Certified Information Systems Security Professional (CISSP) Certification Training Course

Domain 08: Software Development Security

Learning Objectives

By the end of this lesson, you will be able to:

- Discuss different software development methods
- Define object-oriented programming and explain the terms associated with it
- Compare source code analysis tools
- Explain software security and assurance
- Discuss the effectiveness of software security
- Implement web application environment security concepts

Introduction to Software Development Security

Importance of Software Development Security

Nutri Worldwide Inc. has developed a vendor management system for their vendor management process. One of the key features of the new software is the centralized bidding process for contracts.

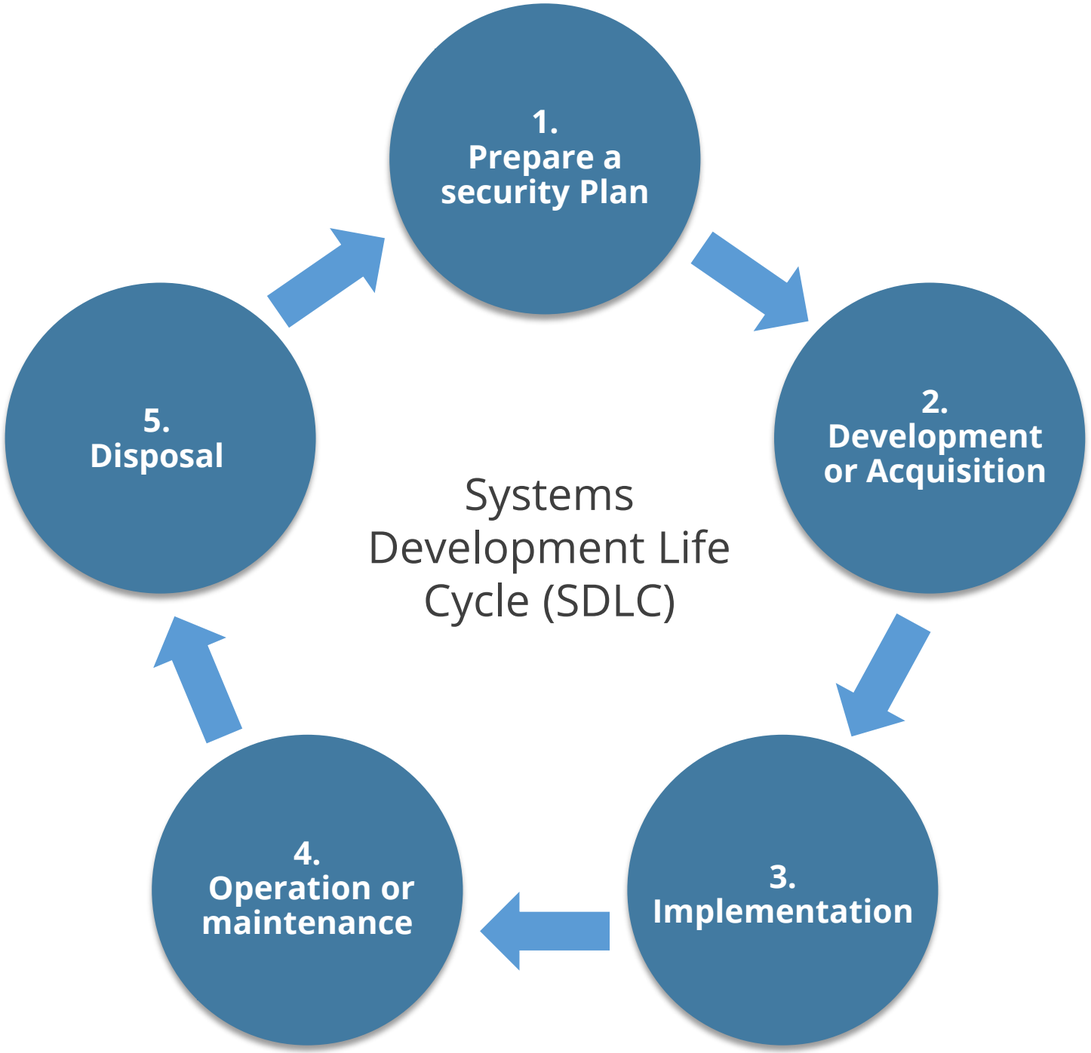
It was noticed that, regardless of the number of bidders, one vendor always managed to get the contract to supply bottles and cans for one of the processing units.

After a thorough investigation it was found that this vendor managed to access the bidding data. During the programming and testing phase of the development of the software, secure programming practices were not implemented.

Understand and Integrate Security in the Software Development Life Cycle (SDLC)

Systems Development Life Cycle

The systems development life cycle is a system development model used throughout the IT industry.

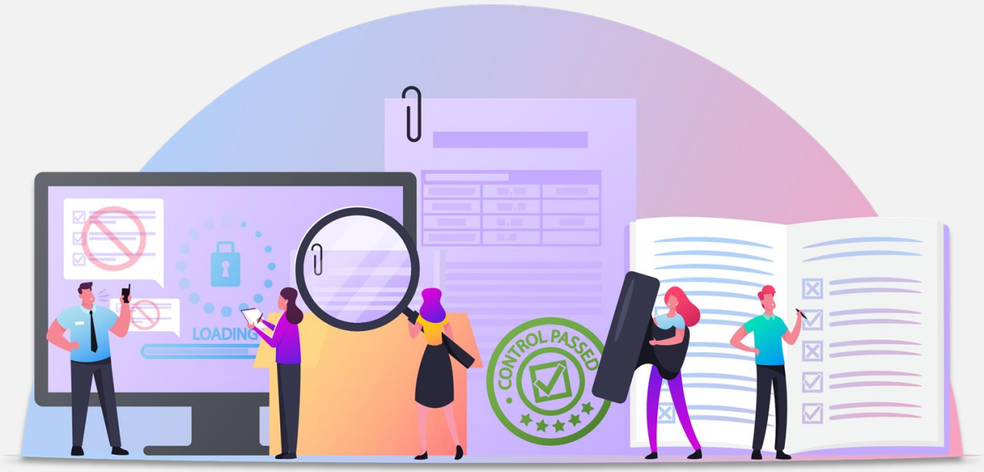


Systems Development Life Cycle

Security controls should be included in every phase of the systems development life cycle.

The security control process includes:

1. Conducting a sensitivity assessment
2. Determining security requirements
3. Creating security specifications
4. Installing or switching on controls
5. Security testing
6. Certification and accreditation
7. Managing change and configuration



Software Development Models

The software development methodology is a framework that is used to structure, plan, and control the process of software development.

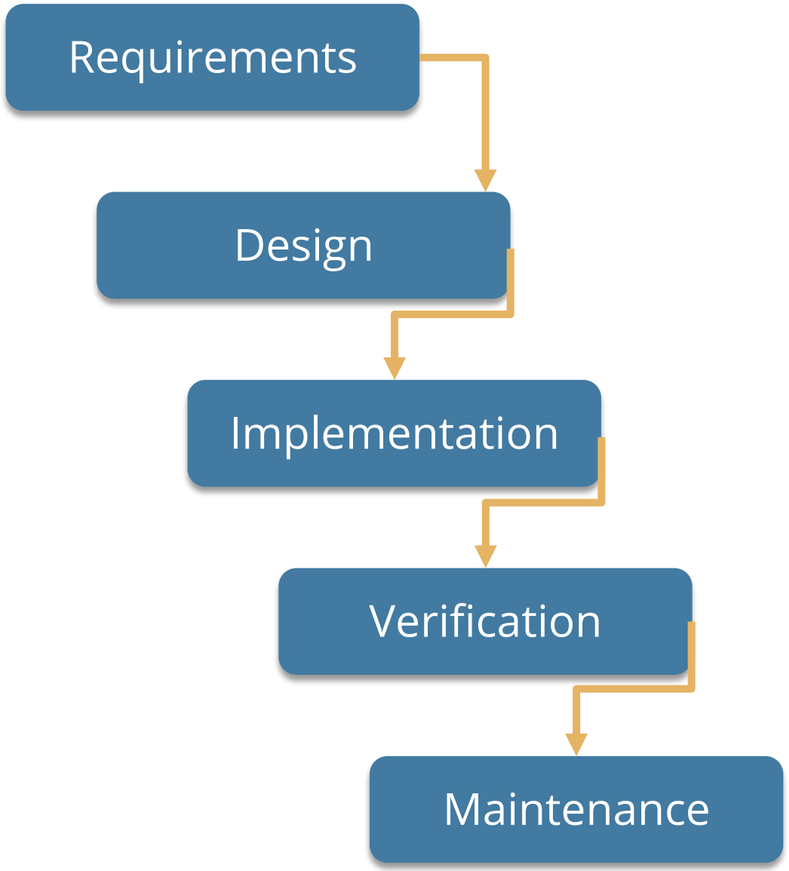
To manage a project efficiently, the manager or development team must choose the software development methodology that will work best for the project at hand.

All methodologies have different strengths and weaknesses and exist for different reasons.

Software Development Models

Waterfall model

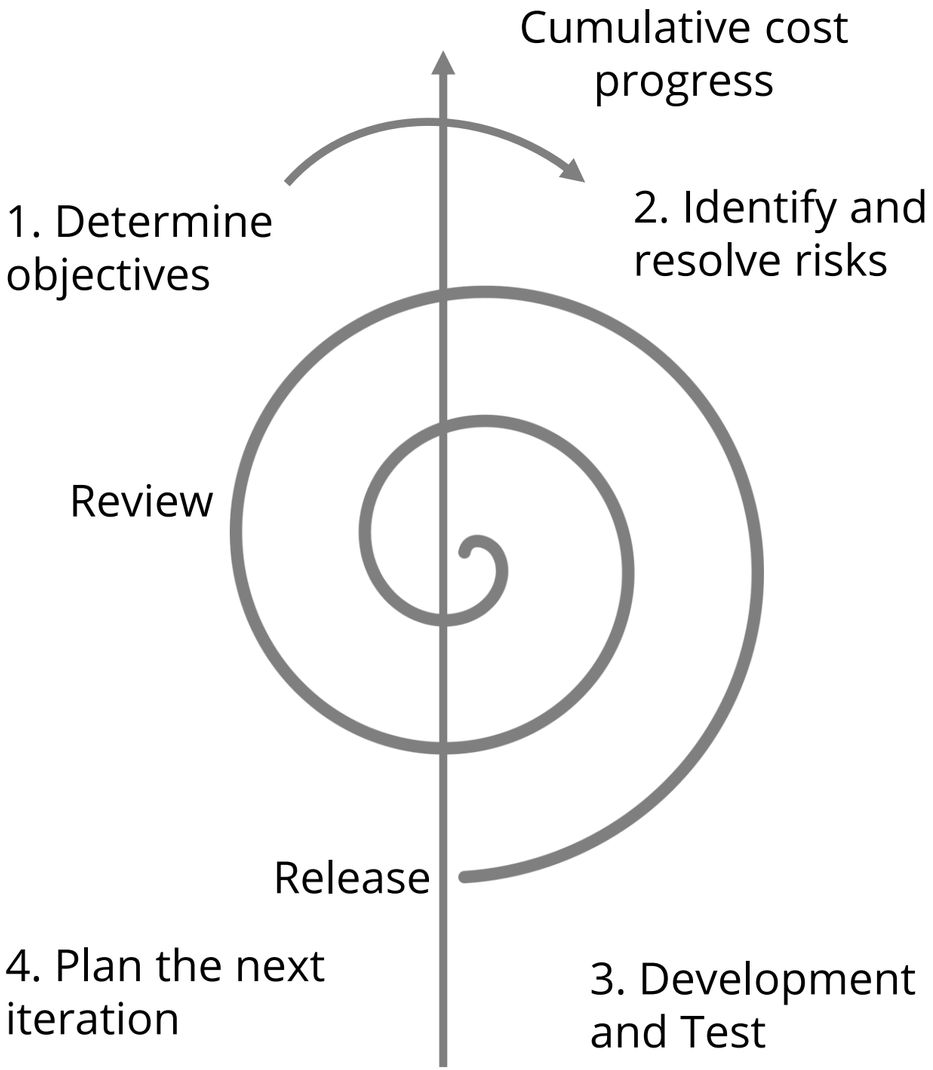
- The waterfall model is a linear application development model that uses rigid phases.
- In 1976, Barry Boehm reinterpreted the waterfall model.
- The modified waterfall model allows one to return to a previous phase for verification or validation.



Software Development Models

Spiral model

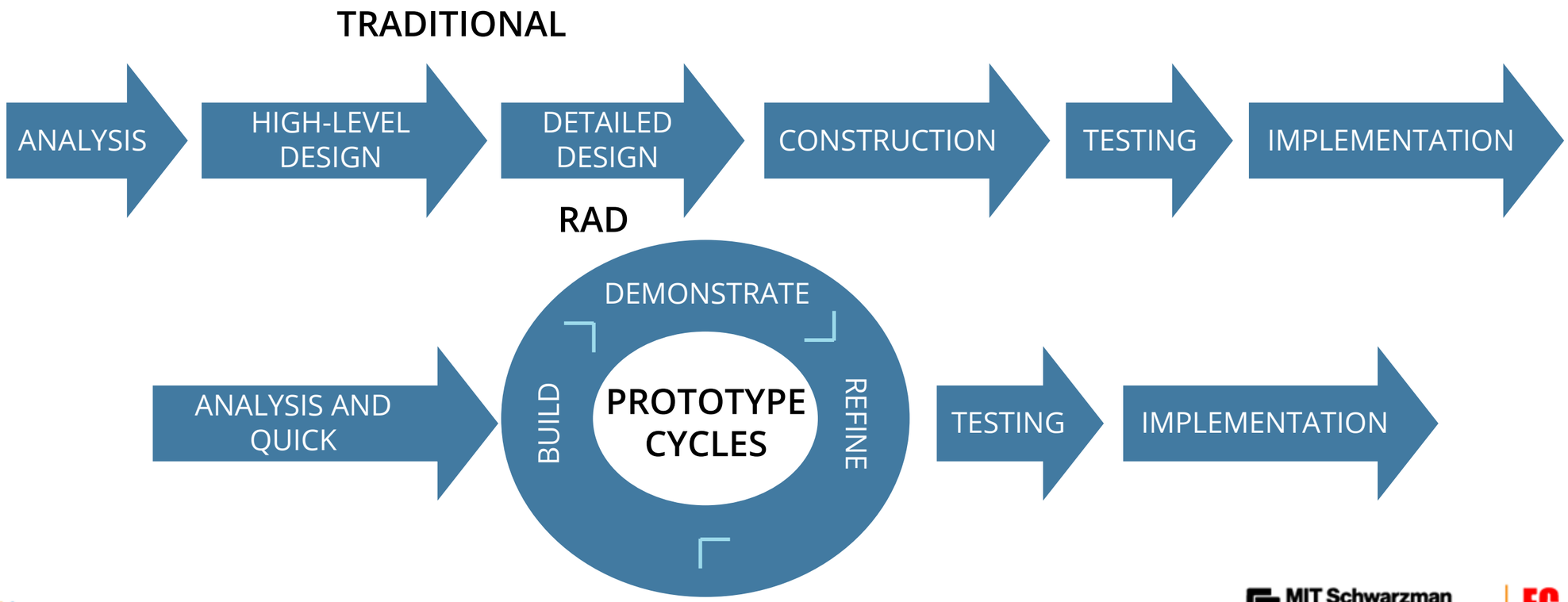
- In 1988, Barry Boehm developed the spiral model.
- It is a meta-model that incorporates several software development models.
- It combines the idea of iterative development with the systematic and controlled aspects of the waterfall model.
- It includes risk management within software development.



Software Development Models

Rapid-application development

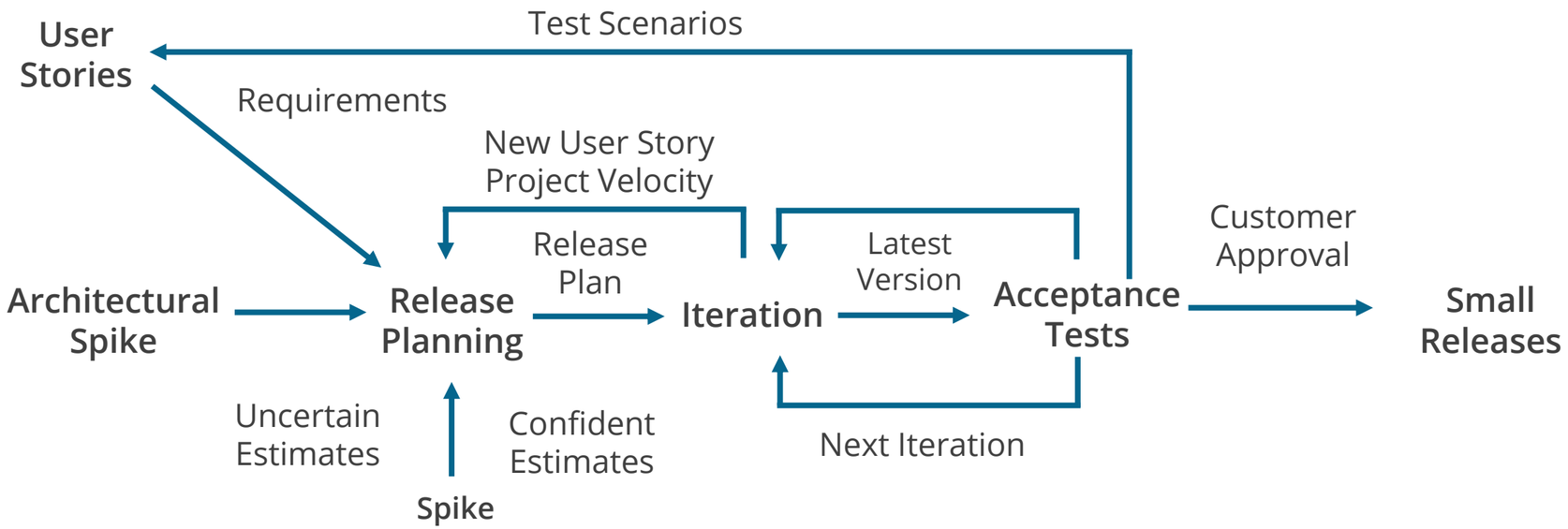
- Rapid-application development (RAD) is a form of rapid prototyping.
- Software is developed via the use of prototypes, dummy GUIs, and back-end databases.
- The goal is to meet the system's business need.



Software Development Models

Extreme programming

- Extreme programming is a discipline of software development.
- It is based on the values of simplicity, communication, and feedback.
- It is a structured approach, relying on subprojects.
- It is an agile software development method.



Software Development Models

Other software development models include:

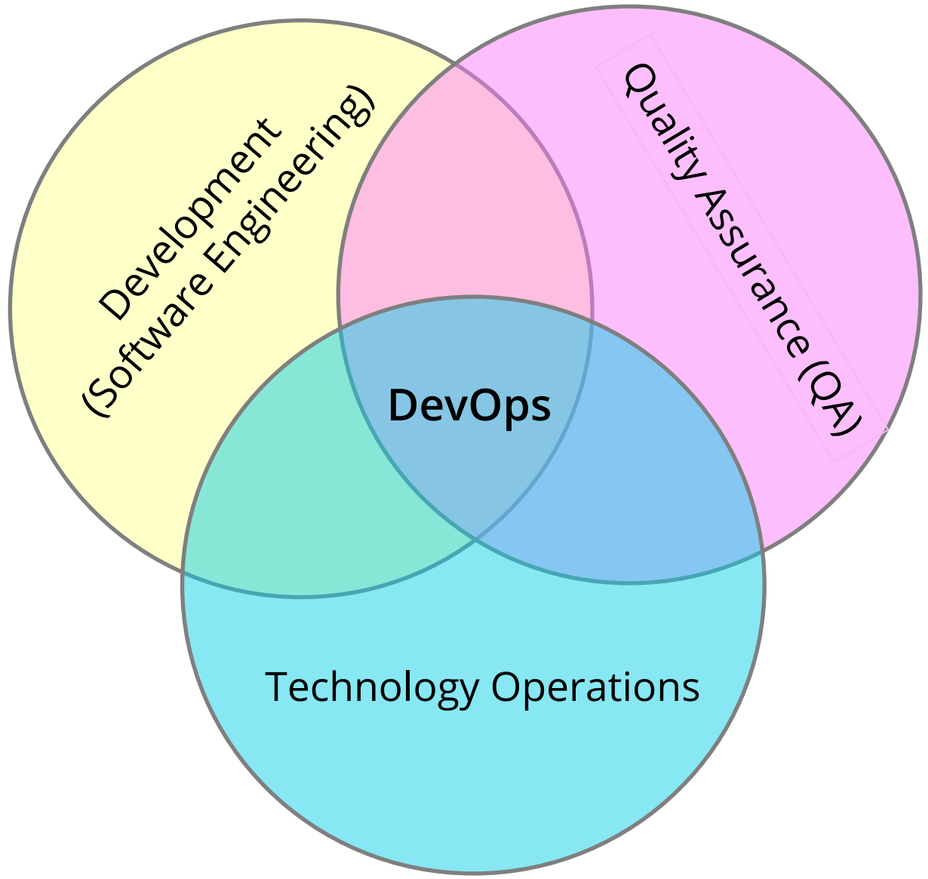
- Prototyping
- Modified prototype model
- Joint analysis development (JAD) model
- Exploratory model
- Computer-aided software engineering (CASE)
- Component-based development
- Reuse model

DevOps

DevOps, derived from the terms **development** and **operations**, is a software development method that emphasizes communication, collaboration, and integration between the organization's software developers and IT staff.

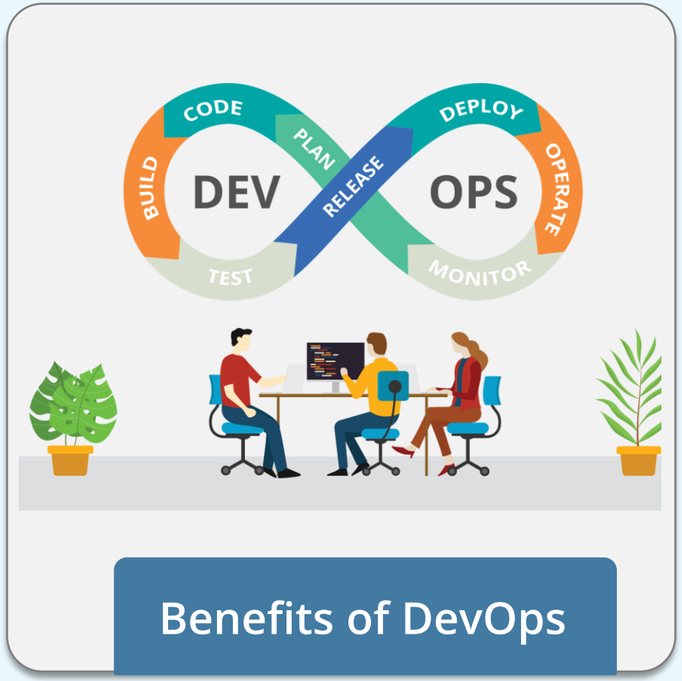
Features:

- It helps an organization quickly produce software products and services.
- It ensures the adoption of Quality Assurance to improve Technology Operations' performance.



DevOps

- Better communication between developers and operations
- Continuous integration and continuous deployment
- Increased software quality and security
- Rapid improvement based on regular feedback
- Faster time to market for software

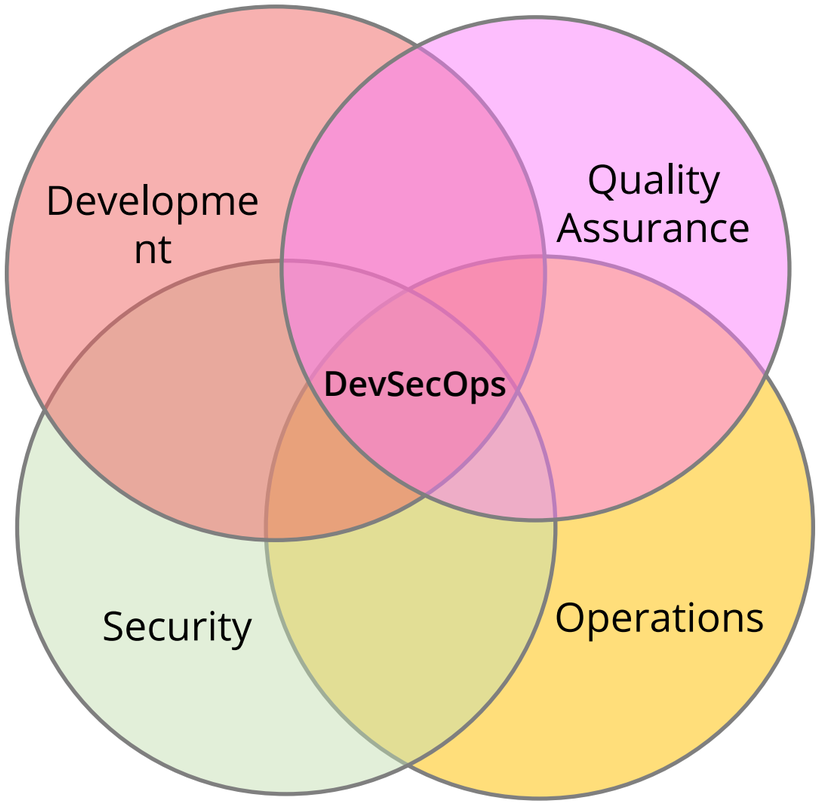


Benefits of DevOps

DevSecOps

Trade secret

- DevSecOps extends the DevOps workflow to include automated security processes and tools.
- DevSecOps follows secure by design principle by using automated review of code and automated application of security testing and educating and empowering developers to use secure design patterns.



DevSecOps

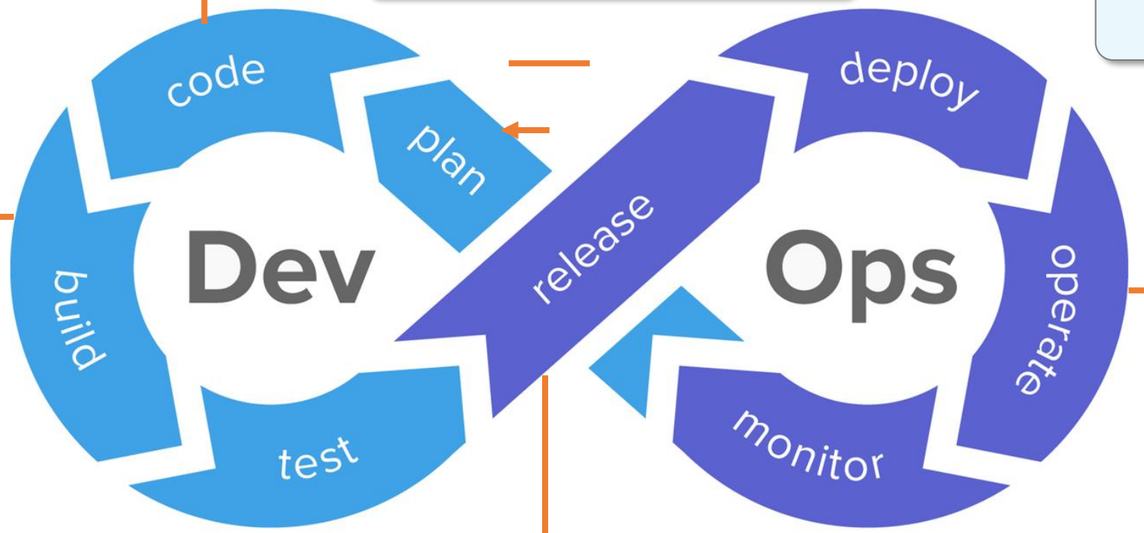
- Identify vulnerabilities in the early stages of the software development life cycle thereby reducing costs and increasing the deployment speed

- Improve overall security by reducing vulnerabilities, reducing insecure defaults, and increasing code coverage and automation using immutable infrastructure

- Reduce the mean time to resolve (MTTR) security incidents by monitoring and using remediation practices

Benefits of DevSecOps

DevSecOps



- Threat model
- Security architecture
- Training

- Secure code review
- Static analysis (SAST)
- Third-party component analysis

- Configuration management
- Hardening
- Infra-vulnerability scanning

- Access control
- Patch management
- Audit and compliance
- Change control

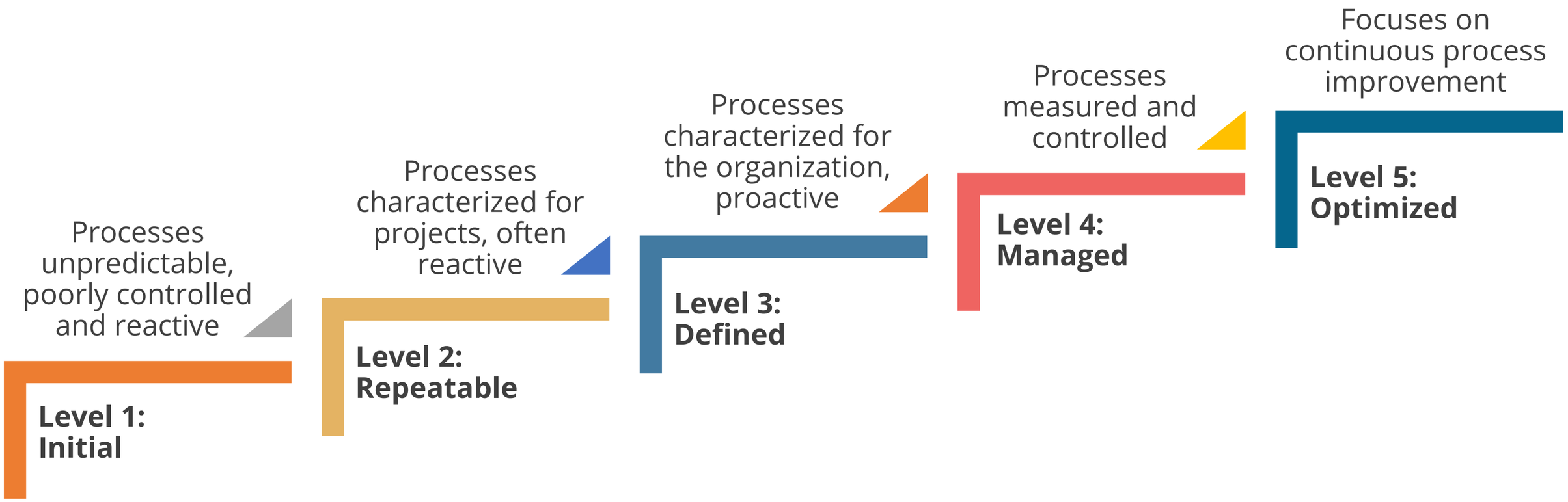
- Penetration testing
- Dynamic analysis (DAST)

- Final security review
- Compliance validation

- Log collection
- Intrusion detection
- Analytics
- SIEM

Software Capability Maturity Model (CMM) Levels

The Capability Maturity Model (CMM) is based on the premise that the quality of a software product is a direct function of the quality of its associated software development and maintenance processes. The five maturity levels of CMM are:



Information source: https://en.wikipedia.org/wiki/Capability_Maturity_Model and https://imgur.com/WERTTou

Software Assurance Maturity Model (SAMM)

The Software Assurance Maturity Model (SAMM) is an open framework that provides an effective, measurable way for all types of organizations to analyze and improve their software security posture.

Software Assurance Maturity Model (SAMM)

Measurable

Defined maturity levels across business practices

Actionable

Clear pathways for improving maturity levels

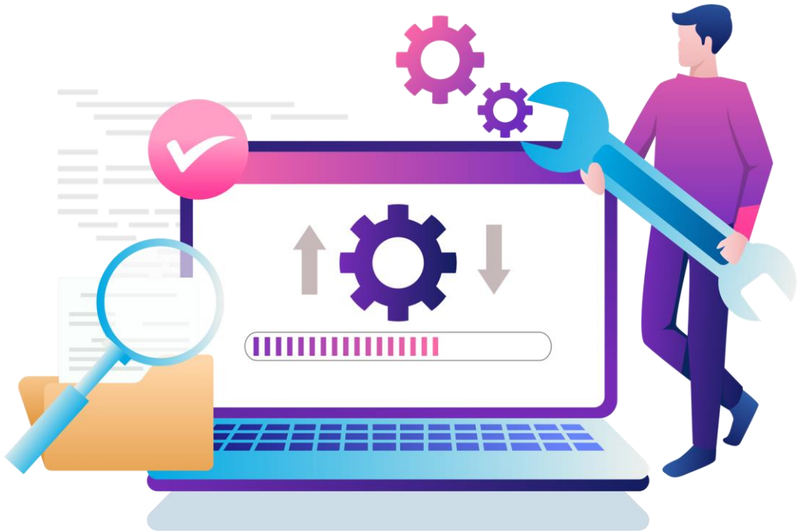
Versatile

Technology, process, and organization agnostic

SDLC: Operation and Maintenance

The various activities in the operation and maintenance phase are:

- Ensure operations' continuity
- Monitor system performance
- Detect vulnerabilities
- Manage and avoid problems in the system
- Secure recovery of systems
- Analyze risk periodically
- Follow change management procedures
- Verify compliance



Change Management

Change management allows companies to manage, monitor, and optimize software changes to ensure that:

Change Management

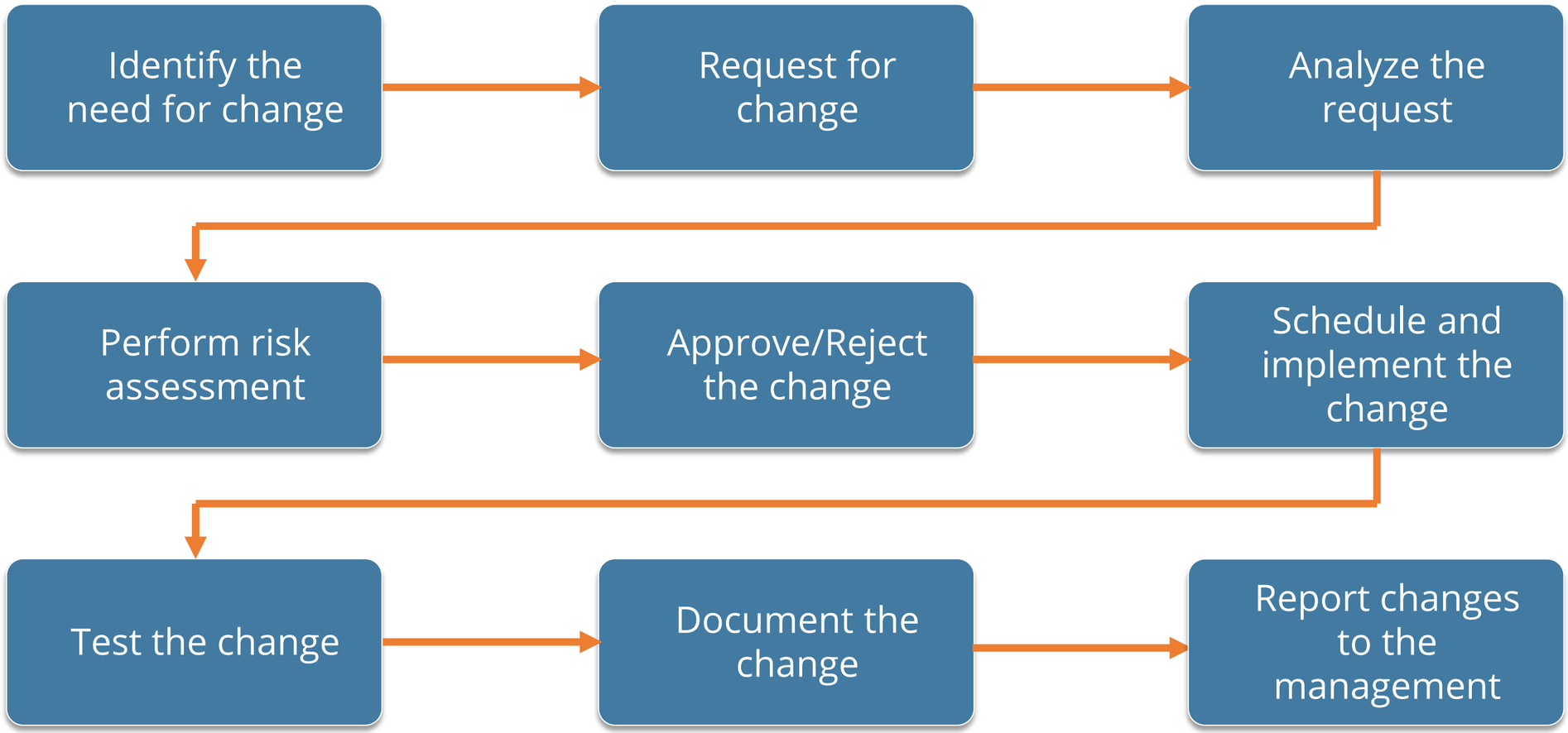
Software change management follows a recognized procedure.

Due diligence is carried out to assess the business impact of any software change prior to a decision being taken on if the rollout takes place.

The scope is verified for completion and accuracy both before and after the deployment or change takes place.

Change Management

The following flow diagram explains the change management process.



Integrated Product Team (IPT)

An integrated product team (IPT) is a multi-disciplinary team that helps facilitate decision-making by:

- Working together to build successful programs
- Identifying and resolving issues
- Making comprehensive and timely recommendations

Integrated Product Team (IPT)

The team comprises of members from the organization's appropriate functional disciplines and is:

- Used to review and make decisions in complex programs and projects
- A forum for collaboration that involves all stakeholders

Identify and Apply Security Controls in Software Development Ecosystems

Application Controls

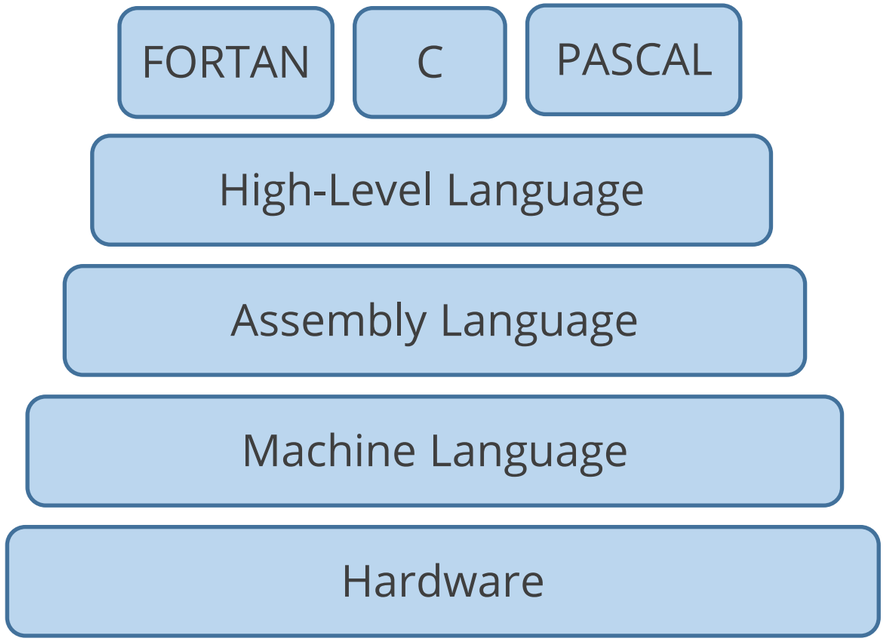
The following are the controls and categories of application controls:

Control Type	Accuracy	Security	Consistency
Preventive	Data checks, custom screens, validity checks, contingency planning, and backups	Firewalls, reference monitor, sensitivity labels, traffic padding, encryption, data classification, one-time passwords, separate test, and development environments	Data dictionary, programming standards, and DBMS
Detective	Cyclic redundancy checks, structured walk through, hash totals, and reasonableness checks	IDS and audit trails	Comparison tools, relationship tests, and reconciliation controls
Corrective	Backups, control reports, before and after imaging reports, and checkpoint restarts	Emergency response and reference monitor	Program comments and database controls

Programming Languages

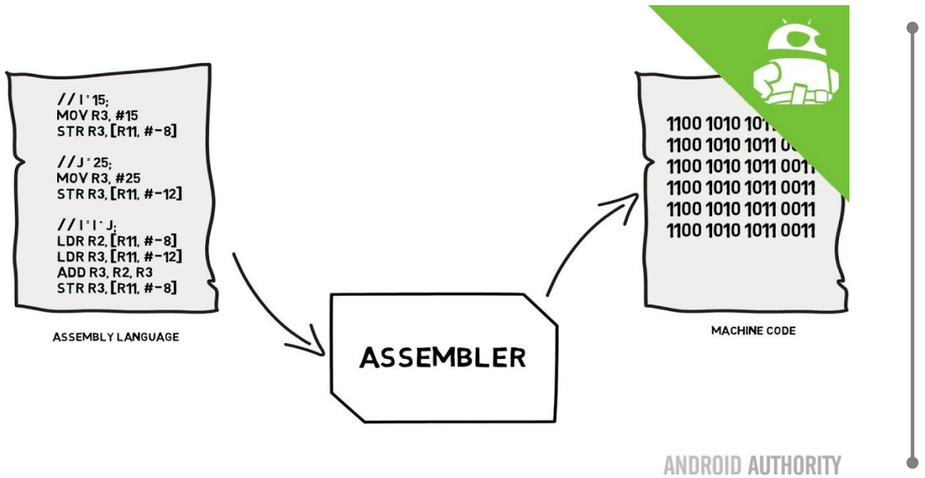
The common types of programming languages are as follows:

- **Machine language**: Machine language is a software program that is executed directly by the CPU.
- **Assembly language**: Assembly language is a low-level computer programming language.
- **High-level language**: In high-level language, programmers write the code using logical words and symbols.



Assembly Language

An assembly language is a low-level programming language designed for a specific type of processor.



Assembly language must be converted into executable machine code by a utility program referred to as an assembler.

Image Source: https://www.androidauthority.com/assembly-language-and-machine-code-678230/

Compiler Vs. Interpreter

Basis of Difference	Compiler	Interpreter
Compiling a Program	A compiled program is compiled only once.	An Interpreted code is compiled each time the program is run.
High-Level Instructions	It translates high-level instructions directly into machine language.	It translates high-level instructions into an intermediate form.
Speed	Compiled programs run faster.	Interpreted programs run slower than compiled programs.
Search for Errors	It searches for all the errors of a program and lists them.	It checks a program statement by statement for errors.
Error List Generation	It generates the error message only after scanning the whole program.	It continues translating the program until the first error is met, in which case it stops.
Debugging	It is comparatively hard.	It is easy.
Use	It is difficult to use.	It is easier to use.
Examples	C, C++	Python, Ruby

Libraries

A **software library** is a set of precompiled helper functions, objects, or modules that are intended to be **reused** during software development.

Software libraries include the following components:

- Standard libraries provided by most languages such as Python, C, C++, C#, Java, Ruby etc.
- Open-source libraries, which are free to reuse, modify and publish without any permission
- Third-party libraries
- Custom libraries

Some of the software library implementation best practices are:

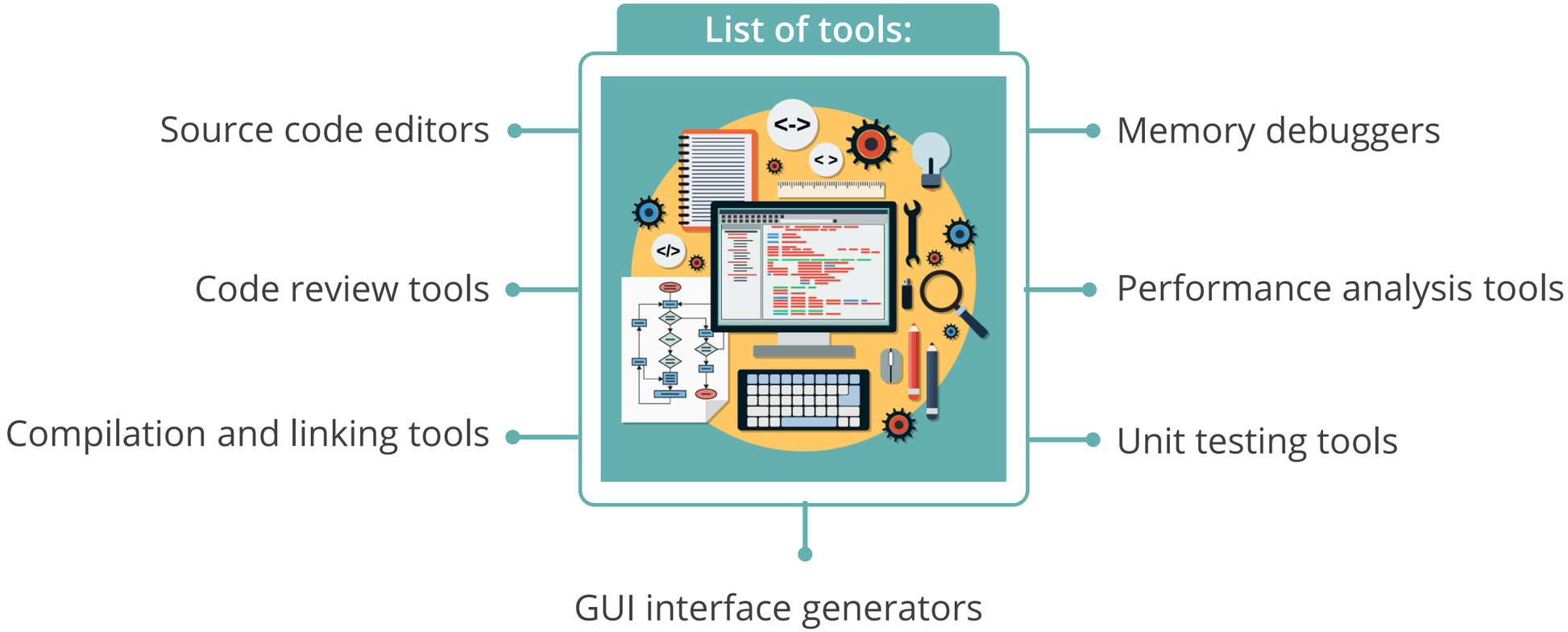
- Use libraries only from trusted sources that are actively maintained and widely used by several applications
- Create and maintain an inventory catalog of all the third-party libraries
- Proactively keep libraries and components up to date

Toolsets

A software development **tool** is a program that software developers use to create, debug, maintain, or otherwise support other programs and applications.

Toolsets

Tools and toolsets can help improve the developer's productivity.



Integrated Development Environment (IDE)

Integrated development environments, or IDEs, are software platforms that provide programmers and developers with a comprehensive set of tools for software development in a single product.

Benefits of using IDEs

- IDEs provide programming capabilities through a text editor or a GUI (Graphical User Interface).
- IDEs come with some pre-installed libraries for the specific programming languages.
- IDEs support compiling, debugging, version control, platform-specific code suggestions, or code deployment.

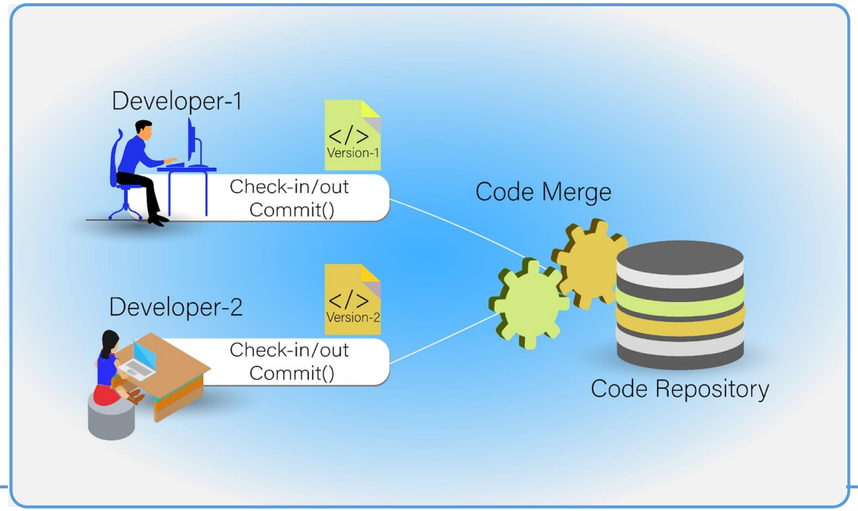
Runtime System

- A **runtime system** refers to the collection of hardware and software resources needed for program execution on a computer system.
- The low-level services provided by a runtime system include processor interfacing, memory loading, and digital-to-binary conversion among others.
- The higher-level services include type checking, code generation, debugging, or code optimization.

For example: Java Runtime Environment (JRE) provides the complete framework for executing and managing Java programs.

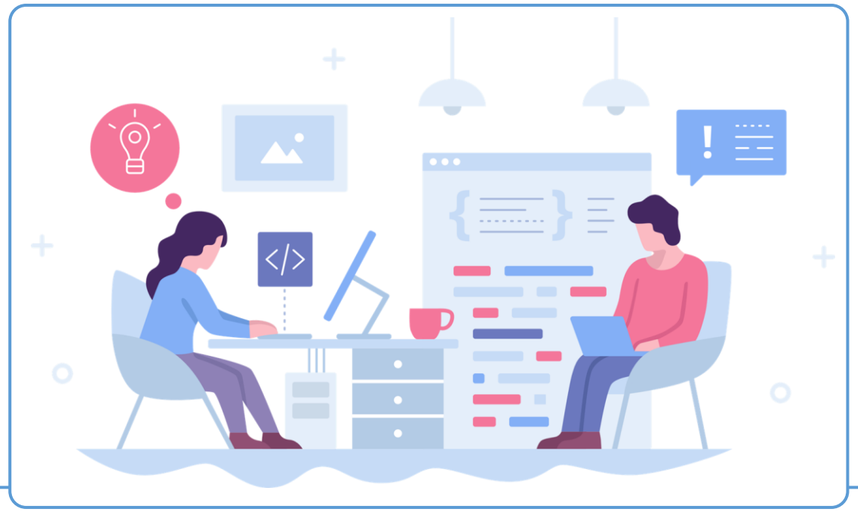
Continuous Integration

Continuous Integration and Continuous Delivery (CI/CD) is a modern software development practice that allows for frequent code change in an incremental, repeatable, and secure manner.



Continuous Integration: Application code changes are regularly built, tested, and merged to a shared repository several times a day.

Continuous Delivery and Continuous Deployment

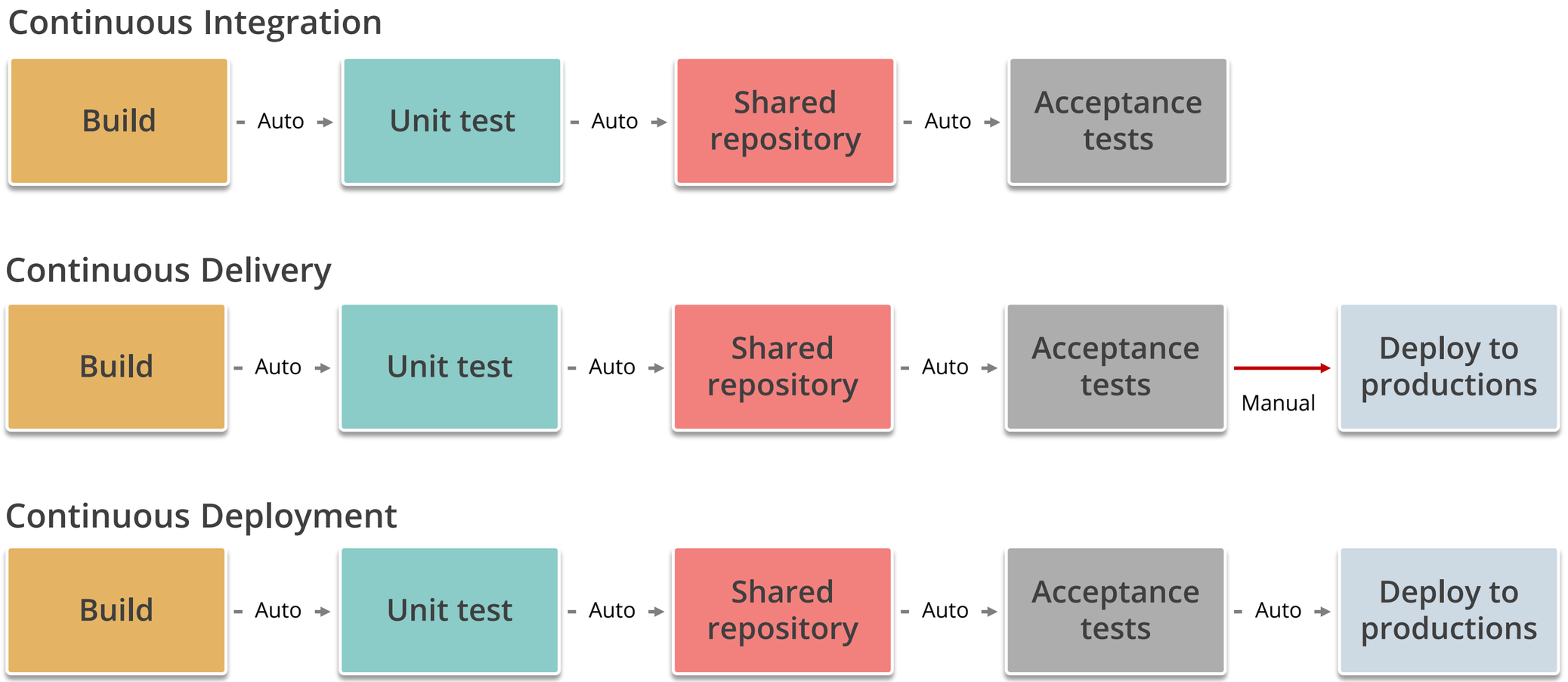


Continuous Delivery: Application code changes are automatically bug tested and uploaded to a repository (like Github), ready to be deployed to production by the operations team (manually).

Continuous Delivery and Continuous Deployment

Continuous Deployment is the next step after continuous delivery. Application code changes are automatically deployed into production whenever it passes the automated tests.

Continuous Integration, Continuous Delivery, and Continuous Deployment



Security Orchestration, Automation, and Response (SOAR)

Security Orchestration, Automation, and Response (SOAR) refers to software solutions and tools that aggregate inputs from a variety of sources and build automated response to low-level known security events.

Capabilities of SOAR technologies include:

- Threat and vulnerability management
- Security incident response
- Security operations' automation

Strengths, Opportunities, Aspirations, and Results

Discussion

Discussion

Why is it important to prevent developers of code from being able to move their compiled programs into production?

Software Configuration Management (SCM)

Software configuration management (SCM) refers to the process to systematically manage, organize, and control the changes in the documents, codes, and other entities during the SDLC.

Information source: https://www.energy.gov/sites/default/files/cioprod/documents/scmguide.pdf

Software Configuration Management (SCM) Processes

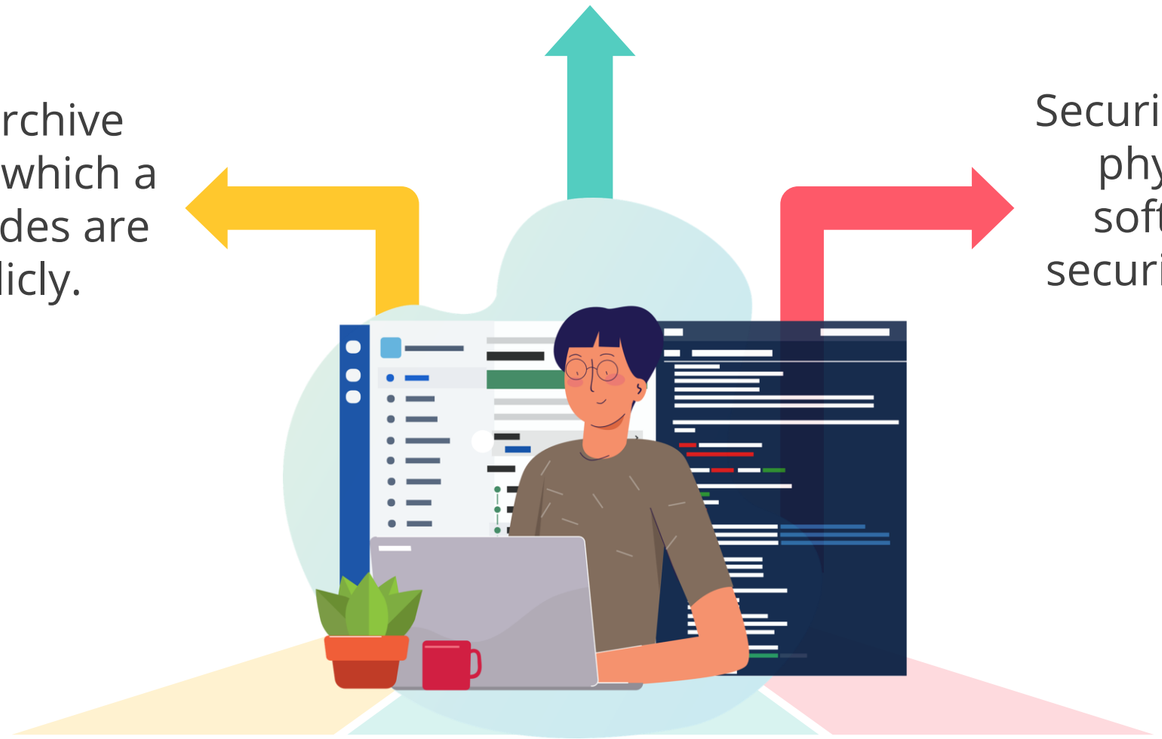
- **Identification** is the process of identifying all the components of a project.
- **Version control** combines procedures and tools to handle different versions of configuration objects that are generated during the software process.
- **Change control** is the process where the proposed changes to the project are reviewed and incorporated into the software configuration if approved.
- **Configuration audit** is the process to confirm that the approved changes have been implemented.
- **Reporting** provides accurate data on the current status and configuration.

Information source: https://www.energy.gov/sites/default/files/cioprod/documents/scmguide.pdf

Code Repositories

For example, a source code repository is used by open-source projects and other multi-developer projects to handle various versions.

Code repository is a file archive and Web hosting facility in which a large number of source codes are stored privately or publicly.



Securing a code repository includes physical, system, operational, software, and communication security, file systems and backups, and access control.

Source Code Analysis Tools

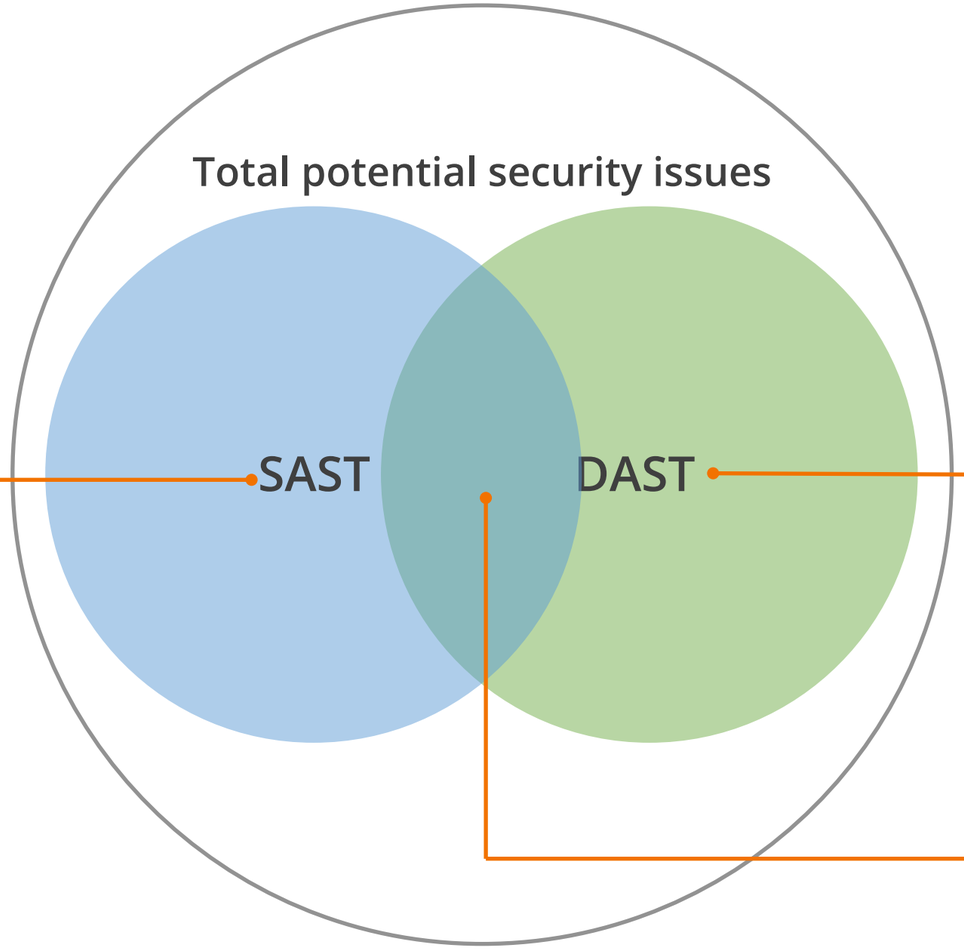
Static application security test (SAST)

- White-box security test
- Requires source code
- Finds vulnerabilities in the earlier stages of an SDLC
- Less expensive to fix vulnerabilities
- Can’t discover runtime- and environment-related issues
- Supports all software

Dynamic application security test (DAST)

- Black-box security test
- Requires a running application
- Finds vulnerabilities towards the later stages of an SDLC
- More expensive to fix vulnerabilities
- Can discover runtime- and environment-related issues
- Predominantly deals with Web apps

SAST vs. DAST



SAST only

- Hard-coded secrets
- Code quality issues
- Issues in dead or unused code
- Insecure crypto functions
- Complex injection issues

DAST only

- Environment configuration issues
- Patch-level issues
- Runtime privilege issues
- Authentication issues
- Session management issues

DAST and SAST

- SQL injection
- Cross-site scripting
- Buffer overflows
- Path traversal
- Format string issues

Source Code Analysis Tools

Interactive application security testing (IAST)

- Combines the advantages of SAST and DAST
- Agents and sensors within an application analyze all interactions to identify vulnerabilities in real time
- Accurately identifies the source of vulnerability
- Performed during the early stages of the SDLC
- Integrates easily into CI/CD pipelines

Runtime application security protection (RASP)

- Incorporates security into a running application on a server
- Detects and blocks cyber attacks on an application in real time without human intervention
- May have an adverse effect on application performance
- May create a sense of false security within a development team

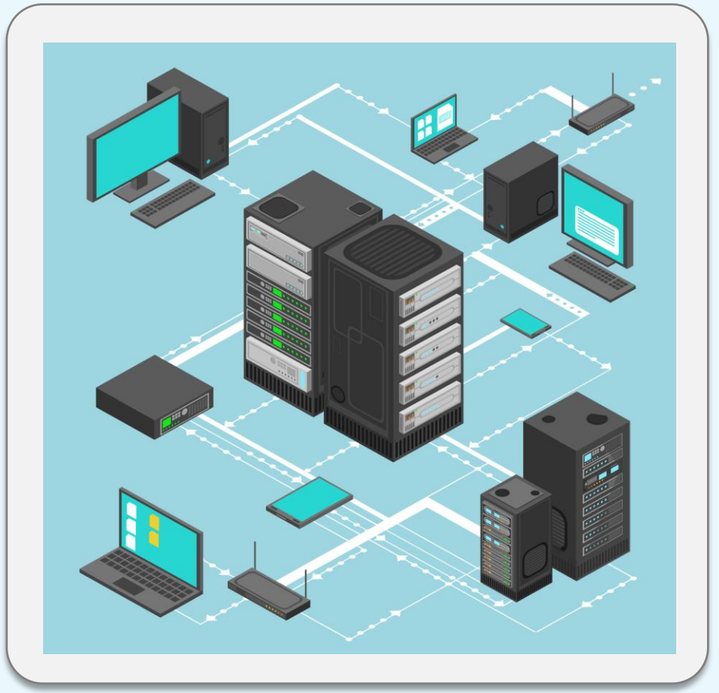
Database and Data Warehousing Environments

Database concepts:

A database is a structured collection of related data that allows queries, insertions, deletions, and many other functions.

The database model should provide the following:

- Persistence
- Data sharing
- Recovery or fault tolerance
- Database language
- Security and integrity



Database Terms

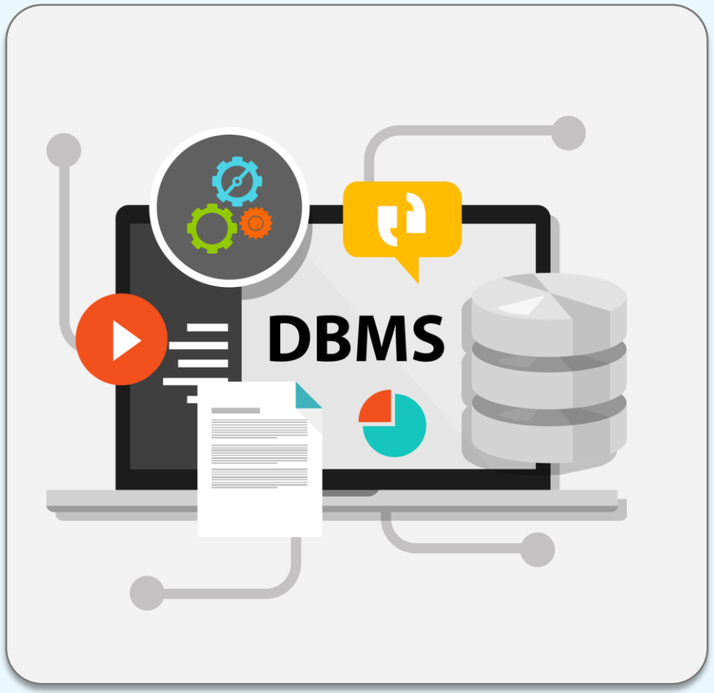
The common database terms are as follows:

Record	File	Database	Database management system (DBMS)
Attribute or column	Tuple or row	Primary key	View
Foreign key	Cell	Schema or structure	Data dictionary

DBMS Controls

The basic DBMS controls are:

- Lock controls
- ACID: Atomicity, consistency, isolation, and durability
- Discretionary access control (DAC)
- Mandatory access control (MAC)
- View-based access controls
- Grant and revoke access controls
- Metadata controls
- Data contamination controls
- Online transaction processing (OLTP) control



Database: Threats and Vulnerabilities

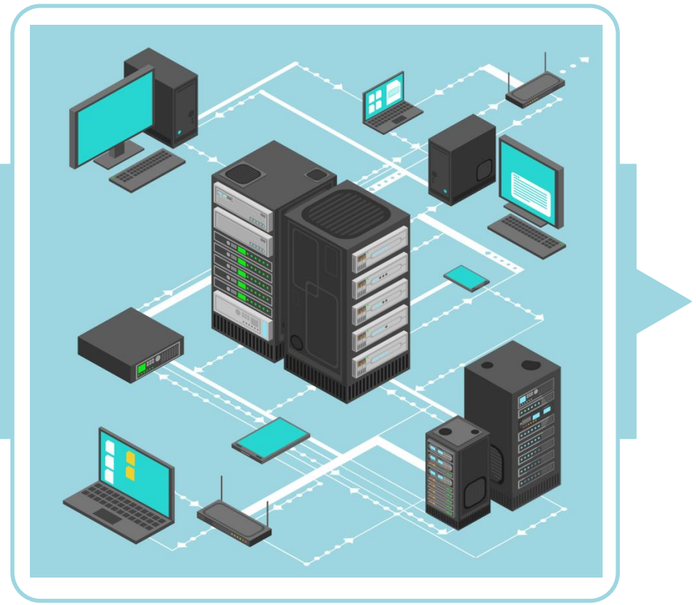
The common database threats and vulnerabilities are:

Aggregation	Bypass attacks	Inference	Polyinstantiation
Views	Concurrency	Data contamination	Deadlocking
Denial of service	Improper modification of data	Interception of data	Query attacks
Server access	Time of check or time of use (TOC or TOU)	Web security	Unauthorized access

Introduction to Data Warehousing

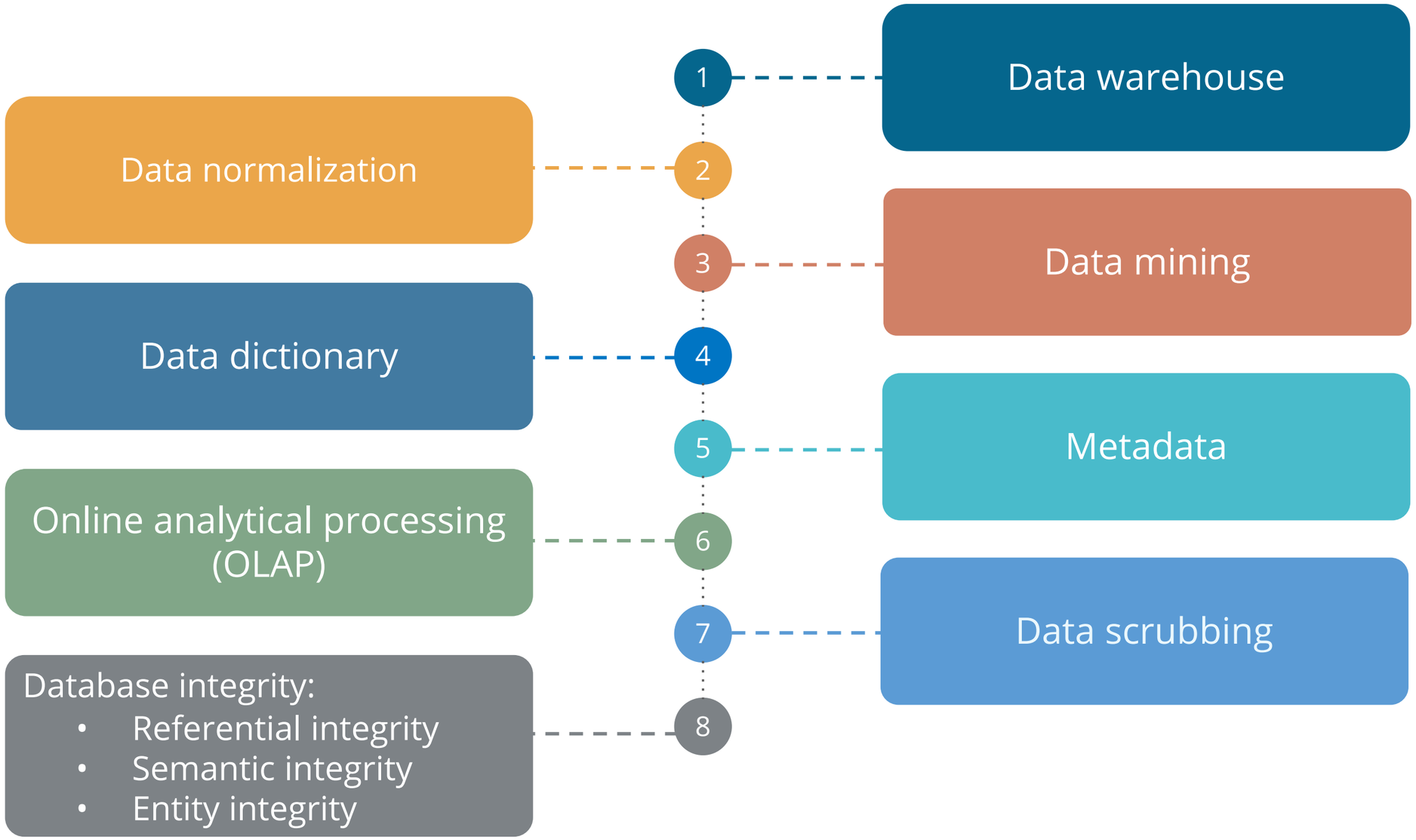
A data warehouse is a database designed to enable business intelligence activities.

It is designed for query and analysis and contains historical data derived from transaction data.



To enhance business intelligence, it works with data collected from multiple sources.

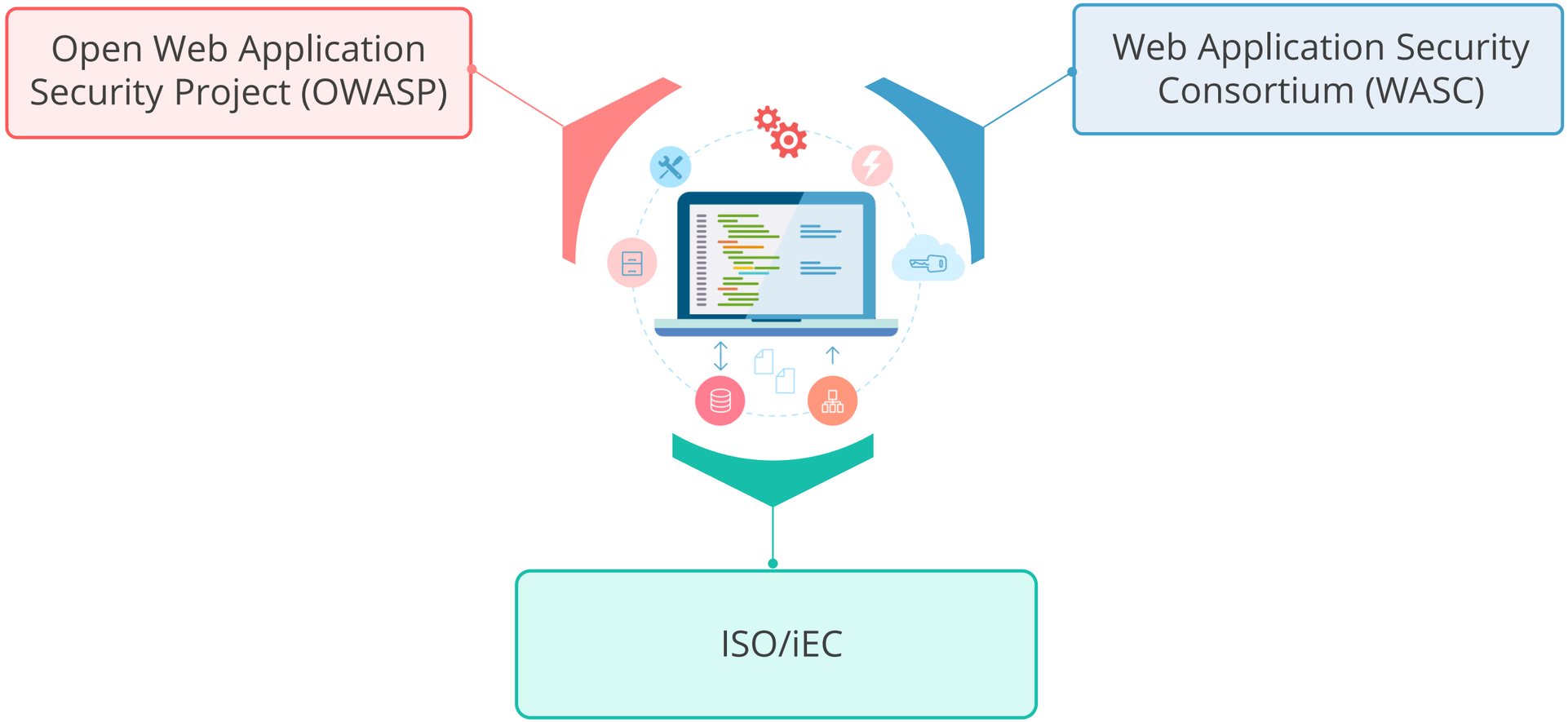
Data Warehousing Concepts



Assess the Effectiveness of Software Security

Secure Software Development: Best Practices

The best practices for secure software development are provided by:



Software Security and Assurance

Security kernel is responsible for enforcing a security policy.

It is a strict implementation of a reference monitor mechanism.

It is a small portion of the operating system through which all references to information and all changes to authorization must pass.

Software Security and Assurance

Three basic conditions of kernel:

- Completeness
- Isolation

- Verifiability

Software Security and Assurance

Processor privilege states	• The processor privilege states protect the processor and the activities that it performs. • It records the processor state in a register that can only be altered when the processor is operating in a privileged state. • The hardware typically controls entry into the privilege mode. • The privilege-level mechanism should prevent memory access.
Bounds checking	• Bounds checking is any method of detecting whether a variable is within some bounds. • It prevents buffer overflows on input.

Software Security and Assurance: Parameter Checking

- Parameter checking is implemented by the programmer.
- It involves checking the input data for disallowed characters, length, data type, and format.
- Other technologies to protect against buffer overflows include canaries.

Software Security and Assurance: Memory Protection

Memory Protection is necessary to protect the memory used by one process from unauthorized access by another.

Memory protection can be ensured by partitioning memory:

- To ensure processes cannot interfere with each other's local memory
- To ensure common memory areas are protected against unauthorized access

Software Security and Assurance: Granularity of Controls

Granularity of controls ensures that the security controls are granular enough to address both program and user.

Inadequate granularity of controls can be addressed by:

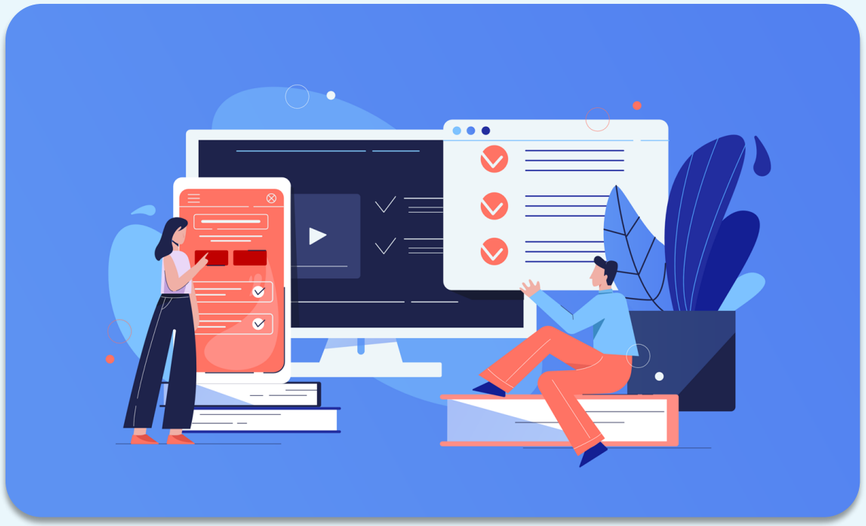
- Proper implementation of the concept of least privilege
- Setting reasonable limits on the user
- Separation of duties and functions
- Ensuring programmers are not the system administrators or users of the application
- Granting users only those permissions necessary to do their job

Software Security and Assurance: Separation of Environments

Separation of environments is essential to control how each environment can access the application and the data.

Control measures to protect the various environments include:

- Physical isolation of environment
- Physical or temporal separation of data for each environment
- Access control lists
- Content dependent access controls
- Role-based constraints
- Role definition stability
- Accountability
- Separation of duties



Business senario

Kevin read the policy which Hilda Jacobs, General Manager: IT Security, Nutri Worldwide Inc., had created for improving the software development process. Per the policy, programmers will write, compile, and carry out initial testing of the application's functionality and implementation in the development environment.

- When the application is ready for production, the users and quality assurance team will carry out functional testing within the testing and quality assurance environment. When the application is accepted by the user community, it is moved into production environment.

Question: Which software security mechanism is the policy based on?

Business senario

Kevin read the policy which Hilda Jacobs, General Manager: IT Security, Nutri Worldwide Inc., had created for improving the software development process. Per the policy, programmers will write, compile, and carry out initial testing of the application's functionality and implementation in the development environment.

- When the application is ready for production, the users and quality assurance team will carry out functional testing within the testing and quality assurance environment. When the application is accepted by the user community, it is moved into production environment.

Question: Which software security mechanism is the policy based on?
Answer: : Separation of environments into development, quality assurance, and one of the production, application, or production environments.

Software Security and Assurance: TOC or TOU

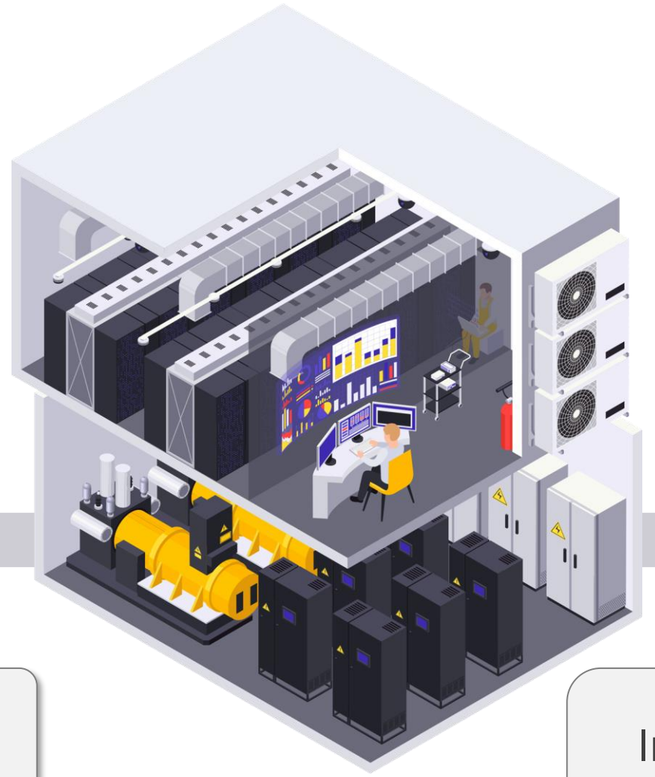
Prevention of Time of Check or Time of Use (TOC or TOU)

- The common Time of Check or Time of Use (TOC or TOU) hazards are file-based race conditions
- To avoid TOC or TOU problems:
  - Programmer should avoid any file system call that takes a filename for an input
  - Files that are to be used should be kept in their own directory
  - Directory must only be accessible by the universal ID (UID) of the program performing the file operation

Software Security and Assurance: Prevention of Social Engineering

- Social engineering is a way where the attackers try to use social influence over users to extract confidential information.
- To protection against social engineering attacks:
  - Provide users and help desk staff a proper framework to work
  - Make users aware of the threat
  - Give users the proper procedures for handling unusual requests for information

Software Security and Assurance: Backup



Backing up operating system and application software is a method of ensuring productivity in the event of a system crash.

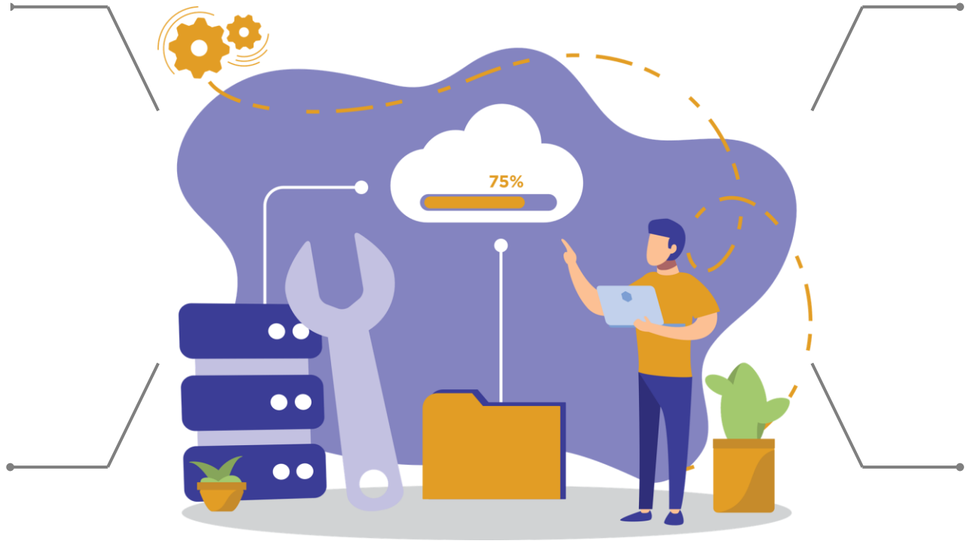
Information is available in the event of an emergency through data, programs, documentation, computing, and communications equipment redundancy.

Software Security and Assurance: Backup

Backup control functions include:

Redundant array of independent disks (RAID)

Disk mirroring

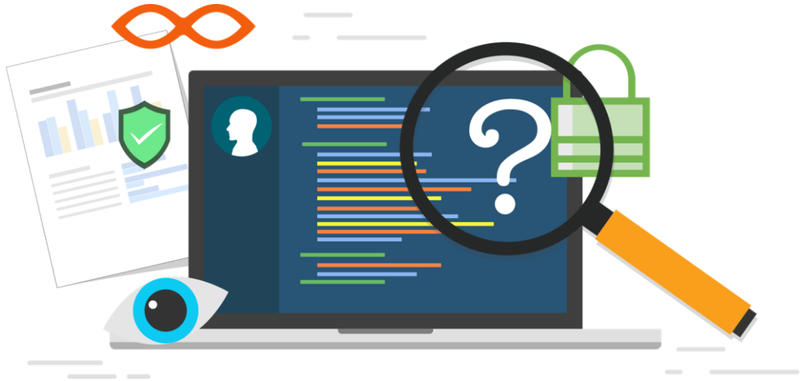


Maintaining the source code

Contingency planning documents

Software Security and Assurance: Software Forensics

- Software forensics is the study of malicious software and protection against malicious code.
- It can be used:
  - To determine whether a problem is a result of carelessness or was deliberately introduced as a payload
  - To obtain information about authorship and the culture behind a given programmer
  - To obtain the sequence in which related programs were written
  - To provide evidence about a suspected author of a program
  - To determine intellectual property issues
  - To recover source code that has been lost



Software Security and Assurance: Cryptography

They are used to protect the confidentiality and integrity of information.

Cryptographic techniques protect information by transforming the data through encryption schemes.

Encryption algorithms can be used to encrypt specific files located within the operating system.

Software Security and Assurance: Password Protection

- Operating system and application software use passwords as a convenient mechanism to authenticate users.
- Password protections include controls on:
  - How the password is selected
  - How complex the password is
  - Password time limits
  - Password length
- The most common solution is to encrypt password files using one-way encryption algorithms or hashing.
- Another feature for password security involves an overstrike or password masking feature.

Software Security and Assurance: Mobile Code Controls

Mobile code controls protect the user from viewing web pages which have programs attached to them

The system should garbage collect memory to prevent both malicious and accidental memory leakage.

Secured systems should limit mobile code or applets access to system resources.

Software Security and Assurance: Sandbox

Sandbox is one of the control mechanisms for mobile code.

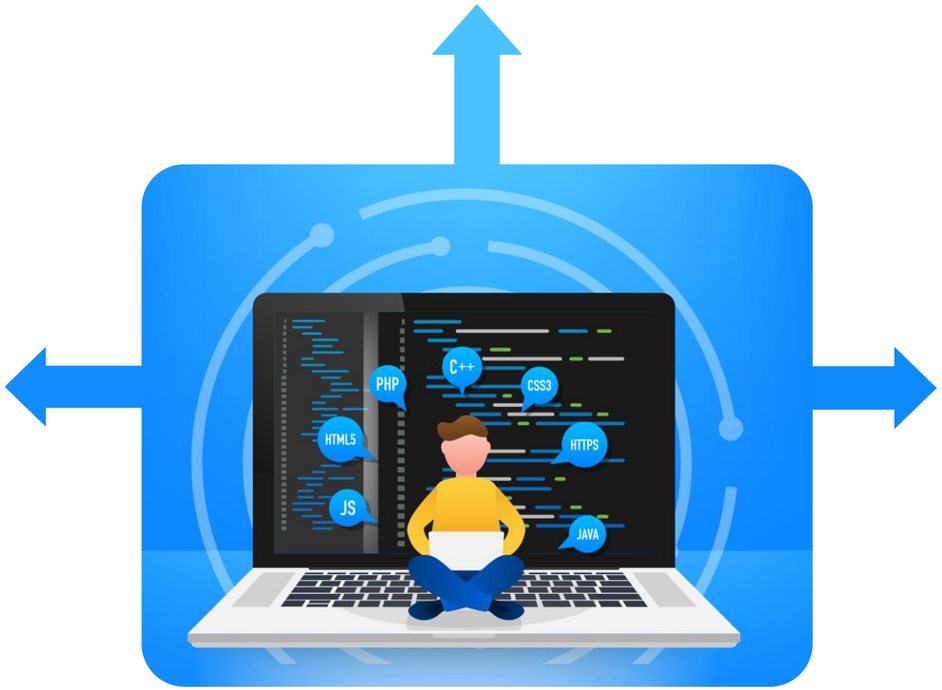
Limits are placed on the amount of memory and processor resources the program can consume.

It provides a protective area for program execution.

A sandbox can be created on the client side to protect the resource usage from applets.

Software Security and Assurance: Strong language support

Strong language support is a method of providing safe execution of programs such as Java.



It ensures that arrays stay in bounds, the pointers are always valid, and code cannot violate variable typing.

Java does an internal check called static type checking.

Software Security: XML and Security Assertion Markup Language

Following are the languages used to provide software security:

XML

- XML stands for Extensible Markup Language.
- It is a world wide web consortium standard for structuring data in a text file.
- XML is called extensible because the symbols are unlimited and can be defined by the user or author.

SAML

- SAML stands for Security Assertion Markup Language.
- It is a format that uses XML to describe security information.
- An important requirement for this is the web browser single sign-on (SSO).
- An example is using cookies.

Audit and Assurance Mechanisms

The following are the audit and assurance mechanisms:

Methods to Assess the Effectiveness of Software Security

- System Authorization
- Auditing and Logging
- Risk Analysis and Mitigation
- Testing and Verification

- It involves certification and accreditation or authorization of systems that process, store, or transmit information.
- It ensures a control framework is selected and uniformly implemented across the organization with the help of standards.

Methods to Assess the Effectiveness of Software Security

- System Authorization
- **Auditing and Logging**
- Risk Analysis and Mitigation
- Testing and Verification

- Most software are released with vulnerabilities; auditing and logging help identify security issues.
- Organizations must address these issues by applying procedures to maintain and check information integrity, accuracy.

Methods to Assess the Effectiveness of Software Security

- System Authorization
- Auditing and Logging
- **Risk Analysis and Mitigation**
- Testing and Verification

Risk analysis and mitigation must be integrated in the SDLC as an ongoing activity, and in change management. It involves:

- Using standardized methods outlined in frameworks such as ISO and NIST to assess risk and report to stakeholders
- Tracking and managing vulnerabilities
- Taking corrective actions for mitigation by reviewing and prioritizing the findings

Methods to Assess the Effectiveness of Software Security

- System Authorization
- Auditing and Logging
- Risk Analysis and Mitigation
- **Testing and Verification**

All mitigations applied should be thoroughly tested and verified by independent security assessors to ensure that the security flaw has been mitigated.

Certification

Certification and accreditation (C and A) is the process used to evaluate and approve a system for use. C and A is a two-step process that includes:

- Technical review that assesses the security mechanism and evaluates their effectiveness
- The process may use safeguard evaluation, risk analysis, verification, testing, and auditing techniques
- The goal is to ensure the system is right for the customer's purpose
- Certification is often an internal verification and is only trusted within the organization

Accreditation

Certification and accreditation (C and A) is the process used to evaluate and approve a system for use. C and A is a two-step process that includes:

- Accreditation is the formal declaration by the designated approving authority (DAA) that an IT system is approved to operate in a particular security mode using a prescribed set of safeguards at an acceptable level of risk.
- Once the accreditation is performed, management can formally accept the adequacy of the overall security performance of an evaluated system.

Assess Security Impact of Acquired Software

Assessing the Security Impact of Acquired Software

Acquired software can introduce new vulnerabilities into the system and may have an impact on the organization's risk posture.

The security of the acquired software can be assessed by:

- Using security tools to test the software for vulnerabilities
- Verifying whether the software development firm followed secure processes
- Checking developer conformance to international standards such as ISO 27034

Commercial-Off-The-Shelf (COTS)

Commercial-off-the-shelf (COTS) refers to hardware or software products that are ready-made and available for purchase in the commercial market.

Commercial-Off-The-Shelf (COTS)

- COTS usually offer paid or free customer support.
- COTS vendor regularly releases software and firmware patches to address vulnerabilities.
- Security vulnerabilities in COTS can introduce significant risk.
- Unavailability of source code limits testing and monitoring COTS.
- COTS helps companies validate if security testing has been performed with international standards such as Common Criteria and FIPS.
- COTS has the ability to perform black box and fuzzy testing.

Free and Open-Source Software

Free and Open-Source Software (FOSS) is licensed to be free to use, modify, and distribute.

This allows other developers the opportunity to contribute to the development and improvement of a software like a community.

Free and Open-Source Software

Free software doesn't mean the software is free of cost. While FOSS is often available free of charge, its main difference with proprietary software is that it is free, as in **freedom**.

The **freedom to run** the program as you wish, for any purpose

The **freedom to study** how the program works and change it so it does your computing as you wish

The **freedom to redistribute** copies so you can help others

The **freedom to improve** the program and release your modified versions to others

Open-Source

Advocates of open-source software believe that if more users view the source code, they will eventually find all bugs and suggest how to fix them. Linus's Law states that **given enough eyeballs, all bugs are shallow.**

On the other hand, advocates of proprietary systems note that open-source software may allow hackers to find security vulnerabilities more easily than closed-source software. Because the code of proprietary software is hidden, it must be more secure.

Open-Source

Security through obscurity isn't enough to protect your system.

Many eyes on the code doesn't guarantee timely review or patch updates.

Both proprietary and open-source software have security vulnerabilities.

Third-Party

Many organizations outsource software development projects to third party suppliers, who will:

Customize or tailor an existing commercial software

Develop new software for organization's business needs

Information source: https://www.veracode.com/sites/default/files/pdf/resources/guides/best-practices-third-party-software-security-veracode-guide.pdf
and
https://apps.dtic.mil/dtic/tr/fulltext/u2/a495389.pdf

Third-Party

Risks

- Intellectual property rights dispute
- Inadequate software specification resulting in increased scope and costs
- Contract dispute which could adversely affect supplier delivery
- Poor security practices resulting in software vulnerabilities
- Lack of ongoing support

Best practices

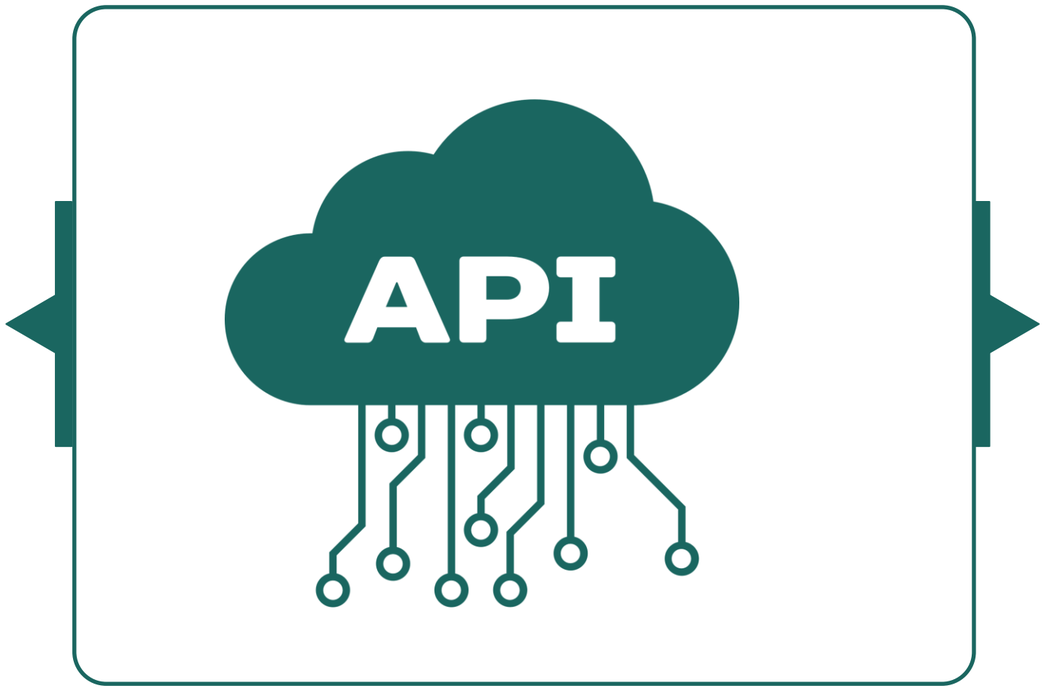
- Develop a third-party security policy
- Take a risk-based approach to application security (threat modeling)
- Verify policies and security practices followed by the vendor
- Establish security metrics and SLA
- Conduct independent application security testing

Define and Apply Secure Coding Guidelines and Standards

Security of Application Programming Interfaces (APIs)

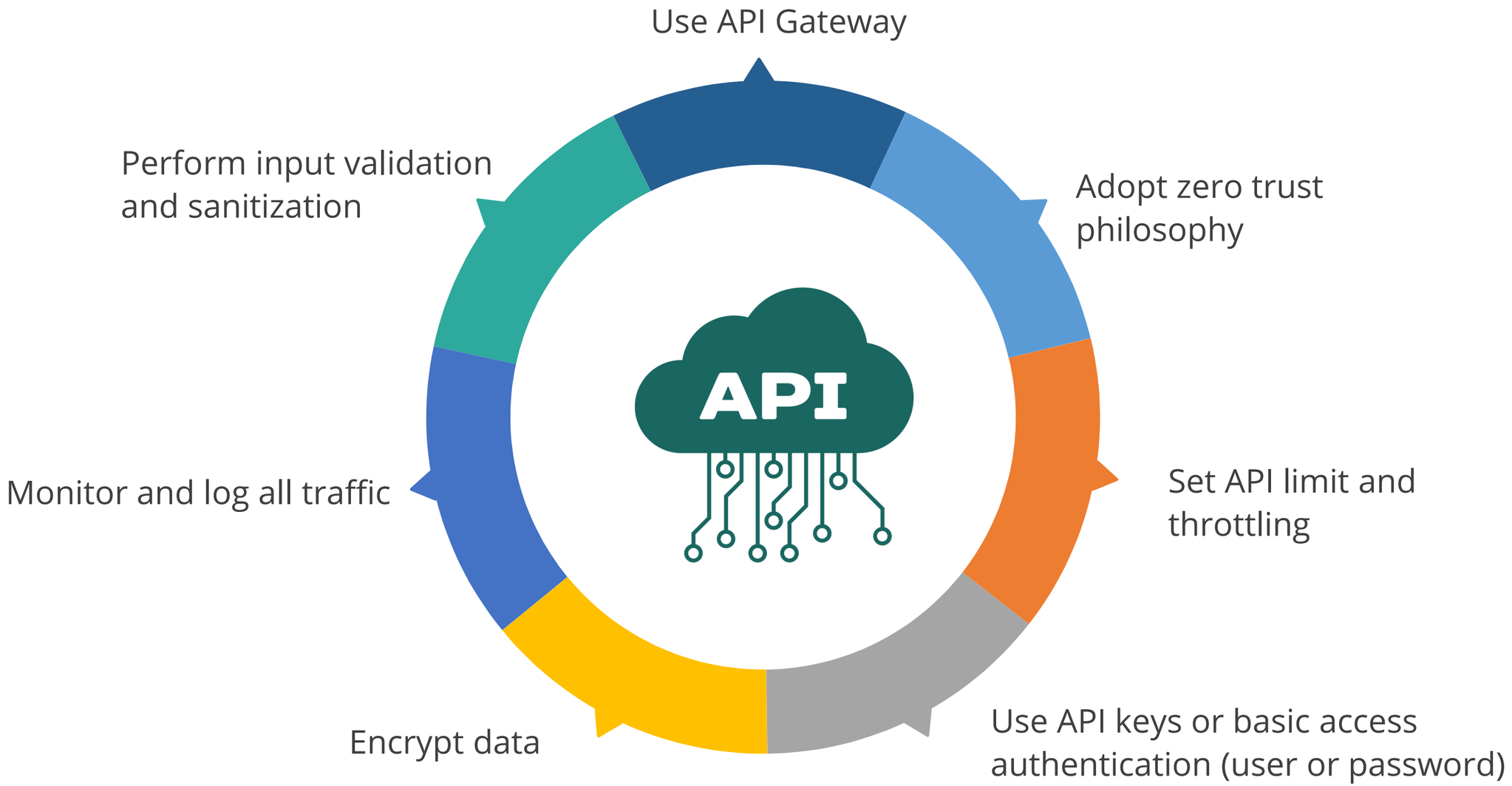
Application Programming Interface (API) is an interface that enables transfer of data between two or more applications.

Simple Object Access Protocol (SOAP):
A protocol and standard for exchanging information between web services in a structured format



Representational State Transfer (REST):
A software architecture style consisting of guidelines and best practices for creating scalable web services

Security of Application Programming Interfaces (APIs): Best Practices



API Formats

Simple Object Access Protocol (SOAP)

- XML based message protocol
- Uses SOAP envelope and then HTTP (FTP/SMTP) to transfer the data
- Only supports XML format
- Slower performance, scalability can be complex and caching is not possible
- Used where REST is not possible, provides WS-* features

Representational State Transfer (REST)

- An architectural style protocol
- Uses only simple hypertext transfer protocol (HTTP)
- Supports many different data formats like JavaScript Object Notation (JSON), eXtensible Markup Language (XML), and Yet Another Multicolumn Layout (YAML)
- Performance and scalability are good and uses caching
- Widely used

OWASP Secure Coding Practices

Input Validation

Output Encoding

Authentication and Password Management

Session Management

Access Control

Cryptographic Practices

Memory Management

General Coding Practices

Data Protection

Communication Security

System Configuration

Database Security

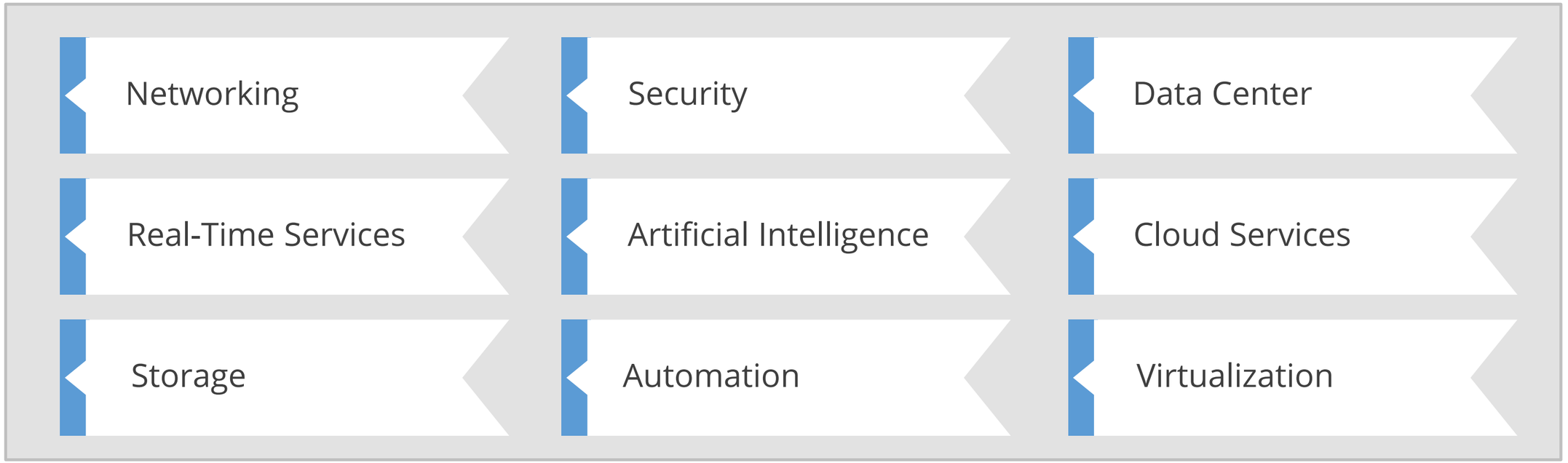
File Management

Information source: https://owasp.org/www-pdf-archive/OWASP_SCP_Quick_Reference_Guide_v2.pdf

Software-Defined Security

Software-defined security (SDS) is a type of security model in which the information security in a computing environment is implemented, controlled, and managed by a security software.

Software Defined Everything (SDx/SDE):



Information sourcehttps://www.gartner.com/smarterwithgartner/securing-the-next-generation-data-center-with-software-defined-security/

Software-Defined Security: Benefits

Enables automated provisioning and orchestration of security controls via policy

Removes human errors via higher levels of automation

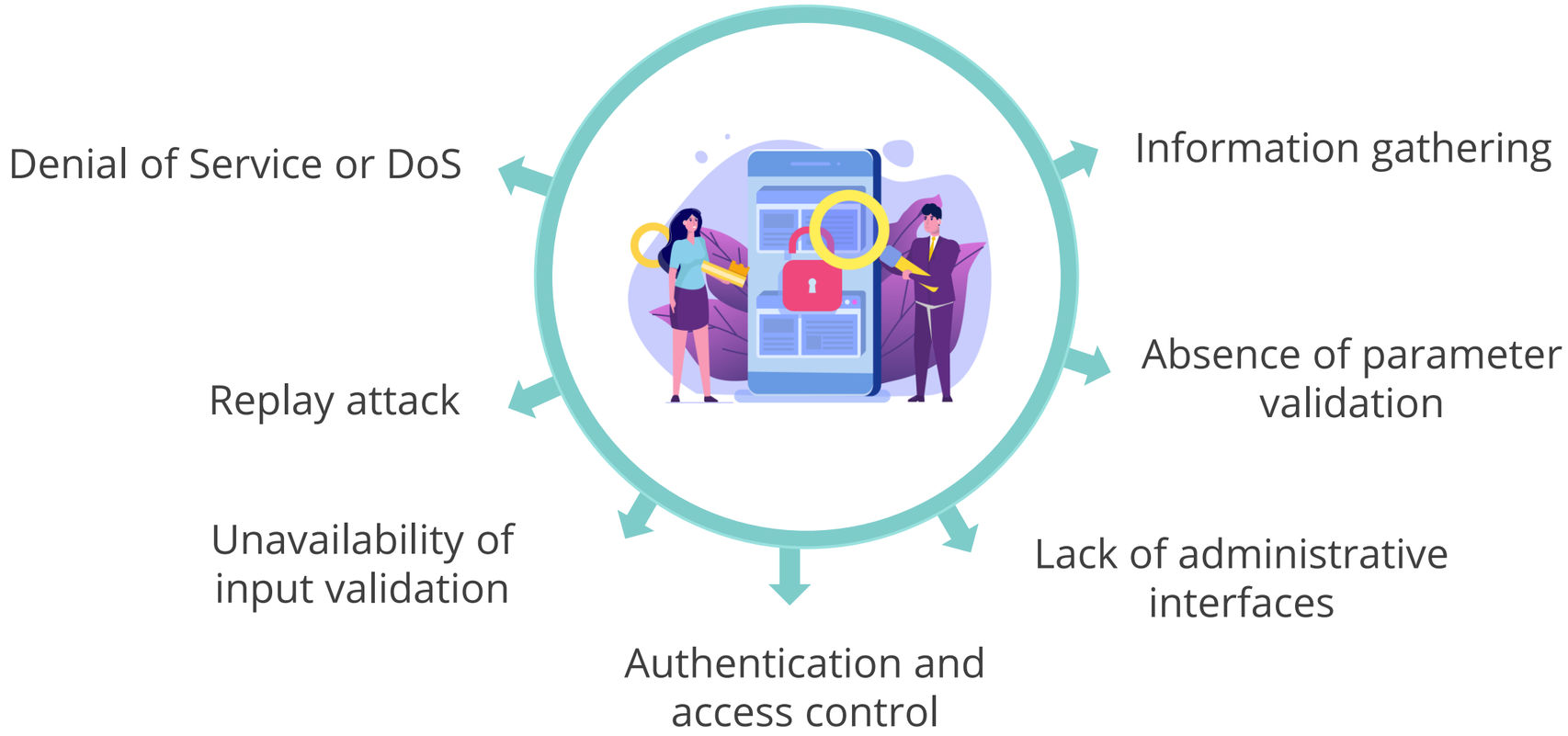
Enables security to protect workload and information regardless of location

Helps limit the scope of data security breaches through segmentation

Enables security to protect dynamic cloud-based workloads

Web Application Environment: Threats and Vulnerabilities

The following are the common types of threats and vulnerabilities of Web Application Environments:



Web Application Environment Security: Specific Protection

Specific protections for Web applications are:

- A particular assurance sign-off process for web servers
- Hardening the operating system used on such servers by:
  - Removing default configurations and accounts
  - Configuring permissions and privileges correctly and
  - Keeping up to date with vendor patches.
- Extending Web and network vulnerability scans prior to deployment

Web Application Environment Security: Specific Protection

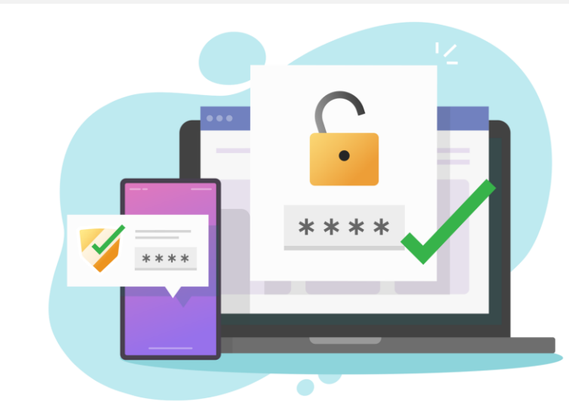
- Passively assessing
  - Intrusion detection system (IDS) and
  - Advanced intrusion prevention system (IPS) technology
- Using application proxy firewalls
- Disabling any unnecessary documentation and libraries

Web Application Environment Security: Administrative Interface Protection

- Administrative interface protection restricts access to authorized hosts or networks and then use strong (possibly multifactor) user authentication.
- They ensure the security of the credentials.
- It uses account lockout and extended logging and audit and protect all authentication traffic with encryption.

Web Application Environment Security: Input Validation

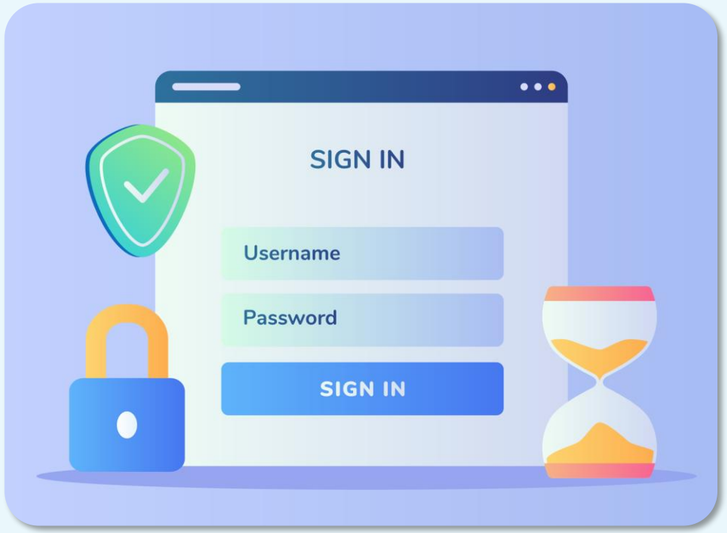
- Input validation ensures that the proxies can deal with problems of:
  - Buffer overflows
  - Authentication issues
  - Scripting
  - Submission of commands to the underlying platform and encoding issues
  - URL encoding and translation



Web Application Environment Security: Sessions Protection

The sessions or periods of apparent attachment to the server are controlled by other technologies, such as cookies or URL data, which must be both protected and validated.

- If using cookies, always encrypt them.
- Do not use sequential, calculable, or predictable cookies, session numbers, or URL data for these purposes.
- Use random and unique indicators.



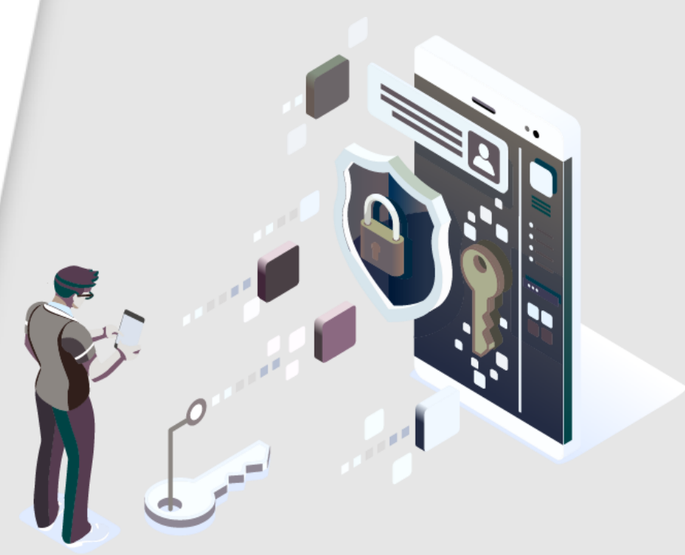
Web Application Environment Security: Web Applications Protection

To make sure the web application is secure, the following methods need to be followed:

- Validate all input and output
- Use defense in depth
- Fail secure (closed)
- Use secure network design
- Make your application or system simple

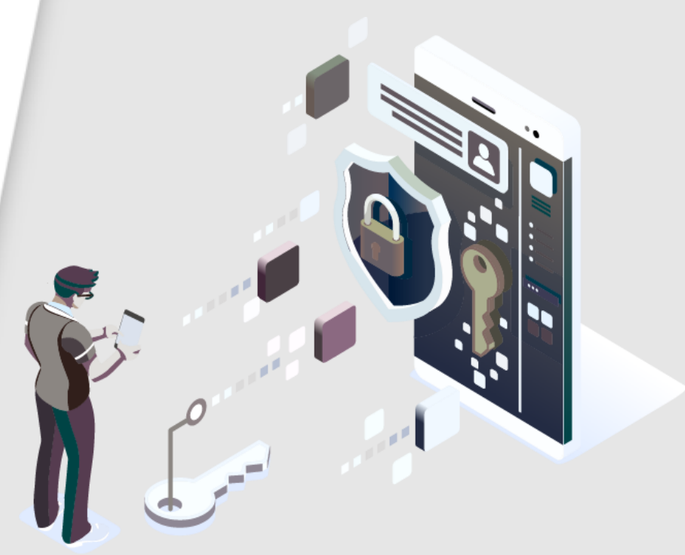
Key Takeaways

- System environments include distributed environment, client-server systems, local environment, distributed data processing (DDP), agents, and applets.
- The various phases of SDLC are prepare a security plan, development or acquisition, implementation, operation or maintenance, and disposal.
- Object-oriented programming uses an object metaphor to design and write computer programs.



Key Takeaways

- A database is a structured collection of related data. The various types are relational model hierarchical model, network model, distributed model, and object-oriented model.
- A data warehouse is a storage facility comprising data from several databases.
- The ten best practices introduced by $(ISC)^2$ can help fulfill the mission of building hack-resilient software.



This concludes **Software Development Security**.

Thank you