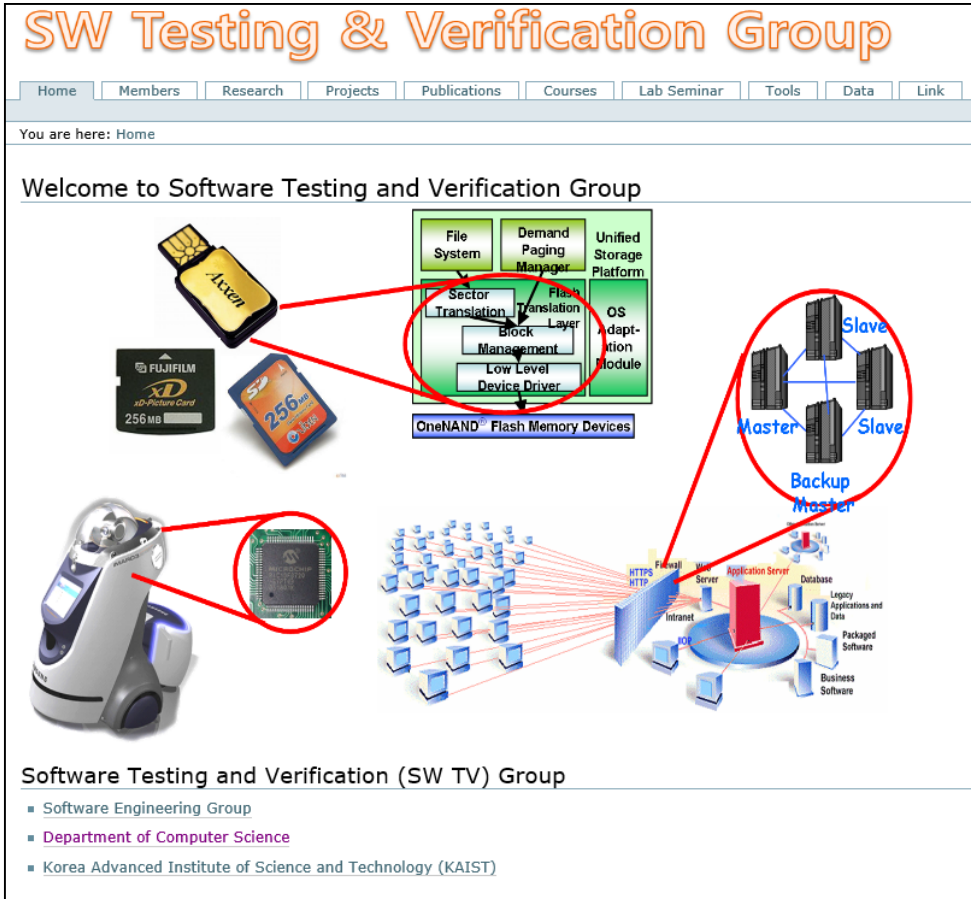
CS492: Automated Software Analysis Techniques

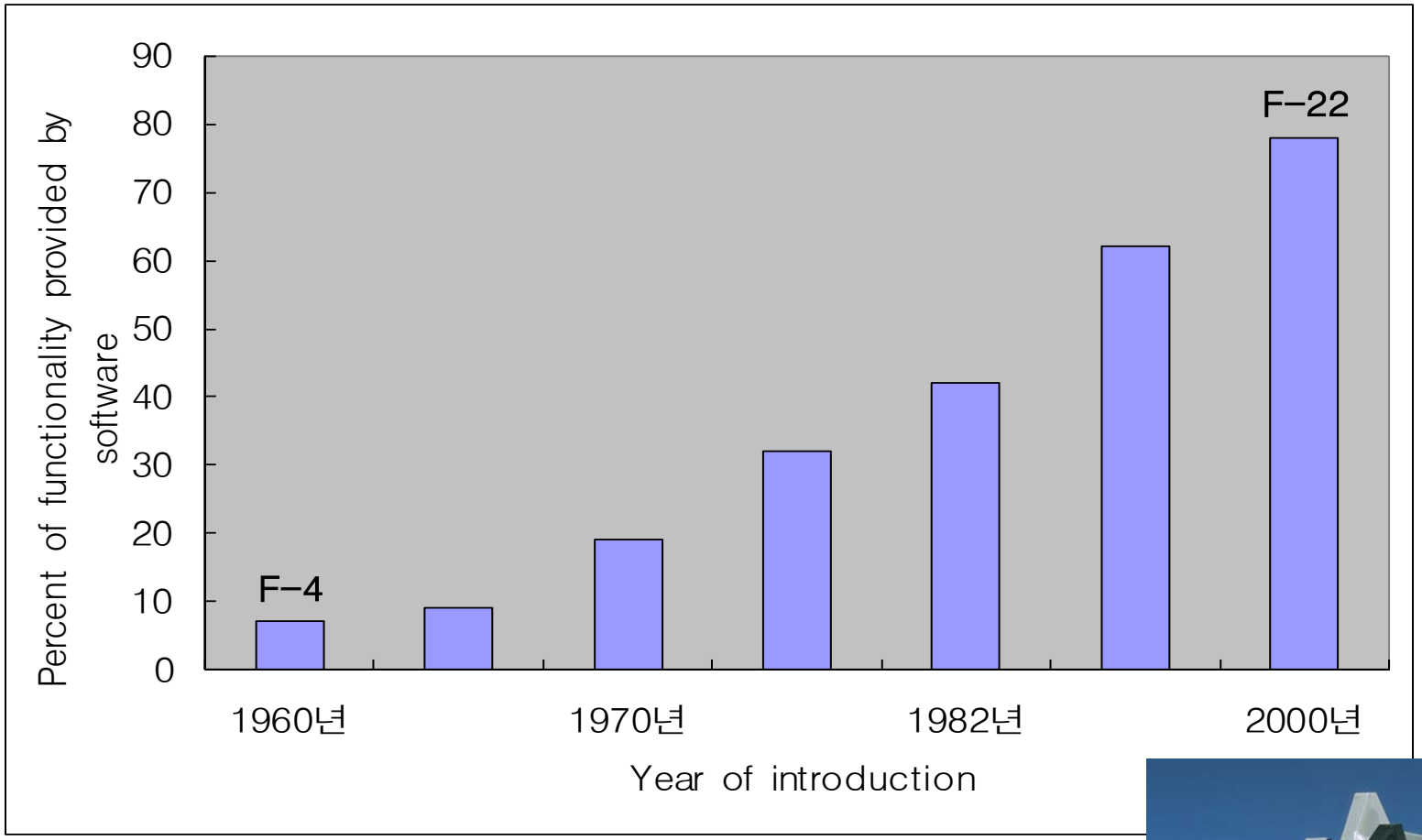
Moonzoo Kim

Software Testing and Verification Group

CS Dept. KAIST



Role of S/W: Increased in Everywhere



F-35 (8 Mloc)

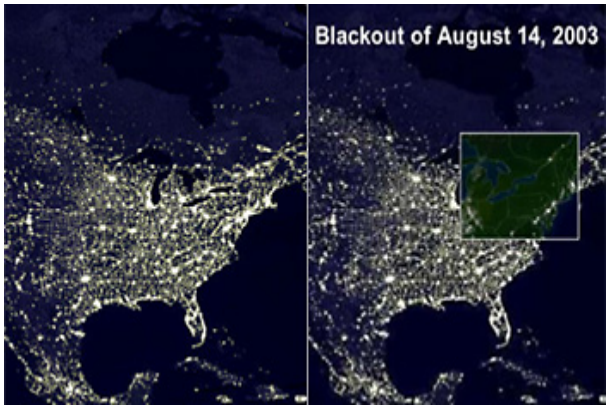
2012년

자료출처: Watts Humphrey 2002

KAIST

Social and Economic Loss due to High Complexity of SW

Although most areas of modern society depends on SW, reliability of SW is not improved much due to it high complexity



Medical accident: Therac 25
- For 1985-1987, excessive radio reactive beam enforced.
- 6 persons died due to the problem
- Data race bug was the cause of the problem

2003 US & Canada Blackout
- 7 states in US and 1 state in Canada suffered 3 days electricity blackout
- Caused by the failures of MISO monitoring SW
- 50 million people suffered and economic loss of 6 billion USD

Toyoda SUA (sudden unintended acceralation)
- Dozens of people died since 2002
- SW bugs detected in 2012
- Fined 1.2 Billion USD in 2014

SOFTWARE CAUSES OF MEMORY CORRUPTION

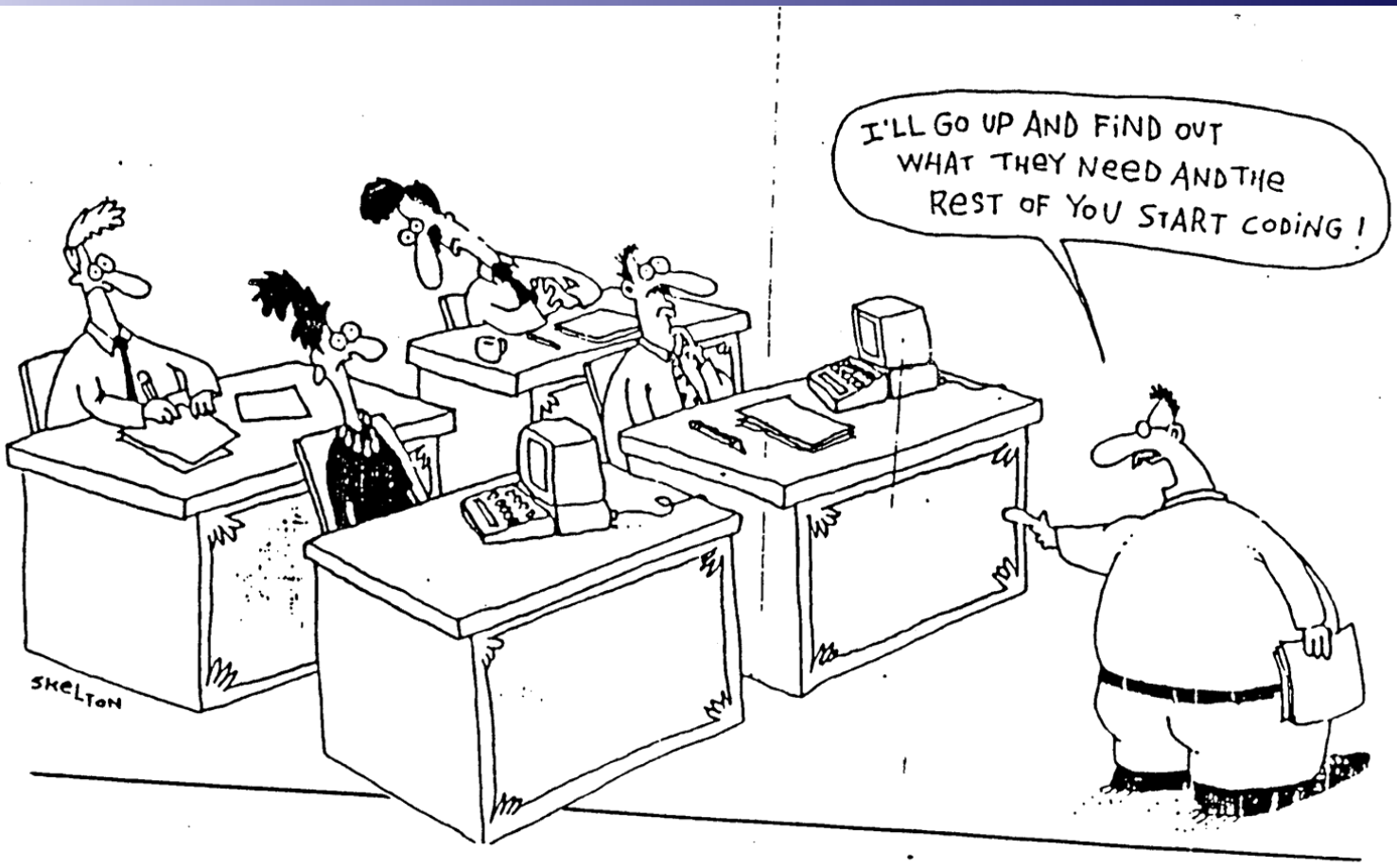
Type of Software Defect	Causes Memory Corruption?	Defect in 2005 Camry L4?
Buffer Overflow	Yes	Yes
Invalid Pointer Dereference/Arithmetic	Yes	Yes
Race Condition (a.k.a., "Task Interference")	Yes	Yes
Nested Scheduler Unlock	Yes	Yes
Unsafe Casting	Yes	Yes
Stack Overflow	Yes	Yes

Static analysis falls short of such complex bugs accurate
- **High false negatives**
- **High false positives**

⇒ **Systematic and dynamic analysis (i.e. automated sw testing) is MUST for high quality SW**

BARR group

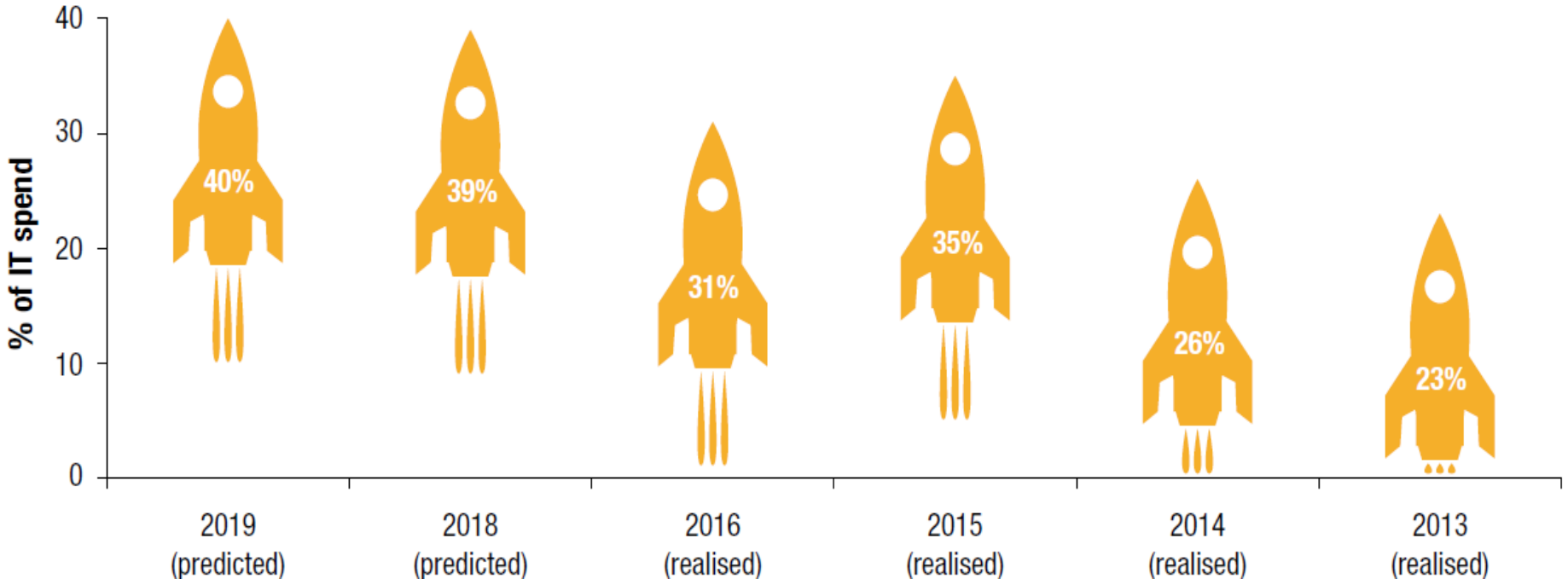
Current Practice for SW



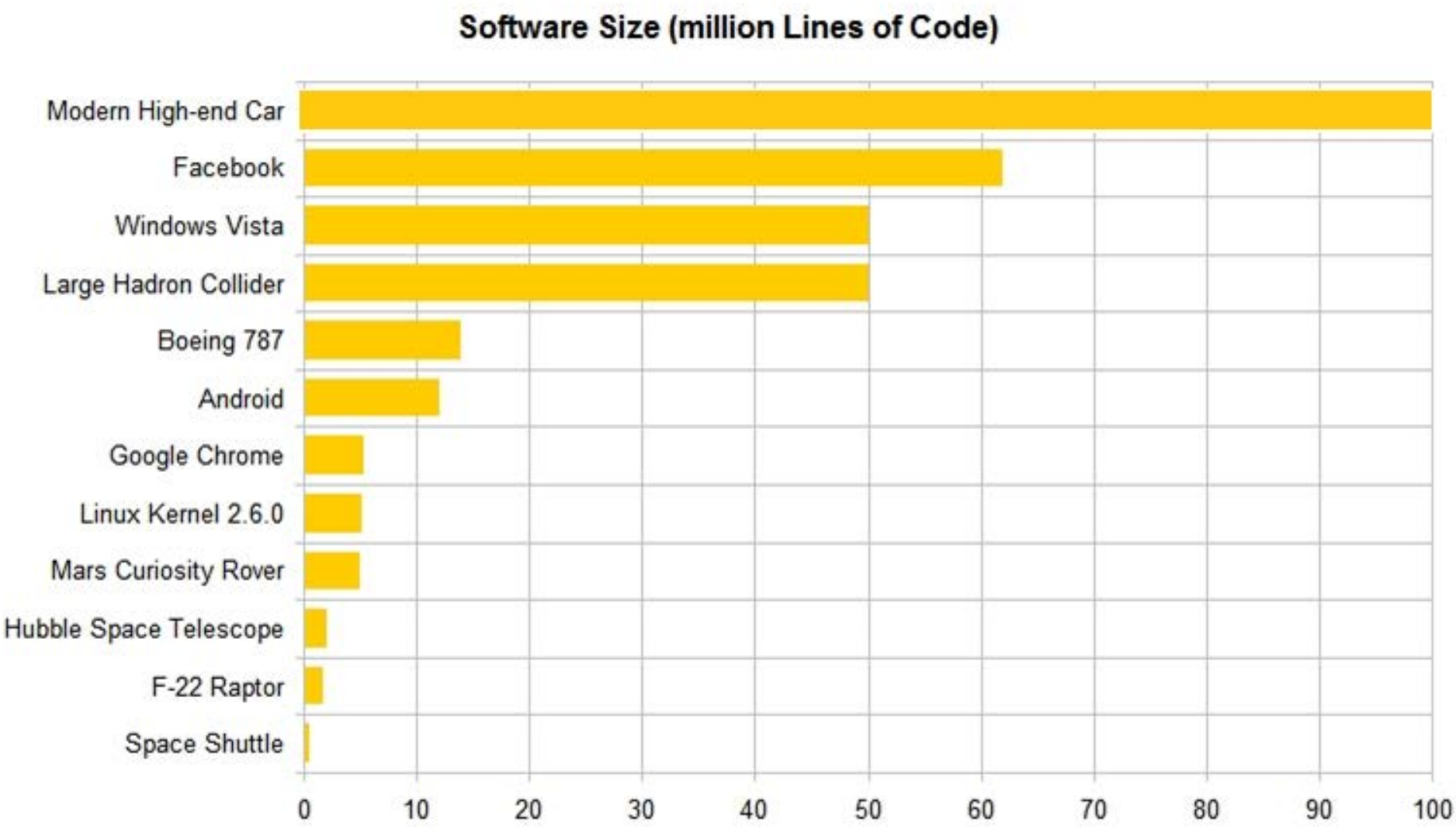
- **SW developers have to follow systematic disciplines for building and analyzing software with high quality**
  - This class focuses on the analysis activities

SW Verification & Testing Market Trends

- SW verification and testing market: 19.3 Million USD (193억원) @ 2015, annual growth: 15% (expected) [IDC]
- 31% of total expenses of IT companies is due to QA and SW testing, increasing to 40% (expected)
  [World Quality Report 2016-2017]

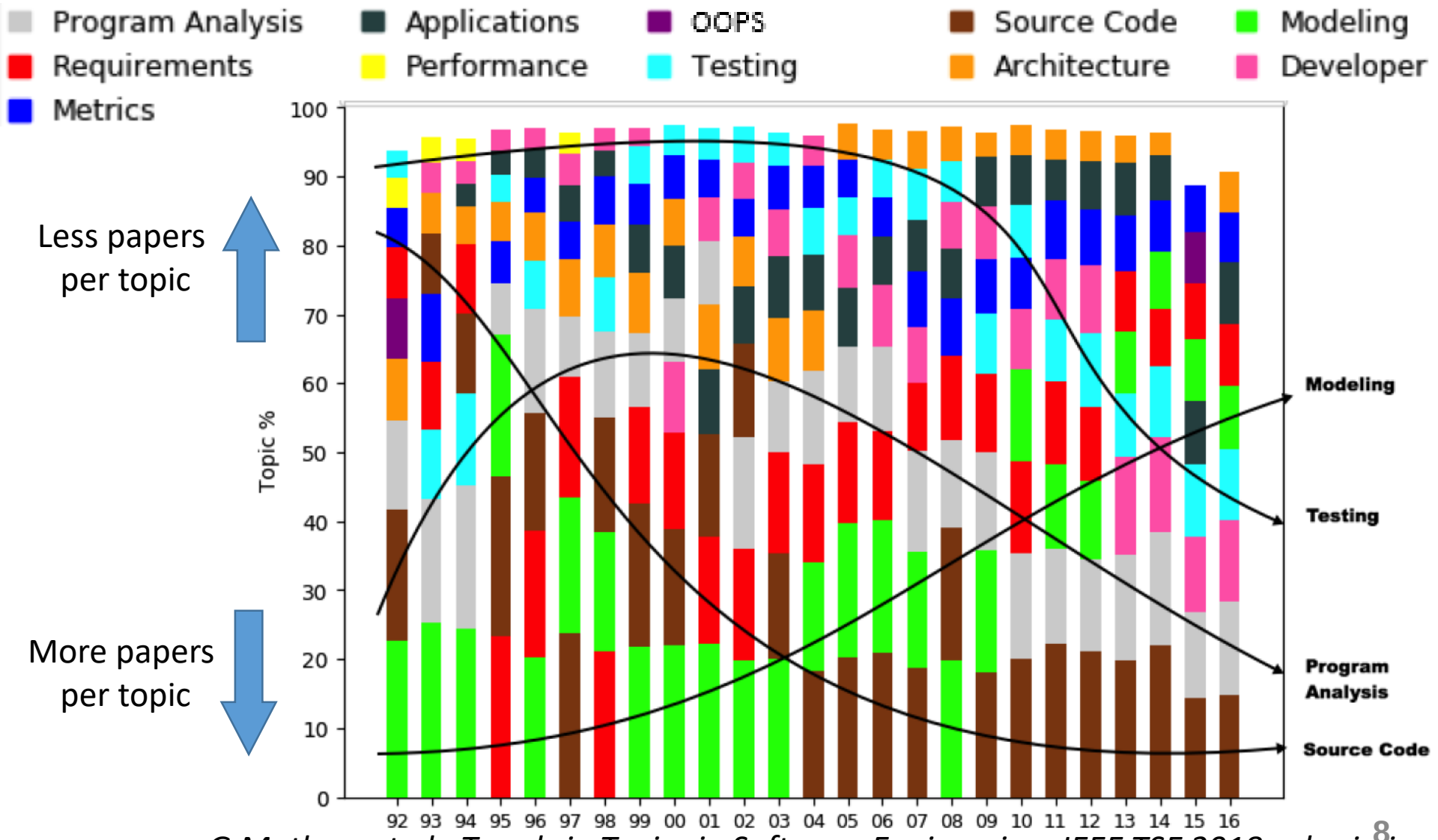


Size and Complexity of Modern SW



A.Busnelli, Counting, https://www.linkedin.com/pulse/20140626152045-3625
http://www.informationisbeautiful.net/visualizations/million-lines-of-code/

SE Research Topic Trends among 11 Major Topics (1992-2016)



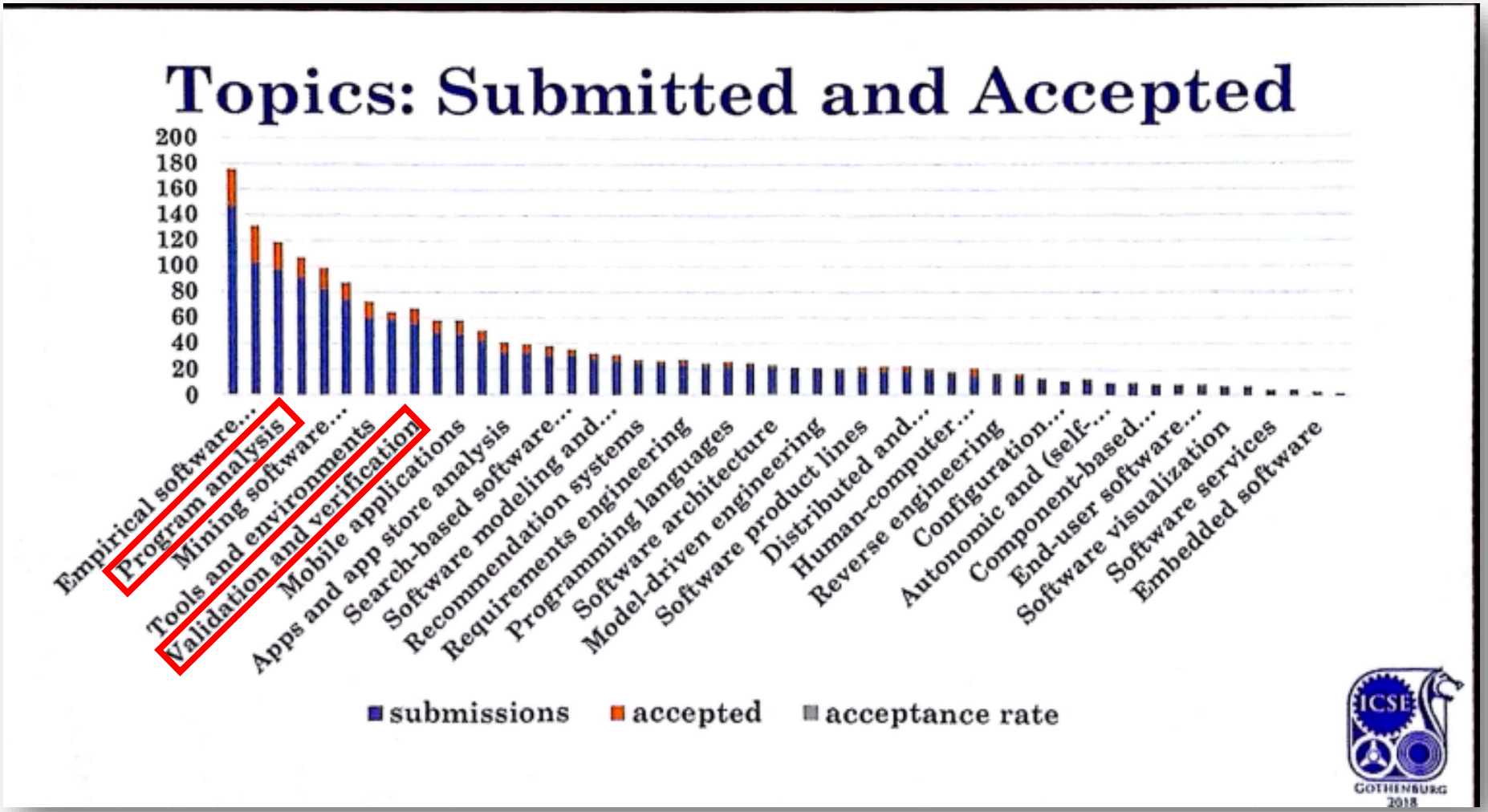
G.Mathew et al., Trends in Topics in Software Engineering, IEEE TSE 2018 submission

Most Cited Papers in Each of the 11 Major SE Topics

Topic	Top Papers
Program Analysis	2012: Genprog: A generic method for automatic software repair; C Le Goues, TV Nguyen, S Forrest, W W 2009: Automatically finding patches using genetic programming; W Weimer, TV Nguyen, C Le Goues, S
Requirements	2009: A systematic survey of program comprehension through dynamic analysis; B Cornelissen, A Zaidma 2009: Software architecture reconstruction: A process-oriented taxonomy; S Ducasse, D Pollet
Metrics	2012: A systematic literature review on fault prediction performance in software engineering; T Hall, S Bee 2009: Predicting faults using the complexity of code changes; AE Hassan
Applications	2011: CloudSim: a toolkit for modeling & simulation of cloud computing; R.Calheiros, R.Ranjan, A.Belog 2011: A survey on privacy in mobile participatory sensing applications; D Christin, A Reinhardt, SS Kanh
Performance	2010: A theoretical and empirical study of search-based testing: Local, global, and hybrid search; M Harm 2011: Software module clustering as a multi-objective search problem; K Praditwong, M Harman, X Yao
OOPS	2009: Incremental Clone Detection; N Gode, R Koschke 2014: Variability in Software Systems - A Systematic Literature Review; M Galster, D Weyns, D Tofan, B
Testing	2011: An analysis and survey of the development of mutation testing; Y Jia, M Harman 2012: Regression testing minimization, selection and prioritization: a survey; S Yoo, M Harman
Source Code	2010: DECOR: A method for the specification and detection of code and design smells; N Moha, YG Guel 2013: Feature location in source code: a taxonomy and survey; B Dit, M Revelle, M Gethers, D Poshyvany
Architecture	2009: Software architecture many faces many places yet a central discipline; RN Taylor 2011: Reverse engineering feature models; S She, R Lotufo, T Berger, A Wasowski, K Czarnecki
Modelling	2009: The "physics" of notations: toward a scientific basis for constructing visual notations in software eng 2009: The Palladio component model for model-driven performance prediction; S Becker, H Koziolek, R
Developer	2009: Guidelines for conducting and reporting case study research in software engineering; P Runeson, M 2012: A decade of agile methodologies - Towards explaining agile software development; T Dingsoyr, SP

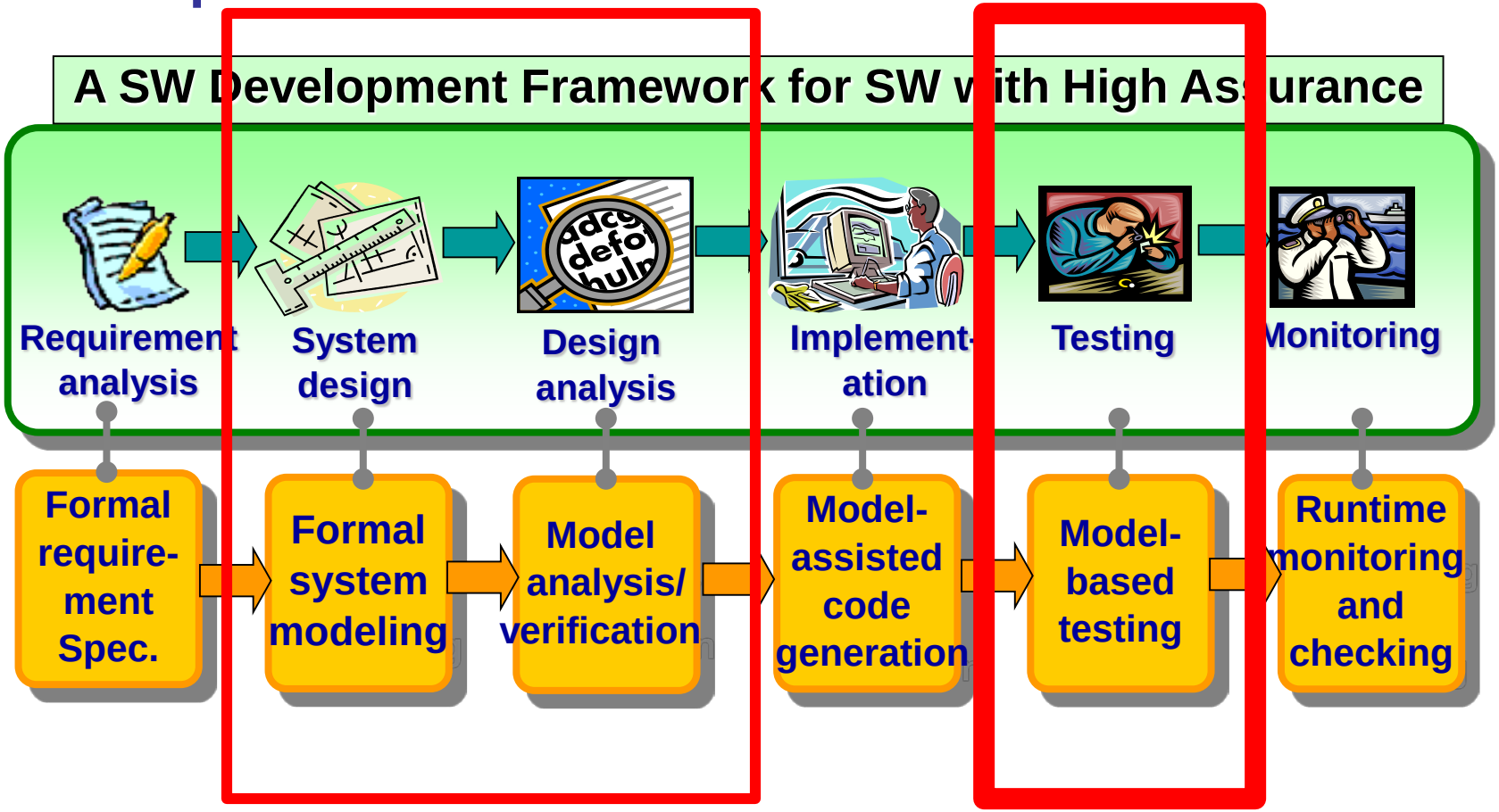
G.Mathew et al., Trends in Topics in Software Engineering, IEEE TSE 2018 submission

ICSE 2018 Topics
(Top SE conf. w/ accept. rate: 20%)

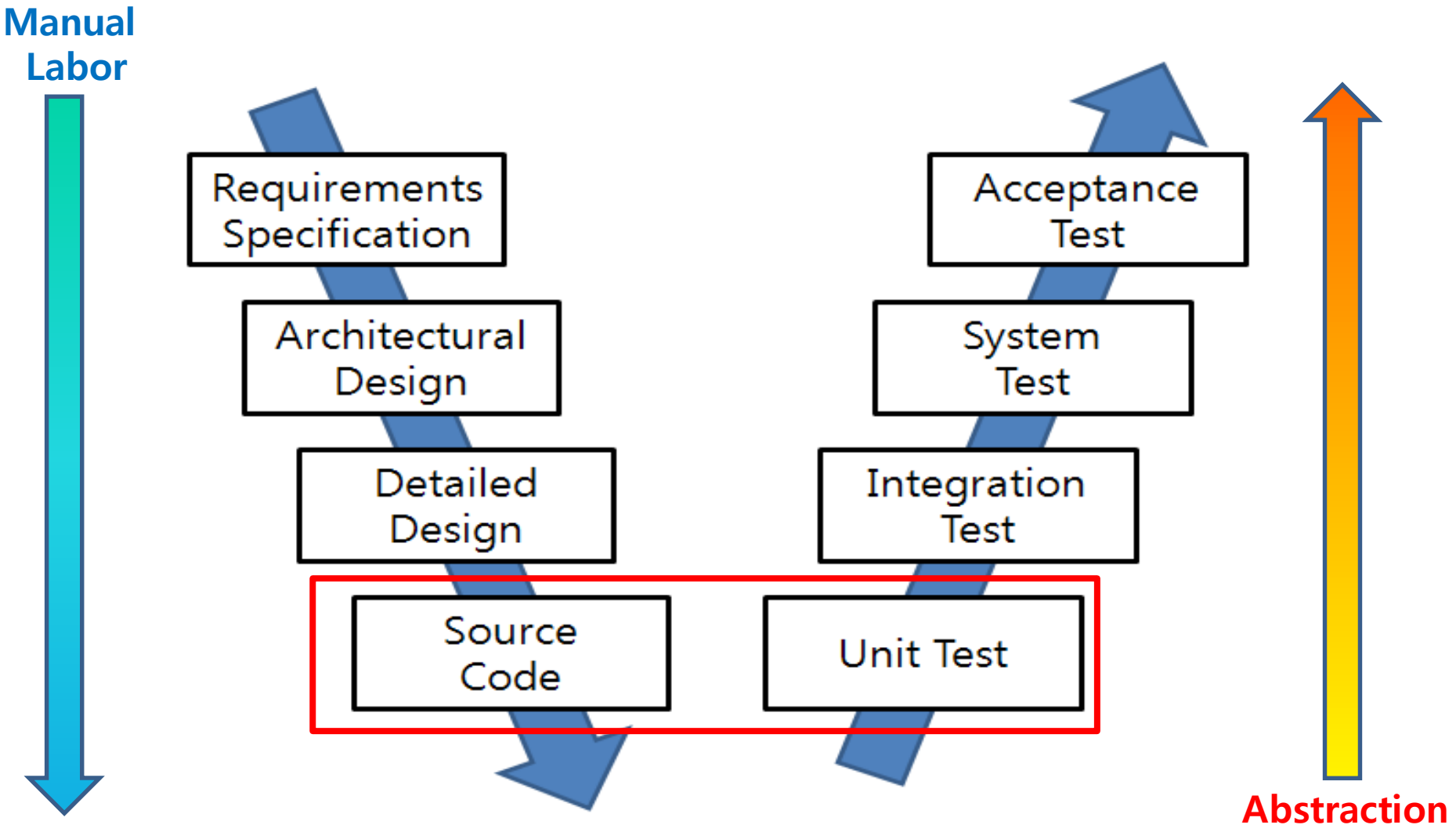


Software Development Cycle

- A practical end-to-end formal framework for software development



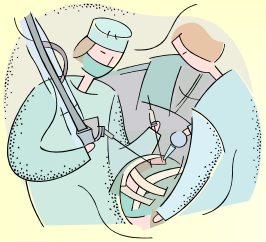
SW Development and Testing Model (a.k.a. V model)



Main Target Systems

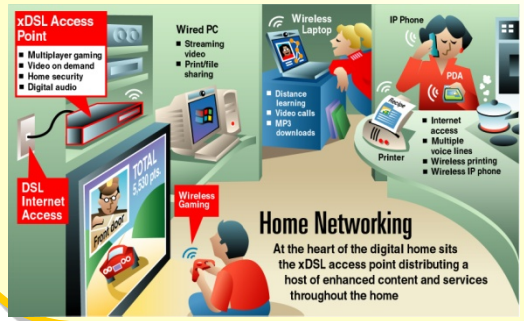
- **Embedded systems where highly reliable SW technology is a key to the success**
  - The portion of SW in commercial embedded devices increases continuously
  - More than 50% of development time is spent on SW testing and debugging

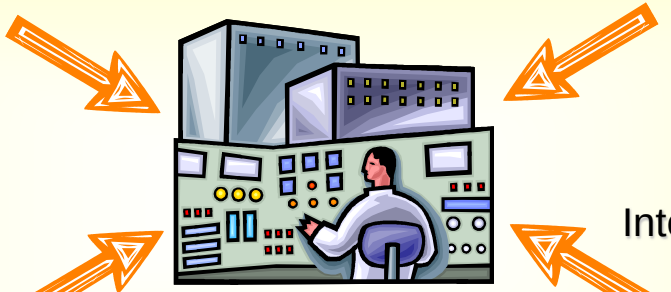
Intelligent Medical Devices



Home Service Robots

Home Network



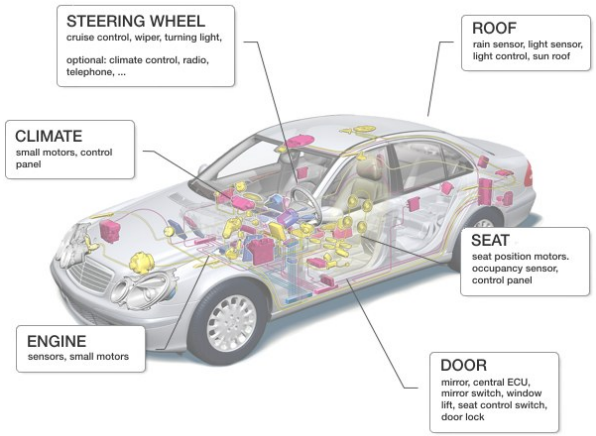


Highly Reliable Systems

Intelligent Mobile Systems

Strong IT Industry in South Korea

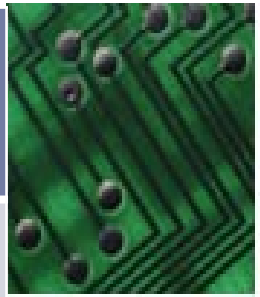
KIA MOTORS



Embedded Software in Two Different Domains

Conventional Testing | Concolic testing | Model checking

	Consumer Electronics	Safety Critical Systems
Examples	Smartphones, flash memory platforms	Nuclear reactors, avionics, cars
Market competition	High	Low
Life cycle	Short	Long
Development time	Short	Long
Model-based development	None	Yes
Important value	Time-to-market	Safety



How to Improve the Quality of SW

1. **Systematic testing (can be still manual)**
   - Coverage criteria
   - Mutation analysis
2. **Testing through automated analysis tools**
   - Scientific treatment of SW with computing power
     - Generate test inputs to detect bugs
     - Localize detected faults
     - Repairing the fault with patches
3. **Formal verification**
   - Guarantee the absence of bugs

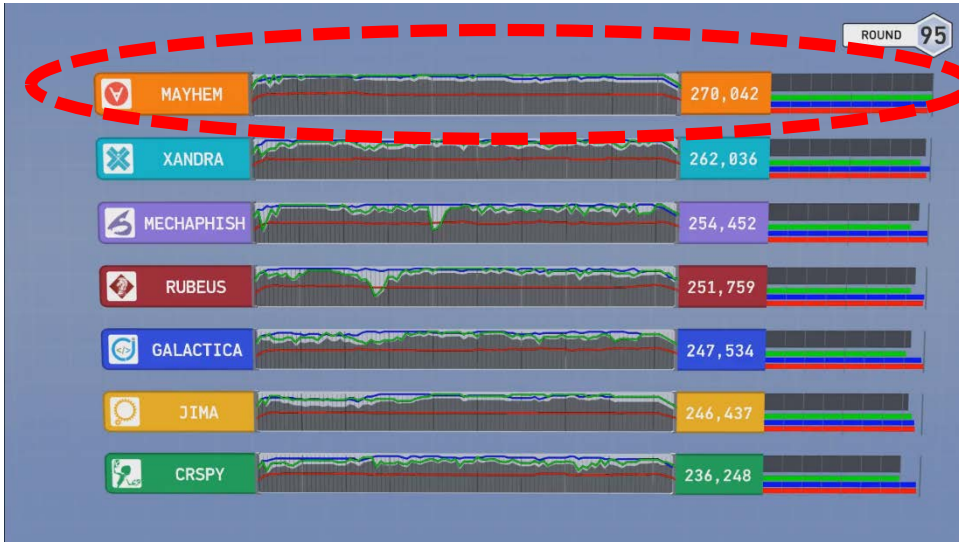
Microsoft Project Springfield

- Azure-based cloud service to find security bugs in x86 windows binary
- Based on concolic testing techniques of SAGE

2016 Aug DARPA Cyber Grand Challenge

-the world's 1st all-machine hacking tournament

- Each team's Cyber Reasoning System automatically identifies security flaws and applies patches to its own system in a hack-and-defend style contest targeting a new Linux-based OS DECREE
- Mayhem won the best score, which is CMU's concolic testing based tool



현대모비스, AI 기반 소프트웨어 검증시스템 도입..."효율 2배로"

2018-07-22 10:00

댓글 가- 가+

'마이스트' 적용...대화형 검색 로봇 '마이봇'도 도입

(서울=연합뉴스) 윤보람 기자 = 현대모비스[012330]가 인공지능(AI)을 활용해 자율주행, 커넥티비티(연결성) 등 미래 자동차 소프트웨어(SW) 개발에 속도를 낸다.

현대모비스는 AI를 기반으로 하는 소프트웨어 검증시스템 '마이스트'(MAIST: Mobis Artificial Intelligence Software Testing)를 최근 도입했다고 22일 밝혔다.

Google에 의해 종료된 광고입니다.
이 광고 그만 보기
이 광고가 표시된 이유

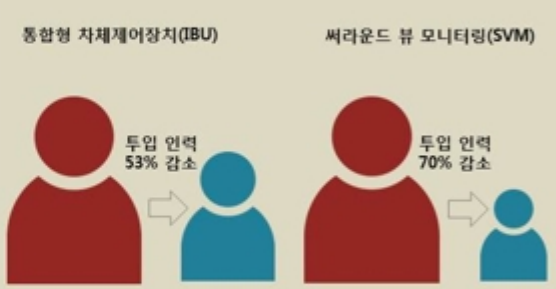
현대모비스가 카이스트 전산학부 김문주 교수와 공동으로 개발한 마이스트는 연구원을 대신해 소프트웨어 검증작업을 수행하는 AI 시스템이다.

연구원들이 설계한 알고리즘을 바탕으로 소프트웨어의 모든 연산과정을 AI로 검증한다. 기존에 수작업으로 이뤄지던 소프트웨어 검증업무를 자동화한 셈이다.

실제 현대모비스가 통합형 차체제어시스템(IBU)과 써라운드뷰모니터링 시스템(SVM) 검증에 마이스트를 시범 적용한 결과 마이스트가 처리한 검증 업무량 비중은 각각 53%, 70%로 높았다.

현대모비스는 하반기부터 소프트웨어가 탑재되는 제동, 조향 등 모든 전장부품으로 마이스트를 확대 적용할 계획이다. 글로벌 소프트웨어 연구기지인 인도연구소에도 적용한다.

■ 현대모비스 인공지능 도입 사례

AI 시스템	목적	도입 효과
마이스트 (Mobis AI Software Testing)	소프트웨어 검증 자동화	
마이봇 (Mobis AI Robot)	소프트웨어 개발문서 검색	딥러닝 기반, 개발문서 20만 건 관리

http://m.yna.co.kr/kr/contents/?cid=AKR20180720158800003&mobile

Questions???

- **Is automated testing really beneficial in industry?**
  - Yes, dozens of success stories at Samsung
- **Is automated testing academically significant?**
  - Yes, 3 Turing awardees in ‘07
- **Is automated testing too hard to learn and use?**
  - No, there are tools available

Research Trends toward Quality Systems

- **Academic research on developing embedded systems has reached stable stage**
  - just adding a new function to a target system is not considered as an academic contribution anymore
- **Research focus has moved on to the quality of the systems from the mere functionalities of the systems**
  - Energy efficient design, ez-maintenance, dynamic configuration, etc
- **Software reliability is one of the highly pursued qualities**
  - USENIX Security 2015 best paper
    - “Under-Constrained Symbolic Execution: Correctness Checking for Real Code” @ Stanford
  - ICSE 2014 best paper
    - “Enhancing Symbolic Execution with Veritesting” @ CMU
  - ASPLOS 2011 Best paper
    - “S2E: a platform for in-vivo multi-path analysis for software systems” @ EPFL

Tool-based Interactive Learning

- **Code analyzer**
  - C/C++ AST parser: Clang
  - Language independent Intermediate representation (IR) : LLVM
- **Model checker**
  - Explicit model checker: Spin home page
- **Software model checker**
  - Bounded model checker for C program: CBMC home page
- **Satisfiability solver**
  - **MiniSAT home page**
- **Satisfiability Module Solver**
  - **Z3 home page**
- **Concolic testing tools**
  - **CROWN home page**

Final Remarks

- **For undergraduate students:**
  - Highly recommend URP studies or independent studies
    - 이아청 detected several crash bugs in Hyundai Mobis SW during 2018 summer interns
- **For graduate students:**
  - Welcome research discussions to apply SW analysis techniques
    - Systematically testing/debugging C programs
    - Concurrency bug detection
- **Pre-requisite:**
  - Knowledge of the C/C++/Java programing language
  - Basic understanding of linux/unix
  - ~6 hours of analysis/programming per week for HW