Strangler Fig Pattern

The **Strangler Fig Pattern** is a software design pattern used to gradually replace or refactor an existing legacy system with a new one, typically in a microservices architecture. The idea is inspired by the way a strangler fig plant grows around a tree, eventually replacing it.

Key Concepts of Strangler Fig Pattern:

1. **Incremental Migration**: Instead of replacing the legacy system in one big step (which can be risky), the new system is developed and integrated alongside the old one. Over time, parts of the old system are incrementally replaced by the new system.
2. **Parallel Operation**: Both the legacy system and the new system can operate simultaneously. Specific functionalities or services can be routed to either the legacy or the new system depending on where they've been implemented.
3. **Decomposing Monoliths**: In microservice architectures, this pattern is especially useful when refactoring monolithic systems into microservices. By isolating and replacing parts of the monolith with new microservices, the system can gradually evolve without a complete re-write.
4. **Controlled Transition**: The system behaviour is closely monitored during the transition phase to ensure the new system is functioning correctly. If issues arise, it's easier to revert to the legacy system.

5. **Router/Gateway**: Typically, a router or API gateway is used to direct traffic between the legacy and new systems. This ensures that users interact with a single interface, even though the backend is being transformed.

Steps to Implement the Strangler Fig Pattern:

1. **Identify Candidate Services**: Break down the legacy system into discrete components or services. Prioritize which parts of the system should be replaced first.
2. **Create a Proxy or API Gateway**: This will handle incoming requests and direct them either to the legacy system or to the new microservices based on which functionalities have been refactored.
3. **Develop New Microservices**: Gradually implement new services or systems to replace the legacy functionality. These services should be independently deployable and scalable.
4. **Incremental Cutover**: As each piece of functionality is migrated, the proxy redirects traffic to the new system. Over time, more traffic is handled by the new services.
5. **Retire Legacy Code**: Once the legacy functionality is fully migrated and verified, the old system can be decommissioned.

Benefits of the Strangler Fig Pattern:

1. **Reduced Risk**: By migrating incrementally, you reduce the risk of large-scale failures.
2. **Flexibility**: It allows for refactoring and system improvements without freezing or halting existing development.
3. **Continuity**: Users experience minimal disruption as both systems can run in parallel during the migration.
    a. This pattern is frequently used in legacy modernization projects to refactor monolithic applications into microservices, providing a safe and efficient path toward modernization.