[기출문제]

|  |  |  |
| --- | --- | --- |
| **번호** | **기출 문제** | **비고** |
| 1 | 모바일 앱 개발의 특성과 이슈에 대하여 설명하고, 애자일(Agile)을 활용하여 모바일 개발환경에 적합한 개발방법을 제시하시오. | 관리101회 3교시 |
| 2 | 폭포수형 개발모델(Waterfall development model)과 애자일 개발모델(Agile development model)의 차이를 테스팅 프로세스(Testing process)의 관점에서 비교하여 설명하시오. | 관리99회 1교시 |
| 3 | 소프트웨어 개발 방법론 Agile Methodology의 정의, 특성, 장단점을 설명하시오. | 관리98회 1교시 |
| 4 | 소프트웨어 개발 라이프사이클에 관한 나선형(Spiral) 모델과 애자일(Agile) 방법에 대하여, 유사점과 차이점을 중심으로 비교하여 설명하시오. | 응용98회 2교시 |
| 5 | 기업의 경쟁 환경이 70년대 가격, 80년대 품질, 90년대 스피드(speed), 2000년대 이후 민첩한 대응(agile) 개념으로 바뀌었다. 스피드와 민첩한 대응의 개념 차이와 민첩하게 대응하는 기업이 되기 위한 방안을 프로세스 및 정보시스템 구축 측면에서 설명하시오. | 응용90회 3교시 |
| 6 | 소프트웨어시스템이 어떤 순서에 의거해서 개발, 운영, 유지보수되어 생명주기를 마칠 때까지의 전체적인 작업 프로세스를 모델화한 소프트웨어 생명주기 모형에 관해서 물음에 답하시오.  1) 폭포수(Waterfall)모델과 프로토타이핑(Prototyping)모델의 특징을 설명하시오.  2) 나선형(Spiral)모델을 개발 4단계 절차 중심으로 설명하시오.  3) 클린룸(Clean Room)모델을 3개의 박스 중심으로 설명하시오. | 관리83회3교시 |
| 7 | S/W개발시 폭포수형(Waterfall) 개발방법과 RUP(Rational Unified Process)에서 강조하는 반복, 점진적(Iterative & Incremontal)개발방법에 대하여 설명하고, 각 방법을 비교하시오. | 관리72회4교시 |

**[Keyword]**

**- Agile Methodology, Product Backlog, Sprint Backlog, Sprint, Daily Scrum, SCRUM,**

**XP**

**문) Agile 개발 방법론**

**답)**

1. **요구사항의 지속적인 변화와 단순 설계의 시작 기법, Agile 방법론의 개념**
   1. Agile 방법론의 정의
      * 절차보다는 사람이 중심이 되어 변화에 유연하고 신속하게 적응하면서 효율적으로 시스템을 개발할 수 있는 방법론
      * 개발과정에서의 시스템의 변경사항을 유연하게 또는 기민하게 대응할 수 있는 방법론
   2. Agile 방법론의 4가지 핵심가치 (Manifesto for Agile)
2. 프로세스나 도구에 앞서, 개인의 상호 협력을 중시
3. 포괄적인 문서에 앞서, 작동하는 소프트웨어를 중심
4. 계약 협상에 앞서, 고객과의 협력을 중시
5. 계획 준수에 앞서, 변화에 대응을 중시
   1. Agile의 등장배경
      * SW 개발 환경의 변화
      * 정보시스템의 ‘time-to-market’ 과 ‘적시배포(Release)가 중요해짐
      * 사용자의 요구가 다양해지고 수명주기가 짧아짐

**Ⅱ. Agile 방법론의 특징 및 종류**

* 1. Agile 방법론의 특징

|  |  |
| --- | --- |
| 특 징 | 설 명 |
| 가변적 요구 대응 | Predictive 하기 보다는 Adaptive |
| 고객만족 | 개발 후반부라도 요구사항의 변화를 적극반영  구동하는 SW를 고객에게 자주 전달하여 고객의 요구  사항 신속 적용 |
| 개발자 동기부여 | 개발자에게 적합한 개발환경 구성  개발자가 책임을 완수할 것으로 신뢰 |
| PM의 역할변화 | 프로젝트 관리자에서 촉진자로 변경  프로젝트 계획수립 및 통제의 책임이 팀원에게 이양 |
| Sweet Spots | 한 작업실에 5~8명의 작업자  Key User 상주: 개발자와 사용자간의 중계역할, 신속  한 피드백 가능 |
| 적용 범위 | 중소형, 아키텍처 설계, 프로토타이핑에 적합 |

* 1. Agile 방법론의 종류

|  |  |  |
| --- | --- | --- |
| 종 류 | 특 징 | 비 고 |
| XP | - 의사소통 개선, 즉각적인 피드백에 의해 단순하게 코딩하여 SW품질을 높이기 위한 방법론  - 1~3주 iteration  - 4가치(용기, 단순성, 의사소통, 피드백)  - 12개 실천항목  설명: 1154ED0F4A67B8D677D437 | 가장 주목 받음  개발 관점 |
| SCRUM | - 스프린트(30일 단위 iteration)로 분리  팀은 매일 스크럼(15분 정도) 미팅으로 계획수립  - 팀 구성원이 어떻게 활동해야 하는가  에 초점  - 통합 및 인수 테스트가 상세하지 않음  설명: 1354ED0F4A67B8D77878B3 | Iteration계획과 Tracking에 중점 |
| RUP | - 완전한 SW 개발 모델 제시  - Visual 모델링 도구 지원 | RUP에서 Agility 성격 특히 강조 |
| Crystal | - 프로젝트 상황에 따라 알맞은 방법론  을 적용할 수 있도록 다양한 방법론 제시  - Tailoring하는 원칙 제공 | 프로젝트 중요도와 크기에 따른 메소드 선택 방법 제시 |

1. **Agile 방법론과 각종 방법론 비교**
   1. SCRUM과 CBD방법론 비교

|  |  |  |
| --- | --- | --- |
| 항 목 | SCRUM | CBD 방법론 |
| 정의 | * 프로토타입을 반복적, 점진적으로 개발하는 Agile 개발방법론 | * 기 개발된 SW컴포넌트를 조립하여 새로운 시스템을 구축하는 방법론 |
| 특징 | * 반복적, 점진적 프로토타입 개발 | * 컴포넌트 검색 및 조립 |
| 확장  방법 | * Sprint중, sprint backlog iteration 기법 | * 컴포넌트간 호출을 위한 인터페이스 사용 |
| 장점 | * 빠른 개발주기 * 프로젝트 초반 요구사항 명확 | * 재사용성 극대화 * 장기적 관점 개발비 감소 |
| 단점 | * 대형 프로젝트 적용 어려움 * 산출물 중시사업에 부적합 | * 공용 컴포넌트 개발,유통의 한계 * 개발 전문업체 부족   “끝” |

* 1. Agile 개발방법론과 전통적 개발방법론 비교

|  |  |  |
| --- | --- | --- |
| 항 목 | Agile 개발 방법론 | 전통적 개발 방법론 |
| 계획수립의 상세수준 | * 바로 다음 반복주기 (Iteration)에 대해서만 상세한 계획수립 | * 다음에 이어지는 단계에 이르기까지의 상세한 계획수립 |
| 요구사항의  베이스라인 | * 요구사항에 대한 베이스라인 설정을 강조하지 않음 | * 초기 요구사항에 대한 Base line 을 설정 |
| 아키텍처  정의방법 | * 실제 개발된 기능 구현을 통하여 빠른 시간 내에 아키텍처의 실현 가능성을 증명해 보이고자 함 | * 모델과 사양을 보다 상세화하는 과정을 통해 어플리케이션과 데이터 아키텍처를 초기에 정의하고자 함 |
| 테스트  방법 | * 잦은 “개발-테스트” 주기를 통하여 많은 시간과 비용이 들어가기 전에 기능을 검증함 | * 특정 기능이 구현되고 나서야 단위-통합-시스템으로 확장해 나가는 방식을 취함 |
| 표준  프로세스  적용 | * 정의되고 반복적으로 수행 되는 프로세스를 강조하지 않는 대신에 잦은 Inspection을 토대로 프로세스를 개발에 유연하게 적용하는 것 강조 | * 개발에 들어가기 전에 표준화된 * 프로세스를 제정하는 것을 중요하게 여김 |

* 1. Agile 개발방법론과 나선형 모델비교 (기출)

|  |  |  |
| --- | --- | --- |
| 항 목 | Agile 개발 방법론 | 나선형 모델 |
| 유사점 | * 짧은 기간 동안 사용자의 요구사항에 대하여 반복적으로 개발되므로 요구사항의 파악 및 대처가 쉬우며 잠재적 위험부담을 줄일 수 있음. | * 개발된 프로토타이핑을 지속적으로 발전시켜 최종 SW에 이르게 되므로 사용자의 요구 사항 파악이 쉬우며 위험 부담이 감소됨. |
| 차이점 | * 중소형, 아키텍처 설계, 프로토타이핑에 적합 * 절차보다는 사람 개개인의 상호 작용이 더욱 중요 | * 대규모 시스템 및 위험 부담이 큰 시스템 개발에 적합 * 절차가 중요하지만 이러한 절차의 관리가 매우 어려움 |

**cf. Agile방법론 장,단점 (기출)**

|  |  |
| --- | --- |
| 구분 | 내용 |
| 단점 | * 방법론 그 자체로서 적용하기에는 프로세스 정립 부족 * 대형 프로젝트에 적용하기에 적합하지 않음 * 감리에 대한 대응이 어려움 * 관리 방법에 대한 가이드라인 부족 * 해당 프로세스를 적용하기 위해 갖추어야 할 제약조건이   실제 중요하면서도, 어려운 부문 임 |
| 장점 | * 방법론 그 자체로서가 아니라, 일부 기업 또는 사상을   선택하여 쓰기에 매우 좋음   * 중, 소형 프로젝트에 적용하기 적합하며, 대형 프로젝트라   할지라도 특정 테스크에 대해 채택하는 것이 바람직한 영역이 있음   * 아키텍처설계 및 프로토타이핑 수립과 같은 테스크 수행 시 적합 |

**문제) SCRUM**

**답)**

1. **Sprint를 활용한 Agile 방법론, SCRUM의 개요**
   1. **SCRUM의 정의**

- 개발 환경을 최대한으로 이용하고 조직의 고정비를 줄이며 반복적인 프로토타입을 기반으로 시장 수요에 가깝게 동기화 하는 세트

- **비즈니스 요구 사항을 만족시키는 소프트웨어를 개발하는데 초점을 맞추기 위해 복잡함을 제거하는 관리 및 제어 프로세스**

**-** 짧은 주기 내에 실행 가능한 제품(Shippable Product)을 개발하는 기법

**-** Agile 방법론 중의 하나로, Product Backlog를 바탕으로 하여 기술적으로

분할되고 재해석된 스프린트(Sprint)를 스크럼 팀(team)을 통해 구현해 나가는

개발방법론

* 1. **등장배경**
     + SCRUM은 1986년 일본에서 개발 환경을 최대한으로 이용하고 조직의 고정비를 줄이며 반복적인 프로토타입을 기반으로 시장 수요에 가깝게 동기화 하기 위함
     + 비즈니스 요구 사항을 만족시키는 소프트웨어를 개발하는데 초점을 맞추기 위해 복잡함을 제거하는 관리 및 제어 프로세스
  2. **SCRUM의 특징**

|  |  |
| --- | --- |
| 특징 | 설명 |
| **협업 중심** | * SCRUM은 주로 팀 수준의 사안을 다룸. * 효율적으로 팀원들이 협업할 수 있는 환경 제공 * 팀원들의 협업을 통해 고품질의 제품을 생산 |
| **사회공학 기법** | - 프로젝트 이해관계자들의 적극적인 협력과 참여를 촉진하여, 관련자의 성취감의 충족을 목적으로 함 |
| **Sprint 수행** | 통상 약 4~6주(30일) 기간의 Time box를 가지며, 이를 반복 수행함 |
| **Daily Meeting** | 8~10명 정도의 SCRUM팀은 매일 15분 정도의 회의를 통해 진척관리와 이슈 중심의 계획 리뷰 실시 |
| **독립적 방법론** | 특정 언어나 방법론에 의존하지 않으며, 개발 언어와 상관없이 적용 가능한 폭넓은 응용 범위의 개발 기법 |
| ※ 스크럼의 성공적인 수행을 위해서는 원활한 의사 소통이 필수이며,  이를 위해 구분 없는 열린 공간을 유지해야 함 | |
| ※ 스크럼은 구체적인 실천방법이기보다는 팀 성과에 대한 가시성을 기반으로 “관찰과 적응” 메커니즘 제공 | |

* 1. **SCRUM이 추구하는 가치**

|  |  |  |
| --- | --- | --- |
| **가치** | **내용** |  |
| **확약** | 약속한 것을 확실히 실현하는 것 |
| **전념** | 확약한 것의 실현에 전념하는 것 |
| **정직** | 비록 자신에게 불리한 일이라도 숨기지 않는 것 |
| **존중** | 자신과 다른 사람에게 경의를 표하는 것 |
| **용기** | 위의 사항을 언제든지 지킬 수 있는 용기 |

1. **SCRUM 개념도 및 구성요소**
   1. **SCRUM 개념도**

****

* Product Backlog로부터 Sprint 내 수행할 Sprint Backlog를 추출하고,

Daily SCRUM을 활용하여 이를 Sprint 기간 내에 구현

* 구현해야 될 기능을 Product Backlog로 만들고, 스프린트 내에 수행할

Sprint Backlog는 Time Boxing을 강조하는 Sprint 기간 내에 Shippable

Product 개발 완료를 목표로 함

* 1. **SCRUM의 구성요소**

|  |  |
| --- | --- |
| **구성요소** | **설명** |
| **Product Backlog** | * 시스템에서 해결해야 하거나, 시스템에 포함되어야 할 기능, 특성과 기술에 대한 모든 기술 나열 * 요구되는 제품의 요구사항의 우선순위 나열 * 프로젝트가 진행되면서 진화되고 변경 |
| **Sprint Backlog** | * 해당 Sprint 기간에 수행되어야 하는 Task 목록으로 Sprint 기간 동안 개발 가능한 기능의 목록을 Product Backlog에서 선택 |
| **Sprint** | * 통상 4~6주(30일) 정도의 Timebox 성격을 가진 잘 정의된 반복 개발 주기 * 각 Sprint 단계 종료 시 새로운 기능이 추가되어 실행 가능 제품이 인도되어야 함 |
| **Daily Scrum** | * 매일 약 15분 정도의 짧은 회의 * SCRUM Master는 진척 사항 검토, 정상적 종료를 방해하는 위험 및 작업 계획을 확인 |

1. **Sprint를 기반으로 반복 수행되는 SCRUM 프로세스**
   1. **SCRUM 프로세스**

****

* 구현해야 할 기능을 Product Backlog로 만들고, Sprint 내에 수행할 Sprint Backlog는 Time Boxing을 강조하는 Sprint 기간 내에 Shippable Product 개발 완료를 목표로 함
  1. **SCRUM 프로세스 설명**

|  |  |  |
| --- | --- | --- |
| **프로세스** | **상세 설명** | **특징** |
| **Prepare Product Backlog** | * 실제로 구현되어야 하는 기능 목록을 나열 * SRS(요구사항정의서)나 TRS(기술요구 사항정의서)로부터 목록 도출 | 추정리소스, 우선순위 등 |
| **Release Planning** | * 해야 할 작업(일)에 대한 계획 수립 * Milestone을 정하여 Release시가 마다 작동 가능한 Product Release | 위험의 조기 발견 |
| **Sprint Planning** | Release Planning 이후 각 Release를 달성하기 위한 Sprint 계획 수립 | 20%버퍼 법칙 적용 |
| **Sprint Tracking** | 일일 미팅(Daily Scrum)을 수행하면서 계획에 따른 프로젝트 수행 | Daily Scrum  Burn down chart |
| **Ending Sprint** | 정해진 기간 동안 Sprint가 종료되면, 정리 실시 | 계획된 일정, 예산 소진 시 까지 |
| **Review Sprint** | * Sprint가 종료된 후에는 Sprint에서 구현된 산출물을 Review하는 단계 * 구현된 코드, 산출물 | Test, Code Review |
| **Update Product Backlog** | * Review과정에서 나온 추가 요건이나 변경사항을 반영하여 Product Backlog update * 우선순위 재조정, 요구사항 구체화 | 우선순위 재조정 및 구체화 |
| **Retrospective** | Scrum팀에 운영중인 방법론 자체에 대한 Review, 문제 개선 | SWOT분석 |

* 1. **SCRUM 프로세스 적용을 위한 구성원의 역할**

|  |  |
| --- | --- |
| **구성원** | **내용** |
| **Product Owner** | * 고객, 관리자, 팀원들과의 협의를 통하여 목표 설정 * 요구사항 정의, Product Backlog 업데이트 * Product Backlog 내 Item의 우선순위 정의 * 요구사항이 적절하게 구현되었는지 검토 및 확인 |
| **SCRUM Master** | 프로젝트 관리자로서 팀원을 코칭하고 프로젝트의 문제 상황을 해결 |
| **Team** | Product Backlog에 따라 Sprint 기간 내 구현 |

1. **SCRUM과 XP와의 비교**

|  |  |  |
| --- | --- | --- |
| **구분** | **SCRUM** | **XP** |
| **형태** | * 개발 프로세스를 위한 프레임워크 * 관리 및 조직적 실천법에 집중 * 특정한 엔지니어링 방법을 포함하지 않음 | * 엔지니어링 방법에 초점을 맞춤 * 프로그램 실천법에 집중 * 개발 프로세스의 핵심 프레임워크는 포함하지 않음 |
| **개발주기** | 4~6주 | 1~2주 |
| **요구사항  변경** | * Sprint 내에서 요구사항 변경을 수용하지 않음 * 변화를 최대한 빨리 발견하고 처리하는 관점에서 접근 | * 리팩토링을 통하여 요구사항 변경을 수용 * 요구사항 변경은 당연히 발생하는 것으로 인정 |
| **개발 우선순위 결정 주체** | Product Owner가 아닌 Team이 개발 우선순위 결정 | Customer가 개발 우선순위를 결정 |
| **공통점** | Agile 방법론으로서 짧은 개발 주기로 반복 개발 | |

1. **SCRUM 적용 현황 및 적용 시 고려사항**
   1. **SCRUM 적용 현황**
      * 금융, 보험 등 보수적인 산업에 있는 기업과 Google, Yahoo, MS 등도 SCRUM을 선택
      * Agile 방법론을 실행하는 팀 중 약 50%가 Scrum을 사용하고, 20%가 Scrum을 XP 구성 요소와 함께 사용하고, 12%가 XP만을 사용
   2. **SCRUM 프로젝트 적용 시 고려사항**
      * 전체 프로젝트에 대한 큰 Mile Stone은 기존 개발방법론을 사용하고, SCRUM 방법론은 각 단계에 대한 Task를 위해서 사용
      * 어떤 전문화된 시스템이나 도구가 아니라, 팀이 최대한의 효율을 낼 수 있는 프로세스가 중요 “끝”

[참조]

1. **Product Backlog와 Sprint Backlog의 비교**

|  |  |  |
| --- | --- | --- |
| **구분** | **Product Backlog** | **Sprint Backlog** |
| **상세 수준** | 조금 상세함 | 매우 상세함 |
| **예측 단위** | 스토리 점수 | 시간 |
| **문서 소유권** | Product Owner | Team |
| **수정 주기** | 매주 | 매일 |
| **기간** | 프로젝트 기간 내 | Sprint 기간 내 |

1. **SCRUM과 XP의 용어 비교**

|  |  |
| --- | --- |
| **SCRUM** | **XP** |
| Product Owner | Customer |
| SCRUM Master | XP Coach |
| Team | Team |
| Sprint | Iteration |

※ XP 방법론의 특징

|  |  |
| --- | --- |
| 핵심 가치 | 주요 내용 |
| 용기 | 고객의 요구사상 변화에 능동적 대처 |
| 의사소통 | 실제 개발자들 사이의 의사소통을 통한 개발 사이클 채택 |
| 피드백 | 빠른 피드백이 기본 원칙, 해결할 수 있는 일 먼저 처리 |
| 단순성 | 부가적 기능, 사용되지 않는 구조와 알고리즘 배제 |

**문제) XP (eXtreme Programming).**

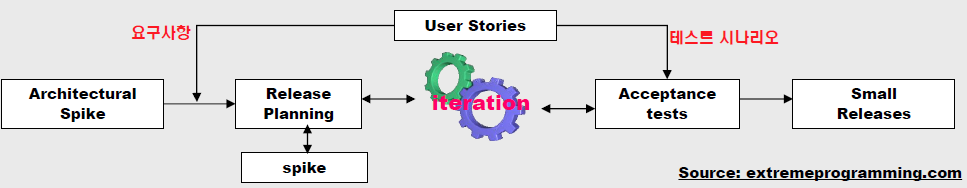
**답)**

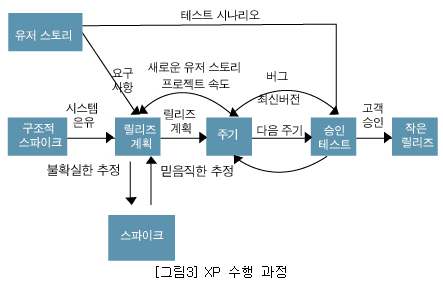
1. 유연하고 신속한 개발 패러다임, XP 의 개요
   1. XP (eXtreme Programming) 방법론의 정의

* 의사 소통을 개선하며, 즉각적인 피드백에 의해 단순하게 코딩하여 SW 품질을 높이기 위한 개발 방법론
* 개발자, 관리자, 고객이 조화를 극대화하여 개발 생산성을 높이고자 하는 접근법
* 주기의 반복을 통해 요구 변화에 신속히 대응하여 위험을 줄이고 고객 관점의 고품질 S/W를 빠르게 전달하는 Agile 방법론의 대표적 기법
* 라이프 사이클 후반부라도 요구사항 변경에 적극적이고, 긍정적인 대처를 권고하는 역발상의SW 개발 방법
  1. XP의 출현배경
* RUP의 산출물 부담과 신속한 개발의 어려움 - Light-weight
* Time to Market 실현과 Products의 적시 배포
* 프로세스 중심의 전통적 방법론으로는 변화의 유연한 수용에 한계
  1. XP 의 특징
     1. 의사소통 개선, 즉각적인 피드백, 단순한 코딩을 통해 SW 품질을   
        높이기 위한 방법론
     2. 개발자, 관리자, 고객간 조화로 개발 생산성을 높이고자 하는 접근법
     3. 고객 요구사항 변경에 적극적, 긍정적인 대처

II. XP 개발 절차 및 다른 방법론과의 비교

가. XP 개발 개념도





* 고객 요구사항을 바탕으로 우선순위별 일정 수립
* 새로운 사용자스토리 발생시 ‘릴리즈 계획’단계로 전환 반복
* 일정한 단위 프로그래밍의 소규모 릴리즈 추구

나. XP 개발 절차 용어

|  |  |
| --- | --- |
| 용어 | 설명 |
| 사용자 스토리 | - UML의 유스케이스와 같이 역할(요구사항 수집, 의사소통 도구)  - 릴리즈 계획을 작성하기 위한 단위  - 기능단위로 필요한 내용을 간단하게 기재, 필요 시 간단한 테스트 사항도 표기 가능  - 시스템을 사용하는 고객이 필요한 것이 무엇인지를 기술 → 인수 테스트 시 사용 |
| 스파이크 | - 어려운 요구사항 혹은 잠재적인 솔루션들을 고려하기 위해 작성하는 간단한 프로그램  - 사용자 스토리의 신뢰성을 증대시키거나 기술적인 문제의 위험을 줄이고자 하는데 목적 |
| 릴리즈 계획 | - 전체 프로젝트에 대한 배포 계획을 생성 |
| 반복 | - 하나의 반복을 1에서 3주 정도로 나누고 반복들을 프로젝트 전반에 균일하게 유지  - XP의 반복은 프로세스의 평가와 계획을 단순하고 신뢰성 있게 만드는 핵심 항목 → 반복 계획 미팅  - 사용자 요구사항 변경, 기술적인 문제 등 상황에 따라 릴리즈및 반복 계획 수정 가능 |
| 승인 테스트 | - 릴리즈 전의 인수 테스트. 고객이 수행 |
| 소규모 릴리즈 | - 작은 배포는 XP 주기의 마지막 단계  - 소규모로 빈번하게 배포하면 고객에게 여러 가지 이득을 조기 제공  - 프로그램은 빠른 피드백을 제공 받음 |

**Ⅲ. XP의 4가지 핵심가치와 12가지 실천 사항**

가. XP의 5가지 핵심가치(Core Value) **용단커피존**

|  |  |
| --- | --- |
| 핵심가치 | 설명 |
| 단순성 | 부가적 기능, 사용되지 않는 구조와 알고리즘 배제 |
| 의사소통 | 개발자, 관리자, 고객 간의 원활한 의사소통 |
| 피드백 | 지속적인 테스트와 통합, 반복적 결함 수정, 빠른 피드백 |
| 용기 | 고객의 요구사항 변화에 능동적인 대처 |
| 존경 | 팀원간의 상호 존경 |

1. 12가지 실천 사항(Practice)

|  |  |
| --- | --- |
| 핵심가치 | 설명 |
| Planning game  (Planning process) | - 사용자 스토리를 이용해서 다음 릴리즈의 범위를 빠르게 결정, 비즈니스 우선순위와 기술적 평가가 결합 |
| Small/short releases | - 필요한 기능들만 갖춘 간단한 시스템을 빠르게 릴리즈, 아주 짧은(2주) 사이클로 자주 새로운 버전 배포 |
| System Metaphor | - 최종적으로 개발 되어야 할 시스템의 구조를 기술  일반적으로 SW 개발 시 작성되는 System Architecture에 해당.  - 그림으로 도식화 하지 않고 문장으로 기술.  - 전체 개발 프로세스에 대한 전체 그림.  - 공통의 name system(개발 및 커뮤니케이션 과정에서 공통된 개념을 공유 가능하게 함) |
| Simple design | - 현재 필요한 것에 집중하여 설계 함  - 현재의 요구사항을 만족시키도록 가능한 한 단순하게 설계함. |
| TDD(Test Driven Development), Testing | - 단위 TEST를 코드 작성 이전에 작성하고 단위 TEST가 모두 통과한 경우 완성되는 개념.  - 개발자가 먼저 단위테스트, 고객은 기능 테스트를 작성하여 요구사항이 모두 반영되었는지를 확인 |
| **Refactoring** | - 프로그램 기능은 변경 없이 중복제거, 커뮤니케이션 향상, 단순화, 유연성 추가 등을 위해 시스템 재구성 |
| **Pair Programming** | - 하나의 컴퓨터에 두 사람이 함께 프로그램(Driver/Partner) |
| Collective ownership | - 공동 소유, 공동 책임  - 시스템에 있는 코드는 누구든지 언제라도 수정 가능함 |
| CI(Continuous Integration) | - 상시 빌드가 가능하여 배포가 가능한 구조 |
| 40-hour week | - 일주일에 40 시간 이상, 2주를 연속 잔업 금지 |
| On-site customer | - 개발자들의 질문에 즉각 대답해 줄 수 DT는 고객을 프로제트에 **풀 타임으로 상주시킴** |
| Coding standards | - 팀원들 간 커뮤니케이션 향상을 위해 표준화된 코드 준수 |

**IV. XP와 다른 방법론과의 비교**

1. SCRUM과 XP와의 비교

|  |  |  |
| --- | --- | --- |
| 구분 | SCRUM | XP |
| 형태 | 개발 프로세스를 위한 프레임워크  관리 및 조직적 실천법에 집중  특정한 엔지니어링 방법을 포함하지 않음 | 엔지니어링 방법에 초점을 맞춤  프로그램 실천법에 집중  개발 프로세스의 핵심프레임워크는 포함하지 않음 |
| 개발주기 | 4~6주 | 1~2주 |
| 요구사항 변경 | Sprint 내에서 요구사항 변경을 수용하지 않음  변화를 최대한 빨리 발견하고 처리하는 관점에서 접근 | 리팩토링을 통하여 요구사항 변경을 수용  요구사항 변경은 당연히 발생하는 것으로 인정 |
| 개발우선순위결정주체 | Product Owner가 아닌 Team이개발 우선순위 결정 | Customer가 개발 우선순위를 결정 |
| 공통점 | Agile 방법론으로서 짧은 개발주기로 반복개발 | |

1. XP와 구조적 방법론, 객체지향방법론의 비교

|  |  |  |  |
| --- | --- | --- | --- |
| 구분 | 구조적 방법론 | 객체지향 방법론 | XP |
| 중심내용 | 시스템의 작업중심 | 객체와 역할 발견 | 반복과 검증 |
| 중요관점 | 데이터흐름 | 데이터와 관련연산 | 나선형 모델 적용 |
| 분석/설계방법 | DFD, 구조차트 | ERD, UML | 객체지향방법론 |

**IV. XP 방법론의 기대효과 및 고려사항**

* 1. XP의 기대효과

|  |  |
| --- | --- |
| 구분 | 내용 |
| 계획측면 | 팀이 개발한 SW의 가치를 최대화, Embedded 소프트웨어  가능한 적게 투자하고, 빠르게, 가장 가치 있는 기능부터 구현 |
| 설계측면 | 의사소통을 통하여 설계, 중복된 코드가 없을 것 🡪 가장 단순하게 개발  가능한 클래스와 method 들은 적게 만들 것 🡪 변경에 유연 |
| 개발측면 | 지속적인 통합, 수정, 테스트, 배포  복잡한 코드 제거, Pair 프로그램으로 완벽한 코딩 |
| 테스트측면 | 각 테스트는 다른 부분과 관련이 없어야 함 🡪 테스트 자동화  테스트 프로그램 만드는 주체는 프로그래머 |

* 1. XP의 도입 시 고려사항

|  |  |  |
| --- | --- | --- |
| 구분 | 고려사항 | 대응방안 |
| 재무관점 | 요구사항이 빈번히 변경되어 프로젝트의 전체 규모와 예산 수립이 어려움 | 릴리즈 단위가 작고 소규모 프로젝트에 적용 |
| 고객관점 | 개발자와 사용자간의 공감대 형성이 부족할 경우 개발 생산성을 저하시킬 수 있음 | 개발 후반부라도 요구사항을 적극반영하고 구동하는 S/W를 고객에게 자주 전달하여 고객의 요구사항 신속 적용 |
| 내부관점 | 대규모 프로젝트 수행이 어려움 | 소규모나 타임 박스화된 서브 프로젝트나 반복주기(Iteration)에 선택적 적용이 적합 |
| 학습 및 성장 | S/W 품질수준이 낮고, 산출물의 범위 변경이 불가능한 경우 적용이 어려움 | 사용자나 개발자가 정확한 요구사항 도출이 힘들 때 적용, 잠재적 가치생성 |
| 개발자에게 Refactoring, Design Pattern 등 기술 요소에 대한 높은 숙련도 요구 | 팀 구성원들의 XP에 대한 충분한 이해와 훈련필요 |

**문제) RAD**

**답)**

**I. RAD (Rapid Application Development) 기법 모델**

가. RAD기법 모델의 정의

-2~3개월의 짧은 개발 주기 동안 소프트웨어를 개발하기 위한 순차적인 프로세스 모델로서 빠른 개발을 위해 Visual Tool, Code Generation Tool을 사용

나. RAD기법 모델의 특징

- 도구의 활용 (CASE도구, RDB, 재사용 Library 등)

- Prototyping 사용 및 사용자가 요구사항 정의, 분석, 설계과정에 적극 참여

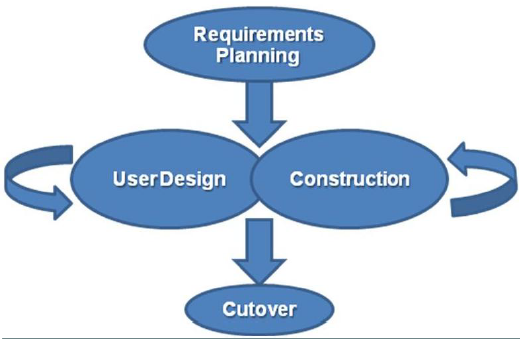
- 소요기간: 60일 ~ 90일 정도로 짧은 기간으로 기술적 위험이 적고 빠른 개발이 요구될 때 적합

**II. RAD기법 모델의 단계**

|  |  |
| --- | --- |
| 단계 | 설명 |
| JRP (Joint Requirement Planning) | - 분석단계로서, 공동 요구사항 정의  - 사용자와 함께 Biz모델 작성/검토 반복을 통한 분석  - 데이터 모델링, 프로세스 모델링(1~2주)를 짧게 가져감 |
| JAD (Joining Application Development) | - 설계 단계, 참여 공동 설계  - 개략적 모델링 후, 사용자와 함께 Prototype 개발/수정/보완 반복을 시스템 설계 (3~5주) |
| Construction phase | Focuses on program and application development task similar to the SDLC |
| Cutover (구축/운영) | 운영에 필요한 지침서를 작성하고 현업부서로 이전 |

**III. RAD개발 모델의 개념도 및 RAD 대표 개발 방법론**

1. RAD 개발 모델의 개념도



1. RAD 모델 기반의 Agile 대표 개발 방법론

|  |  |  |
| --- | --- | --- |
| 종류 | 특징 | 비고 |
| XP | 테스팅 중심, 4가지 가치와 12개 실천항목을 가지고 1~3주 반복 | 가장 주목 받고 있음  개발 관점 |
| SCRUM | 프로젝트를 스프린트로 분리, 팀은 매일 스크럼 미팅으로 계획 수립 | Iteration 계획과 Tracking에 중점 |

**VI. 전통적 생명주기와의 비교**

|  |  |  |
| --- | --- | --- |
| 구분 | RAD기법 모델 | 전통적 생명주기 |
| 목표 | 핵심 요구사항 만족, 시간 단축 | 고품질 구현 |
| 개발인원 | 소규모, 사용자+개발자 | 대규모 |
| 분석/설계 | 개략적 분석/설계 | 완벽한 분석/설계 |
| 기법 | JRP, JAD, Time-boxing | 데이터 모델링, 프로세스 모델링 |
| 특징 | 사용자 지속적 참여  Tools 사용, 적정 규모 | 순차적 접근  하향식 접근 |

**V. RAD기법 모델의 장단점**

|  |  |
| --- | --- |
| 장점 | 단점 |
| 요구사항의 완전한 이해와 프로젝트 범위의 명확한 설정 시 신속한 개발 및 완전한 기능 구현 가능 | 책임감 있는 구성원이 없을 경우 실패  적절한 모듈화 가능성 전제  기술적 위험이 높을 경우 부적합 |

“끝”

|  |  |  |
| --- | --- | --- |
| **98-관리-1** | | **소프트웨어 개발 방법론 Agile Methodology의 정의, 특성, 장단점을 설명하시오** |
| **출제도메인** | 소프트웨어 공학 | |
| **주요 키워드** | 고객만족, 짧은 반복주기, 암묵적 지식, ROI, 품질향상 등 | |
| **난이도** | ★ ★ ☆ ☆ ☆ (별5개 기준) | |
| **참고문헌** | Agile 적용상의 문제점 및 개선방향(Agile Society) | |
| **문제소견** | 애자일의 장단점 뿐만 아니라, 애자일의 적용시 단점에 대한 기술사적인 입장을 제시해야 함 | |
| **기출 풀이 담당 기술사** | 한창열 기술사(제96회 정보관리기술사 / hiyori@gmail.com) | |

**1.SW 개발의 Time-To-Market을 실현하기 위한 Agile Methodology의 개요**

가. Agile Methodology의 정의

- 절차보다는 사람이 중심이 되어 변화에 유연하고 신속하게 적응하면서 효율적으로 시스템을 개발할 수 있는 방법론

나. Agile Methodology의 특성

|  |  |
| --- | --- |
| 특성 | 내용 |
| 고객만족 | - 개발후반부라도 요구사항의 변화를 적극반영  - 구동하는 SW를 고객에게 자주 전달하여 고객의 요구사항 신속 적용 |
| 짧은 반복주기 | - 요구사항의 변경에 대한 프로세스의 적응성 극대화 |
| 암묵적 지식에 의존 | - 문서화보다는 실제 구동하는 SW개발에 중점  - 개발자 개개인의 잠재력 최대 발휘 |
| PM의 역할변화 | - 프로젝트 관리자에서 촉진자로 변경  - 프로젝트 계획수립 및 통제의 책임이 팀원에게 이양 |
| 개발자 동기부여 | - 개발자에게 적합한 개발환경을 구성  - 개발자가 책임을 완수할 것으로 신뢰 |
| 의사소통 | - 구성원간의 의사소통 활성화  - 개발팀원들간의 대화, 협업 강조 |

**2. Agile Methodology의 장단점**

가. Agile Methodology의 장점

|  |  |
| --- | --- |
| 장점 | 설명 |
| ROI 증대 | - 고객들에게 가치 있는 기능들을 빠르고 안정적으로 전달 |
| Delivery Time 감소 | - 요구사항의 변화를 유연하게 수용함으로써 Time to Market 실현 |
| 창의성 향상 | - 팀 자율성 강화를 통한 업무만족도와 창의성 향상 |
| 생산성 향상 | - 불필요한 산출물 제거 및 팀 협업 강화 |
| 제품 품질 향상 | - 고객의 주기적 피드백 및 빈번한 테스트 |

나. Agile Methodology의 단점

|  |  |
| --- | --- |
| 단점 | 내용 |
| 체계적인 문서화 및 지침 부족 | - Agile 프로세스 적용을 지원하는 문서화나 구체적인 지침 부족  - 전체 제품에 대한 통합과 테스팅에 대한 가이드가 부족 |
| 요구사항 변경에 의한 오버헤드 | - 비기능적인 요구사항에 대한 고려가 부족  - 요구사항의 잦은 변경에 따른 테스트 수행노력 증가 |
| 사업 관리 부분 미흡 | - 측정지표가 개발자 위주로 상위관리자의 요구를 충족하기 충족하기 어려움  - 프로젝트 리스크 관리 (Risk management)에 대한 고려 부족 |
| 감리 대응 문제 | - 기존 방법론에 비해 적은 문서량, 감리 기준에 부합되지 않는 산출물로 이슈 발생 |

**3. Agile Methodology의 보완을 통한 성공적 개발 수행 전략**



- Agile Methology의 성공을 위하여 기술적 관리적으로 체계적인 접근 전략이 필요

- 프로젝트 상황에 맞게 방법론 테일러링(Tailoring)도 검토 필요

“끝”

|  |  |
| --- | --- |
| **93-관리-1** | **SCRUM의 특성과 프로세스를 설명하시오** |

|  |  |
| --- | --- |
| **출제도메인** | SW공학 |
| **주요 키워드** | Product backlog, sprint backlog, sprint, daily meeting |
| **난이도** | ★★☆☆☆ (별5개 기준) |
| **참고자료** | - 모의고사 출제 자료  - SRCUM 교재 |
| **문제 소견** | Agile방법론 중 국내 실정에 맞는 실용적인 기법으로 대/중/소 규모에 적절히 기존의 방법론과 결합하여 사용함으로써 성공사례가 늘어나고 있는 개발 방법론 |
| **기출풀이**  **담당 기술사** | 설영주PE(92회 정보관리기술사, mobilejjang@paran.com) |

**1. Sprint를 통한 Shippable Product를 생산하는 SCRUM의 특성**

**가) SCRUM의 주요 특징**

- 일본 히토츠바시 대학의 노나카 이쿠지로와 타케우지 히로시고 ( 1986)에 의해 시작된 짧은 주기 내에 실행 가능한 제품 ( shippable product)을 개발하는 개발 기법

|  |  |
| --- | --- |
| **특징** | **내용** |
| **협업중심** | -스크럼은 주로 팀 수준의 시안을 다루며, 팀원들의 협업을 통해 고품질의 제품을 생산할 수 있게 해 줌 |
| **사회공학 기법** | -프로젝트 이해관계자들이 적극적인 협력과 참여를 촉진 |
| **Sprint수행** | -통상 4~6 ( 30일 ) 정도의 기간의 time box를 가지며 이를 반복 수행 함 |
| **Daily Meeting** | -8-10명 정도의 SCRUM팀은 매일 15분 정도의 일일 미팅을 통해 진척관리과 이슈중심의 계획 리뷰 실시 |
| **독립적 방법론** | -특정언어나 방법론에 의존적이지 않으며, 개발언어와도 상관없이 적용가능한 폭넓은 응용범위의 개발 기법 |

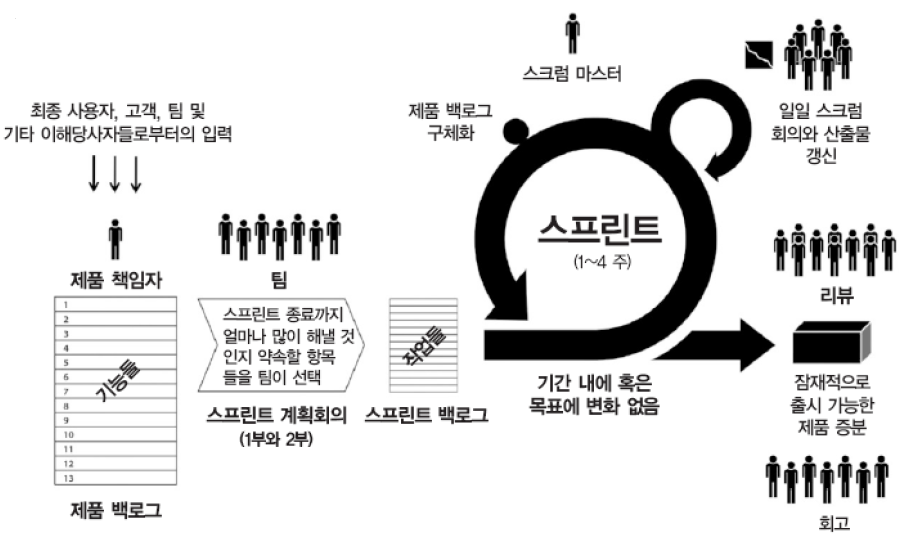
- 스크럼의 성공적인 수행을 위해서는 원활한 의사 소통이 필수이며 이를 위해, 구분 없는 열린 공간를 유지하여야 함.

**나) SCRUM이 추구하는 가치**

|  |  |  |
| --- | --- | --- |
| **가치** | **내용** |  |
| **확약** | 약속한 것은 확실히 실현되는 것 |
| **전념** | 확약한 것의 실현에 전념하는 것 |
| **정직** | 비록 자신에게 있어서 불리한 일에서도 숨기지 않는 것 |
| **존중** | 자신과 다른 사람에게 경의를 표하는 것 |
| **용기** | 위의 사항을 언제든지 지킬 수 있는 요기 |

**2. Sprint를 기반으로 반복 수행하는 SCRUM 프로세스**

**가) SCRUM 프로세스**

****

- 구현해야 될 기능을 Product Backlog로 만들고, 스프린트내에 수행할 Sprint Backlog는 Time Boxing을 강조하는 Sprint기간내에 Shippable Product 개발 완료를 목표로 함

**나) SCRUM Process 설명**

|  |  |  |
| --- | --- | --- |
| **프로세스** | **상세 설명** | **특징** |
| **Prepare Product Backlog** | - 실제로 구현되어야 하는 기능 목록을 나열  - 목록 도출은 SRS(요구사항정의서)나 TRS(기술요구사항정의서) 로부터 함 | 추정리소스, 우선순위 등 |
| **Release Planning** | - 해야 할 작업(일)에 대한 계획 수립  - Milestone을 정하여 Release시가 마다 작동 가능한 Product Release | 위험의 조기 발견 |
| **Sprint Planning** | - Release Planning 이후 각 Release를 달성하기 위한 Sprint 계획 수립 | 20%버퍼 법칙 적용 |
| **Sprint Tracking** | - 일일 미팅(Daily Scrum)을 수행하면서 계획에 따른 프로젝트 수행 | Daily Scrum  Burn down chart |
| **Ending Sprint** | - 정해진 기간 동안 Sprint가 종료되면, 정리 실시 | 계획된 일정, 예산 소진시 까지 |
| **Review Sprint** | - Sprint가 종료된 후에는 Sprint에서 구현된 산출물을 Review하는 단계  - 구현된 코드, 산출물 | Test, Code Review |
| **Update Product Backlog** | - Review과정에서 나온 추가 요건이나 변경사항을 반영하여 Product Backlog update  - 우선순위 재조정, 요구사항 구체화 | 우선순위 재조정 및 구체화 |
| **Retrospective** | - Scrum팀에 운영중인 방법론 자체에 대한 Review, 문제 개선 | SWOT분석 |

- 스크럼은 구체적인 실천방법이기보다는 팀 성과에 대한 가시성을 기반으로 ‘관찰과 적응’ 메커니즘 제공

**3. SCRUM 프로세스 적용을 위한 역할 정의 및 유의사항**

**가) SCRUM 프로세스 적용 위한 구성원의 역할**

|  |  |
| --- | --- |
| **구성원** | **역할 설명** |
| **Product Owner** | - 고객, 관리자,팀원들과 협의를 통해 목표 설정  - 요구사항 정의, Product Backlog 업데이트 수행  - Product Backlog내의 Item 우선순위 정의  - 요구사항이 적절하게 구현되었는지 검토하고 확인하는 역할 수행 |
| **Team** | - Product Backlog에 따라 Sprint에서 구현하는 역할 |
| **Scrum Master** | - 프로젝트 관리자로서 팀원을 코칭하고 프로젝트의 문제 상황을 해결하는 역할 |

**나) SCRUM 프로세스 적용시 유의사항**

- 전체 프로젝트에 대한 큰 Mile stone은 기존 방법론을 적용하고, 그 아래에서 Agile 방법론을 팀의 Task 관리 기법으로 사용

- 어떤 전문화된 시스템이나 도구가 아니라, 팀이 최대한의 효율을 낼 수 있는 프로세스가 중요

“끝”

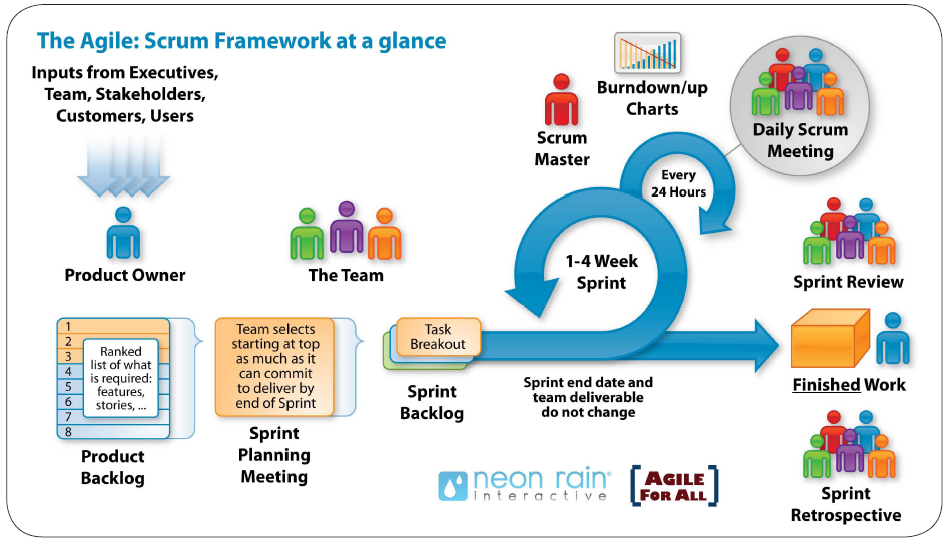
|  |  |
| --- | --- |
| **2013년 5월**  **관리 1교시** | **스크럼(SCRUM) 프로세스에 대해서 설명하시오.** |
| **출제도메인** | 소프트웨어공학 |
| **주요 키워드** | - 스프린트 Sprint, 일일 스크럼, 스프린트 계획, 스프린트 리뷰, 제품 백로그, 스프린트 백로그, 소멸 차트 |
| **출제배경** | - 애자일 방법론에 대한 관심 증가에 따른 출제 대비 |
| **난이도** | ★ ★ ★ ☆ ☆ (별5개 기준) |
| **참고문헌** | 애자일 초심자를 위한 애자일, 위키피디아 |
| **출제기술사** | 한영우기술사(제90회 정보관리기술사,제98회 컴퓨터시스템응용기술사 / hsyrinks@gmail.com) |

**1. 스크럼(SCRUM) 프로세스의 개념**

가. 스크럼의 정의

- 매일 정해진 시간에 정해진 장소에서 짧은 시간의 개발을 하는 팀을 위한, 프로젝트 관리 중심의 Agile 방법론

나. 스크럼 프로세스의 개념도



- 스크럼의 프로세스를 구성하는 것은 스프린트, 3가지 유형의 미팅과 산출물임

**2. 스크럼 프로세스 구성요소 및 주요절차**

가. 스크럼 프로세스의 구성요소

|  |  |  |
| --- | --- | --- |
| 구분 | 구성요소 | 주요 내용 |
| 기간 | 스프린트 Sprint | - 30일의 반복적인 개발 주기 |
| 미팅 | 일일  스크럼 | - Daily Scrum Meeting  - 날마다 진행되는 진척 상황 미팅 |
| 스프린트 계획 | - Sprint Planning Meeting  - 스프린트 목표와 스프린트 백로그를 계획하는 회의 |
| 스프린트 리뷰 | - 스프린트 목표를 달성했는지 작업 진행과 결과물을 확인하는 회의 |
| 산출물 | 제품 백로그 | - Product Backlog  - 개발할 제품에 대한 요구 사항 목록 |
| 스프린트백로그 | - Sprint Backlog  - 각각의 스프린트 목표에 도달하기 위해 필요한 작업 목록 |
| 소멸 차트 | - Burndown chart  개발을 완료하기까지 남은 작업량을 보여주는 그래프, 각 이터레이션 별로 남아있는 작업 량을 스토리 포인트라는 것으로 나타낸 것 |

나. 스크럼 프로세스 주요 절차

|  |  |  |
| --- | --- | --- |
| 구분 | 단계 | 주요 내용 |
| 프로젝트  준비 | 제품 백로그 | - 아이템의 목적 정의 및 우선순위 정리 (내림차순 정렬)  - 추정치는 팀에 의해 산정 기록하고 스프린트에 적당한 아이템 배분  - 상위 아이템들의 추정치 정리 |
| 스프린트  수행 | 스프린트 계획 회의 | - 제품 책임자가 참여하고, 최신 제품 백로그 제공  - 팀 전체가 참여하여 회의 후 스프린트 계획을 정리  - 팀 전체가 동의할 계획인지 확인하고 제품 책임자가 계획의 우선순위에  타당성 확인 |
| 일일 스크럼 | - 팀 전체가 참여하고 문제점과 장애요인들을 찾아냄 |
| 스프린트 | - 이터레이션 길이를 4주 이하로 고정하고, 항상 정해진 기간에 완료  - 팀이 외부로부터 방해 받거나 통제되지 않게 함 |
| 스프린트 종료 데모 | - 매 스프린트 종료 후 데모를 수행하여 동작되는 테스트된 S/W 보여줌  - 이해당사자와 제품 책임자에게 피드백을 받음 |
| 프로젝트  종료 | 스프린트 완료 회고 | - 팀 전원과 제품 책임자가 참석  - 구체적인 개선안들을 도출 및 개선안들의 실제 적용 |

**3. 스크럼이 추구하는 가치**

|  |  |
| --- | --- |
| 요소 | 주요 내용 |
| 확약 | 약속한 것을 확실히 실현하는 것 |
| 전념 | 확약한 것에 대한 실현에 전념하는 것 |
| 숨기지 않음 | 비록 자신에게 있어서 불리한 일에서도 숨기지 않는 것 |
| 존중 | 자신과 다른 사람에게 경의를 표하는 것 |
| 용기 | 위의 사항을 언제든지 지킬 수 있는 용기 |

- 스크럼의 가치와 개발 프로세스가, 지식창조기업에서 이야기 하는 지식 창조 프로세스와 그것을 촉진하는 요소들을 충족시켜 프로젝트를 성공적으로 이끌게 됨.

“끝”

|  |  |
| --- | --- |
| **101회**  **관리 3교시** | **모바일 앱 개발의 특성과 이슈에 대하여 설명하고, 애자일(Agile)을 활용하여 모바일 개발환경에 적합한 개발방법을 제시하시오** |
| **출제도메인** | 소프트웨어공학 |
| **주요 키워드** | - 동기부여, 신속, 정확, 고객만족, Time-to-Market, 점진적, 반복적 |
| **출제배경** | - 모바일 App 기반의 서비스가 지속적으로 늘어남과 동시에 최적화된 개발 방법론의 필요성이 대두되고 있는 상황임 |
| **난이도** | ★ ★ ★ ☆ ☆ (별5개 기준) |
| **참고문헌** | 애자일 초심자를 위한 애자일, 위키피디아 |
| 모범목차 | 1. 모바일 앱(App) 개발의 특성과 이슈  가. 모바일 앱(app) 개발의 특성  나. 모바일 앱(app) 개발의 이슈  2. 모바일 앱개발 환경과 애자일 방법론의 적합성 Tailoring  가. 모바일 앱개발 환경과 애자일 특징  나. 점진적, 반복적 개발 구체화 방안  3. 적시성 위한 애자일(Agile) 기법 활용한 모바일 개발 방법론  가. 애자일 장점을 접목한 모바일 앱 개발 프로세스  나. 모바일 개발환경에서의 애자일 적용 방법  4. 효율적 앱개발 위한 고려사항 |
| **출제기술사** | 김종환PE (제99회 정보관리 / [e3jake@gmail.com](mailto:e3jake@gmail.com)) |

* **모바일 앱개발의 특성과 이슈**

|  |  |  |
| --- | --- | --- |
| 특성 | 이슈 | |
| 다양한 환경 | Mobile  Fragmentation | - 동일한 기능과 디자인으로 다음과 같이 다양한 환경에서 다른 방식의 개발을 진행하는 이슈  - iOS, Android, Window Mobile 등 다양한 OS 지원  - Device, Version 별 미묘한 차이 발생 |
| 등록기간 필요 | Mobile Application 등록/심사 기간 | - 필수적으로 App 등록/심사 기간 중 프로젝트 공백이 발생  - Guide Line을 준수한다 해도 심사 결과를 예측할 수 없음 |
| 실제 Device 테스트 | 임베디드 성격의 테스트 방식 | - 개발 환경(에뮬레이터)와 실제 Device에서의 테스트 결과에 미묘한 차이가 있음 |
| 빠른 기능변화 | 빠른 출시 요구  (시장 선점 우선) | - 일반적으로 품질의 완전성보다는 출시일 준수를 요구  - 여러 차례에 걸쳐 기능을 추가하는 단계별 출시 성격 |
| 다양한 유지관리 | 유지보수 범위 | - 잦은 OS 업데이트와 Device 출시로 인한 유지 보수 범위에 대한 입장 차이  - 발주사는 Platform 업데이트에 지속적 무상 대응을 원하지만, 현실적 어려움 존재 |

* **모바일 앱(app)개발 환경과 애자일 개발 환경의 특징**

|  |  |  |
| --- | --- | --- |
| 구분 | 모바일 앱(app) 개발 환경 | 애자일(Agile) 개발 환경 |
| Business Alignment | 빠르게 변화하는 시장의 요구사항으로 불확실성과 역동성이 높음 | 요구사항의 변화를 수용하여 계획보다는 반복적이고 점진적인 개발 접근 |
| Communication Mgmt | 대부분 20명 이하의 소규모 개발팀 | 대면을 통한 직접적인 의사소통과 협의를 통해 자기-조직화(self-organization)되는 운영 |
| Test Method | 단편화 문제로 많은 테스트와 잦은 수정을 반복 | 테스트 주도 개발(TDD), 지속적 통합(CI), 짝 개발(pair programming), 리팩토링(Refactoring)을 통한 코드 품질 개선 |
| Strategy | 빠른 출시(Time to Market)와 지속적인 업그레이드를 요구 | 짧은 개발 주기를 통해 제품을 빠르게 출시하고 사용자 피드백을 받아 버전을 높여나감 |
| Core Concern | 편리한 사용성과 성능이 핵심 | 사용자 가치를 가장 우선시하며 사용자 중심의 개발 접근 |

* **애자일 개발방식의 특징**

|  |  |  |
| --- | --- | --- |
| 구분 | 점진적(Incremental)개발 | 반복적(Iterative)개발 |
| 개념도 |  |  |
| 정의 | 하나의 완전한 아이디어를 구상하고 한번에 하나씩 구현 | 모호한 아이디어에서 출발하여 점차 구체화 |
| 전략 | 무엇을 먼저 내놓고 어떻게 진화할 것인 것? | 무엇을 어떻게 구체화시키고 검증할 것인가? |
| 방향 | 클 사진(Big Picture) | 교정적 내용 |
| 상호보완 | 서로 보완하며 프로젝트의 목표를 향해 유연하게 이끌어감  모바일 앱개발의 특성상 점진적, 반복적 개발을 함께 고려해야 함. | |

* **애자일 개발에서의 점진과 반복개발**

|  |  |
| --- | --- |
| 개념도 | 설명 |
|  | 1) 수평적 의사소통  2) 짧은 반복 개발 🡪 협업체계  3) 리스크 감소 및 시너지 극대화  ※전략에서 화면까지 계층적인 개발구조가 수직적으로 이뤄짐  🡪순차적 과정(from abstract to concrete)이 아닌 반복과 점진 통한 수평적 분할 개발 필요 |

* **애자일 기반 모바일 앱 개발 프로세스**

|  |
| --- |
| 개발 프로세스 |
|  |
| 1. 테스트 중요성 : 여러 단말에서 동작과 성능 확보 위한 테스트 중요 2. 테스트 기대수준 : 사용자 스토리(user story) 🡪 기능, 대상 추출 🡪기대 수준 도출 3. 개발, 테스트 병행 : 지속 통합(CI), 테스트 주도 개발(TDD), 짝 프로그래밍(pair programming) 4. 다양항 도구 활용 : 시뮬레이션 환경 이용(다양한 OS 및 브라우저 테스트) |