Implementation

Implementation refers to the process of translating a design into a working system. It involves coding, testing, and integrating different software components to create a functional product.

Implementation issues

Focus here is not on programming, although this is obviously important, but on other implementation issues that are often not covered in programming texts:

Reuse

Utilizing existing software components to build new systems, reducing development time and costs.

Reuse Levels

- **Abstraction level:** Applying design concepts from successful projects without directly reusing software.
- **Object level:** Directly using pre-built objects from libraries.
- **Component level:** Reuse of collections of objects and object classes in application systems to construct application parts.
- **System level:** Employing complete, pre-developed software systems.

Reuse Cost

- **Search and Assessment Costs:** Time spent finding suitable software and evaluating its fit for the project.
- **Purchase Costs:** Buying reusable software, which can be high for large off-the-shelf systems.
- **Adaptation and Configuration Costs**: Modifying reusable components to meet project requirements.
- **Integration Costs:** Combining reusable software elements with each other and with new code.

Configuration Management

Managing changes in a software project to ensure smooth integration and efficient development.

Activities

- **Version management:** Tracking different versions of software components.
- **System integration:** Defining how different versions work together to build a system.
- **Problem tracking:** Identifying and resolving bugs and issues.

Example: Using Git to manage different development stages.

Host-Target Development

Developing software on one computer (host) and deploying it on another (target).
Example: Developing a program on a Windows PC for execution on a Linux server.

Open Source Development

Making source code publicly available for anyone to use and improve.
Example: Linux operating system, developed by a global community.

Open Source Licensing Types

- **GPL (General Public License):** Requires derivative works to also be open-source.
- **LGPL (Lesser GPL):** Allows using open-source code without making the entire project open-source.
- **BSD (Berkeley Software Distribution) License:** Permits using and modifying the code without sharing changes

  .

Example: A company using GPL licensed components must also make their application open source.

7.9: When code is integrated into a larger system, problems may surface. Explain how configuration management can be useful when handling such problems.

When integrating code into a larger system, various problems can arise, such as compatibility issues, conflicts, or bugs. Configuration management can help handle these problems in several ways:

1. **Version Control:** Keeps track of all changes made to the codebase. If a problem arises, it's easy to identify and revert to a previous, stable version.
2. **Branching and Merging:** Allows developers to work on separate branches and merge changes systematically. This helps isolate new features or fixes and reduces conflicts during integration.
3. **Build Management:** Automates the process of building the system, ensuring that all parts are correctly compiled and linked. This helps catch integration issues early on.
4. **Environment Consistency:** Ensures that the software is built and tested in consistent environments, reducing the chances of environment-specific bugs.
5. **Change Tracking:** Documents all changes, including who made them and why. This helps trace the source of problems and understand their context.
6. **Automated Testing:** Integrates automated tests to verify that new changes do not break existing functionality. This quickly identifies problems introduced during integration.

In summary, configuration management provides tools and practices to systematically handle code changes, maintain consistency, and quickly address integration issues, ensuring smoother software implementation.