문제는 키워드 추출 혹은 앞에서 몇 단어를 중복없이 따오는 방식으로 요약

맨 마지막 장에는 축약한 영단어가 원래 어떤 단어인지 설명

개같은거.. 즐거웠ㄷ…ㅏ…

1.a.UML? UML 이란 SA 에서 사용되는 표준화된 범용 모델링 언어로 객체지향 소프트웨어 집약 시스템을 개발할 때 산출물을 명세화, 시각화, 문서화할 때 사용한다. 이를 통해 PRJ 의 전체적인 구성을 시각화하여 한 눈에 파악할 수 있게 해주어 잘못 설계된 부분이나 모호하게 설계된 부분을 찾아 수정하는데 용이하고 결과적으로 더 완벽한 SW 를 설계하기 위한 검증에 사용. 경제적인 효율이 높음. 1.b.diagram? A. Use case? usr 와 sys 간 상호작용을 도식화한 그림. 설계하려는 시스템의 high level view 를 제공하여 실제 그 sys 가 어떤 일을 하는지를 간단 명료하게 보여줌. 상호작용의 주체가 되는 usr 가 들어가고 여러가지 원, 타원, 직선으로 구성. B.Sequence sys 내 obj 들 간의 상호작용을 시간 순서에 따라 도식화. 기능의 종류에 따라 다양한 형태의 결과물. 상단에 obj 나열, 좌우방향 화살표를 통해 상호작용 표현, 아래로 내려가면서 시간순서. 어떤 function 의 흐름을 표현. 구성한 기능이 작동하는 흐름에 따라 그리면서 중간에 빠뜨린 부분이 없고 잘 작동하는지 점검 및 예측. C.Class class 내부의 정적인 내용이나 class 사이의 관계를 표기. sys 일부 or 전체의 구조를 나타냄. 의존관계를 명확히 보게 해주고 순환의존이 발생하는 지점을 찾아내 어떻게 고리를 깨는 것이 가장 좋은 지 결정할 수 있게 해줌. class 와 stereo type 르고 구성, 여러 개의 상자와 화살표로 그림. D. State PRG 의 상태와 서로간의 상태천이를 표현하기 위해 도식화. PRG 이 가질 수 있는 상태들을 노드로 표현, 상태들 간에 발생할 수 있는 천이를 화살표로 표현, 그 원인이 되는 기능 표시. PRG 진행과정을 표현하고 코딩하면 오류를 줄임. 1.c draw sq, class diagram? 안해 TE5.4 ~ 5.8 안해 3.a SA? SW 의 구성요소들 사이에서 유기적 관계를 표현하고 SW 의 설계와 업그레이드를 통제하는 지침과 원칙. 3.b important? performance, robustness distributability, maintainability 에 영향을 줌. 3.c invariants of SA pattern? Layered : set of layers(or abstract machine) each of which provide a set of services Repository : sub-systems must exchange data Client-server : distributed system model which shows how data and processing is distributed across a range of components. Pipe-filter : functional transformations process their inputs to produce outputs 3.d apply your PRJ Layered pros: 인접한 계층간의 통신을 이해 쉬움. 예를 들어, 우리는 user interface layer 에 위치한 'Login'이 functionality layer 에 위치한 'Security Management'와 관계있다는 것을 직관적으로 앎. cons: right levels of abstraction 을 결정하는 것이 애매. 예를 들어, 'Exchange a message'(2^{nd} layer)는 몇몇 기능적 요소를 가짐. 이러한 것은 3^{rd} layer 에 보통 위치하지만, messaging 은 기능적 요소만을 가지고 있다고 보기 매우 모호, messaging 은 2^{nd} layer 에 포함될 요소 역시 가지고 있기 때문. Pipe and Filter pros: 포괄적이고 이해하기 쉬움. 사용자가 시스템의 data flow 를 쉽게 인지. Easy to understand and supports transformation reuse. Workflow style 은 많은 비즈니스 프로세스의 구조들에 일치한다. Evolution by adding transformations is straightforward. 순차시스템이나 동시(concurrent) 시스템으로 구현 가능. cons: "Pass Message" 같은 상호 작용하는 기능들은 Pipe and Filter pattern 으로 나타내기가 힘듦. "Manage profile"과 "grade"도 마찬가지다. 데이터 전송을 위한 형식은 다음과 같이 합의되어야함. 통신 변환. 각 변환은 입력을 구문 분석하고 합의된 형식으로 출력을 파싱. 이로 인해 시스템 오버 헤드가 증가하고 호환되지 않는 데이터 구조를 사용하는 함수 변환을 다시 사용할 수 없음을 의미. Repository : pros: 구성 요소는 독립적. 다른 구성 요소의 존재를 알 필요가 없음. 한 구성 요소에 의해 변경된 사항은 모든 구성 요소로 전파가능. 한 곳에서 모든 데이터를 일관되게 관리 가능(예 : 동시에 완료된 백업).cons: 저장소는 단일 실패 지점이므로 저장소의 문제가 전체 시스템에 영향. 저장소를 통한 모든 통신 구성에 비효율적. 저장소를 여러 컴퓨터에 분산시키는 것은 어려울 수 있음. Client-Server : 각 서비스들이 존재하는 서버가 있고, client 는 이 서비스들을 이용하기 위해 서버에 접근하는 유저. cons: 서버가 network 를 통해 서비스를 제공할 수 있고 general functionality 는 모든 서비스에 구현할 필요 없이 사용자에게 제공. pros: network 에 따라 성능이 결정되고 Dos 공격이나 server failure 가 발생할 수 있다. 프로젝트 규모에 비해 너무 많은 요소가 필요. -> 우리는 layered architecture 를 적용하기로 했다. TE6.4 draw conceptual view? 안해 TE6.5 cattle drone architectural? repository archi 로 중앙에 소를 기르는 것과 관련된 components 들이 존재해 드론이 이를 중심으로 소를 자동적으로 기르도록 한다. 이 때, pipe-filter archi 로 소를 기르는 프로세스를 다루도록 하고, 사용자의 입장에서는 layered archi 로 인터페이스 수준에서의 드론에 대한 원격 제어를 실행하도록 하고, 드론의 기능적인 부분은 layer 의 하층인 system 수준에서 이루어지도록 archi 들을 복합적으로 사용할 수 있을 것이다. TE6.6 iTunes? client-server 구조. 여러 사용자가 음악을 듣고 파는 동일한 서비스를 동시다발적으로 사용할 수 있어야 하고, 이러한 것에 대해 client-server archi 가 적합. 5.a design pattern? 특정 문맥에서 공통적으로 발생하는 문제에 대해 재사용 가능한 솔루션 템플릿. 5.b important? 공통적인 문제 해결. 자신이 사용한 패턴을 설명함으로써 디자인 설명 가능. 5.c example? observer pattern 은 한 객체의 상태를 여러 방식으로 display 해야하는 상황에서 사용가능. 이 패턴은 다른 형식으로 보여줘야하는 객체를 분리. 패턴 설명은 대개 상속 및 다형성과 같은 객체지향적 특성을 이용. 5.d your prj? iterator pattern 을 적용해 searchPhoto 와 searchPhotographer 기능을 구현, 재사용 가능, 다른 부분에 영향을 주지 않으면서 작은 수정이 가능. 6.a open-src development? SW sys 의 src code 가 공개되고 volunteer 가 개발 과정에 참여할 수 있는 sw 개발 방식. 6.b pros cons your prj? pros : 싸고 빠르게 개발, 신뢰성 있는 sw sys 만듦. cons : open-src sys 과 기존 sys 호환 안될수있음. 6.c risk ur prj? 우리 prj 와 호환 가능성이 아예 없거나 호환할 수 있도록 코드를 수정하는 비용이 클수도. TE7.3 7.6~8 안해 8.a validation verification? 밸리: are we building the right product? sw 는 사용자들이 원하는 것을 제공해야함. 베리: are we building the product right? sw 는 사양과 일치해야함. 8.b development testing approach? unit : 각각의 prg unit 또는 obj class 를 독립적으로 검사, obj 나 method 의 functionality 를 검사하는데 초점 Component : 몇 개의 unit 들이 합쳐져 만들어진 composite component 를 검사, component interfaces 를 검사하는데 초점. Sys : 몇 개의 또는 전체의 component 가 합쳐진 시스템을 전체적으로 검사,component interactions 를 검사하는데 초점. 9 suggest 8 test case? 표 10. provide 4 independent paths? 1-2-8-10-11 1-2-3-8-9-11 1-2-3-4-5-7-2-8-9-11 1-2-3-4-6-7-2-3-8-10-11 cyclomatic complexity = number of edge - number of node + 2 = 14 - 11 + 2 = 5 TE9.4 discuss how those subsys? (low-quality : expensive to maintain.) Off-the-shelf system 으로 replace .(off-the-shelf : 이미 만들어져 있는 것.) TE9.5 strategic options for legacy sys? 총 4 가지의 options / 1. scrap the system completely : system 이 business process 에 효과적인 기여를 못할 때 chosen. / 2. Leave the system unchanged and continue with regular maintenance : system is still required 하지만 (system 이 꽤 안정하고, user 들이 상대적으로 few change request 를 요구할 때) / 3. Reengineer the system to improve its maintainability : system quality 가 change 에의해 degraded 되었는데, new change 가 여전히 요구되고있을 때 / 4. Replace all or part of a system with a new system : factors(old system 이 operation 할수없는 new H/W 같은) 또는 off-the-shelf system 이 new system 개발을 합리적인 비용으로 가능하게 할 때. TE9.6 problems with support sw? Legacy system rely on support S/W (OS 부터 system 개발에 쓰이는 compilers 까지). These may be obsolete and no longer supported by their original providers. TE9.7 As a sw prj manager ~ 1:Program, data complexity : Larger number of interfaces and more complex these interfaces-> maintenance cost high 2:Number of requests for corrective maintenance : 일정 시간동안 일어난 bug and failure reports. -> 클수록 maintainability 낮을 것. TE9.8 Briefly describe 3 main types~ 1. Corrective maintenance : fault rapairs to fix bugs and vulnerabilities 2. Adaptive maintenance : environmental adaptation to adapt the sw to new platforms and environment. 3. Perfective maintenance : functionality addition to add new features and to support new requirements. 이유 : because the same set of changes may cover all three types of maintenance. TE9.9 diff between sw reengineering & refactoring? Reengineering 은 system 이 한동안 유지되고나서, 유지비용이 점점 증가할 때 함. Refactoring 은 improvement 를 위한 continuous process, throughout the development and evolution process. TE9.10 do sw engineers ~ Yes sw development and maintenance are not separate activities. 12. draw component dia of ur prj 안해 13.a If a "prg" is defined ~ web ~ web 으로 하려면 usr interface, browser engine, rendering engine 을 고려해야함. 13.b the following dia defines interaction ~ 1. adaptor 가 없어서 component 간의 incompatibility 가 발생한다면 해결할 수가 없기 때문에 중간에 adaptor 를 추가해주는 것이 좋음. 예를 들어 Conference reservation 에서 social security number 같은 것을 추출할 수 있는 SSNstripper 같은 adaptor 가 있어야함. 2. 어느 하나의 데이터만 수정하고 싶은 경우 효율성이 떨어짐. 그림에서 airline reservation 의 provide component 가 hotel reservation 에 require component 로 작용하는데 만약 airline reservation 단계의 데이터를 수정하고 싶은 경우 hotel reservation 까지 영향. TE16.1 hinder sw reuse? 재사용의 문제점 : 어떤 것을 재사용할 것인지 선정, sys 에 공통적으로 사용되는 요소들 발견해야함, prg 의 표준화가 부족, 새로운 개발 방법론 도입 어렵, 재사용을 위한 관리 및 지원 부족, 기존 sw 에 재사용 sw 추가 어렵 / 나는 대부분의 프로그램에 공통적으로 사용되는 모듈들만 component 로 만들어서 재사용. 나머진 이식이 힘들어서 사용 x TE16.5 weather station prd line archi ~ 안해 TE16.8 COTS risk 6 1. 소프트웨어 제작자의 도움이 필요할 때 도움을 주지 못할 수 있다. 2. 다른 system 의 data 나 event 와 incompatible 3.기존의 system 과 integrate 했을 때 오작동의 가능성이 있다. 4. 기존의 작동 환경과 호환이 되지 않을 수 있다. 5. COTS 의 구조를 이해하고 기존의 시스템에 integrate 하는데 시간이 필요하다. 6. Software 의 업데이트가 어렵다. TE16.9 COTS integrate 3 problem COTS 는 다른 개발자에 의해 다른 시간에 개발되었고 당연히 integration 에 관련된 것들을 고려하지 않고 개발되었다. 따라서 adaptor 가 필요. Missing information : 어플리케이션 a 가 b 에게 어떤 정보를 요구하는데 b 는 그 정보가 필요하지 않은 경우, a 가 제대로 작동하지 않을 수 있음.(필요한 정보가 충족되지 않을 수) Control incompatibility : A 는 유저의 인풋에 대해 reactive 하고 B 는 Proactive 하고 workflow based, predefined interaction 을 사용하는 경우처럼 control policy 가 다른 경우 control incompatibility 가 발생한다. Semantic matches : 서로 다른 어플리케이션이 실제로는 다른 데이터에 대하여 같은 이름을 사용하는 경우

TE17.6 ~ component src code ~ reusable component ~ Source code 가 없으므로 이 프로그램이 어떻게 exception 을 handling 하는지 알 수가 없어서 validation 이 힘듦. Formal component specification 은 아주 정확하고 자세하게 component 에게 기대하는 작업을 기술하고 있으므로 그것을 실제 component 가 하는 것과 비교하는 방식으로 validation 에 도움을 줄 수 있. TE17.7 memtcare sys patient ~ design ~ 안해 TE22.1 why the intangibility ~ the manager of a shipbuilding project or of a civil engineering project can see the product being developed. If a schedule shlips, the effect on the product is visible-parts of the structure are obviously unfinished. SW is intangible. It cannot be seen or touched. SW prj managers cannot see progress. They rely on others to produce the documentation needed to review progress. TE22.2 explain how company size ~ Company size 와 sw size 가 크면 sw prj management 는 더욱 복잡. Company size 가 크다면 Formal policies and management structure 가 필요하다 그러면 management overhead 가 큼. Formal reporting and budgeting and approval process. sw size 가 작으면 작은 개발팀으로 개발이 가능하므로 같은방에 함께 진행 및 management 이슈들을 논의 가능하지만 일반적으로 여러 개발팀 필요하므로 prj manager 는 팀의 활동을 서로 소통할 수 있도록해야함. TE22.4 in addition to the risks ~ Technology: Communications network saturates before expected transaction limit is reached. People: Level of skill of available people is lower than expected. Organizational: Organizational changes mean that the prj schedule is accelerated. Tools: SW tools cannot handle the volume of data available for large systems. Requirements: New non-functional requirements are introduced that require changes to the system architecture. Estimation: The difficult of the sw is underestimated. Time: It takes more time than I expected to develop. TE22.5 risk monitoring? Risk monitoring 은 제품, 프로세스 및 비즈니스 위험에 대한 가정이 변경되지 않았는지 확인하는 과정. 프로젝트의 모든 단계에서 정기적으로 위험을 모니터링 해야함. 모든 management review 에서 주요 위험을 각각 고려해야함. + 표 TE22.6 fixed-price ~ Fixed price contracts increase the chances of product risks because they remove options from the development process. Because the contract is fixed-price, the contractor is naturally reluctant to increase the effort or time expended on the prj as this will reduce their profits on the work. Therefore, if problems arise they will look for ways to reduce the scope of the product or to reduce the costs of product development (e.g. by reducing the effort devoted to testing). Both of these factors can lead to products that are not as expected by the customer. TE22.9 write a case study ~ 프로젝트 담당자가 자신의 업무가 무엇인지, 또는이를 수행하는 방법을 모르는 경우 전체 프로젝트가 중단됨. 프로젝트 직원이 무엇을하고 있는지 모를 경우 프로젝트 진행 상황을 모니터링 할 수 없게됨. 또한, 고객이 당신에게 기대하는 바를 모르는 경우에는 프로젝트가 시작되지 않음. 프로젝트에서 커뮤니케이션 매우 중요하고 필수적으로 잘 해야함. 16.a 5 risk ur prj? 다들할지 16.b how to motivate ur team? 각자 책임감을 갖고 역할을 분배, 정해진 부분을 최대한 하도록 장려, 팀원들간 소통을 통해 동기부여. 17.a draw an activity ~ 그림 17.b list critical path ? T1-T3, T2-T6 TE23.2 explain why the process~ 실제로 우리가 프로젝트를 실시할 때에는 그때에 가능한 정보로만 개발. 막상 개발을 시작하면 프로젝트와 관련이 되어있지만 개발할 당시 알 수 없었던 일이 발생. 프로젝트가 진행됨에 따라서, 프로젝트와 관련된 정보들이 계속해서 늚. 따라서 우리는 프로젝트 계획을 반복적으로 개정해야 할 필요. 또한 프로젝트가 진행됨에 따라서 개정되어야 할 정보들이 프로젝트 계획에 추가될 것이며, 프로젝트를 모니터링하는 데에 도움을 받기 위해서는 이러한 변화를 계속해서 업데이트 해야함. TE23.3 define prj scheduling. a mechanism to communicate what tasks need to get done and which organizational resources will be allocated to complete those tasks in what timeframe.(정의-영어로) / 우선 프로젝트는 여러가지 task 로 이루어져 있고 개인에게 할당된 task 는 다 다름. 스케줄은 a start 와 end 가 주어지기 때문에 그 기한을 넘기지 않는 것이 중요하며, 개인의 휴가나 일정이 충분히 기록. 그래서 online prj scheduling tools 를 사용. (1)무엇이 스케줄 되는가? (What is being scheduled) (2)누가 끝낼 것인가? (3)스케줄이 언제 끝날 것인가? 에서 시작. 그 이후에, WBS 를 사용하면서 프로젝트 내에서 해야 하는 activity 를 정의. 이후 각 activity 를 하는데 사용될 시간과 노력을 계산. 스케줄을 정확하게 맞추기 위해서 꼭 필요한 단계. 특별히 각 activity 가 서로 관련이 되어있다는 사실을 명심. 이는 task dependency 라고 불리는데, 스케줄은 반드시 이 연관성에 대해서 반영. 또한 어떤 자원이 필요할지에 대해서 계획해야 prj scheduling 을 지킬 수 있음. TE23.4 what is algorithmic cost modeling? (1)다른 측정기법을 사용하여 많은 independent estimates 를 확보. 만약 그들이 다양하다면, estimates 가 수렴할 때까지 비용 정보를 반복. (2)측정하기 어려운 시스템의 경우에는 발생할만한 문제를 찾기 위한 프로토타입을 발전 (3)필요한 측정량을 감소할 수 있고, 전체 비용을 감소할 수 있는 소프트웨어를 재사용. (4)고정비용에 포함되어 있는 시스템 기능의 개발에 필요한 비용을 측정. 따라서 그 측정을 가능하게 하는 디자인을 채택. (5)소프트웨어 요구사항을 중요하고 바람직한 'gold plating'으로 분할. 필요시 'gold plating'을 제거. TE23.5 안해 TE24.1 define the reams quality ~ Quality Assurance: (1)processes & standards that should lead to high-quality products / (2)introduction of quality processes into the manufacturing process / Quality control: the application of these quality processes to weed out products that are not of the required level of quality. / QA, QC 모두 Quality management 의 부분. / 험프리 제시 5 개. / (1)Product introduction, (2)Product plans , (3)Process descriptions, (4)Quality goals, (5)Risks and risk management TE24.2 explain how standards ~ (1)Knowledge of specific types of fault that commonly occur in the type of sw developed by an organization. This might be encapsulated in a standard review checklist. (2)Knowledge of the types of system model that have proved useful for sw maintenance. This can be encapsulated in design documentation standards. (3)Knowledge of tool support that has been useful for a range of projects. This can be encapsulated in a standard for a development environment to be used by all projects. (4)Knowledge of the type of information that is useful to include as comments in code. This can be encapsulated in a code commenting standard. TE24.3 discuss the assessment ~ 우리조는 reliability understandability usability 를 채택해서 했음. TE24.6 what happens during sw quality review ~ 프로젝트 결과물의 퀄리티(quality)를 확인하는 퀄리티 보증 활동으로, 잘못된 작성, 에러 등을 검토하기 위해 문서나, 소프트웨어를 체크, 프로세스 기록을 체크를 하게 된다. TE24.8 what is sw metric? SW Metric 은 객관적으로 측정 할 수있는 소프트웨어 시스템, 시스템 문서 또는 개발 프로세스의 특성. 1. Control Metric : 프로세스 관리지원 예) 보고된 결함을 수리하는데 드는 평균적 노력과 시간 2. Predictor Metric : 소프트웨어의 특성을 예측하는데 도움 예) cyclomatic complexity of a module TE25.1 suggest 5 possible problems ~ 잘못된 버전의 시스템을 수정하거나, 잘못된 버전을 배포하거나, 특정 버전의 시스템 또는 구성요소에 대한 소프트웨어 소스코드의 위치를 까먹거나, 무엇을 수정했는지 모르겠거나, 팀이 같이 작업을 할 수 없게 한다. TE25.2 in version management ~ codeline: 구성 요소에 연관된 sw 구성 요소 및 기타 구성 항목의 버전, baseline : sys 를 구성하는 구성 요소 버전 모음. TE25.3 Imagine a situation ~ 용어의 통일성 위반, 해당 자료의 충돌, 해당 자료의 유실 TE25.4 SW is now often ~ 팀에게 복사된 prj repository 를 제공함으로써 가능. 필요한 기능을 복사본으로 만들 땐 pull 이 필요하고 복사본을 원본에 update 할 땐 commit, push 기능이 필요. TE25.5 Describe the difficulties that may ~ single 층나 마우스 클릭으로 완벽한 sys 을 구축할 수 있어야하는 어려움. TE25.9 6 essential feature ~ 답없음

축약어 : SA = software architecture / SW = software / sys = system / usr = user / obj = object / prj = project / prg = program / archi = architecture / src = source / ur = your / TE = textbook exercise