소프트웨어공학 활용

CASE 개요

한국기술교육대학교
온라인평생교육원

학습내용

- CASE 개요 및 분류
- CASE 구성 및 특성

학습목표

- CASE 도구의 기본 개념을 이해하고 설명할 수 있다.
- CASE 도구의 구성 및 특성을 이해하고 설명할 수 있다.

CASE 개요 및 분류

1 CASE 개요

1 CASE 정의

1 CASE(Compute Aided Software Engineering) 정의 및 개념

소프트웨어 개발 과정 일부 또는 전체를 자동화하기 위한 도구

- 계획, 요구분석, 설계, 코딩, 시험 및 유지보수

소프트웨어 품질과 생산성 향상을 지원하는 시스템 또는 도구

소프트웨어 개발 도구와 방법론이 결합한 것

- 정형화된 구조 및 방법(메커니즘) + 소프트웨어 개발에 적용
- 생산성 향상을 구현하는 공학 기법

CASE 개요 및 분류

1 CASE 개요

1 CASE 정의

1 CASE(Compute Aided Software Engineering) 정의 및 개념

자동화 도구들 지원

- 소프트웨어 개발이 모든 단계에 걸쳐 일관된 방법론 제공

↓

개발자들이 지원 도구 사용

↓

소프트웨어 개발의 표준화 지향

↓

자동화의 이점을 얻을 수 있음

2 등장 배경

- 소프트웨어 위기 해결 (생산성 향상)
- 소프트웨어 개발 자동화
- 소프트웨어 복잡도 증가 극복

CASE 개요 및 분류

1 CASE 개요

1 CASE 정의

3 효과

표준화, 표현의 효율성	표준화된 환경구축 및 문서과정의 자동화, 표현성 확보
재사용성, 안전성	소프트웨어 재사용성 확보 및 안정된 소프트웨어 품질 확보 가능
신속성	전 과정의 신속성, 통합성 제공

2 CASE 목표

1. 소프트웨어 품질 향상
2. SDLC(Software Development Life Cycle) 자동화
3. 개발, 유지보수 생산성 향상

CASE 개요 및 분류

1 CASE 개요

3 CASE의 주요 기능

1. 소프트웨어 생명주기 전 단계의 연결
2. 다양한 소프트웨어 개발 모형 지원
3. 그래픽 지원 등

CASE 개요 및 분류

2 CASE 분류

1 CASE 도구 분류

Upper CASE | Middle CASE | Lower CASE | Integrated CASE

개발 주기 단계 : 계획

- 계획 수립, 요구분석, 소프트웨어 기본설계 작업을 지원하고 이를 Diagram으로 표현
- 여러 가지 명세와 문서를 작성하는 데 사용됨
- 도구 : SREM, PSL/PSA, SERA, FOUNDATION

Upper CASE | Middle CASE | Lower CASE | Integrated CASE

개발 주기 단계 : 분석, 설계

- 생명 주기 상세설계 작업 지원
- 화면, 리포트 등의 작성 지원

CASE 개요 및 분류

2 CASE 분류

1 CASE 도구 분류

Upper CASE | Middle CASE | **Lower CASE** | Integrated CASE

개발 주기 단계 : 구현

- 개발, 시험, 유지보수 작업 지원
- Source Code(원시 코드)와 시스템 명세서 획득
- 도구 : 구문 중심 편집기, 코드 생성기

Upper CASE | Middle CASE | Lower CASE | **Integrated CASE**

개발 주기 단계 : 계획, 분석, 설계, 구현

- 분석, 설계, 구현, 테스트, 유지보수 등 모든 단계를 지원
- Upper, Middle, Lower Case를 통합한 것
- 도구 : Rational ROSE, COOL 등이 국내에서 사용

CASE 개요 및 분류

2 CASE 분류

2 CASE 기능

체계적인 시스템 기술	분석 기능	수명 주기 통합 및 지원
• 그래픽, 구분, 테이블 • 오류 추적, 의사소통 원활	• 완전성 • 일관성 • 비교 평가	• 단계 전환 자동, 단계별 자료 자동으로 공유 • 자료 사전, 자료 상관관계 및 타당성 검사 • 다양한 정보검색 및 출력

CASE 개요 및 분류

2 CASE 분류

3 CASE 도구 발전 배경

1970년대
- 고급 언어 컴파일러
- 구조적 프로그래밍

1980년대
- 4GL 등장
- ISP, 프로토타이핑, 통합 프로젝트 관리 환경, 정보 공학

→ 4GL(비절차적, 그래픽컬한 개발 언어로 사용자도 쉽게 개발할 수 있는 개념)

1990년대
- 역공학, 지능적 CASE
- 객체 지향방법, 인터넷 환경

→ CASE 도구를 이용하여 기존 시스템에 대한 사양서와 설계서 등의 문서를 도출하는 작업

1990년대 말
- 통합 CASE 및 방법

CASE 개요 및 분류

2 CASE 분류

4 CASE 도구 유형

다이어그램 도구

- 시스템 컴포넌트(그래픽 형태)
- 데이터 및 소프트웨어 구성요소 간 제어 흐름과 시스템 구조 표현

프로세스 모델링 도구

- 소프트웨어 프로세스 모델을 생성하는 모델링 도구

프로젝트 관리 도구

- 프로젝트 계획, 비용, 스케줄링 자원 관리 도구

문서 도구

- 사용자와 최종 사용자에게 문서를 제공하는 도구

분석 도구

- 자동으로 요구사항을 수집
- 다이어그램, 데이터 중복 또는 잘못된 부분 확인에 도움을 줌

디자인 도구

- 작은 모듈로 세분화 될 수 있게 설계하는 도구

CASE 개요 및 분류

2 CASE 분류

4 CASE 도구 유형

구성 관리 도구

- 소프트웨어 구성을 관리

제어 도구 변경

- 추적 파일 관리, 코드 관리 등을 자동화

프로그래밍 도구

- 통합개발 환경과 같은 프로그래밍 환경 구성 도구

프로토타이핑 도구

- 제품의 초기 모양과 느낌을 제공

웹 개발 도구

- 웹 개발 도구, 개발되는 과정을 실시간으로 제공

품질보증도구,유지보수도구

- 소프트웨어의 품질 및 유지 보수를 위한 도구

CASE 구성 및 특성

1 CASE 구성

1 기능에 따른 분류

1 분류 방법

- 개발 일정과 자원을 관리하는 프로젝트 관리용
- 실제 산출물을 생산하는 개발지원용
- 정보자원의 통합 및 변경사항을 지원하는 정보저장소(Repository) 관리용
- 프로젝트 관리 및 개발자원을 통합한 체계의 방법론 지원용

2 프로젝트 관리 도구

- 업무 시스템 계획 도구(Business System Planning Tool)
- 프로젝트 관리 도구(Project Management Tool)
  - 프로젝트 계획 도구, 요구사항 추적 도구, 매트릭스와 관리 도구

3 개발 지원 도구

- 분석 및 설계 도구
- 프로그래밍 도구
- 지원도구

CASE 구성 및 특성

1 CASE 구성

1 기능에 따른 분류

4 정보저장소 도구(Database or Repository)

1. 개방된 정보저장소로서 다른 정보저장소와 데이터를 주고받음
2. 객체나 프로젝트 간의 변경 관리 기능을 가져야 함
3. 보안 관리 기능을 가져야 함
4. 필요한 형태의 유틸리티를 사용자가 개발, 사용할 수 있어야 함

5 통합환경이나 방법론 지원도구

프레임워크 도구	CASE 도구들의 통합 처리 능력 제공

CASE 구성 및 특성

1 CASE 구성

2 지원 범위에 따른 분류

01 CASE Tool

02 CASE Tool Kit

03 CASE Workbench

04 CASE Methodology Companion

3 통합 정도에 따른 분류

01 단위 (Stand-alone) CASE

02 통합 (File-based Integration) CASE

CASE 구성 및 특성

1 CASE 구성

4 시스템 유형별 분류

- 01 메인 프레임 기준의 CASE
- 02 UNIX 기준의 CASE
- 03 PC 기준의 CASE

5 규모와 가격으로 분류

- 01 대규모 CASE
- 02 소규모 CASE

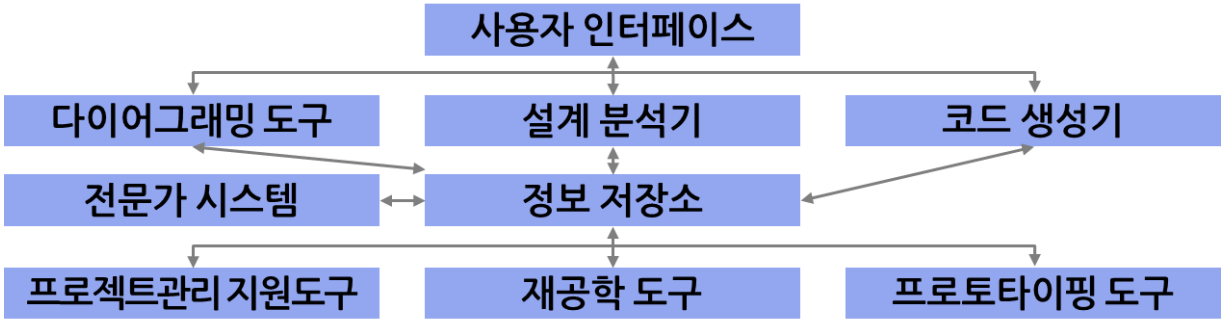
6 CASE 도구 시스템의 구성

Diagram 작성 도구	소프트웨어 명세서 정의, 설계 과정의 표현
설계 분석기	설계 명세서의 정확성, 일치성, 모호성에 대한 검사

CASE 구성 및 특성

1 CASE 구성

6 CASE 도구 시스템의 구성



코드 생성기	명세서로부터 프로그래밍 언어로 된 모듈의 코드 생성
Repository	CASE 도구의 중심, SDLC 동안 정보를 저장
프로젝트 관리 지원도구	프로젝트 관리자를 위한 정보 지원도구
재공학 도구	기존 시스템의 설계 명세서 작성 지원
Prototyping 도구	초기 UI 작성 지원

CASE 구성 및 특성

2 CASE 특성 및 도입

1 CASE 장점과 단점

장점

- SDLC 전 단계의 개발 표준화 작업
- 개발 시 사용자 참여 증대
- 소프트웨어 재사용 증대
- 유지 보수성 향상
- 문서화 용이

단점

- 고가의 비용
- 툴 사용이 어려움
- 완벽한 코딩을 생성하지 못함

2 CASE 도구의 전망 및 발전 방향

1. CASE Tool의 보편화, 이해 및 인식의 전환 필요
2. 일반 사용자 컴퓨팅을 위한 표준화된 방법론에 대한 정립 필요
3. 지능형 CASE
4. 통합 CASE

CASE 구성 및 특성

2 CASE 특성 및 도입

3 도입 필요사항

1 도입 시 준비 사항

- 개발 방법론의 선택, 교육, 사전 협의 및 관리 체계 구축 및 기반 조성 필요
- 경영진, 관리자, 개발자 전원의 올바른 사용에 대한 이해와 준비 필요

2 CASE 도구 선정 평가 항목

비용	호환성	통합성
사용 용이성	숙달기간	유지보수성 등

CASE 구성 및 특성

2 CASE 특성 및 도입

4 CASE 도구 도입의 성패

성공요인

- CASE 대한 정확한 이해
- 교육, 컨설팅 및 자체개발의 필요성 인식과 노력
- 프로젝트와 사용자 능력 및 개발환경에 맞는 도구의 선정

실패요인

- CASE에 대한 환상적 도구라는 잘못된 인식
- 사용자의 능력을 무시한 도입
- 일관성 없는 개발방법론 및 도구 사용 경험에서 오는 지나친 기대
- 많은 기능을 가진 도구의 선호
- 공급자의 기술지원 능력과 사무지원제도 미비

학습정리

1. CASE 개요 및 분류

- Computer Aided Software Engineering의 약자로 소프트웨어 개발 과정에서 사용되는 요구분석, 설계, 구현, 검사 및 디버깅 전체 또는 일부를 컴퓨터와 소프트웨어 도구를 사용하여 자동화하는 것
- 소프트웨어의 위기를 해결하기 위하여 CASE 도구를 이용하여 생산성을 향상함
- 대형 프로젝트를 수행하면서 소프트웨어의 복잡도가 증가하는 것을 극복할 수 있음
- CASE 도구를 이용하면 표준화, 재사용성, 안전성, 표현의 효율성, 신속성의 효과를 볼 수 있음
- 소프트웨어의 품질 향상, 소프트웨어 개발 주기의 자동화, 유지보수 생산성을 향상을 목표로 함
- Upper CASE, Middle CASE, Lower CASE, Integrated CASE 도구로 분류
- Case 도구의 대표적인 유형으로는 다이어그램 도구, 프로세스 모델링 도구, 프로젝트 관리/문서 분석 도구, 디자인 도구, 구성 관리 도구, 구현 및 유지보수 도구가 있음

학습정리

2. CASE 구성 및 특성

- CASE 분류하는 방법은 개발일정과 자원을 관리하는 프로젝트 관리용, 실제 산출물을 생산하는 개발지원용, 정보자원의 통합 및 변경 사항을 지원하는 정보저장소 관리용, 프로젝트 관리 및 개발자원을 통합한 체계의 방법론 지원용으로 분류함
- 지원 범위에 따라 CASE Tool, Tool Kit, Workbench, Methodology Companion으로 분류함
- 장점 : SDLC 전 단계의 개발 표준화 작업이 가능하며 개발 시 적극적인 사용의 참여를 유도하고, 유지 보수성이 증가함
- 단점 : CASE 도구는 고가의 비용이 들며, 도구 사용이 어려우며, 자동화를 통한 완벽한 코딩을 생성하기가 어려움