Computational Thinking

2017-2학기

9장-정확하게 만들기
11장-동시적 행동

박종혁 교수 (컴퓨터공학과)

jhpark1@seoultech.ac.kr

http://www.parkjonghyuk.net

학습개요

- 9장 정확하게 만들기

1. "컴퓨터 오류"란 보통은.
2. 소프트웨어 정확성
3. 검증
4. 소프트웨어 테스팅
5. 화이트 박스 테스트
6. 균등 분할을 통한 블랙박스 테스트
7. 경계값 분석

학습목표

- 많은, 대부분은 아니지만, 컴퓨터 오류라고 불리는 것들은 사실은 사람에 의한 오류이며 보통은 데이터를 잘못 입력해서 생기는 오류라는 것을 이해한다.
- 명세서에서 정확한 것이 무엇인가가 정의되지 않으면 정확성은 불가능하다는 것을 이해한다.
- 검증과 확인 사이의 차이점과 그들이 중요한 이유를 설명할 수 있다.
- 소프트웨어의 정확성을 증명하는 것은 가능하지만 이러한 증명은 비용과 복잡성 때문에 극히 드물게 사용되고 있다는 점을 깨닫는다.
- 정확성과 관련해서 소프트웨어 테스트의 한계를 안다.

9.1 “컴퓨터 오류”란 보통은...

- 컴퓨터가 정확하게 동작하지 않는 것을 통틀어 말함
- 대부분의 사람의 실수에 의한 것들
- 따라서 사람이 실수를 덜 한다면 오류도 덜 나타나게 됨. 하지만 사람은 항상 실수를 하며 더 많은 자료의 값을 입력하면 할수록 더 많은 실수를 하게 됨.
- 실수를 덜 하도록 만들기 보다는 실수를 했을 경우에 검증 시스템을 통해 실제 시스템에 반영이 되지 않도록 만드는 것이 더 중요함

9.1 "컴퓨터 오류"란 보통은... (컴퓨터 오류의 원인)

- **부정확한 자료 입력**
  - 은행 계좌 이체 금액 잘 못 입력, 주민번호 실수 입력
- **시스템 이해 부족**
  - 이메일 답장 할 때 전체 답장, 브라우저 자동 완성기능
- **컴퓨터 출력 결과의 부정확한 해석**
  - 구매가격 잘 못 읽어 구매, 네비게이터 정보 잘 못 이해
- **물리적 피해**
  - 물리적 파손, 침수, 화재
- **하드웨어 고장**
  - 마우스 고장, DVD 플레이어 고장
- **소프트웨어 결함**
  - 프로그램 버그
- **보안 결함**
  - 외부 해킹

1. **소프트웨어가 고객의 요망에 맞게 동작한다.**
   - 확인(Validation) 프로세스를 통해 검사

2. **소프트웨어가 작성된 요구사항 명세서에 맞게 동작한다.**
   - 검증(Verification) 프로세스를 통해 검사

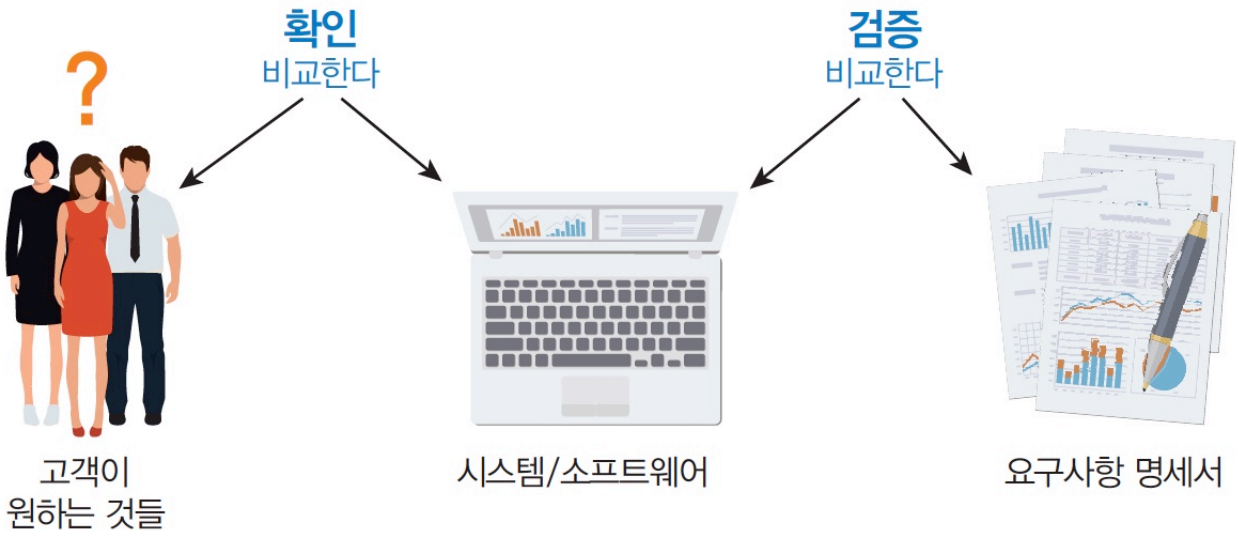


그림 9.2 두 종류의 정확성과 그들이 보장하기 위한 방법

❖ 확인 프로세스

- 고객이 원하는 대로 소프트웨어가 동작하는 지를 점검하는 프로세스

❖ 확인 프로세스 기법들

- 베타 테스트
  - 선택된 사용자들이 시스템을 사용해 보고 의견을 말하게 함
- 사용성 테스트
  - 선택된 개개인들이 시스템의 외관과 느낌을 탐색하게 함
- 인수 테스트
  - 고객이 최종 구매에 앞서 시스템을 확인함

9.3 검증

❖ **소프트웨어 요구사항이 명시된 명세서에 규정된 대로 소프트웨어가 동작하는 지를 점검하는 프로세스**

❖ **요구사항이 명확하고 일관성 있고 완전해야 검증 프로세스가 제대로 진행됨**

- 요구사항 S1) 프로그램이 안전해야 한다.
  - 검증 가능하지 않음
- 요구사항 S2) 모든 사용자는 사용자 이름과 패스워드를 가지고 로그인을 해야 한다.
  - 검증 가능함
- 요구사항 E1) 프로그램은 빨라야 한다.
  - 검증 가능하지 않음
- 요구사항 E2) 0.5초 내로 시스템은 반응해야 한다.
  - 검증 가능함

❖ 요구사항의 분류

- 기능적 요구사항
  - 소프트웨어가 '무엇을 하는 가'를 정의
- 비기능적 요구사항
  - 소프트웨어가 어떻게 잘 동작하는 가를 정의
  - 성능, 보안, 이동성, 신뢰성, 안전, 문서화, 물품 인도, 사용자 친화력 같은 부분을 기술
  - 검증이 가능하지 않은 S1과 E1의 모두 비기능적 요구사항

❖ 요구사항 작성시 모든 요구사항이 검증이 가능하도록 주의를 기울여야 함.

9.3 소프트웨어 테스팅 및 한계

❖ **소프트웨어 테스팅**

- 프로그램을 실행하고 명세서에 기록된 내용과 일치하는 지를 비교함
- 소프트웨어 테스팅은 정확성을 보증하지는 않음
  - 다양한 사용 환경에 대한 모든 경우의 수를 테스트 할 수 없기 때문
  - 따라서 모든 경우에 대한 테스팅 샘플을 만들어 테스트를 함
  - 결국 어떤 테스팅 샘플을 만드느냐가 소프트웨어 테스팅 성능에 중요한 영향을 미침
- **화이트박스 테스트와 블랙박스 테스트로 나뉨**

❖ **소프트웨어 테스팅 한계**

- 테스트 결과가 안전하게 나온다고 해서 시스템이 안전하다고 보장할 수 없음
- 모든 입력 값의 조합으로 테스트 할 수 없음
  - 너무 많은 경우의 수가 존재
- 테스트 샘플이 중요함
  - 모든 입력 경우를 대표할 수 있는 샘플을 선택하여 테스트를 진행해야 함
  - 여론조사와 비슷

테스트 스위트

- **여러 개의 테스트 케이스를 사용해야 함**
- **소프트웨어의 정확성을 확신할 수 있도록 조심스럽게 선택되어야 함**
- **테스트 스위트**
  - 테스트 케이스의 집합
- **효과적인 테스트 스위트는 정확성을 보장하지는 못하지만 많은 소프트웨어 오류를 잡아낼 수 있음**

테스트 케이스

- **테스트 케이스는 소프트웨어 개발과정의 단계마다 구성되어 실행 되어야 함**
  - 모듈 개발 단계, 서브 시스템 개발 단계, 전체 시스템 개발 단계
- **테스트 케이스는 소프트웨어 개발 중에 반복적으로 실행됨**
  - 새로운 기능을 추가하거나 프로그램을 수정할 때마다 회귀 테스트를 실행해야 함.
- **소프트웨어 품질관리팀에서 전문적으로 수행**
- **테스트의 종류**
  - 화이트박스 테스트
  - 블랙박스 테스트

❖ **프로그램의 내부 구조를 아는 경우에 시행하는 테스트**

- 구조 테스트
- 프로그래머가 직접 테스트 수행
- 경로 테스트가 대표적인 화이트박스 테스팅

❖ **화이트 박스 테스트 스위트**

- 명령문 적용범위라고 불림
- 프로그램의 모든 명령어가 테스트 스위트의 적어도 하나의 테스트 케이스에 의해 최소 한번은 실행되어야 함

가능한 경로

테스트 케이스 1) 1-2-3-5
테스트 케이스 2) 1-2-3-4-5
테스트 케이스 3) 1-2-3-4-4-5
테스트 케이스 4) 1-2-3-4-4-4-5

각 테스트 케이스 별로 해당하는 입력 값들을 정하고 테스트 수행

테스트 케이스 1) 전원버튼-앱버튼-5-녹색버튼
테스트 케이스 2) 전원버튼-앱버튼-5-3-녹색버튼
…

테스트 수행 후 나오는 결과가 예상하는 내용과 같은지 비교

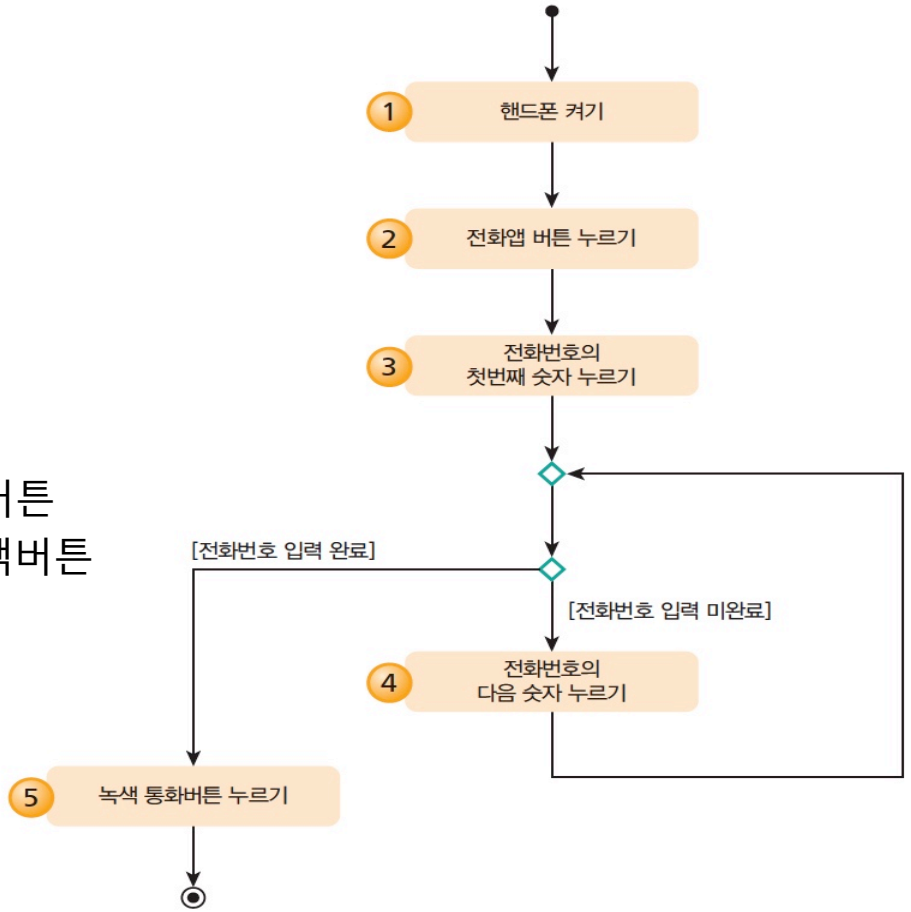


그림 9.9 전화를 거는 활동 다이어그램

❖ 소프트웨어에 입력되는 경우들을 균등하게 여러 그룹으로 나누어 각 그룹의 대표 값을 입력하여 테스팅 함

그림 9.11 피자 자르기는 균등 분할과 같다.

❖ 블랙박스 테스트

- 장점
  - 프로그래밍 경험이 거의 또는 전혀 필요 없음
  - EX) 주사위 게임 '크랩' (그림 9.10 참조)
  - 모든 요구사항은 명세서로 주어짐

일련번호: C1
이름: 컴 아웃(Come Out) 주사위 던지기
동작: 크랩게임은 두 개의 주사위를 한번 이상 던지는 것으로 구성되는 게임이다. 게임을 시작할 때 주사위를 던지는 것을 컴 아웃(Come Out) 주사위던지기라고 한다. 만약 주사위 결과가 2, 3, 12 인 경우 크랩 아웃에 의해 플레이어는 지게 된다. 만약 주사위 결과가 7 또는 11 인 경우 사용자는 내츄럴 위너(natural winner)가 된다. 만약 사용자가 다른 결과를 내게 된다면(예를 들면 4, 5, 6, 8, 9, 10), 이 주사위는 포인트가 되며 다음 주사위 던지기는 포인트 롤(point roll)이 된다.

일련번호: C2 :
이름: 포인트 롤
동작: 플레이어는 주사위를 다시 던진다. 만약 두 주사위의 합이 7이라면 플레이어는 세븐 아웃에 의해 게임을 지게 된다. 만약 주사위의 합이 포인트(컴 아웃 주사위 던지기에서 던졌던 주사위의 합)와 같다면, 플레이어는 포인트를 맞추었으므로 게임에서 이기게 된다. 만약 두 주사위의 합이 7이나 포인트와 다른 값이라면 포인트 롤은 계속 반복된다.

일련번호: C3
이름: 플레이 계속
동작: 크랩 아웃, 당연 승리, 세븐 아웃, 내츄럴 위너, 포인트 맞추기가 나오면, 플레이어는 새로운 게임을 시작할 수 있다(새로운 게임을 시작하는 내용은 요구사항 C1을 참조한다).

그림 9.10 크랩 게임의 요구사항들

- 차이점
  - 테스트 선정 방식에 따라 달라짐
  - 모든 테스트는 소프트웨어 요구사항에 따라서 작성됨
  - 명령어 / 경로 적용범위와 같은 기법들을 사용할 수 없음

❖ 자동차 안정성 테스트

- 경우의 수
  - 차가 움직이지 않음
  - 차가 움직임
  - 차가 최대 속력으로 움직임

테스트 케이스	입력 조건	예상 결과
차가 움직이지 않음	차량 속도 = 0	차가 움직이지 않음
차가 움직임	차량 속도 = 60 km/h	차가 뒤집히지 않음
차가 최대 속력으로 움직임	차량 속도 = 180 km/h	차가 뒤집히지 않음

❖ **분할된 입력 그룹에 따라 각 그룹의 다음 값들을 테스트**

- 경계선에 있는 값을 입력으로 하는 테스트
- 경계선 안쪽에 있는 값을 입력으로 하는 테스트
- 경계선 밖에 있는 값을 입력으로 하는 테스트
- (경계선 안에 있으며) 좀 더 일반적인 값을 입력으로 하는 테스트

❖ **균등분할 테스트와 같이 진행하는 경우가 많음**

❖ 차량 안정성 테스트

테스트 케이스	입력 조건	예상되는 행동
최소값에서	차량 속도 = 0 mph	차는 움직이지 않는다.
최소값 + 1	차량 속도 = 1 mph	차가 뒤집히지 않는다.
최소값 −1	차량 속도 = −1 mph	차의 뒤집힘 방지 소프트웨어 작동하지 않는 상태로 후진한다.
최대값	차량 속도 = 150 mph	차가 뒤집히지 않는다.
최대값 − 1	차량 속도 = 149 mph	차가 뒤집히지 않는다
최대값 + 1	차량 속도 = 151 mph	차량은 지나치게 고속으로 운전하고 있다는 경고 메시지를 준다.
일반적인 경우	차량 속도 = 70 mph차량은 뒤집히지 않는다.	

그림 9.14 자동차 속도만을 고려한 자동차 안정성 소프트웨어에 대한 테스트 스위트

9.8 언제 사용할 것인가?

❖ **소프트웨어 테스팅시 사용**

❖ **일상의 다양한 분야에서 사용가능**

- 장난감 제품 조립
- 부하직원 평가

  - 극단 조건 (경계값)에 집중하여 평가를 진행할 수 있다.

- 자동차 구매

  - 자동차 구매시 차량에 대한 원하는 부분의 요구사항 명세서를 만든 후, 각 명세서의 요구사항 별로 다양한 자료를 수집하여 비교 분석할 수 있다.

Computational Thinking

11장-동시적 행동

학습개요

- 11장 동시적 행동
  1. 병렬성 또는 동시성?
  2. 스케쥴링
  3. 정렬 연결망 (스킵)
  4. 동시성 효과 측정하기 (스킵)
  5. 동시성에서의 해결 과제
  6. 추가자료 - 2018 IT 트렌드, 정보보호 등

❖ **동시적 실행 (concurrent execution), 동시성 (concurrency)**

- 동시에 여러 개의 작업들을 수행

❖ **멀티코어 프로세서 (multi-core processors)**

- 하나의 집적회로에 여러 개의 프로세서들을 포함

❖ **병렬 실행(parallel execution)**

- 여러 개의 장치들에 의한 동시적인 실행은 각 장치들이 병렬로 실행

❖ **수퍼컴퓨터 (supercomputer)**

- 속도 : 페타플롭 (petaflops) (초당 100만의 4제곱개의 명령어들이 실행)
- 수백만 개의 프로세서들을 동시에 실행

❖ 분산컴퓨팅 (distributed computing)

- 인터넷에서의 병렬 실행
- 공통된 문제를 해결하기 위해 여러 개의 대규모의 독립적인 컴퓨터들이 집단적으로 작업

❖ 그리드 컴퓨팅 (grid computing)

- 분산컴퓨팅. 일기예보와 금융 모델링에서 사용

❖ 클라우드 컴퓨팅 (cloud computing)

- 대용량의 공유 데이터를 포함하는 분산컴퓨팅

❖ 동시성(concurrency)

- 소프트웨어 시스템과 관련

❖ 병렬성(parallelism)

- 하드웨어 형태의 동시적인 실행

❖ 스케쥴링(scheduling)

- 어떤 시점에 어느 프로세서에서 작업들을 실행할 것인지를 결정

❖ 다중작업

- 장점 : 시간을 절약할 수 있다는 것이다.
- 더 많은 프로세서가 항상 성능을 향상시키는 것은 아님

Ex) 경기1(프로세스1)과 경기2 (프로세스2) 의 승자가 결정된 후에
경기3 (프로세스3) 은 진행되야 하는 경우

- 작업이 계획될 때 프로세스들 사이의 이러한 종속관계를 고려해야 한다.

그림 11.4 8개의 팀의 단일 예선에 대한 토너먼트 묶음

그림 11.5 4개의 경기장에서 진행된 8개의 팀에 대한 토너먼트의 시간 도식

❖ **성능 개선**

❖ **여러 개의 경기장(다중 프로세서)에서 7개 경기(7개의 작업)가 진행되는 경우**

→ 9시간 만에 종료

❖ **동일한 경기장(단일 프로세서)에서 모든 게임(21개의 작업)이 진행**

→ 21시간(3시간 x 7개의 경기)만에 종료

→ 따라서 다중작업을 사용하면 $2\frac{1}{3}$ 배 빠름

❖ **100개의 프로세서들은 단일 프로세서보다 100배 빨리 작업을 완료할 수 있을까?**

❖ **공유 자원**

- 동시적으로 프로세서들이 실행을 할 때 제한
- 프로세서들은 각각의 입력 데이터 값이 사용가능하게 될 때까지 실행을 시작할 수 없음
- 하지만 이러한 값들은 다른 프로세서들(적어도 이전에 이 값들을 비교했던 프로세서들)과 공유되는 일종의 공유 자원

- 동시성 제약을 지키지 못하면 다중작업이 실패함
- 동시적인 읽기 접근 OK
- 한 명의 사용자가 문서를 읽고 있을 때 누군가 문서를 변경하고 있다면?
- 공유 문서에 대한 여러 개의 동시적인 편집(갱신)이 있다면?

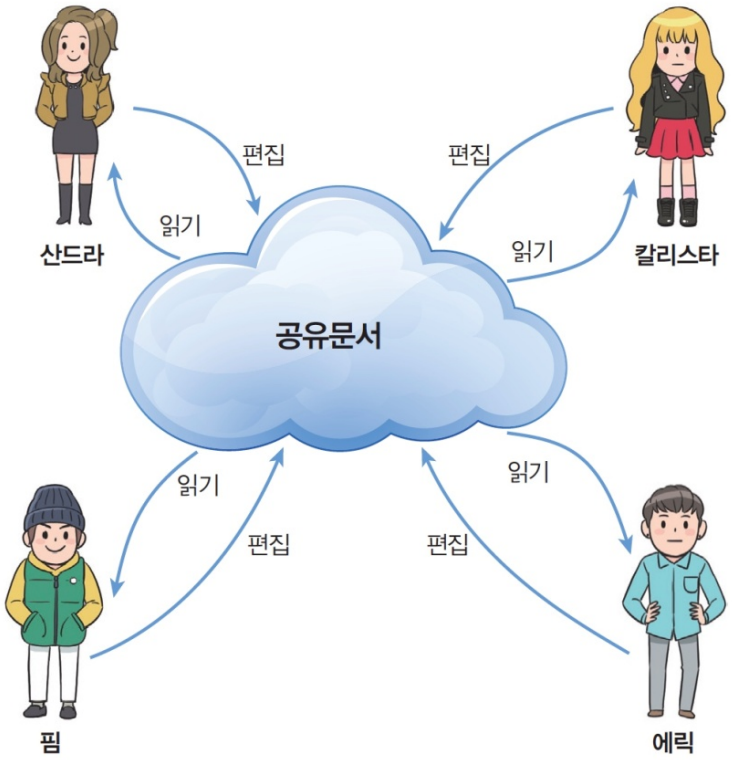


그림 11.11 4명의 사람들이 공유하는 문서 시스템

❖ **TOCTOU(time-of-check time-of-use: 확인시간 사용시간) 조건, 경쟁조건(race condition)**

- 공유자원에 대한 동시적인 갱신 접근이 이루어질 때 발생
- TOC(time-of-check): 프로세서가 자원의 상태를 확인하는 시간
  문서의 제목을 읽는 것
- TOU(time-of-use): 프로세서가 자원의 상태를 갱신하는 시간
  제목을 편집하는 것
- **회피해야 할 TOCTOU 상황** : 첫 번째 프로세스가 공유자원에 대한 확인을 수행하고 자신의 갱신을 완료하기 전에, 두 번째 프로세스가 동일한 자원에 대하여 확인과 갱신을 둘 다 수행할 때

❖ 자원을 잠금

- 일단 하나의 프로세서가 확인을 개시하면, 그 자원은 갱신이 완료될 때까지 잠금이 풀리지 않음
- 다른 프로세서들은 잠긴 자원에 어떤 식으로든 접근할 수 없음
- 잠금을 수행한 프로세스만 잠금을 풀 수 있음
- 동시성에 대한 제약

그림 11.12 경쟁조건을 경험하는 두 대의 자동차

❖ 데드락(deadlock)

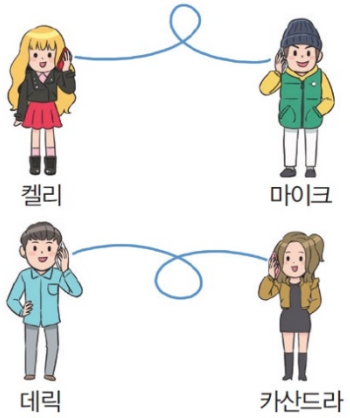
- 프로세서들이 자원에 대해 계속해서 진행할 수 없는 방법으로 잠금을 할 때 데드락이 발생
- 하나 이상의 자원을 해제하지 않고는 프로세서들이 실행하는 것이 불가능

단계 1

켈리는 마이크와 카산드라에게 전화회의를 정하려고 한다.
데릭도 또한 마이크와 카산드라에게 전화회의를 하려고 한다.

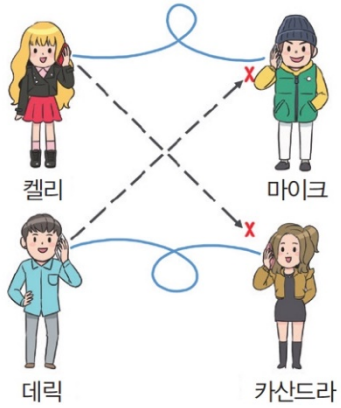
단계 2

켈리는 마이크에게 전화를 걸고 응답을 받는다.
동시에 데릭은 카산드라에게 전화를 걸고 응답을 받는다



단계 3

켈리는 카산드라에게 전화걸기를 시도하고
데릭은 마이크에게 전화걸기를 시도한다.
데드락 상태에 빠진다 – 전화회의를 완성하지 못하게 된다.



❖ **그리드락(grid lock)**

- 교통량에 데드락이 발생할 때

❖ **라이브락(live lock)**

- 일부 프로세서들에게만 데드락이 발생
- 컴퓨터 시스템에서 프로세서들이 자원을 공평하게 부여받지 못할 때 발생
- 일부 특정한 프로세서나 특정한 작업은 때때로 낮은 우선순위가 배정되어서 더 이상 진행되지 못하게 됨

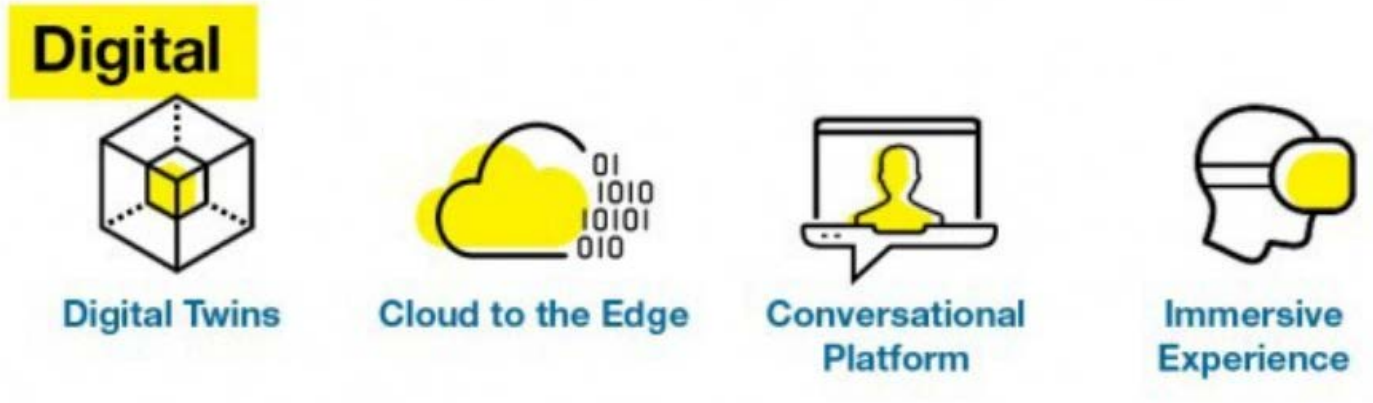
그림 11.14 그리드락 상태에 있는 6대의 자동차

❖ 크라우드 소싱(crowd sourcing)

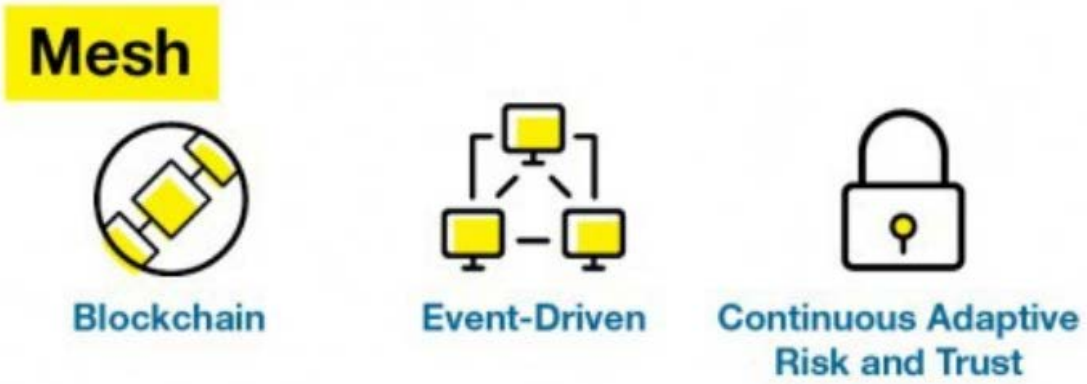
- 크라우드 소싱 해법에서 사람들은 동시성 프로세서의 역할을 함
- 그룹(크라우드)에 있는 모든 사람들은 동일한 문제가 주어지고 문제를 해결하는 과정에서 집단적으로 공유

2018 가트너 선정 10대 전략 기술 트랜드

- **인공지능 강화 시스템**
- **지능형 앱·분석** - 모든 앱과 애플리케이션, 서비스들이 일정 수준의 AI를 포함
- **지능형 사물**



- 디지털 트윈 - 현실 세계에 존재하는 대상이나 시스템의 디지털 버전
- 클라우드에서 엣지로
  - ✓ 클라우드 - 서비스 중심 모델과 중앙 집중화 제어, 조정 구조를 형성
  - ✓ 엣지 - 클라우드 서비스 측면의 비연결, 비분산 프로세스 실행을 가능하게 하는 스타일로 사용
- 대화형 플랫폼 -사용자에게 질문이나 명령을 받은 후, 기능을 수행, 콘텐츠를 제시, 추가 인풋을 요청하는 방향으로 일을 처리
- 몰입 경험 - 혼합 현실(Mixed Reality)은 가상현실(VR)과 증강현실(AR) 기술을 통합, 확장하는 몰입 유형



- 블록체인 - 디지털 통화 인프라에서 디지털 혁신 플랫폼으로 진화 중
- 이벤트 기반 모델 - 디지털 비즈니스의 핵심은 매 순간 이벤트를 감지하고, 매 순간을 활용
- 지속적이며 적응할 수 있는 리스크 및 신뢰 평가(CARTA) 접근법

- 한층 정교한 타깃 공격이 가능해진 세상에서 디지털 비즈니스 이니셔티브를 안전하게 추진할 수 있도록 리스크 및 신뢰 평가(CARTA) 접근법 활용

정보보호

정보통신기술발전의 순기능과 역기능

유출된 개인정보의 위험성

2017 정보보호 10대이슈

1

강대국간 사이버 공방 심화
사이버 전면전 위험 고조

2

사이버위협정보 공유와 협력 확대
대응이 빨라진다

3

돈을 노린 랜섬웨어 공격
사이버범죄 주류에 등극

4

빅데이터·AI·클라우드 활용 사이버보안
패러다임이 바뀐다

2017 정보보호 10대 이슈

Security based Industry & Life

5

분산저장기술 '블록체인'
이론에서 현실로

6

다양화되는 바이오인증
사용자 인증의 대세로

7

보안 고려없는 사물인터넷(IoT)
커져가는 일상의 위험

8

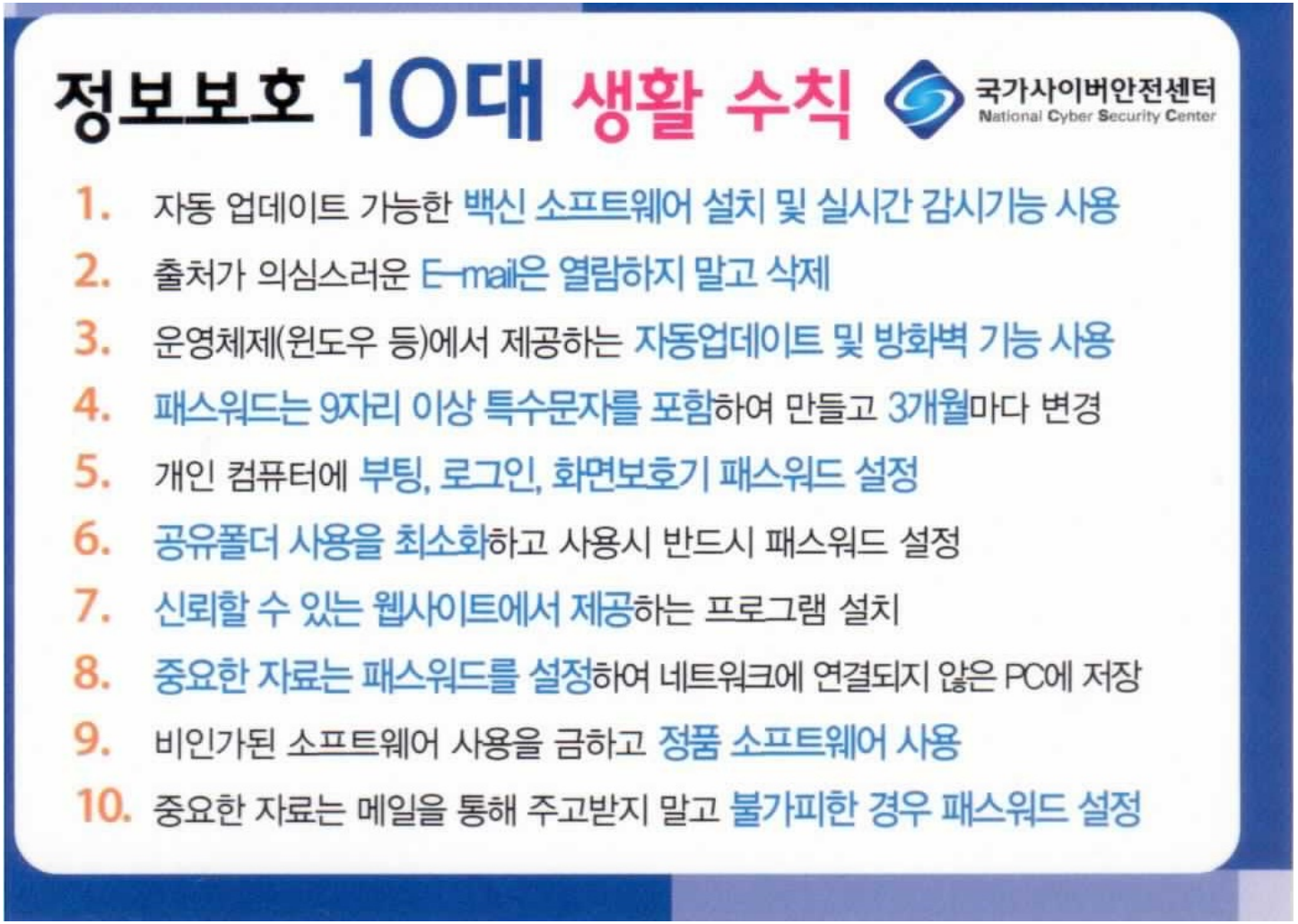
활성화되는 커넥티드 카의 안전띠
사이버보안

9

잊힐 권리 보장
강화되는 개인정보 자기결정권

10

개인정보 보호와 활용의 조화
4차 산업혁명을 좌우한다


정보보호 10대 생활 수칙
국가사이버안전센터
National Cyber Security Center
1. 자동 업데이트 가능한 백신 소프트웨어 설치 및 실시간 감시기능 사용
2. 출처가 의심스러운 E-mail은 열람하지 말고 삭제
3. 운영체제(윈도우 등)에서 제공하는 자동업데이트 및 방화벽 기능 사용
4. 패스워드는 9자리 이상 특수문자를 포함하여 만들고 3개월마다 변경
5. 개인 컴퓨터에 부팅, 로그인, 화면보호기 패스워드 설정
6. 공유폴더 사용을 최소화하고 사용시 반드시 패스워드 설정
7. 신뢰할 수 있는 웹사이트에서 제공하는 프로그램 설치
8. 중요한 자료는 패스워드를 설정하여 네트워크에 연결되지 않은 PC에 저장
9. 비인가된 소프트웨어 사용을 금하고 정품 소프트웨어 사용
10. 중요한 자료는 메일을 통해 주고받지 말고 불가피한 경우 패스워드 설정

참고문헌

- 가트너 선정 2018년 10대 전략 기술 트렌드는?, 디지털데일리, http://www.ddaily.co.kr/news/article.html?no=161253 (2017, 10)
- 가트너, https://www.gartner.com
- 온라인정보보호, KISA, https://www.i-privacy.kr
- 정보보호 10대 생활수칙, 국가사이버안전센터
- 2017년 정보보호 10대 이슈 전망, KISA