|  |  |  |  |
| --- | --- | --- | --- |
| 일자 | Version | 설명 | 담당자 |
| 2011.03.11 | V1.0 | 초안 작성 | 윤진경 |

목차

A1.1.1 개발방법론2-7

1. 정의2

2. 필요성2

3. 구성요소2

4. PSP&TSP3

4.1 PSP 프로세스의 흐름3

4.2 PSP의 특징 및 구조4

4.3 TSP의 특징 및 구조5

4.4 PST&TSP 적용 효과 및 미래6

5. UP6

5.1 소개6

5.2 특징6

5.3 단계별 활동6

6. 채택이유7

A1.1.2 절차8

A1.1.3 산출물 서식9

A1.1.4 가이드 라인10

A1.1.5 ID 부여12

**A1.1.1 개발방법론**

1. **정의 : 소프트웨어 개발방법론이란?**

- 소프트웨어 공학 원리를 소프트웨어 개발 생명주기에 적용한 개념

- 정보시스템을 개발하기 위한 작업활동, 절차, 산출물, 기법 등의 체계

- Methology : Method + Knowledge(Knowhow+Heuristics)

1. **필요성 : 소프트웨어 개발방법론이 필요한 이유는 무엇인가?**

**-** 소프트웨어 위기를 극복하기 위한 방법론 필요성 증대

- 개발경험 축적 및 재활용을 통한 개발생산성 향상(작업표준화/모듈화)

- 효과적인 프로젝트 관리(수행공정의 가시화 포함)

- 정형화된 절차와 표준용어의 제공으로 의사소통 수단 제공

- 소프트웨어 품질의 균일화

- 효율적인 프로젝트 관리(수행공정의 가시화 포함)

- 프로젝트의 유지보수 용이

- 시스템 개발 노하우 축적

1. **구성요소**

|  |  |
| --- | --- |
| **작업절차** | 프로젝트 수행 시 이루어지는 작업 단계의 체계  (단계별 활동, 활동 별 세부작업) |
| **작업방법** | 각 단계별 수행해야 하는 일(절차/작업방법) |
| **산출물** | 각 단계별로 개발해야 하는 산출물 목록 및 양식 |
| **관리** | 진행기록, 계획수립, 진행관리, 품질, 외주, 예산, 인력 관리 등의 기록 |
| **기법** | 각 단계별로 작업수행 시 기술 및 기법의 설명 |
| **도구** | 기법에서 제시된 기법 별 지원도구에 대한 구체적인 사용표준 및 방법 |

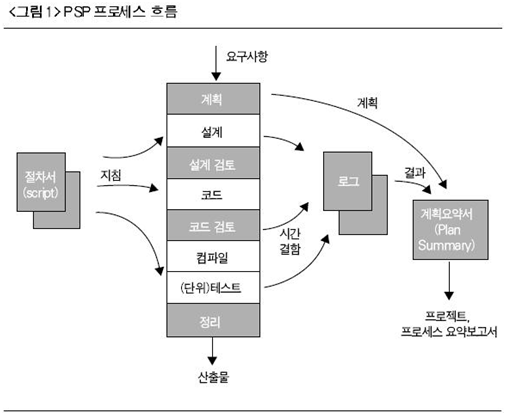
1. **PSP&TSP**
   1. **PSP와 TSP를 하는 목적**

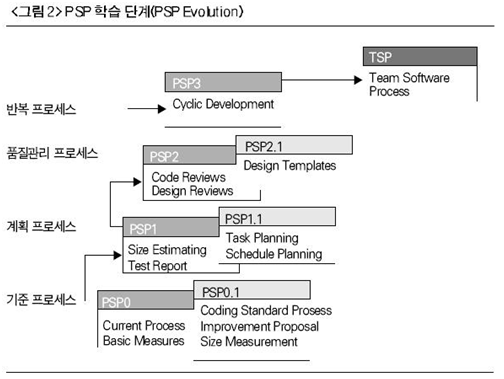
**개인의 개발생산성을 향상 (PSP)**

**조직의 프로세스 개선에 효과적으로 참여(TSP)**

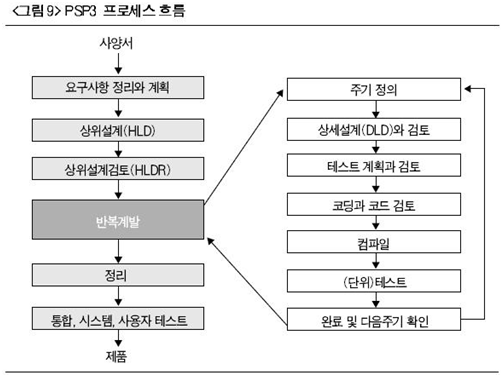
|  |  |
| --- | --- |
| **PSP** | 소프트웨어 엔지니어 개개인에 할당된 과업에 대한 품질 개선과 생산성 향상에 목적을 둔 정량화된 개발 훈련 체계 |
| **TSP** | PSP를 발판으로 CMM의 프레임워크를 개발팀 차원에서 적용함으로 품질 개선과 생산성 향상을 실현하기 위한 방법론 |

**4.2 PSP 프로세스의 흐름**

- 기본적으로 Software Lifecycle을 따르며 매 단계에 대한 측정 기록 및 검토가 있다.

**4.3 PSP 학습 단계**- **PSP 학습**은 기준, 계획, 품질관리, 반복의 4단계의 프로세스로 나뉜다.

|  |  |
| --- | --- |
| **기준 프로세스**  (PSP0) | 현재 자신이 사용하고 있는 개발 상황을 이해하는 단계.  KLOC(Kilo Line Of Code) 당 결함, 시간 등의 측정 기준을  이용하여 프로젝트 계획 요약서(Project Plan Summary)로  정리. 이를 통해 개발 사이클의 매 단계에 대한 자신의  상태를 파악함. |
| **계획 프로세스**  (PSP1) | 수집 자료를 계획에 활용. 작성된 프로젝트 계획 요약서를  통해 신규 개발 프로그램의 크기 확인 및 개발 시간 예상,  일정 수립. 예측 모델로서 PROBE 방법이 권장됨. |
| **품질관리 프로세스**  (PSP2) | 계획과 설계단계에서 요구사항 결함을, 설계검토에서 설계  결함을, 코드 검토에서 코드 결함을 찾아 테스트 결함을  최대한 줄이는 프로세스. 설계 단계의 경우 검토(review)  외에도 설계 검증법이 사용됨. |
| **반복 프로세스**  (PSP3) | PSP의 완성단계로서 PSP0 ~ 2를 통해 전체 개발 과정을 관리하는 방법을 제공. 엔지니어가 충분한 능력을 갖추고 있어야 함 |

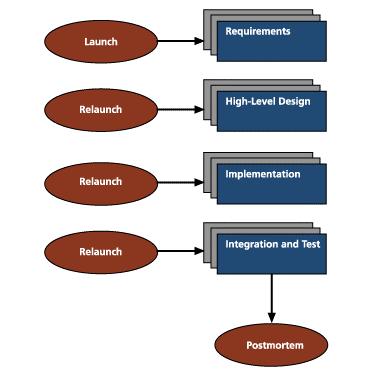
- **PSP3 프로세스 흐름**

**4.4 TSP의 특징 및 구조**  
- 별도의 학습과정이 없고 현장학습을 전제로 함.

- 개발 사이클의 매 단계마다 목표-전략 수립, 요구사항 파악, 일정 수립, 위험 분석을

위한 착수 프로세스(launch, relaunch)가 포함

(때로는 사후검토 - post mortem review가 매 단계마다 포함됨).



**4.5 PSP&TSP 적용 효과 및 미래**

|  |  |
| --- | --- |
| 구 분 | 내 용 |
| 예측력 향상  (Estimating) | - 프로그램 개발에 대한 기초 자료 축적  - 계획 수립의 정확성이 높아짐 |
| 품질 향상 | - 개발 프로세스가 훈련을 통해 정해진 형태로 관리됨.  - 소프트웨어 결함 감소 |
| 개발 기간 단축 | - 결함 감소로 테스트 및 결함 수정 시간 단축 |
| 개발자 동기부여 | - 개발 성과의 축적으로 만족과 보람 제공 |
| 개인 | - 업무의 효율적 수행 가능  - 단결된 팀 구성 및 유지 가능  - 개인과 팀의 능력향상 |
| 프로젝트 | - 정확한 프로젝트 계획 수립 가능  - 계획된 일정 내의 수행능력 향상  - 제품의 품질 목표 설정 및 관리  - 수행 업무에 대한 명확한 보고를 통한 추적 관리 |
| 기업 | - S/W 개발표준 및 전사공통업무 표준이 되는 S/W 프로세스를 재구축  - 정량적 품질관리를 위한 전문인력 및 KPA별 전문인력을 양성  - 정량적 품질관리를 위한 측정기준을 수립 |

1. **UP**

**5.1 소개**

**-** UP(Unified Process)는 1999년 Jacobson, Booch, Rumbaugh에 의해 개발됨

**-** 소프트웨어 개발이 각각 생명주기를 가지는 여러 번의 반복(iteration)을 거쳐

수행되는 모델

**5.2 특징**

**-** 반복마다 실행 가능한 릴리즈가 산출

**-** 반복이 거듭될수록 향상되어 결국 최종 시스템으로 발전됨

**-** 유스케이스 모델(Use Case Model)을 중심으로 분석 모델(Analysis Model), 설계

모델(Design Model), 배치 모델(Deployment Model), 구현 모델(Implementation

Model), 테스트 모델(Test Model)을 산출

**5.3 단계별 활동**

|  |  |
| --- | --- |
| **도입(Inception)** | 소프트웨어 개발 주문에 관련된 사람들(고객, 사용자, 재무적 지원자 등)과의 준비적 상호작용 단계.  요구사항 분석, 원형에 대한 설계, 구현 단계 |
| **상세(Elaboration)** | 어떠한 시스템이 필요한지를 확정하는 단계  요구사항 분석 및 아키텍쳐 확정 단계 |
| **구축(Construction)** | 기본 기능만을 가진 제품 산출 단계로 주요 설계 및  구현을 수행  여전히 릴리즈를 위해서는 기능을 보강해야 함 |
| **이행(Transition)** | 제품 릴리즈 완성 단계로서 구현 및 테스트 수행 |



1. **반복적 모델 적용**
   1. **반복적 모텔**

사용자의 요구 사항, 즉, 정보시스템을 반복적으로 개발하여 완성하는 방법이며, 재사용, 객체지향 개발, RAD 등의 기반이 된 모델임

* + 1. **증분 개발 모델(Incremental Development Model)**

- 폭포수 모델의 변형으로 하향식 구조의 수준별 증분의 분리 개발, 이후에 통합

여러 개의 팀이 나누어 개발 하고자 할 때 유용함

* + 1. **진화적 개발 모델(Evoutionary Development Model)**

- 각 구성요소의 핵심부분을 개발, 개선, 발전시켜 나가는 발전적인 모델

* + 1. **혼합적용 방법**

워터폴 방식으로 순차적으로 개발을 진행함과 동시에 요구사항의 변경이나, 결함 발생을 이유로 각각의 단계가 다시 이루어질 수 있도록 반복적 모델 방법을 사용한다. 아래의 그림과 같이 Requirements -> Analysis -> Design -> Implementation의 워터폴 방식으로 순차적으로 진행하되, 수정해야 하는 상황이 발생 하면 전단계의 Baseline으로 가서 다시 반복한다.

1. **채택이유**

우리 팀은 PSP2 기반으로 TSP 적용하였다. 그 이유는 현실을 고려해 전문 능력을 갖추고 있지 않아 보다 현실 가능한 계획과 설계단계에서 요구사항 결함을, 설계검토에서 설계 결함을, 코드 검토에서 코드 결함을 찾아 테스트 결함을 최대한 줄이는 프로세스인 PSP2를 선택하였다. 그리고 프로세스모델로는 UP을 선택하였다. 그 이유는 위험요소 초기 발견 가능, 아키텍쳐에 대한 정의를 중요한 요소로 삼고 개발하기 때문에 이해가 쉽고 견고하며 변경 관리가 용이하고, 높은 품질을 얻을 수 있는 UP의 장점이 있기 때문이다.

1. **형상관리**
   1. **코드 형상관리**

githurb

* 1. **산출물 형상관리**

**A1.1.2절차**

|  |  |  |
| --- | --- | --- |
| Process | Artifact | 설명 |
| P 1.1 계획 | A 1.1 개발 방법론  A 1.2 프로젝트 계획서  A 1.3 요구사항 추적표  A 1.4 일일보고  A 1.5 조직도 R&R | 팀 구성 / 프로젝트 선정  프로젝트의 일반적인 개요와 개발단계의 방법론 수립  기능을 산정하고 예상 LOC를 예측하여  개발공수, 일정계획 등을 프로젝트 계획 요약서에 기록  팀 역할 및 조직도 구성  문제기술서, 결함조치 보고서 작성  팀 월별 일일보고 작성 후 주차별로 종합 |
| P 1.2 분석 | A 1.7 문제기술서  A 1.8 Use Case Diagram  A 1.9 Use Case Specification | 수집된 요구사항을 분석하여 요구사항 기술서 작성  요구사항 기술서를 바탕으로 Usecase 인식  Usecase Diagram 및 시나리오 작성  기능 / Class를 인식하여 목록 작성  선정된 Class로 Class Diagram 작성 및 Class 명세  실제 화면에 나타날 Interface를 논리/물리적으로 작성 |
| P 1.3 설계 | A 1.10 기능목록/Class 목록  A 1.11 Class 명세  A 1.12 Sequence Diagram  A 1.13 Class Diagram  A 1.14 화면(논리/물리)  A 1.15 ERD/테이블정의서 | 고객의 요구사항의 테스트 성공여부를 파악하여  요구사항 추적서 기록  Sequence Diagram 작성  ERD 작성 및 테이블 정의 |
| P 1.4 구현 | A 1.16 소스코드  A 1.17 CD | 웹으로 만든 후 phonegab을 사용하여 모바일 앱으로  변경. |
| P 1.5  테스트 | A 1.18 테스트결과서  A 1.19 결함조치보고서 | Test Case에 따른 개선방향 분석 / 조치 |
| P 1.6  사후분석 | A 1.20 최종보고서 | 최종적인 산출물을 출력  프로그램을 배포, 프로젝트 공헌도를 팀원 별로 평가 |

**A1.1.3 산출물 서식**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **산출물목록** | **작성도구** | **표지** | **세부사항** |
| A.1.1 | 개발방법론 | Word | 맑은 고딕  28 | 목차, 제목  맑은 고딕  기본폰트 14  내용  맑은 고딕  기본폰트 11 |
| A 1.2 | 프로젝트계획서 | Word |
| A 1.3 | 요구사항 추적표 | Excel |
| A 1.4 | 일일보고 | Excel |
| A 1.5 | 조직도 R&R | Word |
| A 1.6 | 팀원평가 | Excel |
| A.1.7 | 문제기술서 | Word |
| A 1.8 | Use case Diagram | TCC |
| A 1.9 | Use Case Spec. | Word |
| A 1.10 | 기능목록/ Class목록 | Excel |
| A 1.11 | Class 명세 | Ndoc |
| A 1.12 | Sequence Diagram | TCC |
| A.1.13 | Class Diagram | TCC |
| A 1.14 | 화면 (논리/물리) | Excel |
| A 1.15 | ERD/ 테이블 정의서 | TCC/Word |
| A 1.16 | 소스코드(C#) | Word |
| A 1.17 | CD | CD |
| A.1.18 | 테스트 결과서 | Excel |
| A 1.19 | 결함조치보고서 | Excel |
| A 1.20 | 최종보고서 | Word |

**A1.1.4 가이드 라인**

|  |  |  |
| --- | --- | --- |
| **ID** | **산출물목록** | **가이드라인** |
| A.1.1 | 개발방법론 | 개요에 방법론을 어떻게 정의했는지 밝힘  (PSP수준(필수), TSP적용여부, UP 적용 여부)  절차: PSP를 기본으로 팀 프로세스 반영  산출물서식: 수업에서 주어진 서식을 변경할 수 있음.  ID부여  구분1: P (프로세스), A(산출물), G(가이드라인)  구분2: 프로세스 일련번호(액티비티-태스크는 dot으로 구분)  구분3: 이름  예 : P4.1 요구사항정의, A4.1 유즈케이스명세, G4.1 유즈케이스가이드라인 |
| A 1.2 | 프로젝트계획서 | PSP (Humphrey p.112 Table 10.1) TSP 5장  프로젝트 타당성조사 (간략히)  프로젝트 규모계획  규모산정: LOC 또는 FP (최대 – 최소)  투입공수: MH (최대 – 최소)  비용산정: 과기처단가 적용  일정관리 (GANTT차트)  프로젝트 품질관리 / 프로젝트 위험관리 |
| A 1.3 | 요구사항 추적표 | 프로그램 : 기능별로 프로그램이 많을 경우는 횡으로 한 칸씩 계속 적음.  테스트 결과 : S-성공 F-실패 |
| A 1.4 | 일일보고 | UC ID는 REQ, AD, IMP, TEST일 경우 반드시 적을 것.  PM일 경우 적지 않음.  IMP는 LOC를 적음 (Lines Of Code: 설명, 빈 줄을 뺀 java 라인 수)  산출물 구분은 (1) PM (2) REQ, (3) AD (4) IMP (5) TEST (6) MT (7) ED로 구분함. (1) ~ (5)는 산출물번호의 구분임. |
| A 1.5 | 조직도 R&R | H-Chart 방식으로 다음과 같이 작성함  R&R 팀원의 역할 및 책임목록 작성 배분 |
| A 1.6 | 팀원평가 | 밀봉해서 제출함 (팀원간 비밀유지). |
| A.1.7 | 문제기술서 | 현황시스템의 문제점 (외부환경, 고객-사용자로부터)  문제점을 어떻게 개선할 것인지 간편하게 기술하여 시스템의 방향을 적음 |
| A 1.8 | Use case Diagram | StarUML이용하여 표현 |
| A 1.9 | Use Case Spec. | Flow of Events를 따라 사용자와 시스템이 하는 일을 분류.  - 기능별로 시나리오를 나누고 번호를 부여.  - 시나리오 작성.  - 각 시나리오 별 spec을 단계별로 작성.  - 예외처리(Exception) 기술. |
| A 1.10 | 기능목록/  Class목록 | 속성 : 기능은 자세하게 세분화함  메소드 : 가시성을 앞에 표시하고 시그니처 표기함  (예: +turnOn():Boolean) |
| A 1.11 | Class 명세 |  |
| A 1.12 | Sequence Diagram | StarUML이용하여 표현  설계된 시그니처가 정확해야 함(메소드 명은 영문, 매개변수 정확히 해야 함…). |
| A.1.13 | Class Diagram | StarUML이용하여 표현 |
| A 1.14 | 화면 (논리/물리) | 필드 옆에 구분 표시함  RO: Read Only S: Select Box  논리적 화면을 그림. 즉 어떤 데이터가 어느 위치에 있는지를 표시함 (물리적 화면은 실제 화면) |
| A 1.15 | ERD/ 테이블  정의서 | ERD 작성규칙에 따름. |
| A 1.16 | 소스코드 | HARD-COPY로 출력해서 제출함.  보기 좋게 출력함.  불필요하게 문장이 나누어지지 않게 함.  FONT SIZE: 10 FONT: COURIER |
| A 1.17 | CD | CD |
| A.1.18 | 테스트 결과서 | 테스트 케이스 인식 필수 (Junit, CppUnit, TclTest 활용) |
| A 1.19 | 결함조치보고서 | Defect Log (Humphrey PSP p.144 Table 12.2) |
| A 1.20 | 최종보고서 | 마지막 주에 CD와 별도로 작성한 산출물을 묶어서 제출함.  겉 페이지 넣음.  또 설명을 추가할 수 있음. 특히 작성후기, 참고문헌을 넣음. |

**A1.1.5 ID부여**

|  |  |  |
| --- | --- | --- |
| **프로세스** | **산출물** | **가이드라인** |
| P1.1 계획 | A1.1 개발방법론 | G1.1 |
| A1.2 프로젝트계획서 | G1.2 |
| A1.3 요구사항 추적표 | G1.3 |
| A1.4 일일보고 | G1.4 |
| A1.5 조직도 R&R | G1.5 |
| A1.6 팀원평가 | G1.6 |
| P1.2 분석 | A1.7 문제기술서 | G1.7 |
| A1.8 Use case Diagram | G1.8 |
| A1.9 Use Case Spec. | G1.9 |
| P1.3 설계 | A1.10 기능목록/ Class목록 | G1.10 |
| A1.11 Class 명세 | G1.11 |
| A1.12 Sequence Diagram | G1.12 |
| A1.13 Class Diagram | G1.13 |
| A1.14 화면 (논리/물리) | G1.14 |
| A1.15 ERD/ 테이블 정의서 | G1.15 |
| P1.4 구현 | A1.16 소스코드(.java) | G1.16 |
| P1.5 테스트 | A1.18 테스트 결과서 | G1.18 |
| A1.19 결함조치보고서 | G1.19 |
| P1.6 최종보고서 | A1.20 최종보고서 | G1.20 |