프로젝트관리 개념

한국항공대학교 소프트웨어학과 지승도교수
R.S. Pressman

네 개의 P

- People (사람) — 성공적인 프로젝트를 위한 핵심 자원
- Product (산출물) — 만들어야 할 소프트웨어 결과물
- Process (프로세스) — 작업완수에 필요한 일련의 프레임워크 활동들과 태스크들
- Project (프로젝트) — 소프트웨어 구현을 위해 필요한 모든 작업들

People
- 이해당사자들

- *Senior managers (고위급 관리자)* who define the business issues that often have significant influence on the project.
- *Project (technical) managers (프로젝트 관리자)* who must plan, motivate, organize, and control the practitioners who do software work.
- *Practitioners (실무자)* who deliver the technical skills that are necessary to engineer a product or application.
- *Customers (고객)* who specify the requirements for the software to be engineered and other stakeholders who have a peripheral interest in the outcome.
- *End-users (사용자)* who interact with the software once it is released for production use.

People: 사례

소프트웨어 팀

어떻게 이끌어 나갈까?

어떻게 협력할까?

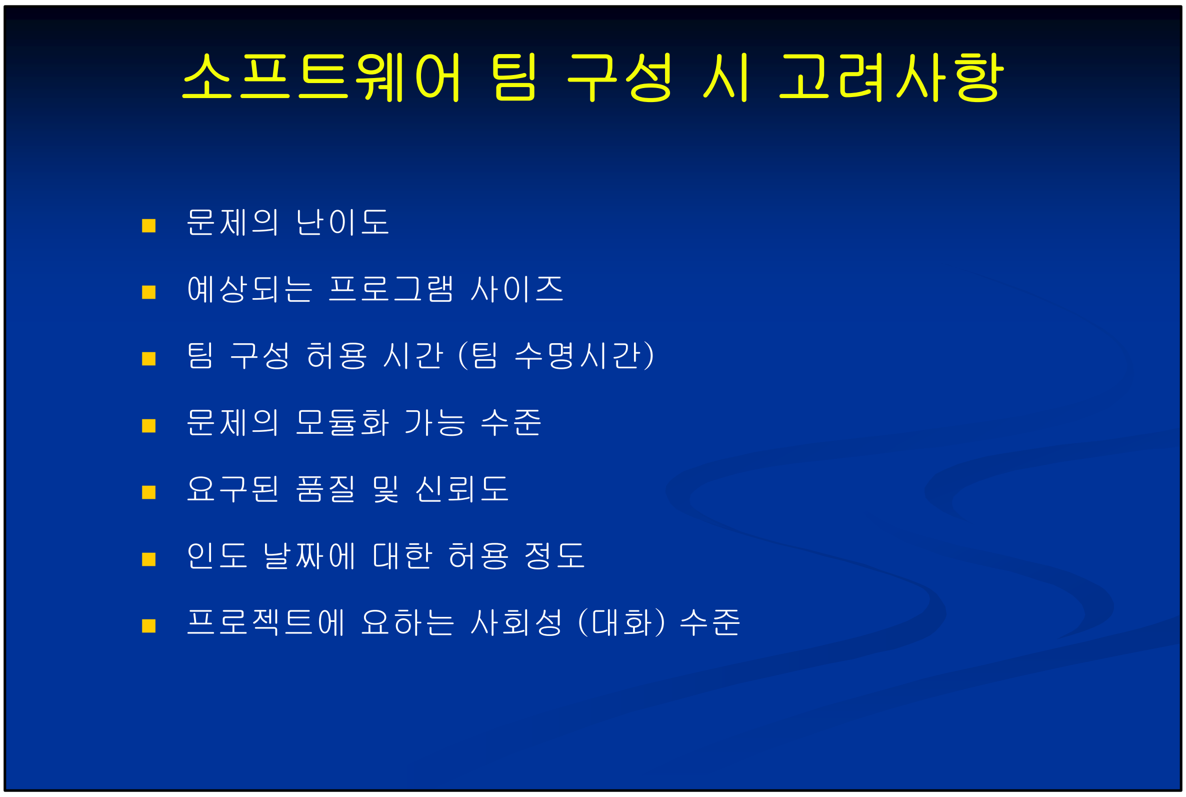
어떻게 조직할까?

어떻게 동기부여를 시킬까?

어떻게 창조적인 아이디어를 얻을 수 있을까?

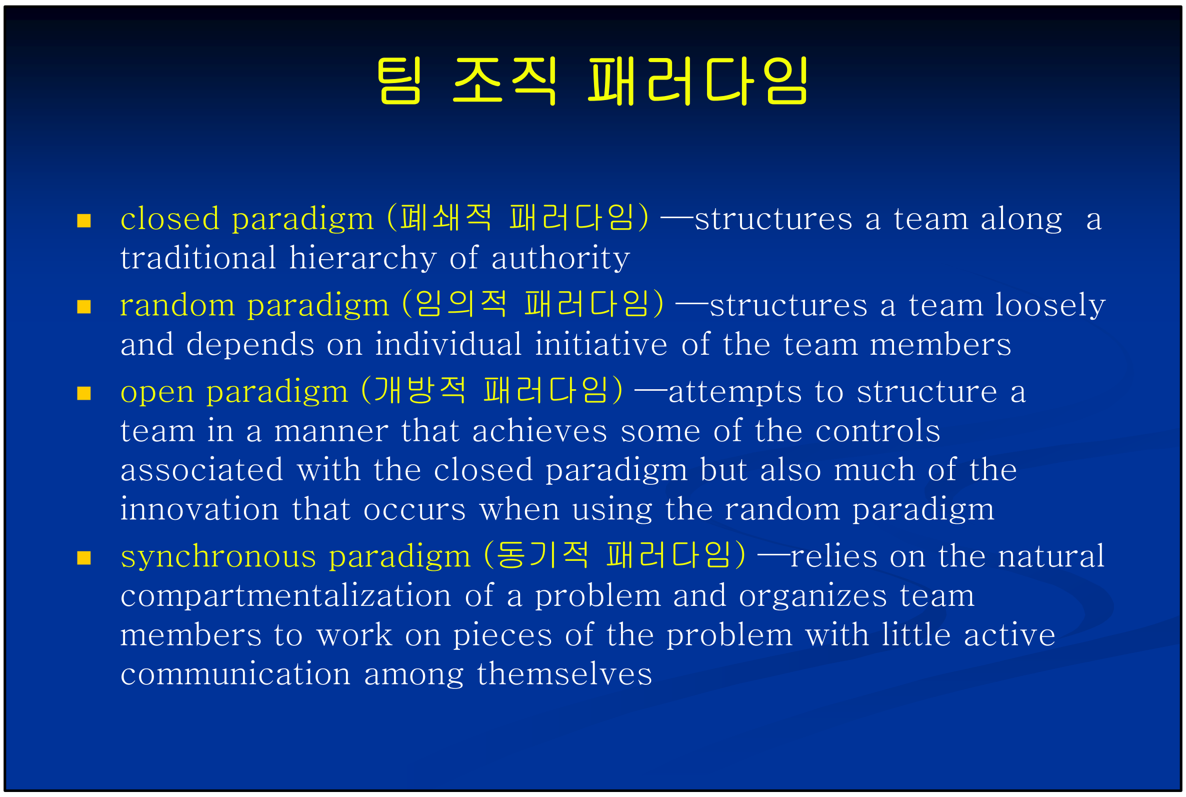
팀 리더

- MOI 모델
  - Motivation (동기부여) The ability to encourage (by "push or pull") technical people to produce to their best ability.
  - Organization (조직) The ability to mold existing processes (or invent new ones) that will enable the initial concept to be translated into a final product.
  - Ideas or innovation (혁신) The ability to encourage people to create and feel creative even when they must work within bounds established for a particular software product or application.



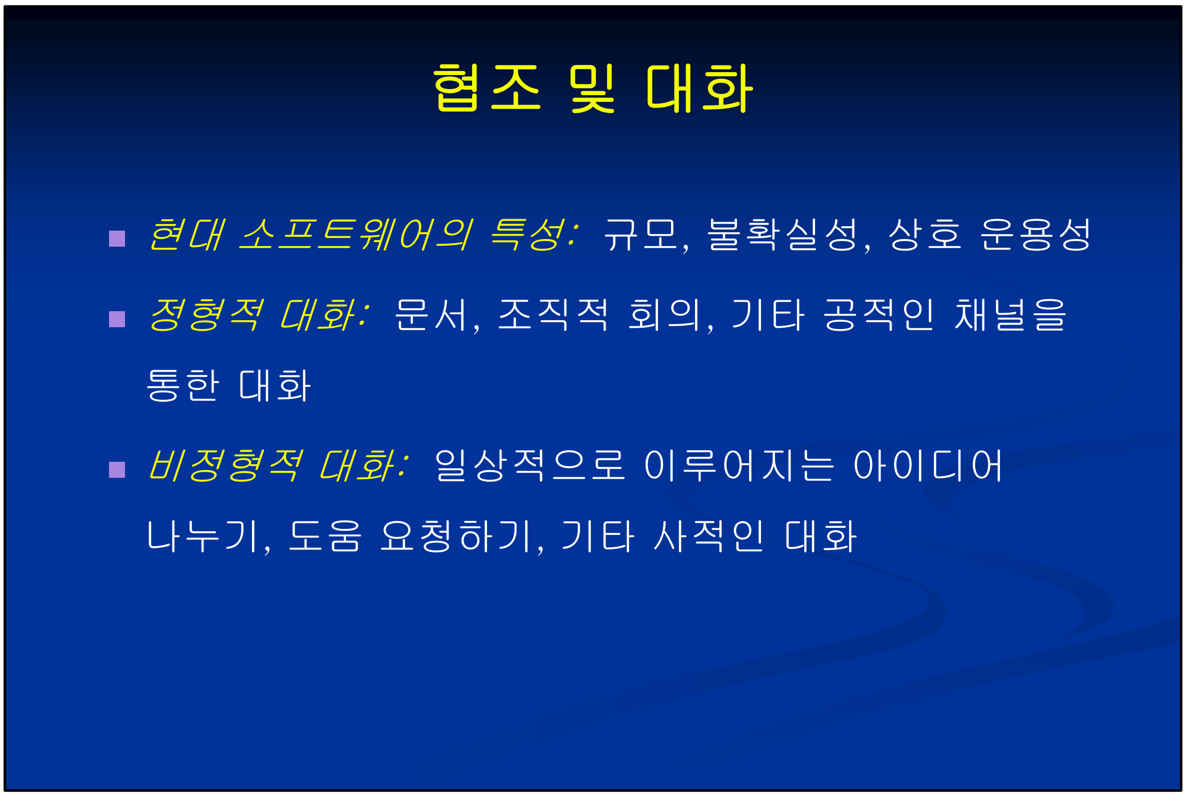
소프트웨어 팀 구성 시 고려사항

- 문제의 난이도
- 예상되는 프로그램 사이즈
- 팀 구성 허용 시간 (팀 수명시간)
- 문제의 모듈화 가능 수준
- 요구된 품질 및 신뢰도
- 인도 날짜에 대한 허용 정도
- 프로젝트에 요하는 사회성 (대화) 수준



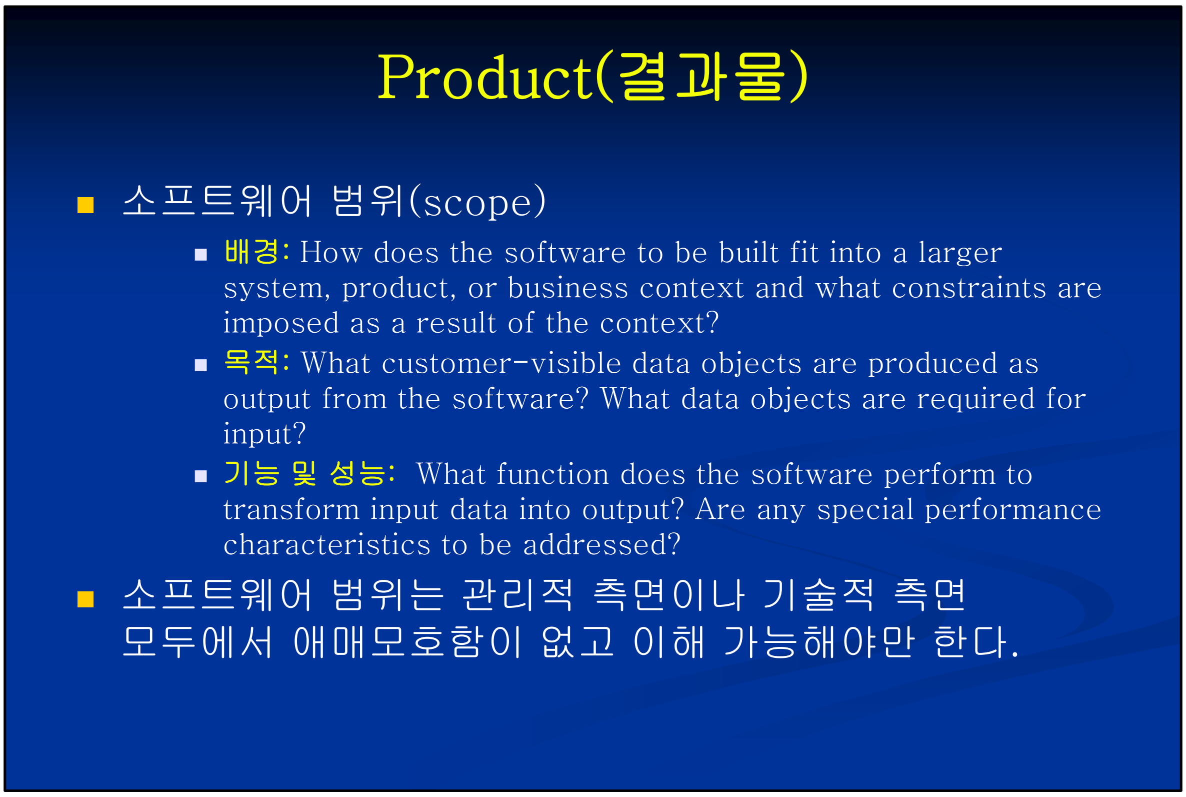
팀 조직 패러다임

- closed paradigm (폐쇄적 패러다임) —structures a team along a traditional hierarchy of authority
- random paradigm (임의적 패러다임) —structures a team loosely and depends on individual initiative of the team members
- open paradigm (개방적 패러다임) —attempts to structure a team in a manner that achieves some of the controls associated with the closed paradigm but also much of the innovation that occurs when using the random paradigm
- synchronous paradigm (동기적 패러다임) —relies on the natural compartmentalization of a problem and organizes team members to work on pieces of the problem with little active communication among themselves



협조 및 대화

- *현대 소프트웨어의 특성:* 규모, 불확실성, 상호 운용성
- *정형적 대화:* 문서, 조직적 회의, 기타 공적인 채널을 통한 대화
- *비정형적 대화:* 일상적으로 이루어지는 아이디어 나누기, 도움 요청하기, 기타 사적인 대화



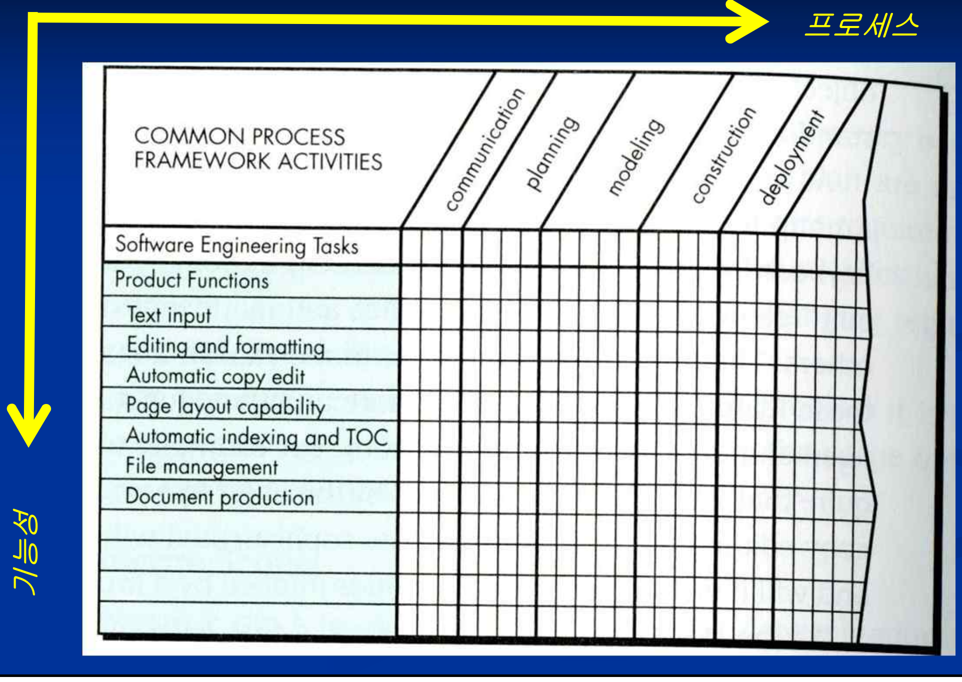
Product(결과물)

- 소프트웨어 범위(scope)
  - 배경: How does the software to be built fit into a larger system, product, or business context and what constraints are imposed as a result of the context?
  - 목적: What customer-visible data objects are produced as output from the software? What data objects are required for input?
  - 기능 및 성능: What function does the software perform to transform input data into output? Are any special performance characteristics to be addressed?
- 소프트웨어 범위는 관리적 측면이나 기술적 측면 모두에서 애매모호함이 없고 이해 가능해야만 한다.

문제 분할

- *구획화(partitioning)* / *정교화(problem elaboration)*
- 분할 및 정복 전략 (Divide-and-conquer)
- 분할: *기능성(Functionality)* 및 *프로세스(Process)*

프로세스 (Process)



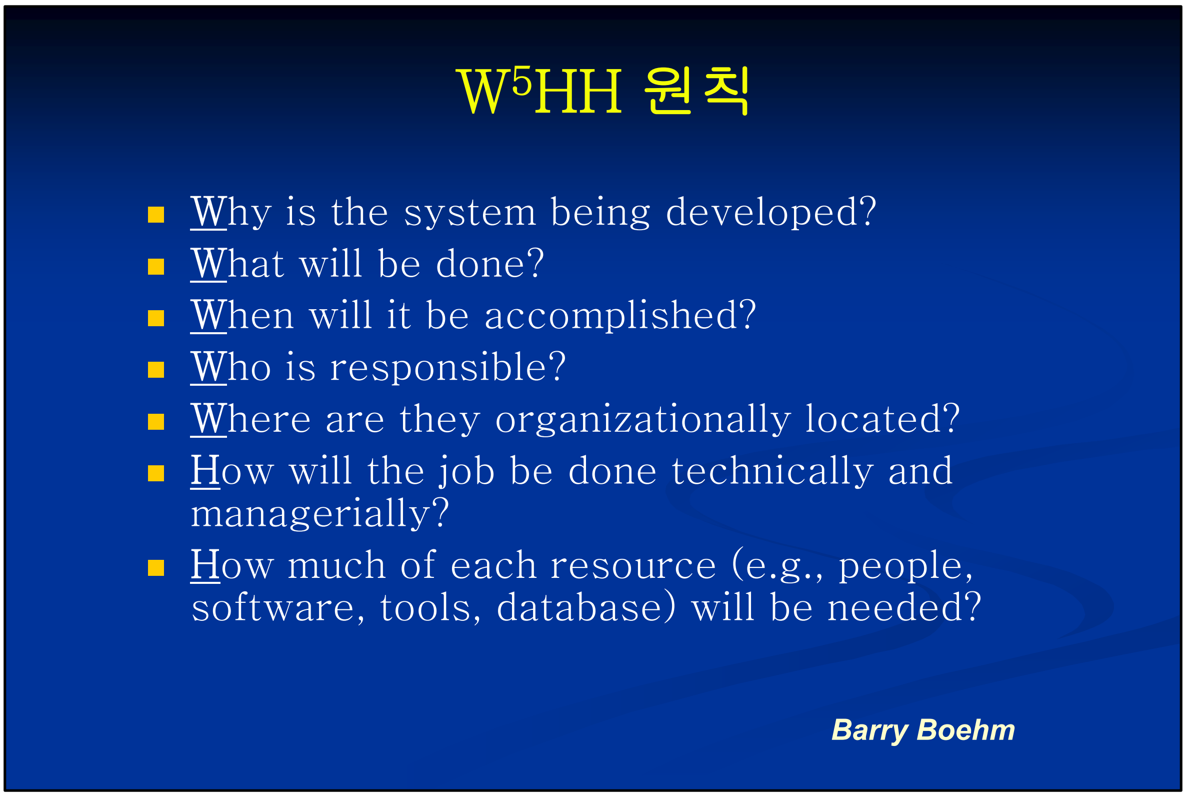
COMMON PROCESS FRAMEWORK ACTIVITIES	communication	planning	modeling	construction	deployment
Software Engineering Tasks					
Product Functions					
Text input					
Editing and formatting					
Automatic copy edit					
Page layout capability					
Automatic indexing and TOC					
File management					
Document production					

프로젝트 (Project)

- Projects get into trouble when …
  - Software people don't understand their customer's needs.
  - The product scope is poorly defined.
  - Changes are managed poorly.
  - The chosen technology changes.
  - Business needs change [or are ill-defined].
  - Deadlines are unrealistic.
  - Users are resistant.
  - Sponsorship is lost [or was never properly obtained].
  - The project team lacks people with appropriate skills.
  - Managers [and practitioners] avoid best practices and lessons learned.

프로젝트: 기본 요령

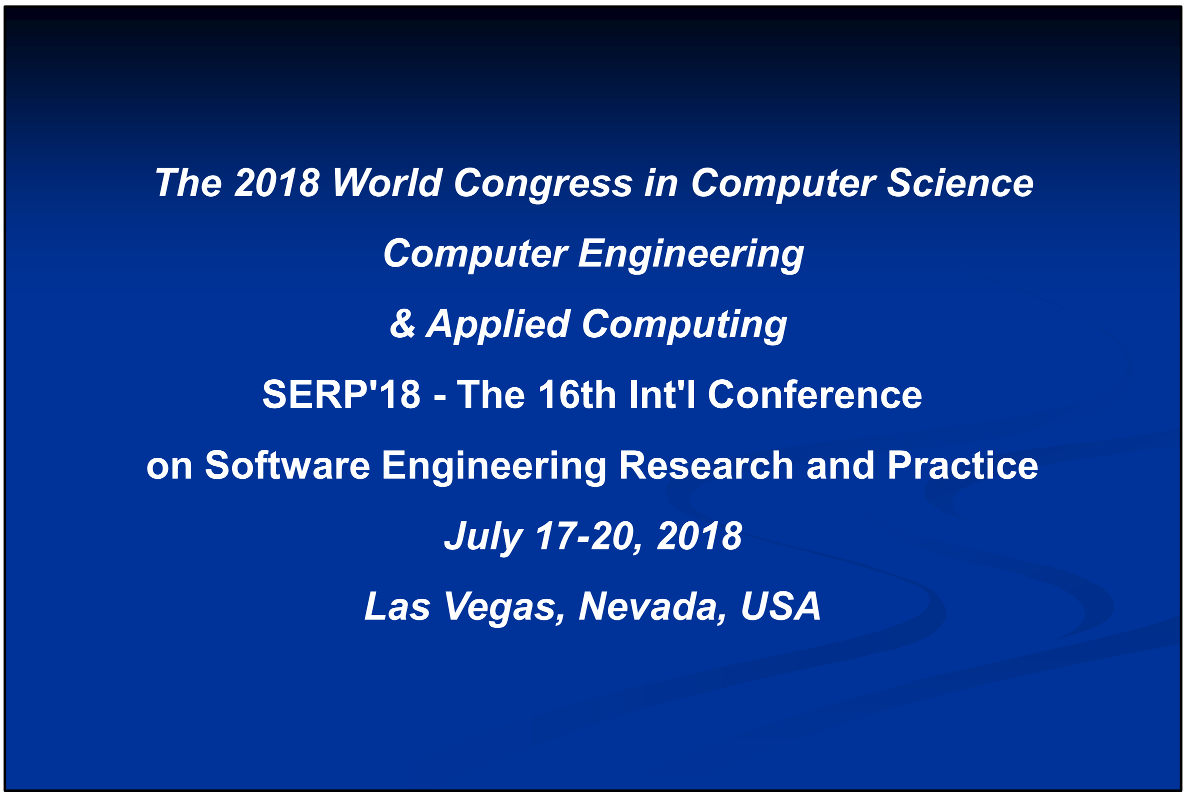
- *Start on the right foot (첫걸음부터 제대로).* This is accomplished by working hard (very hard) to understand the problem that is to be solved and then setting realistic objectives and expectations.
- *Maintain momentum (탄력 유지). The* project manager must provide incentives to keep turnover of personnel to an absolute minimum, the team should emphasize quality in every task it performs, and senior management should do everything possible to stay out of the team's way.
- *Track progress (진도 확인).* For a software project, progress is tracked as work products (e.g., models, source code, sets of test cases) are produced and approved (using formal technical reviews) as part of a quality assurance activity.
- *Make smart decisions (현명한 의사결정).* In essence, the decisions of the project manager and the software team should be to "keep it simple."
- *Conduct a postmortem analysis (사후 분석 실시).* Establish a consistent mechanism for extracting lessons learned for each project.



W^{5}HH 원칙

- Why is the system being developed?
- What will be done?
- When will it be accomplished?
- Who is responsible?
- Where are they organizationally located?
- How will the job be done technically and managerially?
- How much of each resource (e.g., people, software, tools, database) will be needed?

Barry Boehm



The 2018 World Congress in Computer Science

Computer Engineering

& Applied Computing

SERP'18 - The 16th Int'l Conference

on Software Engineering Research and Practice

July 17-20, 2018

Las Vegas, Nevada, USA

- Software Architectures
- Software Design and Design Patterns
- Quality Oriented Software Architecture (Design and support)
- Software Reliability, Safety and Security Methods
- Software Reuse and Component-Based Software Engineering
- UML/MDA and AADL
- Agile Software Engineering and Development
- Object Oriented Technology (Design and Analysis)
- Software Metrics
- Software Assurance and Dependability
- Aerospace Software and System Engineering
- Software Testing, Evaluation and Analysis Technologies
- Project Management Issues
- Distributed and Parallel Systems
- Real-time Embedded Software Engineering
- Theoretic Approaches (Formal Methods, Graph, ...)
- Domain Modeling and Meta-modeling
- Software Maintenance and Evolution
- Component Based Software Engineering
- Software Engineering Standards and Guidelines
- Intelligent CASE Tools and Eclipse Plugins Issues
- Requirement Engineering and Processes
- Human Computer Interaction and Usability Engineering
- Aspect Oriented Software Engineering
- Agent Architectures, Ontologies, Languages and Protocols
- Multi-Agent Systems, Mobile Agents
- Artificial Intelligence Approaches to Software Engineering

미래의 공학을 생각하라!

새벽녘 등불이 지워진 화장을 비추는데

이별을 말하고자 하니 애간장이 끊어지네.

지는 달빛 빈 뜰에 문을 밀고 나서니

살구꽃 성긴 그림자 도포자락에 가득하네.

융합이란 무엇인가?

기술적 시각

+

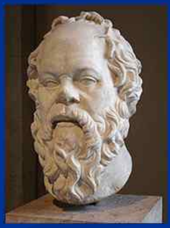
인문적 시각

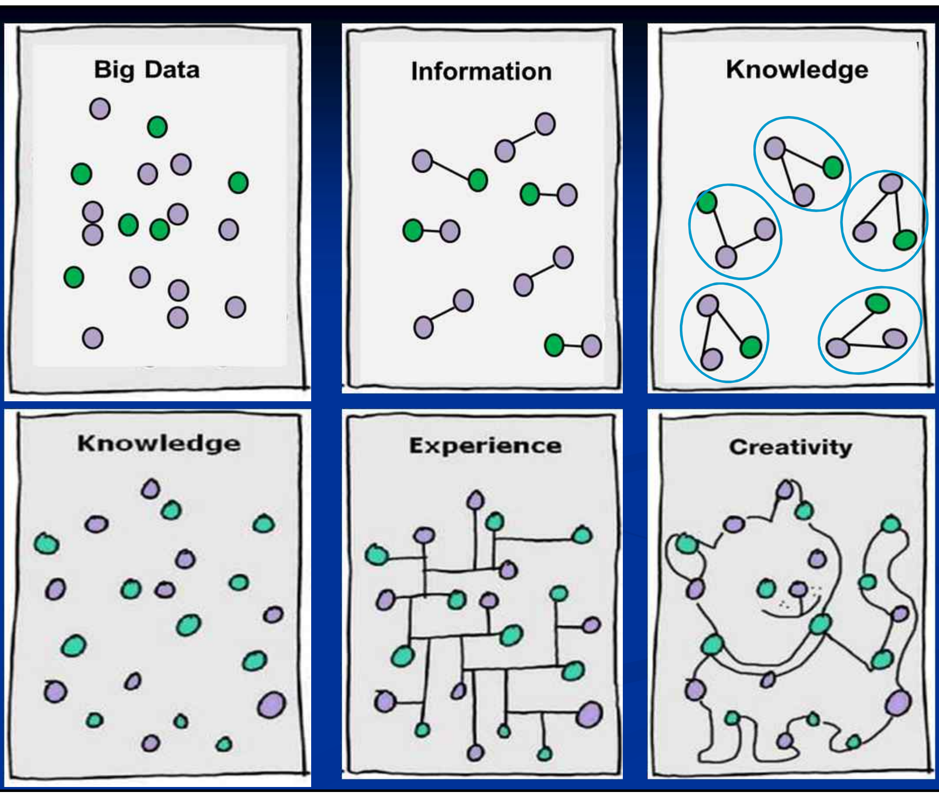
+

경제적 시각

+

철학적 시각





Homework #1: 주제선정

1. Scope

(1) 배경

(2) 목적

(3) 기능 및 성능

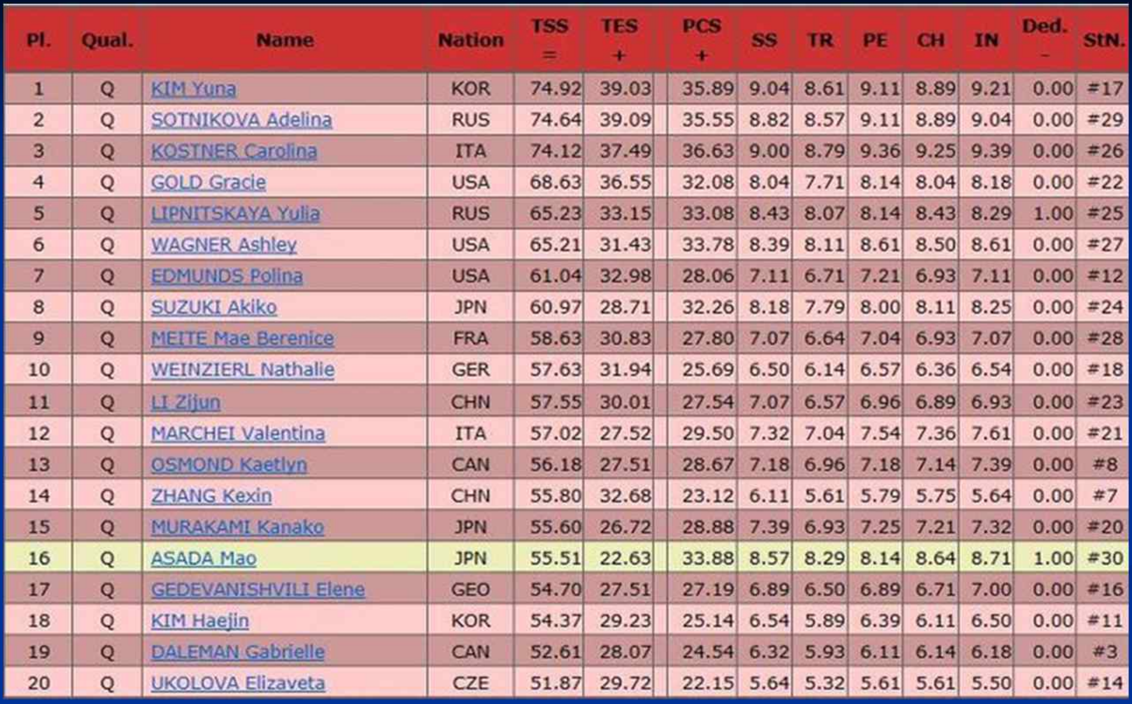
➔ 제출기간: 2주

➔ 과목명, 과제 번호 및 제목, 제출날짜, 학번, 이름

➔ 필수: 결론 또는 느낀 점

프로젝트 측정 (Measure)

파리스의 심판

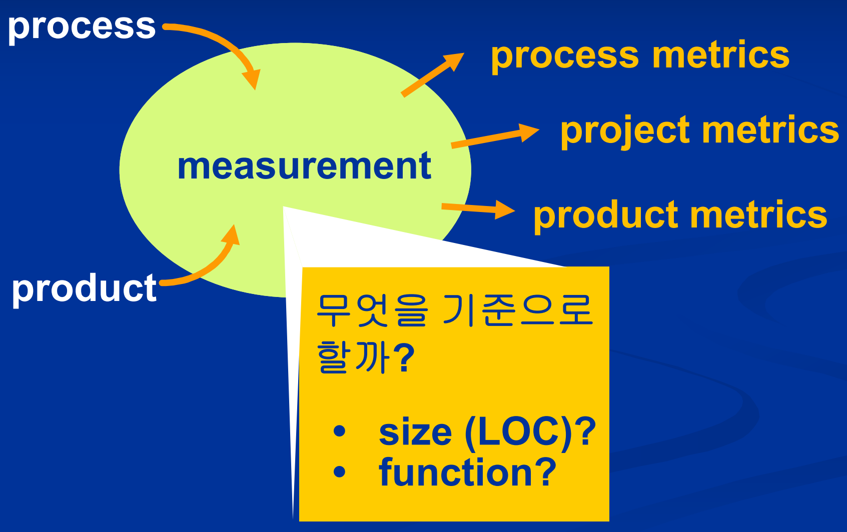


Pl.	Qual.	Name	Nation	TSS =	TES +	PCS +	SS	TR	PE	CH	IN	Ded. -	StN.
1	Q	KIM Yuna	KOR	74.92	39.03	35.89	9.04	8.61	9.11	8.89	9.21	0.00	#17
2	Q	SOTNIKOVA Adelina	RUS	74.64	39.09	35.55	8.82	8.57	9.11	8.89	9.04	0.00	#29
3	Q	KOSTNER Carolina	ITA	74.12	37.49	36.63	9.00	8.79	9.36	9.25	9.39	0.00	#26
4	Q	GOLD Gracie	USA	68.63	36.55	32.08	8.04	7.71	8.14	8.04	8.18	0.00	#22
5	Q	LIPNITSKAYA Yulia	RUS	65.23	33.15	33.08	8.43	8.07	8.14	8.43	8.29	1.00	#25
6	Q	WAGNER Ashley	USA	65.21	31.43	33.78	8.39	8.11	8.61	8.50	8.61	0.00	#27
7	Q	EDMUNDS Polina	USA	61.04	32.98	28.06	7.11	6.71	7.21	6.93	7.11	0.00	#12
8	Q	SUZUKI Akiko	JPN	60.97	28.71	32.26	8.18	7.79	8.00	8.11	8.25	0.00	#24
9	Q	MEITE Mae Berenice	FRA	58.63	30.83	27.80	7.07	6.64	7.04	6.93	7.07	0.00	#28
10	Q	WEINZIERL Nathalie	GER	57.63	31.94	25.69	6.50	6.14	6.57	6.36	6.54	0.00	#18
11	Q	LI Zijun	CHN	57.55	30.01	27.54	7.07	6.57	6.96	6.89	6.93	0.00	#23
12	Q	MARCHEI Valentina	ITA	57.02	27.52	29.50	7.32	7.04	7.54	7.36	7.61	0.00	#21
13	Q	OSMOND Kaetlyn	CAN	56.18	27.51	28.67	7.18	6.96	7.18	7.14	7.39	0.00	#8
14	Q	ZHANG Kexin	CHN	55.80	32.68	23.12	6.11	5.61	5.79	5.75	5.64	0.00	#7
15	Q	MURAKAMI Kanako	JPN	55.60	26.72	28.88	7.39	6.93	7.25	7.21	7.32	0.00	#20
16	Q	ASADA Mao	JPN	55.51	22.63	33.88	8.57	8.29	8.14	8.64	8.71	1.00	#30
17	Q	GEDEVANISHVILI Elene	GEO	54.70	27.51	27.19	6.89	6.50	6.89	6.71	7.00	0.00	#16
18	Q	KIM Haejin	KOR	54.37	29.23	25.14	6.54	5.89	6.39	6.11	6.50	0.00	#11
19	Q	DALEMAN Gabrielle	CAN	52.61	28.07	24.54	6.32	5.93	6.11	6.14	6.18	0.00	#3
20	Q	UKOLOVA Elizaveta	CZE	51.87	29.72	22.15	5.64	5.32	5.61	5.61	5.50	0.00	#14

왜 측정이 필요한가?

(1) To characterize
(2) To evaluate
(3) To predict
(4) To improve

무엇을 측정하나?



무엇을 기준으로 할까?

- size (LOC)?
- function?

측정 이유

- assess the status of an ongoing project
- track potential risks
- uncover problem areas before they go "critical"
- adjust work flow or tasks
- evaluate the project team's ability to control quality of software work products.

소프트웨어 측정

- 직접 측정

➔ LOC (line of code), speed, memory size, # of defect, etc.

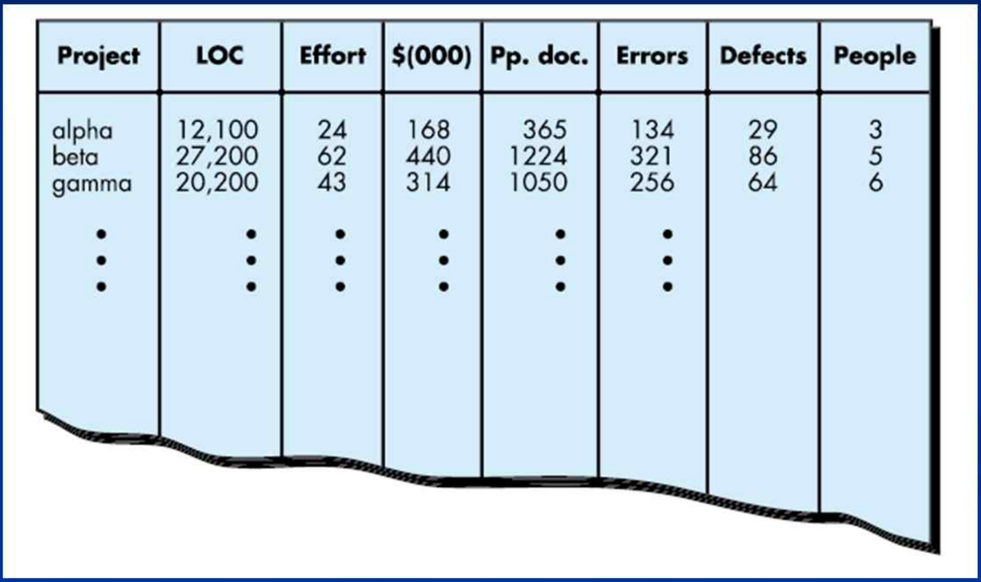
- 간접 측정

➔ functionality, quality, complexity, efficiency, reliability, maintainability, etc.

크기 중심 측정지표 (Size-Oriented Metrics)

- Errors per KLOC (thousand lines of code)
- Defects per KLOC
- $ per KLOC
- Pages of documentation per KLOC
- Errors per person-month
- KLOC per person-month
- $ per page of documentation

Size-Oriented Metrics: Example



Project	LOC	Effort	$(000)	Pp. doc.	Errors	Defects	People
alpha	12,100	24	168	365	134	29	3
beta	27,200	62	440	1224	321	86	5
gamma	20,200	43	314	1050	256	64	6
⋮	⋮	⋮	⋮	⋮	⋮		

기능 중심 측정 지표 (Function-Oriented Metrics)

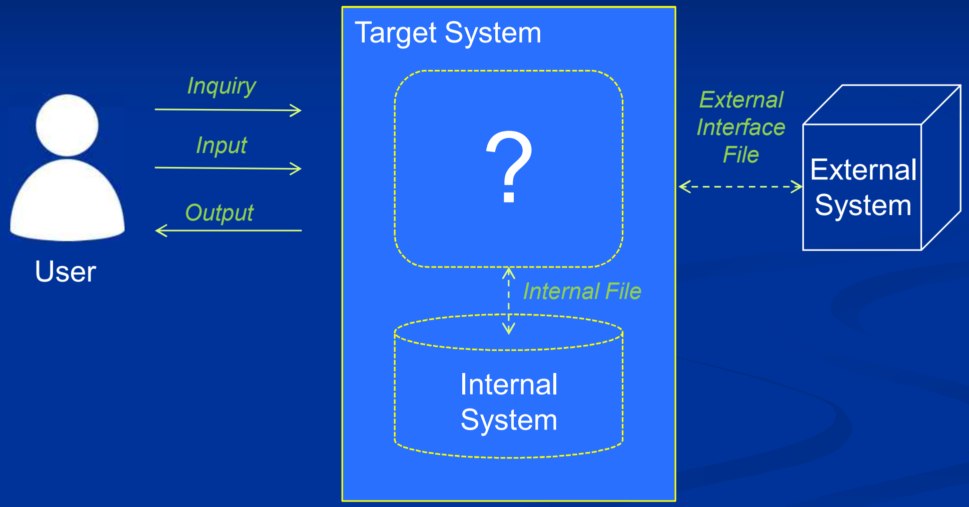
- errors per FP
- defects per FP
- $ per FP
- pages of documentation per FP
- FP per person-month

남자가 여자의 점수를 계산하는 방식

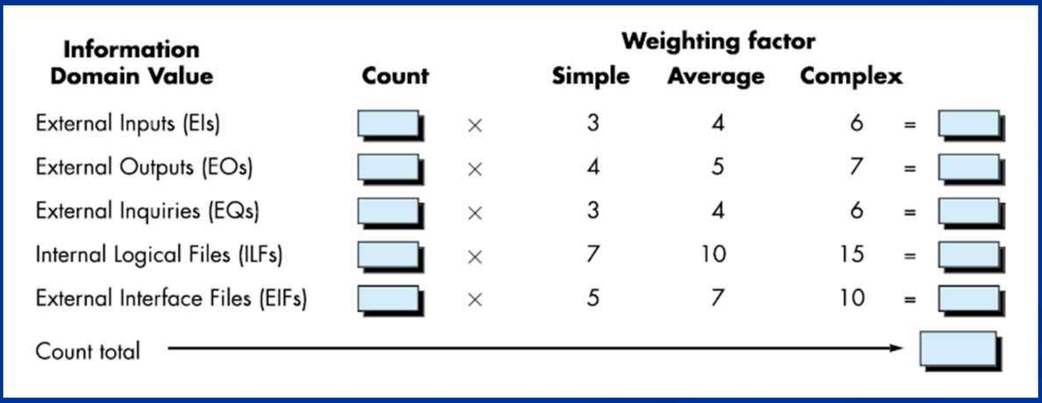
여자의 점수 = (성격 + 개념 + 직업 + 나이 + 집안 + 종교 + …) * 외모

여자가 남자의 점수를 계산하는 방식

남자의 점수 = (키 * 가중치) + (성격 * 가중치) + (직업 * 가중치)
+ (외모 * 가중치) + (집안 * 가중치) + …



기능점수 측정 평가 (Function Point Metrics)



Information Domain Value	Count		Weighting factor Simple	Average	Complex		
External Inputs (EIs)		×	3	4	6	=	
External Outputs (EOs)		×	4	5	7	=	
External Inquiries (EQs)		×	3	4	6	=	
Internal Logical Files (ILFs)		×	7	10	15	=	
External Interface Files (EIFs)		×	5	7	10	=	
Count total							

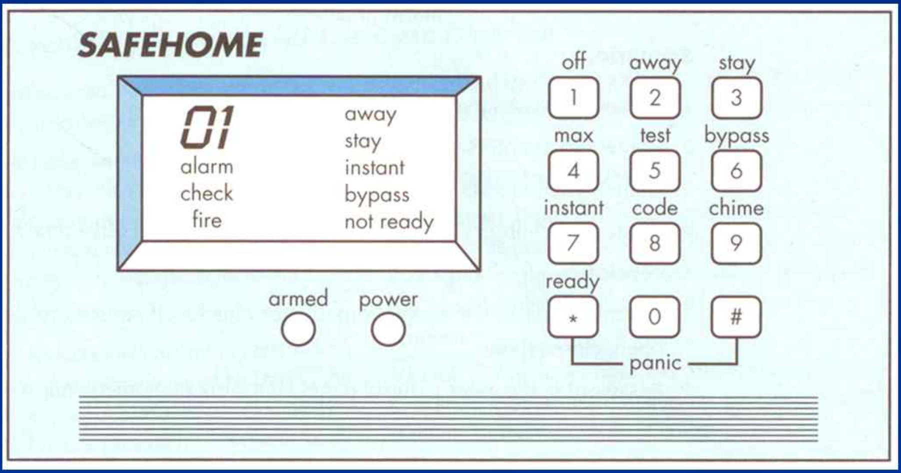
기능점수 측정 평가 (Function Point Metrics)

- Number of external inputs (EIs)
- Number of external outputs (EOs)
- Number of external inquires (EQs)
- Number of internal logical files (ILFs)
- Number of external interface files (EIFs)

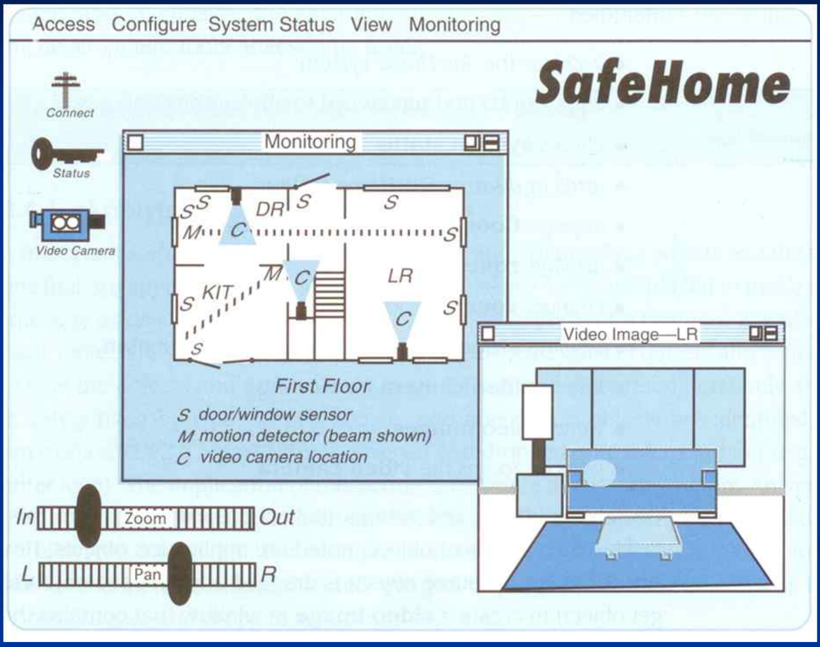
➢ Fi = 1 to 14 (0~5 each)

FP = count-total * (0.65 +0.01 * Σ(Fi))

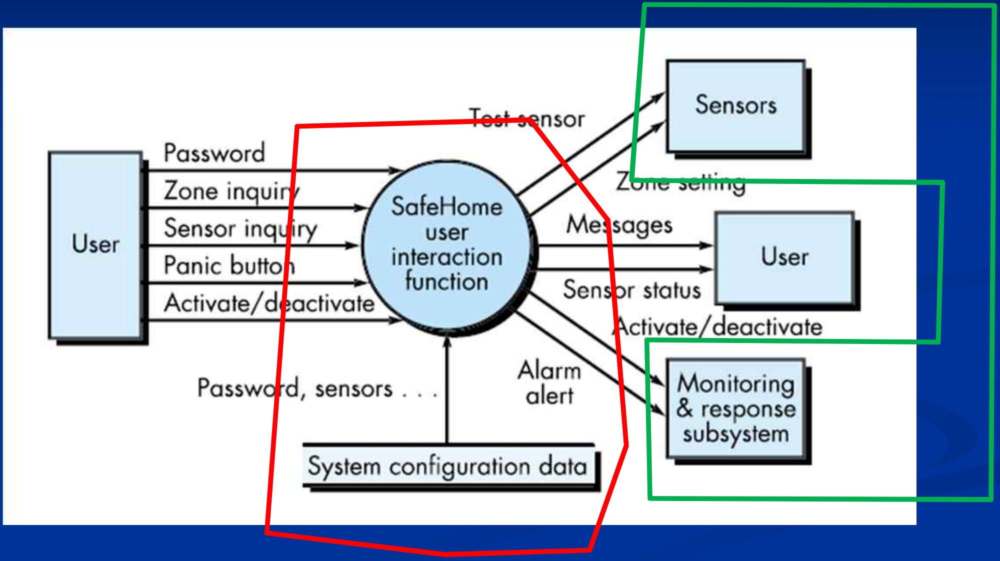
SafeHome 예제



SafeHome Example



Function-Oriented Metrics: 예제



Function-Oriented Metrics: 예제

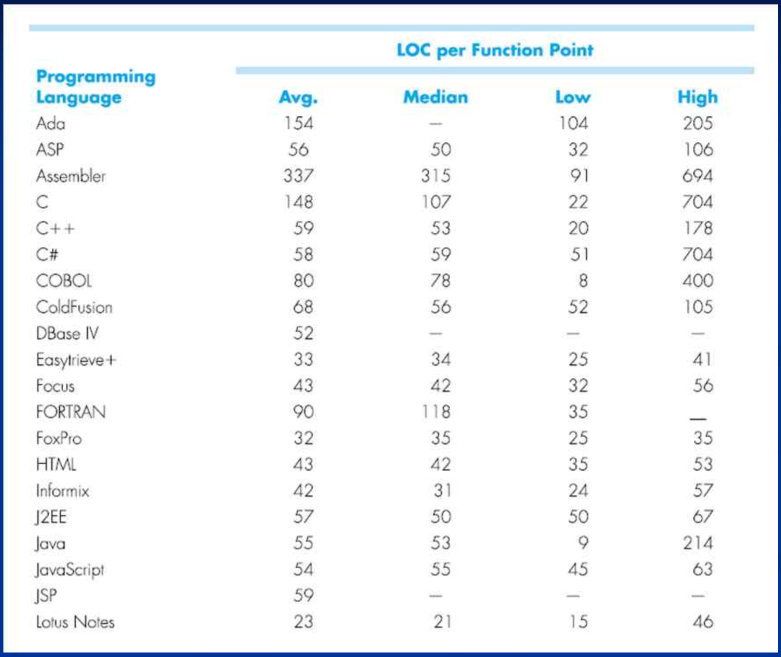
Information Domain Value	Count		Weighting factor Simple	Average	Complex		
External Inputs (EIs)	3	×	[✓ 3]	4	6	=	9
External Outputs (EOs)	2	×	[✓ 4]	5	7	=	8
External Inquiries (EQs)	2	×	[✓ 3]	4	6	=	6
Internal Logical Files (ILFs)	1	×	[✓ 7]	10	15	=	7
External Interface Files (EIFs)	4	×	[✓ 5]	7	10	=	20
Count total						→	50

Function-Oriented Metrics: 예제

1. Does the system require reliable backup and recovery? 3
2. Are specialized data communications required to transfer information to or from the application? 3
3. Are there distributed processing functions? 3
4. Is performance critical? 3
5. Will the system run in an existing, heavily utilized operational environment? 3
6. Does the system require online data entry? 3
7. Does the online data entry require the input transaction to be built over multiple screens or operations? 3
8. Are the ILFs updated online? 3
9. Are the inputs, outputs, files, or inquiries complex? 3
10. Is the internal processing complex? 3
11. Is the code designed to be reusable? 3
12. Are conversion and installation included in the design? 3
13. Is the system designed for multiple installations in different organizations? 3
14. Is the application designed to facilitate change and ease of use by the user? 3

FP = count-total × (0.65 + 0.01 × Σ(Fi)) = 50 × (0.65 + 0.01 × (14 × 3)) = 51

LOC 및 FP 비교표



Programming Language	LOC per Function Point Avg.	Median	Low	High
Ada	154	–	104	205
ASP	56	50	32	106
Assembler	337	315	91	694
C	148	107	22	704
C++	59	53	20	178
C#	58	59	51	704
COBOL	80	78	8	400
ColdFusion	68	56	52	105
DBase IV	52	–	–	–
Easytrieve+	33	34	25	41
Focus	43	42	32	56
FORTRAN	90	118	35	—
FoxPro	32	35	25	35
HTML	43	42	35	53
Informix	42	31	24	57
J2EE	57	50	50	67
Java	55	53	9	214
JavaScript	54	55	45	63
JSP	59	–	–	–
Lotus Notes	23	21	15	46

품질(Quality) 측정

- Correctness — the degree to which a program operates according to specification (after delivery)
  ➔ #_defects / KLOC
- Maintainability — the degree to which a program is amenable to change
  ➔ MTTC (mean-time-to-change)

품질(Quality) 측정

- Integrity — degree to which a program is impervious to outside attack
  ➔ Integrity = Σ (1 – (threat x (1 – security)))

- Usability — degree to which a program is easy to use
  ➔ quantify easy of use

Defect Removal Efficiency

$$DRE = E / (E + D)$$

E is the number of errors found before delivery of the software to the end-user

D is the number of defects found after delivery.