UTT
UNIVERSIDAD TECNOLÓGICA DE TIJUANA
GOBIERNO DE BAJA CALIFORNIA

TEMA:

Framework Selection

PRESENTADO POR:

Paredes Nevarez Alexis Omar

GRUPO:

10B

MATERIA:

Software Development Process Management

PROFESOR:

Ray Brunett Parra Galaviz

FECHA:

07/10/2025

Effective versioning strategies are essential in software development to manage and communicate changes, ensuring clarity for both developers and users. A well-defined versioning system facilitates seamless updates, maintains compatibility, and enhances collaboration.

Common Versioning Strategies:

1. **Semantic Versioning (SemVer):**
    - **Format:** MAJOR.MINOR.PATCH
    - **Usage:**
        - **MAJOR:** Incremented for incompatible API changes.
        - **MINOR:** Incremented for backward-compatible functionality additions.
        - **PATCH:** Incremented for backward-compatible bug fixes.
    - **Example:** Version 2.1.3 indicates the second major release, with one minor update and three patch fixes.
    - **Benefits:** Provides clear expectations about the impact of changes, aiding in dependency management.
2. **Calendar Versioning (CalVer):**
    - **Format:** YYYY.MM.DD or variations like YYYY.MINOR
    - **Usage:**
        - **YYYY:** Year of release.
        - **MM/DD:** Month and/or day of release.
        - **MINOR:** Incremented for additional releases within the same year.
    - **Example:** Version 2023.4 denotes the fourth release in the year 2023.
    - **Benefits:** Emphasizes the release date, useful for time-based releases and scheduling.
3. **Sprint-Based Versioning:**
    - **Format:** SPRINT_NUMBER.PATCH
    - **Usage:**
        - **SPRINT_NUMBER:** Identifier for the development sprint.
        - **PATCH:** Incremented for fixes post-sprint.
    - **Example:** Version M27.1 represents the first patch after the 27th sprint.

    - **Benefits:** Aligns versioning with agile development cycles, providing clarity on feature sets delivered per sprint.

Best Practices for Versioning:

- **Consistency:** Adopt a versioning scheme that aligns with your development and release processes, and apply it uniformly across all releases.
- **Documentation:** Clearly document your versioning policy to ensure all stakeholders understand the significance of version numbers.
- **Automation:** Integrate versioning into your continuous integration/continuous deployment (CI/CD) pipelines to reduce human error and streamline releases.
- **Transparency:** Maintain transparency with users regarding version changes to build trust and set clear expectations.

Selecting the appropriate versioning strategy depends on your project's specific needs, development methodology, and release cadence. A thoughtful approach to versioning enhances communication, reduces integration issues, and contributes to the overall quality and reliability of the software product.