Software Product Lines

Exercise 12: Evolution and Maintenance

ISF (TU Braunschweig) January 2026

1. **Maintenance, Evolution, and Co-Evolution in Product Lines**

   (a) Explain the difference between maintenance and evolution in general, as well as in the context of software product lines. Also, explain the role and importance of co-evolution in software product lines. Use concrete examples to illustrate your answer.

   (b) Classify the following examples based on their maintenance type. Justify your answers.

      i. The feature `OfflineMode` allows users to access content without an internet connection. After deployment, a crash is reported when users try to sync data upon reconnecting. The bug is fixed in a patch release.

      ii. The application is being migrated from Android 12 to Android 14. Some APIs used for notifications have changed, requiring adjustments in the source code to ensure compatibility.

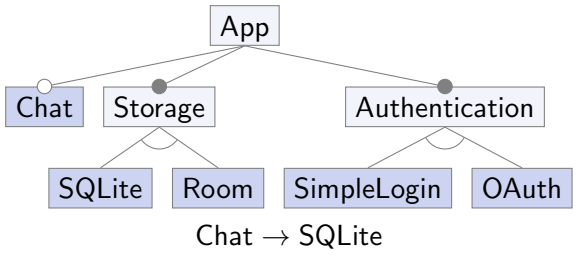
      iii. Users have complained that the application takes too long to start. The development team optimizes the loading process by reducing startup dependencies and lazy-loading non-critical modules.

      iv. To proactively detect potential memory issues, the team adds a logging and diagnostic module that tracks performance degradation over time, even though no crash has been observed yet.

2. **Evolution of Feature Models**

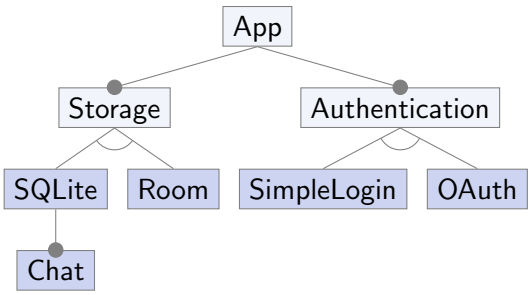
   Consider the following initial feature model:

Initial Feature Model

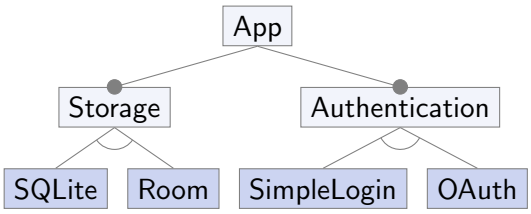


   (a) Starting from this initial feature model, the following changes are applied step by step. For each of the following steps, classify the change compared to the previous step as a *refactoring*, *specialization*, *generalization*, or *arbitrary edit*. Justify your answer.

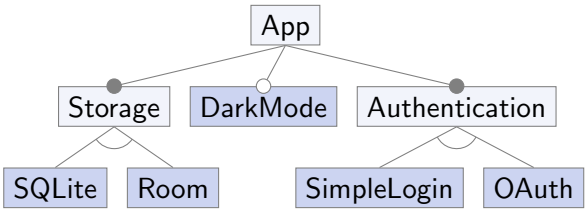
State After Change 1



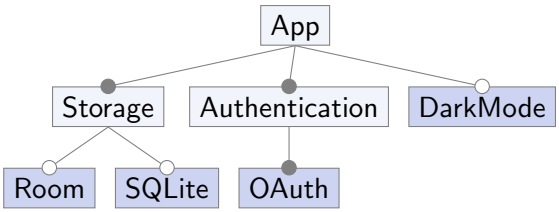
State After Change 2



State After Change 3



State After Change 4



OAuth → Room

(b) In general, how can you compute the classification of the four edit types using a SAT solver?

3. **Code Smells and Refactoring**

(a) What is a code smell? Explain the term and give at least three examples of code smells. Also research and explain the term *variability smell* in the context of software product lines and give at least three examples.

(b) What is a refactoring? What are the goals of refactorings, and why are they sometimes necessary? Which kinds of refactorings of source code do you know? Explain the refactorings with different examples.

4. **Reengineering Variants into a Software Product Line**

You are a software engineer at an educational tech company. Your team is tasked with merging four independently developed school apps into a unified software product line.

All four apps share the following common base features:

- User management (login, account settings)
- Basic content delivery (course materials, videos)
- Student progress tracking

However, each app was individually extended to meet specific customer needs:

- **App A**: Android only. Uses `SQLite` for local storage and supports offline access.
- **App B**: Web-based. Adds a real-time chat system but does not support offline mode.
- **App C**: Android and iOS. Uses the `Room` database. Provides parental control but does not support chat.
- **App D**: Android only. Supports both chat and offline mode, but uses a hardcoded login mechanism.

Each of these apps originated from the same codebase, but has been modified separately over time.

(a) Identify and describe one concrete reengineering task for each of the following categories: reverse engineering, refactoring, and forward engineering. In total, you should describe three tasks (one from each category), that are required to transform the existing variants into a unified software product line.

(b) What product-line adoption strategy does this scenario represent? Justify your answer.