Software reuse

2020년 6월 13일 토요일 오전 1:55

5 Application system reuse

애플리케이션 시스템 재사용

• 애플리케이션 시스템 전체를 다른 시스템에 변경 없이 통합하거나 애플리케이션 조직를 개발하여 재사용할 수 있다.

COMPONENT 재사용

• 하위 시스템에서 단일 개체에 이르는 애플리케이션의 구성요소를 재사용할 수 있다.

객체 및 기능 재사용

• 잘 정의된 단일 객체 또는 기능을 구현하는 소프트웨어 구성요소를 재사용할 수 있다.

이점

신뢰도 증가

-작업 시스템에서 시험하고 시험한 재사용 소프트웨어는 새로운 소프트웨어보다 신뢰할 수 있어야 한다. 소프트웨어의 초기 사용은 모든 설계 및 구현 결함을 드러낸다. 그런 다음 고정되어 소프트웨어를 재사용할 때 고장 횟수가 감소한다.

프로세스 위험 감소

-소프트웨어가 존재한다면, '프로젝트 리스크 감소'를 재사용하는 비용에 대한 확실성이 떨어진다. 그렇게 ftw가 존재한다면, 개발 비용보다 그 소프트웨어를 재사용하는 비용에 대한 불확실성이 적다. 이는 사업비 추계의 오차범위를 줄여 사업관리에 중요한 요소다. 하위 시스템과 같은 비교적 큰 소프트웨어 구성요소를 재사용할 때 특히 그렇다.

전문가의 효과적인 사용

-애플리케이션 전문가가 서로 다른 프로젝트에 대해 동일한 작업을 수행하는 대신, 지식을 캡슐화하는 재사용 가능한 소프트웨어를 개발할 수 있다.

Standards compliance

사용자 인터페이스 표준과 같은 일부 표준은 표준 재사용 가능한 구성요소의 집합으로 구현될 수 있다. 예를 들어, 사용자 인터페이스의 메뉴가 재사용 가능한 구성요소를 사용하여 구현되는 경우, 모든 응용프로그램은 사용자에게 동일한 메뉴 형식을 제공한다. 표준 사용자 인터페이스를 사용하면 사용자가 친숙한 인터페이스를 제공할 때 실수를 저지를 가능성이 적기 때문에 신뢰성이 향상된다.

개발의 가속화

시스템을 가능한 한 빨리 출시하는 것이 더 쉽다.

문제점

유지보수비용증가

TOOL SUPPORT가 부족

CASE TOOLSET(자동화도구) 재사용개발에 지원되지 않는다. 통합하는데 어렵다.

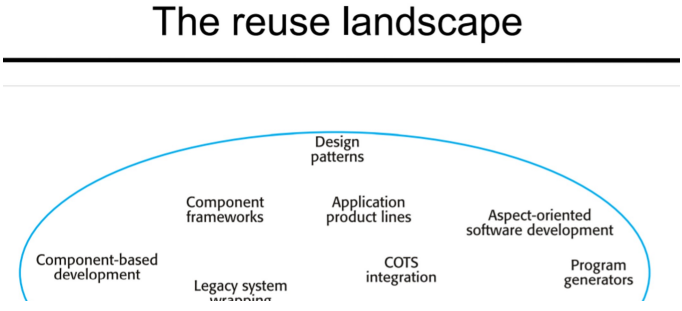
미발견증후군

일부 소프트웨어 엔지니어는 재사용 가능한 구성요소를 개선할 수 있다고 믿기 때문에 구성요소를 다시 쓰는 것을 선호한다. 이것은 부분적으로 신뢰와 관련이 있고, 부분적으로 원본 소프트웨어를 쓰는 것이 다른 사람의 소프트웨어를 재사용하는 것보다 더 어렵다는 사실과 관련이 있다.

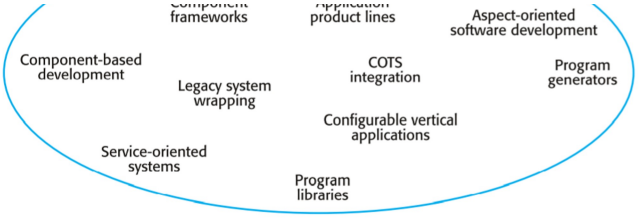
컴포넌트 라이브러리 CLASS화시키는것이힘듬

재사용 가능한 구성요소 라이브러리를 채우고 소프트웨어 개발자가 이 라이브러리를 사용할 수 있도록 하는 것은 비용이 많이 들 수 있다. 소프트웨어 구성요소를 분류, 분류 및 검색하는 현재의 기술은 미숙하다.

재사용 컴포넌트에대한 이해가필요함



류, 분류 및 검색하는 현재의 기술은 미숙하다.



재사용 컴포넌트에대한 이해가필요함
소프트웨어 구성요소는 라이브러리에서 검색되어야 하며, 이해되고, 때로는 n ew 환경에서 작동하도록 조정되어야 한다. 엔지니어는 정상적인 개발 과정의 일부로 구성요소 검색을 일상적으로 포함시키기 전에 라이브러리에서 구성요소를 찾을 수 있는 확신을 가져야 한다.

12~13예시

14 계획요인 재사용
소프트웨어 개발 일정.
예상 소프트웨어 수명.
개발팀의 배경, 기술, 경험.
소프트웨어의 중요성 및 비기능적 요구 사항.
응용 프로그램 도메인.
소프트웨어 실행 플랫폼.

15 Concept reuse
프로그램이나 디자인 컴포넌트를 재사용할 때는 컴포넌트의 원래 개발자가 내린 **디자인 결정**을 따라야 한다.
이것은 재사용의 기회를 제한할 수 있다.
그러나, 보다 추상적인 형태의 재사용은 특정 접근법을 구현 독립적 방법으로 기술한 후 구현이 개발될 때 개념 재사용이다.
개념 재사용을 위한 두 가지 주요 접근법은 다음과 같다.
• 설계 패턴 16
이름,문제설명,솔루션설명,결과
• 생성 프로그래밍.