**基本ルール：バージョン管理とロールバック**

**1. バージョン管理の対象**

* **AIモデル:** コアモデル、部門別モデル、タスク別モデルを含むすべてのAIモデル
* **設定ファイル:** システム構成、パラメータ設定、APIキーなどの設定ファイル
* **コード:** AIエージェントの実行コード、関連するライブラリと依存関係
* **データスキーマ:** データベースの構造定義、データ形式
* **UI/UX:** ユーザーインターフェースのデザインと機能

**2. バージョニング規則**

* **セマンティックバージョニング (SemVer):** major.minor.patch 形式を採用する
  + **major:** 後方互換性のない変更
  + **minor:** 後方互換性のある機能追加
  + **patch:** 後方互換性のあるバグ修正
* **プレリリース版:** -alpha、-beta、-rc などのサフィックスを使用する
* **ビルドメタデータ:** +build などのサフィックスを使用する（例：1.2.3+20250320）
* **コミットログ:** 全ての変更に対して明確なコミットメッセージを記述する

**3. バージョン管理システム**

* **Git:** コード、設定ファイル、データスキーマのバージョン管理にGitを使用する
* **MLflow:** AIモデルのバージョン管理、実験管理、デプロイ管理に使用する
* **専用のUIバージョン管理ツール:** UI/UXの変更履歴と管理に専用ツール（例：Figmaのバージョン履歴機能）を利用する

**4. ブランチ戦略**

* **メインブランチ:** 安定版の最新バージョンを保持する (main または master)
* **開発ブランチ:** 最新の開発中のコードを統合する (develop)
* **フィーチャーブランチ:** 新機能開発や改善ごとに作成する (feature/機能名)
* **リリースブランチ:** リリース候補版の準備に使用する (release/バージョン番号)
* **ホットフィックスブランチ:** 緊急のバグ修正に使用する (hotfix/バージョン番号)

**5. ロールバック戦略**

* **ロールバックの種類:**
  + **即時ロールバック:** 重大な問題発生時に、直前の安定バージョンに迅速に戻す
  + **段階的ロールバック:** 特定のユーザーグループに対して旧バージョンを維持しつつ、新バージョンの影響を評価する
* **ロールバック手順:**
  + **自動ロールバック:** 事前定義された閾値（例：エラー率の急増）を超えた場合に自動的にロールバックを実行する
  + **手動ロールバック:** 管理者による手動操作で特定のバージョンに戻す
* **ロールバック時の考慮事項:**
  + **データ移行:** ロールバックに伴うデータの整合性を確保する
  + **ユーザーへの通知:** ロールバックの実施と理由をユーザーに通知する
  + **原因分析:** ロールバック後に問題の原因を特定し、再発防止策を講じる

**6. ロールバック手順の詳細**

* **AIモデルのロールバック:** MLflowのバージョン管理機能を利用し、以前のモデルを再デプロイする
* **設定ファイルのロールバック:** Gitの履歴から以前のバージョンをチェックアウトし、再適用する
* **コードのロールバック:** Gitの履歴から以前のコミットに戻す
* **データスキーマのロールバック:** 以前のスキーマ定義を適用し、必要に応じてデータ移行スクリプトを実行する
* **UI/UXのロールバック:** UIバージョン管理ツールから以前のバージョンを復元する

**7. テストと検証**

* **自動テスト:** 新バージョンリリース前に、単体テスト、結合テスト、回帰テストを自動的に実行する
* **手動テスト:** 重要機能やUIの変更に対して、人間のテスターによる手動テストを実施する
* **カナリアリリース:** 新バージョンを一部のユーザーに限定公開し、問題がないことを確認してから全体に展開する

**8. リリース管理**

* **リリースサイクル:** 定期的なリリースサイクル（例：2週間ごと）を定める
* **リリースノート:** 各リリースの変更内容、既知の問題、アップグレード手順などを記載したリリースノートを作成し、公開する
* **緊急リリース:** 重大なセキュリティ問題やクリティカルなバグ修正が必要な場合は、通常のリリースサイクル外に緊急リリースを実施する

**9. 監査ログ**

* **バージョン変更履歴:** 全てのバージョン変更、ロールバック操作、リリース履歴を記録する
* **監査ログの保管:** 監査ログは安全な場所に一定期間保管する