**소프트웨어 공학 Homework**

22016156007

소프트웨어학과

김영민

목차

**2주차 (3/20) “Chapter 1. 소프트웨어공학 소개” 연습문제4p**

13. 프로그램과 소프트웨어를 구별하여 설명하시오.

15. 소프트웨어 특징을 하드웨어와 비교하여 설명하시오

17. 대규모 소프트웨어 개발의 어려운 점과 이를 해결하기 위한 소프트웨어 학적인 대안을 설명하시오.

20. 소프트웨어 개발 단계별 특징을 간략히 쓰시오.

**3주차 (4/02) “Chapter 4. 요구사항 분석” 연습문제6p**

22. 요구 분석이 설계, 코딩, 테스트와 비교했을 때 왜 더 어렵다고 생각하는지 그 이유를 쓰시오.

24. 요구 분석의 어려움 중에서 ‘의사소통의 문제’ 대해 해결 방안을 설명하시오.

26. 비기능적 요구 사항의 항목 중 사용성을 높일 수 있는 방법을 예를 들어 설명하시오.

28. 방법론에 따른 요구 사항 정의 결과를 표현하는 도구들이 다르다. 이 도구들을 사용해서 요구 사항 정의를 예를 들어 표현해보시오.

29. 요구 분석 명세서는 이해 당사자마다 사용하는 용도가 다르다. 이해 당사자마다 어떤 용도로 사용하는지 설명하시오.

**4주차 (4/07) “Chapter 2. 소프트웨어개발 프로세스” 연습문제8p**

21. 소프트웨어 프로세스 모델의 목적과 역할을 설명하시오.

22. 폭포수 모델의 장/단점을 설명하시오.

23. 프로토타입 모델의 등장 배경을 설명하시오.

25. 나선형 모델의 위험 분석 단계에서 위험 요소를 찾고, 그 대응책을 설명하시오.

26. 점증적 개발 방법의 예를 만들어 설명하시오.

27. 반복적 개발 방법의 예를 만들어 설명하시오.

5주차 (4/15) “Chapter 2. 계획” 연습문제11p

22. 소프트웨어 개발 기간은 10개월이고, 6명의 개발자는 4개월 동안, 5명의 개발자는 6개월 동안참여한다면 이 소프트웨어 개발에 대한 노력은 얼마인가?

23. LOC 기법에 의해 예측된 총 라인 수가 30000라인이고, 개발자는 5명이 참여한다. 그리고 개발자들은 월 평균 600라인을 코딩한다면 개발 기간은 얼마나 되는가?

24. 계획 단계에서 정규 기능 점수법을 사용하지 않고 간이 기능 점수법을 사용하는 이유를 설명하시오.

다음 표에 대해 임계 경로를 찾는 과정을 설명하시오.

28. 일정 계획에 사용되는 기법을 나열하고 간략히 서술하시오.

29. 임계 경로가 필요한 이유를 쓰시오.

30. 일상생활에서 발생할 수 있는 위험을 찾아 예방, 위험 최소화 방안에 대해 설명하시오.

**6주차 (4/21) “Chapter 5. 상위 설계” 연습문제15p**

21. 건축 설계 과정과 소프트웨어 설계 과정을 비교하여 설명하시오.

23. 과정 추상화, 자료 추상화, 제어 추상화의 예를 들어보시오.

**7주차 (4/28) “Chapter 5. 상위 설계”** **16p**

24. 단계적 분해의 예를 들어보시오.

**8주차 (5/05) “Chapter 6. 하위 설계”** **17p**

21. 기능적 응집의 예를 프로그램으로 작성하시오.

22. 논리적 응집의 예를 만들고 설명하시오.

23. 제어 결합의 예를 찾아 프로그램으로 작성하고 설명하시오

24.공통 결합의 예를 프로그램으로 작성하고 설명하시오.

**9주차 (5/13) “Chapter 6. 하위 설계”** **20p**

25. 상속의 예를 프로그램으로 작성하시오.

26. 다형성의 예를 만들고 설명하시오.

27. 단일 책임의 원칙을 적용하기 전과 후가 어떻게 달라지는지 설명하시오.

28. 개방 폐쇄의 원칙을 적용하기 전과 후가 어떻게 달라지는지 설명하시오.

29. 의존 관계의 원칙을 적용하기 전과 후가 어떻게 달라지는지 설명하시오.

30. 인터페이스 분리의 원칙을 적용하기 전과 후가 어떻게 달라지는지 설명하시오.

**10주차 (5/20) “Chapter 7. 구현”** **22p**

14. 다음 코드는 어떤 표준 코딩 규칙에 어긋나는가?

16. 표준 코딩 규칙에 어긋나는 곳을 찾아 그 이유를 설명하시오.

17. 명칭의 규칙을 지키지 않은 나쁜 예이다. 명칭의 규칙에 맞게 수정하시오.

18. 다음 코드가 포인터 변수의 명명 규칙에 맞지 않은 이유를 써보시오.

19. 프로그램에서 중괄호({}) 가 원시 코드 형식에 관한 규칙을 잘 지킨 예를 써보시오.

20. 다음 문장을 ‘수식에 관한 규칙’에 맞게 고치시오.

21. switch문에서 break문이 빠졌다면 어떤 문제가 발생하는지 설명하시오.

22, switch문에서 default문의 역할을 설명하시오.

23. 다음 코드의 문제점을 설명하고, 수정하시오.

24.; 다음 코드의 문제점을 설명하고, 수정하시오.

25. 다음 코드의 문제점을 설명하고, 수정하시오.

11주차 (5/21) “Chapter 8. 테스트” 27p

23. 동등 분할 기법의 예를 만들어 설명하시오.

24. 경계 값 분석 기법의 예를 만들어 설명하시오.

25. 원인-결과 그래프 기법의 예를 만들어 설명하시오.

12주차 (6/03) “Chapter 9. 품질” 30p

21. 표준 프로세스의 필요성에 대해서 설명하시오.

27. CMMI에서 관리 단계와 정의 단계를 일반적인 예를 들어 설명하시오.

29. 품질 보증의 효과와 문제점에 대해 설명하시오.

13주차 (6/08) “Chapter 10. 프로젝트 관리” 30p

20. 변경 관리의 필요성을 설명하시오.

23. 최종 릴리스되기까지의 과정을 버전 항목 트리를 이용하여 예를 들어 설명하시오.

27. 형상 통제에 대해 설명하시오.

29. 형상 관리의 효과에 대해 설명하시오

**2주차 (3/20) “Chapter 1. 소프트웨어공학 소개”**

**13. 프로그램과 소프트웨어를 구별하여 설명하시오.**

프로그램과 소프트웨어는 일반 통상적으로는 같은 의미로 쓰이기도 하지만 구별하자면 프로그램은 소프트웨어의 한 종류로, 계획된 내용에 따라서 순차적으로 진행되는 소프트웨어를 의미합니다. 소프트웨어는 프로그램보다 컴퓨터의 구조를 생각한 단어로 하드웨어의 반대 개념으로서 의미이며 프로그램 뿐만 아니라 CD, 설명서, 제품 포장 등 패키지 전체를 뜻하기도 합니다.

**15. 소프트웨어 특징을 하드웨어와 비교하여 설명하시오.**

소프트웨어는 컴퓨터 프로그램으로 하드웨어의 특정 목적 달성에 적합하도록 구동 시키기 위해 논리 정연하게 나열한 것으로 컴퓨터 프로그램의 총칭입니다. 하드웨어와 달리 물리적인 형태가 존재하지 않고 논리적인 단어들로 이루어져 있습니다.

**17. 대규모 소프트웨어 개발의 어려운 점과 이를 해결하기 위한 소프트웨어 공학적인 대안을 설명하시오.**

대규모 소프트웨어 개발을 하게 되면 소프트웨어의 규모가 커질수록 개발 과정이 복잡해지고 참여 인력이 많아지면서 개발 기간이 길어지게 됩니다. 이를 해결하기 위해서 먼저 개발 과정의 복잡함을 줄이기 위한 기술을 제시하고 개발에 참여하는 팀을 구성하고 관리를 효율적으로 합니다. 마지막으로 프로젝트를 효율적으로 관리 하기위해 프로젝트관리지식체계를 구축합니다.

**20. 소프트웨어 개발 단계별 특징을 간략히 쓰시오.**

소프트웨어 개발 단계는 총 6단계로 계획, 요구분석, 설계, 구현, 테스트, 유지보수 순으로 단계가 이루어져 있습니다.

1단계로 계획은 개발 비용을 산정하고 일정 계획을 수립하며 이후 위험관리를 수행하는 단계입니다. 2단계에서는 요구분석 단계로 기존 시스템의 문제점을 파악하고 새로운 요구사항 도출 이후 다이어그램을 작성하는 단계로 이어집니다. 이후 최종 산출물로 요구 분석 명세서를 도출합니다. 3단계는 설계 단계인데 여기서 소프트웨어 아키텍처, 객체지향 설계 아키텍처 스타일 GoF의 디자인 패턴, 모듈 평가 등을 진행합니다. 4단계에서는 구현으로 표준 코딩 규칙 등을 진행합니다. 5단계에서는 테스트로 개발자, 사용자 시각에 따른 분류, 사용되는 목적에 따른 분류, 품질 특성에 따른 분류, 소프트웨어 개발 단계에 따른 분류 등을 진행합니다. 6단계는 유지보수로 수정 유지보수 적응 유지보수 기능보강 유지보수 예방 유지보수가 있습니다.

**3주차 (4/02) “Chapter 4. 요구사항 분석”**

**22. 요구 분석이 설계, 코딩, 테스트와 비교했을 때 왜 더 어렵다고 생각하는지 그 이유를 쓰시오.**

설계, 코딩, 테스트에 비해서 요구 분석은 사용자와의 마찰이 더 빈번하기 때문이라고 생각합니다. 설계, 코딩, 테스트는 요구 분석이 완료된 상태에서 정해진 매뉴얼대로 따라가면 되지만 요구 분석은 사용자에 요구에 따라 변할 수 있거나 애매해질 수 있습니다. 이렇듯 요구 분석 단계에서는 사용자와 개발자사이의 상호작용이 반드시 들어가고 사용자 자신도 소프트웨어에 반영할 요구를 구별하기 쉽지 않기 때문에 설계, 코딩, 테스트보다 더 어렵다고 생각합니다.

**24. 요구 분석의 어려움 중에서 ‘의사소통의 문제’ 대해 해결 방안을 설명하시오.**

요구 분석의 어려움 중에서 의사소통 문제를 해결하기 위해서는 사용자가 자신이 원하는 바를 견본으로 구해서 제공자에게 보여주면서 의사소통을 위한 매개체로 사용할 수 있습니다. 하지만 소프트웨어 개발에서는 이러한 견본이 없는 경우가 대부분이기 때문에 사용자의 정확한 의사 전달 능력이 중요하게 됩니다. 이를 좀더 보완하여 사용자의 요구를 정확하고 안전하게 획득하도록 하는 업무를 전문적으로 분석가가 도맡아 시행하기도 합니다.

**26. 비기능적 요구 사항의 항목 중 사용성을 높일 수 있는 방법을 예를 들어 설명하시오.**

사용성은 소프트웨어를 사용할 때 혼란스러워 하거나 사용하는 순간에 고민하지 않게 하는 편의성을 뜻합니다. 이 사용성을 높이기 위한 방법으로는 사용자의 수준에 따라 사용할 수 있게 해주는 것입니다. 예를 들어 온라인 도움말을 통해 사용법을 언제든지 편리하게 도움 받을 수 있게 하거나, 사용자 인터페이스를 일관성 있게 만들어 사용자가 메뉴를 손쉽게 사용할 수 있게 만드는 것도 시스템의 사용성을 높이는 방법이라고 할 수 있겠습니다.

**28. 방법론에 따른 요구 사항 정의 결과를 표현하는 도구들이 다르다. 이 도구들을 사용해서 요구 사항 정의를 예를 들어 표현해보시오.**

방법론에 따라 구분했을 때 구조적 방법론은 자료흐름도(DFD), 자료사전(DD), 처리 명세서를 사용해 요구 사항의 결과를 표현하고, 정보공학 방법론에서는 ERD를, 객체 지향 방법론에서는 UML을 대표적으로 사용합니다. 예를 들어 구조적 방법론에서 DFD를 사용해 표현했을 때 데이터는 사각형 박스로 출원지와 목적지를 나타내고 화살표로 데이터 흐름을 나타냅니다. 또한 직선으로 자료 저장소를 표현하고 원으로 프로세스를 표현합니다. 정보공학 방법론에서는 대표적으로 ER 다이어그램을 사용하는데 데이터를 엔티티와 관계 중심으로 표현하고 작성합니다. 객체지향 방법론에서는 UML의 유스케이스 다이어그램을 대표적으로 사용하는데 유스케이스는 사용자의 요구를 나타내는 기능으로 실제로 코딩할 수 있을 만큼의 가장 작은 단위의 기능을 말합니다. 예를 들어, 학사 관리 앱에서 수강 관리, 수업관리, 성적 관리, 교과 과정 관리, 사용자 정보 관리 등이 각각 유스케이스 하나가 됩니다. 이러한 작은 유스케이스들을 모아서 하나의 전체 시스템을 만들고 표현하게 됩니다.

**29. 요구 분석 명세서는 이해 당사자마다 사용하는 용도가 다르다. 이해 당사자마다 어떤 용도로 사용하는지 설명하시오.**

사용자의 입장에서는 요구 분석 명세서가 사용자와 의사소통 하는 도구로써 사용되면서 동시에 계약서로도 사용됩니다. 개발이 완료된 후에는 이 문서의 내용이 판단 기준이 됩니다.

개발자 입장에서는 요구 분석 명세서를 읽고나서 어떤 시스템을 개발할 것인 것 이해하게 됩니다. 요구 분석 명세서에 기술된 기능적/비기능적 요구 사항을 기반으로 분석, 설계, 코딩을 하게 됩니다. 개발을 완료한 후 모든 내용이 구현되었는지 점검할 때 요구 분석 명세서를 이용하여 점검 항목을 구성합니다. 또한 사용자 지침서 초안 작성을 할 때에도 요구 분석 명세서를 활용합니다.

테스터 입장에서는 요구 분석 명세서를 테스트 수행을 위한 테스트 케이스를 만드는데 사용됩니다. 또한 요구 분석 명세서를 통해서 사용자 지침서 초안 작성용으로 활용합니다.

**4주차 (4/07) “Chapter 2. 소프트웨어개발 프로세스”**

**21. 소프트웨어 프로세스 모델의 목적과 역할을 설명하시오.**

우선 소프트웨어 프로세스의 목적은 최종적으로 고품질의 소프트웨어 제품을 만드는 것으로 주어진 예산과 자원을 효율적으로 활용하여 프로그램을 개발하고 관리, 유지 보수하는 구체적인 방법을 정의하는 것입니다. 각 단계마다 어떠한 순서로 어떤 자원을 활용할 것인지에 대해서 명시하고 프로세스 관리와 소프트웨어 품질 관리를 명확히 하는 것을 목적으로 두고 있습니다. 소프트웨어 프로세스 모델의 역할로는 프로젝트에 대해 전체적으로 기본 골격을 세워주는 것을 기본으로 합니다. 일정을 수립하고 이용할 수 있는 자원을 산정하고 분배하며 최종적으로 개발 비용을 점검할 수 있습니다. 또 사람마다 다르게 쓰는 용어를 표준화 하면서 참여자 간에 의사소통의 기준을 제시하며 개발 진행 상황도 표기해서 명확히 할 수 있습니다. 마지막으로 각 단계별로 생성되는 문서를 포함한 산출물을 활용하여 프로그램을 검토하는 것이 가능합니다.

**22. 폭포수 모델의 장/단점을 설명하시오.**

먼저 폭포수 모델은 절차가 간결하고 이해하기 쉽기 때문에, 또 순차적 모델이므로 단계별 진척 사항에 대한 관리가 용이하다는 점이 있습니다. 또 폭포수 모델은 각 단계별로 정형화된 접근방법을 사용하기 때문에 단계별 산출물을 체계적으로 문서화할 수 있다는 점도 있습니다. 폭포수 모델의 단점으로는 먼저 각 단계의 앞 단계가 완료되어야 다음 단계를 수행할 수 있다는 점이 있습니다. 또 폭포수 모델은 각 단계마다 작성된 결과물이 완벽한 수준으로 작성되어야 다음 단계에 오류를 넘겨주지 않는다는 점이 있습니다. 초기 단계의 오류가 구현 단계에서 발견되거나 개발이 완료되어 사용 중에 발견된다면 수정하는 데 비용과 시간이 많이 소요되기 때문에 전체 일정에 큰 부담을 줄 수 있는 단점 중 하나입니다. 마지막으로 사용자가 중간에 가시적인 결과를 볼 수 없다는 점도 단점입니다. 사용자가 소프트웨어의 작동되는 모습을 보려면 모든 프로그램의 구현이 끝나야만 볼 수 있는데 만약 사용자가 원하지 않는 결과가 아니라 요구사항을 받아 수정을 해야 한다면 다시 시간과 비용이 매우 많이 들기 때문에 폭포수 모델의 단점으로 들 수 있습니다.

**23. 프로토타입 모델의 등장 배경을 설명하시오.**

조금씩 키워 나가는 것과 닮아 생물과 비교할 수 있는 진화적 프로세스 모델은 대표적으로 프로토타입이 있습니다. 좋은 모델이지만 단점 또한 많은 폭포수 모델은 단계를 거슬러 올라가 작업하는 것이 쉽지 않은 점이 가장 큰 단점 중 하나인데 현실에서는 새로운 요구를 끊임없이 수시로 받아 민첩하게 대응해야 하므로 용이하진 않은 부분이 많습니다. 이러한 폭포수 모델의 단점을 보완하여 나온 것이 바로 프로토타입 모델입니다. 프로토타입 모델은 먼저 초기버전 프로토 타입을 만들고 변경된 요구사항을 받아 반영할 때 2차 프로토타입을 만들어 사용자들에게 보여주게 됩니다. 이러한 작업을 반복하여 최종 제품을 완성하는 것인데, 이렇게 프로토타입 모델을 사용하면 반복적인 사용자의 요구사항을 그때그때 반영하고 수정해서 완성 제품을 대폭 수정해야할 수고 없이 폭포수 모델에 비해서 빠른 시간과 비용으로 효율적인 개발을 할 수 있어 등장하게 되었습니다.

**25. 나선형 모델의 위험 분석 단계에서 위험 요소를 찾고, 그 대응책을 설명하시오.**

나선형 모델에서 위험 분석 단계에서는 프로젝트 수행에 방해되는 위험 요소를 찾아 목록을 작성하고 위험에 대한 예방 대책을 논의하는 과정입니다. 여기서 위험요소란 소프트웨어 개발 과정이 순조롭게 진행되는데 방해되는 모든 것을 말하는데, 빈번히 변경되는 요구사항과 팀원들의 경험 부족, 떨어지는 팀워크과 프로젝트 관리 부족 등을 말합니다. 이러한 위험 요소들을 해결하기 위해서는 프로젝트 초기부터 실행이 가능하고 테스트가 가능한 시스템을 개발하여 꾸준히 확인을 해주어야 합니다. 나선형 모델은 폭포수 모델과 프로토타입 모델과는 달리 위험 요소를 최소화하기 위한 방법으로 개발 단계에 위험을 분석할 수 있는 과정을 따로 지정해 소프트웨어 개발 과정 초기에 발생할 수 있는 실패 요인과 위험 요소를 찾고 관리하여 위험을 최소화하게 됩니다. 또 위험요소를 평가하여 개발에 얼마나 영향을 주는지 대안을 찾고 분석하여 프로젝트의 진행 여부를 결정하게 됩니다.

**26. 점증적 개발 방법의 예를 만들어 설명하시오.**

점증적 개발 방법은 단계적 개발 모델 중 하나로 소프트웨어 개발에서 중요하다고 생각되는 부분부터 차례로 개발한 후 그 일부를 사용하면서 개발 범위를 점차 늘려가는 방식입니다. 한가지 업무가 끝나면 비로소 그 다음 업무를 시작하고, 끝나면 다시 다음 업무를 시작해서 하나씩 늘려가는 방식입니다. 점증적 개발 방법의 예를 여러가지 들어보면, 먼저 과제를 할 때 1번 문제를 풀고나서 순차적으로 2번, 3번… 문제를 풀어나가는 방식이 있겠고, 정보 시스템 개발을 할 때 가장 중요하다고 생각하는 학사 관련 시스템을 먼저 개발한 후 회계 업무를 비롯한 다른 부서의 업무 시스템들을 차츰 순서 있게 만드는 것, 3층을 목표로 하는 건물을 지을 때 먼저 1층을 먼저 완벽하게 지은 후 2층을 짓고 다시 2층을 완벽하게 짓고 나서 비로소 3층을 짓을 방식도 예를 들 수 있겠습니다. 또 공장 컨베이어 벨트에서 한 단계가 온전하게 다 끝나야 비로소 다음 단계로 진입하고 계속해서 단계를 거치면서 제품을 완성해 나가는 방식 또한 점증적 개발 방법의 예로 들 수 있겠습니다.

**27. 반복적 개발 방법의 예를 만들어 설명하시오.**

반복적 개발 방법은 초기에 시스템 전체를 일차적으로 개발하여 인도한 후, 각 서브 시스템의 기능과 성능을 변경 및 보강하여 완성도를 높이게 됩니다. 이렇게 업그레이드된 릴리스 버전을 다시 내놓는 것이 반복적 개발 방법입니다. 반복적 개발 방법의 예로는 한정식 상차림처럼 모든 음식을 한 상에 가득 차려 놓은 후 조금씩 모두 맛보고, 그 중에 맛 잇는 음식은 한번 더 먹고, 정말 맛있는 음식은 여러 번 반복적으로 먹는 방식을 예로 들 수 있습니다. 또 도서 집필의 예로 1장부터 10장 까지 큰 줄거리를 중심으로 개괄적인 1차 원고를 작성한 뒤 다시 전체적으로 내용을 보강하고, 이러한 과정을 몇 번 반복하여 만족할 만한 수준으로 원고가 완성되면 책으로 출간하는 방식, 그림을 그릴 때 먼저 전체적으로 스케치를 그린 후 더 자세하게 그리고 싶은 부분을 더 그려서 보강하고 이러한 과정을 거듭 반복하여 마침내 그림을 완성하는 방식, 단층 건물을 지을 때 먼저 전체적인 뼈대를 만들고 나서 보강해야 할 부분을 구현하고 이를 반복해서 집을 완성하는 방식 또한 예로 들 수 있습니다.

**5주차 (4/15) “Chapter 2. 계획” 연습문제**

**22. 소프트웨어 개발 기간은 10개월이고, 6명의 개발자는 4개월 동안, 5명의 개발자는 6개월 동안참여한다면 이 소프트웨어 개발에 대한 노력은 얼마인가?**

노력은 인원과 개발 기간의 곱이므로

M/M = (6 \* 4) + (5 \* 6) = 24 + 30 = 54

**23. LOC 기법에 의해 예측된 총 라인 수가 30000라인이고, 개발자는 5명이 참여한다. 그리고 개발자들은 월 평균 600라인을 코딩한다면 개발 기간은 얼마나 되는가?**

노력은 추정LOC를 1인당 월 평균 생산 코드 라인 수로 나눈 값이고 개발 기간은 M/M을 참여 인원으로 나눈 값이므로

M/M = 30000/ 600 = 50, 개발 기간 = (M/M)/참여 인원 = 50/5 = 10

따라서 개발 기간은 10개월입니다.

**24. 계획 단계에서 정규 기능 점수법을 사용하지 않고 간이 기능 점수법을 사용하는 이유를 설명하시오.**

간이 기능 점수법은 프로젝트 초기 단계에서 각 기능의 정확한 요소를 모르는 경우 유형별 평균적 점수인 평균 복잡도 가중치를 사용하여 소프트웨어 기능의 크기를 측정하는 방법으로 절차는 정규 기능 점수법과 동일하지만 정규 기능 점수법은 유형별 복잡도를, 간이 기능 점수법은 평균 복잡도를 사용하는 차이가 있습니다. 이때 유형별 평균 복자도를 적용해 기능 점수를 산출하는데 예산 수립 과정이나 사업 초기에는 기능 점수를 산정할 수 있을 만큼 자료가 충분하지 않고 평균 복잡도 가중치를 미리 정하고 제시하는 것이 정규 기능 점수법보다 사용이 편리하여 정규 기능 점수법보다는 간이 기능 점수법을 사용합니다.

**26. 다음 표에 대해 임계 경로를 찾는 과정을 설명하시오.**

|  |  |  |
| --- | --- | --- |
| 소작업 명 | 선행 작업 | 소요 기간(일) |
| A | - | 10 |
| B | - | 15 |
| C | A | 15 |
| D | - | 10 |
| E | B,D | 10 |
| F | A,B | 5 |
| G | A | 20 |
| H | D | 25 |
| I | C.F | 15 |
| J | G,E | 15 |
| K | I | 5 |
| L | K | 10 |

**CPM**

C

K

L

I

A

F

J

B

G

E

D

H

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 작업 | A | B | C | D | E | F | G | H | I | J | K | L |
| ES | 0 | 0 | 10 | 0 | 15 | 15 | 10 | 10 | 25 | 30 | 40 | 45 |
| 작업 기간 | 10 | 15 | 15 | 10 | 10 | 5 | 20 | 25 | 15 | 15 | 5 | 10 |
| EF | 10 | 15 | 25 | 10 | 25 | 20 | 30 | 35 | 40 | 45 | 45 | 55 |
| LS | 0 | 5 | 10 | 0 | 20 | 20 | 10 | 10 | 25 | 30 | 40 | 45 |
| 여유시간 | 0 | 5 | 0 | 0 | 5 | 5 | 0 | 0 | 0 | 0 | 0 | 0 |

**임계 경로**

C

K

L

I

A

F

J

B

G

E

D

H

**28. 일정 계획에 사용되는 기법을 나열하고 간략히 서술하시오.**

일정 계획 기법은 먼저 작업 분할 구조도(WBS)로 시작하여 네트워크 차트(PERT/CPM)와 간트 차트를 이용합니다. 작업 분할 구조도는 프로젝트 목표를 달성하기 위해 필요한 활동과 업무를 세분화하는 작업입니다. 프로젝트 구성 요소들을 계층 구조로 분류하여 프로젝트의 전체 범위를 정의하고, 프로젝트 작업을 관리하기 쉽도록 세분화하게 됩니다. 네트워크 차트는 작업 분할 구조도의 작업 순서, 소요 기간 등을 네트워크 형태의 그래프로 표현하고 나서 완료일에 프로젝트를 마치려면 어떤 작업을 중점적으로 해야 하는지 찾아내서 전체 작업 일정을 세분화하여 작업 지연을 사전에 예방할 수 있도록 하는 기법입니다. 간트 차트는 프로젝트 일정 관리를 위한 바 형태의 도구로 프로젝트의 주요 활동을 파악한 후, 각 활동의 일정을 시작하는 시점과 끝나는 시점을 연결한 막대 모양으로 표시하여 전체 일정을 한눈에 볼 수 있게 하는 기법입니다.

**29. 임계 경로가 필요한 이유를 쓰시오.**

임계 경로는 그래프에서 여유 시간이 없는 경로를 말하는데, 임계 경로에는 여유시간이 없으므로 모든 일정은 임계 경로에 좌우되어 임계 경로에 있는 작업이 1일 지연되면 전체 일정도 똑같이 1일이 지연되게 됩니다. 하지만 반대로 전체 프로젝트 완료 시간을 단축하고 싶다면 임계 경로의 작업시간을 줄이면 되기 때문에 임계 경로는 프로젝트의 수행시간을 결정하는데 매우 중요한 척도입니다.

**30. 일상생활에서 발생할 수 있는 위험을 찾아 예방, 위험 최소화 방안에 대해 설명하시오.**

1. 현 시국에 코로나 확산 방지를 위하여 먼저 마스크를 쓰고 외출하며 혹시 모를 손에 증식하는 바이러스 제거를 위하여 수시로 손 세정제를 뿌리며, 또 비누로 손을 깨끗이 자주 씻습니다.

2. (프로젝트와 연계) 본 학교관생 및 관계자를 제외한 타인이 학교의 시설 확인 시스템 서비스를 사용할 경우 이상이 생길 수 있으므로 사전에 로그인 되어 있는 신원 확인이 가능한 사람만 해당 서비스를 이용 가능하도록 설계합니다.

**6주차 (4/21) “Chapter 3. 상위 설계” 연습문제**

**21. 건축 설계 과정과 소프트웨어 설계 과정을 비교하여 설명하시오.**

건축 분야에서 설계는 고객의 원하는 요구 사항을 반영해 원하는 집의 정보를 건축 설계사에게 제시하면 건축 설계사는 집주인의 요구사항과 경험을 통해 터득한 제약 사항을 등을 토대로 설계한 도면을 집주인에게 보여주며 의견을 교환합니다. 이후 합의한 과정대로 최종 산출물인 설계 도면을 만들게 됩니다. 이 설계 도면은 용도에 따라 매우 다양한 종류로 제작되게 됩니다. 이와 비교하여 소프트웨어에서 설계는 앞서 설명한 건축 설계 과정과 유사한데 사용자의 요구사항에 따라 요구 분석 명세서가 만들어 지면 개발 팀은 이 명세서를 참조하여 설계서를 작성한 뒤 이를 기반으로 구현하게 됩니다. 여기서 요구 분석 명세서를 기반으로 만든 설계서는 건축의 다양한 도면처럼 상위 설계, 하위 설계, 아키텍처 설계, 사용자 인터페이스 설계 등의 다양한 형태로 구체화됩니다.

**23. 과정 추상화, 자료 추상화, 제어 추상화의 예를 들어보시오.**

추상화는 전체에 대한 상세한 설명을 하지 않고 특정한 이미지로 개념을 표현하는 것으로 프로그램 전체에 대해 알고리즘 형태로 작성하는 과정 추상화, 데이터를 모아 데이터 구조 형태로 표현한 데이터 추상화, 여러 줄의 내용을 간략히 줄여서 표현한 제어 추상화가 있습니다.

과정 추상화의 예로는 학생들의 학점을 계산하는 알고리즘을 예로 들 수 있는데 만약 학생들의 점수의 합계를 구한다고 해도 “합계를 구한다”라고만 할 뿐이지 자세한 과목명은 언급하지 않고 또한 학생 이름과 평균 점수를 출력한다고 할 때도 어떻게 출력 할 것인지 자세하고 디테일한 부분은 생략하는 것을 예로 들 수 있습니다.

데이터 추상화의 예로는 클래스가 있습니다. 클래스는 사용자에게 클래스가 제공할 수 있는 사용법만 제공하고 사용자에게 불필요한 데이터와 함수를 감추는데 이 데이터의 값을 변경할 수 있는 메서드를 함께 정의해서 클래스라는 데이터 객체를 구성하게 됩니다. 데이터와 연산을 묶어 캡슐화 함으로써 데이터를 부적절하게 사용할 수 없도록 막는 것입니다. 이렇게 사용자에게는 꼭 필요한 기능을 제공하면서 필요 없는 부분은 캡슐화 해서 숨기는 방법을 예로 들 수 있습니다.

제어 추상화의 예로는 프로그램 언어에서 고급언어를 사용하는 것을 들 수 있습니다. 기계언어를 추상화해서 간단한 어셈블리어로 만들고 또 다시 어셈블리어를 추상화해서 간단한 고급 언어 표현으로 나타내면서 단계를 거칠수록 더욱 간결해지는 것을 예로 들 수 있습니다.

**7주차 (4/28) “Chapter 5. 상위 설계”**

**24. 단계적 분해의 예를 들어보시오.**

단계적 분해는 기능을 점점 작은 단위로 나누어 점차적으로 구체화 하는 방법입니다. 단계적 분해를 설명하기 위해서 구조적 분석 방법에서 사용하는 자료흐름도를 사용하였습니다. 현재 저희 팀 프로젝트 주제인 교내시설 실시간 혼잡도 분석 시스템을 분해하였습니다.

정보 제공

회원 관리

교내시설 실시간 혼잡도 분석 시스템

**8주차 (5/05) “Chapter 6. 하위 설계”**

정보 수집

회원 정보 수집

센서로 정보 수집

혼잡도 분석결과 조회

혼잡도 통계 정보 제공

회원 정보 삭제

회원 정보 조회

회원 정보 수정

**21. 기능적 응집의 예를 프로그램으로 작성하시오.**

기능적 응집은 함수적 응집이라고도 하는데 단일 기능의 요소로 하나의 모듈을 구성하는 것으로 응집도가 가장 높은 경우입니다. 기능적 응집의 예로 현재 저희 팀 프로젝트에 구상하고 있는 교내시설 실시간 혼잡도 분석 시스템에서 모듈 하나로 오로지 혼잡도 계산만을 시행만을 수행하는 함수를 예로 들 수 있겠습니다.

**22. 논리적 응집의 예를 만들고 설명하시오.**

논리적 응집에 의해 만들어진 모듈은 여러 요소를 하나의 모듈로 만든 이유가 순서와 무관합니다. 단지 요소들 간에 공통점이나 관련된 업무가 있거나 비슷한 기능을 수행한다는 이유로 하나의 모듈로 구성한 것을 말합니다. 저희 팀 프로젝트의 기능 중에서 예를 들면 해당 어플 이용 고객의 정보의 입력과 출력을 scanf() 와 printf()를 결합시킨 입출력 모듈로 구성할 수 있고, 이용자 정보 출력과 혼잡도 정보 출력을 같은 출력 기능으로 묶어서 모듈로 구성할 수도 있습니다.

**23. 제어 결합의 예를 찾아 프로그램으로 작성하고 설명하시오**

제어 결합은 모듈간 서로 간섭하는 관계입니다. 제어 결합은 제어 플래그라는 매개변수를 사용하여 호출하는 모듈이 호출되는 모듈의 내부구조를 잘 알고 논리적 흐름을 변경하는 관계로 묶이게 됩니다. 제어 결합의 프로그램 예로 switch, tag, flag등이 사용됩니다. 제어 결합의 프로그램을 작성해보면

int num = 0;

printf(“숫자 1, 2, 3중 고르시오”);

switch(num){

case 1:

printf(“안녕하세요.”);break;

case 2:

printf(“반갑습니다.”);break;

case 3:

printf(“감사합니다.”);break;

}

이러한 프로그램을 작성할 수 있습니다. 1을 입력하면 안녕하세요 출력 2입력시 반갑습니다 출력, 3입력시 감사합니다 를 출력하는 프로그램입니다.

**24.공통 결합의 예를 프로그램으로 작성하고 설명하시오.**

공통 결합은 모듈들이 공통으로 변수를 같이 사용하게 됩니다. 그러므로 모듈들이 전역변수를 사용할 때 이런 관계가 성립하게 됩니다. 만약에 공통 결합에서 사용하는 전역변수가 변하게 되면 변수를 사용하는 모든 모듈들이 영향을 받게 됩니다. 공통 결합을 사용하는 아래의 프로그램을 작성하였습니다.

void fun1(int val); // 전역변수 num을 공유하는 공통 결합

void fun2(int val); // 전역변수 num을 공유하는 공통 결합

Int num = 2; // 전역 변수

Int main(void){

Int result1, result2;

printf(“%d”,num); // 0 출력

result1 = fun1(1);

result2 = fun2(2);

printf(“%d, %d”,result1, result2);// 3, 4 출력

num++ // 젼역 변수의 값이 변경

result1 = fun1(1);

result2 = fun2(2);

printf(“%d, %d”,result1, result2);// num의 값이 변해 결과값 또한 변경됨

}

void fun1(int val){val = num + val;}

void fun(int val){val = num \* val;}

이런 식으로 공통 결합에서 변수 값이 변하면 해당 변수를 사용하는 모듈이 함께 영향을 받아서 값이 변하게 됩니다.

**9주차 (5/13) “Chapter 6. 하위 설계”**

**25. 상속의 예를 프로그램으로 작성하시오.**

객체지향에서 상속은 상위 클래스의 모든 것을 하위 클래스가 물려받아 자신의 것처럼 사용하는 것을 말합니다. 상속을 사용했을 때 클래스 전체 구조를 파악하기 용이해지고 기존 클래스를 상속받아 재사용성이 증대되며 동시에 확장에도 용이, 수정이 용이해 유지보수에 좋고, 추상화 단계를 표현하기 좋아 많이 사용되는 개념입니다. 상속의 예를 저희 팀 프로젝트의 프로그램으로 예를 들어보면 회원 정보를 입력하는 클래스를 부모 클래스로 만들고 난 후 이 클래스를 상속 받고 회원 정보 수정, 삭제 기능을 추가한 자식 클래스로 생성하여 상속의 이점을 활용하는 경우가 있겠고, 또 혼잡도 분석 데이터를 입력 받는 클래스를 부모 클래스로 두고 이를 상속 받는 데이터를 입력 받으면서 계산까지 하는 자식 클래스를 생성하는 것도 상속의 예로 들을 수 있겠습니다.

**26. 다형성의 예를 만들고 설명하시오.**

다형성은 같은 코드의 메서드가 객체에 따라 다르게 동작하고 서로 다른 코드를 제공하는 것으로 변경과 확장이 용이하고 유지보수에도 좋다는 장점이 있습니다. 다형성의 예를 팀 프로젝트 프로그램으로 예를 들어보면 혼잡도를 계산하는 클래스를 식당 혼잡도를 계산하는 클래스, 라운지 혼잡도를 계산하는 클래스, 도서관 혼잡도를 계산하는 클래스 등으로 재정의해서 혼잡도를 계산한다는 내용은 동일하지만 서로 다른 수치의 식으로 값을 계산합니다.

**27. 단일 책임의 원칙을 적용하기 전과 후가 어떻게 달라지는지 설명하시오.**

좋은 설계를 위해서 모듈의 응집도를 높이고 결합도를 낮추는 설계입니다. 객체지향에서 클래스 하나가 단일 기능을 갖도록 설계해야 하지만 가끔 기능이 두개 존재하는 경우가 종종 발생합니다. 단일 책임 원칙은 이러한 일을 방지하고 클래스는 한가지 책임만 갖도록 설계하는 것을 말합니다. 단일 책임의 원칙을 적용하기 전에는 두개의 기능이 존재하는 모듈이지만 단일 책임 원칙 적용 후 오직 하나가 되도록 설계하여 응집도를 높이고 결합도를 낮추도록 설계합니다.

**28. 개방 폐쇄의 원칙을 적용하기 전과 후가 어떻게 달라지는지 설명하시오.**

개방 폐쇄의 원칙을 지키지 않으면 하위 클래스의 특정 기능을 상위 클래스에서 미리 구현하는 것과 같습니다. 이렇게 되면 이 기능이 필요 없는 다른 하위 클래스에서 강제로 상속받아야 하는 경우가 발생하게 됩니다. 따라서 개방 폐쇄의 원칙을 적용하여 자유로운 상속을 통해 확장은 쉽게, 변경은 어렵게 하여 변하는 것과 변하지 않는 것을 구분하고 확장이 필요한 곳은 확장되기 쉽게, 폐쇄가 필요한 곳은 변화에 영향을 받지 않도록 설계합니다. 변경이 필요한 경우 기존 코드를 변경하지 않으면서 상속과 확장을 통해 변경할 수 있게 설계 하는 것이 중요합니다.

**29. 의존 관계의 원칙을 적용하기 전과 후가 어떻게 달라지는지 설명하시오.**

의존 관계 원칙을 적용하기 전에는 클래스를 설계할 때 상속의 개념을 사용하는 경우가 많은데, 상속 개념을 적용할 때 구체 클래스에서 상속을 받는 구조로 설계할 수 있는데 이를 막는 방법이 의존 관계 원칙입니다. 의존 관계 원칙을 적용하면 향후 변경에 따른 영향을 최소화할 수 있고, 느슨한 결합도를 유지할 수 있습니다. 따라서 변화에 따른 충격에서 자유로울 수 있도록 추상 클래스를 만들고 그 추상 클래스와 상속 관계를 유지하도록 설계해야 합니다.

**30. 인터페이스 분리의 원칙을 적용하기 전과 후가 어떻게 달라지는지 설명하시오.**

인터페이스 분리의 원칙을 적용하기 전에는 참조하려는 클래스의 메서드가 많고 사용자가 알아야 할 메서드는 너무 많은데 이럴 경우 하나의 메서드를 변경할 때라도 번거롭게 됩니다. 따라서 인터페이스 분리의 원칙을 적용하여 각 클라이언트에 맞는 인터페이스만 분리하여 사용하고 시스템을 확장할 때 그 변화의 폭을 필요한 기능에 대한 변경이나 확장으로 제한할 수 있다는 이점이 있습니다. 또한 클래스 안에 메서드가 많으면 가독성도 떨어지고 수정이 어려운 단점을 극복할 수 있습니다.

**10주차 (5/20) “Chapter 7. 구현”**

**14. 다음 코드는 어떤 표준 코딩 규칙에 어긋나는가?**

Int num; -> 변수를 초기화 해주지 않았다.

Int sum = 0;

Int sum(); -> 변수명과 함수명은 다르게 사용해야 하므로 s 를 대문자로 써야한다.

**16. 표준 코딩 규칙에 어긋나는 곳을 찾아 그 이유를 설명하시오.**

int num&sum = 0; -> 허용되지 않은 특수 문자를 사용하였다.

#define size 10 -> 매크로명에 소문자를 사용하였다.

const int hight = 20; -> 상수명에 소문자를 사용하였다.

int Sum = 0; -> 변수 명 첫 글자를 대문자로 시작하였다.

int Data(){ } -> 함수 명 시작을 동사로 하지 않았다.

**17. 명칭의 규칙을 지키지 않은 나쁜 예이다. 명칭의 규칙에 맞게 수정하시오.**

int num&sum = 0; -> int num\_sum

#define size 10 -> SIZE

const int hight = 20; -> HIGNT

int Sum = 0; -> sum

int Data(){ } -> getData

**18. 다음 코드가 포인터 변수의 명명 규칙에 맞지 않은 이유를 써보시오.**

int a = 0;

int \*sum += a; -> int \*pSum += a;

포인터 변수를 명명할 때에는 앞에 p를 붙이고 이후 앞 글자를 대문자로 쓴다.

**19. 프로그램에서 중괄호({}) 가 원시 코드 형식에 관한 규칙을 잘 지킨 예를 써보시오.**

int main(){

int a = 5;

if(a>4){

printf(“Hello World!!”);

}

}

**20. 다음 문장을 ‘수식에 관한 규칙’에 맞게 고치시오.**

1) a + + -> a++

2) a=b+c+d; -> a = b + c + d;

3) a > b ? x : -x; -> (a > b) ? x : -x;

4) sum = kor + (++eng)

-> ++eng;

sum = kor + eng;

5)if(a==-&&b==0) -> if((a==0)&&(b==0))

6) sizeof(a)

**21. switch문에서 break문이 빠졌다면 어떤 문제가 발생하는지 설명하시오.**

switch 문 안에 break 문이 빠졌다면 switch 문을 빠져나오지 못하고 다음 case문을 수행하게 됩니다.

**22, switch문에서 default문의 역할을 설명하시오.**

모든 case문에 부합하는 경우가 없을 때 마지막으로 default 문에서 처리를 해줍니다.

**23. 다음 코드의 문제점을 설명하고, 수정하시오.**

float i = 1.1

for(i = 1.1; i < 2.5 ; i + 0.1) -> for문을 사용할 대에는 정수로 다뤄야 한다.

-> int i = 1;

for(i = 1; i < 3; i++)

**24.; 다음 코드의 문제점을 설명하고, 수정하시오.**

int i = 1, j = 10 -> 세미콜론이 누락되었음

for(i = 0; i < 10; i++)

i = I + j; -> for에서 카운트로 다루고 있는 변수를 계산에서도 다루고 있음

-> int i = 1, j = 0;

for(j ; j < 10; j++)

i = I + j;

**25. 다음 코드의 문제점을 설명하고, 수정하시오.**

int I = 0;

while(1) {

if(i == 0) {

break;

}

else if(i == 20) {

break;

}

}

* 반복문을 중단하기 위해서 break문을 한번만 사용해야 합니다.
* 마무리를 else if 로 끝맺으면 안됩니다. If 하나만 사용하던가 else로 마무리 해야 합니다.

int i = 0;

while(1) {

if((i == 0) || (i ==20)) {

break;

}

}

11주차 (5/21) “Chapter 8. 테스트”

**23. 동등 분할 기법의 예를 만들어 설명하시오.**

교내 시설 혼잡도 프로그램에서 동등 분할 기법을 적용해 보면 먼저 시설 별 혼잡도 데이터를 입력 받은 후 이 데이터를 분류하여 시설 좌석의 50% 미만 시에 쾌적, 50%이상 70% 미만 시 보통, 70% 이상 100%이하 시 혼잡, 100% 초과 일 시(자리가 전부 찼는 대도 내부에 사람이 있을 경우 ex. 기다리거나 대기중인 사람) 포화 상태로 상황을 분류하여 적용합니다. 이후에 입력 한 값에 대해 예상되는 원하는 값이 제대로 출력되는지 실제 값과 비교하면서 테스트를 진행합니다. 동등 분할 기법은 단순하고 이해가 편하기 때문에 사용자도 쉽게 작성이 가능하다는 장점이 있습니다.

**24. 경계 값 분석 기법의 예를 만들어 설명하시오.**

프로그램들 중 경계 값을 이상, 이하, 미만, 초과와 같은 표현으로 정확하게 표현하 지 못하고 애매하게 경계 값을 나누어 놓는 실수를 자주 범하는 경우가 있다. 이러한 실수를 사전에 방지하기 위해서 경계 값 분석 테스트를 활용하여 경계에 있는 값, 즉 경계 값과 경계 이전 값, 경계 이후 값을 가져와 테스트 데이터로 생성하여 테스트하는 방법을 시행합니다. 앞선 교내 시설 혼잡도 프로그램의 예를 가져와서 표를 작성하였습니다.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 테스트 케이스 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 입력 값 | -1 | 0 | 1 | 49 | 50 | 51 | 69 | 70 | 71 | 99 | 100 | 101 |
| 예상 결과 | 오류 | 쾌적 | 쾌적 | 쾌적 | 보통 | 보통 | 보통 | 혼잡 | 혼잡 | 혼잡 | 혼잡 | 포화 |
| 실제 결과 | 오류 | 쾌적 | 쾌적 | 쾌적 | 보통 | 보통 | 보통 | 혼잡 | 혼잡 | 혼잡 | 혼잡 | 포화 |

이렇게 경계 값 분석 기법은 동등 분할 기법과 유사한 면이 있지만 영역 내의 임의의 값이 아닌 범위의 경계 값 을 테스트 케이스로 적용한 다는 차이점이 있습니다.

**25. 원인-결과 그래프 기법의 예를 만들어 설명하시오.**

앞선 예제인 동등 분할 기법과 경계 값 분석 기법은 입력 한경의 복합성을 완전하게 고려하지 못한 경우입니다. 이러한 단점을 보완하기 위해서 원인- 결과 그래프 기법을 사용합니다. 원인 결과 그래프 기법은 여러 입력 조건을 결합해서 하나 이상의 결과를 도출해 내어 상단의 단점을 극복합니다.

원인-결과 그래프 기법의 과정으로는 먼저 프로그램을 적합한 크기로 분할하고 이후 명세서에서 원인과 결과를 찾습니다.

|  |  |
| --- | --- |
| 원인 | 결과 |
| 원인 1: 혼잡도  원인 2: 예약 유무 | 결과 1: 혼잡도 수준 별로 차등 출력  결과 2: 예약 유무에 따라 차등 출력 |

원인은 하나의 입력 조건이 되고 결과는 출력 조건이 되게 됩니다. 이후 원인-결과 그래프를 작성하게 됩니다.

예약 X

포화

혼잡

보통

쾌적

**^**

예약 O

100이상

70이상

50이상

혼잡도

50미만

명세서가 의미하는 내용을 분석하여 원인과 결과를 올바르게 연결하는 논리그래프를 작성하게 됩니다. 이후 그래프에 제한 조건을 제한 기호를 이용해 표시하고 의사 결정 테이블로 변환하게 됩니다. 의사 결정 테이블에서 우선 입력할 대상 목록을 정하고 입력 항목에서 가능한 조합들을 생성합니다. 교내 시설 내 혼잡도 분석 시스템에 원인-결과 그래프 기법을 적용하여 혼잡도 기준과 자리 예약 유무를 조합한 테스트 케이스를 작성하였습니다.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 총점 | 100이상 | T | T | F | F | F | F | F | F |
| 70이상 | F | F | T | T | F | F | F | F |
| 50이상 | F | F | F | F | T | T | F | F |
| 50미만 | F | F | F | F | F | F | T | T |
| 예약 유무 | 예약 | T | F | T | F | T | F | T | F |

이후 예약이 가능한 좌석이 존재하는 시설인 경우 자리가 공석 이어도 이용이 불가능 하기 때문에 예약이 존재하지 않는 시설보다 한단계 더 높은 혼잡도를 가지도록 설정하였습니다.

그 다음은 기준에 맞춰서 결과 값 그래프를 작성하였습니다.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 총점 | 100이상 | T | T | F | F | F | F | F | F |
| 70이상 | F | F | T | T | F | F | F | F |
| 50이상 | F | F | F | F | T | T | F | F |
| 50미만 | F | F | F | F | F | F | T | T |
| 예약 유무 | 예약 | T | F | T | F | T | F | T | F |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 결과 값 | 포화 | T | T | T | F | F | F | F | F |
| 혼잡 | F | F | F | T | T | F | F | F |
| 보통 | F | F | F | F | F | T | T | F |
| 쾌적 | F | F | F | F | F | F | F | T |

12주차 (6/03) “Chapter 9. 품질”

**21. 표준 프로세스의 필요성에 대해서 설명하시오.**

저희 조 팀프로젝트 주제인 시설 별 혼잡도 분석 시스템에서 포준 프로세스의 필요성을 찾아보았습니다. 먼저 표준 프로세스는 소프트웨어 개발에서 레시피와 같은 역학을 수행하는 프로그램을 개발하기 위한 매뉴얼입니다. 해당 혼잡도 분석 시스템이 사용자의 요구사항에 맞게 정상적으로 구동되도록 하기 위해, 조직에게 일관된 표준을 제시하여 작업의 속도와 생산성을 효율적이게, 명확한 방향을 제시해 주고 계획을 세울 수 있기에 표준 프로세스는 시설 별 혼잡도 분석 시스템에 필수적입니다.

**27. CMMI에서 관리 단계와 정의 단계를 일반적인 예를 들어 설명하시오.**

성숙도 단계별 프로세스 영역 중 2단계인 프로세스의 관리 단계에는 요구 사항 관리, 프로젝트 계획 수립, 프로젝트 감시 및 통제, 협력 업체 관리, 측정 및 분석, 프로세스/제품 품질 보증, 형상 관리가 있고 3단계인 정의 단계에는 요구 사항 개발, 기술적 솔루션, 제품 통합, 검증, 확인, 조직 차원의 프로세스 개선, 조직 차원의 프로세스 정의, 조직 차원의 교육 훈련, 통합된 프로젝트 관리, 위험 관리, 의사결정 분석 및 해결이 있습니다. 여기서 저희 팀 프로젝트인 시설 별 혼잡도 분석 시스템을 적용하여 예를 적용해 보았습니다.

먼저 2단계에서 관리 단계에서의 전사 차원의 표준 프로세스는 없지만 개별 프로젝트의 프로세스를 정의하여 분석 시스템의 프로젝트 예산, 일정, 기능을 예상하고 추산합니다. 정의만큼 체계적이지는 않지만 적용한 프로세스에 따라서 이 프로그램이 사용자가 요구를 만족시키는지를 확인, 분석합니다. 이후 프로젝트 정책을 세워 혼잡도 시스템의 기능들을 어떻게 구현할 지 설정하고 혼잡도 시스템의 기능을 명확히 문서로 작성하여 구현이 가능하도록 해줍니다.

3단계의 정의 단계에서는 전 단계에서 대략적으로 구성하고 설계하였던 프로세스를 더 체계화하고 표준화 하여서 조직 전체가 표준 프로세스에 따라 분석 시스템을 개발할 수 있도록 가이드라인을 제시합니다. 이후 2단계에서 정의하였던 개발 공정, 비용, 일정 등을 더 구체적으로 통제가능 하도록 일관성 있는 프로세스를 구축합니다.

**29. 품질 보증의 효과와 문제점에 대해 설명하시오.**

품질 보증이란 소프트웨어의 오류와 결함을 제거해 나가면서 더 좋은 품질의 소프트웨어를 만드는 과정으로 저희 팀 프로젝트 시스템을 예로 들면 혼잡도 정보를 검색 조회할 때 조회 속도가 사용자 요구사항에 부합하는 속도를 출력해낼 수 있어야 하고 데이터를 다룰 때 사용하는 자원을 요구 명세서에 작성되어 있는 기준 이상을 충분히 만족 시킬 수 있도록 하기 위해 개발 이전에는 시스템 품질에 영향을 끼치는 문제점들을 발견하여 제거하고 개발 이후에는 목표한 수준의 품질을 달성하고 유지하고 있다는 것을 보증해 주는 것을 말합니다. 이러한 품질 보증은 요구사항을 명확히 정립할 수 있어 프로그램 이해도가 증진되고 표준화를 유지하여 문제점 발견 시 제거와 재발 방지에 도움을 줍니다.

품질 보증의 문제점으로는 품질 보증에 대한 사람들의 인식이 부족하여 잘 인지하지 못하고 품질 요원의 중요성을 잘 깨닫지 못하는 이러한 배경에서 경험 많은 품질 보증 요원의 숫자가 부족해지게 되었습니다. 또한 품질 보증에 대한 제도의 표준화 정립이 부족하고, 절차 확립이 미비하다는 문제점을 가지고 있습니다.

13주차 (6/08) “Chapter 10. 프로젝트 관리”

**20. 변경 관리의 필요성을 설명하시오.**

소프트웨어는 시간이 지남에 따라 어떠한 이유에 의해서 변경되게 됩니다. 지속적인 버전 업에 의해서, 혹은 예상치 못한 치명적인 오류로 인해서 소프트웨어는 끊임없이 변하게 되는데 이러한 소프트웨어의 변화를 상정 범위내에서 제대로 통제하기 위해서 관리가 필요한데 이것이 변경 관리입니다. 변화가 필수불가결한 세상을 반드시 따라가야만 하는 소프트웨어와 프로젝트는 반드시 세상의 시스템에 발맞춰 가기 위해서 같이 변화하게 되고 이러한 변화는 소프트웨어 개발 생명주기의 모든 단계에서 일어나게 됩니다. 이러한 과정 속에서 변경은 프로젝트의 모든 부분에서 영향을 주게 되고 통제되지 않을 시에 엄청난 혼란을 야기시키므로 이러한 변경되는 과정을 관리하는 것은 반드시 필요합니다.

**23. 최종 릴리스되기까지의 과정을 버전 항목 트리를 이용하여 예를 들어 설명하시오.**

먼저 최종 릴리스 버전을 트리를 활용하여 설명하겠습니다.

원시 코드

V1.0

V1.1

V1.2

V1.3

V1.4

V1.5

설계 사양서

V1.0

V1.1

V1.2

V1.3

V1.4

요구 분석 명세서

V1.0

릴리스 1.0

V1.1

V1.2

릴리스 2.0

V1.3

릴리스 3.0

혼잡도 분석 시스템의 요구 명세서를 만들고 1.1 -> 1.2 -> 1.3으로 버전 업 하였고 설계 사양서는 1.1 -> 1.2 -> 1.3 -> 1.4 로 버전업, 원시 코드는 1.1 -> 1.2 -> 1.3 -> 1.4 로 버전업을 하였다고 가정하고 릴리스는 총 3.0까지 출시하였습니다. 여기서 릴리스 1.0을 출시하기위해 요구 분석 명세서 1.1 -> 설계서 1.2 -> 원시코드 1.1을 사용하였습니다. 이후 릴리스 버전의 버전 업을 위해 계속해서 새로운 각각 요구 분석 명세서, 설계서, 원시코드의 새로운 버전을 차용합니다.

**27. 형상 통제에 대해 설명하시오.**

프로젝트를 진행하면서 사용자의 요구사항은 수시로 바뀔 수 있습니다. 이러한 변경은 프로젝트를 진행하면서 큰 문제인데 이러한 빈번한 변경을 정해진 양식에 맞춰 작성하고 이러한 변경사항 중에서 수용하거나 거절할 것을 구분하고 수용한 사항들을 승인. 검토하여 변경을 통제하는 과정을 형상 통제라고 합니다. 예를 들어 현재 팀 프로젝트의 주제인 시설 별 혼잡도 분석 시스템에 대해서 사용자가 사용자 닉네임 사용과, 대학교 인근 시설 혼잡도 분석도 추가하고 싶다고 요구 사항을 변경하였을 때 양식에 맞춰서 변경 요구 사항을 정리하여 작성한 후 형상통제위원회에서 제출합니다. 사용자 닉네임 사용 변경 사항에 관해서는 현재 상황에서 충분히 합리적으로 수행 가능하다고 판단하여 수용하였고, 대학교 인근 시설 혼잡도 분석 추가 요구 사항은 현재 상황에서 비용과 시간적으로 어렵다고 판단하여 거절합니다. 이렇게 사용자의 요구 사항이더라도 가능한 요구사항인지 구분하여 통제하는 것을 형상 통제라고 합니다.

**29. 형상 관리의 효과에 대해 설명하시오**

수시로 변경되는 소프트웨어의 프로젝트를 효과적으로 관리하기 위해서 일련의 과정에 따라서 형상 관리를 하게 됩니다. 형상 관리를 통해서 소프트웨어의 급작스러운 수정으로 인한 오류를 안정적으로 처리하거나, 사용하던 문서나 코드가 변경되거나 사라지는 상황에서 안정적으로 백업하기 위해서 사용됩니다. 이러한 형상 관리 효과로 기관 별로, 팀 별로, 업무 별로, 각각 생산된 산출물 들의 현황의 이력의 리스트를 파악하기가 편하고 이로 인해 산출물들을 관리하기가 용이합니다. 이로써 프로그램의 개발과 유지보수가 편해지고 관리가 용이한 산출물들은 자연스럽게 오류와 버그가 상대적으로 적어지게 됩니다. 결과적으로 소프트웨어의 오류가 적어지게 되고 변경된 사항에 대해서 영향을 분석하기 쉬워져 높은 품질의 소프트웨어를 얻을 수 있게 됩니다.