2017 한국컴퓨터종합학술대회

소프트웨어공학 역사 워크샵 발표자료집

Proceedings of the Software Engineering History Workshop

일시: 2017년 6월 20일(화) ~ 6월 21일(수)

장소: 제주 라마다 호텔 탐라홀 8F, Room J

주최: 한국정보과학회

주관: 한국정보과학회 소프트웨어공학 소사이어티

한국정보과학회
KOREAN INSTITUTE OF INFORMATION SCIENTISTS AND ENGINEERS

SOFTWARE ENGINEERING SOCIETY

개회사

역사는 "안개 속의 미래를 비추어주는 횃불"이라고 사람들은 말합니다.

많은 것들이 빠르게 바뀌어 가는 21세기의 현 시점에, 누구나 미래를 밝혀, 더 잘 보고 싶을 가질 것입니다.

우리 소프트웨어공학인들도 마찬가지로 미래를 더 선명히 보고, 올바른 방향으로 나가기를 희망할 것입니다

오늘 소프트웨어공학 역사 워크샵을 개최하게 된 것은 올해가 소프트웨어 소사이어티가 생긴지 30주년 되는 해이기 때문만은 아닙니다. 이 시점에 과거를 회고해 봄으로써 소프트웨어공학인들과 소프트웨어공학 소사이어티가 어떤 길로 가야 할지를 더 잘 알고자 하는 이유도 있습니다.

그 길에는 기술적인 측면도 있고, 대외적인 측면, 사회적인 측면, 문화적인 측면도 있습니다. 오늘 이 워크샵에서 이 모든 측면이 다 다루어질 수는 없습니다.

그러나 이 워크샵은 우리들로 하여금 여러가지 측면에서 우리나라 소프트웨어공학의 미래에 대해서 진지하게 생각하게 하고, 미래를 위한 판단을 위해 과거를 더 깊이 들여다 보는 계기가 될 것입니다.

바쁘신 가운데에도 오늘 발표를 준비해 주신 모든 발표자께 깊이 감사 드립니다. 오늘 워크샵의 발표자 한 분 한 분은 우리나라 소프트웨어 공학 역사의 주인공인 동시에 증인입니다. 그 때문에 이 워크샵은 매우 중요하고 뜻 깊은 자리입니다.

또한 이 워크샵에 관심을 가지고 참석해주신 참석자 여러분께도 감사 드립니다. "현명한 자는 (경험이 아니라) 역사에서 배운다"고 합니다. 역사에서 배우기 위해 오신 참석자 여러분의 현명함에 경의를 표합니다.

끝으로 항상 그랬던 것처럼 오늘의 워크샵을 준비하기 위해 노고를 아끼지 않은 소프트웨어공학 소사이어티 운영위원님들께도 깊이 감사 드립니다.

한국정보과학회 소프트웨어공학소사이어티 회장 강성원

워크샵 준비 위원회

대회장:

강성원 교수(KAIST)

프로그램위원장:

이병정 교수(서울시립대)

고인영 교수(KAIST)

학술위원장:

이관우 교수(한성대)

조직위원장:

이정원 교수(아주대)

조직위원:

이선아 교수(경상대)

이찬근 교수(중앙대)

워크샵 프로그램

날짜	시간	주제	연사	장소
6/20 (화)	11:00 ~12:00	운영위원회의	운영위원 및 연사	라마다호텔 2층 더블루
	12:00~1:00	점심	참가자 모두	더블루
	1:00~1:10	개회사	강성원 교수	한라홀 8F Room J 사회: 이병정 교수
	1:10~1:40	"현장 중심의 소프트웨어공학"	박수용 교수	
	1:40~2:10	"기업관점에서 미래 소프트웨어 교육의 방향"	전진옥 대표	
	2:10~2:40	"위기의 소프트웨어공학"	배두환 교수	
	2:40~3:10	"Lessons learned from Mordern Software Architecture Design"	김수동 교수	
	3:10~3:30	Break		
	3:30~4:00	"소프트웨어 테스트 분야 발전 동향"	최병주 교수	한라홀 8F Room J 사회: 고인영교수
	4:00~4:30	"SW 안전성 보증 프로세스"	한혁수 교수	
	4:30~5:00	"정형 기법의 개념 및 역사"	권기현 교수	
	5:00~5:30	"SE 연구와 소회 및 SE 교육에서 오픈소스 교육의 중요성"	이은석 교수	
	5:30~5:40	폐회사		
	6:00~9:00	Dinner		
6/21 (수)	9:00~12:00	운영분과회의	각 분과별 회의	

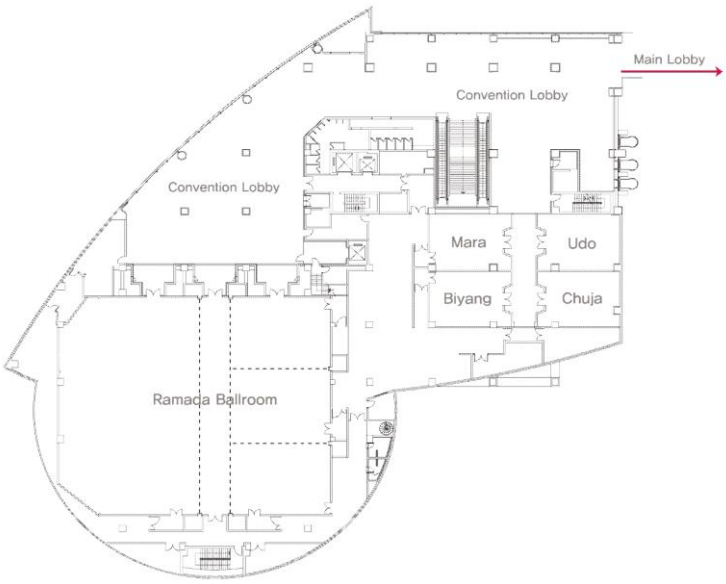
행사장 위치(지도)

○ 제주특별자치도 제주시 탑동로 66 라마다 프라자 제주호텔

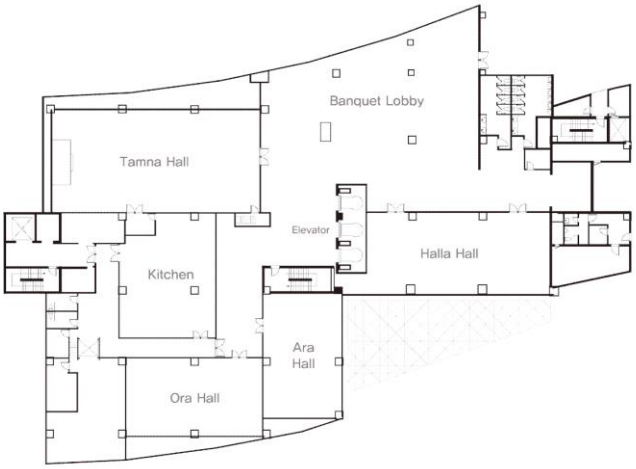
○ TEL . 064-729-8100(라마다호텔)

제주국제공항에서 라마다 프라자 제주 호텔까지는 차량으로 10분가량 소요되며, 공항에서 대중교통(버스) 이용 가능합니다.

행사장 안내



(2F)



(8F)

주요시설

-2층 로비

- 2층 라마다 볼룸(Room A, B, C, D 분할 사용)
- 2층 세미나룸(우도홀, 마라홀, 비양홀, 추자홀)
- 8층 로비
- 8 층 중연회장 (탐라홀, 한라홀, 아라홀, 오라홀)

현장 중심의 소프트웨어 공학

박수용
서강대학교 컴퓨터공학과
sypark@sogang.ac.kr

요 약

1. 지난 20여년간의 연구 정리
   - 요구 공학: 항상 필요하다고는 이해하나 현장에서는 그리 적용되지 않음
   - Agent Oriented Software Engineering: 그냥 시대 흐름에 따른 컨셉 연구
   - 적응형 소프트웨어: 미래 지향적이기는 하나 현실적이지는 못함

2. 연구 결과의 산업체 기여
   - 산학 과제는 많이 하였으나 어느 정도 산업에 기여도가 있었을까 생각 할 때 미미하다고 생각됨
   - 논문은 그런대로 다수 발표 하였으나 어떤 기여를 하였을까 그냥 논문을 위한 논문이 아니었을까 하는 생각
   - 기업에서의 강연은 꽤 하였으나 산업인력 양성에는 기여가 있는가 생각 할 때 얼마나 실무에 맞는 인재를 양성 하였는지 의문이 듦

3. 대학에서의 소프트웨어공학 교육
   - 전체적인 방법론 강의 및 실습 중심
   - 학생들은 소프트웨어공학의 필요성을 느낄 수 있을까? 현장에 가서 하고자 할까?
   - 직장인을 위한 야간 대학원에서는 수요가 많이 있었음

4. 소프트웨어공학의 연구와 교육과 산업 현장과의 거리
   - 진정 교육 및 연구가 산업과 밀접 하여야 하나? - 시간의 차이, 현실과 아카데미와의 차이 등

5. 무엇을 연구 하고 교육 하여야 하나 ?
   - SE 분야가 워낙 다양해서 하나로 일원화 하기는 힘듦
   - 협업 시 필수적인 기본적인 툴 교육: github, 형상관리, 테스팅 도구 등의 활용 교육이 필요
   - 패턴 : 아키텍쳐 패턴, 설계 패턴 등 실무적인 교육이 필요
   - 직장인들을 위한 SE 전문 교육 필요
   - CMU 의 소프트웨어공학 석사과정, GMU의 소프트웨어공학 석사과정등의 산업체에서 소프트웨어 개발자로 5년 이상 경험한 인력 들에 대한 고급 소프트웨어 개발 자로서의 교육 과정이 필요함

비트컴퓨터

기업 관점에서

미래 소프트웨어 교육의 방향

2017.06.20

전 진 옥 CEO

4차 산업혁명

(출처: http://www.zdnet.co.kr/news/news_view.asp?artice_id=20160712173539)

Fast Follower 전략도 통하지 않을 만큼 창의성을 갖춘 선두그룹과 후발주자 간 격차가 심화

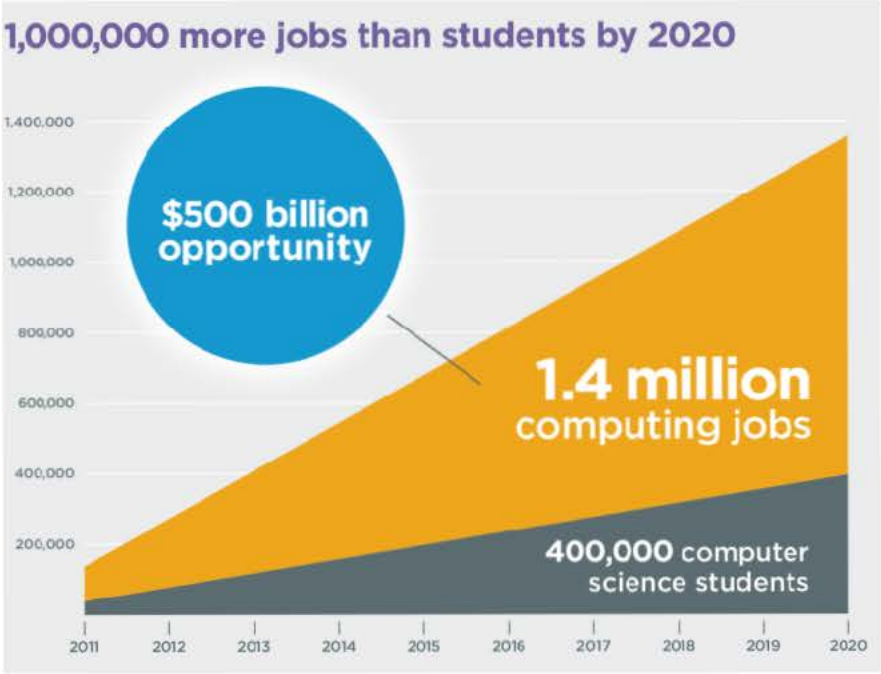
- Klaus Schwab(다보스 포럼 회장) 국회 4차 산업혁명 포럼, 2016. 10. 18.

소프트웨어 중심 사회로 이전

"All companies are now software companies"

이제 모든 기업이 SW기업이다.

Newsweek, The Top Tech Trends for 2015, 2015.1.3



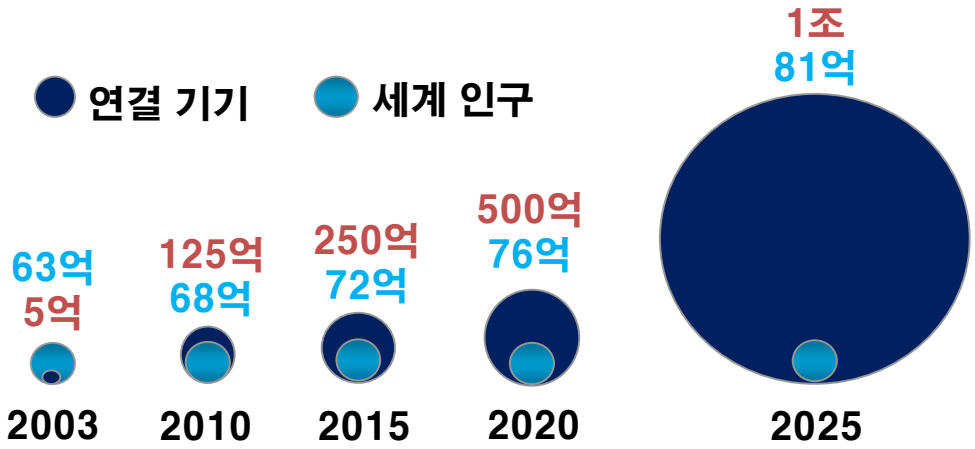
4차 산업혁명의 도래와 산업의 변화

비트컴퓨터

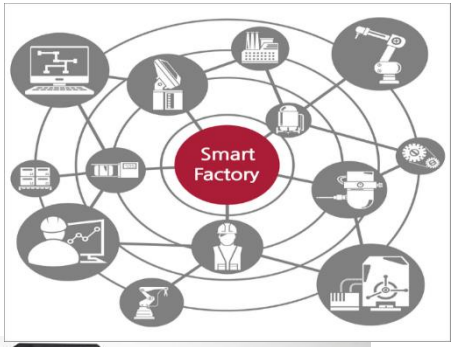
디바이스	데이터수집	데이터저장	데이터 분석	서비스
데이터 **발산** 단계 디바이스로 부터	대용량 데이터 **수집** 및 집적 단계	데이터를 저장하고 **전처리** 하는 단계	전처리 된 데이터를 **분석/가공**하는 단계	추출된 가치를 **서비스** 등 적용하여 전달하는 단계

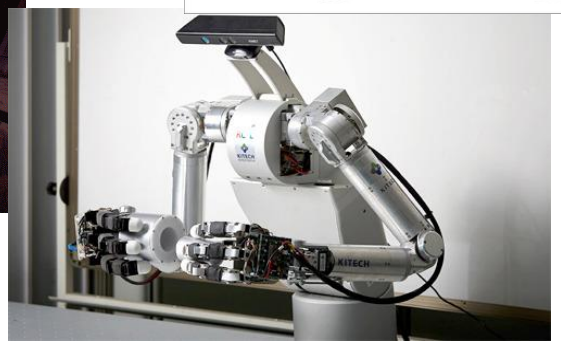
(출처) 창조경제연구회, 2016

생산과 서비스 변화



(출처) Cisco IBSG





그러나 우리는 ...

3년째 제자리에 머물고 있는 국가경쟁력

[2016년 WEF(세계경제포럼) 국가경쟁력 순위]

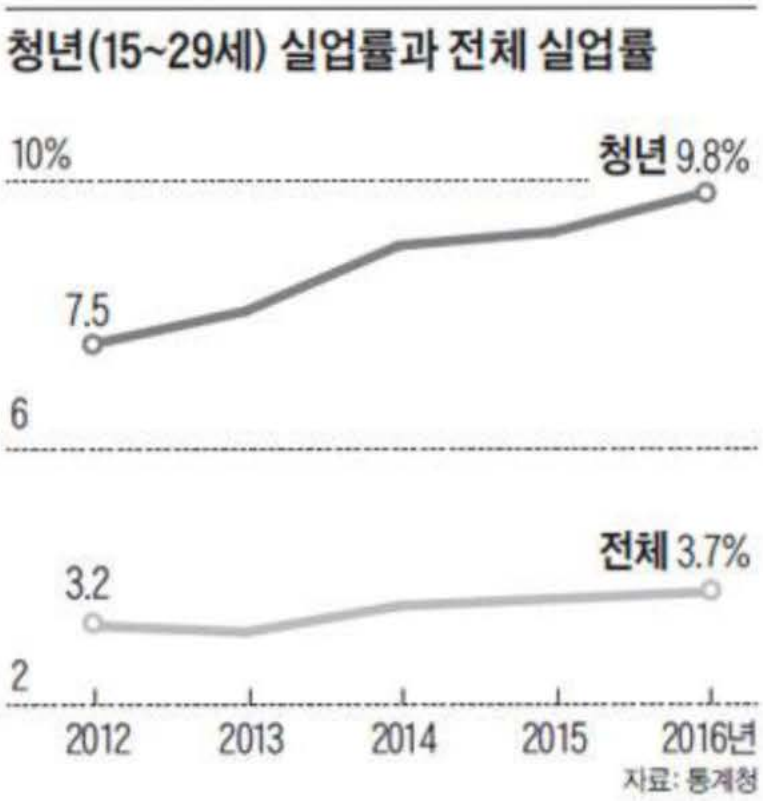
총 138개국 기준

순위	2015 순위	국가	순위	2015 순위	국가	순위	2015 순위	국가
1	1	스위스	17	19	벨기에	34	32	태국
2	2	싱가포르	18	14	카타르	35	36	리투아니아
3	3	미국	19	23	오스트리아	36	41	폴란드
4	5	네덜란드	20	20	룩셈부르크	37	40	아제르바이잔
5	4	독일	21	22	프랑스	38	34	쿠웨이트
6	9	스웨덴	22	21	호주	39	55	인도
7	10	영국	23	24	아일랜드	40	48	말타
8	6	일본	24	27	이스라엘	41	37	인도네시아
9	7	홍콩	25	18	말레이시아	42	50	파나마
10	8	핀란드	26	26	한국	43	45	러시아
11	11	노르웨이				44	43	이탈리아
12	12	덴마크				45	46	마우리투스
13	16	뉴질랜드	29	25	사우디아라비아	46	38	포르투갈
14	15	대만	30	30	에스토니아	47	49	남아프리카공화국
15	13	캐나다	31	31	체코	48	39	바레인
16	17	아랍에미레이트	32	33	스페인	49	44	라트비아
			33	35	칠레	50	54	불가리아

26위

깊어지는 청년 실업률

(출처) 삼성경제연구소, 2003. 4



(출처) 조선일보, 2017. 2

그러나 우리는 …

인력의 미스매치는 더욱 심화

(출처) 조선일보 2017. 2

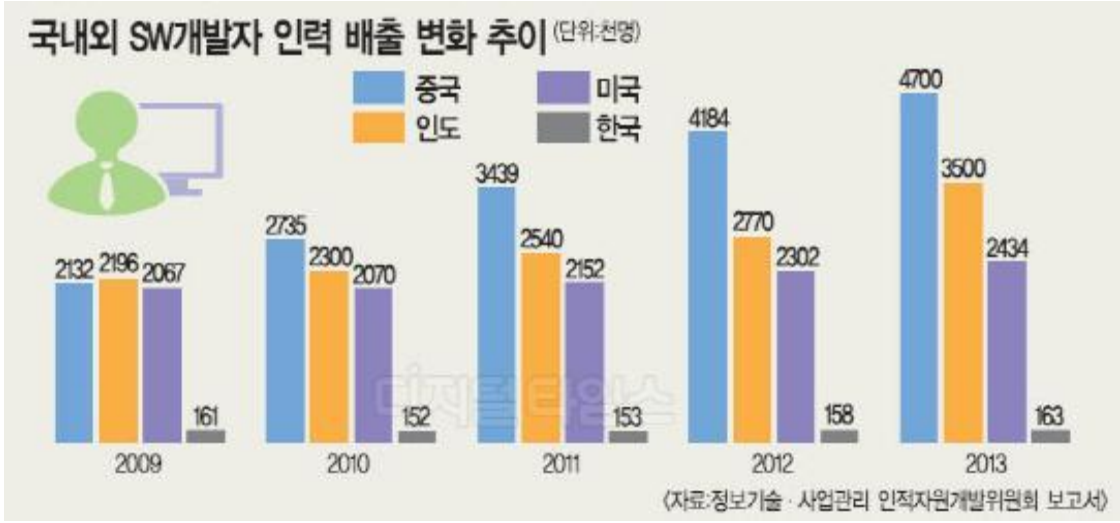
부산의 한 4년제 대학을 나온 A 씨는 3년째 '취업준비생' 딱지를 떼지 못하고 있다. 그는 거의 모든 국내 유명 대기업에 지원했지만 서류전형에서 번번이 고배를 마셨다. 지난해 대학을 졸업한 그는 "고용이 불안정한 중소기업은 아예 생각하지 않고 있다"고 잘라 말했다. 이어 "첫 직장이 중요하다는 선배들의 충고에 따라 당분간 공공기관 시험에 몰두할 계획이다"라고 덧붙였다

실업자 절반이 대졸… 中企 79%는 "인력난" (출처) 동아일보, 2016. 12. 13

그러나 우리는 ...

소프트웨어 인력의 확보는 더욱 어려움

(출처) 디지털타임즈, 2016.8



(출처) 디지털타임즈, 2016.3

인재의 중요성

" 확실히 기술, 전략, 세계적 연대 그리고 혁신 같은 차원은 모두 미래의 경쟁적 우위에 영향을 미칠 중요한 요소이다. 그러나 이러한 각 분야는 아직도 **인간의 재능에 의존하고 인간의 재능에 따라 움직이고 있다.** 따라서 미래의 경제적, 전략적 이익은 시장에서 **가장 똑똑하고 다양한 최고의 인재를** 가장 효과적으로 유인하고 개발하고 보유할 수 있는 조직에게 돌아갈 것이다. "

피터 드러커 교수의 저서『미래의 조직』中 디지털 경제와 인적 자본

소프트웨어 이상형 인재의 Key Word

1. 창의적인 사고
2. 융합적인 역량
3. 아키텍쳐

창의적 인재에 대한 불편한 진실 - 1

한국과 미국 기업이 바라는 소양과 자질

1. 전문지식과 폭 넓은 교양으로 무장된 사람
2. 국제 감각과 외국어 구사능력을 가진 사람
3. 진취적인 사람
4. 도전과 성취 의식이 있는 사람
5. 유연한 사고와 창의력을 가진 사람
6. 올바른 가치관을 가진 사람
7. 인간미 있는 사람
8. 책임감이 있는 사람
9. 협력하는 사람
10. 예의 바른 사람

(출처) 전경련관련설문조사

신규 IT 직원을 채용하는데 있어 가장 중요한 항목은?
(미국 Fortune 500대 기업)

1. Problem solving skills
2. Leaning new things quickly
3. Ability to analyze and interpret data
4. Teamwork skills
5. Oral communication skills
6. Motivation
7. Innovative Thinking
8. Being a self-starter
9. Written communication skills
10. Job-specific computer skills

(출처) Information Week

창의적 인재에 대한 불편한 진실 - 2

"KAIST 기계과는 기계공학을 이론적으로 발전시켜 세계적으로 선도하고 또 기업에 우수한 인력을 공급하는 두 가지 목표를 갖고 있지만 미국의 학교들은 기업에 대한 기여는 별로 신경 쓰지 않고 학문적인 것만 생각하는 것이 우리와 달랐다"

- KAIST H 기계학과장, 2008. 1. 2

"우선은 `세계에서 처음'하는 일에 좀 더 가치를 두는 것이 중요하다는 점(이것은 대학이 할 일)이고, 그리고 그 분야가 한국의 성장동력, 그리고 전략적 분야와 멀게라도 일치하면 좋다는 점(이것은 정부의 할 일)이다. 실제로 세계에서 처음으로 무엇을 한다는 것에 미국의 대학은 `과도하리 만큼' 집착하고 있다. 심지어 그것이 그 국가의 산업과 연관이 있든 없든 무관하게 말이다"

- 서울대 전기전자공학부 P 교수, 2008. 11

교육 혁신이 필요 : 내용 및 방법

"지금 교육은 백년 전에나
필요한 사람을 길러내고 있어요"

직업활동에 도움 되는 교과목 우선 ?

교과목은 누가 결정하나?

[4C 교육 목표: 창의력의 배양]

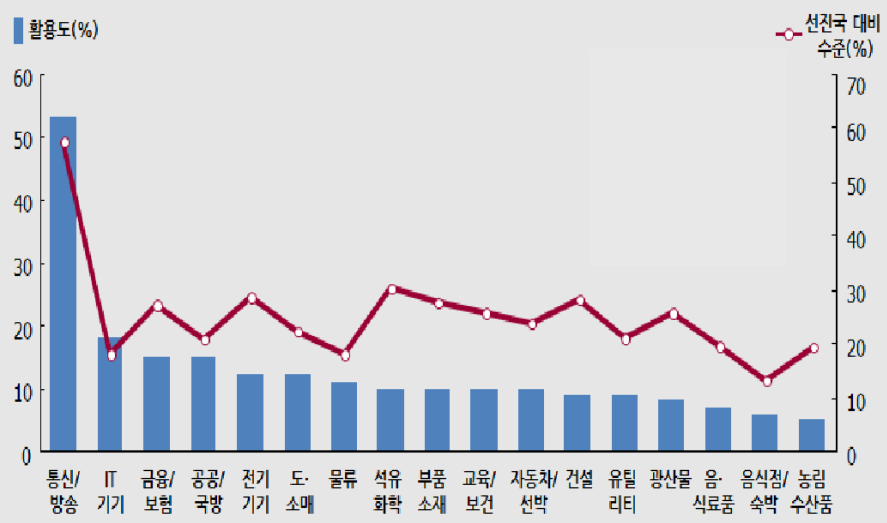
(출처) 김진현소장, SPRI, 2015. 12

소프트웨어 활용도를 통해 본 융합의 불편한 진실 - 1

BIT 비트컴퓨터

[한국의 산업별 소프트웨어 활용도 및 선진국 대비 수준]

방송통신을 제외한 거의 대부분의 분야에서 선진국의 30% 수준



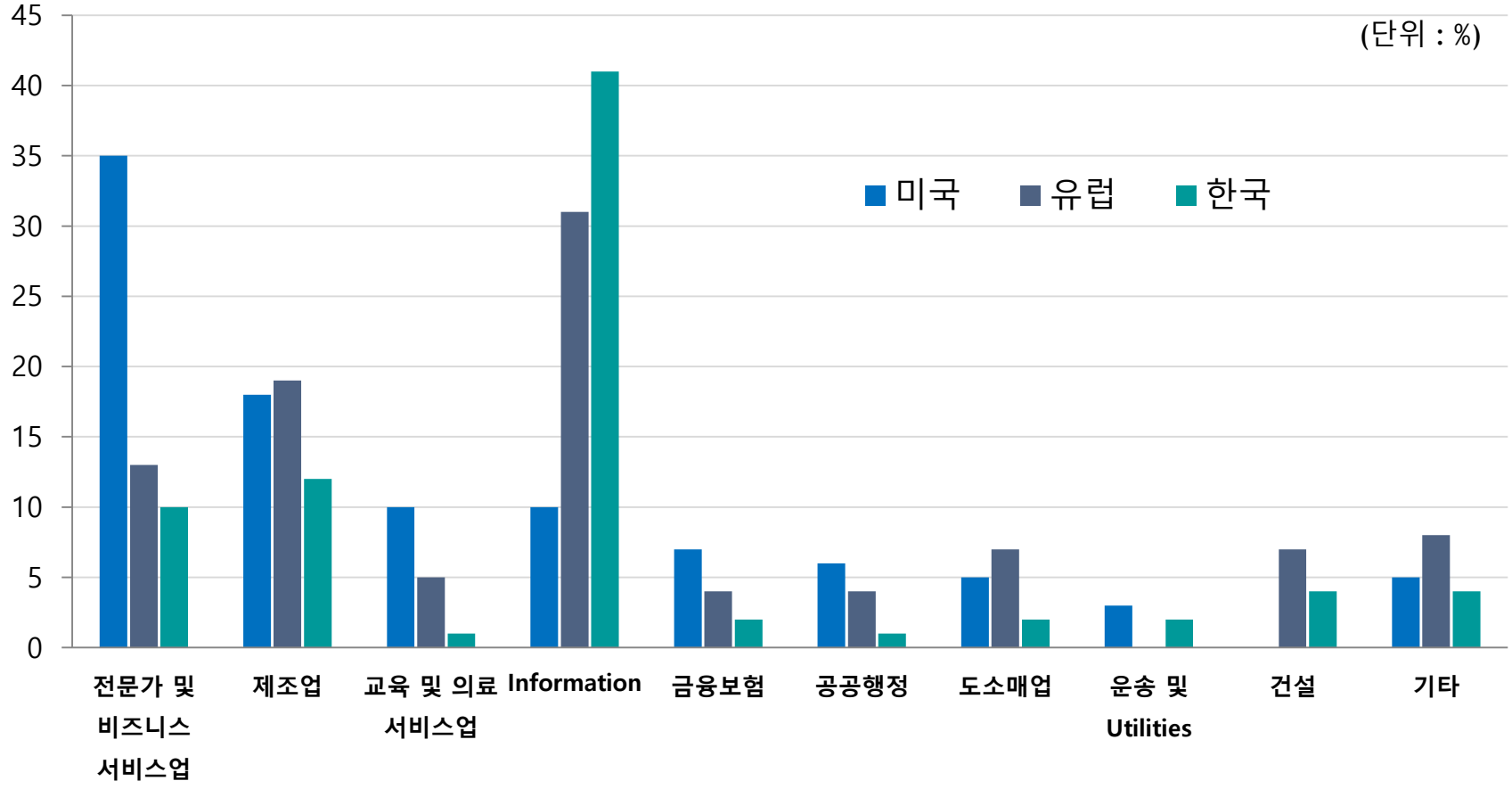
[소프트웨어 산업의 경쟁력 순위]

OECD 19 개국 중 14위

구분	산업 규모		R&D 투자 규모		효율성		종합순위
	생산액 (억$)	순위	투자액 (억$)	순위	점수	순위	
미국	4,018	1	338	1	89	3	1
일본	1,409	2	22	4	100	1	2
영국	1,193	3	29	2	100	1	3
한국	213	10	8	7	63	15	14
OECD 평균	568	-	25	-	72	-	-

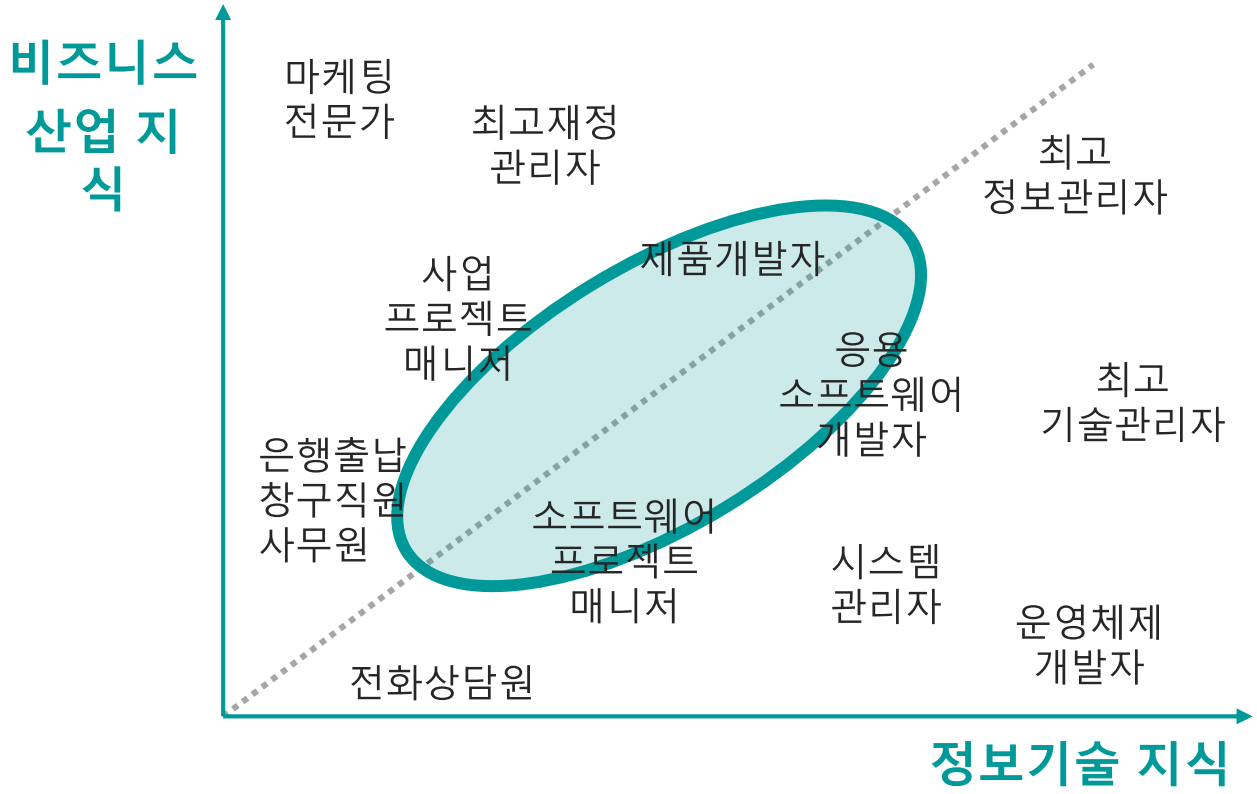
(출처) 삼성경제연구원, 2011

주요 산업별 ICT 전문가 비중



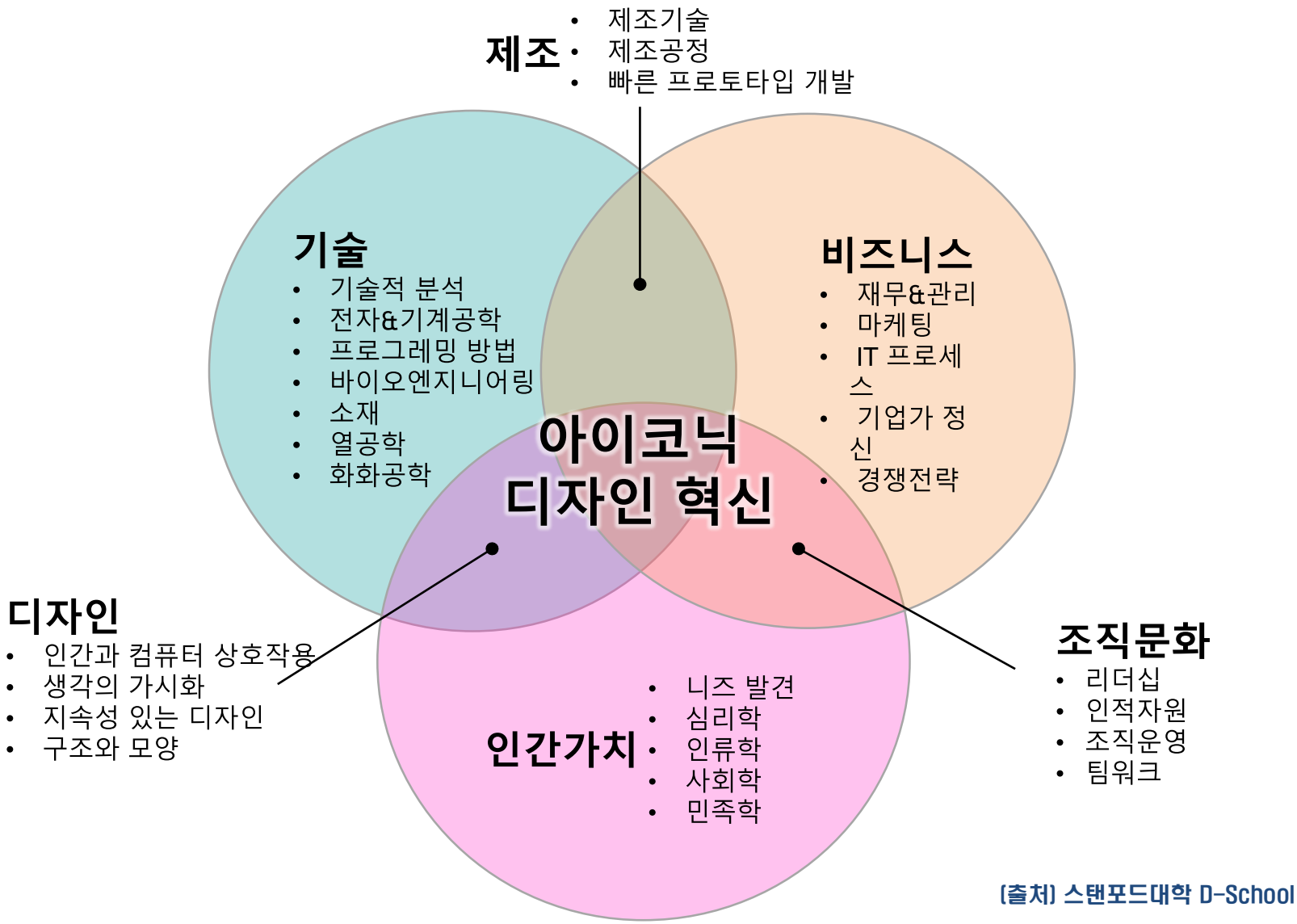
(출처) OECD, 2012.4, 통계청, 2011, KISDI, 2012

융합 인재의 요구



- SW 적용의 확대, convergence는 SW 기술 뿐만 아니라 Domain Knowledge 요구
- 시스템은 점점 더 복잡해 지고(실시간, 저 전력 등) 하드웨어, 네트워크 등 시스템 전반에 대한 광범위한 지식 요구

학제간(Interdisciplinary) 교육



(출처) 스탠포드대학 D-School

아키텍처 인재에 대한 불편한 진실 - 1

삼성 사내방송 SBC는 이날 오전 특별기획 '삼성 소프트웨어 경쟁력 백서 2부, 우리의 민낯'을 방영했다. SBC는 이날 방송에서 **"설계가 잘된 소프트웨어는 뭔가를 새롭게 바꾸거나 확장하기 쉬운 반면, 설계가 잘못되면 작은 개선을 하는 것조차도 쉽지 않다"**고 꼬집었다.

이날 방송의 초점은 소프트웨어의 큰 그림을 그리는 아키텍처(architecture) 역량의 현 수준을 진단하고 삼성이 가진 소프트웨어 리더로서의 규모에 비해서 아키텍처로서의 역량이 부족하다고 진단하고 **"기본적으로 구조설계를 하는 아키텍처 역량이 뒷받침되지 않으면 소프트웨어 역량을 발휘할 수 없다"**고 말했다.

즉 소프트웨어의 뼈대를 잘 설계해야 하지만 그 동안에는 아키텍쳐가 제대로 설계되지 않아 매번 소프트웨어를 꾸미고 변경하는데 시간과 인력을 낭비했다고 진단했다. 특히, 설계가 잘못되면 작은 개선을 하는 것조차도 쉽지 않다고 강조했다.

〔출처〕 “소프트웨어 경쟁력 높이지 못했다 “ 연합뉴스 외, 2016. 7. 5.

아키텍처 인재에 대한 불편한 진실 - 2

비트컴퓨터

분야별 소프트웨어 개발자 경력별 분포 현황

(단위: 명, %)

구 분	0~2년	3~5년	6~8년	9~11년	12~14년	15년 이상	합 계
패키지 SW	21,153	8,491	2,053	420	295	295	32,707 (11.8)
IT서비스	140,391	69,566	18,775	3.488	2,850	3,481	238,551 (86.0)
임베디드 SW	3,630	1,747	391	81	74	91	6,014 (2.2)
합 계	165,174 (59.6)	79,804 (28.8)	21,219 (7.7)	3.989 (1.4)	3,219 (1.2)	3,867 (1.4)	277,272

(출처) 소프트웨어산업협회, 2016. 7. 8.

경력 5 년 이하가 전체의 88.4%(244,978명)를 차지하고 있으며,
경력 9 년 이상은 전체의 4.0% (11,075명)

아키텍쳐의 양성

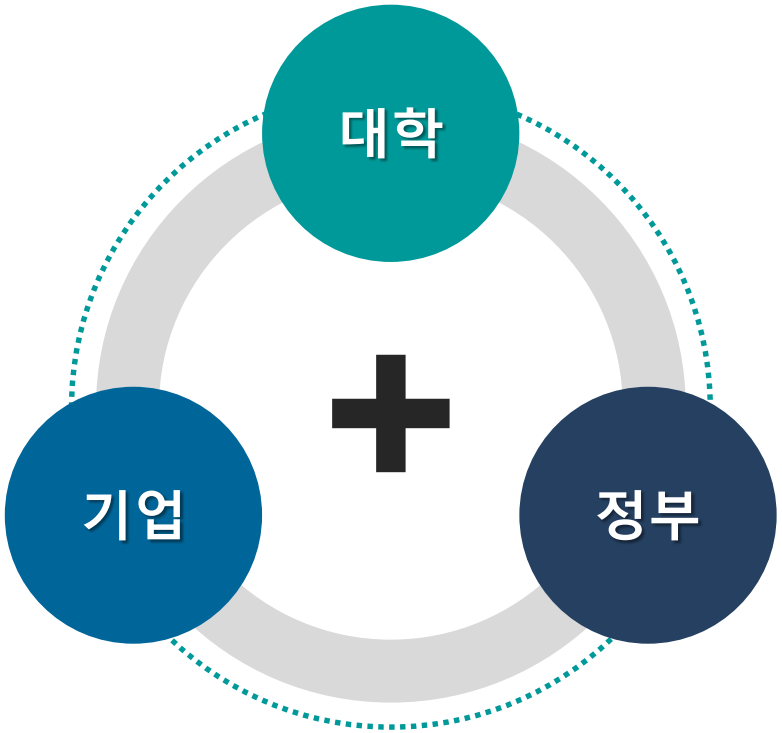
- 미국의 경우 2만불 달성한 1988년 이후
  고급 인재 양성 정책으로 인력의 고도화 실현

	1988년	1997년
Computer Programmer	45%	30%
System Architecture	38%	60%

(출처) 소프트웨어진흥원, 2005

- 우리나라의 경우 SW 개발자 가운데 프로그래머가 차지하는 비율이 64% 수준 (2016년 4월 기준, 소프트웨어 산업협회)
- 일본에서는 "Software Factory" 에서 SW 개발자 Career Path 관리를 통한 System Architecture 양성

소프트웨어 인력양성 방향



대학

- ✓ 문제 해결 능력 등의 교육을 통한 장기적 관점에서 인재 양성
- ✓ 학제간(Interdisciplinary) 교육을 통한 폭넓은 사고와 지식 숙지

기업

- ✓ 급변하는 기술 환경에서 개발자 Career Path관리를 통한 고급인력 양성
- ✓ 대학과 역할 분담을 하여 필요로 하는 직무교육 담당

정부

- ✓ 국가 IT인력양성 중장기 계획 수립 및 일관성 있는 추진
- ✓ 양적 관리에서 질적 관리

감사합니다

위기의 소프트웨어공학

배 두환
전산학부
한국과학기술원

목차

- 4차 산업 혁명과 SW
- “Computer Programming is a Dying Art”
- 우리 나라에서의 “SW”는?
  - 정부 정책에서의 “SW”는?
  - 우리 소프트웨어공학의 모습은?
- 문제점 및 해결책?

1 차 산업 혁명 이전

- 18세기 이전
- 농업, 유목

1차 산업 혁명

- 18세기
- 증기기관을 이용한 기계화

2차 산업 혁명

- 19세기 후반- 20세기 초
- 전기 에너지 기반의 대량 생산

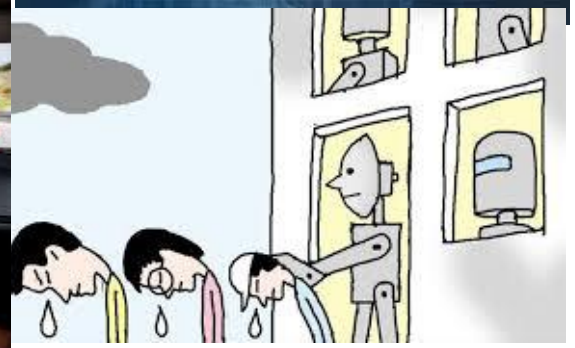
3차 산업 혁명

- 1960년대 - 2015년
- 컴퓨터 기반의 지식 정보화

4차 산업 혁명

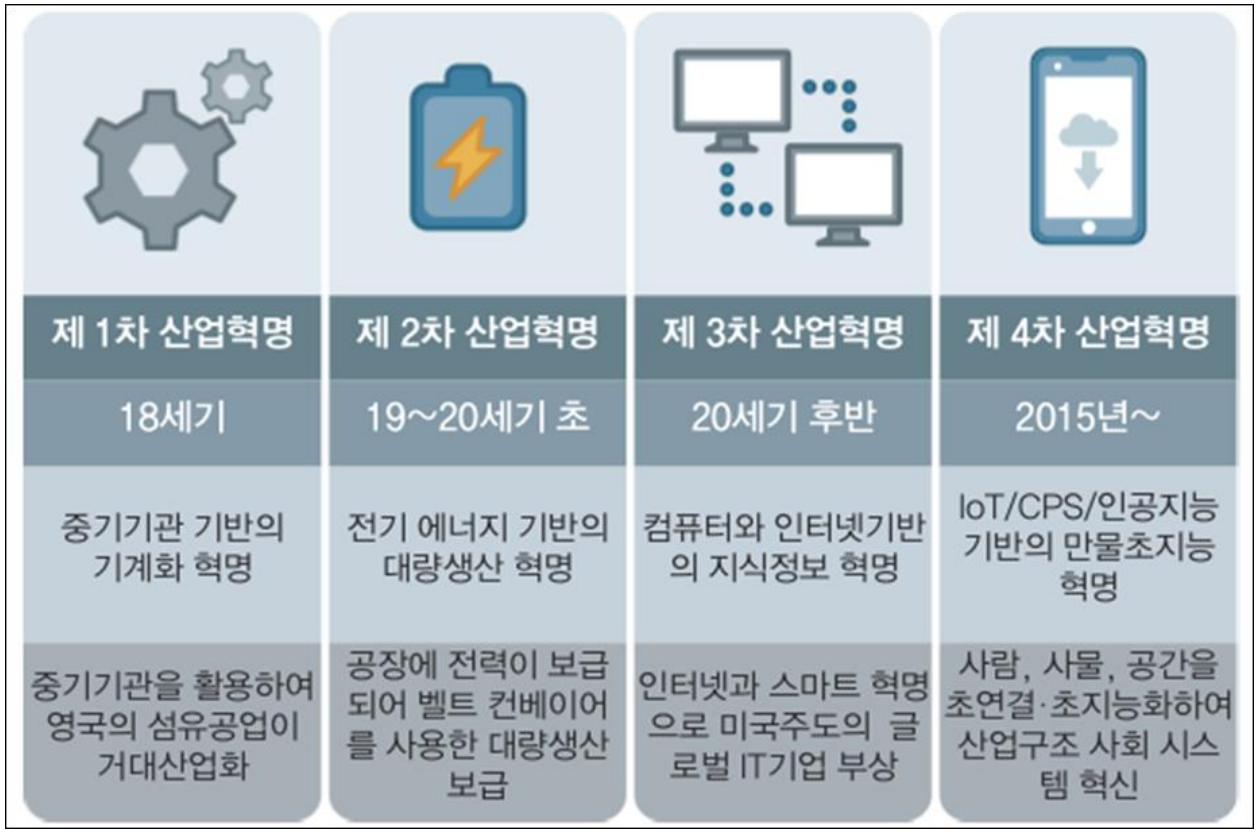
- 2016년 – ??
- 사람, 사물, 공간의 초 연결/초 지능을 통한 혁신




산업 혁명의 핵심 원동력은?

- 1차: 증기 기관:
  제임스 와트(공학자)
- 2차: 전기:
  패러데이, 맥스웰, 톰슨
  (과학자)
- 3차: 컴퓨터:
  콘라드 추제 (공학자)
  앨런 튜링(컴퓨터과학자)
- 4차: SW: 인공지능, 빅 데이터,..



산업 혁명과 사라진 직업들

4차 산업 혁명과 선진국

(출처:주간 조선, 2017년 6월 17일, “관이 낳은 버블 한국에만 있는 4차 산업 혁명”)

- 2016년 1월 스위스 세계 경제 포럼(WEF), 클라우스 수밥의 제안
  - 3차 산업 혁명을 기반으로 한 디지털과 생명공학, 물리학등을 융합하는 기술 혁명
- 독일: 인터스트리 4.0
  - Digital transformation
  - Digital disruption
- 미국: Digitalization/Innovation
  - Facebook, Apple, Amazon, Netflix, Google (FAANG)

- 정부의 역할: “개별 기업이 아닌, 정부만이 할 수 있는 분야..”
  - “정부가 할 일은 안하고, 안 할 일은 한다.”

4차 산업 혁명과 Software

- 4차 산업혁명 시대의 인류 절반의 실직자들 중에 프로그래머가 차지하는 비중은?
  - 전혀 없음
  - 절반 정도
  - 절반 이상
- 만약 프로그래머가 줄어든다면(늘어난다면) 소프트웨어 공학자의 수요는?
  - 줄어든다
  - 늘어난다
- 그 많은 소프트웨어는 누가 만드나?
  - 일반인
  - 소프트웨어 공학자/프로그래머
  - 지능 정보시스템

목차

- 4차 산업 혁명과 SW
- “Computer Programming is a Dying Art”
- 우리 나라에서의 “SW”는?
  - 정부 정책에서의 “SW”는?
  - 우리 소프트웨어공학의 모습은?
- 문제점 및 해결책?

"Computer Programming is a Dying Art"(1/4)

(Source: Newsweek, by Kevin Maney, 2014)

- Mama, don't let your babies grow up to be programmers. It's a dying art.
- http://www.newsweek.com/2014/06/06/computer-programming-dying-art-252618.html

“Computer Programming is a Dying Art”(2/4)

- “There is definitely a need for people to learn kind of a computer science way of thinking about problems, but not necessarily the language du jour,” says Erik Brynjolfsson, a professor at the MIT Sloan School of Management
- Irving Wladawsky-Berger, formerly of IBM and now at New York University, “We should definitely teach design. This is not coding, or even programming. It requires the ability to think about the problem, organize the approach, know how to use design tools.”
- But, in 2030, when today’s 10-year-olds are in the job market, they’ll need to be creative, problem-solving design thinkers who can teach a machine how to do things.

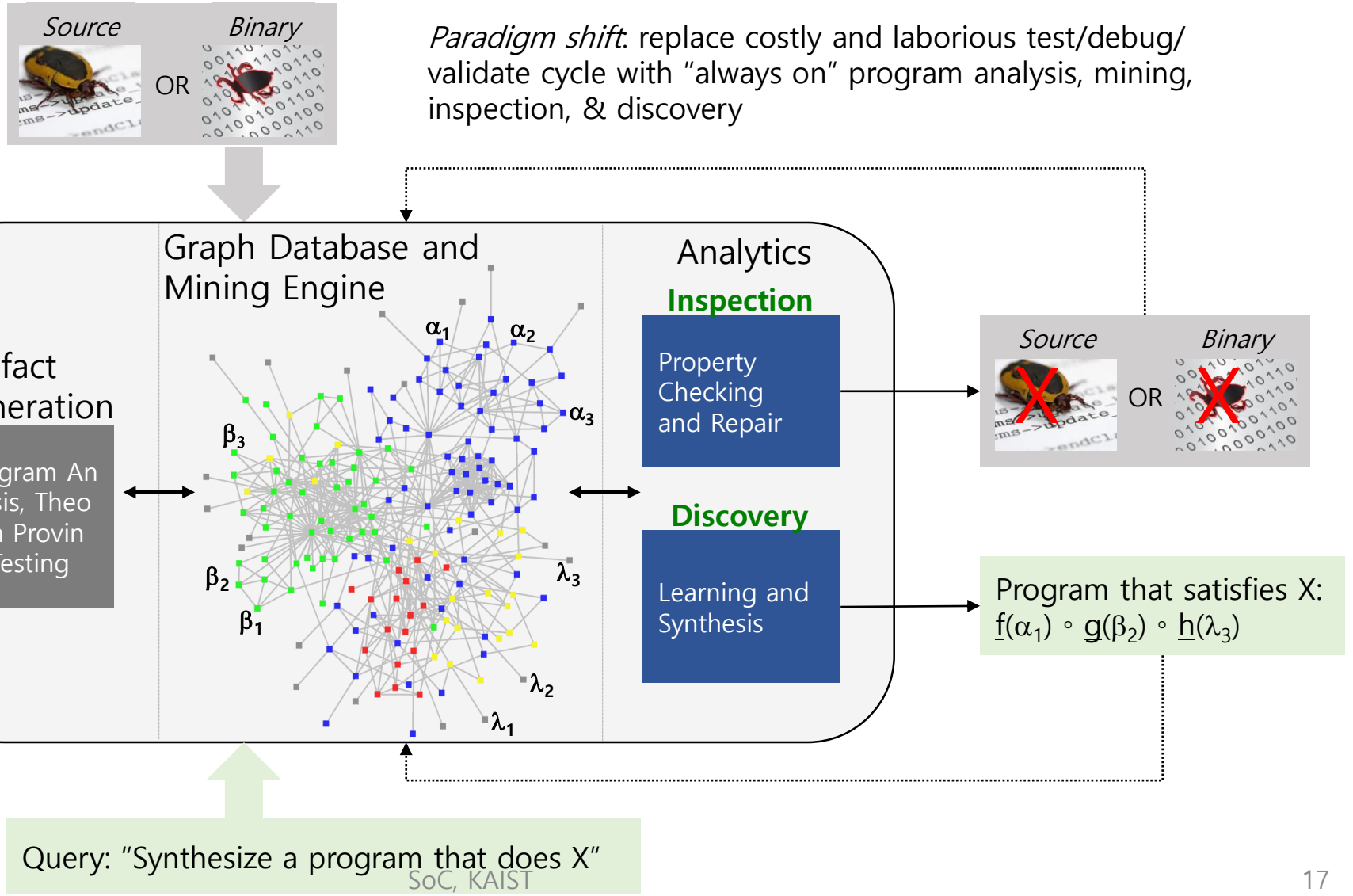
“Computer Programming is a Dying Art”(3/4)

- IBM Research chief John Kelly likes to say we’re at the dawn of an era of brain-inspired cognitive computers—and the beginning of the end of the 60-year reign of programmable computers that required us to tell them what to do step by step.
- The next generation of computers will learn from their interactions with data and people.
- In another decade or so, we won’t program computers—we’ll teach them (최근의 알파고의 모습)

"Computer Programming is a Dying Art"(4/4)

- MUSE(Mining and Understanding Software Enclaves)
- http://www.darpa.mil/program/mining-and-understanding-software-enclaves
- The MUSE program is interested in close and continued collaboration of experts from a range of fields, including but not limited to: programming languages, program analysis, theorem proving and verification, testing, compilers, software engineering, machine learning, databases, statisticians, systems and a multitude of application domains. The program intends to emphasize creating and leveraging open source technology.

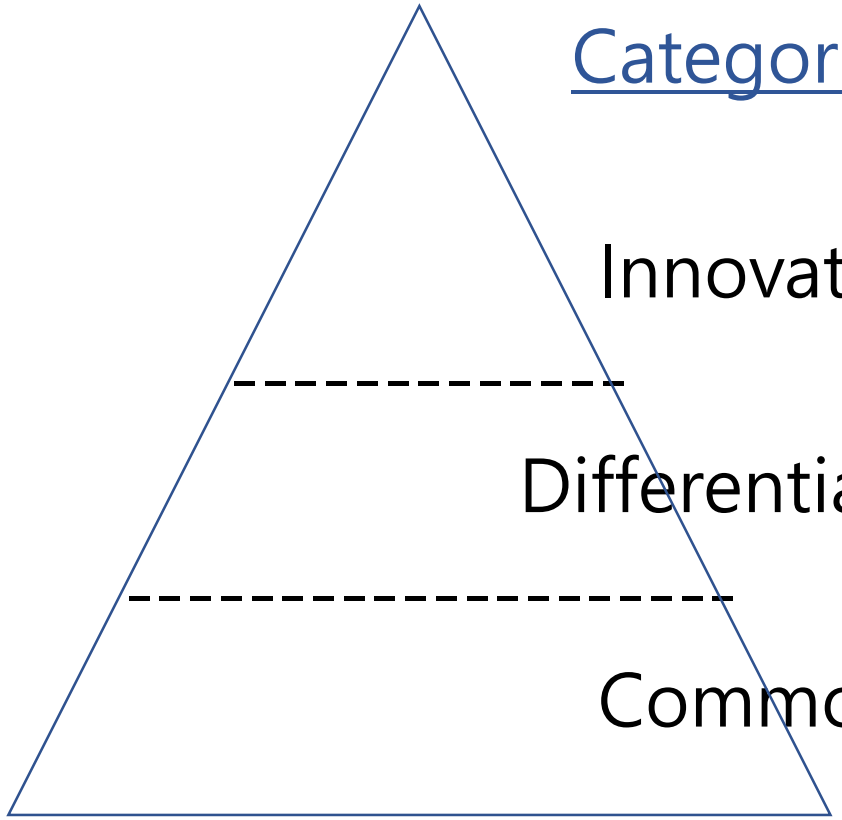
A New SW Development Paradigm?: MUSE



Paradigm shift: replace costly and laborious test/debug/ validate cycle with "always on" program analysis, mining, inspection, & discovery

SW Categorization by Jan Bosch

- What is/will be (traditional) SE for?



Categories	Characteristics
Innovation	Creative, New market
	Traditional SE?
Differentiation	Value-added, Time-to-Market
	Traditional SE?
Commodity	Open-Source
	Traditional SE?

목차

- 4차 산업 혁명과 SW
- “Computer Programming is a Dying Art”
- 우리 나라에서의 “SW”는?
  - 정부 정책에서의 “SW”?
  - 우리 소프트웨어공학의 모습은?
- 문제점 및 해결책?

"SW" 가 사라지고 있다

- 미래부의 미래성장동력 확보 분야
  - 2017년도 업무계획 보고(2017년 1월 6일)

◇ **글로벌 창업·혁신의 중심지**로 **'판교 창조경제 밸리'** 구축
◇ **창조경제혁신센터 민간참여 확대** 및 **자립기반 마련**
◇ **'미래기술 1·2·3호 펀드'**(총 1,500억)의 **본격 운용**
◇ **자유공모형 기초연구 확대**(8,779억원, 1,152억원↑), **생애 첫 연구비**(1,000과제, 총 300억원) 신설
◇ **바이오경제 견인** 및 **탄소 자원화 실증 플래그십 프로젝트** 추진
◇ IoT, 클라우드, 빅데이터, 모바일, 정보보호 **기반 융합 신시장 창출**
◇기본적인 **국가서비스**(국방,안전, 교육 등)에 지능정보기술 활용
산업영역별(제조업·의료·교통·스마트홈 등) **지능형 융합서비스** 확산
◇ 지능정보기술이 가져 올 **사회변화**(고용, 교육, 복지 등) 대책 수립
◇ 범국가적 **지능정보사회 추진체계 마련**

“SW”는 가고 “지능 정보”의 시대가 온다

- 2017년도 미래부의 차세대 정보 컴퓨터 사업 현황
  - SE

  (1) 자동 프로그래밍 관련 소프트웨어 원천기술 개발 (4개 과제 내외 선정)
  (2) 지능형 CPS 또는 Embedded System 소프트웨어 원천기술 개발 (4개 과제 내외 선정)

  - 정보 및 지능시스템

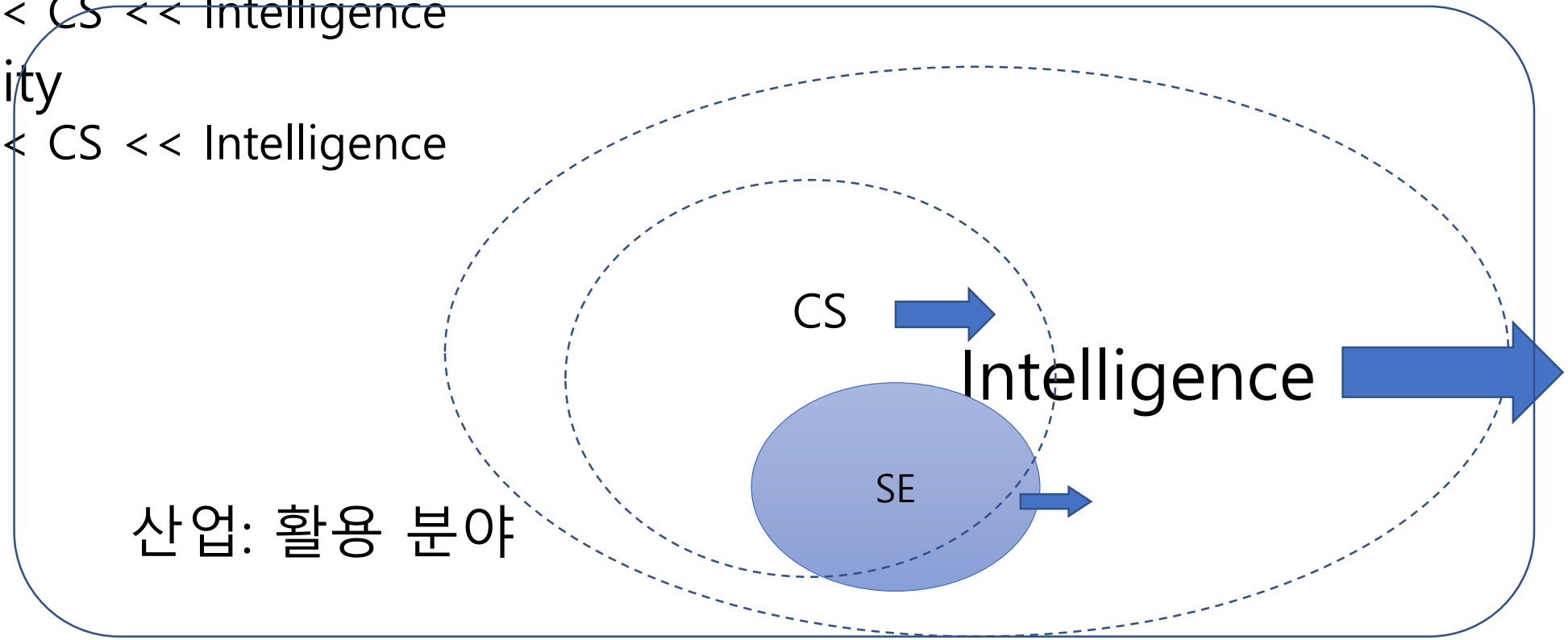
  (3) 동영상 기반 상황인지를 위한 영상처리 및 이해 소프트웨어 원천기술 개발 (2개 과제 내외 선정)
  (4) 문맥 및 상황 정보를 포함한 한국어 이해 및 처리 소프트웨어 원천기술 개발 (1개 과제 내외 선정)
  (5) 차세대 지능형 기계학습 원천기술 개발 (1개 과제 내외 선정)

  - HCI

  (6) 초실감 원격가상 인터랙션 원천기술 개발 (1개 과제 내외 선정)
  (7) 인간 대면 상태 인지 스마트 인터랙션 원천기술 개발 (1개 과제 내외 선정)

SE, CS, Intelligence(AI, ML)

- 4차 산업 혁명 시대에서의 확산 속도(→ 의 크기)
  - SE < CS << Intelligence
- Visibility
  - SE < CS << Intelligence



우리 나라에서의 SW 현실은?

- 많은 학생들이 SW 관련 과목을 듣고 있다.
- 기업체는 필요한 SW 인력 수급이 어렵다고 한다.
- "4차 산업 혁명의 원동력! 소프트웨어"는 정말인가?
  - 전산, 컴퓨터 전공자와 타 전공자와의 인식의 차이는 없는가?

- Software Engineering 의 모습은?

SE의 학문적 Visibility:
of Papers with First Author = Korean

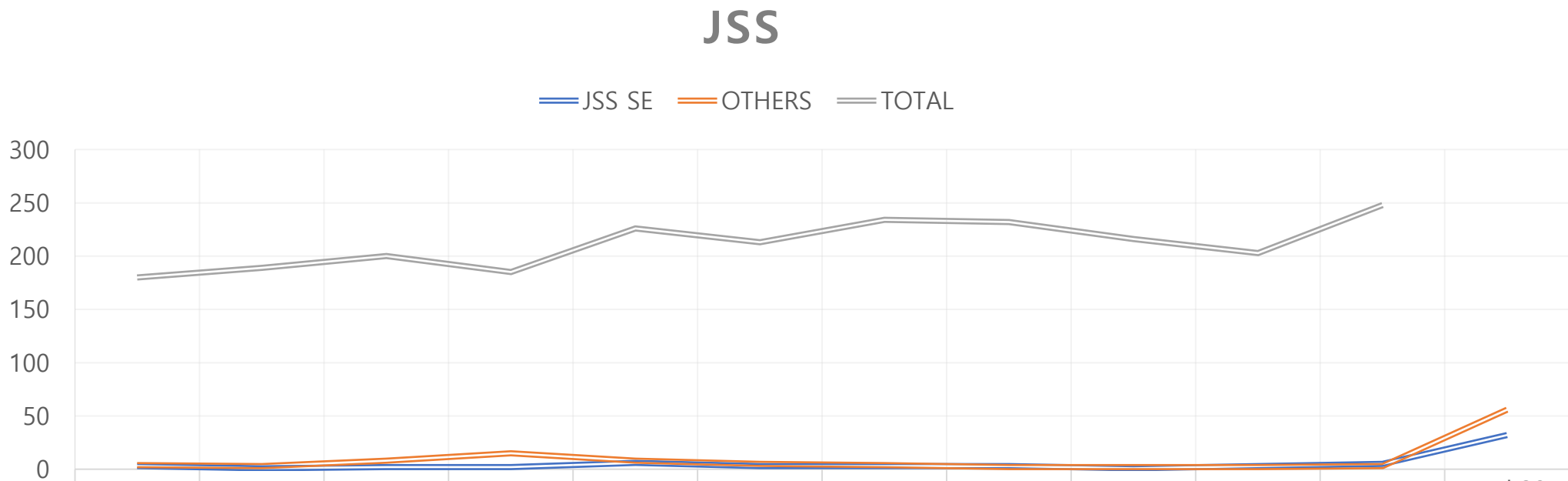
- Journals/
  Top conferences 2015 : 2016

	Total	SE	Others	Total	SE	Others
TSE	1/66	1	0	1/92	1	0
IST	2/158	2	0	0/135	0	0
JSS	5/203	3	2	8/248	5	3
ICSE	1/84	1	0	0/96	0	0
FSE	1/73	0	1	2/76	0	2
ASE	4/55	2	2	2/72	1	1
Total	13/639	8	5	13/669	7	6

SE의 학문적 Visibility:
of Papers with First Author = Korean

- ISSTA2016: 0/37
- RE2016: 0/44
- MODELS 2016: 0/38
- ICSME2016: 0/38
- APSEC2016: 3/45
- ISSRE2016: 2/45 (SE 2)
- ACM SIGMOD 2106 (4/100), 2017(4/100)
- ICML 2017 (11/434)
- ISCA 2017 (5/54)
- ICSE 2017 (2/69: one security, one PL)

JSS Publication Analysis (2006 – 2016)



	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	total(32, 56)
JSS SE	3	1	2	2	6	3	3	3	1	3	5	32
OTHERS	4	3	8	15	8	5	4	2	2	2	3	56
TOTAL	180	189	200	185	226	213	234	232	216	203	248	

현황/문제점 및 해결책?

- 논문 쓰기가 어렵다.
- SE 교수 충원이 쉽지 않다.

- 앞으로 좋아질 가능성이 있나?
- 소프트웨어공학은 미래가 어떨까?
  - 만약 “Programming is a dying art” 가 된다면, SE는 어떨까?

경청해 주셔서 고맙습니다.

- 오늘 좋은 자리를 만들어 주신 강 성원 SE 소사이어티 회장님외 여러분들께 감사드립니다.

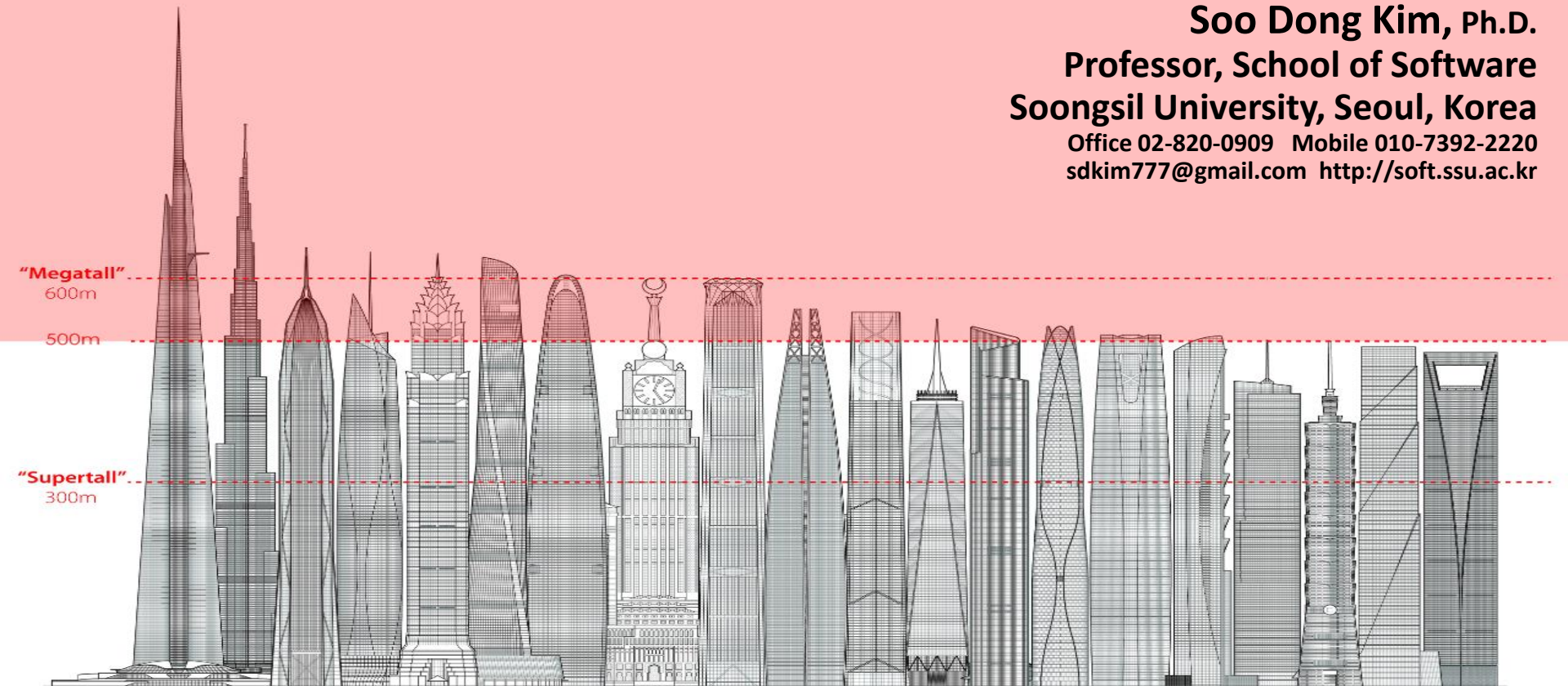
- 우리 모두의 중지를 모아 나아갈 방향을 찾아야겠습니다.

한국정보과학회
소프트웨어공학 역사 워크샵

LESSONS learned
Modern SOFTWARE ARCHITECTURE DESIGN

June 20, 2017

Soo Dong Kim, Ph.D.
Professor, School of Software
Soongsil University, Seoul, Korea
Office 02-820-0909 Mobile 010-7392-2220
sdkim777@gmail.com http://soft.ssu.ac.kr



Abstract

Architecture design is the key activity in developing high quality software systems. It becomes even more important for constructing modern software systems with emerging IT trends such as machine learning and digital technology platform.

In this lecture, the speaker shares the 7 lessons learned from his experience of designing modern systems architecture. Such systems typically embed high functional complexity and challenging non-functional requirements.

Contents

Reviewing Modern Building Architectures

- **Functional**
- **Non-Functional**
  - Quality

Lesson 1

Architecture Design is the key success factor for developing modern software systems.

Lesson 2

Architecture Design Process and Template are essential for constructing high-quality Architecture Descriptions.

Lesson 3

Architecture Styles can be Systematically Selected and Integrated.

Lesson 4

View-specific Design the Glue for Integrating Everything.

Lesson 5

Architecture Design for Non-Function Requirement (NFR) an be systematically performed.

Lesson 6

Traceability Enforcement among AD Artifacts is an Effective Architecture Validation.

Lesson 7

Architectural BOK is more important than Project Experiences for Professional Architects.

Thank you

소프트웨어 테스트 분야 발전 동향

– 2017 KCC 소프트웨어공학 소사이어티 “소프트웨어공학 역사” 워크샵 –

2017.6.20

이화여대 최병주 http://home.ewha.ac.kr/~bjchoi bjchoi@ewha.ac.kr

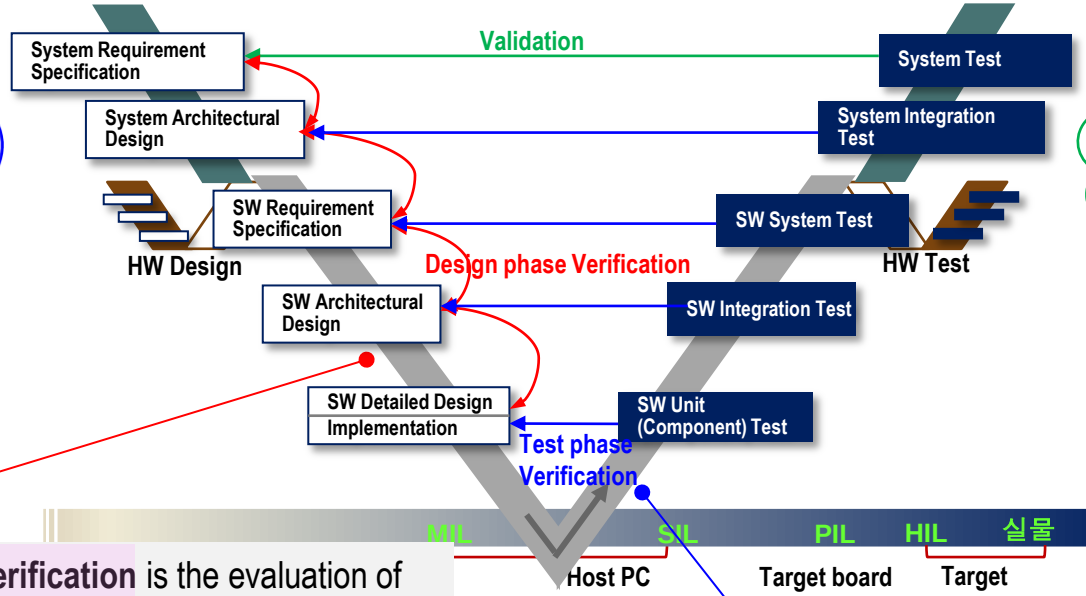
목 차

- 개념: Verification, Validation, Review, Test
- 표준
- 국내동향:연구
- 국내동향:산업
- Discussion: 나아갈 방향

개념: Verification, Validation, Review, Test

Verification

"Are we building the **product right** ?"



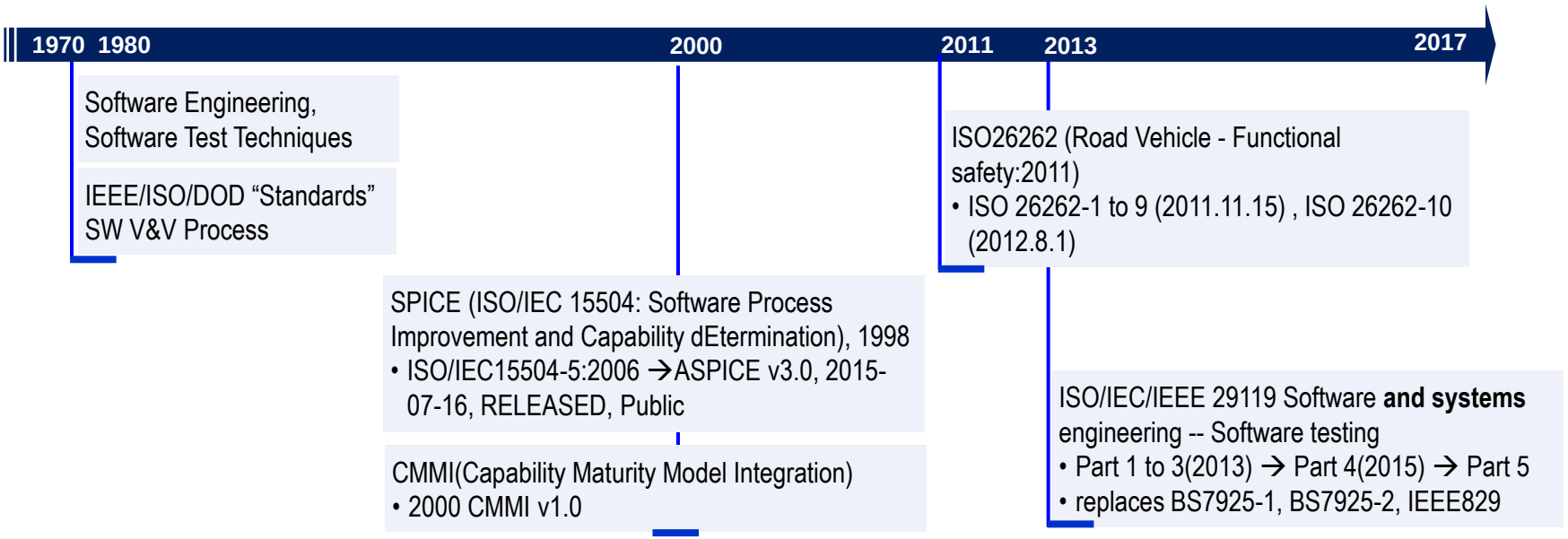
Validation

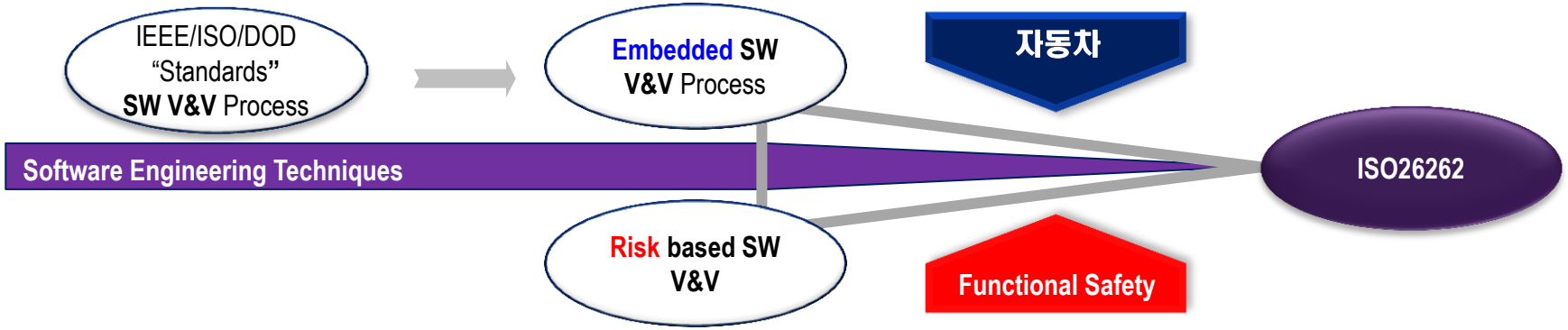
"Are we building the **right product** ?"

- **Design phase verification** is the evaluation of the **work products**, such as requirement specification, architectural design, models, or software code, thus ensuring that they **comply with** previously established requirements **for** correctness, completeness and consistency
- Evaluation can be performed **by** review, simulation or analysis techniques
- The evaluation is planned, specified, executed and documented in a systematic manner

- **Test phase verification** is the evaluation of the work products **within** a test environment to ensure that they comply with their requirements
- The tests are planned, specified, executed, evaluated and documented in a systematic manner

관련 표준





동향: 국내 연구

조사방법

- **범위**
  - 한국정보과학회 논문지- 컴퓨팅의 실제: 2000-2016
  - 한국정보과학회 논문지- 소프트웨어 및 응용: 2000-2016
  - 한국정보과학회 학술대회: 1978-2016)
- **내용**
  - SW Design Phase Verification
  - SW Test Phase Verification
  - Validation
- **특이사항**
  - IT 연구 분야의 제품 개발위주 논문 → 개발 후 성능평가 및 검사된 경우 **제외**
  - 요구분석 및 설계 검증 방법, 모델 체킹/ 일관성검사 포함
    (**단**, 코딩규칙, 코딩 가이드라인, 설계 규칙은 **제외**)
  - TC에는 오라클 및 입력 데이터 생성 모두 포함
  - 디버깅, 벤치마킹, 진단, 모니터링 **제외**

조사결과

시험단계
- 요구검증
- 설계검증
- 코드검증, 정적분석
- 단위시험
- SW통합
- SW시스템 시험
- SW-HW통합 시험
- 시스템통합 시험
- 시스템 시험
- GUI
- 품질평가/인증
- 기타 (모두)

시험 연구분야
- 모델 체킹, 평가, Testability
- TC, 체크리스트
- 커버리지, 기법
- 테스트/검증프로세스, 테스트성숙도모델
- 자동화
- 재 테스트
- 기능 측정
- 성능측정(시간,자원)
- 보안성
- 신뢰성
- 사용성
- 강건성
- 테스트레포트/관리
- 기타 (잘 모름)

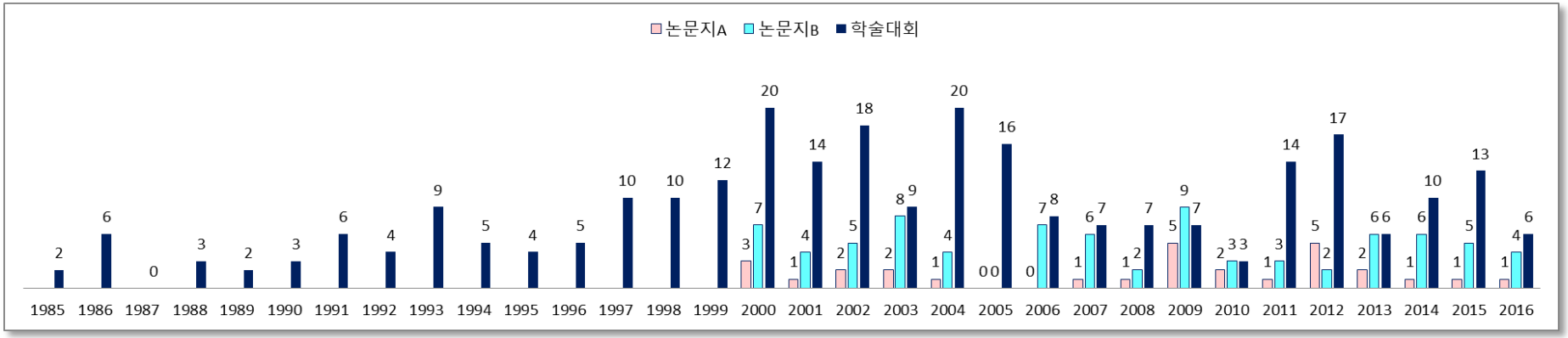
적용 도메인

SW
- 구조적SW
- 병행성
- 객체지향
- CBD/EJB,CORBA,DCOM/SOA
- 프로덕트라인
- 웹
- 클라우드 서비스
- 빅데이터
- 오픈SW

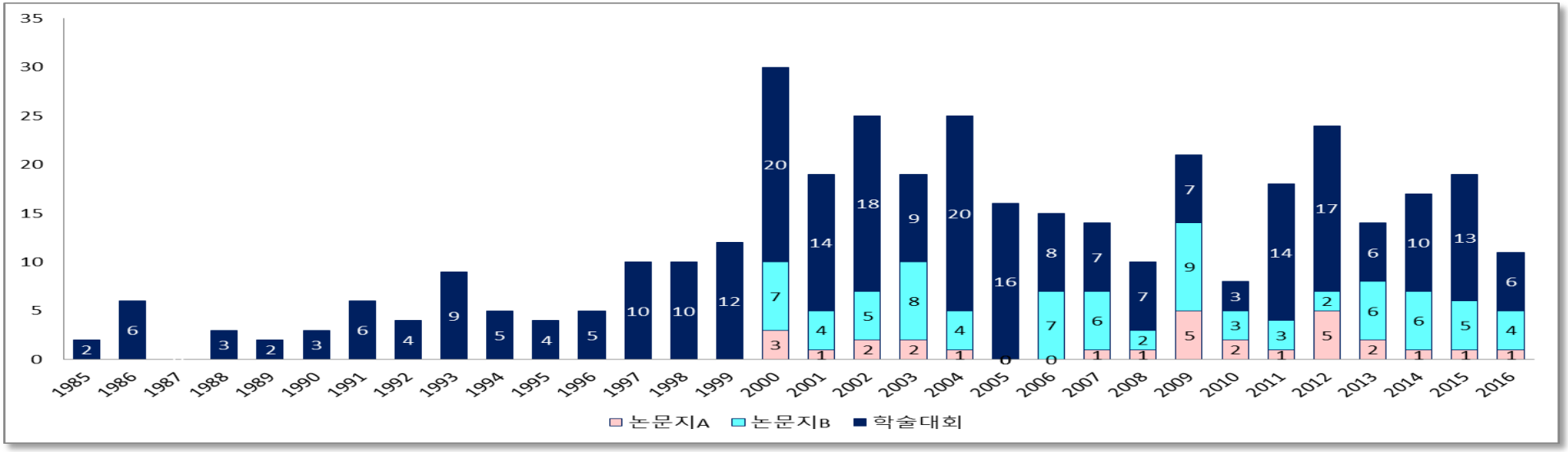
임베디드 SW
- 안드로이드,iOS/모바일앱/모바일플랫폼
- 열차
- 자동차
- 무기체계
- TV/멀티미디어
- 스마트홈 시스템
- 통신SW
- 원전
- 의료용
- 로봇
- 커널/RTOS
- 일반 임베디드 SW

발표 개수

논문지A: 컴퓨팅의 실제	논문지B: 소프트웨어 및 응용	정보과학회 학술대회

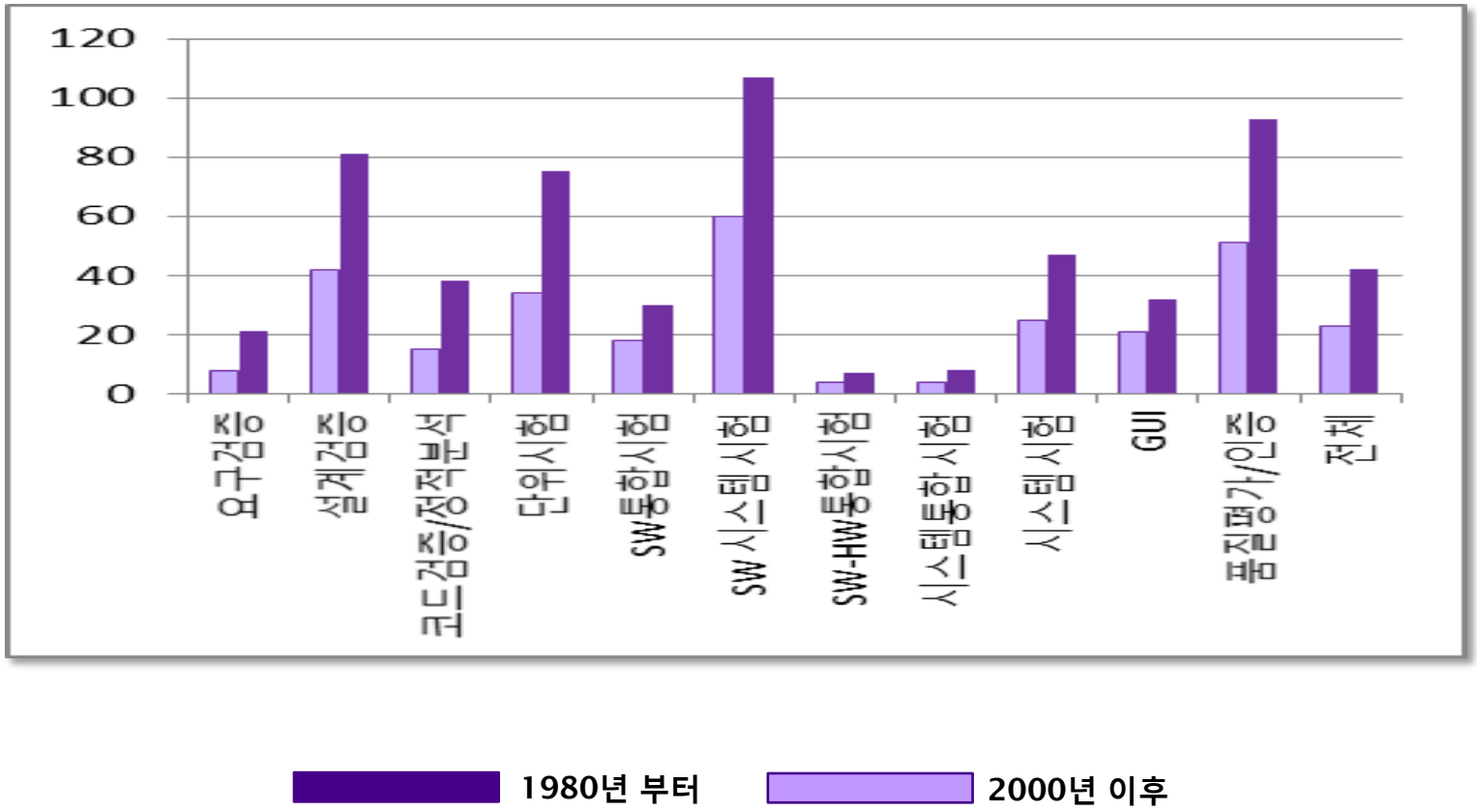


1985 2000 2016



" 주의: 단기간에 조사하는 관계로, 누락된 논문이 있을 수 있음을; 결과가 다소 차이가 있음을 양해 부탁 드립니다 "

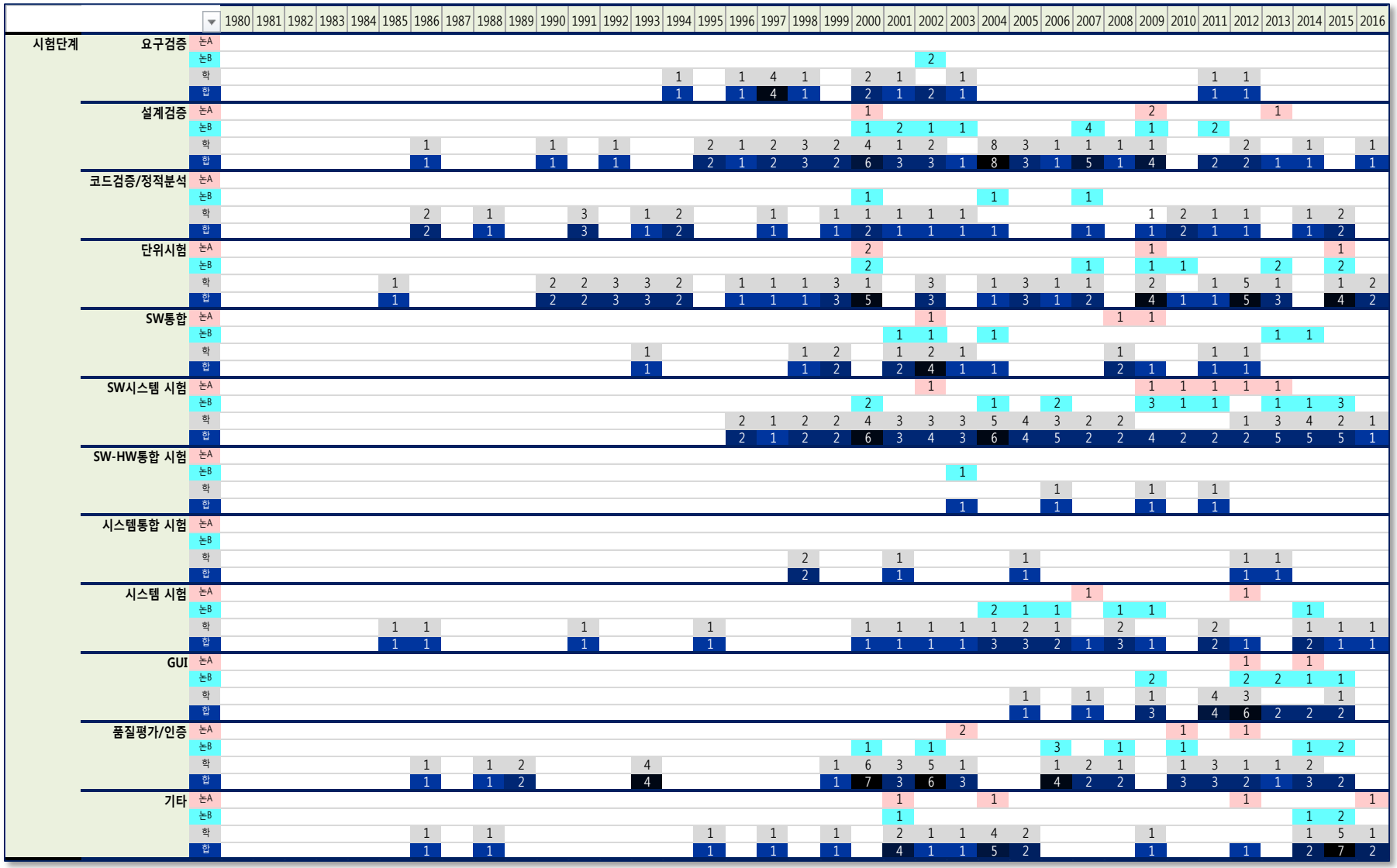
시험 단계



" 주의: 단기간에 조사하는 관계로, 누락된 논문이 있을 수 있음을; 결과가 다소 차이가 있음을 양해 부탁 드립니다 "

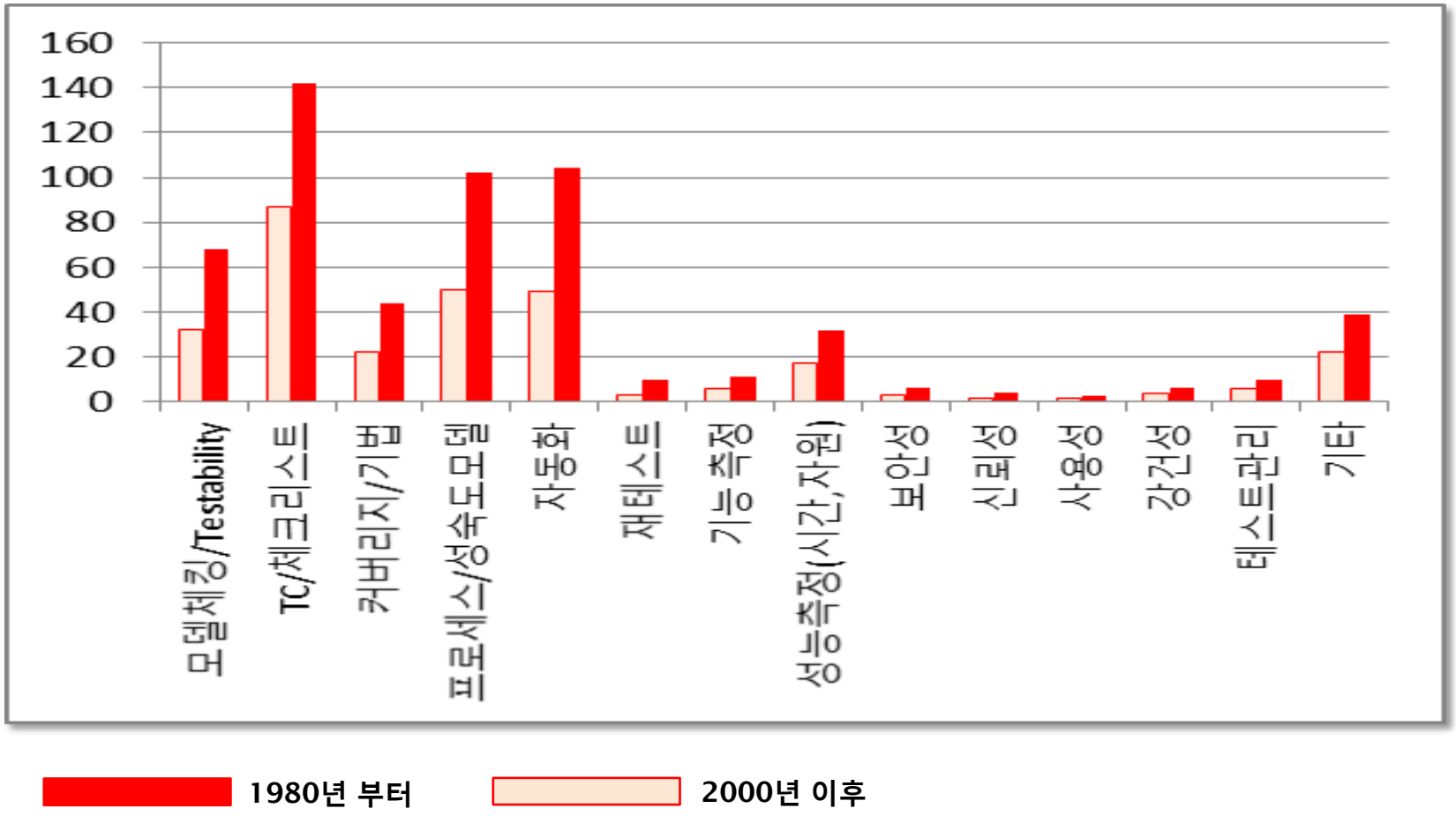
경향_시험단계

논문지A: 컴퓨팅의 실제	논문지B: 소프트웨어 및 응용	정보과학회 학술대회	합



" 주의: 단기간에 조사하는 관계로, 누락된 논문이 있을 수 있음을; 결과가 다소 차이가 있음을 양해 부탁 드립니다 "

연구분야

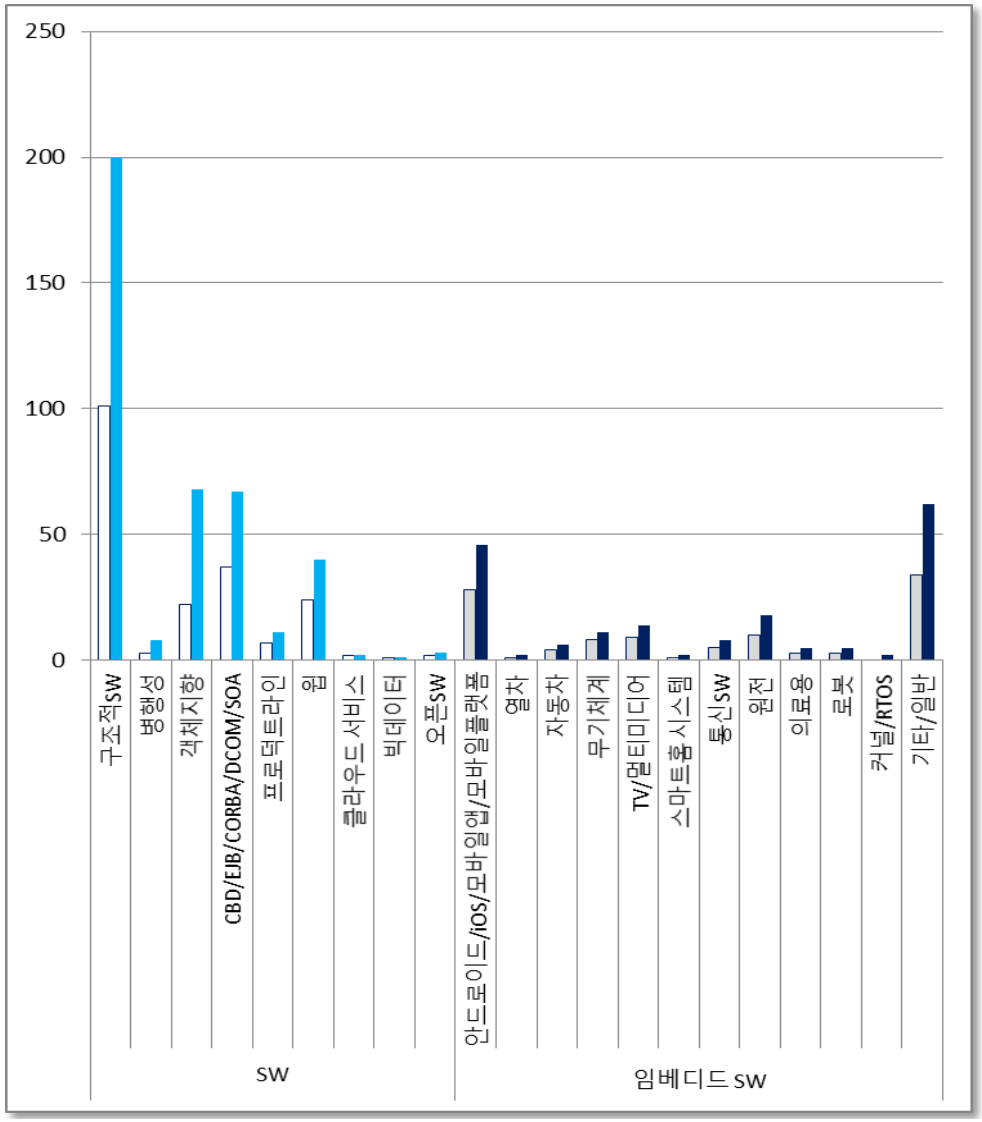


" 주의: 단기간에 조사하는 관계로, 누락된 논문이 있을 수 있음을; 결과가 다소 차이가 있음을 양해 부탁 드립니다 "

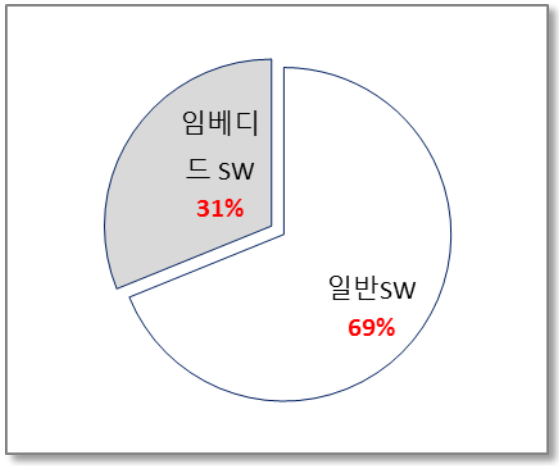
경향_연구분야

"주의: 단기간에 조사하는 관계로, 누락된 논문이 있을 수 있음을; 결과가 다소 차이가 있음을 양해 부탁 드립니다"

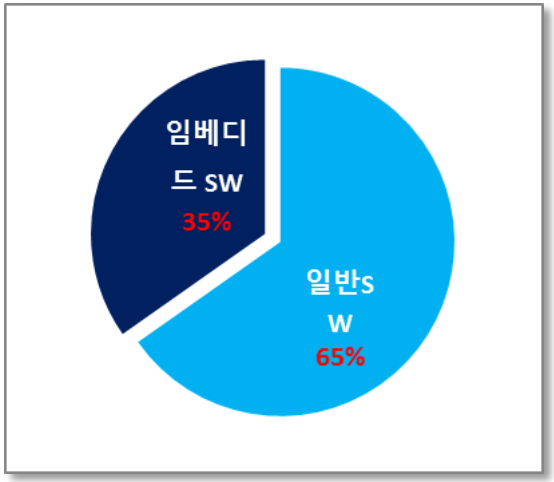
적용 도메인



1980년 부터



2000년 이후

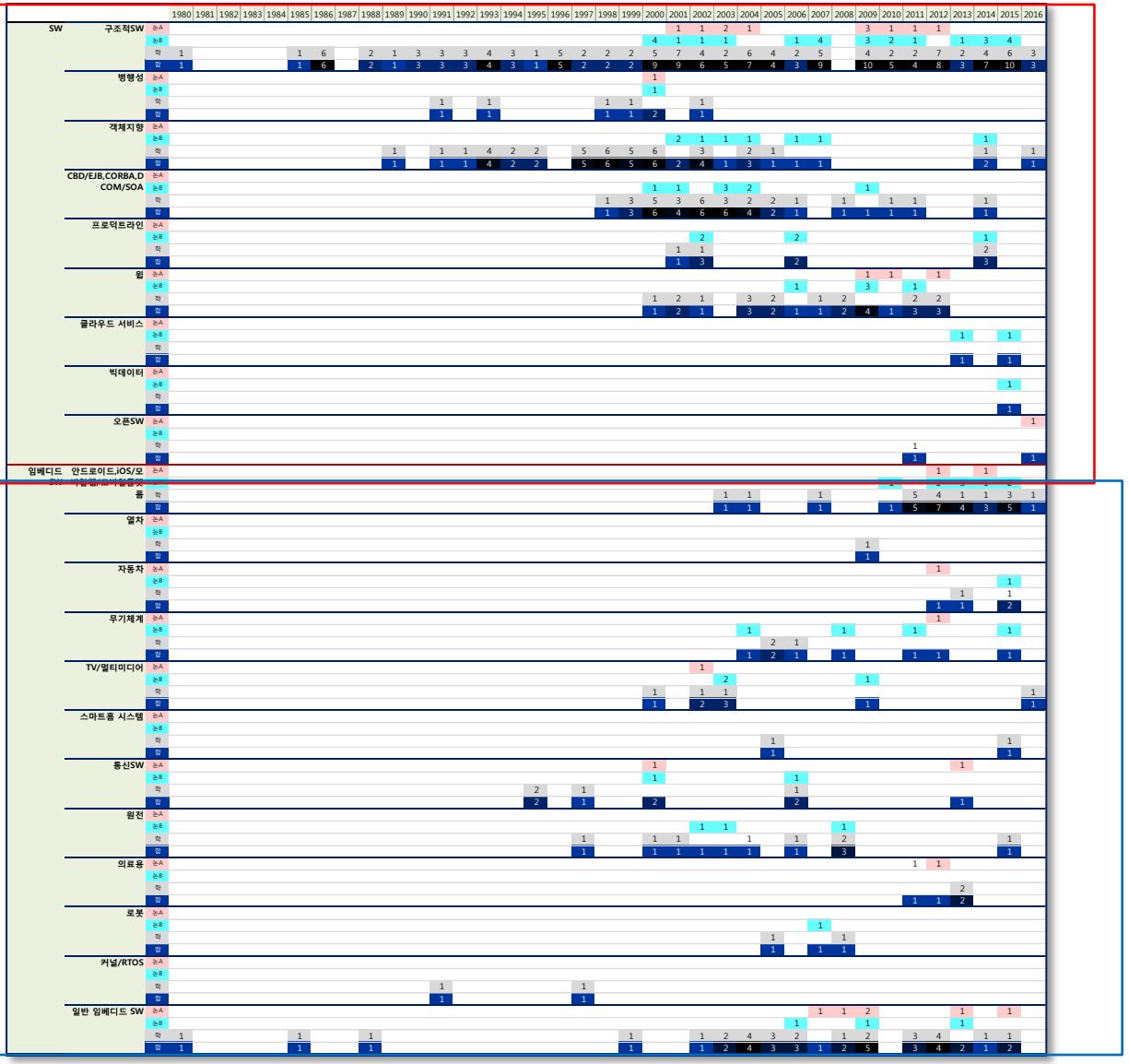


" 주의: 단기간에 조사하는 관계로, 누락된 논문이 있을 수 있음을; 결과가 다소 차이가 있음을 양해 부탁 드립니다 "

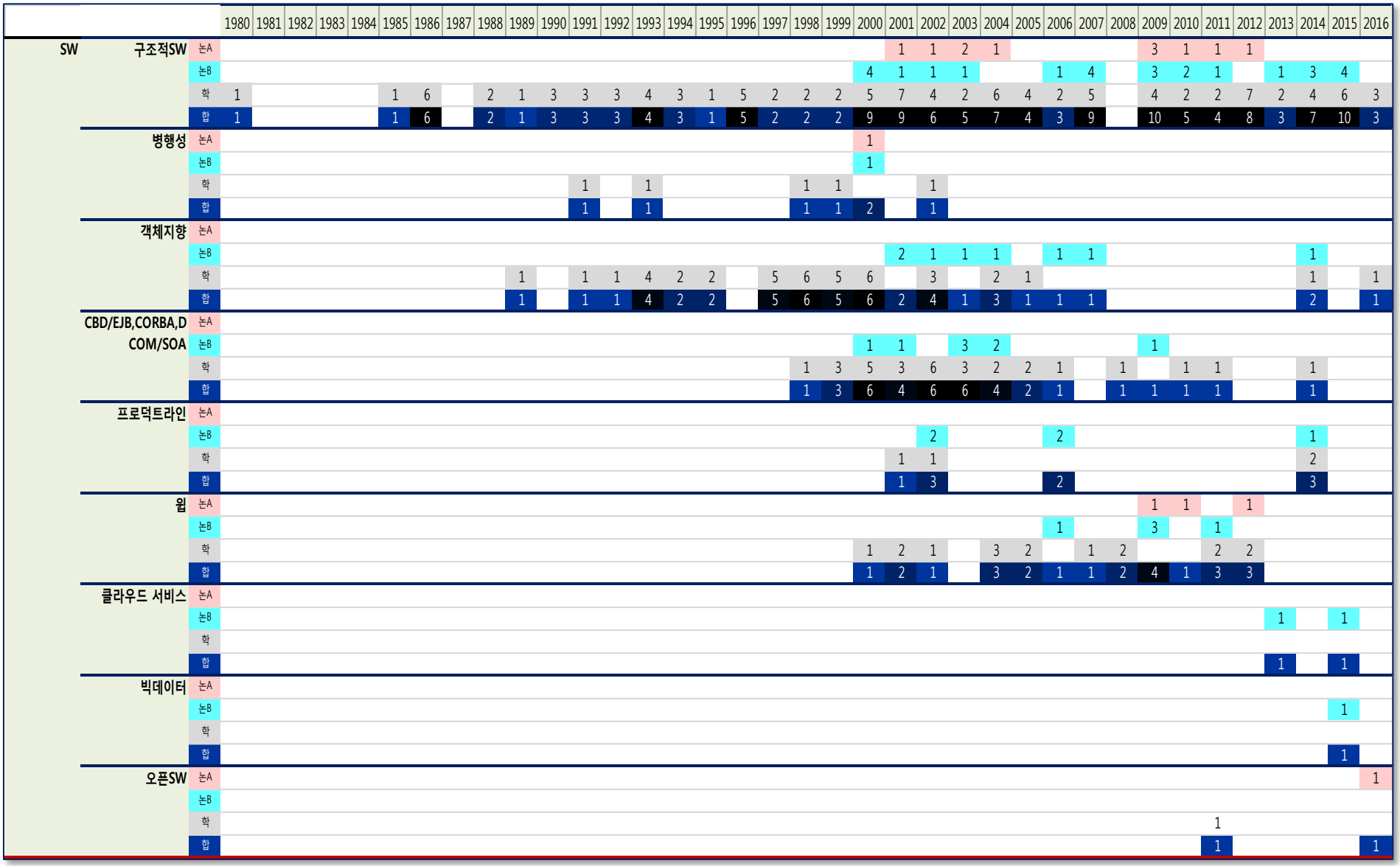
경향_적용도메인

일반 SW

임베디드 SW



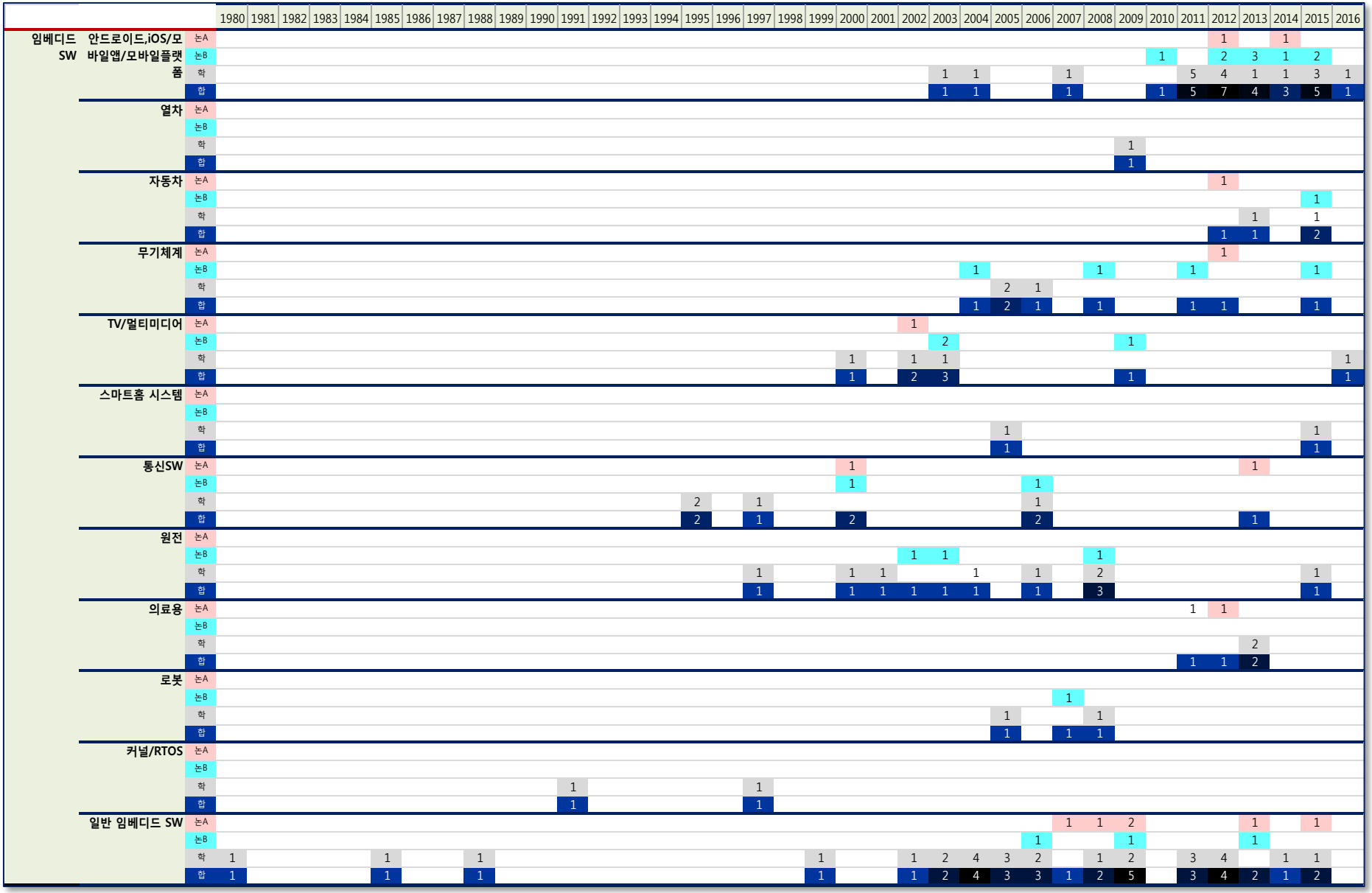
일반 SW



" 주의: 단기간에 조사하는 관계로, 누락된 논문이 있을 수 있음을; 결과가 다소 차이가 있음을 양해 부탁 드립니다 "

임베디드 SW

논문지A: 컴퓨팅의 실제 | 논문지B: 소프트웨어 및 응용 | 정보과학회 학술대회 | 합



동향: 국내 산업체

생략

Discussion: 나아갈 방향

SSARC

SW 안전성 보증 프로세스

상명대학교 SANGMYUNG UNIVERSITY

소프트웨어 안전성 보증 연구센터
[SSARC : *Software Safety Assurance Research Center*]
센터장 : 한혁수 교수
han.hyuksoo@gmail.com
hshan@smu.ac.kr

[목 차]

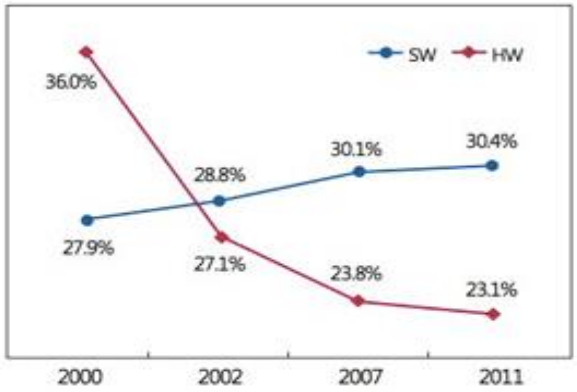
1. SW 안전성

SW 비중 증가

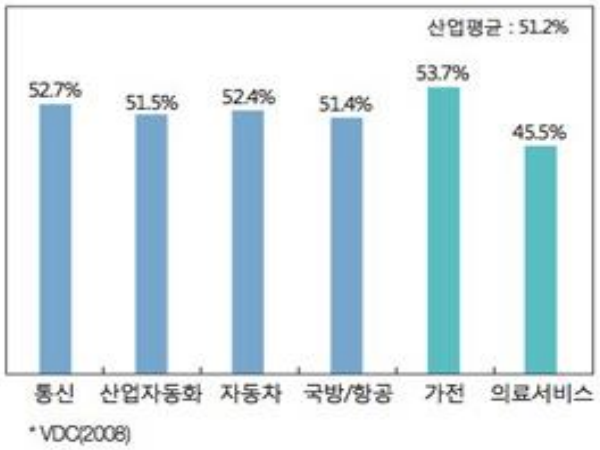
[컴퓨터의 빠른 속도] + [SW의 유연성(flexibility)] = [**SW 중심 사회**]

SW 중심 사회

: SW가 혁신과 성장, 가치창출의 중심이 되고 개인 기업, 국가의 경쟁력을 좌우하는 사회



세계 IT 시장 내 SW 비중 증가



업종별 SW 비중 증가

SW에 의한 제어

인프라	교 통	생 활	재난관리
• 원자력발전소제어시스템 • 전력관리시스템 • 배전자동화시스템 • 댐수문관측시스템 • 수도오존공정시스템 • 지역난방제어시스템	• 열차자동정지시스템 • 항공관제시스템 • 고속도로교통정보시스템 • 선박교통관제시스템 • 선로전환기제어시스템 • 철도운영종합관제시스템	• 혈액관리시스템 • 출입국관리정보시스템 • 신종감염병대응시스템 • 식품이력추적관리시스템 • 국가환경종합시스템 • 보호관찰무인정보시스템	• 종합홍수예보시스템 • 재난방송시스템 • 재난상황전파시스템 • 산불상황관제시스템 • 국가재난관리정보시스템 • 긴급구조시스템

출처) 한국정보통신기술협회(TTA). "SW 안전진단 및 컨설팅." (2015)

SW 오류로 인한 피해

1991 MIM-104 Patriot Missile Failure
*due to "...**rounding error**"*

1996 Explosion of Ariane 5
*due to "...**conversion of a 64 bit integer into a 16 bit** signed integer lead to an overflow..."*

1998 Loss of Mars Climate Orbiter
*due to "...mix-up between **pounds and kilogram**..."*

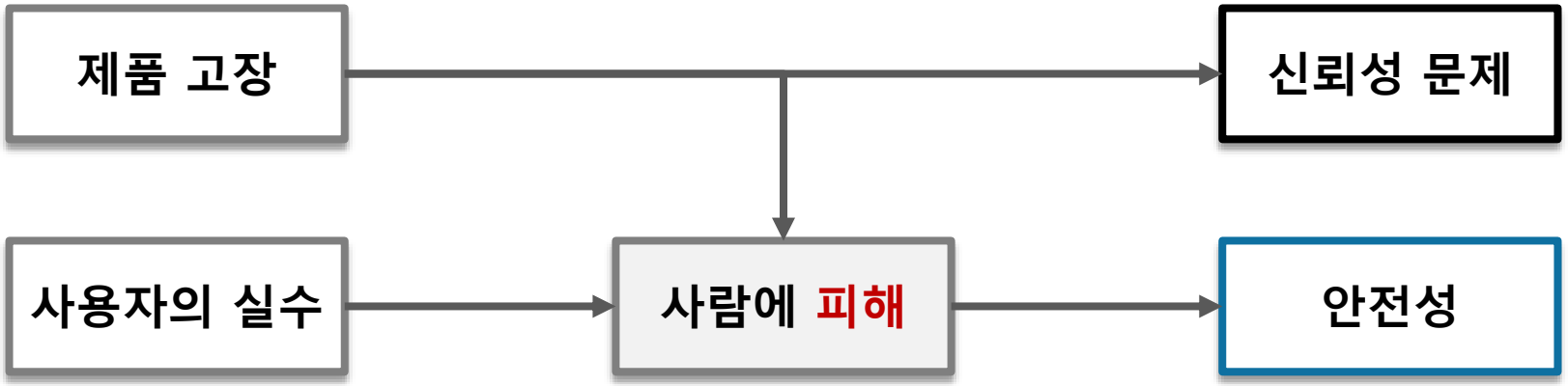
1999 USS Yorktown dead in the water
*due to "...**input and Division by '0'.** „ X / 0 = undefined..."*

SW 안전성(Safety)?

출처) http://discussworldthing.blogspot.kr/2012/01/software-bugs-causing-catastrophic.html

안전성(Safety)과 신뢰성(Reliability)

- **안전성(Safety)은 제품에 문제가 발생했을 때 인명 피해를 최소화(Risk Reduction)하는 것**
- **신뢰성(Reliability)의 목표는 특정 기간 내(5년, 10년 등)에 제품 고장률을 낮추는 것**
  - 시스템/부품이 주어진 임무를 완수할 확률과 수명
- **따라서 신뢰성은 안전성을 확보하기 위한 요소**



안전성(Safety)의 정의

The state of being “Safe“

[IEC61508] “Freedom from unacceptable risk”

SW 안전성(Safety)과 품질(Quality)

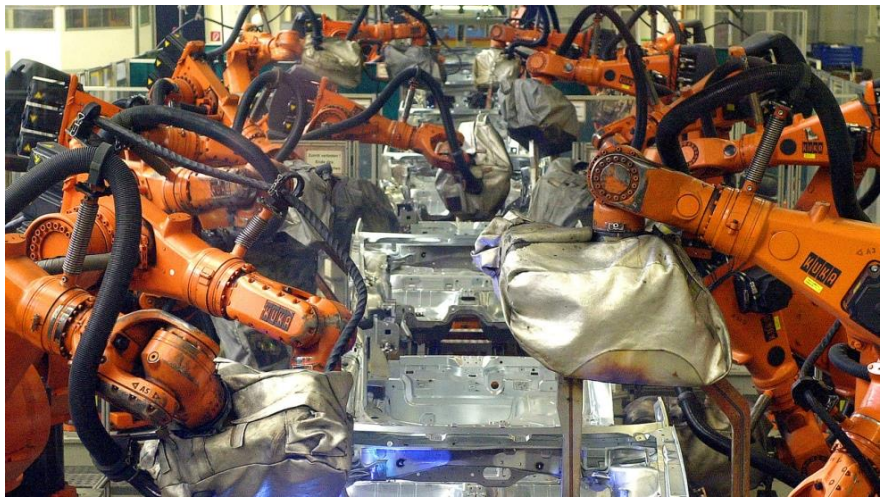
SW Safety is not only about reducing error rates

Dr. Michael F. Siok, PE, ESEP
Lockheed Martin Aeronautics Company

[Safety]
: Mandatory Requirements

[Quality]
: Can be Business Issue

獨 폭스바겐 공장서
'로봇 살인' 사고 발생…
근무 중 직원 1명 사망



[Robot kills man at Volkswagen plant in Germany]

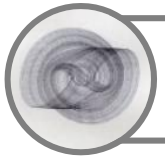
출처) http://m.motorgraph.com/news/articleView.html?idxno=7030

SW 결함에 대한 우려

SW 규모와 복잡도가 증가할수록 결함 확률이 높아짐

[SW의 특징]

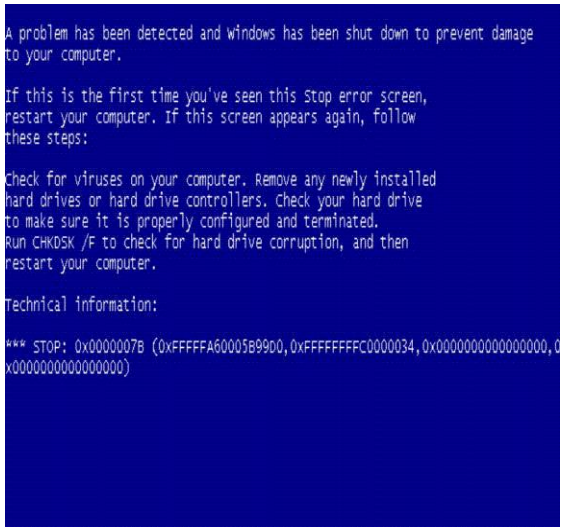
사람 중심의(Human Intensive) 작업



물리적인 형태가 없는
무형의 **논리적인 요소(Invisibility)**

비 선형(Non Linearity)의 복잡한 구조

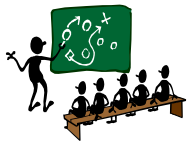
이진수 체계



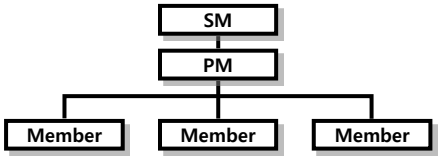
결함(Defects) 발생

SW 개발의 대형화

수백 명의 개발자
- 의사소통 및 상호 협력의 어려움




- 조직 및 팀 구조

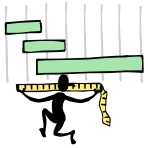


모호하고 복잡한 요구사항
- 수백 페이지의 요구사항

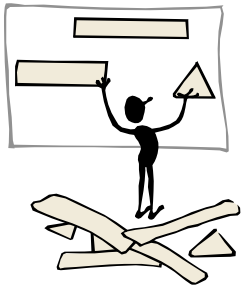

- 빈번한 요구사항의 변화


오랜 개발 시간
- 기간, 공수, 비용의 산정,

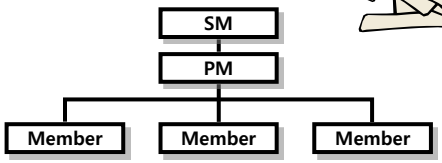

- 진도 및 Risk관리




천만 줄이 넘는 코드
- 통합의 어려움



- 테스팅



SW의 '안전성' 보증

- Hazard 요소를 파악하고, 그 원인을 제거하여

  SW 오류로 인한 시스템의 사고를 예방하는 것

Prevention, Elimination, and/or Control of hazards
that may be caused or induced by SW.

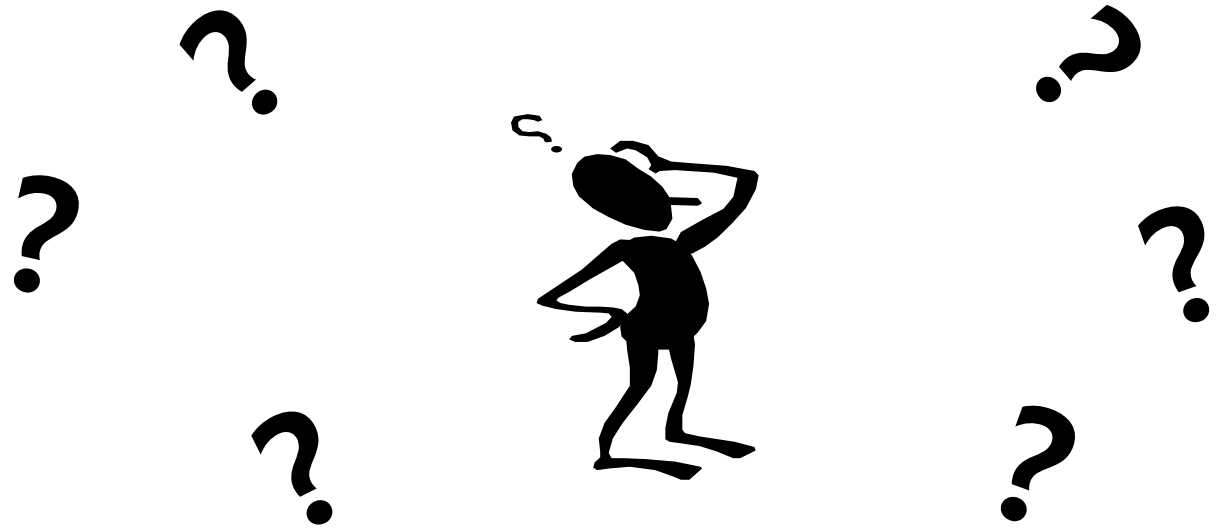
2. SW 안전성 확보를 위한 노력

안전성 확보 방안

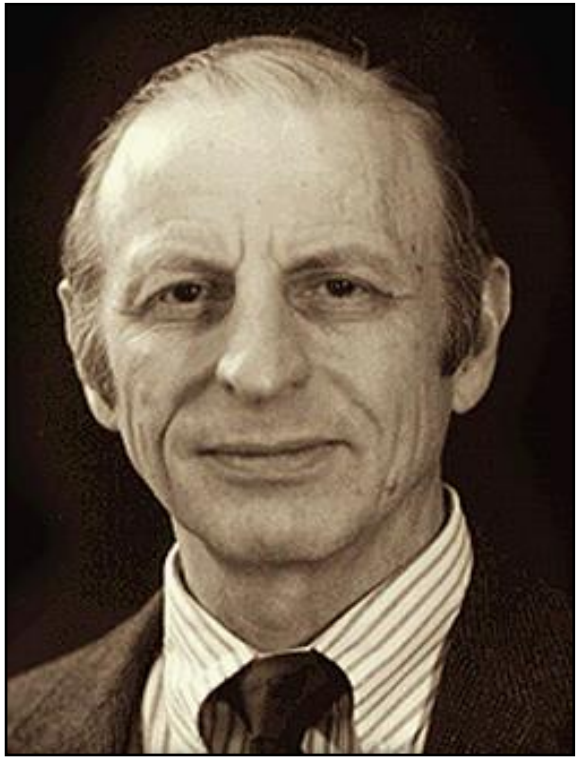
안전성은 어떻게 확보할 수 있는가?

훌륭한 인력만으로 가능한가?

기술만 있으면 되는가?



소프트웨어 개발에서 프로세스의 중요성



"The quality of a software system is governed by
the quality of the process used to develop and evolve it."

— Watts Humphrey

"제품의 품질"은

제품을 개발하고 운영하는

"프로세스의 품질"

에 의해 결정된다.

제품이 어떤 방식으로 개발 되었는가?

품질의 3요소

사람

구성원들의 능력

기술

조직 및 개인의
기술적 역량

BALANCED
INTEGRATION

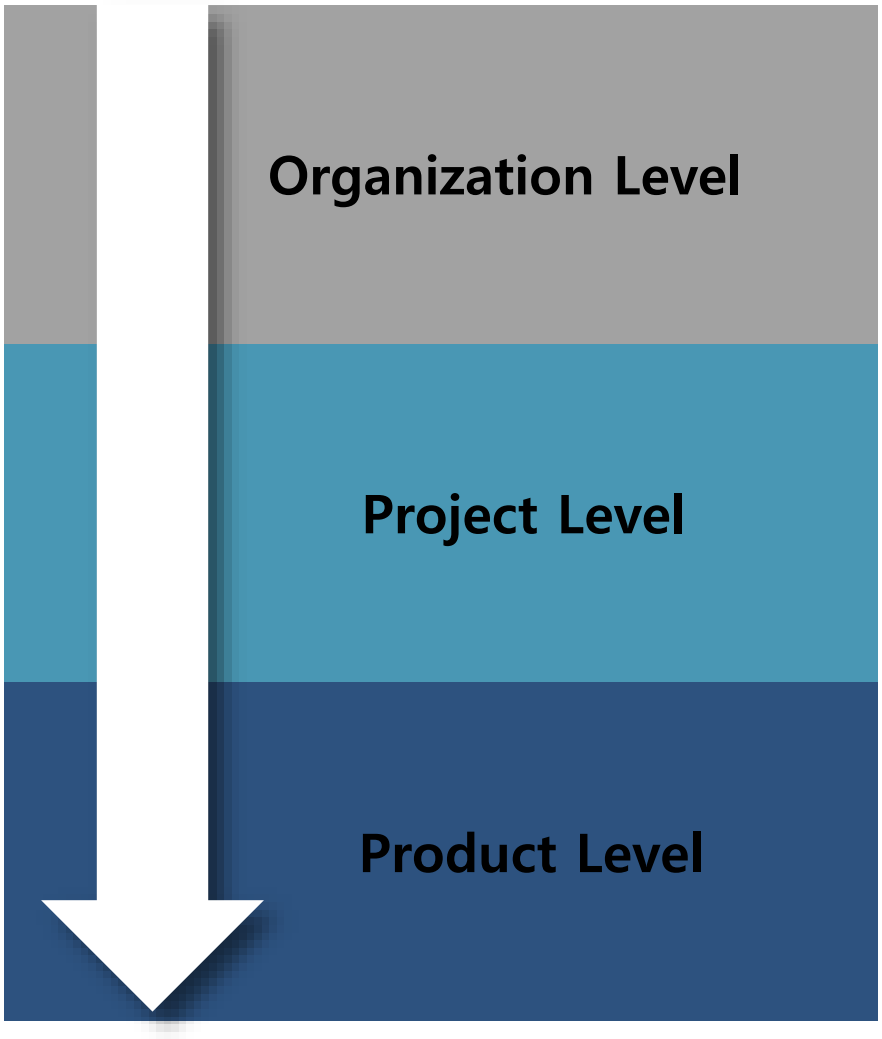
프로세스

일하는 방식

조직 중심과 제품 중심

조직 프로세스 중심

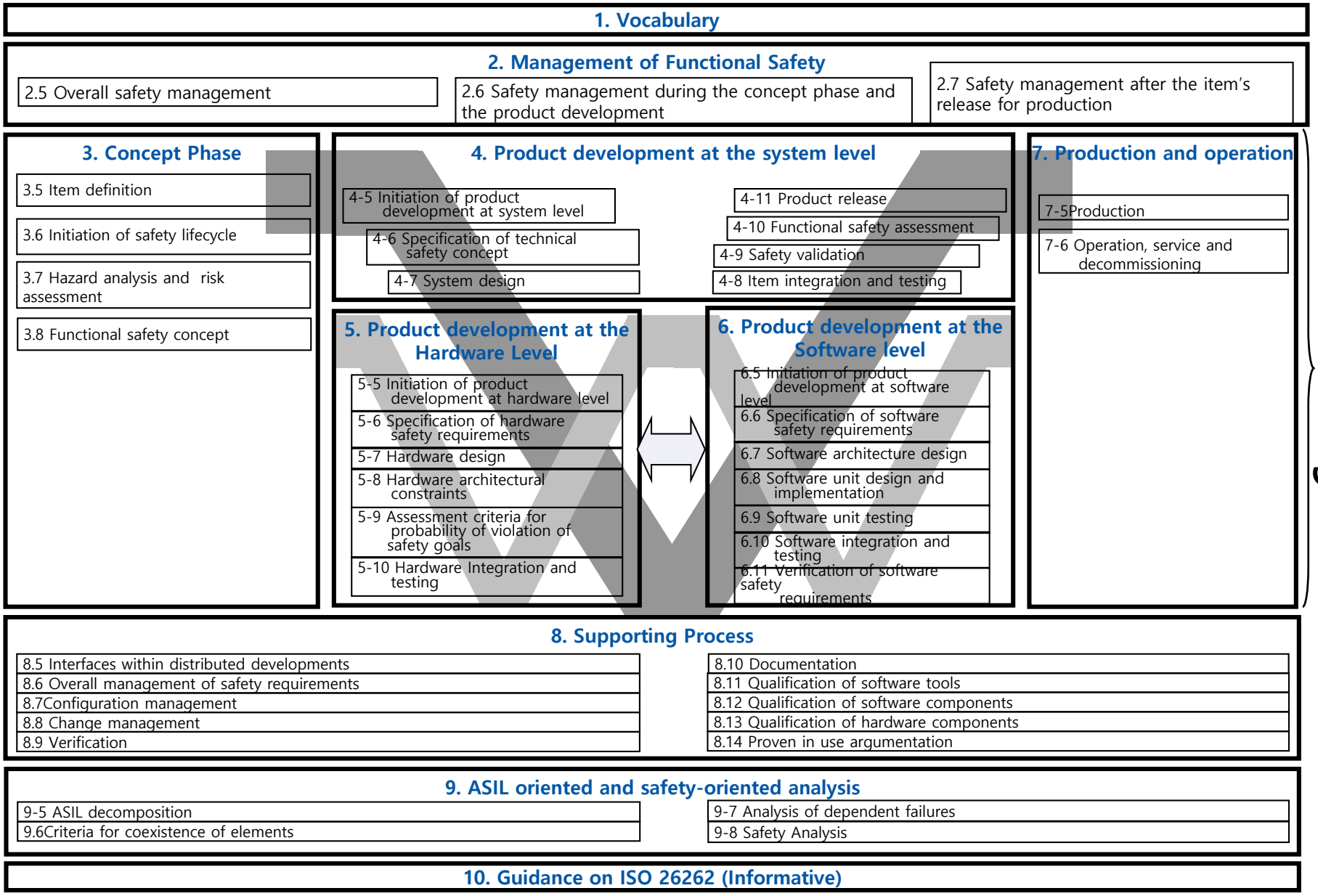
예시) CMMI, A-SPICE



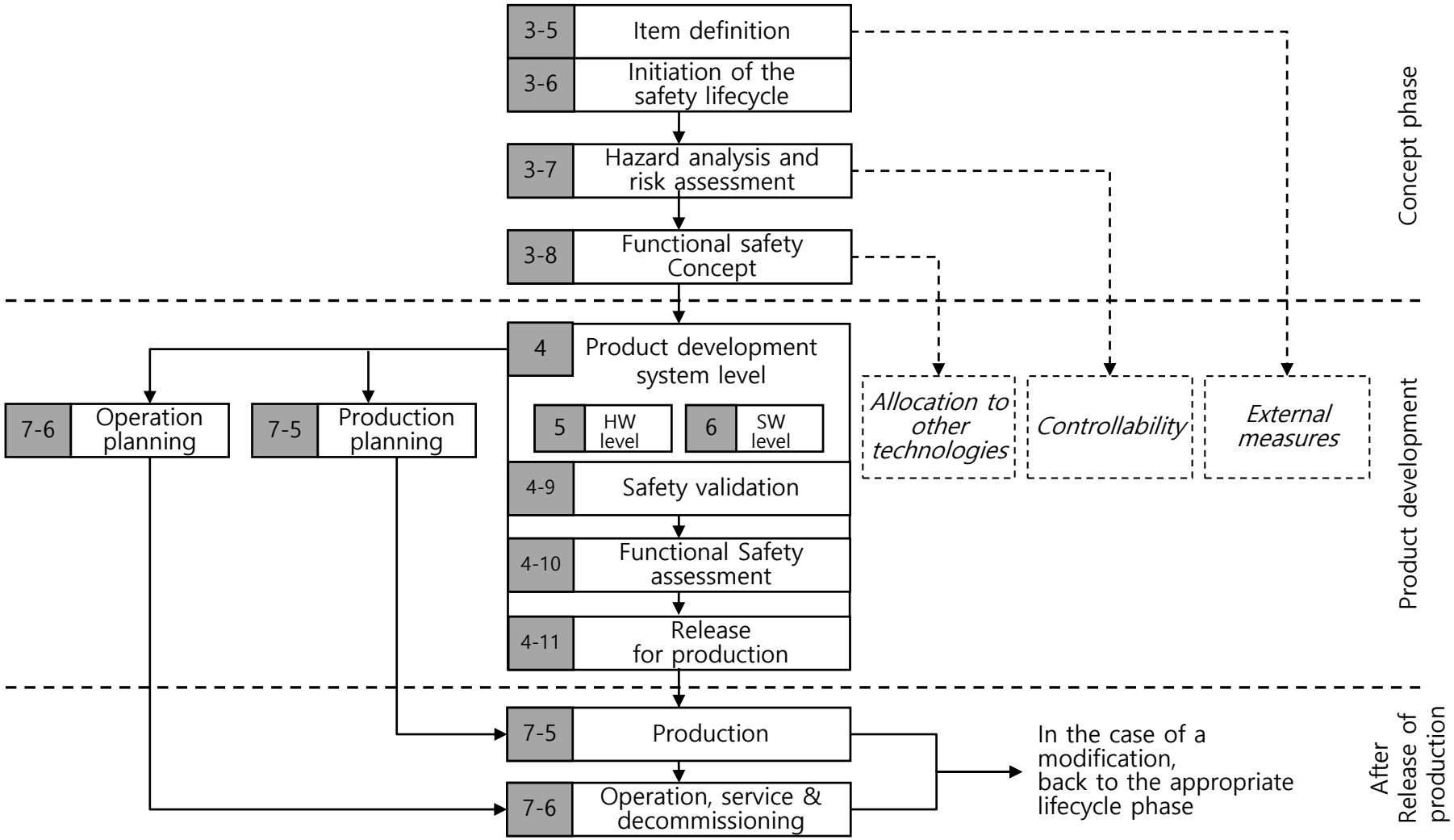
제품 중심

예시) ISO 26262

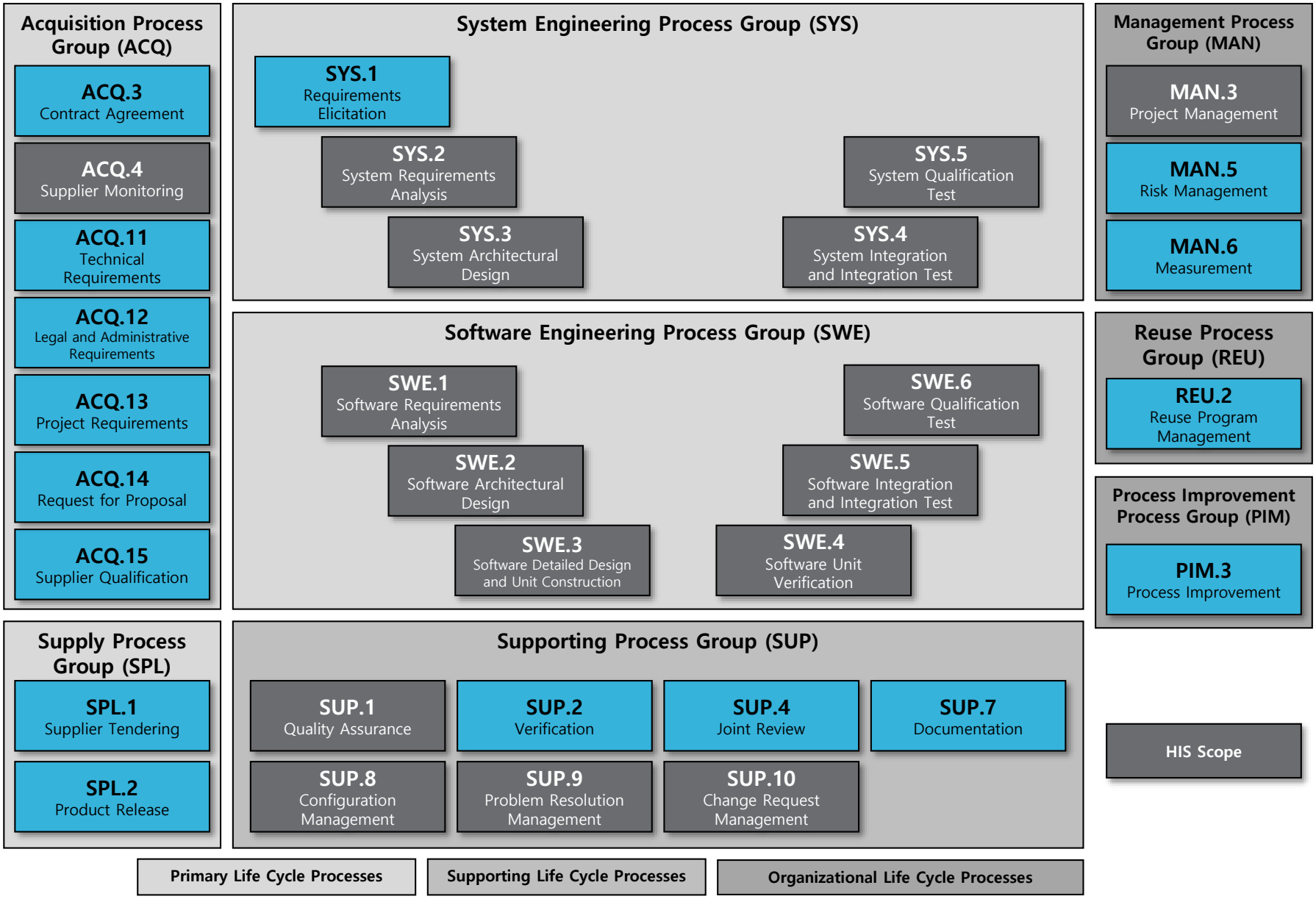
ISO 26262


1. Vocabulary
2. Management of Functional Safety
2.5 Overall safety management
2.6 Safety management during the concept phase and the product development
2.7 Safety management after the item's release for production
3. Concept Phase
3.5 Item definition
3.6 Initiation of safety lifecycle
3.7 Hazard analysis and risk assessment
3.8 Functional safety concept
4. Product development at the system level
4-5 Initiation of product development at system level
4-6 Specification of technical safety concept
4-7 System design
4-11 Product release
4-10 Functional safety assessment
4-9 Safety validation
4-8 Item integration and testing
5. Product development at the Hardware Level
5-5 Initiation of product development at hardware level
5-6 Specification of hardware safety requirements
5-7 Hardware design
5-8 Hardware architectural constraints
5-9 Assessment criteria for probability of violation of safety goals
5-10 Hardware Integration and testing
6. Product development at the Software level
6.5 Initiation of product development at software level
6.6 Specification of software safety requirements
6.7 Software architecture design
6.8 Software unit design and implementation
6.9 Software unit testing
6.10 Software integration and testing
6.11 Verification of software safety requirements
7. Production and operation
7-5Production
7-6 Operation, service and decommissioning
Core processes
8. Supporting Process
8.5 Interfaces within distributed developments
8.6 Overall management of safety requirements
8.7Configuration management
8.8 Change management
8.9 Verification
8.10 Documentation
8.11 Qualification of software tools
8.12 Qualification of software components
8.13 Qualification of hardware components
8.14 Proven in use argumentation
9. ASIL oriented and safety-oriented analysis
9-5 ASIL decomposition
9.6Criteria for coexistence of elements
9-7 Analysis of dependent failures
9-8 Safety Analysis
10. Guidance on ISO 26262 (Informative)

ISO 26262 기능안전 생명주기(Safety Lifecycle)

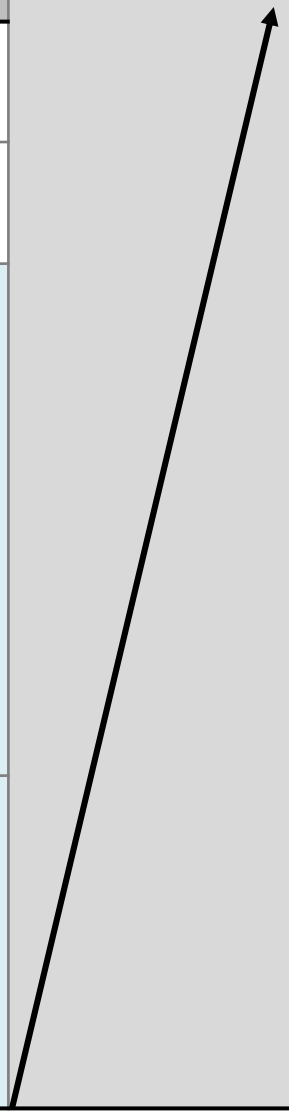


A-SPICE의 PRM(Process Reference Model)



CMMI(Capability Maturity Model Integration)

Level	Focus	Process Areas
5 Optimizing	Continuous Process Improvement	• Organizational Performance Management • Causal Analysis and Resolution
4 Quantitatively Managed	Quantitative Management	• Organizational Process Performance • Quantitative Project Management
3 Defined	Process Standardization	• Organizational Process Focus • Organizational Process Definition • Organizational Training • Risk Management • Requirements Development • Technical Solution • Product Integration • Verification • Validation • Integrated Project Management • Decision Analysis and Resolution
2 Managed	Basic Project Management	• Requirements Management • Project Planning • Project Monitoring and Control • Supplier Agreement Management (N/A) • Measurement and Analysis • Process and Product Quality Assurance • Configuration Management



3. SW 안전성 분석 방법

사고 원인 모델(Accident Causality Model)

- **Accident** (or An unplanned event or sequence of events which mishap)
  - Results in human death or injury, damage to property, or to the environment.

Accident

Cause

Flammable Materials

Oxygen

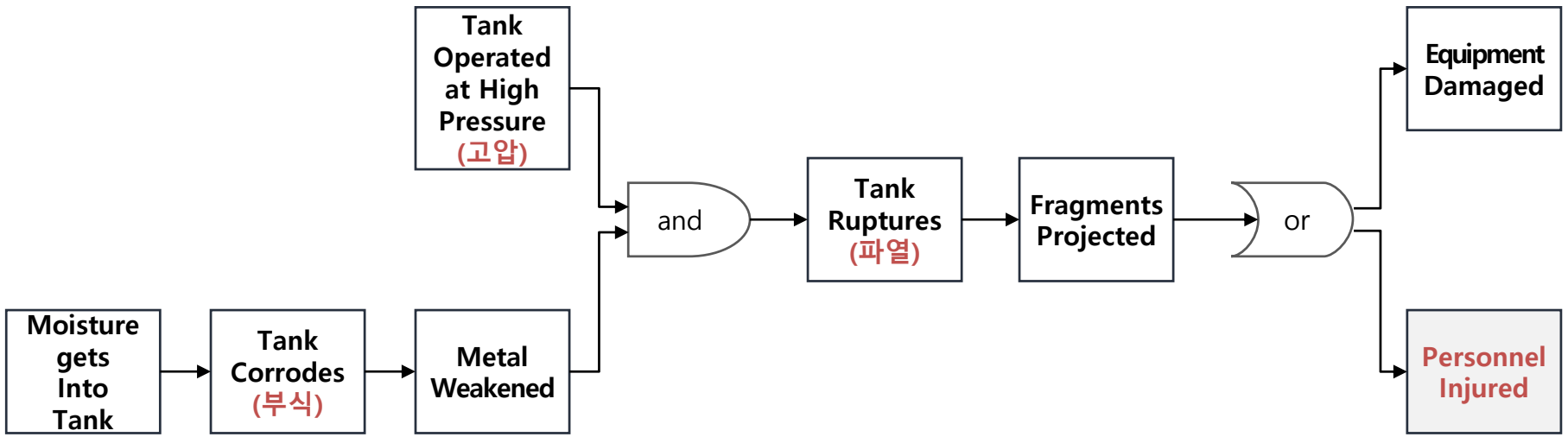
Ignition

Causal/Hazardous Condition

Hazard 분석

- **Harm :** Physical Injury or damage to the health of people
- **Hazard :** **Potential source of Harm**

"Investigating an accident before it happens"



주요 Hazard 분석 방법들

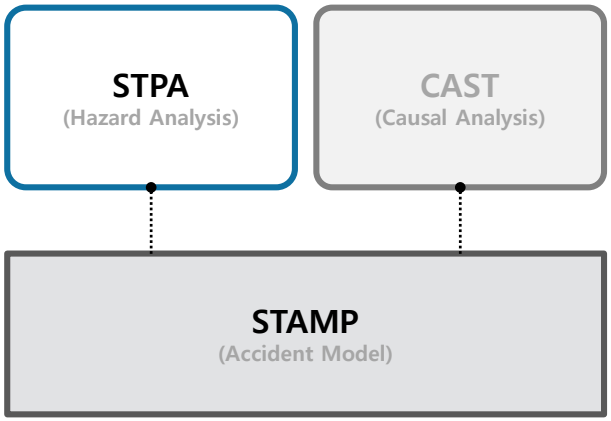
[ISO 26262 적용 가능한 기능안전 분석 기법 종류]

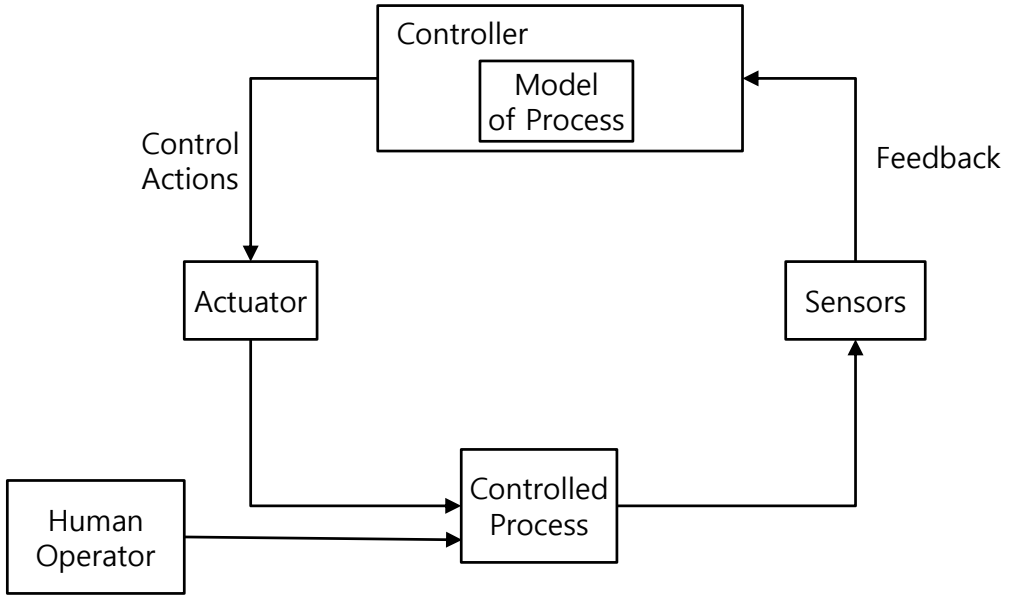
컨셉 단계 Hazard 분석	• Preliminary Hazard List(PHL) • Preliminary Hazard Analysis(PHA) • **Hazard and Operability Studies(HAZOP)** • Fault(or Functional) Hazard Analysis(FHA)
사전 설계 Hazard 분석	• System Hazard Analysis(SHA) • Subsystem Hazard Analysis(SSHA)
상세 설계 Hazard 분석	• **Failure Mode and Effect Analysis(FMEA)** • **Fault Tree Analysis(FTA)** • Event Tree Analysis(ETA) • Energy Trace and Barrier Analysis(ETBA) • Sneak Circuit Analysis(SCA) • Software Hazard Analysis(SWHA) • Common Cause Failure Analysis(CCFA) • Cause and Effect Analysis(CEA) / Cause Consequence Analysis(CCA)
운용/보건 설계 Hazard 분석	• Operating & Support Hazard Analysis (O&SHA)

- **FMEA**
  (Failure Mode and Effect Analysis)
- **FTA**
  (Fault Tree Analysis)
- **HAZOP**
  (Hazard and Operability)

STPA(System Theoretic Process Analysis)

- **MIT의 STAMP 모델에 기반한 Hazard 분석 방법**
  - 전체적인 사건 프로세스를 망라하는 사고 시나리오 파악을 목적으로 함
  - 시스템 이론에 따라 사고가 컴포넌트들 간의 상호작용에 의해 발생한다고 가정하고 컴포넌트들 간의 **Unsafe Control Action**을 통해 Hazard를 분석





4. SW 안전성 보증 프로세스

SW 안전성 보증 프로세스

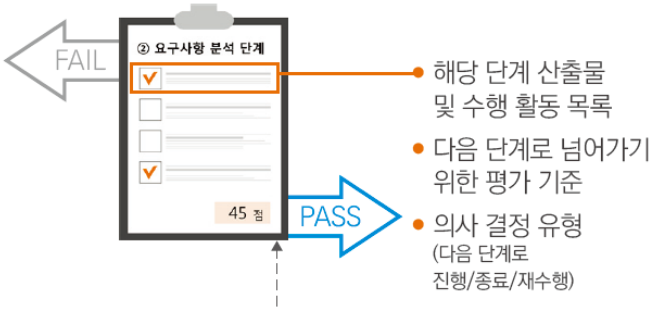
요구사항

❶ 안전성 분석

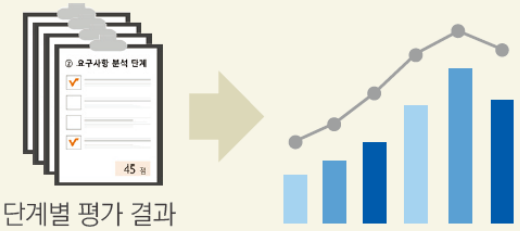
SW-FMEA
SW-Failure Mode Effects Analysis
STPA
System Theoretic Process Analysis
CAST
Causal Analysis based on STAMP

Safety
안전성 요구사항

❸ 안전성 Gate : 안전성 감사 활동



- 해당 단계 산출물 및 수행 활동 목록
- 다음 단계로 넘어가기 위한 평가 기준
- 의사 결정 유형 (다음 단계로 진행/종료/재수행)



단계별 평가 결과

❷ 안전성 개발 프로세스

❺ 안전성 Verification

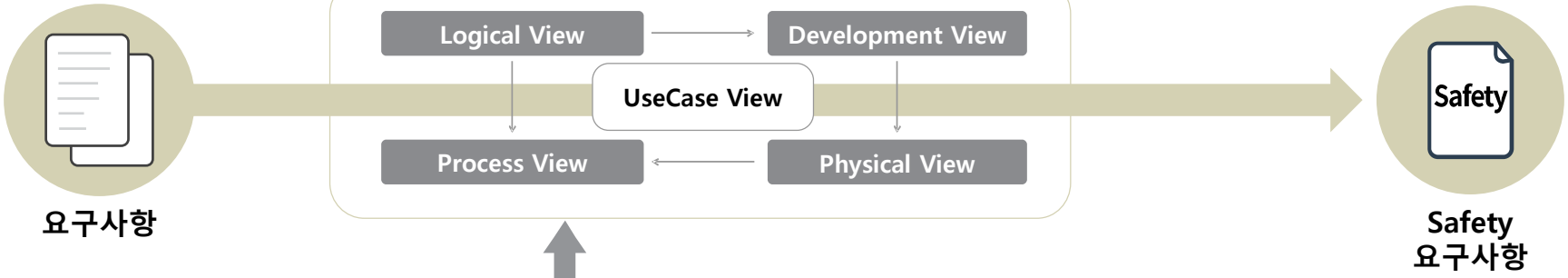
❹ Visualization : 안전성 지원 도구 및 활동

Redmine	git	checkstyle	nsiq	CPD	VectorCAST	Jenkins
JIRA	Subversion	cppcheck	LDRA	QAC		VMWare
		Polyspace	Codesonar			

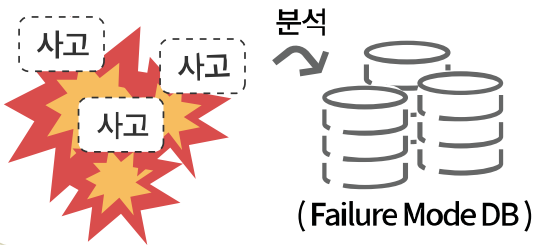
상용도구 오픈소스 도구

① 안전성 분석

2 4+1 View 모델 기반 시스템 기능 및 구조 파악



1 사고분석



3 FMEA 기반 안전성 분석

SRS#	UC#	Failure mode	Guide Word	Root Cause	Corrective Action
10	UC-010	Faulty timing	Too early or too late	열차, 선로, 관제센터가 오래되거나 반복된 정보를 전달하는 것에 대한 감지 및 처리 방안 부재	열차, 선로, 관제센터가 생성/처리/전송하는 데이터의 실시간 기준 명시
...	...	...	...	...	...

(FMEA Sheet)

---

1 사고분석

FMEA를 활용한 과거 사고 원인 분석 및 Failure Mode DB 축적

2 4+1 View 모델 기반 시스템 기능 및 구조 파악

UML Diagram을 활용한 대상 시스템이 구조 및 상위 기능/상태 파악

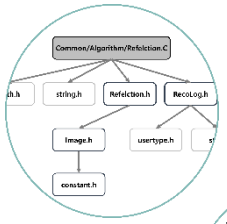
3 FMEA 기반 안전성 분석 수행

UML Diagram, Failure Mode DB를 활용하여 FMEA 기반의 안전성 분석 수행

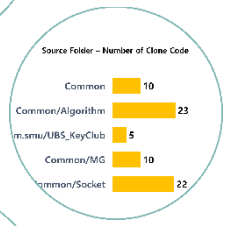
② 안전성 개발 프로세스

소스코드 품질 향상

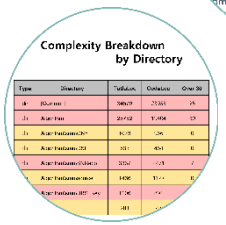
리팩토링(Refactoring)기법 적용을 통한 소스코드 품질 향상 및 개발 역량 강화



헤더/함수 의존성 이상



Clone Code
(중복 코드)



순환 복잡도 위반

C언어
리팩토링 기법

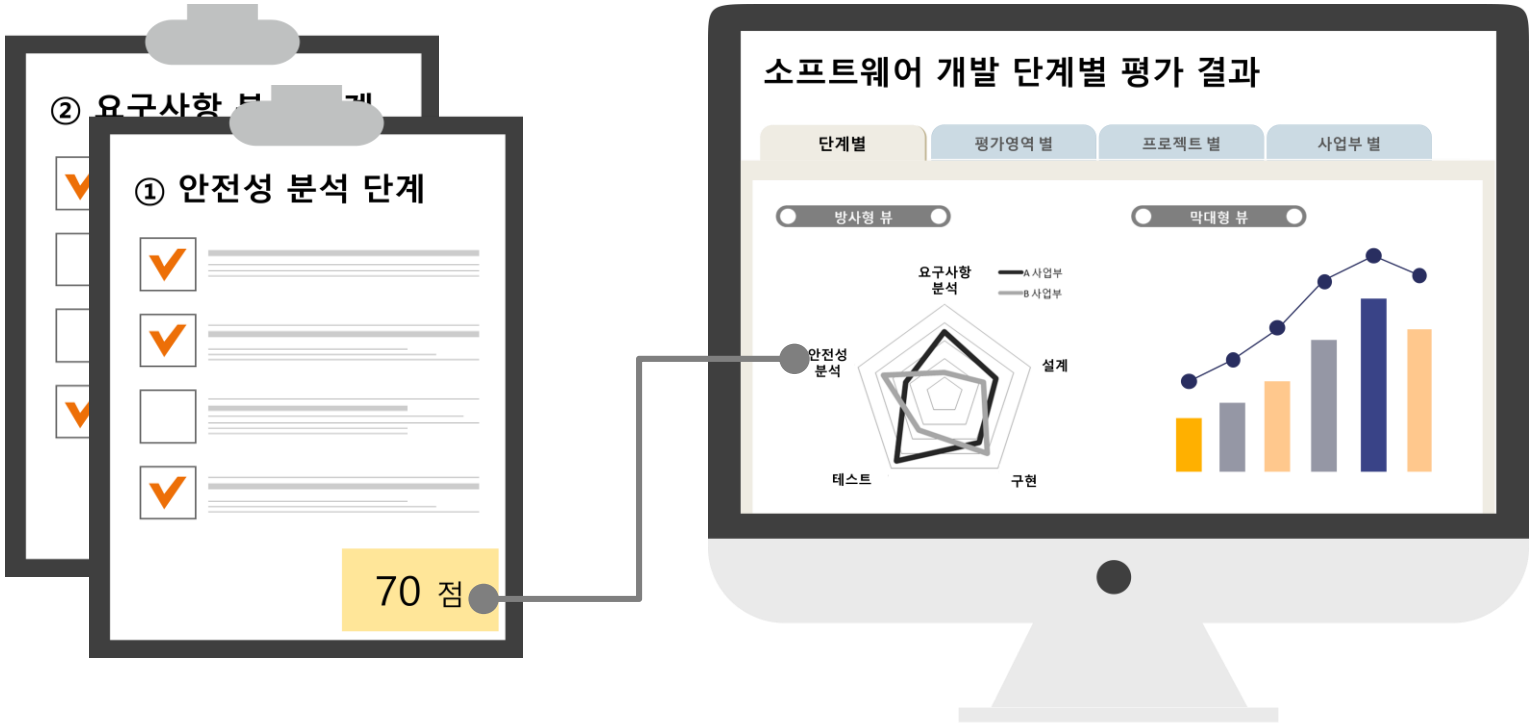
- 함수 추출
- 파일 추출
- 전역 변수 개선
- 헤더 참조 개선
- 중복 제거
- 주석 표준화
- 코드 컨벤션 표준화

[품질 향상된 C 코드]

③ 안전성 Gate

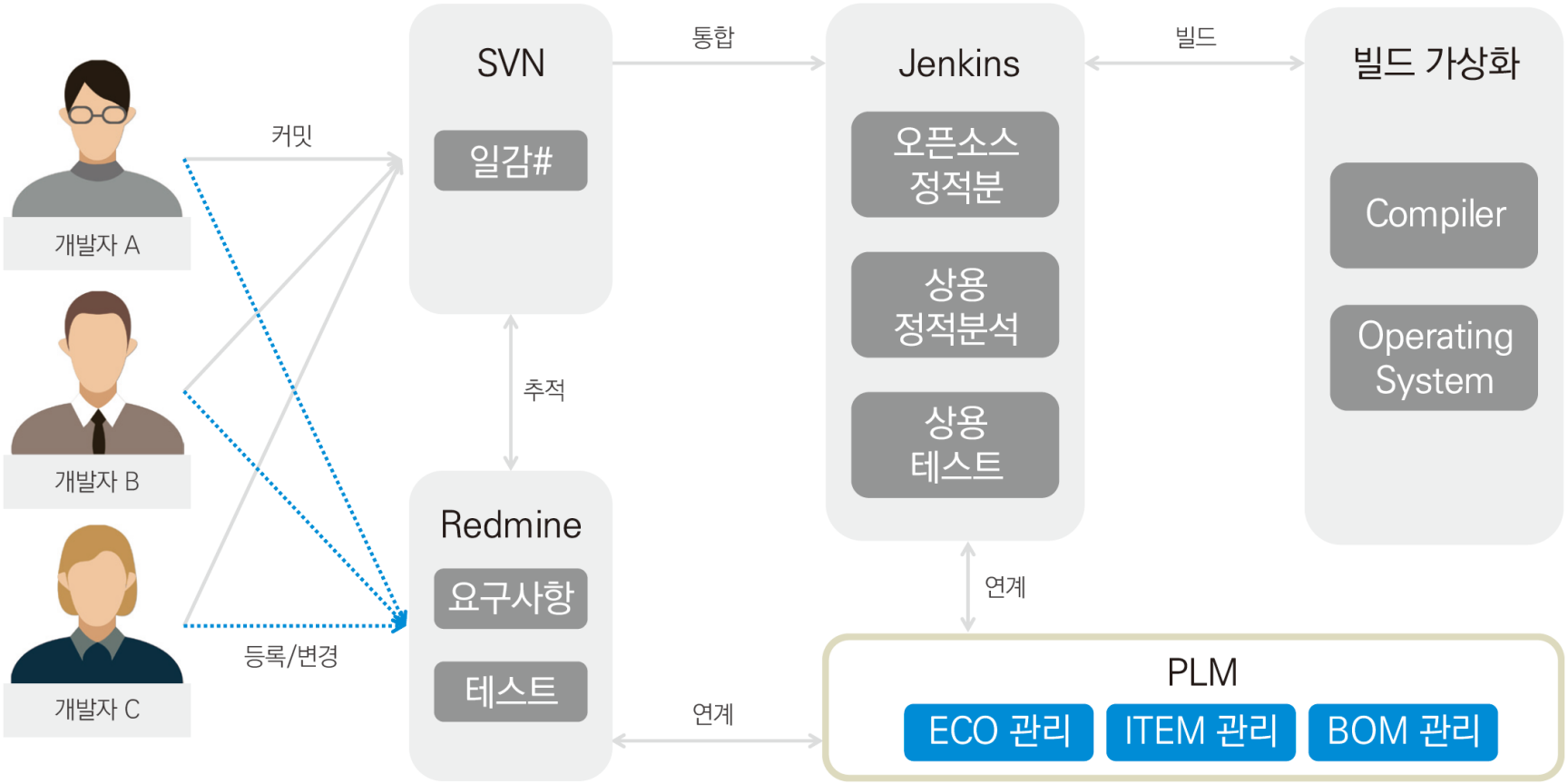
안전성 감사 활동

품질 게이트(Quality Gate) 설치를 통해 SW 품질 상태와 안전성 요구사항 준수 여부를
개발단계 전체에 걸쳐 확인



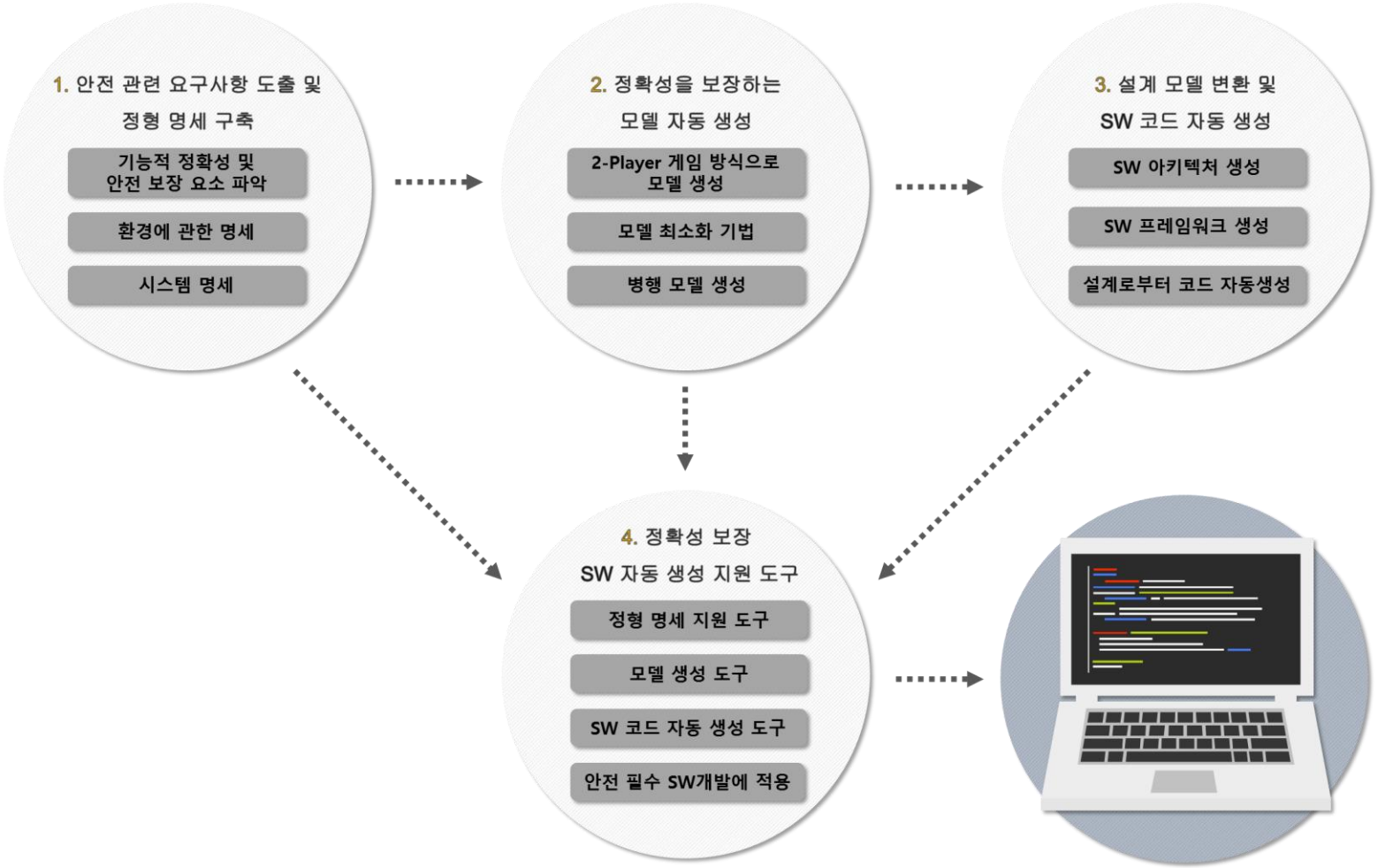
안전성 지원 도구 및 활동

SW 개발 품질의 시각화를 통해 요구사항과 소스코드 추적 및 소스코드 문제점 조기 발견



정확성 기반의 소프트웨어 안전성 확보기술

소프트웨어 정확성 확보를 위해 모델에 기반한 정형 기법과 안전성 테스트 자동화 기술 연구



[결론] 안전 DNA

안전을 최우선으로 인식하는 것이 중요함

안전성은 모든 작업을 수행할 때,
안전을 위한 노력의 결과로 얻어짐

Safety is the sum of repeated efforts day in and day out

Q & A

"It is not enough to see a particular structure underlying a particular problem….This can lead to solving a problem, but it will not change the thinking that produced the problem in the first place" – P. M. Senge 1990

Thank you

정형 기법의 개념 및 역사

2017.06.20

권 기 현 교수

경기대학교 컴퓨터과학과

발표 순서

KGU 경기대학교

- 안식년 소개
- 소프트웨어 공학 소사이어티와의 추억
- 정형 기법의 개념
- 국내 정형 기법의 역사
- Q & A

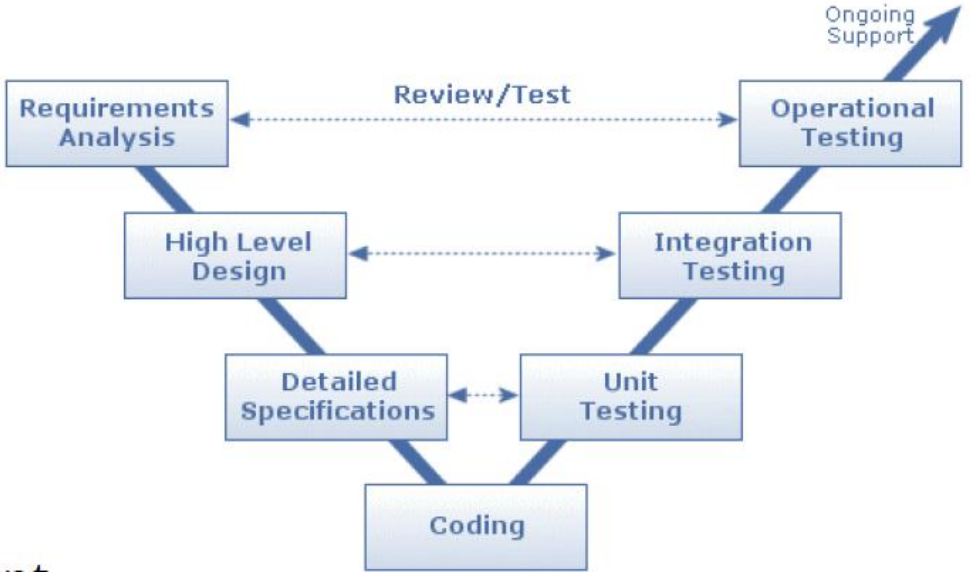
소프트웨어 개발 모델

KGU 경기대학교

- 소프트웨어 개발 절차 및 작성해야 할 문서들을 정의함

- 대표적인 예
  - 폭포수 모델
  - V-모델
  - 애자일 개발
  - 정형 기법

- 모델간의 비교



정형 기법

KGU 경기대학교

- 목표
  - 수학적 증명을 통해서 아예 정확한 소프트웨어를 개발한다.
  - Proof-By-Construction
- 주요 구성 요소
  - 정형 명세
  - 정형 검증
  - 정형 정제
- 구분
  - 중량급 정형 기법: Z, VDM, B, etc.
  - 경량급 정형 기법: LTL synthesis

LTL, Linear Temporal Logic

KGU 경기대학교

- 속성 명세 언어

- 구문
  - 상수: tt, ff
  - 명제 연산자: $\neg,\ \wedge,\ \vee,\ \Rightarrow,\ \Leftrightarrow$
  - 시제 연산자: G, F, X, U, W
  - 관심 이벤트: $AP = \{p, q, \ldots\}$

$$\varphi ::= \text{tt} \mid \text{ff} \mid p \mid$$
$$\neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \Rightarrow \varphi_2 \mid \varphi_1 \Leftrightarrow \varphi_2 \mid$$
$$G\varphi \mid F\varphi \mid X\varphi \mid \varphi_1 U \varphi_2 \mid \varphi_1 W \varphi_2 \mid (\varphi)$$

LTL, Linear Temporal Logic

KGU 경기대학교

- 모델 $\mathcal{M} = (AP, S, R, S_0, L)$
  - AP is the set of atomic propositions.
  - S is the set of states.
  - $R \subseteq S \times S$ is the transition relation.
  - $S_0 \subseteq S$ is the set of initial states.
  - $L: S \to 2^{AP}$ is the labeling function.
    - 2^{AP} is the power set of AP
    - $L(s) \subseteq AP$ for all $s \in S$

- 경로
  - $\pi = s_0 s_1 s_2 \cdots$ 상태들의 나열

LTL, Linear Temporal Logic

KGU 경기대학교

- 의미론
  - 기본 경우
  - $\sigma \vDash \mathrm{tt}$
  - $\sigma \vDash p$ iff $p \in A_0$

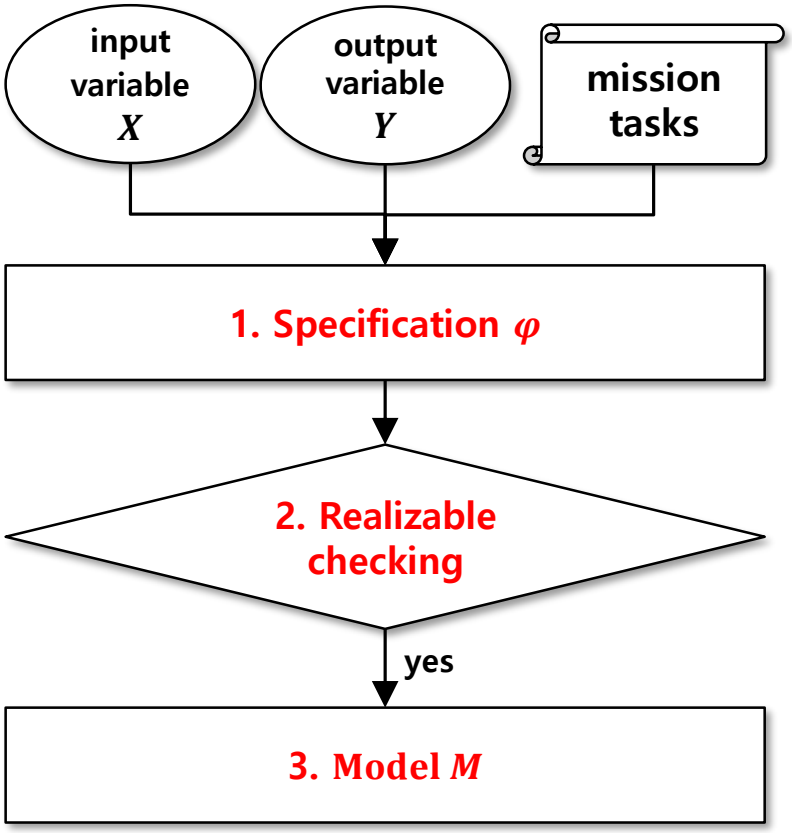
  - 재귀 경우 (명제 연산자)
  - $\sigma \vDash \neg\varphi$ iff $\sigma \nvDash \varphi$
  - $\sigma \vDash \varphi_1 \wedge \varphi_2$ iff $\sigma \vDash \varphi_1$ and $\sigma \vDash \varphi_2$

  - 재귀 경우 (시제 연산자)
  - $\sigma \vDash \bigcirc\varphi$ iff $A_1 A_2 \ldots \vDash \varphi$
  - $\sigma \vDash \varphi_1 U \varphi_2$ iff some $j \geq 0$, $A_j A_{j+1} \ldots \vDash \varphi_2$
    and for all $0 \leq i < j, A_i A_{i+1} \ldots \vDash \varphi_1$

LTL 관련된 검사들

- Satisfiability checking
- Model checking
- Realizability checking

LTL 합성


input variable X
output variable Y
mission tasks
1. Specification φ
2. Realizable checking
yes
3. Model M

LTL 합성 단계

KGU 경기대학교

1. Specification
   - GR(1), a fragment of LTL,
   - Assumption-Guarantee style

$$\varphi = \varphi_e \rightarrow \varphi_s$$

assumptions about environment

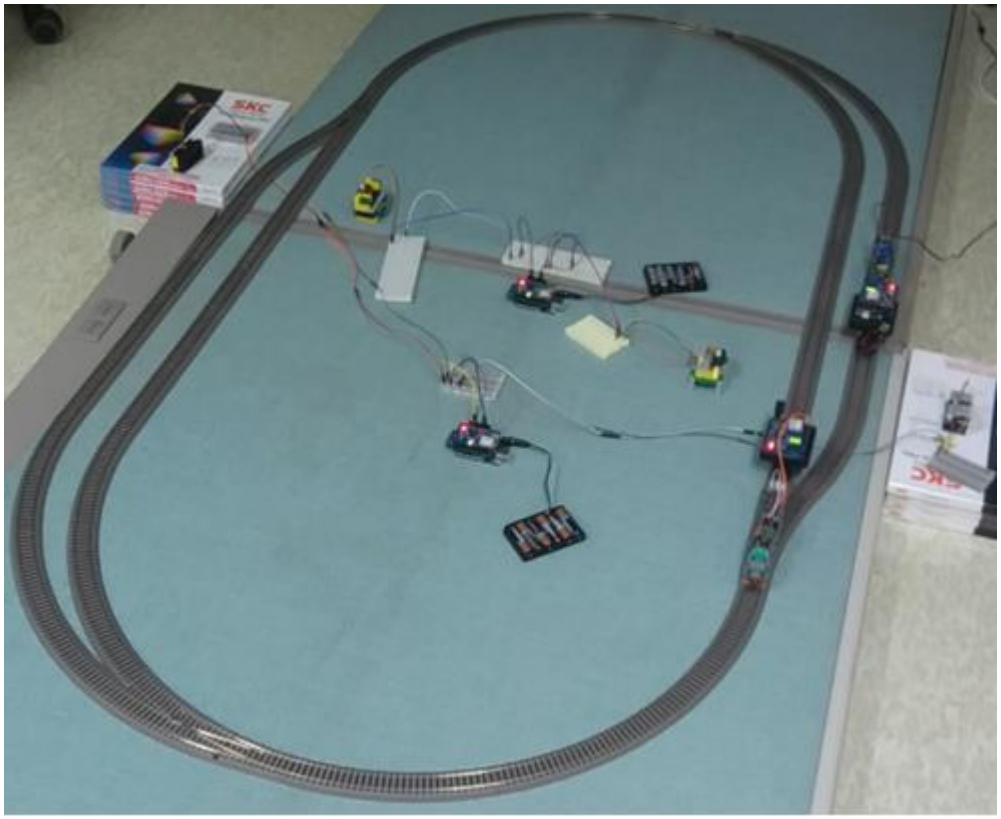
desired system behavior

2. Realizable Checking as a Game
   - 2-player game
   - 상대방의 모든 움직임에 대해 이기면 realizable

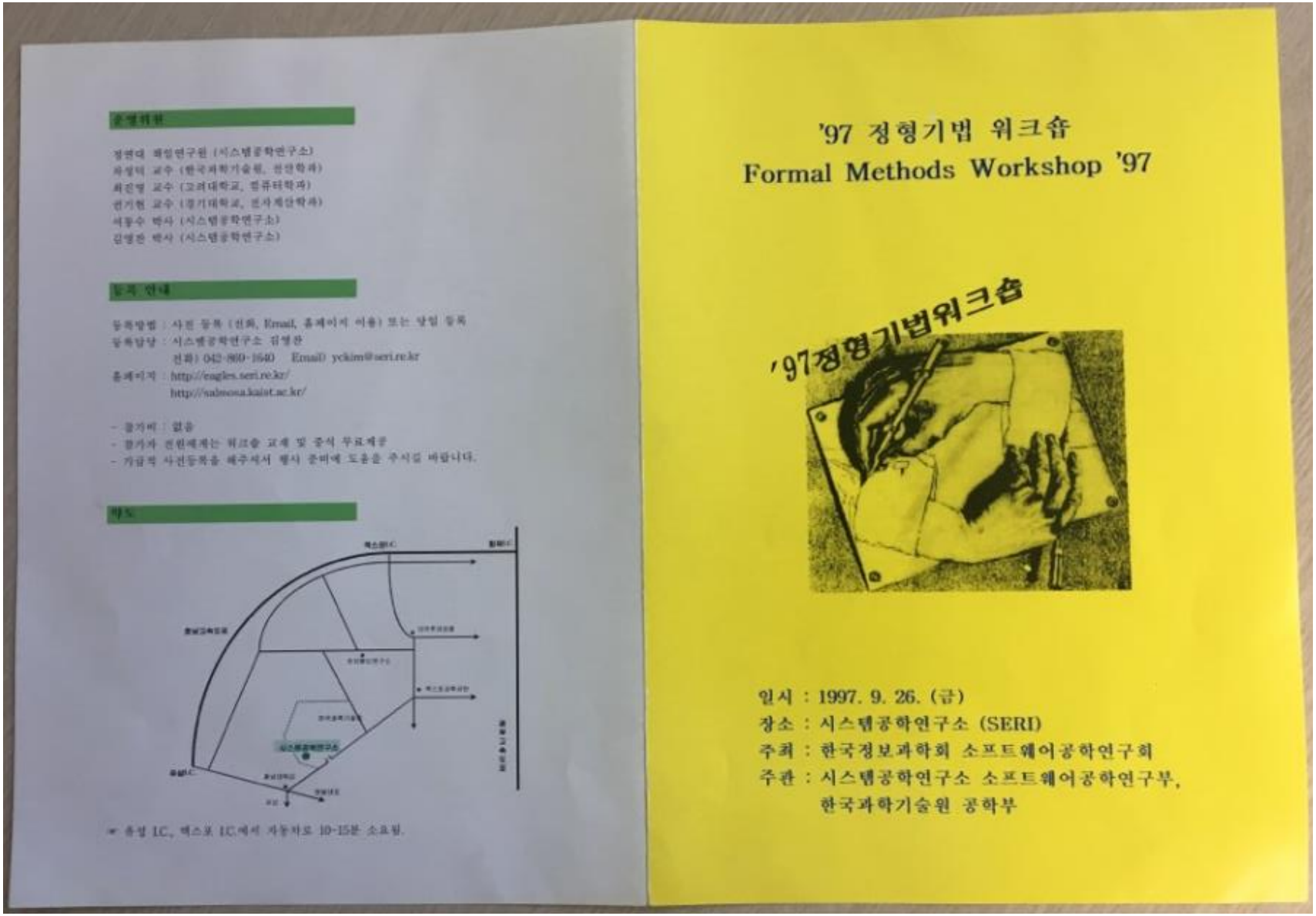
3. Synthesized Model
   - Moore machine (or Transducer)
   - $\mathcal{M} = (Q, X, Y, q_0, \delta, \gamma)$

LTL 합성의 활용

- LTL 명세로부터 열차 제어기를 자동 생성



국내 정형 기법의 역사

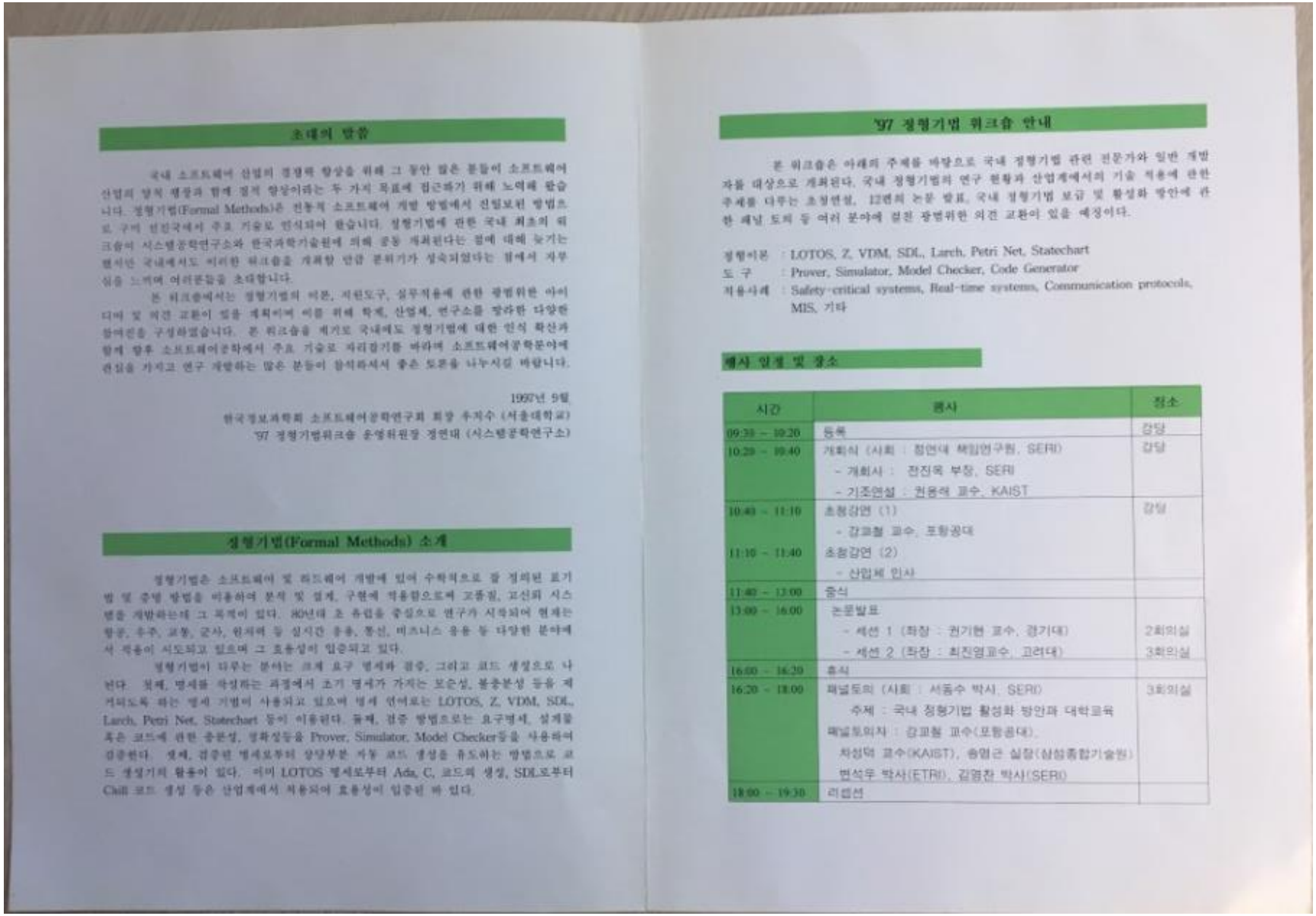


'97 정형기법 워크숍
Formal Methods Workshop '97

'97정형기법워크숍

일시 : 1997. 9. 26. (금)
장소 : 시스템공학연구소 (SERI)
주최 : 한국정보과학회 소프트웨어공학연구회
주관 : 시스템공학연구소 소프트웨어공학연구부,
한국과학기술원 공학부

국내 정형 기법의 역사



초대의 말씀

국내 소프트웨어 산업의 경쟁력 향상을 위해 그 동안 많은 분들이 소프트웨어 산업의 양적 팽창과 함께 질적 향상이라는 두 가지 목표에 접근하기 위해 노력해 왔습니다. 정형기법(Formal Methods)은 전통적 소프트웨어 개발 방법에서 신뢰도의 방법으로 구미 선진국에서 주요 기술로 인식되어 왔습니다. 정형기법에 관한 국내 최초의 워크숍이 시스템공학연구소와 한국과학기술원에 의해 공동 개최된다는 점에 대해 늦기는 했지만 국내에서도 이러한 워크숍을 개최할 만큼 분위기가 성숙되었다는 점에서 자부심을 느끼며 여러분들을 초대합니다.

본 워크숍에서는 정형기법의 이론, 지원도구, 실무적용에 관한 광범위한 아이디어 및 의견 교환이 있을 계획이며 이를 위해 학계, 산업체, 연구소를 망라한 다양한 참여진을 구성하였습니다. 본 워크숍을 계기로 국내에도 정형기법에 대한 인식 확산과 함께 향후 소프트웨어공학에서 주요 기술로 자리잡기를 바라며 소프트웨어공학분야에 관심을 가지고 연구 개발하는 많은 분들이 참석하셔서 좋은 토론을 나누시길 바랍니다.

1997년 9월

한국정보과학회 소프트웨어공학연구회 회장 우치수 (서울대학교)

'97 정형기법워크숍 운영위원장 정연대 (시스템공학연구소)

정형기법(Formal Methods) 소개

정형기법은 소프트웨어 및 하드웨어 개발에 있어 수학적으로 잘 정의된 표기법 및 증명 방법을 이용하여 분석 및 설계, 구현에 적용함으로써 고품질, 고신뢰 시스템을 개발하는데 그 목적이 있다. 80년대 초 유럽을 중심으로 연구가 시작되어 현재는 항공, 우주, 교통, 군사, 원자력 등 실시간 응용, 통신, 비즈니스 응용 등 다양한 분야에서 적용이 시도되고 있으며 그 효용성이 입증되고 있다.

정형기법이 다루는 분야는 크게 요구 명세와 검증, 그리고 코드 생성으로 나뉜다. 첫째, 명세를 작성하는 과정에서 초기 명세가 가지는 모순성, 불충분성 등을 제거되도록 하는 명세 기법이 사용되고 있으며 명세 언어로는 LOTOS, Z, VDM, SDL, Larch, Petri Net, Statechart 등이 이용된다. 둘째, 검증 방법으로는 요구명세, 설계물 혹은 코드에 관한 증명성, 정확성등을 Prover, Simulator, Model Checker등을 사용하여 검증한다. 셋째, 검증된 명세로부터 상당부분 자동 코드 생성을 유도하는 방법으로 코드 생성기의 활용이 있다. 이미 LOTOS 명세로부터 Ada, C, 코드의 생성, SDL로부터 Chill 코드 생성 등은 산업계에서 적용되어 효용성이 입증된 바 있다.

'97 정형기법 워크숍 안내

본 워크숍은 아래의 주제를 바탕으로 국내 정형기법 관련 전문가와 일반 개발자를 대상으로 개최된다. 국내 정형기법의 연구 현황과 산업계에서의 기술 적용에 관한 주제를 다루는 초청연설, 12편의 논문 발표, 국내 정형기법 보급 및 활성화 방안에 관한 패널 토의 등 여러 분야에 걸친 광범위한 의견 교환이 있을 예정이다.

정형이론 : LOTOS, Z, VDM, SDL, Larch, Petri Net, Statechart
도 구 : Prover, Simulator, Model Checker, Code Generator
적용사례 : Safety-critical systems, Real-time systems, Communication protocols, MIS, 기타

행사 일정 및 장소

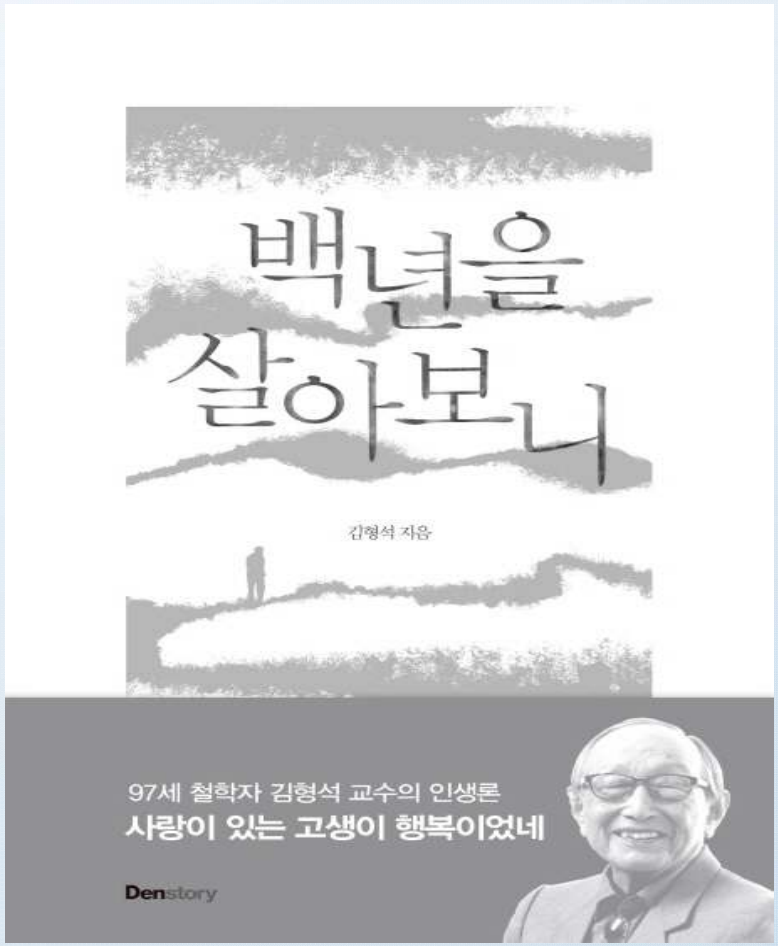
시간	행사	장소
09:30 - 10:20	등록	강당
10:20 - 10:40	개회식 (사회 : 정연대 책임연구원, SERI) - 개회사 : 전진욱 부장, SERI - 기조연설 : 전용석 교수, KAIST	강당
10:40 - 11:10	초청강연 (1) - 강교철 교수, 포항공대	강당
11:10 - 11:40	초청강연 (2) - 산업체 인사	
11:40 - 13:00	중식	
13:00 - 16:00	논문발표 - 세션 1 (좌장 : 권기현 교수, 경기대) - 세션 2 (좌장 : 최진영교수, 고려대)	2회의실 3회의실
16:00 - 16:20	휴식	
16:20 - 18:00	패널토의 (사회 : 서동수 박사, SERI) 주제 : 국내 정형기법 활성화 방안과 대학교육 패널토의자 : 강교철 교수(포항공대), 차성덕 교수(KAIST), 송영근 실장(삼성종합기술원) 변석우 박사(ETRI), 김영찬 박사(SERI)	3회의실
18:00 - 19:30	리셉션	

질의 응답

30년 SE 해보니...그리고 지금...

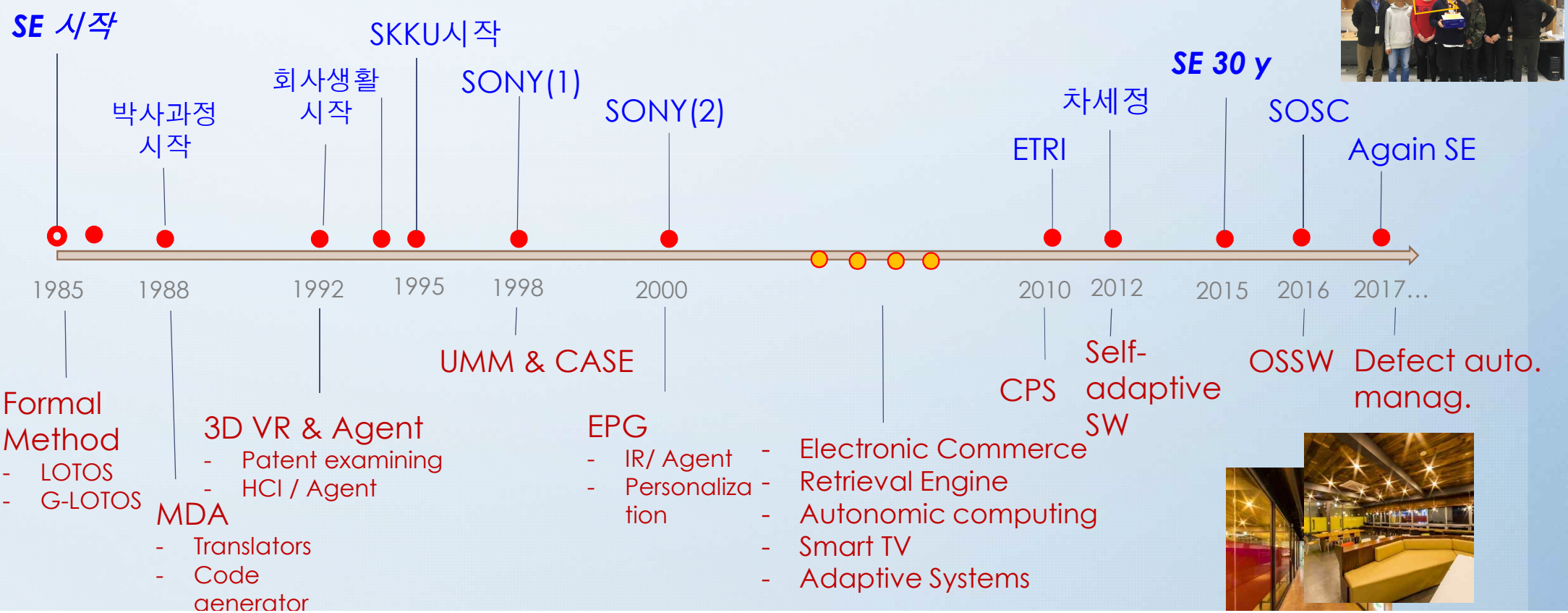
성균관대학교 소프트웨어대학
교수 이은석

백년을 살아보니 – 김형석 교수님

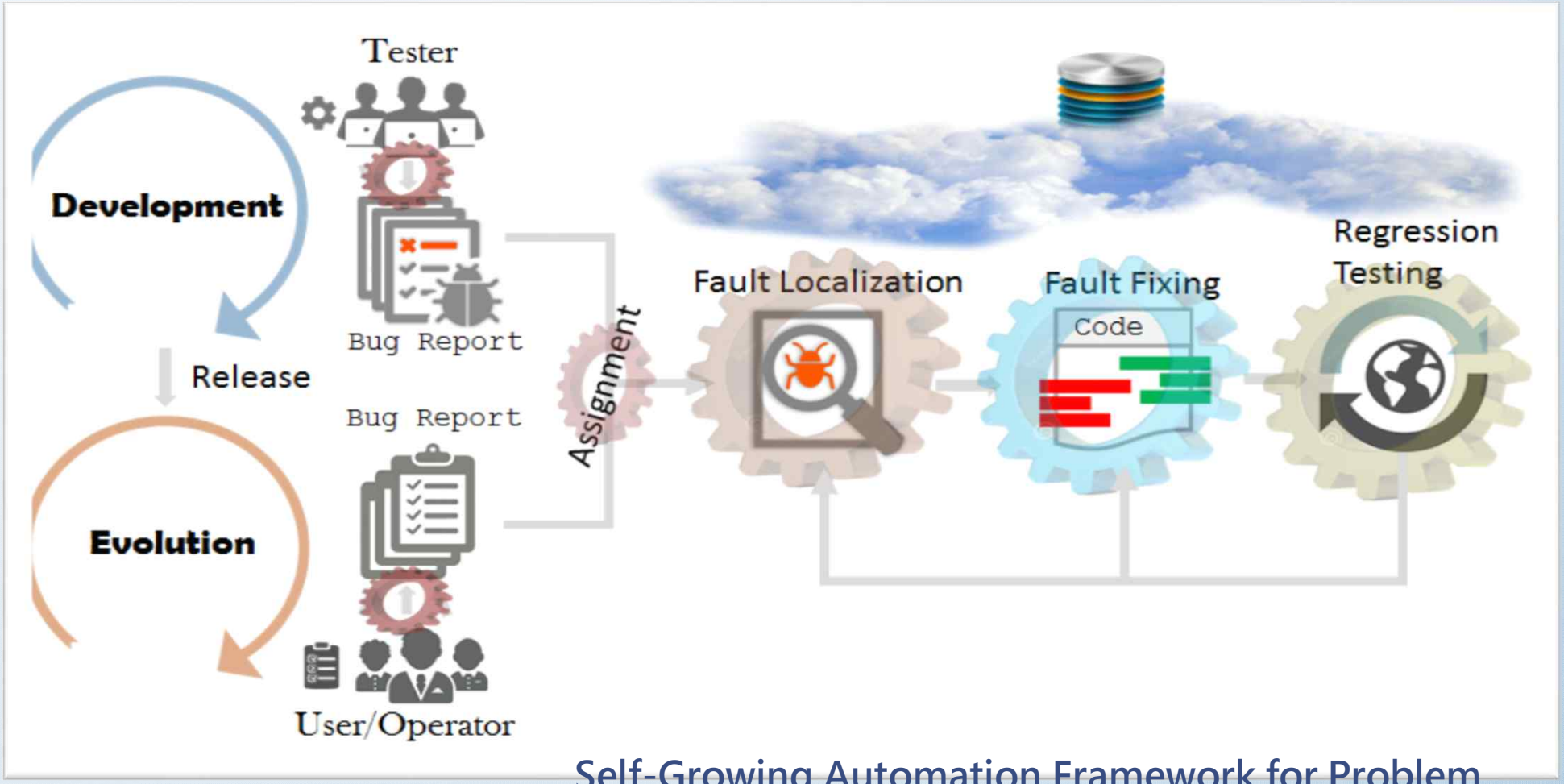


해보니 그게...

- 1985 – 2015 &



Automation Framework for PS



Self-Growing Automation Framework for Problem Solving

SE & 오픈소스 SW 교육

오픈소스SW기반 SE교육, 이은석 외 정보과학회지 2017년4월

- 78%의 회사가 오 픈소스 기반으로 운영,
- 소프트웨어가 필요할 때 66% 이상의 회사가 오픈소스가 있는지 먼저 확인,
- 39%의 회사가 오픈소스 프로젝트를 계획 중,
- 47%의 회사가 자사의 툴과 프로젝트를 오픈소스로 공개,
- (Gartner)2017 년까지 IT 분야 글로벌 3000여개 기업의 OSS 활용률 은 99%까지 확대될 전망
- (IDC)2016년 전 세계 OSS 매출은 619억 달러에 달하며 연 평균 성장률도 19%로 예상
- 구글, 페이스북, MS, 아마존 등 자사의 서비스, 플랫폼. JDK 등 오픈소스로 공개
- …
- …

NORTH BRIDGE+BLACKDUCK, 2016 The Future of OPEN SOURCE, Apr.25.2016.

국내 SE 교육 사례

구분	강의내용(구성)
성균관대학교 (2016)	1.OT and Introduction to Software Engineering/ 2.Sociotechnical Systems/ 3.Software Processes/ 4.Agile Software Development/ 5.Requirement Engineering/ 6.Object Modeling Technique/ 7.System Modeling/8.Architectural Design/ 9.Design and Implementation with OSS/ 10.Software Testing/ 11.Project Management/ 12.Project Estimation/ 13.Software Process Assessment and Improvement with CMMI and SPICE/ 14.Wrap up and Project Demo
서울대학교 (2016)	1.Orientation/ 2.Software Requirement & Data Modeling/ 3.Software Requirement & Data Modeling/ 4.Object-Oriented Analysis & Design/ 5.Object-Oriented Analysis & Design/ 6.Clean Code/ 7.Software Quality/ 8.SW개발 패러다임의 변화/ 9.Agile Software Development/ 10.Agile 방식의 팀 프로젝트/ 11.Agile 방식의 팀 프로젝트/ 12.Agile 방식의 팀 프로젝트
KAIST (2016)	1.Intro/ 2. Intro.to SE/ 3.Necessity of clear technical communication, Propositional logic/ 4.Tutorial for 1st order logic/ 5.First order logic/ 6.SE, Software process structure/ 7.Process models, Agile Development/ 8.Unified Modeling Language/ 9.Design Concepts/ 10.Hints for desigining Safehome system, Architectural design/ 11.Component-level design/ 12.Product Metrics/ 13.Testing overview/ 14.Testing Conventional Applications, Category partitioning-methods as a black box testing technique/ 15.White-box testing/ 16.White-box testing
연세대학교 (2015)	1.OT/ 2.소프트웨어 공학의 개요/ 3.소프트웨어 생명주기 모델/ 4.프로젝트 관리/ 5.프로젝트 계획 및 통제/ 6.프로젝트 계획 및 통제 위험관리/ 7.요구사항 개발 및 관리/ 8.요구사항 개발 및 관리/ 9.설계 응집도와 결합도 개념/ 10.설계 정보 은닉, 구조도/ 11.설계 자료흐름도를 구조도로 변환하는 방법/ 12.테스트의 원리/ 13.블랙박스 테스트와 화이트박스 테스트/ 14.동등 클래스 분할/경계 값 분석/ 15.형상 관리
고려대학교 (2015)	1.Introduction/ 2.Software process and process improvement/ 3.Project management and managing people/ 4.Software cost estimation and metrics/ 5.Quality management and configuration management/ 6.Requirements engineering/ 7.Requirements engineering/ 8.System models/ 9.Architectural design/ 10.Object-oriented design and UML/ 11.UML/ 12.Real-time software design/ 13.Testing and quality assurance/ 14.Testing and quality assurance/ 15.Verification and validation/ 16.Wrap up and project demo

해외 SE 교육 사례

구분	강의내용(구성)
CMU (2016) 15-313	https://www.cs.cmu.edu/~ckaestne/15313/2016/index.html 1.Git and GitHub/ 2.Measuring Software Characteristics / 3.Software Requirements/ 4.Requirements Interviews/ 5.Architectural Assessment and Decisions/ 6.Let's Create a Startup!/ 7.Midterm review/ 8.Static Analysis (FindBugs)/ 9.Dynamic Analysis(ASM)/ 10.QA Plan/ 11.Agile methods/ 12.Process comparison/ 13.No recitation/ 14.Team dysfunctions
CMU (2014) 15-413	http://www.cs.cmu.edu/~aldrich/courses/15-413/ Carry out an open source software development project for a real client Practice software engineering processes and techniques 1.Intro/ 2.Process/ 3.OSS-process/ 4.Process-at-github/ 5.Architecture/ 6.QA topic overview (inspection, testing, static analysis) Project list http://www.cs.cmu.edu/~schmerl/courses/15-413/projects.html http://www.cs.cmu.edu/~charlie/courses/15-413/2016-spring/index.html 15, 16, 17년 강좌는 개설되고 있으나, 자세한 강의 내용 없음. Prof. Charlie Garrod
CMU (2016) 17-624 (대학원)	http://mse.isri.cmu.edu/software-engineering/documents/syllabi/17-624-m11-opensourcesw.pdf Open Source Software Development: Software Engineering Master program

해외 SE 교육 사례

UIUC (2016) CS427	https://wiki.illinois.edu/wiki/display/cs427fa16/Schedule 1.Introduction/K Framework / 2.Bug Reporting/Software Configuration Management (Version Control)/ 3.Testing1/Software Configuration Management (Build)/ 4.Software Configuration Management (Change Control)/Testing 2 / 5.Core XP Practices CATME /More XP: Planning /Game/ 6.Testing3/Testing4/ 7.Refactoring/Scaling Engineering/ 8.Code Smells/Reverse Engineering/ 9.Metrics/Project Team Icebreaker/ 10.Requirements and Risks/ 11.Metrics/Testing5/ 12.Program Transformation/Regression Test Selection/ 13.Components and Reuse/ 14.Design patterns(1,2)/ 15.Design patterns(3)/Testing Mistakes
UT Austin (2016) CS373	http://www.cs.utexas.edu/users/downing/cs373/ It is also strongly focused on using tools to improve the quality of software development, including: 1.VM development with Docker/ 2.Automated builds with make source control with git and GitHub/ 3.Issue tracking with GitHub/ 4.Compiling with python 3.5.2 / 5.Code static analysis with pylint / 6.Unit testing with unittest/ 7.Code coverage with coverage / 8.Code profiling with cProfile / 9.Documentation with pydoc / 10.Automated formatting with autopep8/ 11.Continuous integration with Travis CI/ 12.Using the NumPy
UC Berkeley (2015) CS169	https://sites.google.com/site/ucbcs169fa15/lectures 1.Introduction/ 2.Web Apps/HTML/CSS/ 3.Model-View-Controller/ 4.Software Processes/ 5.Finish Processes. Requirements and Specifications/ 6.UML/ 7.Advanced Version Control with git/ 8.Unit Testing/ 9.Test-Driven Development/ 10.Test Doubles/ 11.Design for Testability/ 12.Code Coverage Analysis/ 13.Delta Debugging/ 14.Deploying Web Services

성대에서의 오픈소스 SW 교육

오픈소스SW기반 SE교육, 이은석 외 정보과학회지 2017년4월

- (OSS개론) OSS의 역사/ OSS 문화(해커문화)의 진화/ OSS 활용방법(툴, 소스, 패키지)/ OSS 프로젝트 및 커뮤니티/ 글로벌 OSS 커뮤니티 참여 방법과 개발문화/ OSS 개 발 방법과 그 프로세스/ OSS 개발 단계별 적용 방법 론/ OSS에서 활용하는 개발환경과 소프트웨어 툴/ OSS 비즈니스 전략/ OSS에 적용되는 라이선싱과 지켜야 할 규칙/ OSS 컴플라이언스 및 거버넌스/ OSS 의 확산되는 영향력/ OSS와 타 영역의 융복합/ OSS 전망과 시사점 및 주의할 점
- (OSS실습) 리눅스 데스크톱 사용법/ 리눅스 서버를 이용한 개 발환경 구축/ Git/GitHub 사용 및 활용전략/ 글로벌 커 뮤니티 참여 및 공헌/ 기존 OSS 커뮤니티 참여활동 실습/ OSS 프로젝트 실습을 통한 개발환경 활용 실습 / OSS 툴을 활용한 테스팅 실습/ OSS 컴플라이언스 검증 실습/ 개인별 팀별 GitHub 사이트를 통한 개발 자 포트폴리오 구축

Q & A