security from day one**
- Prioritize protection in:
  - 🛡️ Critical apps
  - 🔐 Sensitive data processing

Why Early Matters

- Easier to build security **in**
- Costlier to bolt it **on**

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Programming Languages & Security Implications

Types of Programming Languages

- 🧠 **Machine Language**: Binary (1s and 0s); CPU-specific
- 🔧 **Assembly Language**: Mnemonics for CPU instructions
- 💬 **High-Level Languages**: C, Python, Java, etc.

Execution Methods

- ⚙️ **Compiled** (C, Java): Source → Executable
- 📄 **Interpreted** (Python, JS): Source runs via interpreter
- 🧩 **Runtime (e.g., JVM)**: Adds OS portability

Security Considerations

- Compiled code: harder to inspect, easier to hide backdoors
- Interpreted code: readable by users, but easier to tamper with
- Tools: Decompilers, Disassemblers, Obfuscation

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Shared Libraries & Security Risks

What Are Libraries?

- Reusable code components (e.g., sort, math, ML)
- Available as:
  - 🆓 **Open-source**
  - 💼 **Commercial**
  - 🏢 **Internal/private**

Real-World Risk Example:

- 🩸 **Heartbleed (CVE-2014-0160)**
  - Flaw in OpenSSL (TLS/SSL library)
  - Affected thousands of apps unknowingly
  - Mass emergency patching effort

Security Takeaway:

- Always know **what's inside your code**
- Monitor for **library vulnerabilities**
- Keep components **updated and vetted**

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Development Tool Sets (IDEs)

What Are IDEs?

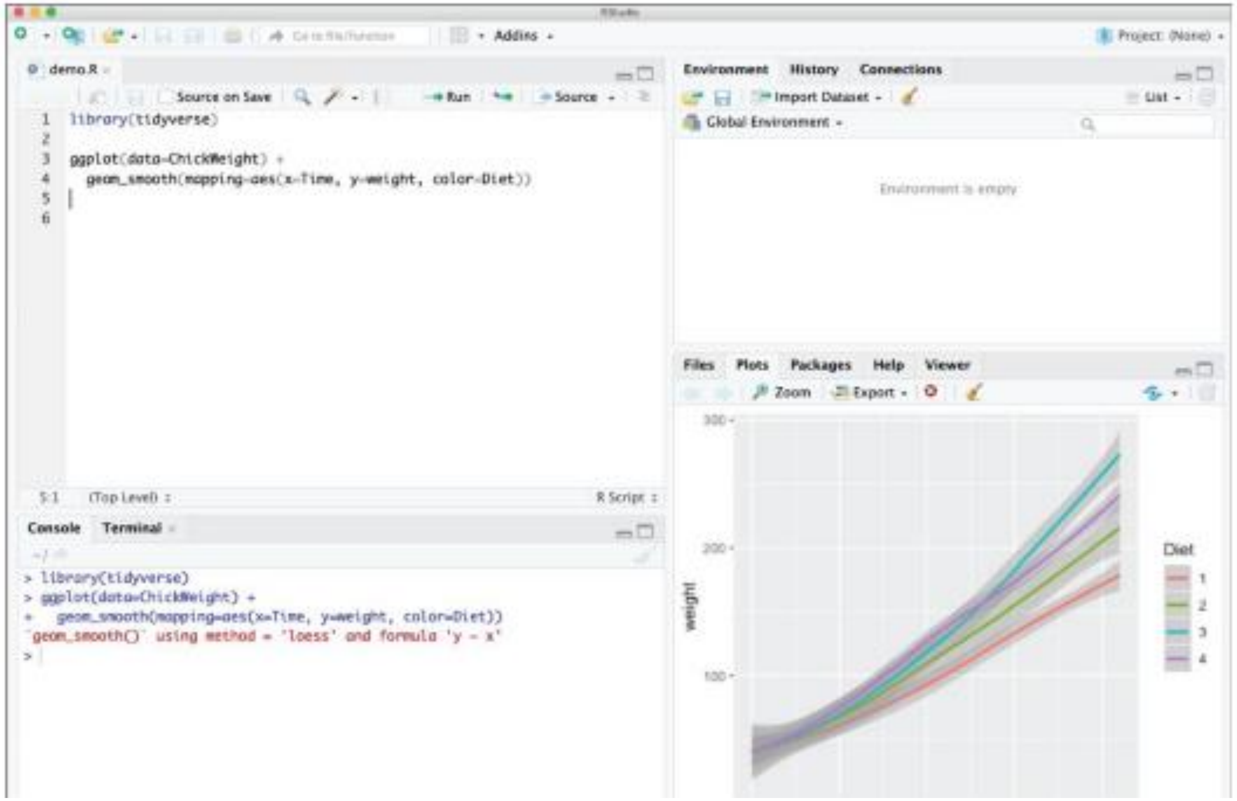
- **Integrated Development Environments** (IDEs) are all-in-one tools for:
  - 🧑‍💻 Writing code
  - 🧪 Testing & debugging
  - ⚙️ Compiling & running

Common Examples:

- **RStudio** (for R)
- **Visual Studio**
- **PyCharm**
- **Eclipse / IntelliJ**

Key Insight:

- IDEs streamline development workflows
- Tool choice affects efficiency, but also **security practices** (e.g., linting, secure coding plugins)



```
library(tidyverse)

ggplot(data=ChickWeight) +
  geom_smooth(mapping=aes(x=Time, y=weight, color=Diet))
```

```
> library(tidyverse)
> ggplot(data=ChickWeight) +
+   geom_smooth(mapping=aes(x=Time, y=weight, color=Diet))
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
>
```

Figure 20.1 RStudio Desktop IDE

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Object-Oriented Programming (OOP)

What is OOP?

- Programming style focused on **objects**, not just logic flow
- Used in modern languages: Java, C++, .NET

OOP Features:

- 🔁 **Encapsulation**: Objects are self-contained (black box)
- 👨‍👩‍👧 **Inheritance**: Subclasses reuse parent behaviors
- 🎭 **Polymorphism**: Objects respond differently to same method
- ➕ **Delegation**: Pass tasks to other capable objects

Design Qualities:

- ✅ High **cohesion** = focused, well-designed class
- 🔄 Low **coupling** = modular, maintainable, secure code

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Common Object-Oriented Programming Terms

- **Message** – Input or communication sent to an object
- **Method** – Internal function executed by the object
- **Behavior** – Observable result/output of a method
- **Class** – Blueprint defining shared attributes & methods of objects
- **Instance** – A concrete, individual object based on a class
- **Inheritance** – Child class inherits attributes/methods from parent class
- **Delegation** – Passing a task from one object to another
- **Polymorphism** – Same message, different behaviors based on the object
- **Cohesion** – Strength of relationship among methods within a class (high = good)
- **Coupling** – Level of dependency between objects (low = good)

TIP If you're interested in learning more about the difference between cohesion and coupling, see http://ducmanhphan.github.io/2019-03-23-Coupling-and-Cohesion-in-OOP.

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Assurance in Secure Development

What is Assurance?

- Confidence that **security controls** work as intended
- Ensures **alignment with policy** across the system life cycle

How It's Achieved:

- 📋 Formal **assurance procedures**
- 🔒 Built into **every stage of the SDLC**
- 🏛️ **Common Criteria** used in government settings

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Avoiding & Mitigating System Failure

Failure is Inevitable — Prepare for It

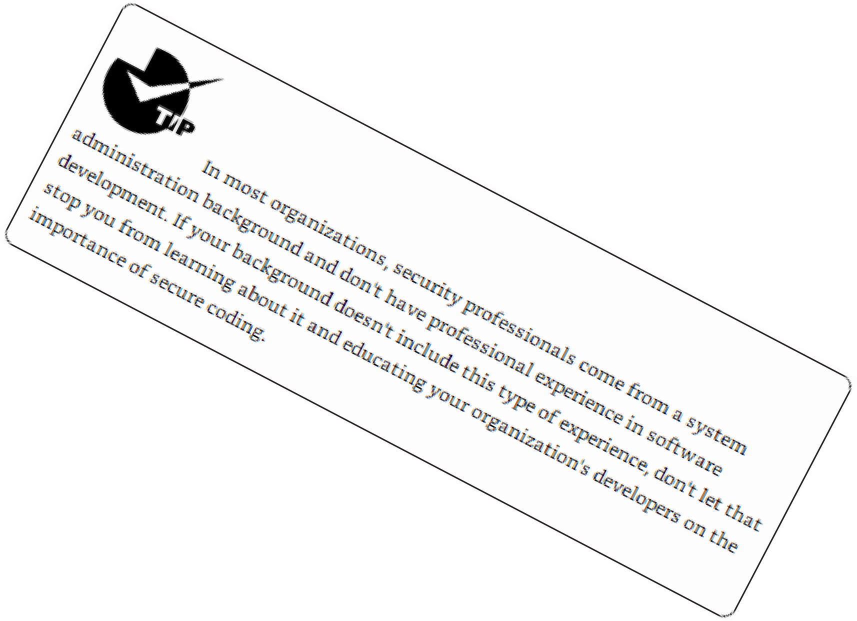
- Plan for system failure through design
- Use **input validation** & **fail-safe controls**

🧪 **Input Validation:**

- ✅ **Limit check** (e.g., valid month = 1–12)
- ❌ Block malicious input: quotes, script tags, etc.
- 🔐 **Escaping input**: Replace risky characters (e.g. < becomes &lt;)
- 🚫 Avoid client-side validation only — **always validate on the server**

Fail Behavior:

- 🔐 **Fail-Secure**: Deny access on failure (more secure)
- 🔒 **Fail-Open**: Permit access on failure (convenience-first, less secure)



TIP

In most organizations, security professionals come from a system administration background and don't have professional experience in software development. If your background doesn't include this type of experience, don't let that stop you from learning about it and educating your organization's developers on the importance of secure coding.

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Authentication, Session Management & Logging

🔐 Authentication & Session Management

- Tie authentication strength to data sensitivity
- Prefer **enterprise auth systems** (e.g., SSO, LDAP)
- Use secure session tokens:
  - Long, random IDs
  - Transmit via **HTTPS only**
  - Expire & require **reauthentication**

❌ Error Handling

- Disable **debug mode** in production
- Never expose server paths, DB schema, or system details

📝 Logging (OWASP Guidance)

Log the following:

- Input validation failures
- Failed authentication & access attempts
- Tampering, invalid tokens, system exceptions
- Admin changes, TLS failures, crypto errors

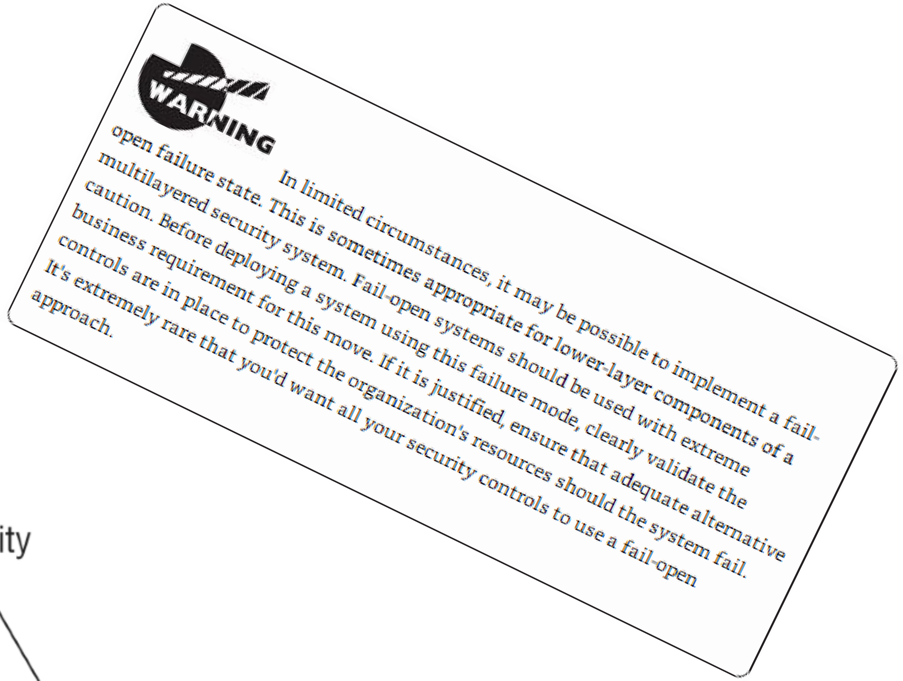
Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

2025 CISSP MENTOR PROGRAM

CHAPTER 20

Software Development Security

Fail-Secure vs. Fail-Open Design



WARNING In limited circumstances, it may be possible to implement a fail-open failure state. This is sometimes appropriate for lower-layer components of a multilayered security system. Fail-open systems should be used with extreme caution. Before deploying a system using this failure mode, clearly validate the business requirement for this move. If it is justified, ensure that adequate alternative controls are in place to protect the organization's resources should the system fail. It's extremely rare that you'd want all your security controls to use a fail-open approach.

🔐 Fail-Secure

- Denies access on failure
- Prioritizes **security** over availability
- Example: Windows BSOD (STOP error)
- May shut down system or lock app
- Often requires **admin reboot**

🔒 Fail-Open

- Allows access on failure
- Prioritizes **availability**
- Risk: bypasses failed security controls
- Rarely acceptable in high-security environments

⚠ Security vs. Usability Trade-Off

- More security = more cost, complexity, and user friction
- IT admins often need to **enable security post-install**

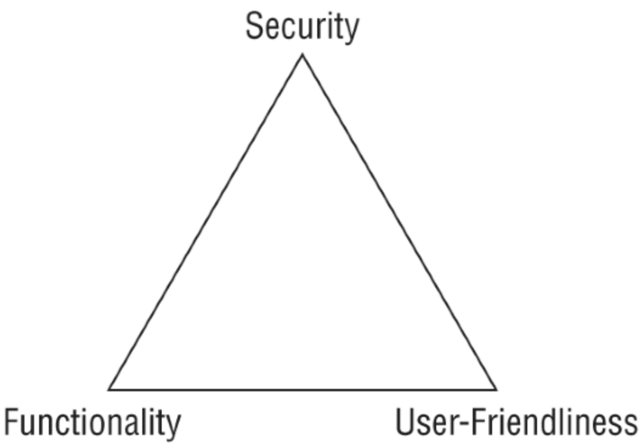
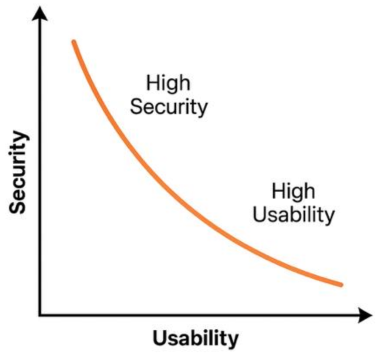


FIGURE 20.2 Security vs. user-friendliness vs. functionality

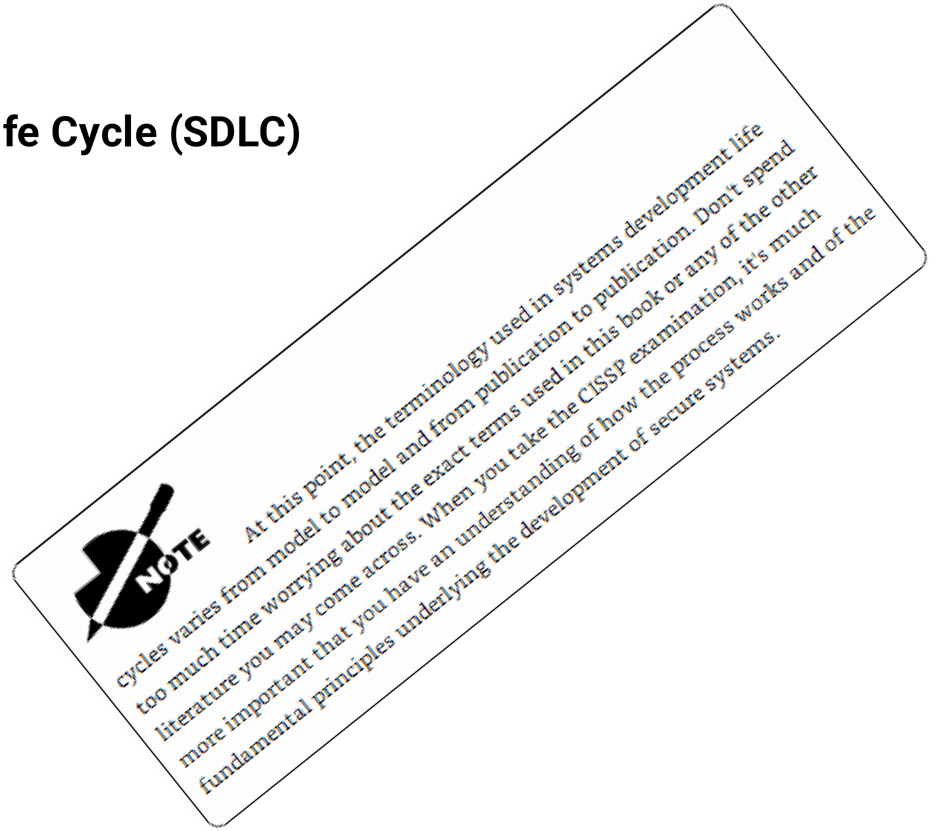


Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Systems Development Life Cycle (SDLC)

🔄 Secure SDLC Principles

- Embed **security throughout** development
- Use **formalized life cycle models** to guide secure practices
- Core development activities:
  - Conceptual definition
  - Functional requirements
  - Control specifications
  - Design review
  - Coding
  - Code review walk-through
  - System testing
  - Maintenance & change management



NOTE

At this point, the terminology used in systems development life cycles varies from model to model and from publication to publication. Don't spend too much time worrying about the exact terms used in this book or any of the other literature you may come across. When you take the CISSP examination, it's much more important that you have an understanding of how the process works and of the fundamental principles underlying the development of secure systems.

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

SDLC Phase 1 – Conceptual Definition

📄 **Conceptual Definition Overview**

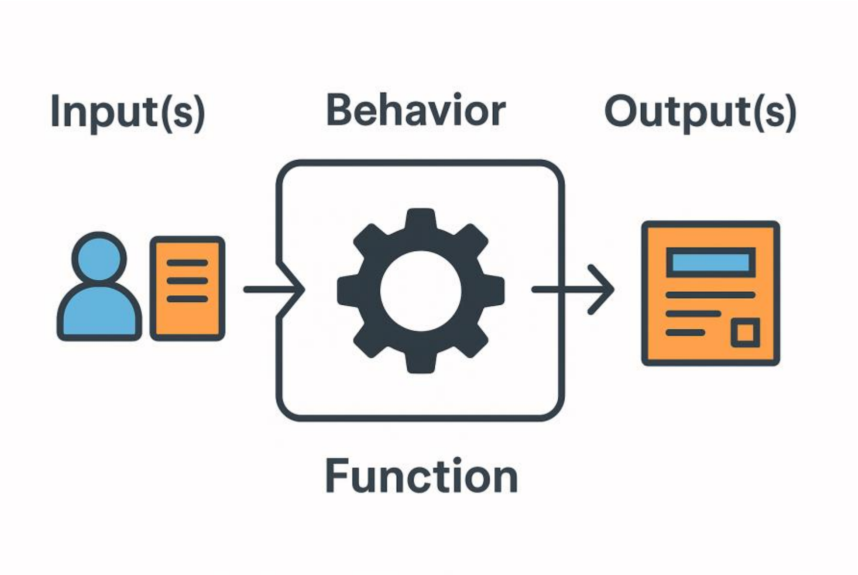
- High-level **statement of system purpose**
- Agreed on by developers, customers, and management
- Establishes:
  - Project scope
  - General system requirements
  - **Data classifications** & security handling needs
- Revisited throughout development to stay on mission

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

SDLC Phase 2 – Functional Requirements Determination

⚙ **Functional Requirements Overview**

- Defines **what the system must do**
- Delivered as a **Functional Requirements Document (FRD)**
- Includes:
  - 📥 **Input(s):** What data goes in
  - ⚙ **Behavior:** What the system does with the data
  - 📤 **Output(s):** What comes out
- Must be agreed upon by all stakeholders
- Becomes a reference throughout **design, development, & testing**



Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

SDLC Phase 3 – Control Specifications Development

🛡️ **Control Specifications Development**

- Begins after functional requirements are set
- Designs **security into the system architecture**
- Key control areas:
    - ✅ Access control enforcement
    - 🔐 Data confidentiality & encryption
    - 📜 Audit trail & accountability
    - 🚨 Detection of illegitimate activity
    - 🔄 Availability & fault tolerance (if critical)
- Should be **proactive**, not retrofitted
- Revisit controls **after design changes**

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

SDLC Phases 4 & 5 – Design Review & Coding

📐 Design Review

- Finalizes **system architecture and module layout**
- Assigns **tasks and timelines** to development teams
- Validates design alignment with:
  - ✔ Functional Requirements
  - 🔒 Control Specifications
- Includes **security stakeholder participation**

💻 Coding

- Developers begin writing software
- Follows approved design documentation
- Applies **secure coding practices**

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

SDLC Phases 6–8: Code Review, Testing & Maintenance

🔍 Code Review Walk-Through

- Developer-led peer reviews at coding milestones
- Identifies logic, design, and **security flaws**
- Enhances code quality before testing

🧪 System Test Review

- Internal testing → User Acceptance Testing (UAT)
- **Functional + Security testing** required
- Use **regression testing** for updates
- Maintain written test plans & results

🔧 Maintenance & Change Management

- Supports operational longevity
- Use **formal change control process**
- Prepare for routine and unplanned updates

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Life Cycle Models in Software Development

🔄 Why Life Cycle Models Matter

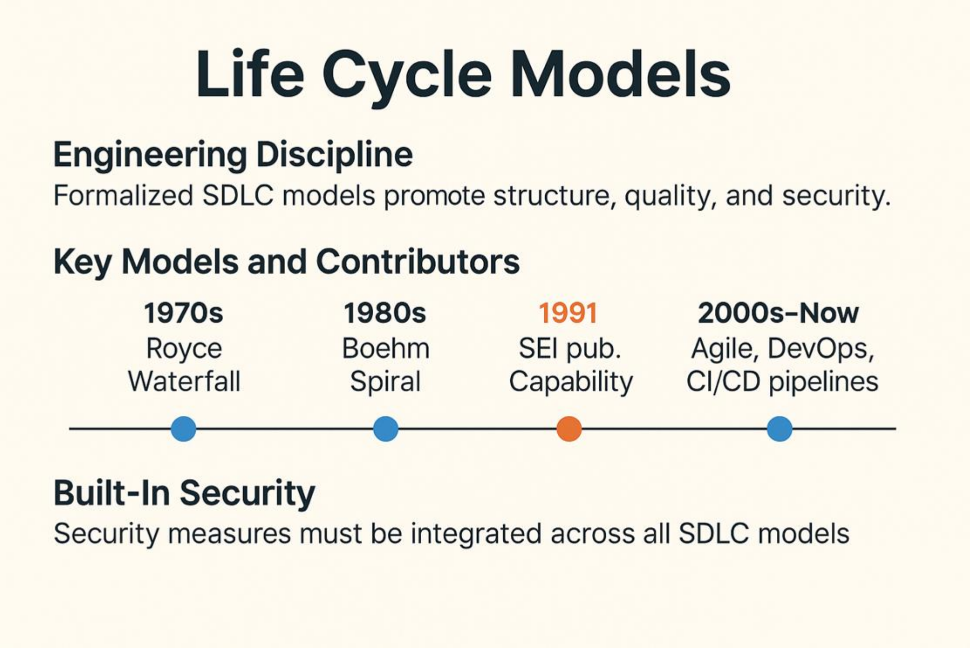
- Bring engineering discipline to software development
- Formalized SDLC models promote **structure**, **quality**, and **security**
- Models help reduce risk of project failure

📚 Historical Milestones

- 1970s–80s: Royce, Boehm propose early SDLC models
- 1991: SEI publishes Capability Maturity Model (CMM)
- Security must be integrated into **every SDLC model**

🛠 Role of Security Professionals

- Ensure chosen model is **appropriate for the org**
- Confirm **management approval** is secured
- Embed **security controls** into all SDLC phases



Life Cycle Models

Engineering Discipline

Formalized SDLC models promote structure, quality, and security.

Key Models and Contributors

Built-In Security

Security measures must be integrated across all SDLC models

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Waterfall Model (Iterative Approach)

- Developed by Winston Royce (1970)
- Sequential SDLC phases: one phase completes before the next begins
- Iterative waterfall includes feedback loops
- Limits rework: can only step back one phase
- Modified model adds validation and verification
- Largely replaced by Spiral & Agile in modern development

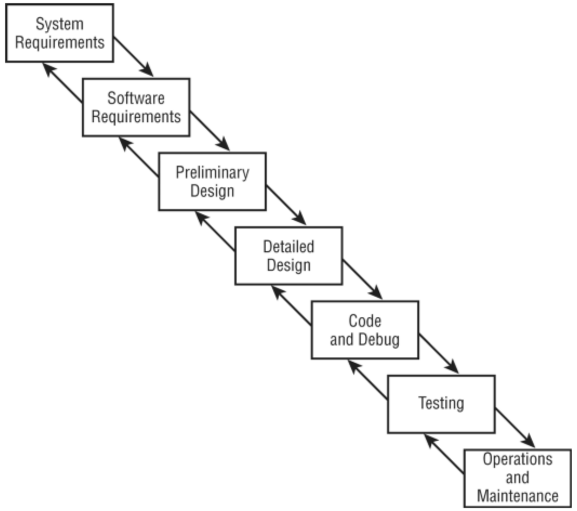
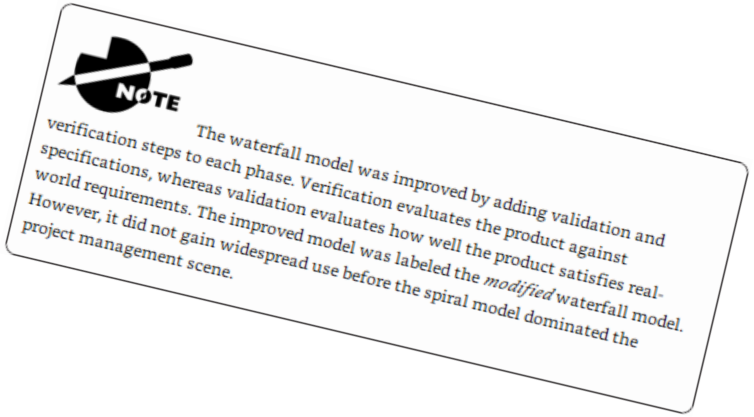


FIGURE 20.3 The iterative life cycle model with feedback loop



NOTE The waterfall model was improved by adding validation and verification steps to each phase. Verification evaluates the product against specifications, whereas validation evaluates how well the product satisfies real-world requirements. The improved model was labeled the *modified* waterfall model. However, it did not gain widespread use before the spiral model dominated the project management scene.

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Spiral Model – Iterative & Risk-Driven Development

- Introduced by Barry Boehm in 1988
- A metamodel: iterations of the Waterfall model
- Each “loop” produces a prototype (P1, P2, P3…)
- Emphasizes continuous refinement through iterations
- Incorporates evolving requirements and risk assessment
- Supports flexible design and enhanced quality control

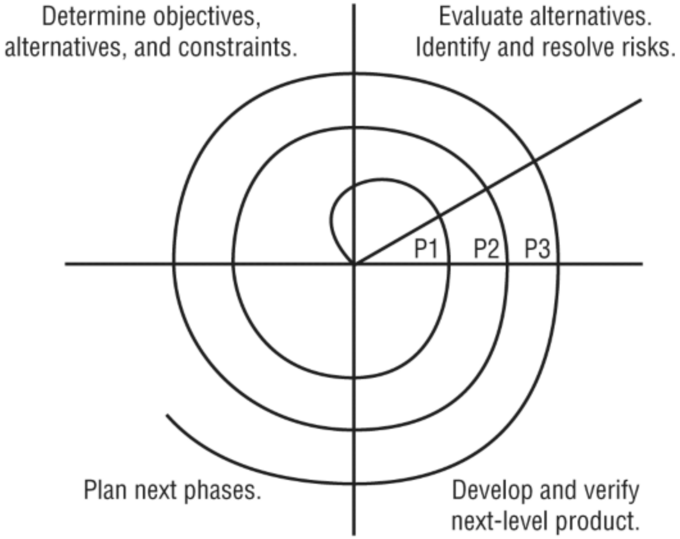

Determine objectives, alternatives, and constraints.
Evaluate alternatives. Identify and resolve risks.
P1
P2
P3
Plan next phases.
Develop and verify next-level product.

FIGURE 20.4 The spiral life cycle mode

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Agile Software Development

- Emerged in the mid-1990s to replace rigid development models
- Agile Manifesto introduced in 2001 by 17 pioneers
- Values delivering working software quickly and iteratively
- Emphasizes:
- Individuals & interactions over processes/tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Agile Manifesto: The 12 Principles

Agile Principles Promote:

- ✅ Early & continuous delivery of valuable software
- 🔁 Embracing changing requirements
- ⏱️ Frequent delivery in short cycles
- 👥 Daily business–dev collaboration
- 💪 Empowered, trusted individuals
- 🗣️ Face-to-face communication
- 📈 Working software = progress
- ♻️ Sustainable development pace
- 🧠 Technical excellence + good design
- ✂️ Simplicity (do only what's needed)
- 🧩 Self-organizing teams
- 🔍 Regular reflection & improvement

The Agile Manifesto also defines 12 principles that underlie the philosophy, available here: http://agilemanifesto.org/principles

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Agile Methods & Scrum + Integrated Product Teams (IPTs)

🌀 **Agile = Philosophy**, not a method
Popular Agile methodologies:

- **Scrum** ✅
- Kanban
- Lean
- RAD
- AUP
- DSDM
- Extreme Programming (XP)

🧩 **Scrum Highlights**:

- Daily **scrum meetings** led by **Scrum Master**
- Work split into **1–4 week sprints**
- Focus on short-term deliverables
- End of each sprint = working product

🤝 **Integrated Product Teams (IPTs)**

- Origin: U.S. Dept. of Defense (1995)
- Cross-functional teams
- Enable **parallel decision-making**
- Ensure all aspects are addressed early

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Scaled Agile Framework (SAFe)

🏢 **SAFe = Agile at Enterprise Scale**
Organized into **4 Configuration Levels**:

1. **Essential SAFe**
   - Core Agile practices (e.g., Scrum)
   - Teams work in **Agile Release Trains (ARTs)**
   - **Program Increments (PIs)** last 8–12 weeks

2. **Large Solution SAFe**
   - For systems needing **multiple ARTs**
   - Adds roles/artifacts for **coordination & alignment**

3. **Portfolio SAFe**
   - **Strategic direction → Actionable work**
   - Driven by **Lean Portfolio Management (LPM)**
   - Focused on delivering **maximum business value**

4. **Full SAFe**
   - Combines all levels
   - Ideal for **large, integrated solutions**

🧩 **Key Concept**: Every task (Story → Feature → Capability → Epic) traces to a business goal.

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

SAFe: 10 Core Principles

🧠 **SAFe is grounded in 10 Agile-inspired principles**:

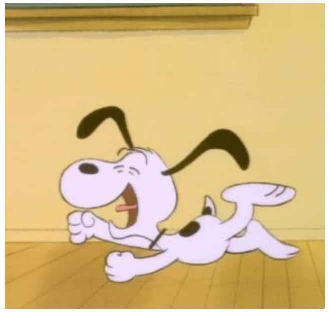
1. 💰 **Take an economic view**
2. 🧠 **Apply systems thinking**
3. 🎲 **Assume variability; preserve options**
4. 🔁 **Build incrementally with fast learning cycles**
5. 📈 **Base milestones on objective evaluation**
6. 🔄 **Make value flow without interruptions**
7. ⏱️ **Apply cadence and synchronize planning**
8. 💡 **Unlock intrinsic motivation of knowledge workers**
9. 🧭 **Decentralize decision-making**
10. 🏗️ **Organize around value**

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Joke break

Why was the computer cold?

It left its Windows open.

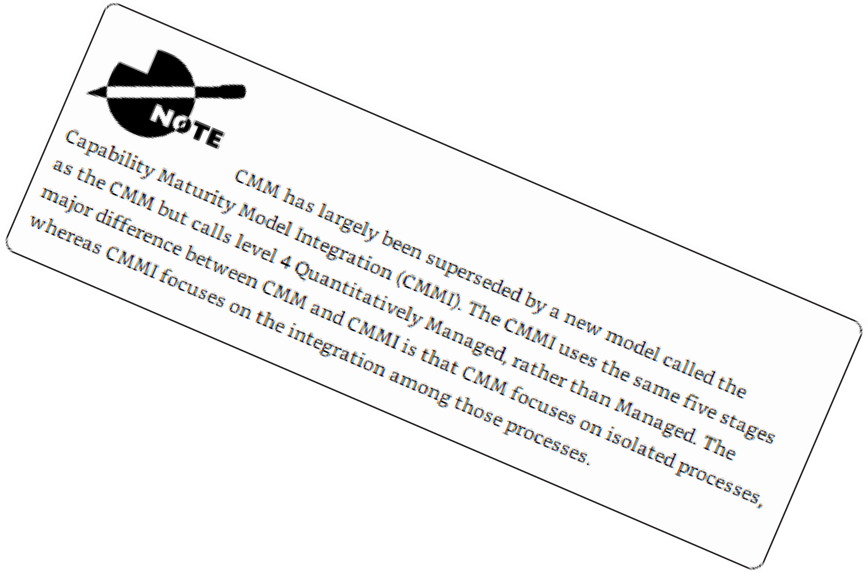


Source: ChatGPT

Capability Maturity Model (CMM)

📌 Overview

- Developed by the **Software Engineering Institute (SEI)** at Carnegie Mellon University.
- Provides a structured framework for **software process improvement**.
- Focuses on **maturing** an organization's development processes from **chaotic** to **disciplined**.
- Does **not explicitly address security**, but secure practices should be integrated by cybersecurity professionals.



NOTE

CMM has largely been superseded by a new model called the Capability Maturity Model Integration (CMMI). The CMMI uses the same five stages as the CMM but calls level 4 Quantitatively Managed, rather than Managed. The major difference between CMM and CMMI is that CMM focuses on isolated processes, whereas CMMI focuses on the integration among those processes.

🔢 Maturity Levels

1. Initial
- Ad hoc, chaotic process.
- Success depends on individual effort.
- Little or no process discipline.

2. Repeatable
- Basic project management practices exist.
- Some reuse of code and consistent results possible.
- Key Areas: Requirements Management, Project Planning, QA, Configuration Mgmt.

3. Defined
- Standardized, documented software development processes.
- All projects follow organization-wide standards.
- Key Areas: Training Program, Integrated Mgmt, Peer Reviews.

4. Managed
- Use of **quantitative metrics** to understand and control processes.
- Key Areas: Quantitative Process Mgmt, Software Quality Mgmt.

5. Optimizing
- Continuous process improvement using feedback loops.
- Key Areas: Defect Prevention, Technology Change Mgmt, Process Change Mgmt.

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Software Assurance Maturity Model (SAMM)

Developed by: OWASP
Purpose: Framework to integrate and assess security within software development and maintenance processes.

SAMM’s 5 Core Business Functions

Function	Description
1. Governance	Strategy, policy, compliance, metrics, education, and guidance.
2. Design	Threat modeling, security requirements, architecture reviews.
3. Implementation	Secure builds, deployment processes, defect management.
4. Verification	Architecture and requirements-driven security testing.
5. Operations	Incident response, environment & operational security controls.

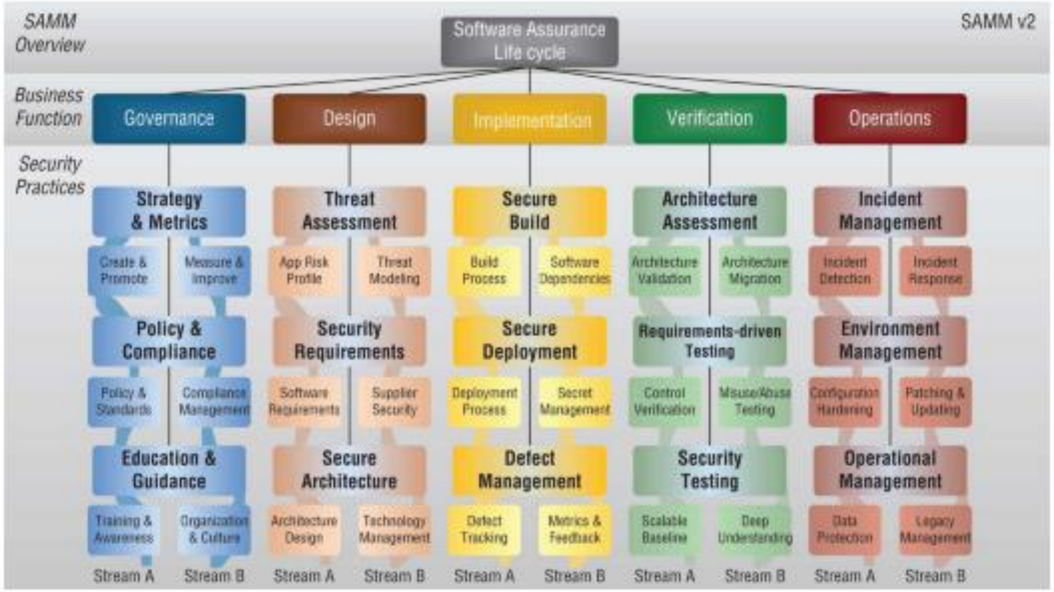


FIGURE 20.5 Software Assurance Maturity Model

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

IDEAL Model (Software Development Process Improvement)

- Developed by **Software Engineering Institute (SEI)**
- Implements many **SW-CMM attributes**
- **Five Phases of IDEAL:**
  - **Initiating:** Outline business reasons; build support; set up infrastructure
  - **Diagnosing:** Analyze current state; recommend general changes
  - **Establishing:** Develop specific action plans
  - **Acting:** Implement solutions; test, refine, deploy
  - **Learning:** Analyze outcomes; adjust actions as needed

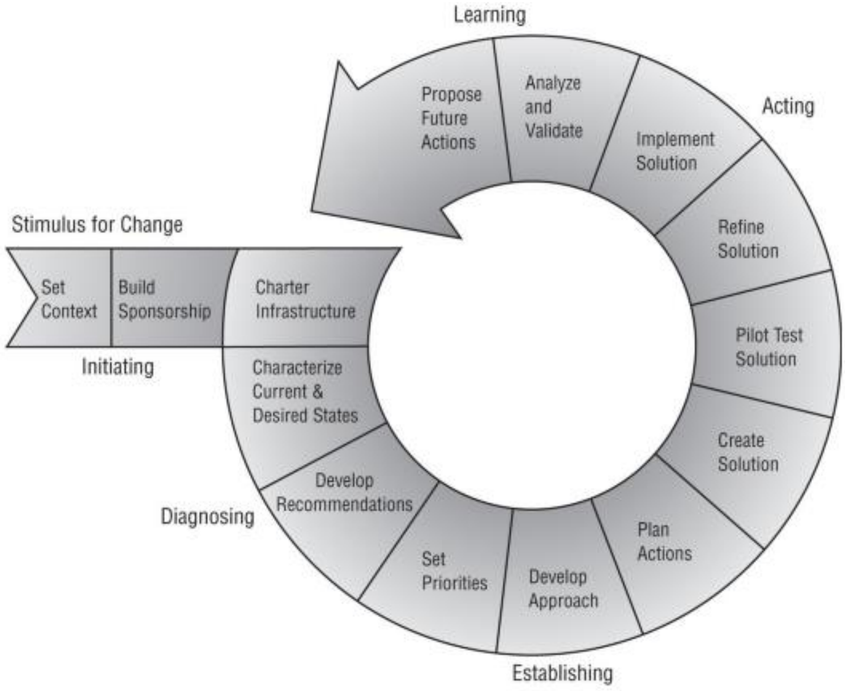
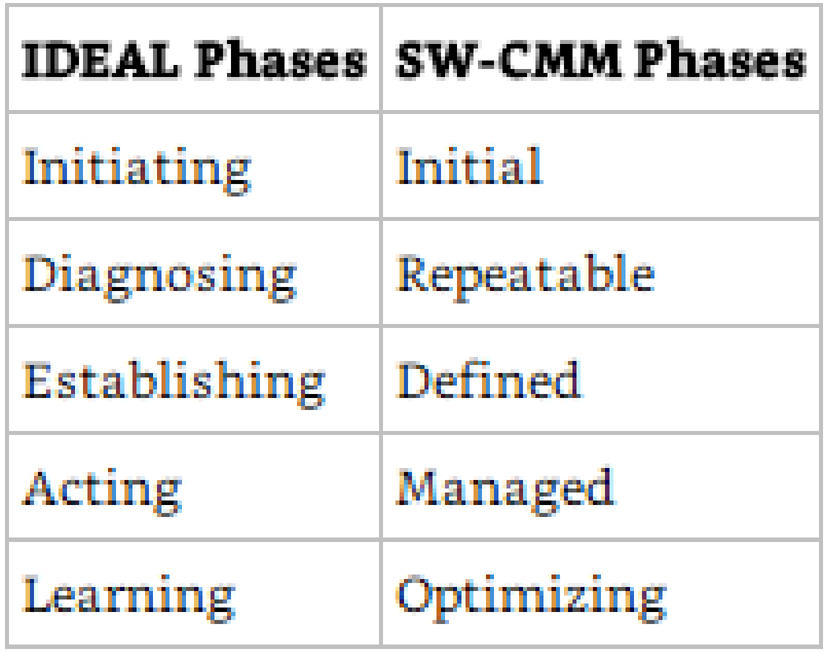


FIGURE 20.6 The IDEAL model

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Memorizing SW-CMM & IDEAL Models

- **Mnemonic:** "I… I, Dr. Ed, am lo(w)."
- Helps recall **initial letters** of:
  - **IDEAL model phases**
  - **SW-CMM levels**
- **Technique:**
  - Write initials in **two columns**
  - Left = **IDEAL model**
  - Right = **SW-CMM levels**
- Enables reconstructing both models in order



IDEAL Phases	SW-CMM Phases
Initiating	Initial
Diagnosing	Repeatable
Establishing	Defined
Acting	Managed
Learning	Optimizing

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Gantt Chart vs. PERT

📁 Gantt Chart

- **Bar chart** showing tasks over time
- Visualizes **schedules and interrelationships**
- Helps **plan, coordinate, track tasks**
- Useful for **shared team resources**

📊 PERT (Program Evaluation Review Technique)

- **Project scheduling tool**
- Estimates **lowest, most likely, highest** sizes
- Calculates **standard deviation for risk assessment**
- Shows **task dependencies clearly**
- Improves **project management & coding efficiency**

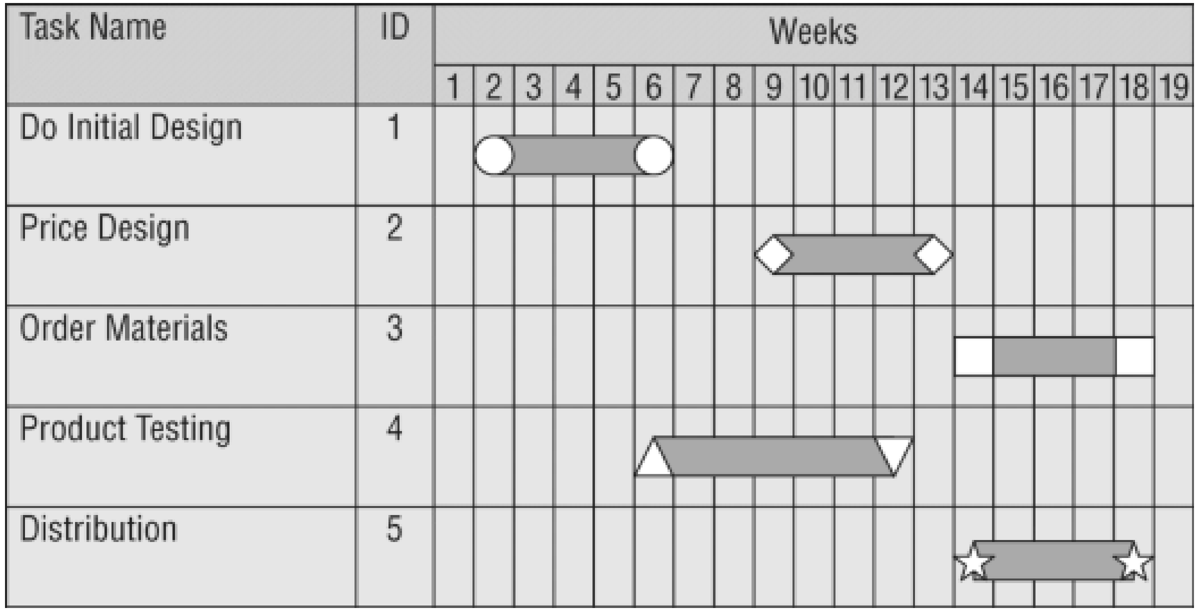


FIGURE 20.7 Gantt chart

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Change & Configuration Management

🔧 **Purpose:**

- Manage **feature additions, bug fixes, modifications** post-release
- Requires **structured procedures** like software development
- Changes logged in **central repository** for:
  - Auditing
  - Investigation
  - Troubleshooting
  - Analysis

🛠️ **Three Components:**

1. **Request Control**
   - Framework for **user modification requests**
   - Managers perform **cost/benefit analysis**
   - Developers **prioritize tasks**
2. **Change Control**
   - Developers **recreate & analyze issues**
   - Framework for **multi-developer solutions & testing**
   - Includes:
     - Quality control adherence
     - Update/deployment tools
     - Proper documentation
     - Restricting code effects to **minimize security risks**
3. **Release Control**
   - **Approval before release**
   - Removes **debugging code/backdoors**
   - Ensures **only approved changes** are released
   - Includes **acceptance testing** for end-user impact

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Change Management as a Security Tool

🔍 **Key Scenario:**

- **Change management** helps detect **unauthorized system changes**
- Organization used **file integrity monitoring tools**
- Problem: Overwhelmed by **normal file modification alerts**

✅ **Solution Implemented:**

- **Tuned monitoring policies**
- Integrated alerts with **change management process**
- Alerts sent to **central monitoring center**
- Admins only alerted if **change is unapproved**
- 🎯 **Result:**
- Reduced **admin workload reviewing alerts**
- Improved **security team efficiency**

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Software Configuration Management (SCM)

🛠 **Purpose:**

- Controls **software versions** organization-wide
- Tracks and manages **software configuration changes**

🔑 **Four Main Components:**

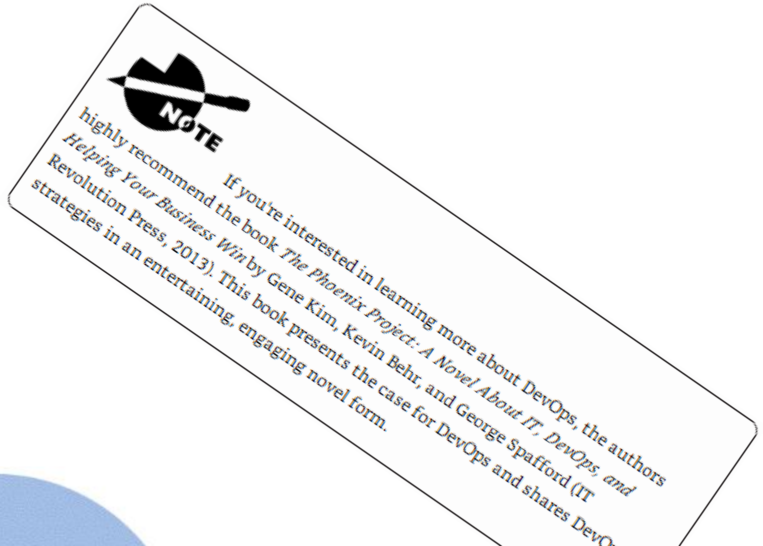
1. **Configuration Identification:**
   - Document configurations of **software products**
2. **Configuration Control:**
   - Changes align with **change control & config policies**
   - Updates only from **authorized distributions**
3. **Configuration Status Accounting:**
   - Formal procedures to **track authorized changes**
4. **Configuration Audit:**
   - Periodic audits ensure production matches **records**
   - Detects **unauthorized changes**

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

2025 CISSP MENTOR PROGRAM

CHAPTER 20

Software Development Security



NOTE

If you're interested in learning more about DevOps, the authors highly recommend the book *The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win* by Gene Kim, Kevin Behr, and George Spafford (IT Revolution Press, 2013). This book presents the case for DevOps and shares DevOps strategies in an entertaining, engaging novel form.

The DevOps (and DevSecOps) Approach

Problem with Traditional IT Functions:

- **Development, QA, Operations** in silos
- Conflicts and **bureaucratic delays**
- Teams **"throw problems over the fence"**

DevOps Solution:

- Merges **Development + Operations**
- Promotes **collaboration & agility**
- Reduces **development, testing, deployment time**

Key Features:

- Aligned with **Agile methodologies**
- Supports **CI/CD (Continuous Integration / Continuous Delivery)**
- Frequent deployments: **daily to hundreds per day**

DevSecOps:

- Integrates **security** into DevOps
- Uses **software-defined security controls**
- Security integrated directly in **CI/CD pipeline**

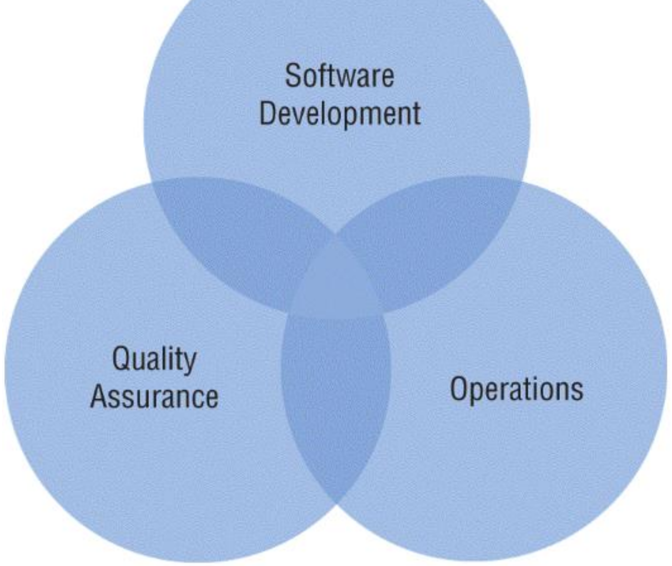


FIGURE 20.8 The DevOps model

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Application Programming Interfaces (APIs)

🌐 **Modern Web Apps:**

- Interact with **multiple external services**
- Examples: credit card processing, social media sharing, shipping providers, referral programs

🔧 **API Purpose:**

- Allows **direct function calls** to services (bypassing web pages)
- Example social media API calls:
  - Post status
  - Follow/Unfollow user
  - Like/Favorite post

⚠ **API Security Considerations:**

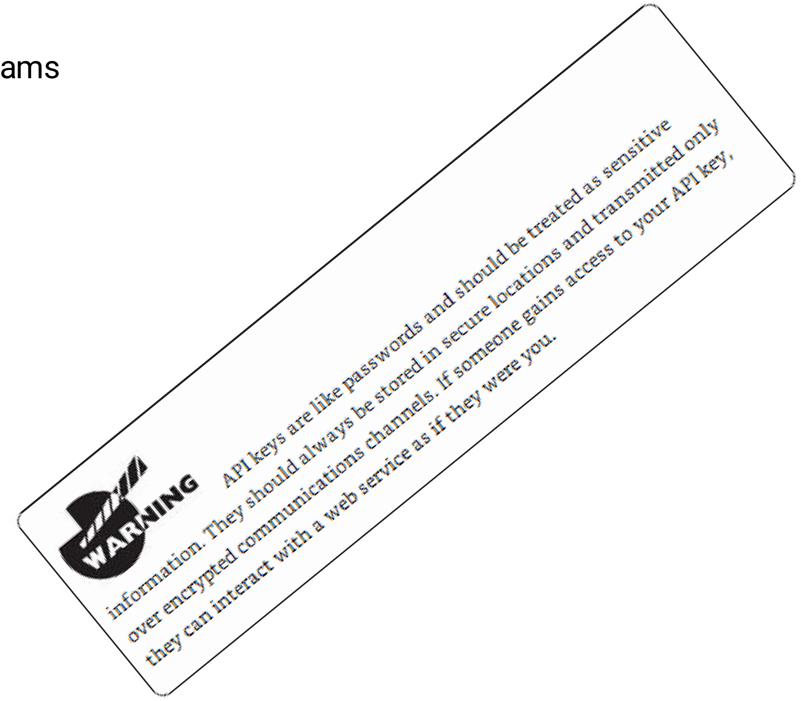
- **Authentication & Authorization:**
  - Public APIs (e.g. weather) vs. restricted APIs (e.g. placing orders)
  - Use **API keys** for secure access
- Validate **credentials and authorization** for each call

🛠 **curl Tool:**

- Open-source tool for **API testing and exploitation**
- Sends requests directly (e.g. POST data in JSON format)

🔍 **Testing APIs:**

- APIs must be tested **thoroughly for security flaws**



WARNING

API keys are like passwords and should be treated as sensitive information. They should always be stored in secure locations and transmitted only over encrypted communications channels. If someone gains access to your API key, they can interact with a web service as if they were you.

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Software Testing

🛠️ Purpose:

- Identify **risks and vulnerabilities** before deployment
- Modify code or use **compensating controls** to mitigate risks

📝 Best Practices:

- Design tests **in parallel** with software modules
- Develop **special test suites** covering all code paths
- Use **reasonableness checks** (e.g. detect out-of-bound results)

🔍 Testing Considerations:

- Test **normal, invalid, out-of-range inputs**
- Avoid live data in early development stages
- Include **use cases (normal activity)** and **misuse cases (attacks)**
- ⚖️ **Separation of Duties:**
- Testers ≠ Programmers to avoid **conflict of interest**
- **Third-party testing** ensures objectivity

🔬 Testing Philosophies:

1. **White-Box Testing:**
    - Examines **internal logic & source code** line by line
2. **Black-Box Testing:**
    - Tests from **user perspective**, no code access
    - Example: **Final acceptance testing**
3. **Gray-Box Testing:**
    - Combines both approaches
    - Tests **inputs/outputs with partial code access**

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

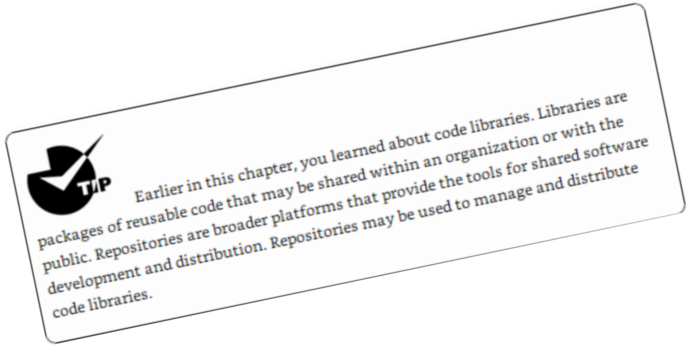
Code Repositories & Security Risks

📁 Purpose of Code Repositories:

- Central **storage for source code**
- Supports **collaborative development** (globally dispersed teams)
- Examples: **GitHub, Bitbucket, SourceForge**

🔧 Functions Provided:

- Version control
- Bug tracking
- Web hosting
- Release management
- Developer communications



TIP Earlier in this chapter, you learned about code libraries. Libraries are packages of reusable code that may be shared within an organization or with the public. Repositories are broader platforms that provide the tools for shared software development and distribution. Repositories may be used to manage and distribute code libraries.

⚠ Security Risks:

- **Access controls:**
  - Public vs. private repositories
  - Read access: Risk of **data leakage**
  - Write access: Risk of **code tampering**

🔑 Sensitive Information Risks:

- Never include **API keys** in public repos
- APIs for IaaS (AWS, Azure, GCP) tie usage to **developer accounts/credit cards**
- Bots scan public repos for exposed keys to exploit immediately

🔒 Best Practices:

- Exclude **passwords, API keys, internal server info, database names**
- Implement **strict access controls** (read/write) based on need

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Service-Level Agreements (SLAs)

📄 Purpose:

- Ensure **agreed-upon service levels** between provider & customer
- Used for **data circuits, applications, systems, databases**, and other critical components

🔧 Common SLA Components:

- **System uptime %** (e.g. 99.9% availability)
- **Maximum consecutive downtime** (seconds/minutes)
- **Peak load & average load** metrics
- **Responsibility for diagnostics**
- **Failover time** (if redundancy exists)

💲 Remedies & Enforcement:

- Financial penalties or contractual remedies for **non-compliance**
- Both parties **monitor performance metrics**
- Example: Circuit down > 15 min → **1 week fee waiver**

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Third-Party Software Acquisition

💻 **Types of Third-Party Software:**

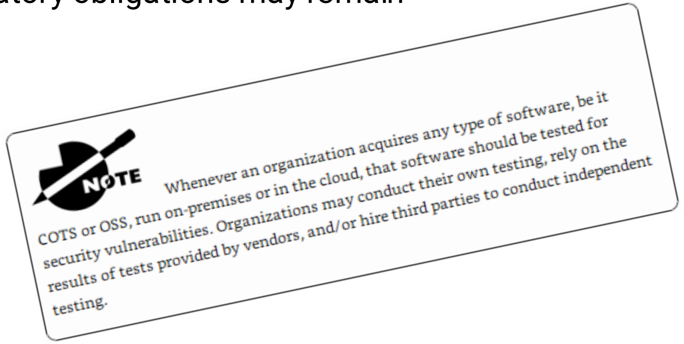
- **COTS (Commercial Off-the-Shelf):**
  - Purchased to run **on-premises or IaaS servers**
- **SaaS (Software-as-a-Service):**
  - Delivered over **internet browsers** (vendor-managed)
- **OSS (Open-Source Software):**
  - Community-created, **freely available**
  - Often included within **COTS packages**

🔧 **Example – Email Services:**

- **On-Premises:**
  - Buy & install **Microsoft Exchange** on servers
- **SaaS:**
  - Outsource to **Google or Microsoft 365**
  - Org manages accounts & settings only

🔒 **Security Responsibilities:**

- **COTS/On-Premises:**
  - Proper **configuration & hardening**
  - Monitor **security bulletins & patches**
- **SaaS:**
  - Vendor maintains **most security controls**
  - Org responsible for:
  - **Vendor security assessments, audits, compliance verification**
  - Legal & regulatory obligations may remain



NOTE Whenever an organization acquires any type of software, be it COTS or OSS, run on-premises or in the cloud, that software should be tested for security vulnerabilities. Organizations may conduct their own testing, rely on the results of tests provided by vendors, and/or hire third parties to conduct independent testing.

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Databases & Data Warehousing Overview

💾 **Organizational Databases:**

- Contain **critical operational data:**
  - Customer contact info
  - Orders, HR records, trade secrets
- Often include **personal user data:**
  - Credit card activity
  - Travel habits, purchases, phone records

🔒 **Security Importance:**

- Protect against **unauthorized access, tampering, destruction**

🗂 **DBMS Architectures:**

- **Hierarchical DBMS**
- **Distributed DBMS**
- **Relational DBMS (RDBMS):** Most common

📊 **Upcoming Topics:**

- **DBMS security concepts:**
  - Polyinstantiation
  - ODBC (Open Database Connectivity)
  - Aggregation & inference
  - Machine learning considerations

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Hierarchical & Distributed Databases

🌳 Hierarchical Databases:

- **Tree structure (one-to-many)**
  - Each node: **0, 1, or many children**
  - Each child: **only one parent**
- **Examples:**
  - Corporate organization charts
  - **NCAA March Madness brackets**
  - **DNS hierarchy**
  - Biological taxonomy: kingdom → species

🌐 Distributed Databases:

- Data stored in **multiple databases**
- Databases are **logically connected**
- Appears as **single entity to users**
- **Many-to-many relationships:**
  - Fields can have **multiple parents and children**

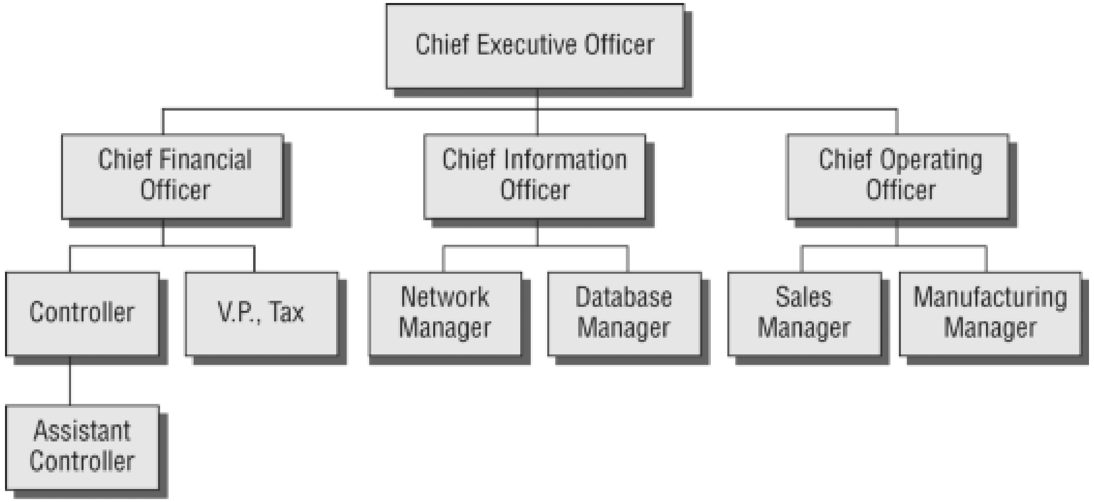
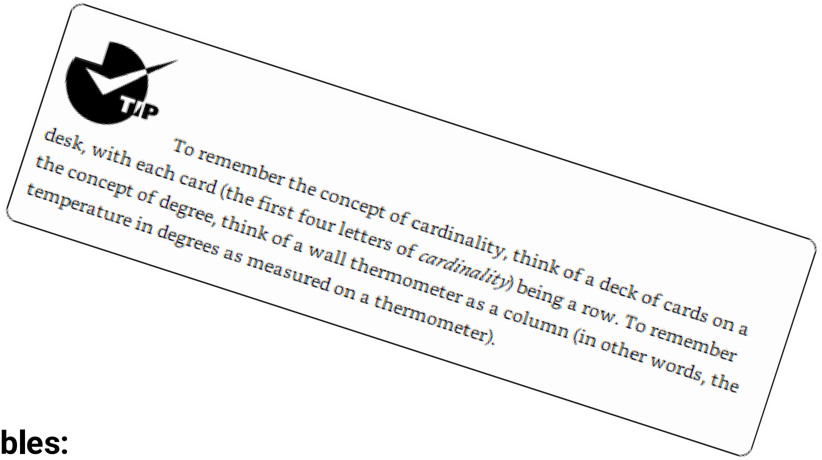


FIGURE 20.9 Hierarchical data model

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart



TIP To remember the concept of cardinality, think of a deck of cards on a desk, with each card (the first four letters of *cardinality*) being a row. To remember the concept of degree, think of a wall thermometer as a column (in other words, the temperature in degrees as measured on a thermometer).

Relational & Object-Oriented Databases

Relational Databases (RDBMS):

- Data stored in **2D tables** (rows & columns)
- Tables = **relations** with **attributes (columns)** and **tuples (rows)**
- **Examples of tables in sales DB:**
  - Customers (client contact info)
  - Sales Reps (employee identity info)
  - Orders (customer orders)

Key Concepts:

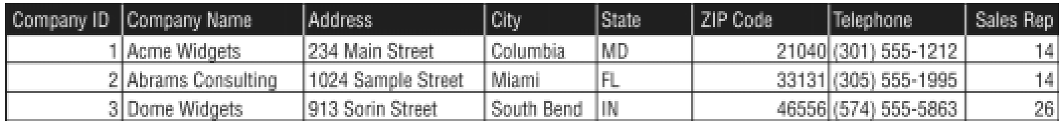
- **Cardinality:** # of rows
  - *Mnemonic:* Cards on desk = rows
- **Degree:** # of columns
  - *Mnemonic:* Degrees on thermometer = columns
- **Domain:** allowable values for an attribute

Table Relationships:

- **Foreign keys link tables:**
  - Customers → Sales Reps (assigned rep)
  - Orders → Customers (who placed order)

Object-Oriented Databases (OODBs):

- Combine **object-oriented programming + databases**
- Benefits:
  - **Code reuse & maintainability**
  - Supports complex apps: multimedia, CAD, video, expert systems



Company ID	Company Name	Address	City	State	ZIP Code	Telephone	Sales Rep
1	Acme Widgets	234 Main Street	Columbia	MD	21040	(301) 555-1212	14
2	Abrams Consulting	1024 Sample Street	Miami	FL	33131	(305) 555-1995	14
3	Dome Widgets	913 Sorin Street	South Bend	IN	46556	(574) 555-5863	26

FIGURE 20.10 Customers table from a relational database

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Database Keys & SQL Overview

Types of Keys:

- **Candidate Key:**
  - Subset of attributes to uniquely identify records
  - Table can have **multiple candidate keys**
- **Primary Key:**
  - **Chosen candidate key** to uniquely identify records
  - **Only one per table** (e.g. Company ID)
- **Alternate Key:**
  - Candidate key **not selected** as primary key
  - E.g. Telephone if unique, but Company ID chosen as primary
- **Foreign Key:**
  - Enforces **referential integrity** between tables
  - References primary key in another table

SQL (Structured Query Language):

- **Standard DB language** for storage, retrieval, modification, admin
- Variants: Microsoft **Transact-SQL**, Oracle **PL/SQL**
- **Primary security feature:** Fine-grained authorization
  - Set permissions by **table, row, column, or cell**

Database Normalization:

- Process of organizing tables to reduce **redundancy & misplaced data**
- Uses **normal forms (1NF, 2NF, 3NF)**
  - **Cumulative compliance:** Must meet lower forms first
- **Goal:** Efficient, well-organized databases

SQL Components:

- **DDL (Data Definition Language):** Defines/modifies **schema & structure**
- **DML (Data Manipulation Language):** Interacts with **data contents**

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Database Transactions & ACID Model

🔄 **Database Transactions:**

- Discrete set of **SQL instructions**
- Must **succeed or fail as a group** (no partial success)
- Example: **Bank transfer**
  - Add $250 to account A
  - Subtract $250 from account B

💥 **Without transactions:**

- Failure between operations = **data inconsistency**
- E.g. Money appears/disappears

🔧 **Transaction Outcomes:**

- **Commit:** Successfully completed; permanent
- **Rollback:** Aborted; database reverts to prior state

⚖️ **ACID Model Attributes:**

- **Atomicity:** All-or-nothing processing
- **Consistency:** Must start/end in **valid state**
- **Isolation:** Transactions operate **independently**
- **Durability:** Committed transactions are **preserved permanently**

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Security for Multilevel Databases

🔒 Multilevel Security (MLS):

- Uses **data classification schemes** to enforce access control
- Databases may hold info at **multiple classification levels**
- Must verify **user labels & access rights**

⚠️ Database Contamination:

- Mixing data of **different classifications**
- Creates **security challenges**

🛠️ Solutions:

- **Keep data separated** by classification
- Use **trusted front ends** for MLS with legacy/insecure DBMS

👁️ Using Views for MLS:

- **Views = SQL statements** acting as virtual tables
- Can:
  - Combine data from multiple tables
  - **Restrict user access** to specific attributes/records
- **Benefits:**
  - Saves storage space
  - Provides **controlled access**
- **Downside:**
  - Slower data retrieval due to **real-time calculations**

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Concurrency Control in Databases

🔄 **Concurrency (Edit Control):**

- Ensures **data integrity and availability**
- Applies to **single-level and multilevel databases**

⚠ **Concurrency Issues:**

1. **Lost Updates:**
    - Two processes update same data **without awareness of each other**
    - Example:
        - Inventory = 10
        - Two stations each add 1 simultaneously
        - Both update to **11 instead of 12**
2. **Dirty Reads:**
    - Process reads **uncommitted transaction data**
    - Example:
        - Receiving station updates inventory but crashes
        - Another process reads **incomplete/incorrect data**

🔧 **Concurrency Mechanisms:**

- **Locks:**
    - Allow **exclusive data changes** by one user/process
    - Other users denied view/change until lock released
- **Unlocks:**
    - Restores access after **updates complete**
- **Auditing:**
    - Track document/field changes for **detection control**

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Aggregation in Databases

🔗 **What is Aggregation?**

- Uses SQL functions to **combine records**
- Generates **useful summaries or insights**

⚠ **Aggregation Attacks:**

- Combine **low-level or low-value data** into **high-value or classified information**
- Example:
  - **Records clerk** updates individual transfer records (unclassified)
  - Uses aggregation to **count total troops per base** (classified info)

🔒 **Security Implications:**

- Aggregation functions can **reveal sensitive data indirectly**
- Must enforce:
  - **Strict access control** to aggregation functions
  - **Need-to-know & least privilege principles**
  - **Defense-in-depth** to prevent unauthorized data inference

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Inference Attacks in Databases

🧠 **What is Inference?**

- Combines **nonsensitive information** to deduce **classified info**
- Uses **human deductive reasoning**, unlike aggregation’s mathematical approach

🔍 **Example:**

- **Accounting clerk** retrieves total salary data (allowed)
- Knows **hire/termination dates**
- Compares totals before & after **sole employee’s hire date**
- Deduces **individual salary** (sensitive info not directly accessible)

🔒 **Defense Against Inference Attacks:**

- **Strict user permissions** & access control
- **Intentional data blurring** (e.g. rounding salaries to nearest million)
- **Database partitioning** (covered next) to limit data cross-referencing

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Other DBMS Security Mechanisms

🔧 **Semantic Integrity:**

- Prevents **violations of structural rules**
- Ensures data is:
  - Within **valid domain ranges**
  - **Logical and unique** as per constraints

🕒 **Time & Date Stamps:**

- Used in **distributed databases**
- Changes applied in **chronological order** for integrity

🔒 **Granular Object Control:**

- **Content-dependent access control:** Based on **data contents**, increases processing overhead
- **Cell suppression:** Hides or restricts individual **fields/cells**

🌐 **Context-dependent Access Control:**

- Decisions based on **overall context**
- Evaluates if data is benign/malign **within broader activity**

📁 **Database Partitioning:**

- Splits DB into **parts with distinct security levels**
- Mitigates **aggregation & inference attacks**

📝 **Polyinstantiation:**

- Multiple rows with same primary key but **different classification levels**
- Example:
  - Ship location table with **secret vs top-secret entries** for same ship
  - Prevents users from noticing missing data at lower clearance

🎭 **Noise & Perturbation:**

- Inserts **false/misleading data** to thwart attacks
- Must avoid **impacting operations**

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Open Database Connectivity (ODBC)

🔗 **What is ODBC?**

- Allows apps to **communicate with different databases**
- Acts as a **proxy between applications & database drivers**

💻 **Benefits:**

- **Programmer freedom:**
  - Apps aren’t tied to specific DB systems
- Simplifies **cross-DB development & integration**

📊 **How it Works:**

- Applications → **ODBC layer** → Database drivers → **Backend DB**

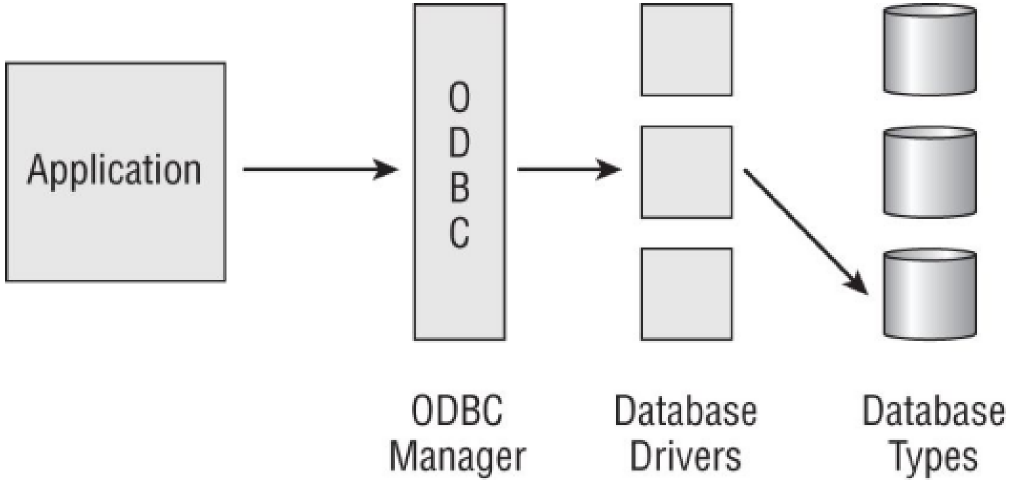


FIGURE 20.11 ODBC as the interface between applications and a backend database system

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

NoSQL Databases Overview

📈 **Why NoSQL?**

- Alternative to **relational databases (RDBMS)**
- Used when:
  - **Speed is critical**
  - Data doesn't fit **tabular format**

📦 **Common NoSQL Types:**

1. **Key-Value Stores:**
   - Simplest form; stores **key–value pairs**
   - **High-speed apps, large datasets**
   - Minimal structure overhead
2. **Graph Databases:**
   - Data stored as **nodes (objects) & edges (relationships)**
   - Ideal for **social networks, geographic data, networks**
3. **Document Stores:**
   - Keys map to **complex documents (e.g. XML, JSON)**
   - More versatile than basic key-value stores

🔒 **Security Considerations:**

- **NoSQL security models differ** from relational DBs
- Security professionals must:
  - Understand **solution-specific security features**
  - Collaborate with DB teams on **controls & design**

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Storage Threats in DBMS Security

🔒 **DBMS Limitations:**

- Protects **“front-door” access only** (queries, application interactions)
- Data remains vulnerable at **storage level** (memory, physical media)

🚨 **Threat 1: Illegitimate Access**

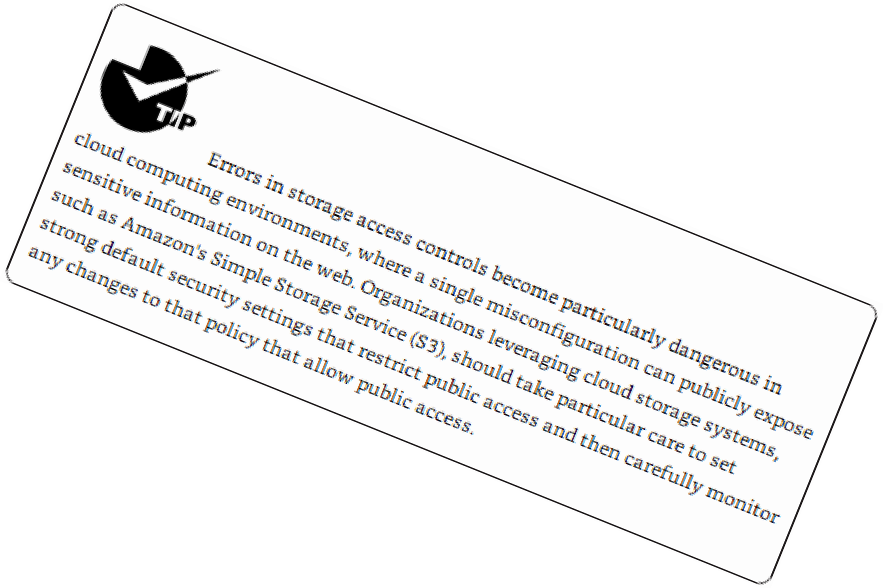
- **Inadequate file system controls** → unauthorized browsing
- Attackers may **bypass OS controls** to access storage directly
- **Countermeasure:**
  - **Encrypted file systems** accessible only via primary OS
  - Enforce **multilevel security controls** to prevent cross-classification leaks

⚠️ **Threat 2: Covert Channel Attacks**

- Use **shared storage/media to transmit data covertly** between classifications
- **Examples:**
  - Writing sensitive data to shared memory
  - Manipulating **disk free space or file sizes** to encode info

🔧 **Mitigation Strategies:**

- Implement **encrypted storage**
- Conduct **covert channel analysis** (refer to Chapter 8)
- Enforce **strict access and classification controls**



TIP

Errors in storage access controls become particularly dangerous in cloud computing environments, where a single misconfiguration can publicly expose sensitive information on the web. Organizations leveraging cloud storage systems, such as Amazon's Simple Storage Service (S3), should take particular care to set strong default security settings that restrict public access and then carefully monitor any changes to that policy that allow public access.

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Understanding Knowledge-Based Systems

🧠 **Purpose of Knowledge-Based Systems:**

- Automate **routine, time-consuming tasks**
- Reduce **human workload** on repetitive computations

🤖 **Advancements in AI:**

- Systems now simulate **human reasoning** to some extent
- Incorporate **artificial intelligence capabilities**

🔍 **Three Main Types:**

- **Expert Systems**
- **Machine Learning Systems**
- **Neural Networks**

💻 **Applications:**

- Support **computer security decision-making**
- Enhance **automation and analysis capabilities**

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Expert Systems

🧠 **Purpose:**

- Embody **expert knowledge** to make consistent decisions
- Often outperform humans on **routine decisions**

🔧 **Core Components:**

1. **Knowledge Base:**
   - Codifies knowledge as **“if/then” rules**
   - Example rules:
     - If hurricane Category ≥4 → Flood waters 20 ft
     - If winds >120 mph → Wood-frame structures destroyed
     - If late season → Hurricanes strengthen near coast

2. **Inference Engine:**
   - Uses **logical reasoning & fuzzy logic**
   - Analyzes knowledge base + user input to **make decisions**

💡 **Advantages:**

- Decisions are **unbiased by emotions**
- Effective for:
  - **Emergency analysis** (e.g. hurricanes)
  - **Stock trading**
  - **Credit approvals** (objective decisions)

⚠️ **Limitations:**

- Only as strong as its **knowledge base & algorithms**

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Machine Learning

🤖 **What is Machine Learning?**

- Uses **analytic capabilities** to develop knowledge from data
- **Learns directly from datasets** without explicit human programming
- Builds and updates **models of activity**

🔍 **Two Major Categories:**

1. **Supervised Learning:**
   - Uses **labeled data** (includes correct answers)
   - Algorithm learns by example
   - Example:
     - Train model to detect **malicious logins** using dataset labeled as malicious vs. benign
2. **Unsupervised Learning:**
   - Uses **unlabeled data**
   - Finds **patterns or groups** without predefined categories
   - Example:
     - Identify **clusters of similar logins**, analyst reviews groups for anomalies

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Neural Networks

🧠 What are Neural Networks?

- **Chains of computational units** simulating human brain processes
- Subset of **machine learning (deep learning)**
- Unlike expert systems with rules, neural networks use **weighted computational decisions**

💡 Key Benefits:

- **Linearity** (process complex data linearly)
- **Input-output mapping** (relate inputs to outputs accurately)
- **Adaptivity** (learn and adjust based on training)

🔧 How They Learn:

- Require **training with known inputs & outputs**
- Adjust **weights via Delta rule (backpropagation)**
- Enables learning from **experience & pattern recognition**

📈 Applications:

- **Voice recognition**
- **Face recognition**
- **Weather prediction**
- Exploring **thinking & consciousness models**

🔒 Security Relevance:

- Rapid, consistent analysis of **massive logs & audit trails**
- Detect **anomalies and threats** efficiently

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Joke break

Why did the expert system get fired?

Because it kept making “if/then” excuses

Source: ChatGPT

Chapter 20 Summary

💾 Data as a Critical Resource:

- Most valuable asset for many organizations
- Must safeguard **data, systems, and applications**

🛡️ Key Security Requirements:

- Protect against:
  - **Malicious code**
  - **Database vulnerabilities**
  - **System/app development flaws**

🔐 Access Controls & Audit Trails:

- Essential for **database security and accountability**

📚 Continuous Learning:

- Database security is **rapidly evolving**
- **Collaborate with DBAs** and invest in deeper understanding

🔧 Development Security Controls:

- Implement during **design, development, deployment, maintenance**
- Examples:
  - **Process isolation**
  - **Hardware segmentation**
  - **Abstraction**
  - **Service-Level Agreements (SLAs)**

🚀 Security Integration:

- Introduce security **early in planning**
- **Monitor continuously** throughout lifecycle

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

CHAPTER 20

Software Development Security

Chapter 20 Study Essentials

🗃 **RDBMS Architecture:**

- **Tables (relations):** Store related data
- **Rows (records/tuples):** Individual entries
- **Columns (fields/attributes):** Data categories
- **Keys:** Define relationships
  - **Primary, candidate, alternate, foreign keys**

⚠ **Database Security Threats:**

- **Aggregation:** Combining low-value data → high-value info
- **Inference:** Deduce sensitive info from nonsensitive data

🤖 **Knowledge-Based Systems:**

- **Expert Systems:**
  - **Knowledge base:** “If/then” rules
  - **Inference engine:** Draws conclusions
- **Machine Learning:**
  - Learns patterns from data **algorithmically**
- **Neural Networks:**
  - Simulate brain processes with **layered calculations**
  - Require **extensive training**

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

Chapter 20 Study Essentials (cont.)

💻 **Systems Development Models:**

- **Waterfall:** Sequential phases, can step back **one phase**
- **Spiral:** Iterative waterfall producing **prototypes**
- **Agile:** Rapid, iterative, **customer-focused**
- 🔧 **Scrum (Agile Approach):**
- **Daily scrum meetings**
- **Short sprints** deliver finished products
- **Integrated Product Teams (IPTs):** Early DoD approach

📈 **Software Development Maturity Models:**

- **Purpose:** Improve software quality & processes
- **Examples:**
    - **SW-CMM**
    - **IDEAL**
    - **SAMM**

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

CHAPTER 20

Software Development Security

Chapter 20 Study Essentials (cont.)

🔧 **Change & Configuration Management:**

- **Change Management Components:**
  - **Request Control:** Users request changes, management prioritizes
  - **Change Control:** Developers analyze, implement, and test changes securely
  - **Release Control:** Final approval, ensures only authorized, clean code is released
- **Configuration Management:**
  - Controls **software versions organization-wide**
  - Includes **auditing and logging** for accountability & risk mitigation

🧪 **Software Testing:**

- Should be **integrated in development design**
- Acts as a **management tool** to improve design, development, production

💛 **DevOps & DevSecOps:**

- **DevOps:**
  - Integrates **development + operations**
  - Emphasizes **automation & collaboration**
- **DevSecOps:**
  - Adds **security operations** into DevOps pipeline
  - Uses **CI/CD (Continuous Integration/Delivery)** to automate deployment securely

💻 **Coding Tools:**

- **Programming Languages:** Compiled or interpreted
- **Development Toolsets & IDEs:** Simplify coding process
- **Software Libraries:** Provide **shared, reusable code**
- **Code Repositories:** Manage **code versions & collaboration**

🛒 **Acquired Software Impacts:**

- Use of **COTS (Commercial Off-the-Shelf)** and **OSS (Open-Source Software)**
- **Increases attack surface** → Requires security **review and testing**

Source: CISSP Official Study Guide, 10th Ed., Chapple & Stewart

YAY! YOU MADE IT!

That was A LOT of information. Now what?

- Keep up in the book. We just went through Chapter 20.
- Be sure to review and focus on the “Study Essentials” sections for each chapter.
- If you’re ambitious, do the “Written Lab” section for each chapter too.
- When you’re ready, take a stab at the “Review Questions” for each chapter.
- If you haven’t already, feel free to check out the CISSP cheat sheet on Discord.
- Jot down your questions, post them in Discord, and/or ask them in the Live Mentor Session (July 9th)

That’s it for now, **CONGRATS** for making it through this. ☺

See you Wednesday night.