# 전공기초 프로젝트

장 진 욱

장진욱 / 경영공학박사

- Tel : 031-960-4306 / Fax : 0504-401-6218
- CP : 010-5676-6218
- E-mail : jinwchng@nonghyup.ac.kr

연구영역

- Fourth Industrial Revolution & digital transformation, location Image deep learning, Software Quality Assurance, Test Process

강의과목

- Python/java 프로그래밍
- 멀티미디어활용
- 컴퓨팅 사고력
- 소프트웨어 공학

경 력

- 스팟인포 대표이사
- 숭실사이버대학교 겸임교수
- 상명대학교 강사
- SK communications PMO Manager
- 국방부 정보사령부 전산장교

주요논문

- Jin-Wook Jang, "Substantiation of location image classification model using projective template matching and convolutional neural network", International Journal of Advanced and Applied Sciences, 9(5) 2022, Pages: 69-74 [SCOPUS]
- 장지욱, "합성곱 신경망 빅데이터 학습을 통한 장소 이미지 식별 서비스 모델 개발", 정보처리학회지 제 29 권 제 1 호, 2022. 3. [KCI]
- 장진욱, "Development of Location Image Analysis System design using Deep Learning", 한국컴퓨터정보학회 논문지, 제27권 01호, 2022. 1. [KCI]
- Jinwook Jang, "Empirical Study of Deep Learning-based Technology for Place Image Collecting", Turkish Journal of Computer and Mathematics Education, Vol.12 No.13, 2823-2829, 4 June 2021. 6. [SCOPUS]
- 장진욱, "Design of Deep Learning-based Location information technology for Place image collecting", 한국컴퓨터정보학회 논문지, 제25권, 09호, 2020. 9. [KCI]
- 장진욱, "The Video on Demand System Failure Evaluation of Software Development Step", 한국컴퓨터정보학회 논문지, 2019. 4. [KCI]
- 장진욱, "A Study on Development of Video Navigation System with real-time GPS Information", 한국컴퓨터정보학회 논문지, 2018. 8. [KCI]
- Jinwook Jang, "Improvement of Automobile Control Software Testing Process Using Test Maturity Model", Journal of Information Processing Systems, 2018. 6. [SCOPUS]

연구과제

- 한국연구재단, “신진연구사업, 스마트영농을 위한 컴퓨팅사고 프로그램개발 및 효과연구“, 2022. 5~2025. 4[연구책임]
- 농림축산식품부, "스마트팜 영농 컨설팅 전문가시스템 개발", 2021. 4~12. [공동연구원]
- 한국연구재단, "생애첫연구사업, 딥러닝을 이용한 위치정보 이미지 수집 및 분류기술 연구", 2020.3~2023.2.[책임연구원]
- 정보통신산업진흥원 , “2019년 SW공학기술현장적용 사업: 넥스젠(주), LMS솔루션 SW품질개선", 2019.4~12.[책임연구원]
- 경기도경제과학진흥원, 창업프로젝트, “딥러닝을 이용한 장소이미지 분석시스템 개발” , 스팟인포 창업, 2019. 5.[책임연구원]

•저 술

- KCU 데이터베이스, 장진욱 지음, 퍼플, 2015.08.24(ISBN 13-1400000252567)
- 개발자도 알아야할 소프트웨어 테스팅 용어, 2007, 에스티에이컨설팅, 검수
- 장진욱, “사용자 스토리 중심의 임베디드 소프트웨어 인수 테스팅 소개” , 임베디드월드, 2009. 1

2013-03-04

# 강의계획

❖ 강의목표
- ➢ 협업을 통한 프로젝트 수행 능력 향상
- ➢ 프로젝트 목표 달성을 위한 창의적 아이디어 도출 및 해결능력 배양

❖ 프로그래밍 설계 및 구현
- ➢ 프로젝트 수행을 통한 프로그래밍 능력 배양

❖ 강의 진행방법
- ➢ 강의, 토론, 실습

❖ 평가기준
- ➢ 참여도 : 출석, 발표, 질문
- ➢ 중간산출물
- ➢ 최종산출물

# 강의계획

1주 오리엔테이션 – 설문, 프로젝트의 의미, 산출물의 정의
2주 프로젝트 계획 - 조편성, 과제도출
3주 프로젝트 대상 선정 – 대상시스템 토의
4주 요구사항 도출 – 아이디어 정리
5주 프로그램 설계 – 프로그램 구성도 작성
6주 프로그램 설계 – 입력 데이터 정의
7주 프로그램 구현 – 출력 데이터 정의
8주 중간발표 – 요구사항, 설계, 구현 발표, 상호토의, 평가
9주 프로그램 구현
10주 프로그램 구현
11주 프로그램 구현
12주 프로그램 구현
13주 프로그램 구현
14주 프로그램 구현
15주 프로그램 구현
16주 최종발표 – 구현 결과 데모, 상호토의, 평가

# 강의 성과 관리

❖ 정량적 평가
- 설문내용 : 최초, 중간, 종강 시
- 활용 : 교육목표를 달성 할 수 있은 항목으로 정의

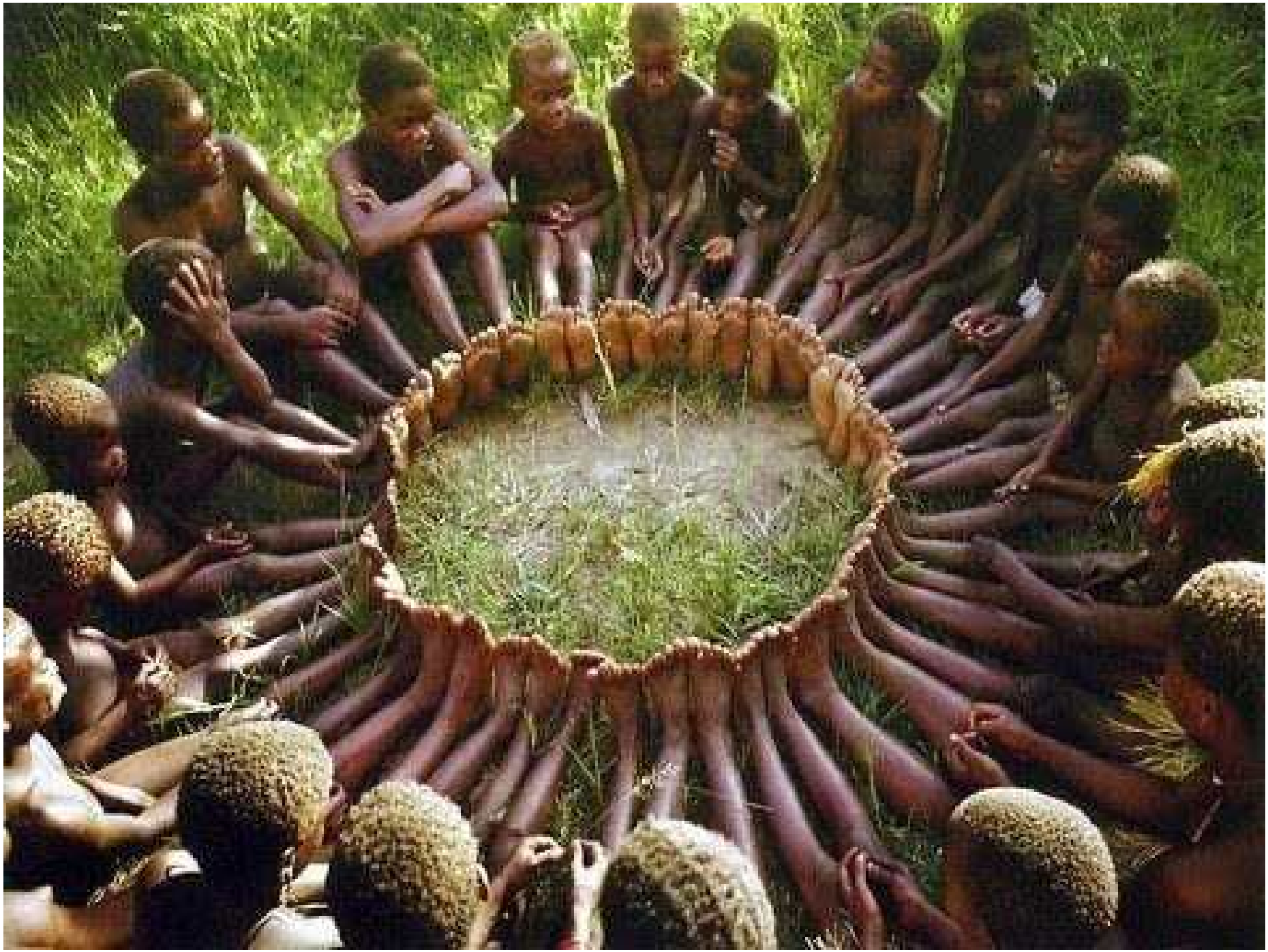
❖ 정성적 평가
- 프로젝트 산출물
- 중간평가 산출물의 충실도, 학생 상호평가 중심
- 기말평가 프로젝트 산출물인 프로그램 구현정도 중심

❖ 강의평가 항목 수업 중 언급
- 학생들의 수긍 유도

# 과목설문
# (과목대표)

# 오리엔테이션


프로젝트 = 함께하는 것



# 프로젝트의 이해

- ✓ 3요소는 비용, 품질, 납기
- ✓ 프로젝트를 통해 달성해야 하는 명확한 목표가 존재하고, 이를 기준으로 성공 여부를 평가
- ✓ 일정상의 제한 (정해진 시작 및 종료일자)
- ✓ 합리적으로 잘 정의된 시작과 끝이 존재
- ✓ 서로 관련되어 있는 과업과 자원들의 통합
- ✓ 특성상 초기의 개념적인 목표에서 시작하여 프로젝트가 진행되면서 점차 구체화/상세화 되는 특징
- ✓ 목표를 달성하기 위해서는 관련된 활동을 세밀하게 조정하고 통합


때로는 의도하지 않은 도움을 받는다

# 세 살 버릇 여든

**처음 계획이 중요**


꺄악!
딱콩
딱콩
이 놈 자식! 엄마가 그러지 말랬지!
도대체 커서 뭐가 될려고 그래?
딱콩
딱콩
요 녀석이! 아직도...
www.headbig.com

# 처음 계획이 중요

| 단계 | 구분 | WBS | 세부항목 | 산출물 | 진행율 | 기간 시작일 | 기간 종료일 |
|---|---|---|---|---|---|---|---|
| 사업관리 | 사업관리 | | | | | | |
| | | 커뮤니케이션 도구 | 서버 도입 | 위키서버 | 100% | 04/02 | 04/03 |
| | | | 서버셋팅 | 위키시스템 | 100% | 04/04 | 04/05 |
| | | | 사용 가이드 작성 | 사용 가이드 | 100% | 04/05 | 04/06 |
| | | 착수보고 | 착수보고 자료작성 | Kick발표자료 | 100% | 04/21 | 04/22 |
| | | | Kick-off발표(참여) | | | | |
| | | WBS(UBCare) | WBS작성 및 보완 | WBS | 100% | 05/04 | 05/06 |
| | | 사업변경신청 | | 변경신청서 | 100% | 04/27 | 05/11 |
| | | 워크샵 | | 회의록 | 0% | | |
| | | 주간보고 | 주간보고(매주) | 주간보고(위키 시스템) | 100% | | |
| | | 월간보고 | 월간보고(4월) | 월간보고서 | 13% | 04/30 | 11/30 |
| | | 회계 법인 검토 | | 사업비 관리대장 및 증빙 | 0% | 11/23 | 11/27 |
| 교육 | 테스팅 교육 | | | | | | |
| | | 테스팅 교육 | Test Driven Development 교육 | TDD교육자료 | 0% | 05/18 | 05/20 |
| | | | 테스팅 실무 교육 | 소프트웨어 테스팅 실무 교육 자료 | 0% | 06/16 | 06/19 |
| | | | 테스트 설계 기법 교육 | | 0% | 06/16 | 06/19 |
| | | 자동화 교육 | Case Maker | 교육자료 | 0% | 06/23 | 06/23 |
| | | | TPMS | 교육자료 | 0% | 06/25 | 06/26 |
| | | | CodeScroll | 교육자료 | 0% | 06/08 | 06/09 |
| | | | Resort | 교육자료 | 0% | 07/22 | 07/23 |
| | | 프로젝트 참여자 교육일정 수립 | ※현재 일정 미정 | | | | |
| 진단 | TMMI진단 | | | | | | |
| | | TMMI 진단 계획수립 | | 계획서 | 100% | 04/16 | 04/17 |
| | | TMMI 진단 실시(인터뷰, 문서검토 수행) | | 인터뷰 내용 정리 | 100% | 04/20 | 04/23 |
| | | 집단 지성 토론 | | 진단 지성토론 결과물 | 100% | 04/20 | 04/23 |
| | | 진단결과 보고서 작성 | | | 100% | 04/27 | 05/08 |
| | | 결과보고서 리뷰 | | | 0% | 05/08 | 05/12 |
| | | 진단결과보고 | | 진단결과보고서 | 0% | 05/12 | 05/12 |
| 정책 | 정책 프로세스 구축 | | | | | | |
| | | 수요기업 정책 분석 | | | 50% | 05/18 | 05/22 |
| | | 정책 프로세스 커스터마이징 | | 테스트 정책 프로세스 | 0% | 05/25 | 05/29 |
| | | 정책서 작성 | | 테스트 정책서 | 0% | 06/01 | 06/03 |
| | | 정책서 리뷰 | | | 0% | 06/04 | 06/04 |
| 전략 | 전략 프로세스 구축 | | | | | | |
| | | 수요기업 전략 분석 | | | 0% | 07/27 | 07/31 |
| | | 전략 프로세스 커스터마이징 | | 테스트 전략 프로세스 | 0% | 08/03 | 08/07 |
| | | 전략서 작성(리스크 기반 전략 수립) | | 테스트 전략서 | 0% | 08/10 | 08/14 |
| | | 테스트 전략서 리뷰 | | | 0% | 08/17 | 08/21 |
| 프로 | 프로젝트 레벨 프로세스 구축 | | | | | | |
| | | 프로젝트 테스트 관리 프로세스 수립 | | 테스트 프로세스 | 0% | 08/24 | 08/28 |
| 레벨 | 레벨 프로세스 구축(일부) | | | | | | |
| | | 레벨 테스트 대상선정 | | | | | |
| | | 레벨 테스트 프로세스 커스터마이징 | | | | | |

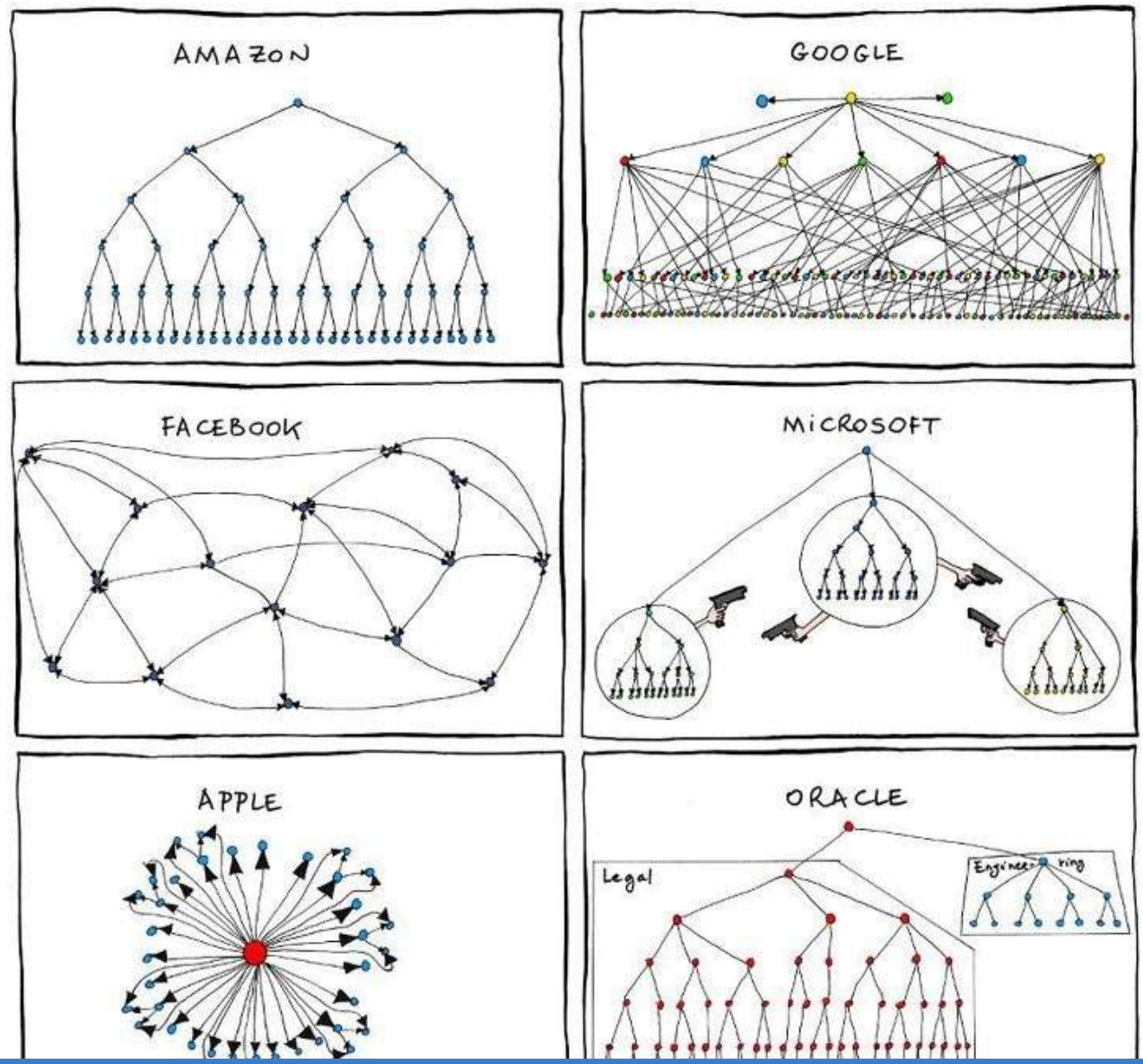
# 프로젝트 계획 WBS

## 착한기업지수(GBI) 상위 1~10위

| 순위 | 기업(브랜드) | GBI |
|---|---|---|
| 1 | 유한킴벌리 | 74.0 |
| 2 | 우정사업본부(우체국택배) | 71.1 |
| 3 | 한국야쿠르트 | 69.1 |
| 4 | 아시아나항공 | 69.0 |
| 5 | 대한항공 | 68.4 |
| 6 | 교보문고 | 68.2 |
| 7 | 다음커뮤니케이션 | 68.0 |
| 8 | 풀무원 | 67.8 |
| 9 | LG전자 | 67.7 |
| 10 | 아모레퍼시픽 | 67.6 |

자료: 서울여대 착한경영센터

# 우리도 착한 조를 찾아요




# 조편성 : 문제해결을 위한 조직



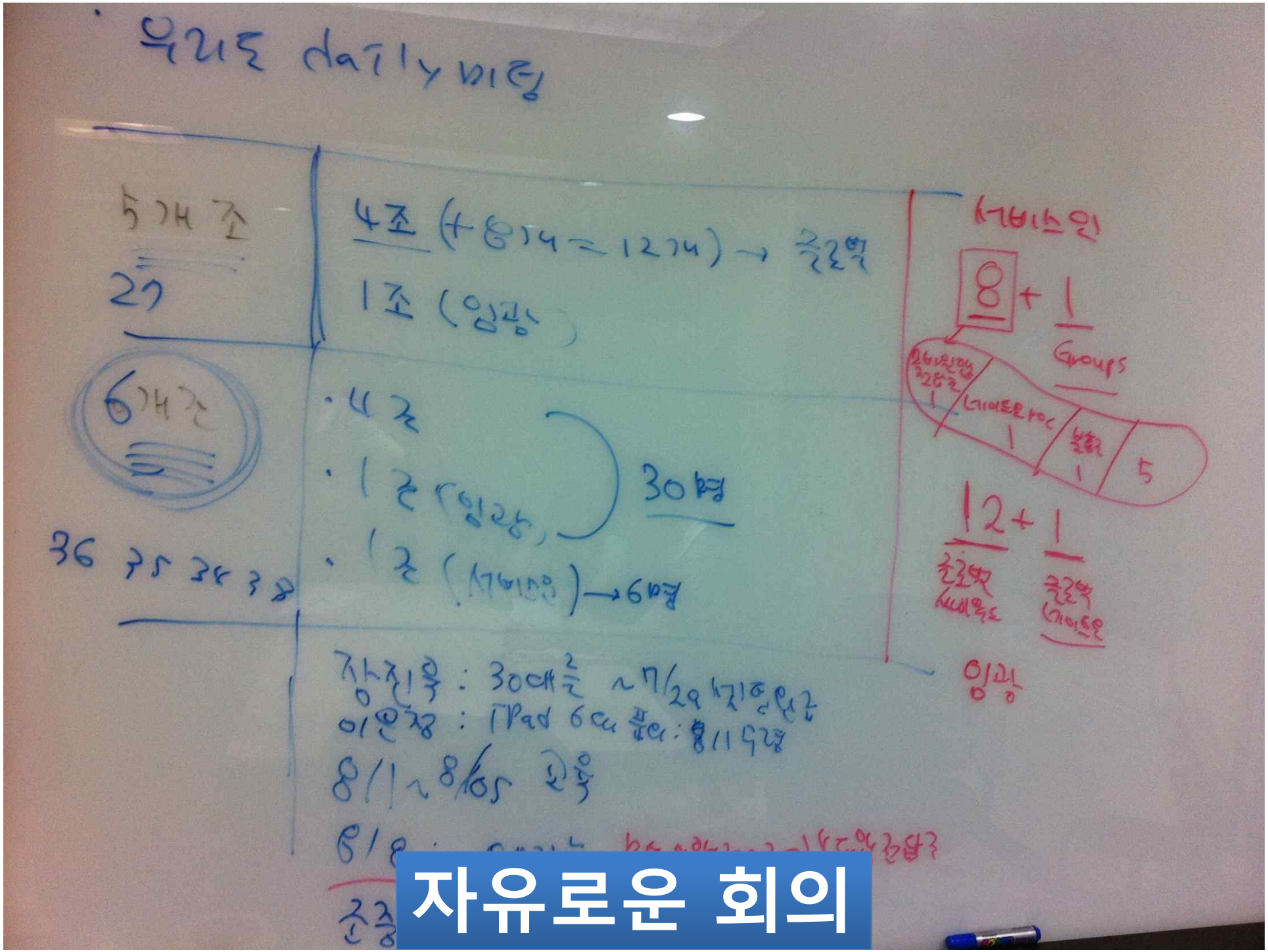
iPhone 10
The tallest iPhone yet.

iPhone30
再一次，改变沟通的方式


# 다양한 아이디어를 해결하는 과정


# 다양한 아이디어를 해결하는 과정


자유로운 설계


자유로운 회의

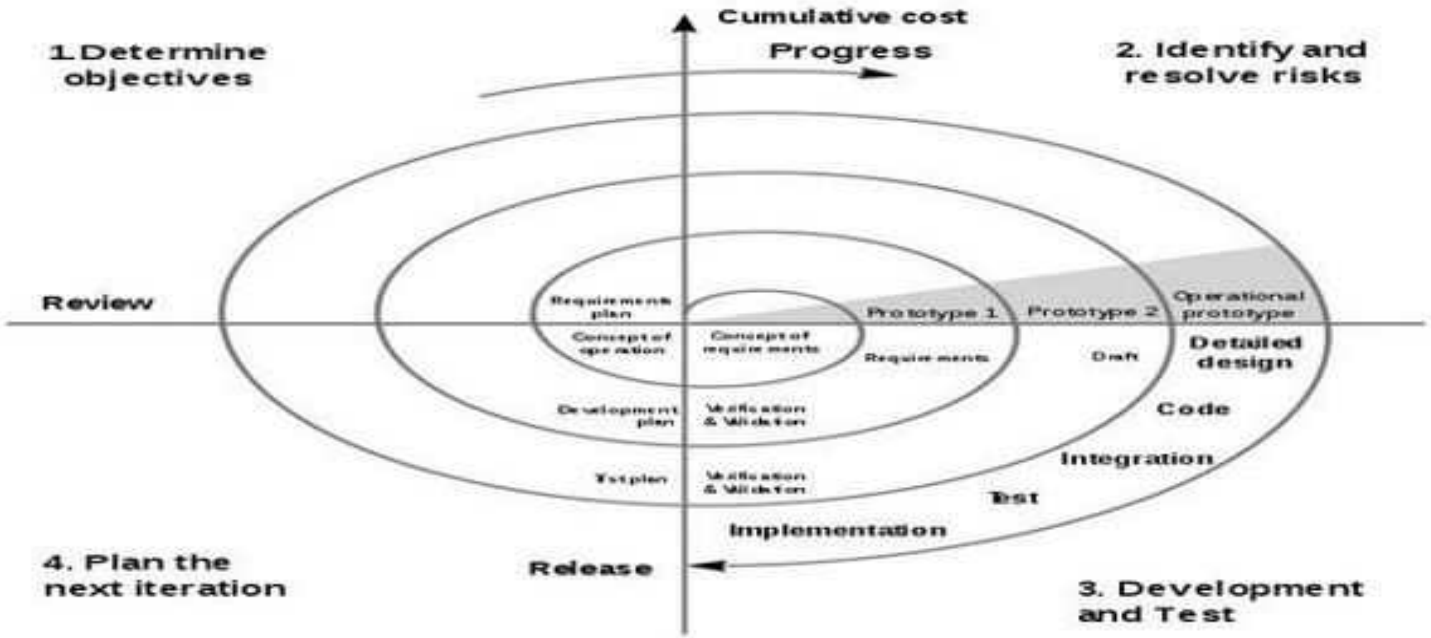
## 스파이어럴 모델(Spiral Model )

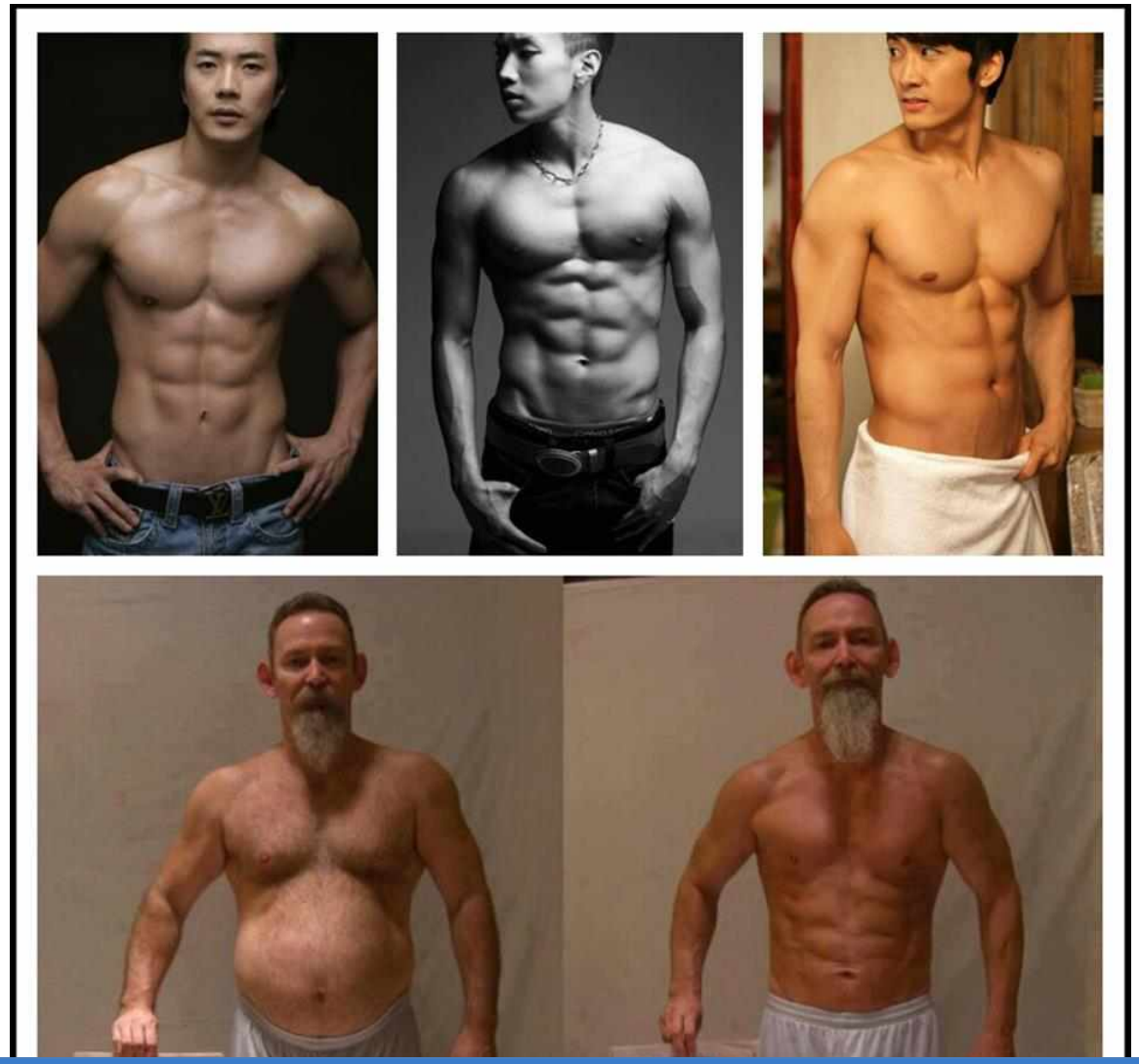
스파이어럴 모델은 소프트웨어 개발 프로세스에서 설계와 프로토타입을 합친 모델로서, TOP-DOWN과 BOTTOM-UP의 장점을 모두 갖추고 있다. 이는 IT를 활용한 시스템 개발방법론이며, 전형적인 Water-fall(폭포수형 방식)에 prototyping(프로토타입 방식)을 접목하여 대규모의 복잡한 프로젝트에 적용하여 위험을 줄일 수 있다. 스파이어럴 모델은 배리 보엠이 1986년의 논문, "A Spiral Model of Software Development and Enhancement"에서 정의되었다.

스파이어럴 모델은 반복적인 개발을 통해 SW product의 개선된 버전을 허용하며, 설계와 고객의 검토를 반복/진행하게 된다. 이는 소프트웨어 개발에 위험관리를 포함하며, 기술적인 측면과 관리적인 측면에서 위험을 인식하게 한다. 또한 어떻게 위험을 줄이고 계속적인 개발과정이 진행될 수 있도록 한다. 스파이어럴 모델은 요구정의와 분석, 시스템 설계, 그리고 실행을 위한 주요 SW product의 반복적이고 지속적인 정제에 기본 개념을 두고 있다. 중심부에서 시작하여 4단계의 반복과정을 거치면서 시스템이 진화적으로 발전한다.

- 1단계 : 목적을 결정하고 새로운 반복을 위한 제약사항과 대안을 결정
- 2단계 : 대안을 평가하고 위험의 정의 및 해결
- 3단계 : 본 반복과정의 SW product 개발과 검증 및 테스트
- 4단계 : 다음 반복과정을 위한 계획수립

스파이어럴 모델을 적용할 경우 폭포수 모델에 비하여 SW 기능을 더욱 빨리 고객에게 보여줄 수 있는 장점이 있으며, 위험과 불확실성을 관리할 수 있다. 이는 다음 단계의 진행을 위한 가시적인 의사결정을 가능하게 하기 때문이다.





# 프로젝트 기초 체력 만들기

# 회 고

## ➢ PMI (Plus/Minus/Interested)

| 구분 | P | M | I | 등등.. |
|---|---|---|---|---|
| 내용 | | | | |

# 준비 사항

- ✓ 우리가 구현해 보고 싶은 프로그램
- ✓ 우리가 만들어 보고 싶은 프로그램
- ✓ 언제 시작해서 언제 완료할까
- ✓ 우리는 어떤 프로그램을 만들 수 있을까
- ✓ 프로그램을 만들려면 어떤게 필요할까
- ✓ 계획, 설계, 구현, 테스트를 통하여 좋은 소프트웨어를 만들 수 있을까

- ✓ 이 시간을 통하여 얻고자 하는것
- ✓ 프로젝트의 이해
- ✓ 프로그래밍 설계의 필요성
- ✓ 과연 성공한 프로젝트는 어떤 프로젝트이고 어떤기준으로 설명할 수 있는지


여자
남자
Maraschino
Cayenne
Maroon
Plum
Eggplant
Grape
Orchid
Lavender
Carnation
Strawberry
Bubblegum
Magenta
Salmon
Tangerine
Cantaloupe
Banana
Lemon
Honeydew
Lime
Spring
Clover
Fern
Moss
Flora
Seam Foam
Spindrift
Teal
Sky
Turquoise
Red
Purple
Pink
Orange
Yellow
Green
Blue


개
개발자
Gray #f94433
Gray #ac203b
Gray #85343d
Gray #874994
Gray #663c84
Gray #8c2590
Gray #a16799
Gray #af99c7
Gray #f38da3
Gray #d2157b
Gray #ec90b7
Gray #e90086
Gray #f57d7e
Gray #f27727
Gray #fc9b7b
Gray #f7d305
Gray #f1e311
Gray #ccdf82
Gray #68bd46
Gray #0aae4f
Gray #069665
Gray #057054
Gray #3ba246
Gray #abcf37
Gray #68c3b2
Gray #8bccd0
Gray #0687a7
Gray #078dca
Gray #0fb8b5


먼저 갑니다!
6시 땡!
잠깐! 강대근씨!
요즘 하루도 빠짐없이 칼퇴근하는데 너무 심한 거 아닌가?
죄송합니다. 약속이 있어서요.
매일 약속이 있다고? 자네, 친구도 별로 없잖나?
하하, 친구와의 약속이 아니구요.
설마 애인?
저 자신과의 약속이죠.
절대 야근을 하지 않겠다는!

# 집단 지성 토론 소개 (Introduction of World Cafe)

# 주제선정

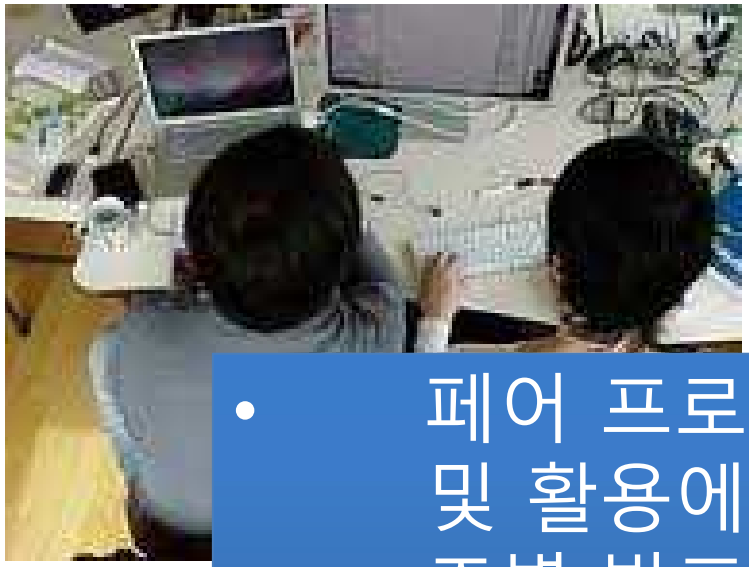
조명 : (1~8조) **********
조장/조원 :
개발주제 : .....
.....
.....자유로운 아이디어.....
.....
.....
.....
개발언어 :
개발환경 :
일정 : 2강, 3강,....

조원 1 : 역할의 정의(리더, 개발리더, 지원자, 문서작성자 등)
조원 2 : 역할의 정의
조원 3 : 역할의 정의
조원 4 : 역할의 정의
(조원 5) : 역할의 정의

☞**자유토의 사진을 찍어서 제출, 조장 내용 보관, 발표, 칭찬하기**

# 중간과제

# 과제명



- 페어 프로그래밍을 조사하고 그 효과 및 활용에 대하여 PPT 3장으로 정리 조별 발표

1. 예) 페어 프로그래밍 정의
2. 방법 및 효과
3. 우리의 활용안

# 성공을 막는 13가지 작은 습관

1. 맞춤법 실수
2. 행동보단 말
3. 성급한 결정
4. 불평불만
5. 허풍떨기
6. 남 탓하기
7. 요령 찾기
8. 열정 있는 척
9. 목적 없이 살기
10. 부탁 다 들어주기
11. 인생을 쉽게 생각
12. 생각없이 행동
13. 현실부정

# 산출물의 정의

# 중간산출물

1. 프로젝트 소개
2. 대상 프로그램 배경, 목적, 사용언어, 환경
3. 개발일정표
4. Product back log-세부 기능목록
5. 시스템 구성도(모듈설계)
6. 개발 프로그램

# 최종산출물

1. 프로젝트 소개
2. 대상 프로그램 배경, 목적
3. 개발일정표
4. 시스템 구성도
5. 테스트 결과
6. Showcase

# Q & A

# 프로젝트 주제 선정 및 토론
# (집단지성 토론)

# 집단 지성 토론 (Introduction of World Cafe)

- 카페처럼 편안한 분위기에서 사람들이 모여하는 창조적 집단 토론
  - 한 테이블에 3, 4~5명이 앉아 토론 (친숙함 + 편안함)
  - 여러 사람이 테이블을 바꾸며 골고루 대화
- 지식의 공유나 생성을 유도하기 위한 새로운 토론 프로세스
- 국내에서는 2008년 6월 11일, “P-CAMP 세번째 만남” 에서 처음 시행

# 집단 지성 토론 소개 (Introduction of World Cafe)





출처 : www.theworldcafe.com

# 집단 지성 토론 소개 (Introduction of World Cafe)

# 토론 방법 - First Step

## 착석해 주세요 – 임의의 테이블에 앉습니다.

- 1개의 테이블에 5명이 앉습니다.
- 같은 테이블에 지인이 있다면, 다른 테이블로 이동합니다.
- Tip : 월드 카페는 되도록 다양한 사람들이 섞여 앉았을 때 효과가 있습니다.

## 첫 번째 대화(25분) – 대 질문을 큰 주제로 대화

- 먼저 돌아가면서 자기 소개를 할까요? (1분씩)
- 자기 소개 후에는 대 질문을 큰 주제로 하여 대화를 시작합니다.
- 이때 대화 내용을 테이블 보 = 전지에 기록합니다.
- 그림 또는 마인드맵 형식 어떤 것이든 좋습니다.
- 다만, 새로운 사람이 왔을 때 어떤 논의가 있었는지 바로 알아볼 수 있도록 합니다.

## 첫 번째 대화 마무리(5분) – 호스트 정하고 이동하기

- 5명중에 한 명을 테이블 호스트로 정합니다.
- 테이블 호스트를 제외한 다른 분들은 또 다른 테이블로 이동합니다.
- 역시 지인이 있는 테이블은 피하도록 합니다.

# 토론 방법 - Second Step

## 첫 번째 대화 요약 설명(5분) - 호스트

- 호스트는 새로운 사람들을 맞이 합니다.
- 먼저 호스트가 자기 소개를 하고, 현재 테이블보에 그려진 내용을 5분내로 요약 설명합니다.

## 두 번째 대화(20분) – 새로운 주제

- 대 주제를 테마로 새로운 주제를 선정하여 대화를 시작합니다.
- Tip: 이전 테이블에서 나누었던 대화 내용을 염두에 두고, 관련 내용이 나오면 서로 연결 지어 봅니다.
  -> 공통 패턴 찾기

## 두 번째 대화 마무리(5분) – 원래 테이블로 돌아가기

- 호스트는 남습니다.
- 다른 멤버는 원래 테이블로 돌아 갑니다.

## 두 번째 대화 요약 설명(5분) – 호스트

- 여행을 떠난 사람들이 돌아왔습니다.
- 두 번째 대화 내용을 요약해서 들려줍니다.

## 세 번째 대화(20분) – 원래의 대 질문으로 대화하기

- 첫 번째 대화를 한 사람들과 다시 원래의 대 질문으로 돌아와 대화 합니다.

# 토론 방법 - Last Step

## 갤러리 만들기(15분)

- 호스트는 각 테이블보를 가지고 나와 벽면에 붙입니다.
- 불명확한 부분이나 수정 내용이 있으면 새로운 테이블보에 다시 정리하여 그리는 것도 좋습니다.
- 다른 사람들은 벽면에 붙은 갤러리를 구경하며 대화 합니다.

## 공유하기(20분) – 인덱스 카드 만들기

- 호스트는 3차에 걸친 대화에서 가장 중요하다고 생각되는 내용을 인덱스 카드에 적습니다.
- 한 장의 카드에는 한 장의 주제만 적습니다.
- 이 카드들을 모아서 벽면에 붙입니다.
- 유사한 것들은 가까이, 서로 다른 것은 멀리 보냅니다.
- 그룹이 형성되면 큰 제목을 써 붙입니다.

# 토론 가이드

여러 사람이 골고루 대화하도록 합니다.

논쟁의 여지가 있다면 서로의 입장을 분명히 하고 이해하는 수준을 목표로 합니다.

이 테이블의 현재 대화 주제가 대질문과 어떤 관련이 있는지 생각해 봅니다.

여러분들의 대화가 더 원활히 진행될 수 있도록 웨이터/웨이트리스들이 간혹 도와드리기도 할 것입니다.

# 프로젝트 주제

1. 웹 아이디 자동관리 프로그램
2. 동영상 플레이어
3. 소셜 커뮤니케이션 tool
4. 웹 아이디 자동관리 프로그램
5. 정보공유 게시판
6. 테트리스 프로그래밍
7. 가위바이보 묵찌빠 프로그램
8. 전화번호 입력 및 조회 프로그램
9. 기타

A. 선정 주제의 개발배경 및 목적

B. 개발목표

C. 개발내용

D. 프로그램

# 개발주제(예)

| |
|---|
| 안드로이드 화면 분할 공유 |
| Interactive Projection System |
| DOTOPIC: “Dot to dot” workbook application for Android OS |
| 내손안에 즐거움 |
| 털털한 PET |
| 불어 행맨 게임 |
| Food Data Mining |
| Mobile Food Mining |
| 데이터 마이닝을 통한 특정 텍스트 추출/인지 |
| Optimal Traffic Engineering over 3G and WiFi |
| 안드로이드 기반의 모션 인식을 통한 동작 제어 |
| 인공지능 기법을 이용한 센서 장애 예측 기술 개발 |
| 교통 데이터를 통한 실시간 교통 이벤트 등록/처리/감지 시스템 연구 |

# 준비 사항

- ✓ 우리 조(조명 ??) 대상 프로젝트
- ✓ 요구사항설계
- ✓ 필요한 기능은 ?
- ✓ 입력 데이터는 ?
- ✓ 출력 데이터는 ?
- ✓ 우리가 구현해야 하는 로직은 ?

# 테스트

- 테스트 케이스의 정의
  - 특정 소프트웨어 컴포넌트와 ***그 예상결과를 검증하는 상세한 액션***을 기술한다
  - 특정 프로그램 경로를 실행해 보거나 특정 요구사항의 준수여부를 확인하기 위해 개발된 ***입력값, 실행 조건, 예상된 결과값의 집합***
  - 테스트 항목을 위한 ***입력값, 예상된 결과, 실행 조건의 집합을 명세한 문서***

- 테스트 케이스의 형태

**목 적** **조건** **특정 입력값 과 단계** **예상결과 값**

출처 : 소프트웨어 테스팅 마이크로소프트에선 이렇게 한다
, IEEE Std 610.12-1990

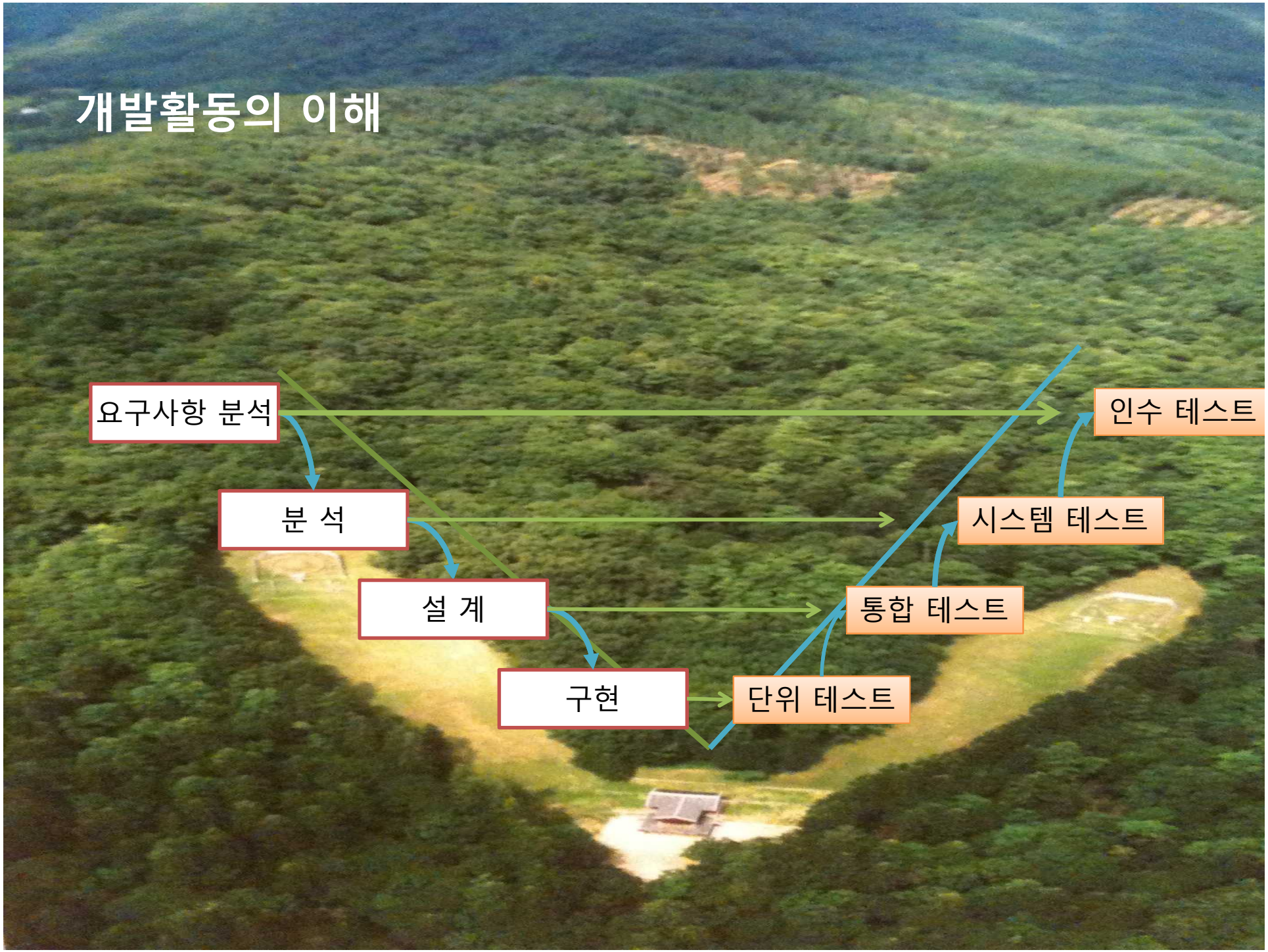
# 테스트 케이스 설계

| 순번 | 대분류 | 소분류 | 주요항목 | 테스트 항목 | 예상 결과(올바른 결과) |
|---|---|---|---|---|---|
| 1 | | | ✓ | | |
| 2 | | | | | |
| 3 | | | | | |

- ❖ 테스트케이스는 하나의 액션을 구체적으로 기술한다.
- ❖ 테스트케이스는 QA 단계에서만 고려되는 것이 아니라, 서비스를 구상하고 구현하는 단계에서부터 고려되어야 한다.

# 프로젝트 진행 Setting

# 개발활동의 이해



# SCRUM

2022

Jinwook Jang

# What is Scrum?

*스크럼(Scrum)은*
*럭비 경기에서 사용하는 용어로 사소한 반칙이*
*일어났을 때 양 팀의 선수들이 하나의 집단을 형성하여*
*그 가운데에 넣어진 공을 발로 빼앗는 대형을 말합니다...*

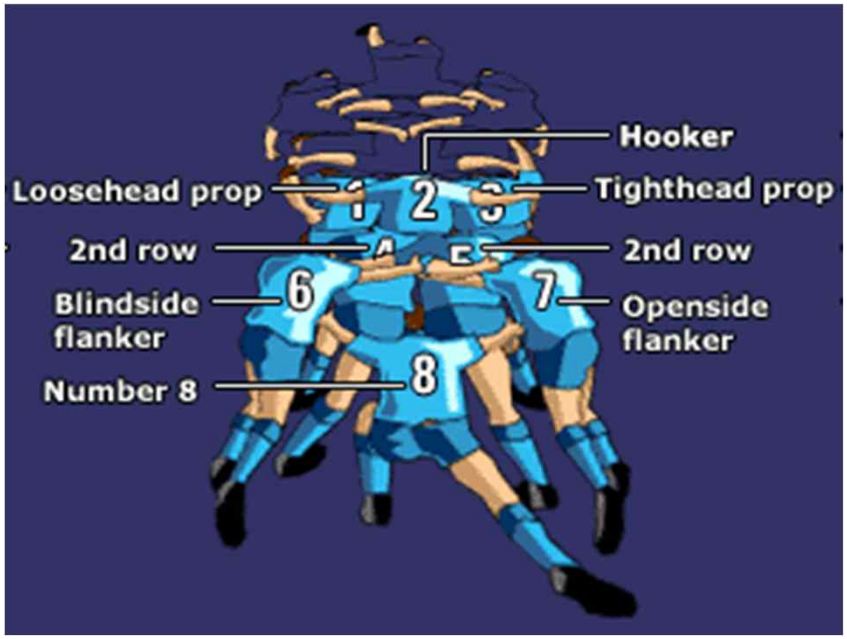
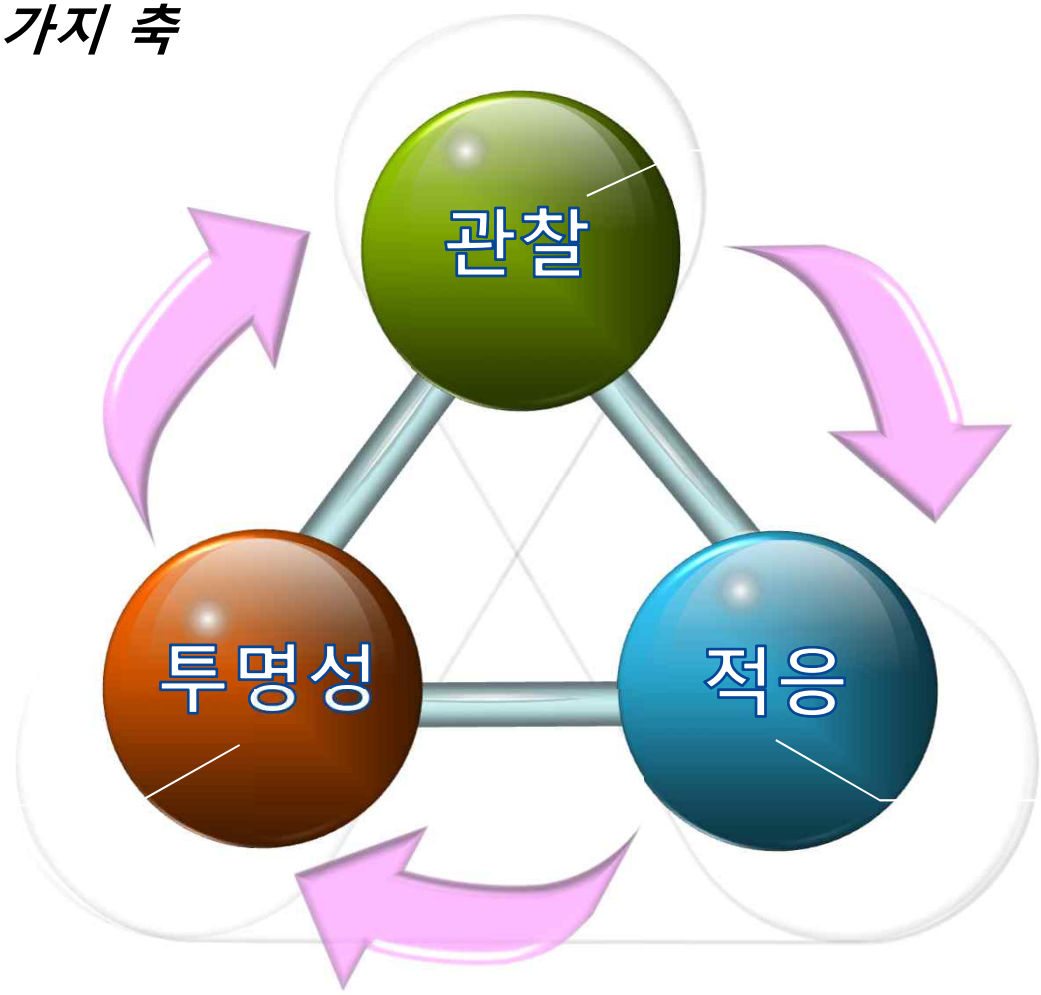



Fig. 6: The Scrum
The scrum is a pushing contest between two teams of eight players. The ball is fed in from the side by the scrumhalf and the team that drives their opponent backwards and hooks the ball out past the last foot of their own scrum wins posession. A scrum down is used to restart play after certain infractions such as a "knock on" or a forward pass.

# 스크럼의 3 요소

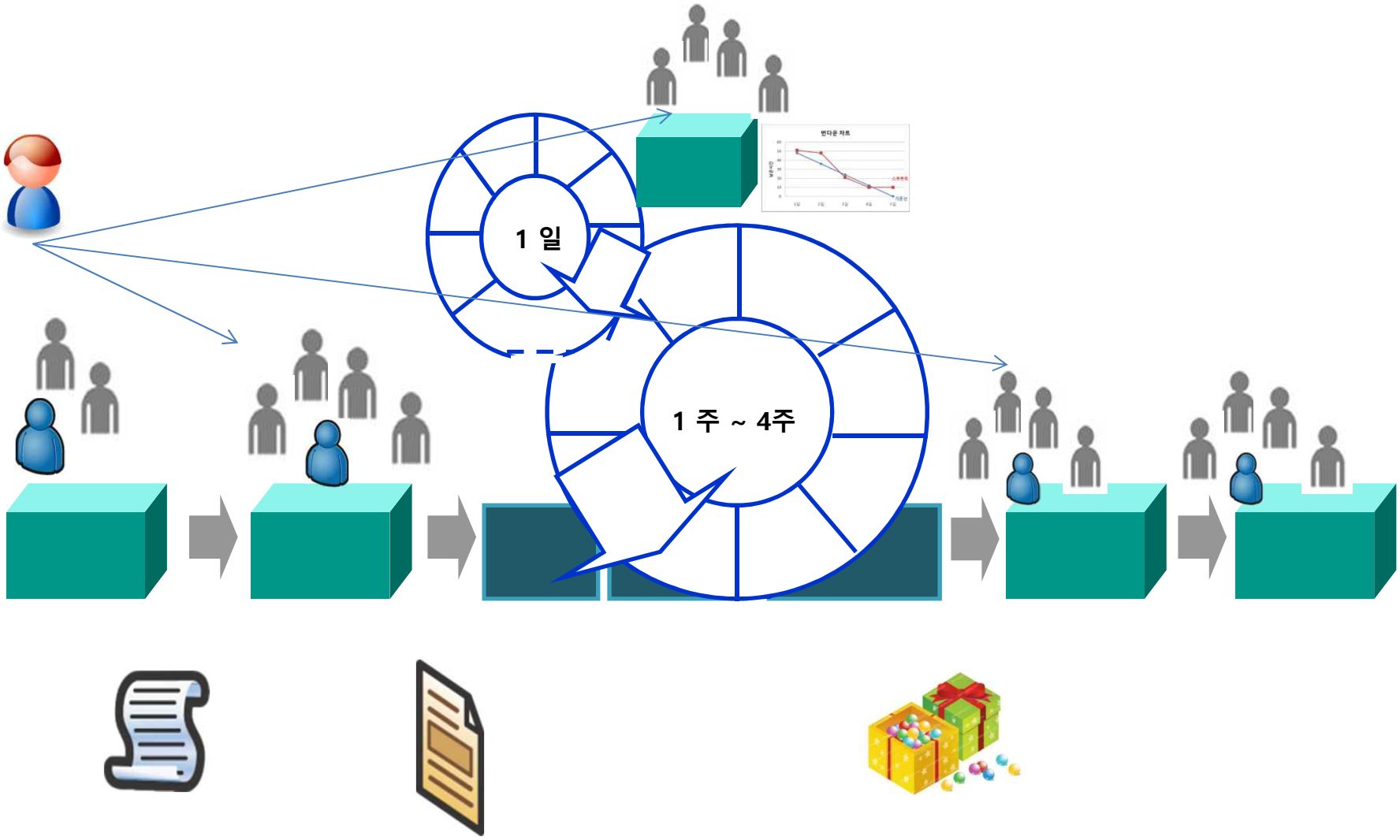
***스크럼(Scrum) 3 가지 축***



# 스크럼 소개

- 애자일 방법
- 생산성을 향상시키는 아주 효과적인 방법
- 주기는 30일 이내 (짧을 수로 효과적이다)
- 스크럼의 구성
  - 스프린트(Sprint)
  - 스크럼 미팅(Scrum Meeting)
  - 제품 백로그(Product Backlog)
  - 스프린트 백로그(Sprint Backlog)
  - 스크럼 마스터(Scrum Master)
  - 제품 책임자 (Product Owner)
  - 팀(team)
- 스크럼 관리
  - Self-Management : Coaching but no one Control

# 프로세스



# 등장인물

- 제품 책임자( Product Owner)
  - 제품 기능정의, 출시일과 출시 내용 결정
  - 제품의 수익을 책임짐(ROI)
  - 시장 가치에 따라 기능 구현의 우선순위를 매김
  - 매 스프린트 마다 기능 및 그 우선순위를 수정 할 수 있음
  - 작업 결과의 승인여부 결정

- 스크럼 팀
  - 교차기능적(Cross-functional), 5명에서 9명 정도
  - 정의된 스프린트 목표를 달성하기 위한 업무를 결정
  - 스스로 관리한다(Self Management)
  - 스프린트 작업 결과를 제품책임자(Product Owner)와 리뷰
  - 스스로 목표를 약속함
  - 개방된 공간에서 함께 배치됨
  - 목표 : 매 스프린트 마다 출시가 가능한 제품을 보여줌

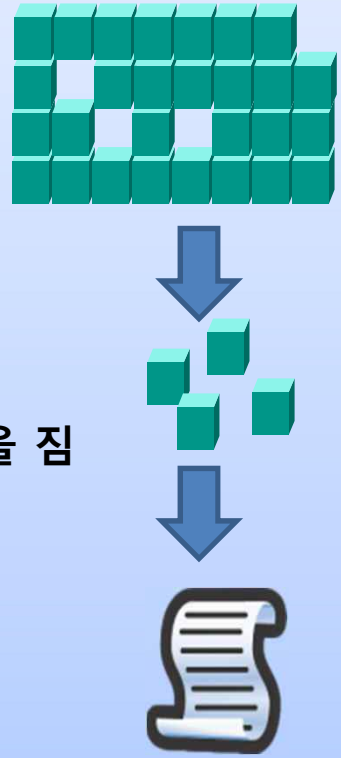
# 등장인물

- 스크럼 마스터(Scrum Master)
  - 팀이 잘 운영되고 생산적이 되도록 노력함
  - 여러 사람과 조직이 서로 잘 협력하도록 도우며, 장애를 제거함
  - 외부의 간섭으로부터 팀을 보호함
  - 프로세스를 준수케 함 (일일 스크럼 미팅, 스프린트 계획 미팅 , 리뷰 ~)
  - 리더이자 조력자
  - 개발과 고객 사이의 장벽을 제거
  - 목표달성 방법에 대한 조언
  - 팀의 창의성과 권한이양을 촉진하여 팀에 생기를 불어 넣는다
  - 생산성 향상의 책임
  - 조직의 변화를 가져오는 역할을 함

    *"However, a dead Scrum Master is a useless Scrum Master"*

# INPUT/OUTPUT

**제품 백로그 (Product backlog)**



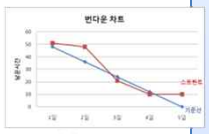
- 제품당 한 개의 목록이 존재함
- 제품에 대한 기능,기술,이슈의 목록임
- 명료, 우선순위, 크기(size) 가 산정되어 있어야 함
- 우선순위가 높은 항목은 상세하게 기술되어야 함
- 제품 책임자( Product Owner)가 우선순위에 대한 책임을 짐
- 누구나 기여할 수 있음
- 알아보기 쉽게 관리되고 게시됨
- 작성자 : 제품 책임자(Product Owner)
- 추가 가능한 항목들 :리스크, 테스트, 다른 제품과의 의존관계, 관련 담당자
- 아이템 하나하나의 완료기준(Done)이 명확해야 한다

# INPUT/OUTPUT

스프린트 백로그 (Sprint backlog)

- 제품 백로그의 아이템을 완료하기 까지 필요한 작업 목록
- 해당 스프린트에서 해야 할 작업만 정리한다
- 작업 목록은  팀이 스프린트 계획을 통해 정한다
- 누가 무엇을 할지는 팀원이 스스로 정한다
- 작업의 크기는 작을 수록 좋다.  한 사람이 최대 8시간 미만 수준
- 남은 작업량을 매일 업데잇한다
- 스프린드 도중에 산정된 남은 작업량을 계속 수정 한다
- 작업목록에는 전체 스프린트의 5~10% 정도는 제품 책임자와 제품 백로그 다듬는 시간을 가진다(Product Backlog Refinement)

# INPUT/OUTPUT



- 번다운 차트
  - 스프린트 기간 동안 , 작업을 완성하는데 필요한 시간을  매일 업데이트 한다
  - 차트를 통해 팀이 해야 할 남은 작업시간을  알 수 있다
  - 실제 소요한 시간을 기록하지는 않는다. 추정만 있음
  - 스프린트가 끝났는데도 번다운 차트에 작업할 시간이 남아있으면 해당 스프린트는 실패한 것임
  - 일일 스크럼 미팅을 통해 남은 작업 시간을 수정 한다

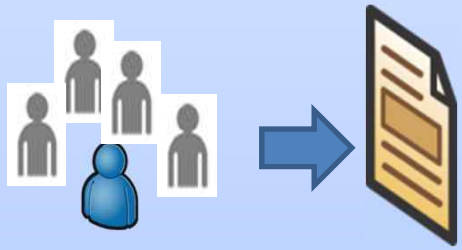

# 스크럼 미팅

1

- 스프린트 계획 1 부
  - who: 제품 책임자 , 팀
  - Input : 제품 백로그 (제품에 대한 이해관계자의 의견 수렴)
  - Task : 요구사항 명확화 및 아이템 선정
  - Output : 제품 백로그
  - 팀원은 아이템에 대한 의견 제시

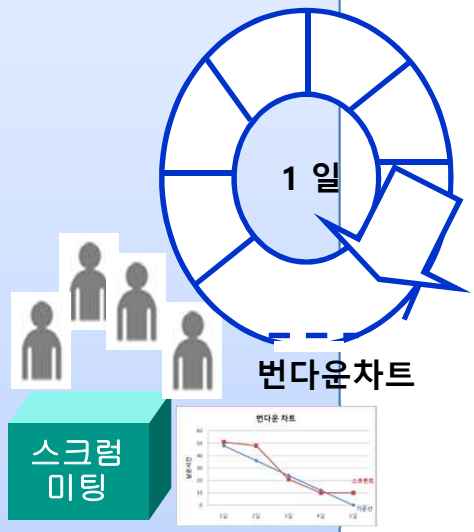
2

- 스프린트 계획 2 부
  - who : 팀, 제품 책임자
  - Input : 선정된 제품 백로그
  - Task : 초기설계 및 계획 (선정된 제품 백로그 아이템을 세분화 하는 작업 포함)
  - Output : 스프린트 목표, 스프린트 백로그, 수정된 제품 백로그
  - 타임박스 운영 : 2 ~ 4시간 이내



# 스크럼 미팅

- 일일 스크럼
  - 매일 15분간 진행 현황 관련 미팅
  - 같은 장소 같은 시간에 실시
  - 3가지 질문
    - 지난 미팅 이후 무엇을 했나?
    - 다음 미팅까지 무엇을 할 것인가?
    - 이슈사항 또는 장애 요소가 무엇인가?
  - 시간준수 (지각 하면 불이익) 의무
  - 토론은 하지 않는다 (토론을 별도 회의를 통해)
  - 제품 백로그에 구체화 할 내용이 있는 지 검토 있는지 확인(refinement)

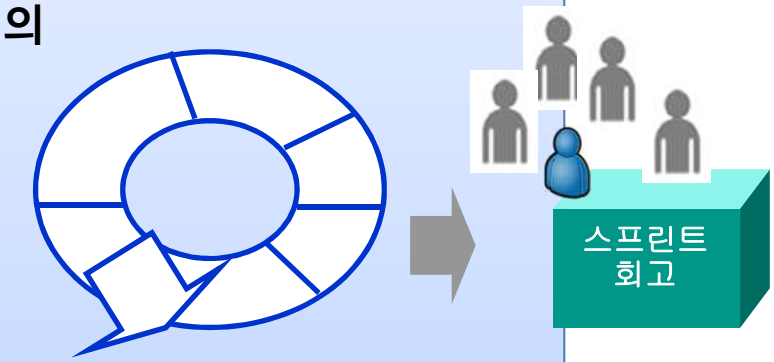


# 스크럼 미팅

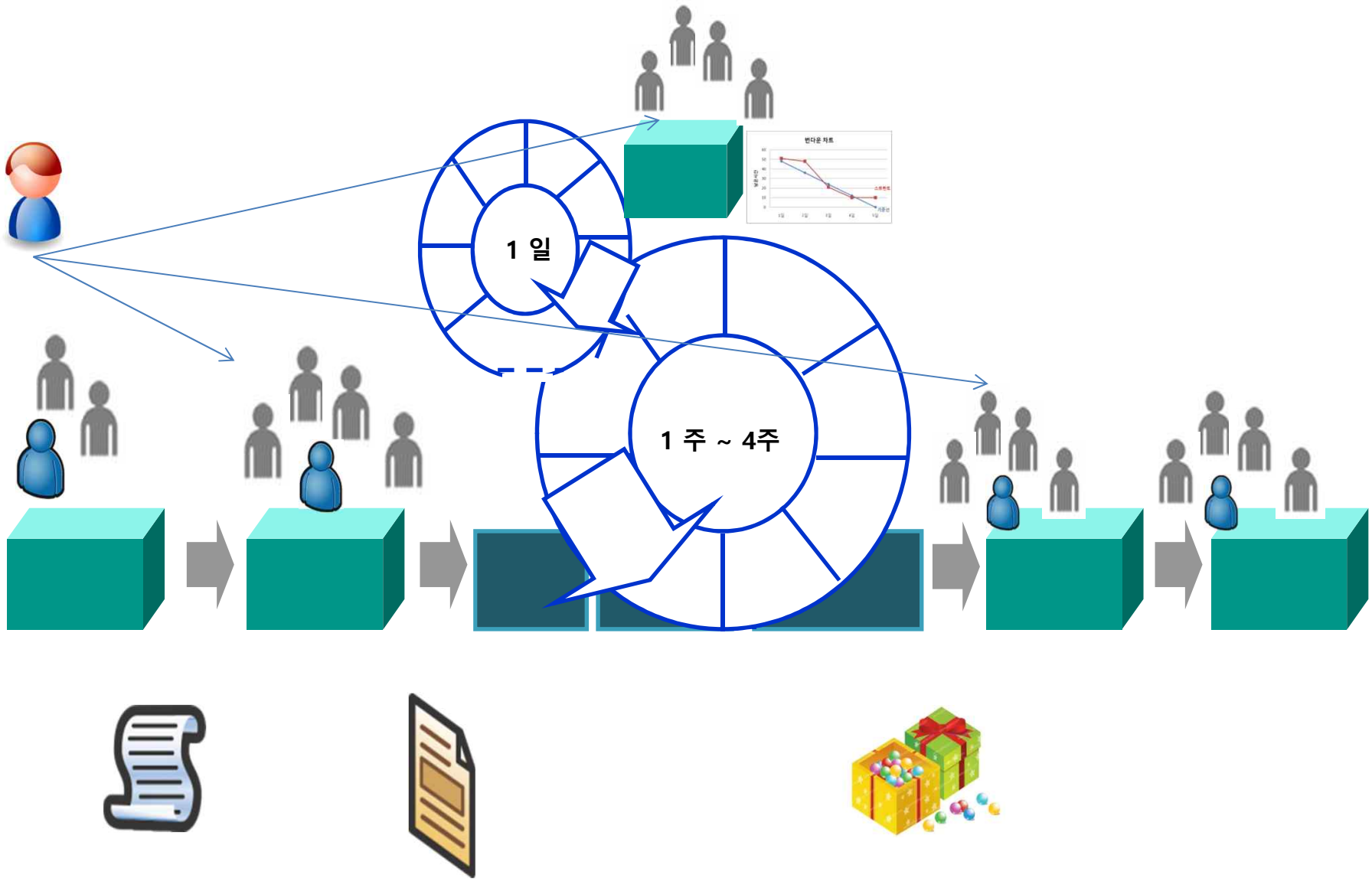
- 스프린트 리뷰
  - 출시 가능한 제품에 대한  관찰 및 적응
  - 데모 보다는 토론에 집중
  - 보고 보다는 공동학습
  - 리뷰를 위해 별도로 준비 하지 않는다.
  - 파워포인트 같은 것을 쓰지 말고 -> 사용자 관점의 데모(인수시험)
  - 참석자 : 스크럼마스터, 제품 책임자, 팀, 이해 관계자
  - 작업이 끝나지 않은 기능은 데모하지 않는다(완료되지 않은 백로그 아이템은 리뷰 대상이 아님)
  - 제품 책임자는 제품 백로그의 우선순위 등을 업데이트 한다

# 스크럼 미팅

- 스프린트  회고
  - 매 스프린트 종료 후 하는 프로세스 개선 회의
  - 스크럼 마스터가 진행한다
  - 잘된 점 , 개선이 필요한 점
  - 가장 골치 아팠던 문제에 대한 해결책 논의
  - 회고 방법
    - 기존 액션을 검토 (완료되지 않은 작업이 있다면 그것 먼저 회고 )
    - 2개 정도의 액션을 집중적으로 회고
    - 생산적 이였는가에 초점을 맞춰서 회고
  - 주요 검토 내용
    - 해야 할 작업들이 잘 정의 되었는가?
    - 완료의 정의가 잘 되어 있는가?
    - 업무에 대한 합의가 업데이트 되어있는가?



# 프로세스



# 좋은 스크럼의 징후

- 산정(Estimation)이 매일 업데이트 된다.
- 일일 스크럼 미팅에 모두 정시에 참석한다.
- 팀원들이 서로 돕는다.
- 팀원들이 도움을 요청한다.
- 문제를 드러내고 같이 해결한다.
- 서로 대화와 상호작용이 많다.
- 팀 특유의 많은 이상한 행동이 나타난다.

# 나쁜 스크럼의 징후

- 스프린트에 계획된 일 보다 훨씬 많은 일들이 발생한다.
- 팀원이 2일 이상 하나의 작업(스프린트 백로그에 정의된)을 진행중이다.
- 산정값이 그대로이거나, 더 커지는데 아무도 이를 신경 쓰지 않는다.
- 일일 스크럼 외에는 서로 상호작용이 없다.
- 제품 책임자를 만나서 물어볼 수 없다.
- 팀 외부로부터 간섭이 많다.
- 숨어 있거나 2개 이상의 백로그가 존재한다.
- 밀린 일이 많은 현상을 그대로 인정한다.
- 매 스프린트마다 잠재적으로 출시 가능한 제품을 만들어 내는 것을 실패한다.

# 병이 보내는 증상표


가래
정상
흰색
분홍거품
심장병
검세
붉은색
폐결핵
무색투명
천식
화농성반투명
폐렴
눈
정상
한쪽 눈 시야장애
뇌졸중 전조
노란색
간질환
잔상, 흰색 눈동자
백내장
손톱
정상
흰색
만성간염
노랑
황달
검정
곰팡이 감염
빈달감소
변비
안색
정상
창백
빈혈
검정
신장질환
노랑
간질환
붉은기
혈액순환장애
소변
정상
혈뇨
요로감염
거품뇨
당뇨병
대변
정상
초록
식중독
검정
위출혈
혈변
치질

녹초가 된 몸으로 늦은 밤 퇴근을 해도
직장인들은 바로 눕기보단 꼭 뭔가를 한다


그 '뭔가'란 대체로 꼭 안 해도 되는 행동들…


그런데도 졸린 눈을
부비며 굳이 하는 이유는



그것이 잠자는 시간을
갉아먹는다 해도 말이지


엄마는 이제 멀리 가서 못와..
우리 현정이가 앞으로 직접 묶고
학교 가야 해. 할 수 있지?
네 엄마..
앞으로

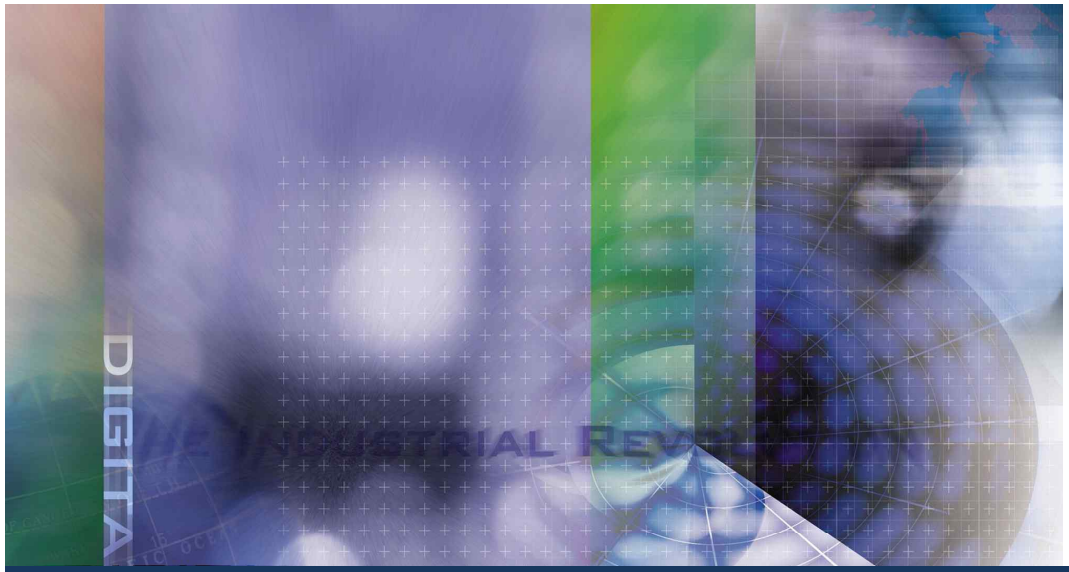

[ 45- 외할머니 소식 ]

스타트!


2차원개그_마인드C® twitter 원작자: @2mindc 배포자: @aroomspace


끝

# 프로젝트 사례(예)



# 스크럼(Scrum)을 활용한 프로젝트 진행사례

## - 레이더(Radar) 개발 -

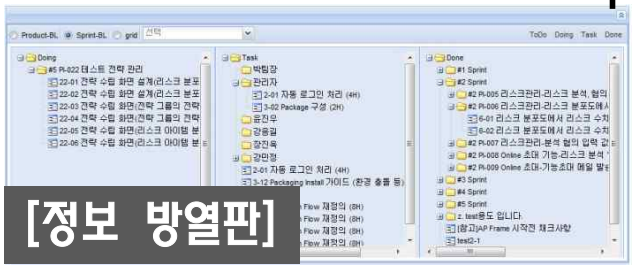
# 목 차

# 1. 프로젝트 소개

- Product 명 : 레이더(Radar) 개발
- Project 개요 : (소프트웨어) 리스크 관리 및 전략 관리 시스템 개발
- 주요기능
  - 리스크 관리 : 리스크 팩터 및 아이템 관리, 리스크 분포도 등
  - 전략 관리 : 전략 아이템, 전략 수립
  - On-line 협업 기능, 사용자 환경 설정 등
- 상 황
  - 짧은 개발 기간 (3/4 분기까지 완료)
  - 적은 참여 인력 (6명)
  - 제품의 완성도 보장

# 2. Scrum 도입 배경과 준비

- 의사결정 회의
  - 시장분석, 고객분석, 제품기능과 활용
  - Scrum 운영 결정, Scrum 팀 결정 (TFT)
- Scrum 팀 구성
  - 조직 구성 - 역할 정의
  - 환경 구성 - 자리 재배치
  - 정보공유 도구 – 정보 방열판
- Scrum 교육
  - Scrum 사상, 진행방식
  - Product / Sprint Backlog 교육



[정보 방열판]

+
- 조직차원 요구
- Scrum Master 교육 및 전파 교육
- 실무적용 시도 공감
- 비공식 Scrum 운영경험
- 기존 개발 방법에 대한 개선 필요

-
- 조직에서의 Scrum에 대한 막연한 기대에 부담
- 새로운 조직의 구성에 따른 업무 효율저하 우려

# 3. Product Backlog

- Product Backlog 정의
  - Scrum 팀 처음 공식 회의(Member 전체 참여)
  - Scrum 운영방식 결정
    - ✓ 전체 Sprint 차수 결정(6차), Sprint 주기 결정(1주)
  - 제품 개발 계획 수립
    - ✓ Sprint별 개발기능 정의
    - ✓ Sprint별 개발규모 산정(단위:일)

• 화요일 ~ 월요일

• 4주차는 BackUp Sprint로 구성 (시행착오 고려)

[Product Backlog]

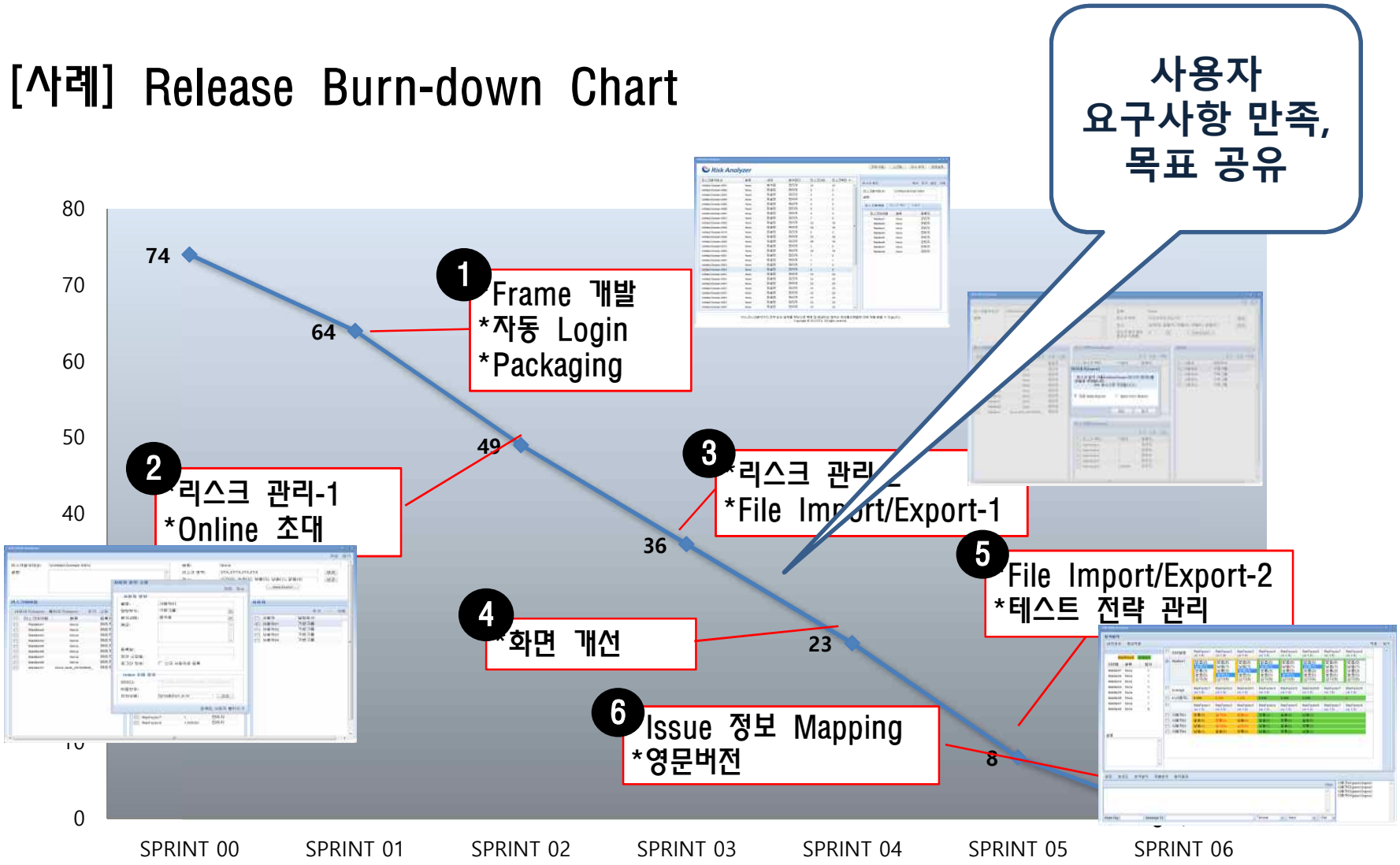
리스크 분석기 Back log

기준 작업 시간 : MD

| Category | Item # | Item Name | Item Discription | Class | S 74 |
|---|---|---|---|---|---|
| AP Frame 처리 | PI-001 | | Web Browser 대신 Standalone 답게 Win Form 에 들어가도록 개발 | 신규 | 3 |
| 자동 Login | PI-002 | | 리스크 분석기를 실행하면 자동 로그인 처리 | 신규 | 2 |
| Package | PI-003 | | Package 배포(실행 File 형태로 배포, Setup 실행 시 PHP, MySQL, Apache 가 사용자 PC에 Setup) | 신규 | 2 |
| Package 구성 | PI-004 | | Uninstall 기능, 디자인, 로고, 평가판/구매판 구분, 다양한 PC 사양에 Install | 신규 | 3 |
| Sprint 01 (2010.06.01 ~ 2010.06.07) | | | | | 10 |
| 리스크 관리 | PI-005 | | 리스크 분석, 협의, 조회 기능 | 신규 | 5 |
| 리스크 관리 | PI-006 | | 리스크 분포도에서 리스크 수치 조정 기능 | 신규 | 5 |
| 리스크 관리 | PI-007 | | 분석 협의 입력 값 자동 저장 | 신규 | 3 |
| Online 초대 기능 | PI-008 | | 리스크 분석 입력 또는 협의에 초대받은 사용자는 제공된 Link를 통해 리스크 분석기에 Web Browser로 접속(Login) | 신규 | 1 |
| Online 초대 기능 | PI-009 | | 초대 메일 발송 기능 | 신규 | 1 |
| Sprint 02 (2010.06.08 ~ 2010.06.14) | | | | | 15 |

# 3. Product Backlog

## [사례] Release Burn-down Chart



# 4. Sprint Backlog

- 회의 진행
  - Sprint 시작 첫날 회의(Scrum 전체 인원 참여)
  - 1시간 내·외 진행(초과할 경우 다음 회의로)
- 진행 내용
  - Product Backlog를 참고한 Sprint별 세부 Task 정의, 규모산정
  - 결과에 따라 Product Backlog Update, Showcase 내용 반영

[Sprint Backlog]

| Prodcut Item | 스프린트에서 작업할 시간 | | 57 | 57 | 45 | 2 | | |
|---|---|---|---|---|---|---|---|---|
| PI-001 | Task1 | 1-01 메신저 소스 정리 & Frame에 삽입 (4H) | 4 | 4 | 0 | | | |
| PI-002 | Task2 | 2-01 자동 로그인 처리 (4H) | 4 | | | | | |
| | Task3 | 2-02 리스크 분석기 자동 로그인 Test (2H) | 2 | 2 | 2 | | | |
| PI-003 | Task4 | 3-01 리스크 분석기 실행시 PHP, MySQL 서비스 Up/Down 처리 | 8 | 8 | 0 | 0 | 0 | Done |
| | Task5 | 3-02 Package 구성 (2H) | 2 | 2 | 2 | 0 | 0 | Done |
| | Task6 | 3-03 Package 사용자 약관 (3H) | 3 | 3 | 3 | 0 | 0 | Done |
| | Task7 | 3-04 리스크 분석기 설치/운영 가이드 (4H) | 4 | 4 | 4 | 4 | 0 | Done |
| | Task8 | 3-05 리스크 분석기 사용자 가이드 (2H) | 2 | 2 | 2 | 2 | 0 | Done |
| | Task9 | 3-06 Packaging Directory 지정 (2H) | 2 | 2 | 2 | 0 | 0 | Done |
| | Task10 | 3-07 리스크 분석기 실행 Test (4H) | 4 | 4 | 4 | 4 | 0 | Done |
| | Task11 | 3-08 Packaging 구성 Test (4H) | | | | | | |
| | Task12 | 3-09 Packaging 초기 설정 (DB, Default) (4H) | 4 | 4 | | | | |
| | Task13 | 3-10 Packaging Install 가이드 (환경 충돌 등) (4H) | 4 | 4 | 4 | 4 | 0 | Done |
| | Task14 | 3-11 Packaging 초기 설정 (DB, Default) Test (4H) | 4 | 4 | 4 | 0 | 0 | Done |
| | Task15 | 3-12 Packaging Install 가이드 (환경 충돌 등) test (4H) | 4 | 4 | 4 | 4 | 0 | Done |
| PI-004 | Task16 | 4-01 리스크 분석기 Logo 디자인 의뢰(2H) | 2 | 2 | 2 | 0 | 0 | Done |

- Task 규모산정 (단위:시간)
- 정보 방열판을 통한Update 및 공유

# 4. Sprint Backlog

## [사례] 피보나치 카드를 활용한 규모산정


☞ Sprint가 거듭될 수록 시간을 줄여줌
(잔여 Task : Sprint #1 '10H' ⇒ Sprint #6 '2H')

# 5. Daily Scrum Meeting

- 회의 진행
  - 매일 오전 출근 후 미팅(9:15)
  - Scrum 전체인원 참여
  - 시간은 15분 이내
  - Stand Meeting(회의실, 휴게실, 커피전문점 등)
- 내 용
  - 이전 Scrum 회의 이후부터 진행한 일
  - 다음 Scrum 회의 까지 진행할 일
  - 장애요소(이슈)

• Scrum 회의에서는 정보 방열판 사용 안함

☞ 정보 방열판은 수시 또는 업무 종료(퇴근)전 Update를 원칙으로 함

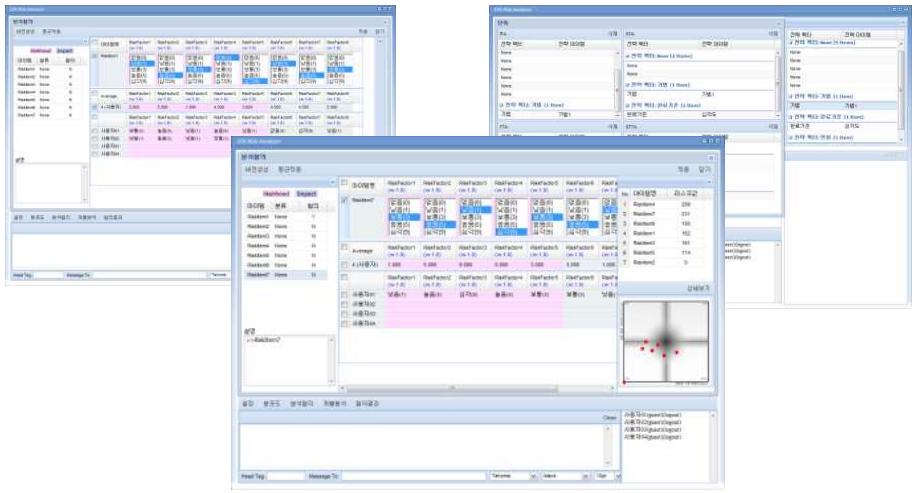
# 6. Showcase

- 회의 진행
  - Sprint 마지막 날 진행 원칙
  - 참여 : Scrum Member, 의사결정권자, 그 외 참여 희망자
  - 시간 : 20분 내외
- 주요내용
  - Sprint 진행내용 Demo
  - 참여자 의견은 다음 Sprint Backlog 회의 시 논의

• 1일 지연된 경우가 2회

• 1시간 정도 소요경험

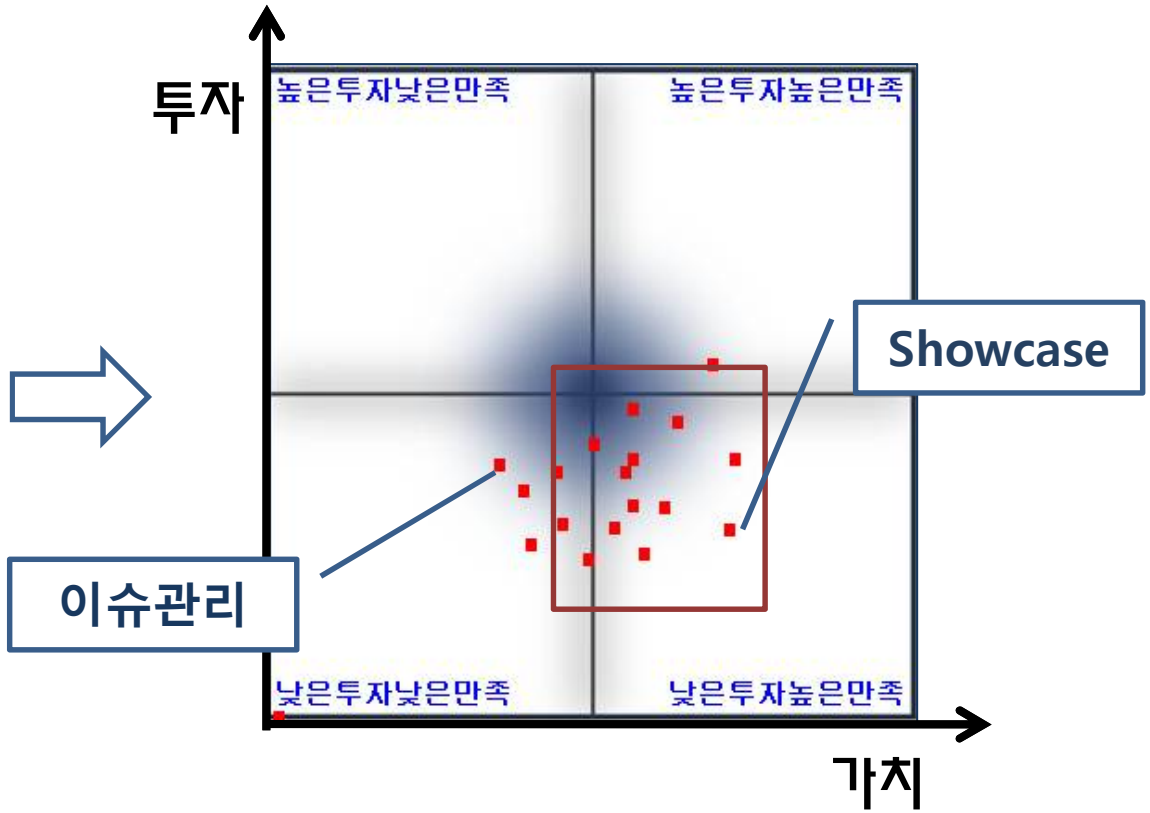
부담스러운 자리지만 성과를 인정받는 자리로



# 7. 자가진단

- Scrum 수행 회고
  - 21개의 Scrum 성공을 위한 프렉티스 및 요소에 대해 자체평가

| No | 아이템 |
|---|---|
| 1 | Standing 스크럼 미팅 |
| 2 | Sprint Backlog 회의 |
| 3 | Sprint Showcase |
| 4 | Pair Programing |
| 5 | Pair Testing |
| 6 | 정보 방열판 활용 |
| 7 | 개발 요구사항 정리 및 기획 |
| 8 | 조직 구성 |
| ..... | |
| 20 | 팀웍 및 인간적인 유대관계 |
| 21 | 자발적 업무 이행 |




☞ 상대적인 투자는 최소화하고 가치를 높이는 방향 선택
(쉽게 시도해 볼 수 있는 항목부터 진행)

# 8. 느낀점

기획한 Radar와 개발된 Radar가 다르지 않았다

업무공유 기회가 많아져 협업이 원활해 졌다

Scrum 팀 자체의 의사결정 권한을 가지고 개발일정을 유연하게 대응할 수 있었다.

집단지성을 활용한 기획과 개발을 통해 단기간에 완성도를 높일 수 있었다

성과를 바로 바로 확인 할 수 있었다

# 감사합니다.

# 프로젝트 회고

- **주 제**

1. **계획대비 미 구현된 기능이 있다면 이유는 ?**
2. **프로젝트 구성원과 함께 하면서 어려웠던 점은 ?**
3. **프로그래밍을 잘 하기위해 필요한 지식은 ?**

# 프로젝트 회고

## ➢ PMI (Plus/Minus/Interested)

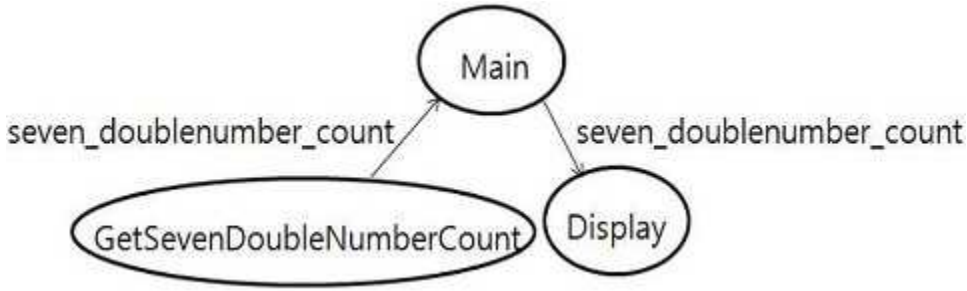
| 구분 | P | M | I | 등등.. |
|---|---|---|---|---|
| 내용 | | | | |

작업중

# C언어 예제 : 설계에서 코드화

**1부터 100,000 사이에 (100,000포함) 7의 배수가 몇 개 있는지 구하시오.**

**1. System chart**




C프로그래밍을 몇번 해보신 분이라면, 왜 단순 출력만 해주는 Display라는 함수를 따로 만들어야 되는지 의문을 가지실 겁니다. 그냥 main 함수에서 출력해주는게 쉽지 않냐고 말씀하실수도 있습니다.
하지만, 지금부터 연산부분과 출력부분(Form부분)을 분리해서 코딩하는 습관을 들여놓으시면, 나중에 편하실겁니다. 여러가지 장점이 있지만, 제가 느낀 가장 큰 장점은 다른언어로 옮길때 엄청난 편의가 있다는 점입니다.

# C언어 예제 : 설계에서 코드화

**2. 수작업의 이해**

| 숫자 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | … |
|---|---|---|---|---|---|---|---|---|
| 7의배수 | X | X | X | X | X | X | ○ | … |
| count | 0 | 0 | 0 | 0 | 0 | 0 | 1 | … |

**3. 처리 과정**

1. 100,000번 반복한다.
    1.1 수를 세린다.
    1.2 7의 배수를 세린다.

2. 7의 배수의 수를 출력한다.

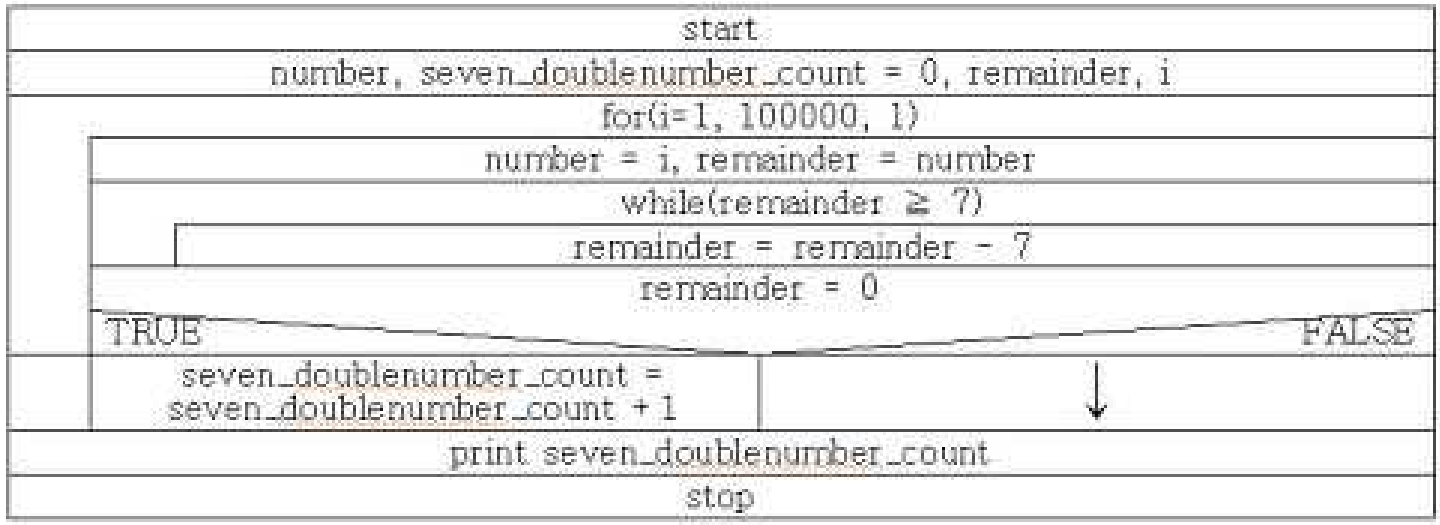
3. 끝낸다.

# C언어 예제 : 설계에서 코드화

## 4. 자료명세서

| number | 숫자 | 정수형 |
|---|---|---|
| seven_doublenumber_count | 7의 배수 Count | 정수형 |
| remainder | 나머지 | 정수형 |
| i | 반복제어변수 | 정수형 |

## 5. NS-Chart




NS Chart를 자세히 보신 분이라면, 왜 나머지를 구하는데 %연산자를 쓰지않고 remainder라는 변수를둬서 나머지를 구하는지 의야해 하실겁니다. 이유는 간단합니다. NS Chart에는 %연산자가 없습니다^-^;; 우선 여기엔 remainder 변수를 사용하여 나머지를 구하고, 실제 소스화 할때는 %연잔자를 사용하도록 하겠습니다.

# C언어 예제 : 설계에서 코드화

**6. NS-Chart 검토표**

| | 초기값 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| number | ? | 1 | 2 | 3 | 4 | 5 | 6 | 7 | ... |
| seven_doublenumber_count | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ... |
| remainder | ? | 1 | 2 | 3 | 4 | 5 | 6 | 0 | ... |
| i | ? | 1 | 2 | 3 | 4 | 5 | 6 | 7 | ... |

초기값 부분에 ?로 표시되어 있는 부분은, 초기화가 되지 않아서 쓰레기값이 있다는걸 표시합니다.

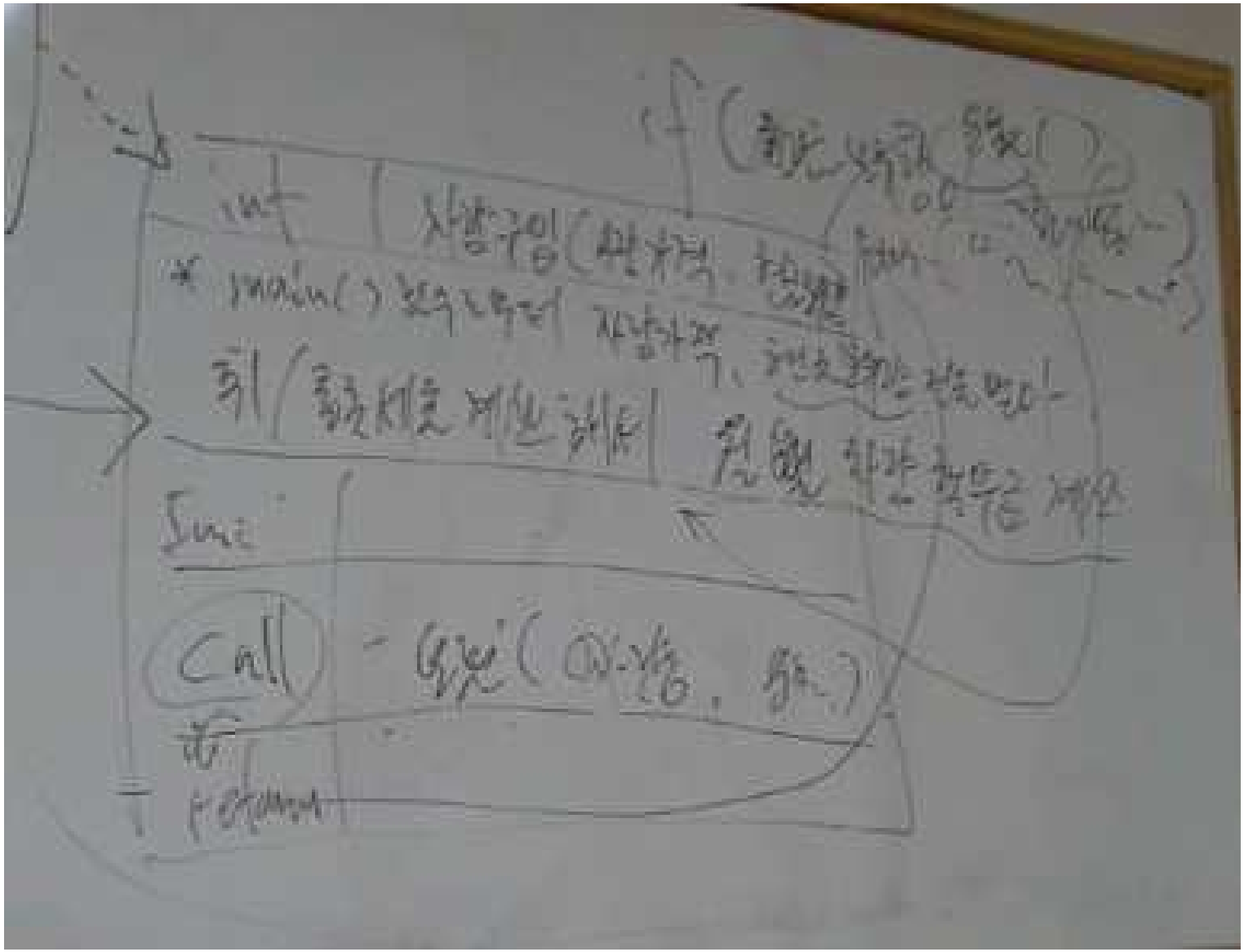
# C언어 예제 : 설계에서 코드화

## 7. 소스화

```
01.#include <stdio.h>
02.
03.#define DOUBLE_NUMBER 7
04.
05.typedef unsigned long int UShort;
06.
07.UShort  GetSevenDoubleNumberCount  (UShort seven_numberdouble_count);
08.void  Display     (UShort seven_numberdouble_count);
09.
10.int main (int argc, char *argv[])
11.{
12.UShort seven_numberdouble_count = 0;
13.
14.seven_numberdouble_count = GetSevenDoubleNumberCount(seven_numberdouble_count);
15.
16.Display(seven_numberdouble_count);
17.
18.return 0;
19.}
20.
21.UShort GetSevenDoubleNumberCount (UShort seven_numberdouble_count)
22.{
23.UShort i;
24.UShort number = 1;
25.UShort remainder;
26.
27.for(i=1 ; i<=100000 ; i++)
28.{
29.number = i;
30.remainder = number % DOUBLE_NUMBER;
31.
32.if(remainder == 0)
33.{
34.seven_numberdouble_count++;
35.}
36.}
37.
38.return seven_numberdouble_count;
39.}
```

view sourceprint?

```
01.#include <stdio.h>
02.
03.#define DOUBLE_NUMBER 7
04.
05.typedef unsigned long int UShort;
06.
07.UShort  GetSevenDoubleNumberCount  (UShort seven_numberdouble_count);
08.void  Display     (UShort seven_numberdouble_count);
09.
10.int main (int argc, char *argv[])
11.{
12.UShort seven_numberdouble_count = 0;
13.
14.seven_numberdouble_count = GetSevenDoubleNumberCount(seven_numberdouble_count);
15.
16.Display(seven_numberdouble_count);
17.
18.return 0;
19.}
20.
21.UShort GetSevenDoubleNumberCount (UShort seven_numberdouble_count)
22.{
23.UShort i;
24.UShort number = 1;
25.UShort remainder;
26.
27.for(i=1 ; i<=100000 ; i++)
28.{
29.number = i;
30.remainder = number % DOUBLE_NUMBER;
31.
32.if(remainder == 0)
33.{
34.seven_numberdouble_count++;
35.}
36.}
37.
38.return seven_numberdouble_count;
39.}
40.
41.void Display (UShort seven_numberdouble_count)
42.{
```



# C언어 예제 : 설계에서 코드화

**8. 결과**

**Output:**

```
1   7의 배수 : 14285개
```

## 7-3-라.설계 원칙

함수를 작성하는 문법과 호출하는 방법, 인수를 받아들이고 리턴하는 방법을 익히는 것은 그다지 어렵지 않다. 그러나 함수를 정말로 함수답게 잘 나누고 디자인하는 것은 무척 어렵고 단기간에 체득되지 않는다. 함수는 프로그램을 구성하는 단위로서 잘 나누어 놓으면 프로그램의 구조가 탄탄해지고 확장하기도 쉽고 재사용성도 좋아진다. 잘 짜여진 프로그램을 분석해 보면 함수의 분할 구조가 감탄스러울 정도로 잘 되어 있음을 볼 수 있고 그런 함수를 만드는 능력이 부러워지기까지 한다.

그러나 함수를 잘못 디자인하면 코드는 더 커지고 프로그램은 더 느려지며 조금이라도 수정하려면 어디를 건드려야 할지 판단하기 힘든 나쁜 구조가 만들어진다. 함수에 의해 코드는 꼬이기만 하고 엉망이 된 코드 사이로 버그가 창궐할 수 있는 환경만 만들어지니 아예 함수를 만들지 않느니만도 못한 상태가 되기도 한다.

프로그래밍에 처음 입문한 사람들에게 함수 디자인이라는 주제는 아주 어렵고 힘든 고비이다. 책을 읽어서 비법을 얻는 것은 불가능하고 잘 하는 사람에게 개인 지도를 받아도 어렵고 혼자서 연습해 보기는 더욱 더 어렵다. 함수 디자인은 오로지 많은 분석과 실습만으로 얻어지는 경험이다. 그래서 꾸준한 연습만이 해결책이다. 다음은 함수를 잘 만드는 기본적인 지침들이다.

**1**

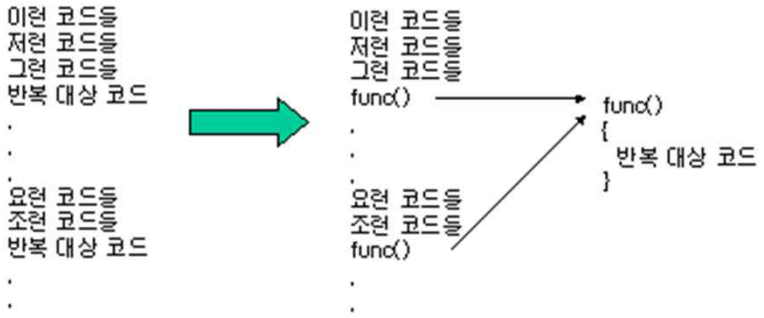
함수의 이름을 최대한 설명적으로 작성하여 이름만으로 무엇을 하는 함수인지, 이왕이면 어떻게 쓰는 것인지도 알 수 있도록 한다. 마치 함수의 이름이 주석인 것처럼 해야 한다. 함수는 이름으로 호출되므로 좋은 이름을 붙여 두면 함수를 관리하는 사람과 쓰는 사람 모두가 편해진다. 아주 간단한 규칙인 것 같지만 함수를 설계하는 첫 번째 원칙이 될 정도로 좋은 이름을 붙이는 것은 중요하다. 특히 팀 작업을 하거나 오랫동안 관리해야 할 코드라면 더욱 더 정성스럽게 이름을 붙여야 한다.

함수명은 보통 동사와 목적어 그리고 약간의 수식어로 구성된다. GetScore, DrawScreen, TestGameEnd 등은 이름만으로 어떤 동작을 하는지 쉽게 알 수 있으므로 좋은 함수명이다. 동작이 좀 더 구체적이라면 GetHighestScore, GetAverageScore 등의 수식어를 붙이는 것도 좋다. 이런 이름은 무엇을 어떻게 하는지를 분명히 표현한다.

반면 Score, Draw, Test 따위는 점수나 그리기와 상관이 있는 동작을 하는 것 같아 보이기는 하지만 구체적으로 무엇을 어떻게 하는지를 표현하지 못하므로 좋지 않다. 함수를 만든 사람은 당장은 이 함수들을 이해할 수 있다 하더라도 조금만 시간이 지나면 함수의 본체를 다 읽어 봐야 무엇을 하는 함수인지 알 수 있으므로 코드를 유지 및 확장하기 어려워진다. 함수에 좋은 이름을 붙이는 것은 어려운 기술이 아니라 조금의 관심만 기울이면 누구나 할 수 있는 기술이며 이름을 구성하는 적절한 영어 단어만 잘 선정하여 조립하면 된다.

**2** 두 번 이상 중복된 코드는 반드시 함수로 분리한다. 똑같은 코드를 중복된 채로 내버려 두면 프로그램의 크기가 쓸데없이 커진다. 10줄짜리 코드를 10번 반복한다면 나머지 90줄은 불필요하게 용량만 차지하는 것이다. 용량의 문제보다 더 심각한 것은 코드를 유지, 확장하기가 아주 곤란해진다는 점이다. 요구가 바뀌어 해당 동작을 수정해야 한다고 해 보자. 이 동작을 함수로 분리해 두었으면 함수만 고치면 되지만 중복되어 있다면 일일이 찾아가서 고쳐야 한다. 실수로 한 곳을 고치지 않으면 이것이 바로 버그의 원흉이 된다.

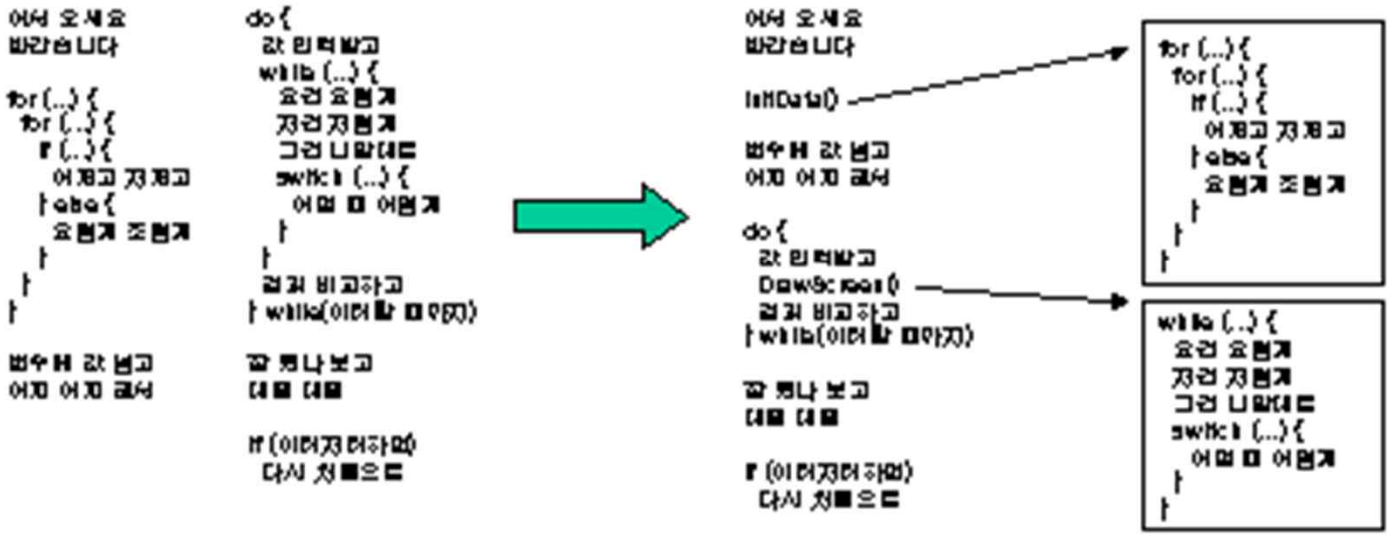
중복되는 회수에 상관없이 앞에서 이미 만들었던 코드와 비슷한 코드를 또 작성해야 한다면 일단 그 부분을 함수로 만들고 기존 코드를 함수 호출로 수정해야 한다. 즉 다음과 같이 구조를 만든다.




아니! 고작 두 번 중복되었을 뿐인데 이런 것들도 함수로 분리해야 한단 말인가 하는 생각이 들지도 모르겠다. 그 대답은 당연히 그렇다이다. 한 번 중복된 코드는 조만간 다시 필요해질 가능성이 아주 높다. 뿐만 아니라 십중팔구 그 코드는 잠시 후 확장되어야 한다. 그래서 중복이 발견되는 즉시, 그것이 단 두 군데 뿐이더라도 무조건 함수로 분리하는 습관을 가져야 한다. 네 번, 다섯 번 중복될 때 분리하겠다고 생각한다면 이미 프로그램은 엉망이 되어 가고 있는 것이다.

**3** 반복되지 않더라도 한 단위로 볼 수 있는 작업은 함수로 만든다. 설사 이 함수를 딱 한 번만 호출하고 다른 곳에서 호출할 확률이 아주 희박하더라도 이렇게 하는 것이 좋다. 예를 들어 어떤 구조체나 배열을 초기화하는 코드 덩어리는 Init~ 라는 이름으로 분리하고 화면을 출력하는 코드 덩어리는 Draw~ 따위의 이름을 주어 분리한다.

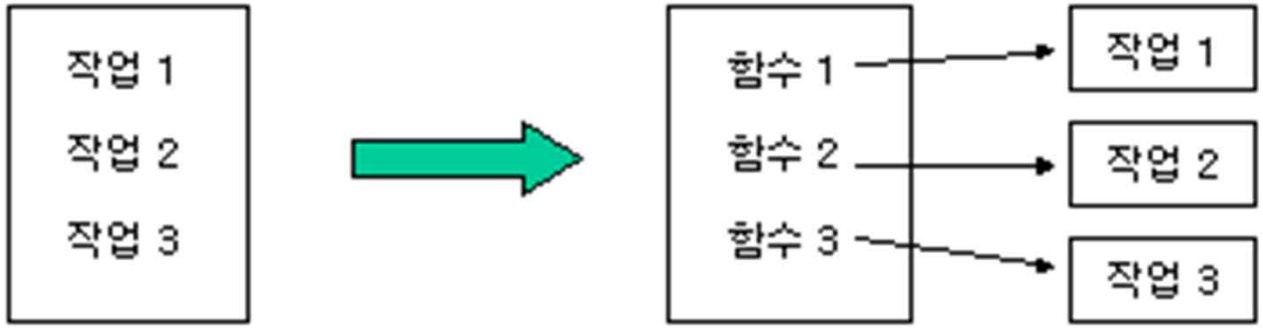
함 함수의 소스가 아주 길어져서 수백줄이 되면 그 많은 코드들의 어떤 부분이 어떤 작업을 하는지 얼른 파악되지 않는다. 게다가 다른 일을 하는 코드들이 한 곳에 섞여 있으면 필시 꼬이게 마련이며 이런 복잡한 코드는 대체로 메인 코드인 경우가 많다. 이 코드들의 그룹을 나누어 함수로 분리해 두면 메인 코드는 이 함수들을 조립하는 수준으로 간단해진다.



이렇게 분리되면 메인 코드를 읽기 쉬워지고 이미 완성된 코드들은 더 이상 신경쓰지 않아도 되는 이점이 있다. 또한 이 함수들이 현재 프로젝트에서는 반복되지 않더라도 다른 프로젝트에서는 재사용되기 쉽다.

**4**

함수는 한 번에 하나의 작업만 해야 한다. 함수는 프로그램을 구성하는 부품이며 부품이란 전체를 구성하는 원자적인 단위이다. 물론 한 함수가 두 가지 일을 동시에 수행할 수도 있고 그렇게 하는 것이 더 효율적일 때도 있다. 그러나 이 함수를 나누어 더 작은 부품을 만들어 놓으면 두 일을 각각 따로 실행해야 할 필요가 있을 때 작은 부품을 활용할 수 있다. 만약 꼭 여러 가지 일을 한꺼번에 해야 하는 함수가 필요하다면 각각의 함수를 만든 후 이 함수들을 호출하는 함수를 하나 더 만들면 된다.




이렇게 되면 이 함수를 호출하는 기존의 코드는 영향을 받지 않으면서 새로운 작은 단위의 작업을 호출할 수도 있게 된다.

5 입력과 출력이 직관적이고 명확해야 한다. 인수는 함수에게 주어지는 작업거리인데 함수가 하는 일에 꼭 필요한 정보만 최소한의 인수로 받아들여야 한다. 나머지 인수로부터 알 수 있는 값이나 연산에 사용하지 않는 불필요한 정보는 전달될 필요가 없다. 예를 들어 화면의 특정 위치에 메시지를 출력하는 OutMessage 함수를 작성한다고 해 보자.

```
void OutMessage(int x, int y, char *str, int len)
```

이 함수에서 메시지의 길이 len은 불필요한 인수이다. 세 번째 인수 str이 널 종료 문자열이라면 str로부터 길이를 계산할 수 있다. 물론 메시지의 일부만을 출력하는 기능이 있다면 이럴 때는 len이 필요할 것이다. 함수의 작업 결과는 가급적이면 리턴값으로 보고해야 하는데 설사 그 값을 호출원에서 사용하지 않는다 하더라도 일단 보고할 내용이 있다면 보고하는 것이 좋다.

6 함수는 자체적으로 에러 처리를 해야 한다. 함수는 독립된 작업을 하며 재사용 가능한 부품이므로 그 자체로서 완벽하게 동작할 수 있어야 한다. 현재 작성중인 프로젝트에서 잘 실행된다고 해서 이 함수가 안전하고 완벽하게 동작한다는 것을 보장할 수는 없다. 특히 입력된 인수의 유효성을 잘 점검해야 한다. 포인터의 경우 NULL이 전달될 수도 있고 정수가 터무니없이 크다거나 음수가 될 수 없는 값에 대해 음수가 전달되는 경우에도 에러 처리를 해야 한다. 그래야 어떤 프로젝트로 가져 가든 별도의 수정없이 재사용 가능한 부품이 된다.

여기서 논한 함수 설계에 대한 지침은 어디까지나 일반적인 참고사항일 뿐이다. 특수한 실무 환경에서는 이 지침과는 다르게 함수를 만들어야 하는 불가피한 경우도 존재한다. 이 지침이 함수 설계를 잘 하고 싶은 사람들에게 아주 중요한 내용인 것은 사실이지만 이 지침들을 몽땅 외운다고 해서 함수 설계 경험이 금방 늘어나는 것은 아니다. 마치 "여성의 마음을 사로잡는 100가지 비법"을 달달 외운다고 해서 사교계의 달인이 될 수 없는 것과 같다. 일단 이 권고안을 머리속에 새겨 두고 끊임없이 분석, 연습해 봐야 한다.

# 프로젝트 멘토풀 아이디어

- **오프라인**

1. **교수, 연구생 등 참여**

- **온라인**

1. **건국서울 어코드 사업 싸이트를 통한 홍보>학생 업로드>졸업생 및 지인 안내를 통한 진행**
2. **한이음 싸이트 > 교수> 프로젝트 등록 > 신청 멘토를 통해 진행**


http://www.cyworld.com/konkuk_seoul_accord


http://www.hanium.or.kr/portal/hanium/business.do?vp=0

# 과목설문