강의노트

소프트웨어 진화

❖ 학습안내

이번 시간의 학습내용과 학습목표를 확인해보세요.

▣ 학습내용

- 프로그램 진화역학
- 소프트웨어 유지보수
- 정보시스템 외주 개발

▣ 학습목표

- 소프트웨어 진화프로세스를 고려하여 소프트웨어를 구현할 수 있다.
- 소프트웨어 유지보수 단계와 활동을 설명할 수 있다.
- 정보시스템 외주 개발에 대하여 설명할 수 있다.

❖ 학습내용

[1] 프로그램 진화역학

1. 프로그램 진화역학

◈ 프로그램 진화역학 개요

- 정보시스템은 기업의 요구조건과 다양한 환경변화 등에 따라 계속적으로 변화가 이루어져야 함
- 시간이 지나면서 변경되고 개선되어지면서 성능이나 기능이 개선되어짐
- 이와 같이 소프트웨어는 진화되어야 함

◈ 프로그램 진화역학의 정의

프로그램 진화역학

- 시스템 변경을 연구하는 것

◈ 시스템 변경에 대해 제안된 법칙

시스템 유지보수는 필연적인 프로세스이다.

- 환경이 변화하고 요구조건이 늘어나기 때문에 항상 시스템은 수정되어야 하며 이 과정은 반복됨

시스템이 변경됨에 따라 시스템의 구조가 나빠진다.

- 이러한 현상이 발생하는 것을 피할 수 있는 방법은 예방차원의 유지보수에 조금 더 투자하는 것임

대형시스템들은 이미 초기단계에 자신에게 설정된 자기의 동력 (변경 정도의 크기)이 정해진다. (라이프 사이클)

- 시스템의 변경작업에는 한계가 있으며, 어느 정도의 변경작업 이후에는 본 시스템을 전면적으로 개편하는 작업을 하여야 함

대부분 대형 프로젝트들이 이미 포화된 상태에서 작업한다.

- 이미 시스템을 구현하는 경우 이미 개발자 수급, 시스템 자원, 기능 등이 한계상황인 경우에서 작업되는 경우가 많은데 이러한 경우를 고려해야 함

❖ 학습내용

[1] 프로그램 진화역학

1. 프로그램 진화역학(계속)

◈ 시스템 변경에 대해 제안된 법칙(계속)

새로운 기능을 추가하여 배포하는 만큼 필연적으로 결함들을 야기한다.

- 당연히 더 많은 기능들을 추가하여 시스템을 보강하였다면, 당연히 추가된 기능들의 잠재적인 오류가능성이 추가됨

소프트웨어가 유지보수 되지 않고 새로운 기능이 추가되면, 소프트웨어 사용자들은 점점 더 불행해 진다.

- 앞의 내용과 유사한 사항으로 항상 기능이 추가 된다는 것은 잠재적인 에러도 동반 될 수 있기 때문에 기능 추가가 사용자 입장에서 반드시 좋은 상황은 아님

2. 리만의 법칙

◈ 리만의 법칙 배경

- Lehman은 1969년에 소프트웨어 유지보수와 시스템 진화에 대해 처음으로 언급
- 20여 년에 걸쳐, 그의 연구는 여덟 가지 “진화 법칙”을 공식화
- 주요한 발견은 유지보수는 진화론적인 개발이며, 시간이 지남에 따라 시스템과 소프트웨어에 어떤 일이 일어나는지 이해하는 것이 유지보수 의사결정에 도움을 준다는 사실을 언급
- 어떤 사람은 정보시스템의 추가적 기능 추가(Extra Input) 또는 제약사항 (Constraint)을 해결하는 작업을 제외하고, 유지보수는 지속적인 개발이라고 함
  왜냐하면 현존하는 대형 소프트웨어는 절대 완전하지 않으며 계속 진화하기 때문임
- 소프트웨어(또는 시스템)가 진화함에 따라 복잡성을 줄이기 위한 특정 조치가 일어나지 않는다면, 소프트웨어는 점점 더 복잡해짐

- 시스템이 환경 속에 설치된 후, 환경이 변화할 수 있고, 따라서 시스템 요구사항이 변경될 수 있음
- 해당 환경에서 계속적으로 유용하려면 시스템은 반드시 변경되어야 함
- 프로그램 진화역학은 시스템의 변경 프로세스를 연구하는 학문임

❖ 학습내용

[1] 프로그램 진화역학

2. 리만의 법칙(계속)

◈ 리만의 법칙

- 진화하는 '대형 소프트웨어'에 적용될 수 있는 몇 가지 '법칙(Laws)'을 제안
- 고전적인 법칙이나 현재까지도 그 원칙이 존중됨

1 지속적인 변화(Continuing Change)

- 계속 변화하지 않으면 소프트웨어의 유용성이 저하됨

2 증가하는 복잡도(Increasing Complexity)

- 변경이 일어날수록 시스템의 구조는 점점 더 복잡해지고 나빠짐
- 구조 개선을 위한 추가 노력이 필요하게 됨

3 거대 프로그램 진화(Large Program Evolution)

- 프로그램 크기, 릴리스 간격, 보고되는 에러의 개수 등은 변경 후에도 거의 일정함

4 조직적 안정성(Organizational Stability)

- 생명주기 동안 개발 생산성은 상당히 일정하며, 시스템 개발에 추가적으로 투입되는 자원과 무관함

5 친숙함의 보존(Conservation of Familiarity)

- 시스템 생명주기 동안, 각 출시 때마다 변화되는 양은 상당히 일정함

6 지속적인 성장(Continuing Growth)

- 사용자를 만족시키기 위해서는 지속적으로 기능이 추가되어야 함

❖ 학습내용

[1] 프로그램 진화역학

2. 리만의 법칙(계속)

◈ 리만의 법칙(계속)

7 감소하는 품질(Declining Quality)

- 운영 환경에 맞게 시스템을 변경시키기 않는다면 품질은 저하될 것임

8 피드백 시스템(Feedback System)

- 큰 폭의 제품 개선을 이끌어내기 위해서는 피드백 시스템을 활용해야 함

◈ 리만의 법칙 적용 가능성

- 리먼의 법칙은 리먼이 2000년대 초 FEAST 프로젝트에서 확인한 법칙이며, 시스템 진화 작업의 특징으로 유지보수를 계획할 때 고려해야 할 것들임
- 대기업에서 개발된 대규모 시스템에는 적용 가능하지만, 다른 종류의 시스템에도 적용될 수 있을지는 확실하지 않음

리만의 법칙이 중·소규모 시스템에도 적용될 수 있는지?

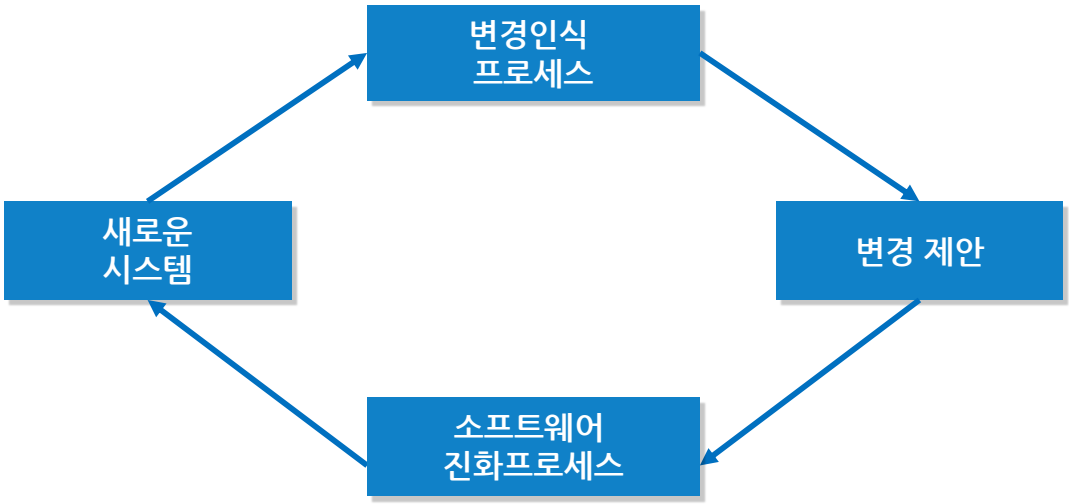
현존 정보시스템은 업무(Business)의 변화가 매우 빠르고 적시성(Time-to-market)이 중요한 요소이기 때문에 수시로 변경뿐 만 아니라 재개발이 이루어지고 있음

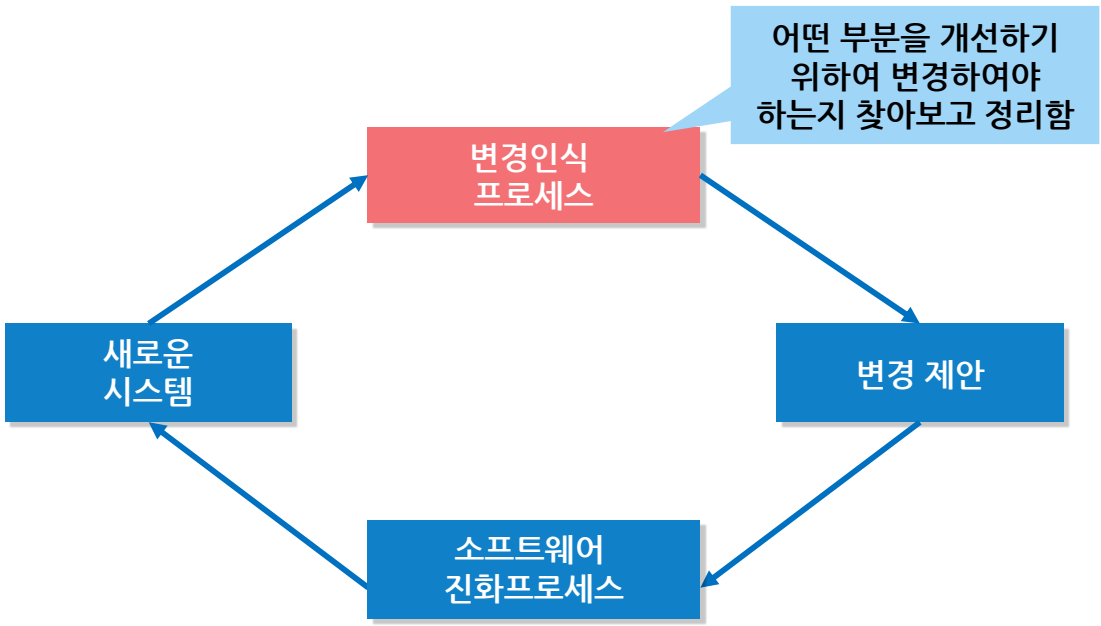
❖ 학습내용

[1] 프로그램 진화역학

3. 진화프로세스

◈ 소프트웨어가 진화되는 과정



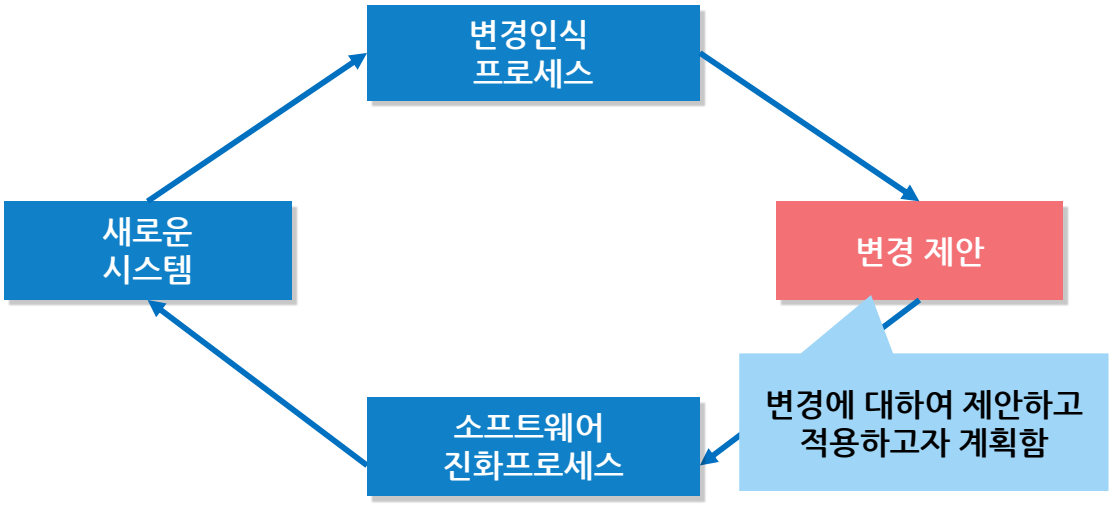


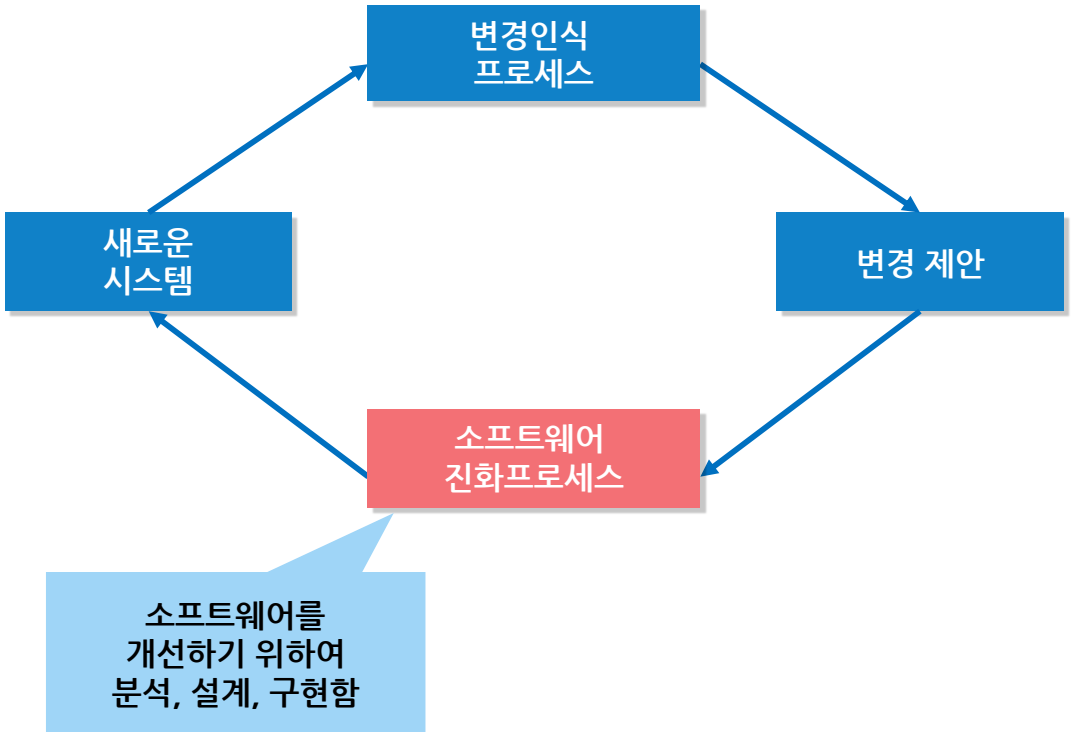
❖ 학습내용

[1] 프로그램 진화역학

3. 진화프로세스(계속)

◈ 소프트웨어가 진화되는 과정(계속)



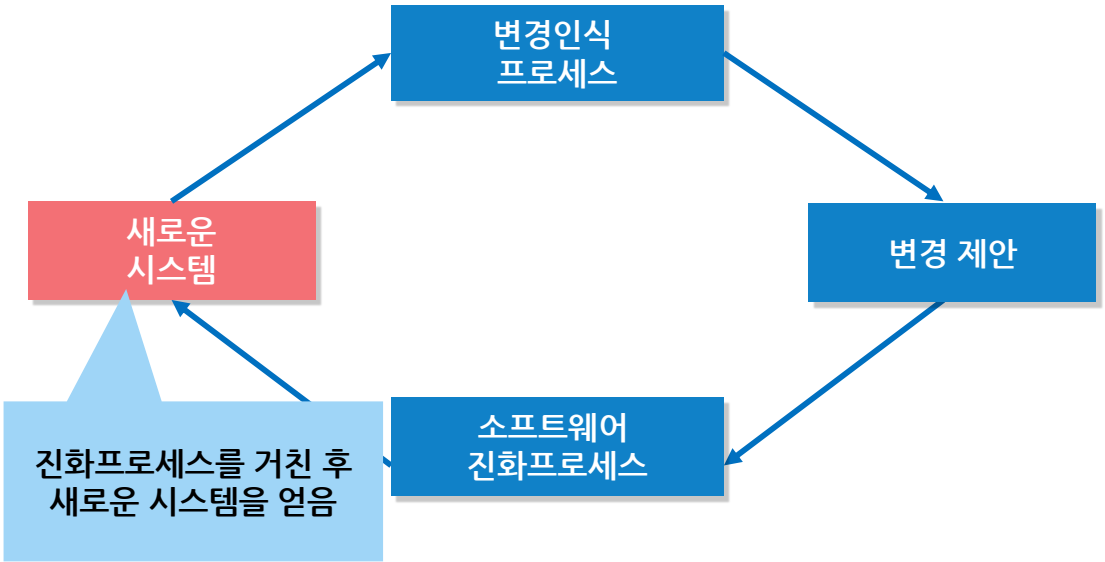


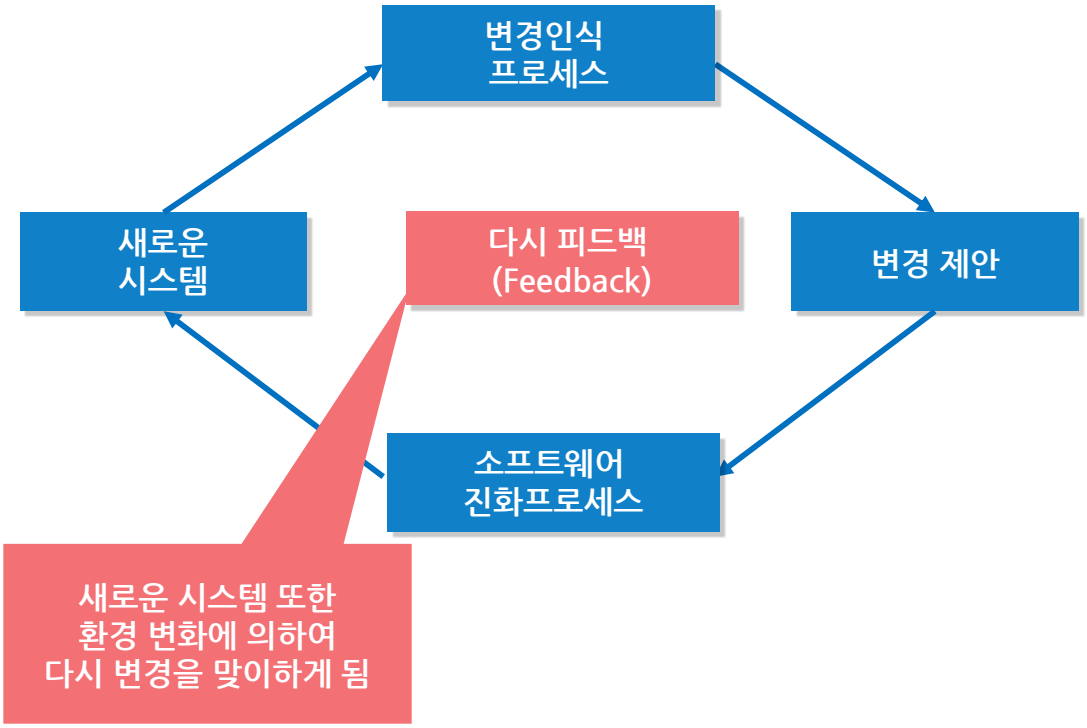
❖ 학습내용

[1] 프로그램 진화역학

3. 진화프로세스(계속)

◈ 소프트웨어가 진화되는 과정(계속)





❖ 학습내용

[2] 소프트웨어 유지보수

1. 유지보수의 정의

◈ 유지보수란?

- SDLC(Software Develop Life Cycle)의 마지막 단계로 소프트웨어의 생명을 연장 시키는 작업
- 소프트웨어 공학 재검토 과정의 각 단계에서 고려
- 오류의 수정, 원래의 요구를 정정, 기능과 수행력을 증진시키는 일련의 작업
- 소프트웨어가 인도된 후 결함의 제거, 성능향상, 변화된 환경에 적응토록 수정
- 소프트웨어 유지보수 및 운영 전담조직이 필요(Maintenance-Bound)
- 개발은 제작중심의 작업이며 유지보수는 운영중심의 작업
- 소프트웨어가 인수되어 설치된 후 일어나는 모든 소프트웨어 공학적 작업

2. 유지보수의 필요성

◈ 필요성

- 유지보수 비용이 전체 비용의 70~80%를 차지
- 소프트웨어 인력이 신규 프로젝트보다 유지보수 업무에 투입되는 낭비 요소 발생
- 유지보수의 비효율성으로 인해 패키지 소프트웨어의 도입 확산
- 프로젝트보다 기존 소프트웨어 개선에 더 많은 인력과 비용 소요
- 소프트웨어기능의 복잡화에 따른 난해함으로 문서화 등의 관리업무가 증가
- 개발은 1-2년 정도지만 유지보수는 5년 또는 10년 정도로 장기
- 용역개발보다 패키지의 선택이 확산됨에 따라 유지보수 부문이 증가 예상
- 소프트웨어가 보다 좋은 성능을 가지고 진화되기 위해서는 소프트웨어가 잘 유지보수 될 수 있는 기반 아래에서 이루어져야 함
- 유지보수는 진화과정에 있어서 반드시 고려되어야 하는 활동

❖ 학습내용

[2] 소프트웨어 유지보수

2. 유지보수의 필요성(계속)

◈ 유지보수의 목표

- 소프트웨어의 성능 개선
- 소프트웨어의 하자 보수
- 새로운 환경에서 동작할 수 있도록 이식 및 수정
- 예방적 조치

분류기준	유지보수의 종류
사유	교정, 적응, 완전 유지보수
시간	계획, 예방, 응급, 지연 유지보수
대상	데이터 / 프로그램, 문서화, 시스템 유지보수

❖ 학습내용

[2] 소프트웨어 유지보수

3. 유지보수의 유형

◈ 수정 유지보수(Corrective Maintenance)

▪ 일반적으로 시스템이나 소프트웨어의 문제, 결함을 수정하기 위한 유지보수작업

처리오류	수행오류	구현오류

비정상적인 프로그램 중단, 입력 데이터 검증 누락,
출력 프로그램의 부정확

처리오류	수행오류	구현오류

느린 응답시간 또는 부적절한 트랜잭션 처리율

처리오류	수행오류	구현오류

프로그램 설계에 있어서 표준, 범칙 또는 불 일관성/불 완전성

❖ 학습내용

[2] 소프트웨어 유지보수

3. 유지보수의 유형(계속)

◈ 적응 유지보수(Adaptive Maintenance)

- 새로운 환경의 적응과 새로운 요구사항에 대한 소프트웨어의 적응을 위한 유지보수 작업

프로그램 환경의 변화	데이터 환경의 변화	처리 환경의 변화

프로그램 환경 변화에 소프트웨어를 적응시키도록 수행

프로그램 환경의 변화	데이터 환경의 변화	처리 환경의 변화

데이터 매체의 변경, 일반 파일에서 데이터 베이스 관리시스템의 변환

프로그램 환경의 변화	데이터 환경의 변화	처리 환경의 변화

새로운 하드웨어 플랫폼 또는 운영체제로 이전

◈ 완전 유지보수(Perfective Maintenance)

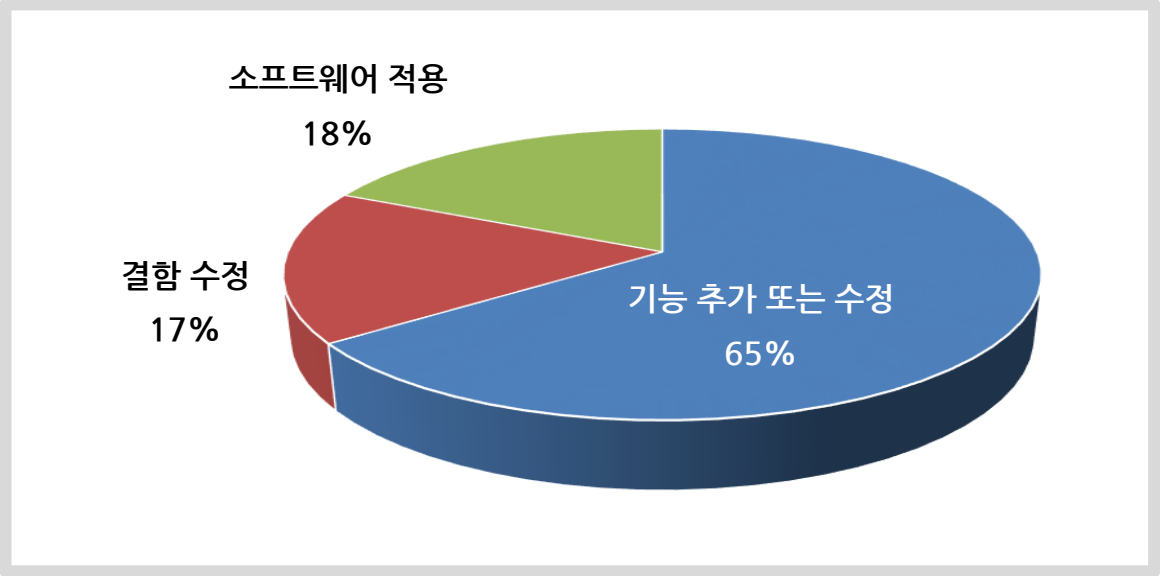
- 수행력 향상, 프로그램 특성을 변경 또는 첨가, 또는 프로그램의 장래 유지보수성을 향상시키기 위해 수행
- 새로운 요구사항을 구현하기 위하여 관련 소프트웨어를 완전하게 수정하는 유지보수 작업

❖ 학습내용

[2] 소프트웨어 유지보수

3. 유지보수의 유형(계속)

- 완전 유지보수(Perfective Maintenance)(계속)
- 유지보수 활동 비중



[3] 정보시스템 외주 개발

1. 외주 개발 개요

- 정보시스템 외주 개발의 정의
  - 핵심역량 집중을 위해 기업 내 부가가치가 낮은 IT 자원, 관리, 운영 등을 외부에 위탁하고 기업은 핵심업무에만 주력하도록 하는 전략
  - 조직이 명확한 전략 목표를 가지고 정보시스템 관련 정보 자원 관리 활동의 전부 또는 일부를 외부의 전문기관에 위탁 관리하게 하는 장기적인 계약
  - 기업이 보유한 전산장비와 인력 등의 전산자원 중 부가가치가 낮거나 내부에서 소화할 수 없는 업무를 외부에 위탁하여 처리하고 기업은 핵심기능만을 보유하여 경영에 전념하는 전략
  - 현재는 IT외주 개발(IT Outsourcing)이 잘 정착된 제도

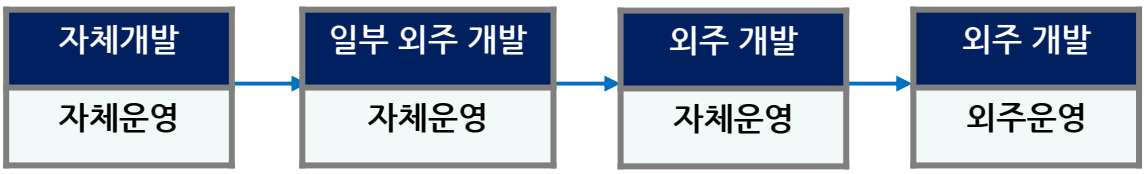
❖ 학습내용

[3] 정보시스템 외주 개발

1. 외주 개발 개요(계속)

◈ 정보시스템 외주 개발의 정의(계속)

◈ 외주 개발 전환 과정 개념



◈ 정보시스템 외주 개발의 필요성

외주 개발의 필요성

- 외부의 고급기술을 이용하여 급변하는 기술환경에 적응하고 이직 등 인력 이동에 따른 부작용 해소
- 전산비용의 절감과 핵심사업 위주로 역량 집중이 필요하며 매번 RFP 통한 계약발주 TCO 절감 효과
- 비용 절감, 원가 절감
- 자체 기술력의 부족, 개발인력 부족
- 전문지식을 가진 업체나 기술인력으로부터 기술이전
- 전략적 차원에서 전문분야에 집중하여 고부가가치의 업무형태로 전환

◈ 외주 개발의 효과

업무수행부문의 위탁으로 경쟁우위를 위한
전략적 핵심 이슈 전념 가능

외주 개발 회사가 신기술, 도구, 방법론,
전문가 등의 기술 또는 기능 이전 가능

❖ 학습내용

[3] 정보시스템 외주 개발

1. 외주 개발 개요(계속)

◈ 외주 개발의 효과(계속)

내부 운영, 기능 및 프로세스의 생산성 제고

물적/인적자원의 유연성 제고

비핵심 분야의 운영비용 절감

경영측면	핵심사업의 역량집중 가능
재정측면	투자에 대한 Risk 관리 가능
기술측면	Quality의 향상 기대
정책측면	위기관리의 적절한 대응 가능
조직측면	인력 및 적절한 배치의 문제 해소

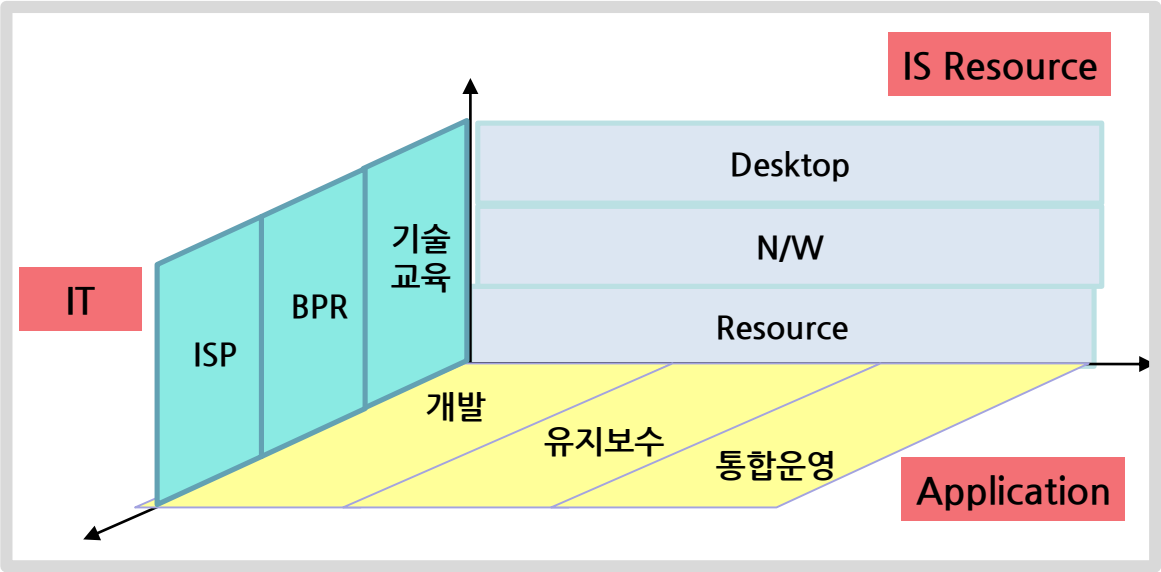
❖ 학습내용

[3] 정보시스템 외주 개발

2. 외주 개발 대상과 유형

◈ 외주 개발의 대상

- IT, IT System Resource, Application이 대상이 될 수 있음



◈ 외주 개발의 유형

- 전체 외주, 부분 외주, IT자회사 설립, 협력 개발 등의 유형이 있음

유형	장점	단점
전체 외주 개발	▪ 책임소재 명확 ▪ 친밀한 관계유지 ▪ 외주 개발 비용 절감 효과	▪ 독점적 선택 문제 ▪ 가격경쟁의 어려움
부분 외주 개발	▪ 경쟁으로 인한 품질향상 ▪ 효율적인 가격경쟁 가능	▪ 책임소재 불명확, 요구사항파악문제 ▪ 관리비용 발생

❖ 학습내용

[3] 정보시스템 외주 개발

2. 외주 개발 대상과 유형(계속)

◈ 외주 개발의 유형(계속)

▪ 전체 외주, 부분 외주, IT자회사 설립, 협력 개발 등의 유형이 있음(계속)

유형	장점	단점
IT 자회사 설립	▪ Family 의식 ▪ 의사소통 및 단결감 조성	▪ 나태함 ▪ 필요이상의 전산투자 발생 가능
협력 개발	▪ IT 기획, 총괄과 수행의 분리 ▪ 급변하는 환경에 적합	▪ 전략과 수행의 Gap 발생 가능

◈ 외주 개발의 형태

▪ 계약 관점

도급 일정한 업무 결과에 대해 발주자가 보수지급, 성과물 수반

위임 발주자로부터 일정한 업무처리의 위임을 받은 수탁자가 업무처리, 성과물 수반 필수 아님

파견 파견선과 파견기술자 간에 체결된 파견계약에 정해진 업무만 수행

❖ 학습내용

[3] 정보시스템 외주 개발

2. 외주 개발 대상과 유형(계속)

◈ 외주 개발의 형태(계속)

▪ 개발 생명주기 관점

감리위탁	외주업체에 대한 감독 업무 위임
업무위탁	업무분석, 기본설계 등 상위공정 부분 외부 전문가의 지원을 받아 업무수행
개발위탁	상세설계 이하 개발과 설치
인력위탁	코딩, 테스트 등 기술분야별 전문가
운영위탁	24시간 연중 무휴, 유지보수 전문 대행
교육위탁	기술 향상, 업무지식 함양

❖ 학습내용

[3] 정보시스템 외주 개발

3. 외주 개발 추진 및 효과

◈ 외주 개발 추진 절차

- 정보시스템 전략 및 외주 개발 수행계획 수립
- 외주 개발 대상업무의 선정

- RFP에 의한 우선 서비스 제공자 선정
- 서비스 제공자 확정

❖ 학습내용

[3] 정보시스템 외주 개발

3. 외주 개발 추진 및 효과(계속)

◈ 외주 개발 추진 절차(계속)

- 서비스 수준, 비용, 업무분담에 대한 협상
- 협상 결과를 바탕으로 계약

- 정보 자원의 이전 및 서비스의 전환
- 자산, 인력 등의 이전

❖ 학습내용

[3] 정보시스템 외주 개발

3. 외주 개발 추진 및 효과(계속)

◈ 외주 개발 추진 절차(계속)

- 서비스 수준에 따른 성과관리
- 계약기간 만료 후 계약 갱신

- 제3의 서비스 제공자와 계약 또는 내부 전환

❖ 학습내용

[3] 정보시스템 외주 개발

3. 외주 개발 추진 및 효과(계속)

◈ 외주 개발 추진 고려사항

- 대상 업무 선정 시 업무별 중요도와 핵심 역량 정도를 파악하여 업무 선정
- 협상 및 계약단계에서 명확한 서비스 수준에 대한 협상을 통한 SLA 도출
- 전환 및 이행 시 자산의 이전과 인력의 고용유지 정책 필요
- 계약관리 시 협상단계에서 도출된 SLA를 바탕으로 성과를 측정 관리하는 SLM

◈ 외주 개발 위험요소와 제거방안

구분	위험요소	제거방안
통제 문제 (Control)	▪ 성능측정, 관리부재 ▪ 서비스 레벨, 성과측정 애로 ▪ 미래환경의 불확실성에 따른 계약의 안정성 유지 곤란	▪ Pilot, 계약사항 위주 검토 ▪ 적절한 Pilot 시행을 통한 검증, 보완 ▪ 계약사항 조정, 변경을 위한 유연성 확보
보안 문제 (Security)	▪ 전략정보, 프라이버시 정보 등의 ▪ 유출에 의한 피해	▪ ESM, ITA, SLA ▪ 필요 보안수준의 명확한 정의 ▪ 관리적 / 시스템적 보안수준의 제고
고객 비즈니스 이해 부족 (Knowledge)	▪ 고객 산업, 업무, 문화의 이해 부족으로 인한 성과 저하 및 불필요한 충돌 발생 가능성	▪ 유사 영역의 경험보유 벤더 선정 ▪ 효과적 커뮤니케이션 채널 확보

❖ 학습내용

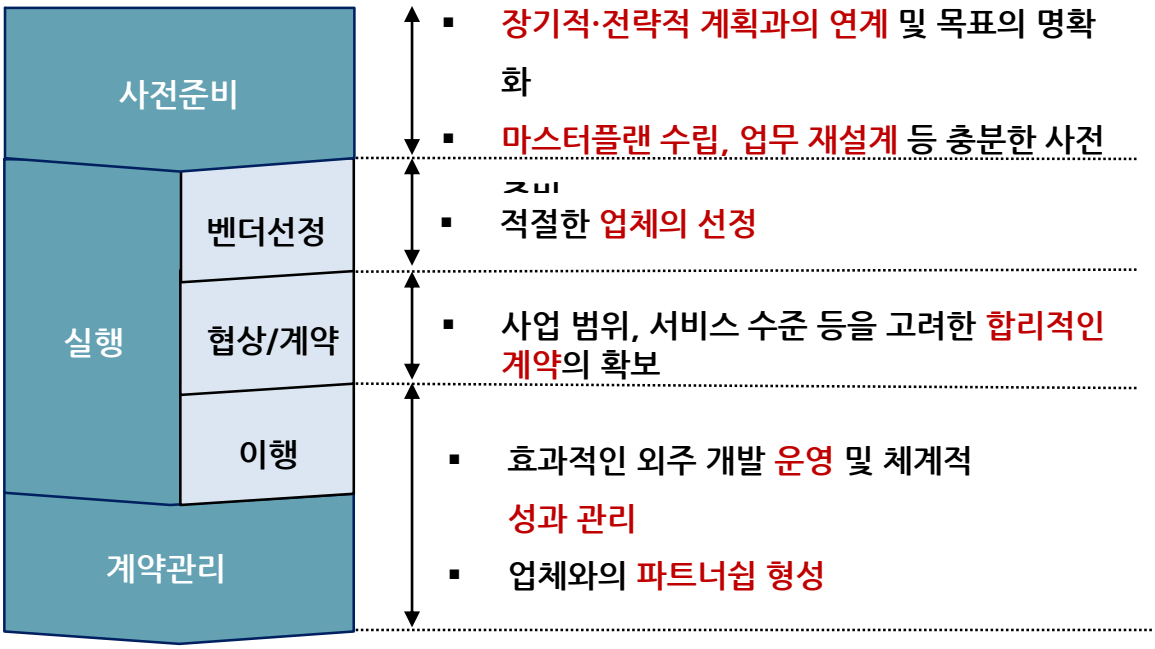
[3] 정보시스템 외주 개발

3. 외주 개발 추진 및 효과(계속)

◈ 외주 개발 위험요소와 제거방안(계속)

구분	위험요소	제거방안
외주 개발 전략 변경문제 (Reversibility)	▪ 교체, 변경에 따른 문제점 ▪ 외주 개발 대상 기능 및 서비스, 벤더의 변경, 혹은 외주 개발 기능의회수 등에 따르는 높은 교체비용	▪ 장기적 계획에 기반한 외주 개발 추진 ▪ 핵심(Core) 기능 및 핵심 인력 유지 ▪ CBD
의존성 문제 (Dependency)	▪ 정보시스템 서비스에 대한 벤더에의과도한 의존으로 인해 필요한 정보시스템 서비스의 조달 시 유연성 및 선택범위 제한	▪ 핵심부분의 관리 ▪ IT 트렌드 및 필수 Knowledge (기술, 벤더 외주 개발 관리 등)에 대한 지속적 Follow-up

◈ 외주 개발 수행 시 성공요인



❖ 학습내용

[3] 정보시스템 외주 개발

3. 외주 개발 추진 및 효과(계속)

◈ 외주 개발 수행 시 해결과제

구분	해결과제
서비스 제공자	▪ 외주 개발 서비스에 대한 평가 방법론 수립 ▪ 엄격한 SLA의 실행 ▪ 외주 개발 서비스에 대한 비용산정 기법의 개발 ▪ 체계적인 정보시스템 운영 관리 방법론의 수립
발주자	▪ 외주 개발의 본질 이해 ▪ 외주 개발을 통한 효과의 기대수준 조정 ▪ 외주 개발의 수행 방법론 이해 ▪ 외주 개발 전문가 양성

❖ 핵심정리

1. 프로그램 진화역학

- 정보시스템은 기업의 요구조건과 다양한 환경변화 등에 따라 계속적으로 변화가 이루어져야 함
- 시스템이 환경 속에 설치된 후, 환경이 변화할 수 있고, 따라서 시스템 요구사항이 변경될 수 있으며 해당 환경에서 계속적으로 유용하려면 시스템은 반드시 변경되어야 함

2. 소프트웨어 유지보수

- 소프트웨어 유지보수는 오류의 수정, 원래의 요구를 정정, 기능과 수행력을 증진시키는 일련의 작업
- 유지보수의 목표유지보수의 목표: 소프트웨어의 성능 개선, 소프트웨어의 하자 보수, 새로운 환경에서 동작할 수 있도록 이식 및 수정, 예방적 조치
- 유지보수의 유형: 수정 유지보수, 적응 유지보수, 완전 유지보수

3. 정보시스템 외주 개발

- 기업이 보유한 전산장비와 인력 등의 전산자원 중 부가가치가 낮거나 내부에서 소화할 수 없는 업무를 외부에 위탁하여 처리하고 기업은 핵심기능만을 보유하여 경영에 전념하는 전략인 IT외주 개발(IT Outsourcing)은 현재 잘 정착된 제도