Correctness by Construction

최진영

고려대학교 정형기법 연구실

2009. 7. 9

목차


CAD DRWING
2007.11.01
2F PLAN
Ando-tadao(安藤忠雄)
1/200(A3)


△잠실 '제2롯데월드'(112층 555m)
△송도 '인천타워'(151층 610m)
▽상암동 '국제비즈니스센터'(130층 580m)
▽부산 '월드비즈니스센터'(110층 500m)

에러의 의한 비용

	설치 단계 발견	코딩 단계 발견	통합 단계 발견	베타제품에서 발견	개발완료/제품 출시 후 발견
설계과정 에러	1x	5x	10x	15x	30x
코딩과정 에러		1x	10x	20x	30x
통합과정 에러			1x	10x	20x

- **미 상무성 산하 국립기술표준연구소(NIST) 조사 자료**
  - **설계과정**에서 만들어지는 에러가 **개발완료 후에** 발견된다면, 설계단계에서 에러를 수정하는 비용보다 **30배 추가지출 발생**
  - **통합과정**에서 만들어지는 에러가 **개발완료 후에** 발견된다면, 통합과정에서 수정하는 비용보다 **20배 추가지출 발생**.
    - ※출처 : http://www.nist.gov/director/prog-ofc/report02-3.pdf
      - The Economic Impacts of Inadequate Infrastructure for Software Testing

Remarks by Bill Gates

17th Annual ACM Conference on Object-Oriented Programming, Seattle, Washington, November 8, 2002

- "… When you look at a big commercial software company like Microsoft, there's actually as much testing that goes in as development. We have as many testers as we have developers. Testers basically test all the time, and developers basically are involved in the testing process about half the time…"
- "… We've probably changed the industry we're in. We're not in the software industry; we're in the testing industry, and writing the software is the thing that keeps us busy doing all that testing."
- "…The test cases are unbelievably expensive; in fact, there's more lines of code in the test harness than there is in the program itself. Often that's a ratio of about three to one."

간단한 퀴즈

A simple program

Input : Read three integer values from the command line. The Three values represent the length of the sides of a triangle.

Output : Tell whether the triangle is

- 부등변삼각형 (Scalene) : no two sides are equal
- 이등변삼각형(Isosceles) : exactly two sides are equal
- 정삼각형 (Equilateral) : all sides are equal

Create a Set of Test Cases for this program.

Solution – 1 Point for each Correct Answer

- Q1 : (4, 2, 1) a invalid triangle
- Why not a valid triangle? (a,b,c) with a > b + c
- Define valid triangles: a ≤ b + c

Solution – 1 Point for each Correct Answer

- Q2 : Some permutations of previous (1,2,4), (2,1,4)
- Fulfill above definition, but are still invalid.
- Patch definition of valid triangles:

$a \leq b + c$ and $b \leq a + c$ and $c \leq a + b$

Solution – 1 Point for each Correct Answer

- Q3 : (4,2,2) a invalid triangle with equal sum
- Fulfill above definition, but is invalid.
- Patch definition of valid triangles:

a < b + c and b < a + c and c < a + b

Solution – 1 Point for each Correct Answer

- Q4 : some permutations of previous (2,2,4), (2,4,2)

Solution – 1 Point for each Correct Answer

Q5 : (3,4,5) a valid scalene triangle

Solution – 1 Point for each Correct Answer

- Q6 : (3,3,3) an equilateral triangle

Solution – 1 Point for each Correct Answer

- Q7 : (3,4,3) a valid isosceles triangle

Solution – 1 Point for each Correct Answer

- Q8 : all permutations of valid isosceles triangles

  (3,4,3), (3,3,4), (4,3,3)

Solution – 1 Point for each Correct Answer

- Q9 : one side with zero values (0,4,3)

Solution – 1 Point for each Correct Answer

- Q10 : one side with negative values (–1,4,3)

Solution – 1 Point for each Correct Answer

- Q11 : all sides zero (0,0,0)

Solution – 1 Point for each Correct Answer

- Q12 : at least one value is non-integer (1,3,2.5)

Solution – 1 Point for each Correct Answer

- Q13 : wrong number of arguments (2,4) or (1,2,3,3)

Solution – 1 Point for each Correct Answer

- Q14 (the most important one):
  - Did you specify the expected output in each case?

Solution – 1 Point for each Correct Answer

- Q1-13 correspond to failure that have actually occurred in implementation of the program.
- How many questions did you answer?
- Highly qualified, experienced programmers score 7.8 on average

Specification: Practice

Example: A sorting Program

• Public static Integer [ ] sort(Integer [ ] a)
{ . . . }

• Testing sort ():
– sort({3,2,5}) == {2,3,5}
– sort({}) == {}
– sort ({17}) == {17}

• Specification
– Requires: a is an array of integers
– Ensures: returns the sorted argument array a

Example Cont' d

• Example: A sorting Program

• Public static Integer [ ] sort(Integer [ ] a)
{ . . . }

• Specification
– Requires: a is an array of integers
– Ensures: returns the sorted argument array a

Is this a good specification?

Sort({2,1,2)} == {1,2,2,17}

Example

• Example: A sorting Program

• Public static Integer [ ] sort(Integer [ ] a)
{ . . . }

• Specification
– Requires: a is an array of integers
– Ensures: returns a sorted array with only elements from a

Sort({2,1,2)} == {1,1,2}

Example

• Example: A sorting Program

• Public static Integer [ ] sort(Integer [ ] a)
{ . . . }

• Specification
– Requires: a is an array of integers
– Ensures: returns a permutation of a that is sorted

Sort(null) throws NullPointerException

Example

• Example: A sorting Program

• Public static Integer [ ] sort(Integer [ ] a)
{ . . . }

• Specification
– Requires: a is a non-null array of integers
– Ensures: returns the unchanged reference a containing a permutation of the old contents of a that is sorted.

Design-by-Contract

- For each class method, a **precondition** and a **postcondition** are specified.
- Whenever a class method is called, the client and the method are bound by a contract. The **client** guarantees to satisfy the precondition at the point of call; in return the **method** guarantees to satisfy the postcondition at the point of return
- To assist in ensuring that contracts are honoured, additional elements such as **class invariants** and **assertions** may be included in the specification

배경

- **보안 필수**(Security-Critical), **안전 필수**(Safety-Critical) 시스템
  - 오류 발생으로 인해 치명적인 문제점을 유발
  - **낮은 오류율**(Low-Defect Rate)이 보장되어야 함
- 인증
  - 개발된 시스템이 보안 속성(security property)또는 안전성 속성(Safety property)을 만족함을 보여야 한다.
  - 많은 인증들이 속성이 만족함 보증하는 **증거** 제시를 요구

Correctness by Construction

- 안전필수, 보안필수 시스템을 위한 근본적이며, 효과적이며, 경제적인 개발 방법론
  - 고무결성(High Integrity) 시스템 개발을 보장
  - PRAXIS에 의해 개발
  - <-> Build and Debug
- TOKENEER
  - 높은 생산성(38loc/day)과 낮은 오류율(1개 defect)
  - CC EAL5+ 인증

Correctness by Construction

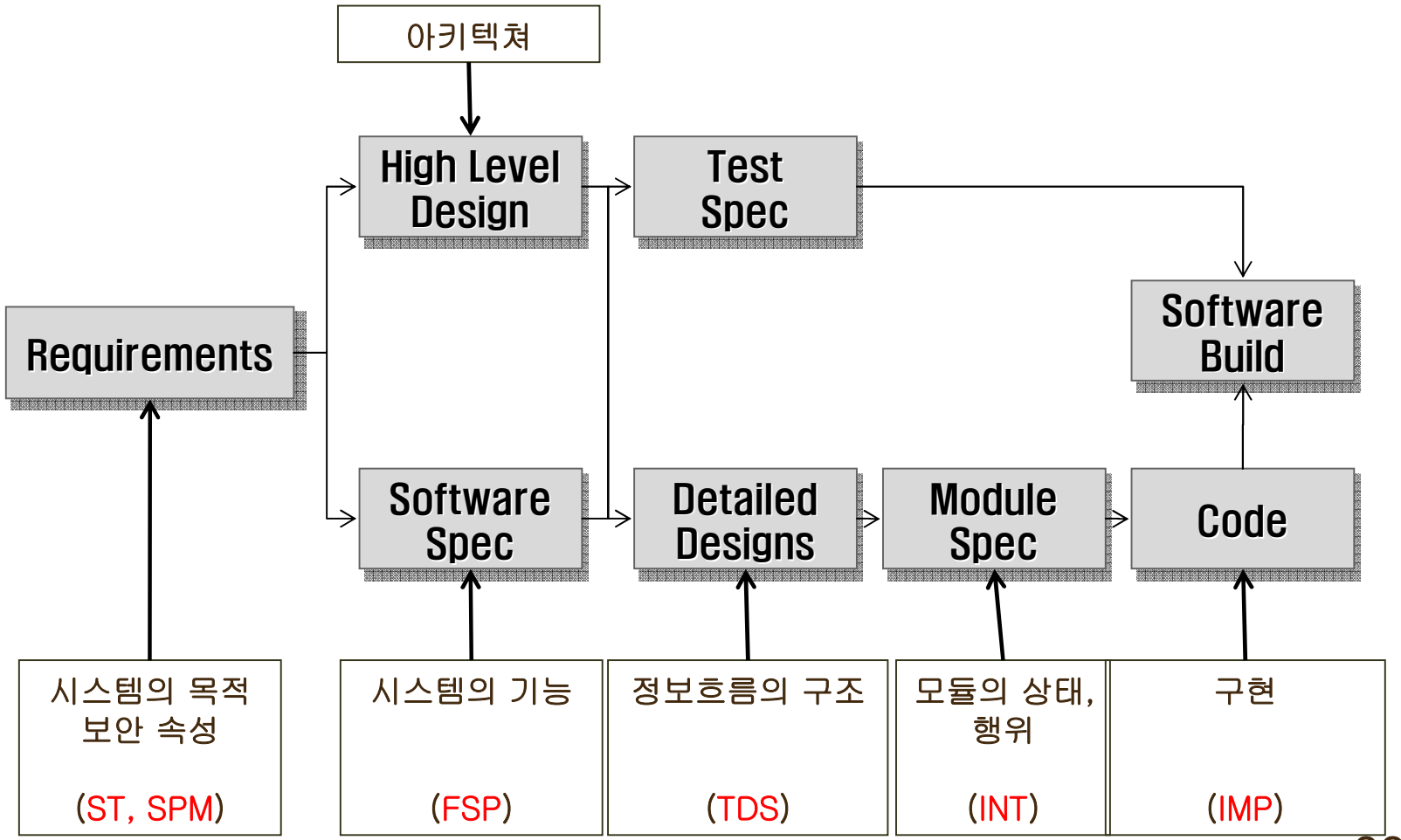
- 기본 원리
  - 에러가 발생하기 힘들도록 개발한다
  - 에러를 발견 시 발견된 단계의 모든 오류들을 제거한다.
  - 개발 시 개발 프로세스의 산출물로써 자연스럽게 프로젝트 목적에 부합되는 증거(Evidence)를 생성한다

Correctness by Construction

- 전략
  - 정형적인 표현법을 사용한다
    - 표현의 애매모호함은 오류 도출의 주요 요인 중 하나
  - 각 단계의 산출물마다 검증한다
    - 에러 발생 초기에 검출 가능 점진적인 개발을 한다
    - 오류의 근원을 찾기가 쉬움
  - 반복을 피한다
    - 반복하여 언급할 시 일관성이 지켜지지 않을 수 있음
    - 개발 단계마다 다른 관점의 정보들을 추가
    - 소프트웨어 명세 – 기능 명세, 하이레벨 디자인 – 아키텍쳐
  - 간단하게 작성한다
    - 쉽게 검증할 수 있도록 개발
  - 어려운 부분부터 먼저 수행한다
    - 복잡하고 이해하기 힘든 부분에 잠재적인 오류가 존재할 확률이 높음
  - 심사숙고 한다
    - 시스템 목적에 적합한 방법과 툴을 사용

Correctness by Construction

- 개발 단계

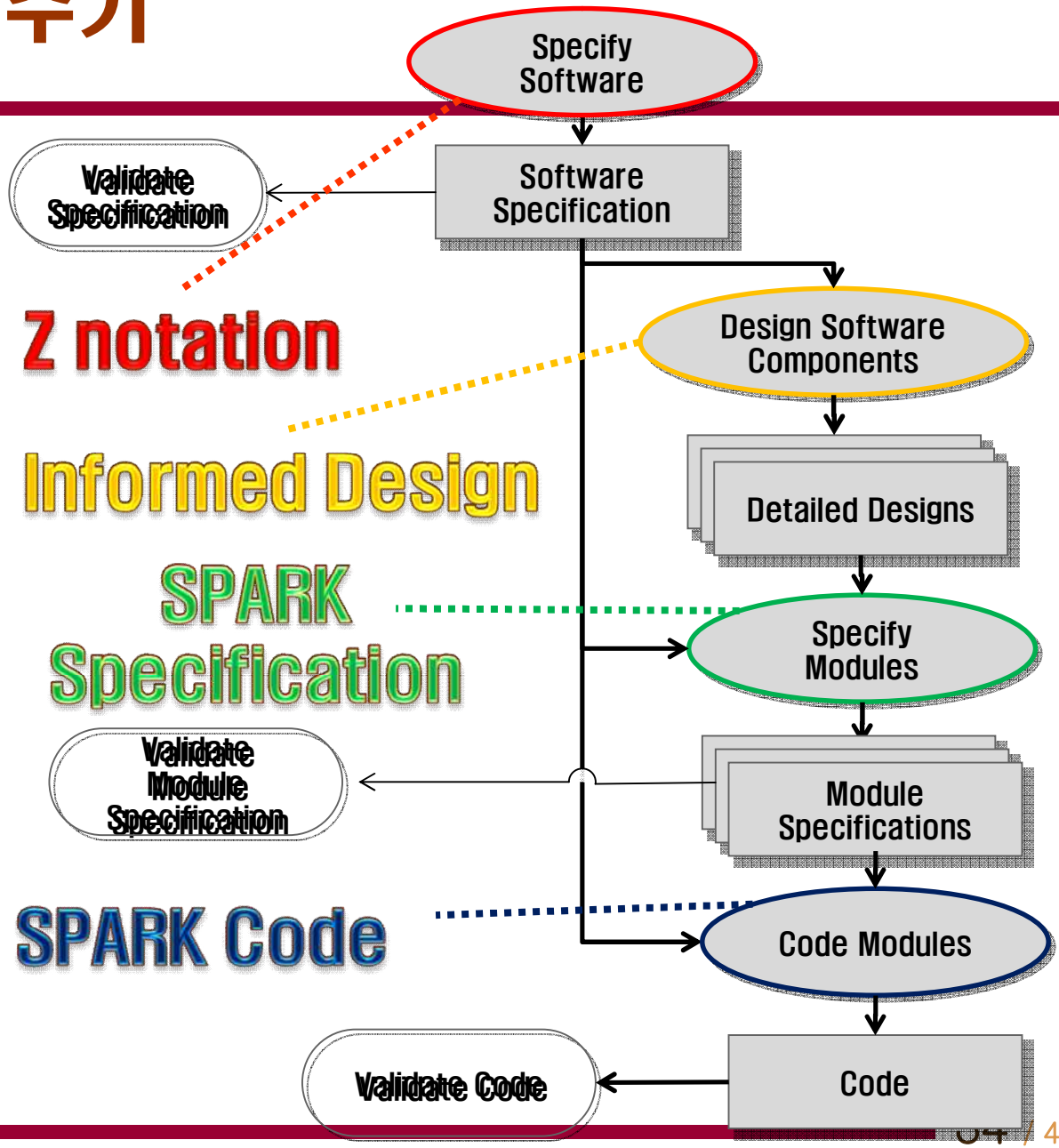


Correctness by Construction

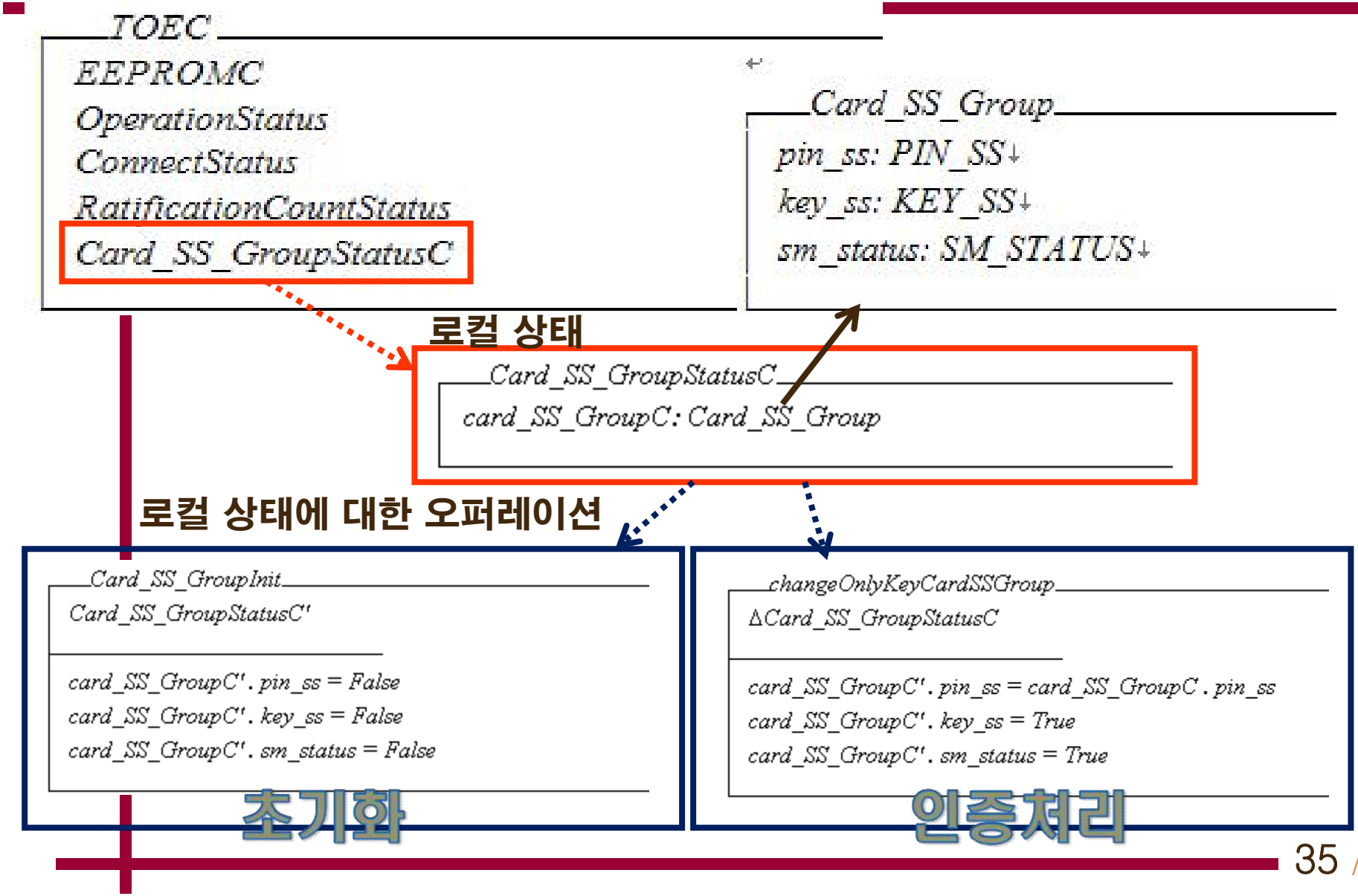
- 특징
  - 병행적 개발(Parallelism)
    - 두 개의 다른 독립적인 단계의 병행적인 개발이 가능.
  - 오류 근원 분석(Root cause analysis)
    - 오류가 발견된 단계에서 뿐만 아니라 오류가 시작된 단계까지 거슬러 가며 수정.
  - 프로세스 개선(Process improvement)
    - 오류가 어떻게 발생되었는지를 분석하여 같은 오류가 다시는 발생되지 않도록 예방한다.
  - 융통성(Flexibility)

KCOS 개발 주기

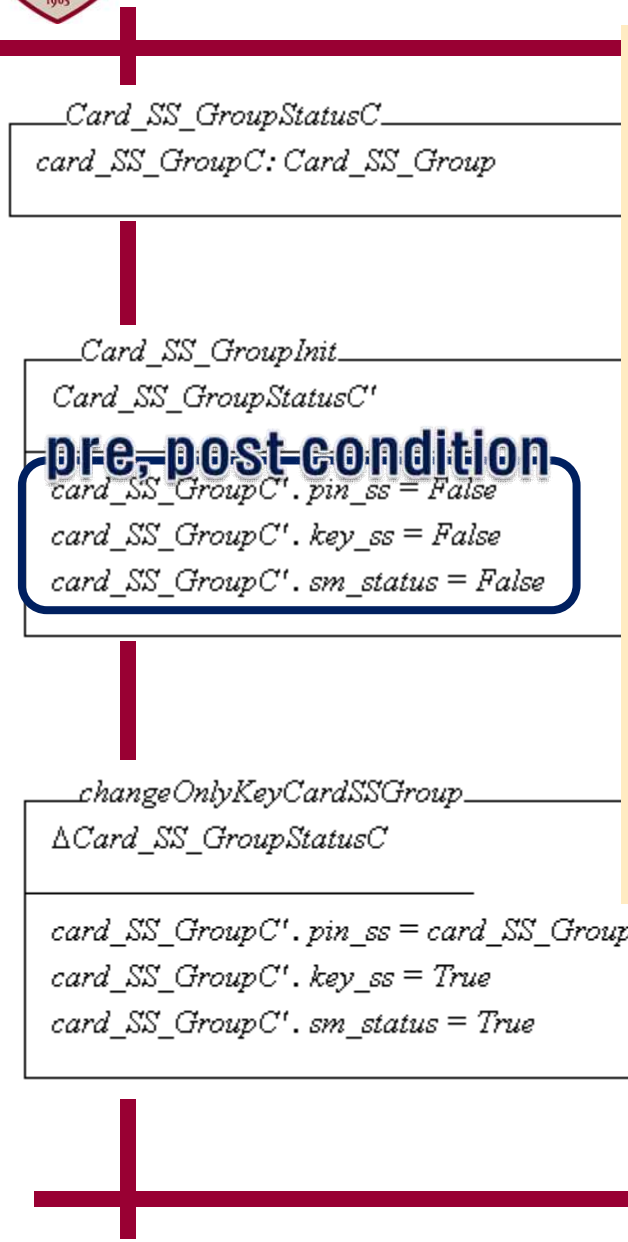

KCOS
merchants BANK
DEBIT
4000 0000 0000
12/05
VISA
JANE LYNX


Specify Software
Software Specification
Validate Specification
Z notation
Design Software Components
Informed Design
Detailed Designs
SPARK
Specify Modules
Specification
Validate Module Specification
Module Specifications
SPARK Code
Code Modules
Validate Code
Code

Software Specification(FSP)



Module Specification(INT)



pre, post condition

```
package Card_SS_GroupStatus
--# own Card_SS_Group : Card_SS_GroupT;
--# initializes Card_SS_Group;
is

procedure Init;
--# global out Card_SS_Group;;
--# derives Card_SS_Group from Card_SS_Group;
--# post Card_SS_Group.pin_ss = false and
--# Card_SS_Group.key_ss = false and
--# Card_SS_Group.sm_status = false;

procedure changeOnlyKeyCardSSGroup;
--# global in out Card_SS_Group;
--# derives Card_SS_Group from Card_SS_Group;
--# post Card_SS_Group.pin_ss = Card_SS_Group.pin_ss and
--# Card_SS_Group.key_ss = true and
--# Card_SS_Group.sm_status = true;
end Card_SS_GroupStatus;
```

For information flow analysis

Code-verification
(pre, post condition)

SPARK Code(IMP)

```
package body Card_SS_GroupStatus is

        Card_SS_Group : Card_SS_GroupT;

        procedure Init is
        begin
                Card_SS_Group.pin_ss := false;
                Card_SS_Group.key_ss := false;
                Card_SS_Group.sm_status := false;
        end Init;

        procedure changeOnlyKeyCardSSGroup is
        begin
                Card_SS_Group.pin_ss :=
                        Card_SS_Group.pin_ss;
                Card_SS_Group.key_ss := true;
                Card_SS_Group.sm_status := true;
        end changeOnlyKeyCardSSGroup;
begin
        Card_SS_Group.pin_ss := false;
        Card_SS_Group.key_ss := false;
        Card_SS_Group.sm_status := false;
end Card_SS_GroupStatus;
```

Code verification
(pre, post condition)

SPARK Code Verification

```
**********************************************************
                Semantic Analysis of SPARK Text
      SPARK95 Examiner with VC and RTC Generator Release 7.6 / 06.08
                        Demonstration Version
**********************************************************


     CREATED 18-JUN-2009, 14:25:46  SIMPLIFIED 18-JUN-2009, 14:25:51
             (Simplified by SPADE Simplifier, Demo Version)

                  procedure Card_SS_GroupStatus.Init



For path(s) from start to finish:

procedure_init_1.
*** true .          /* all conclusions proved */
```

```
**************************
 of SPARK Text
 Generator Release 7.6 / 06.08
) Version
**************************
```

```
     CREATED 18-JUN-2009, 14:25:46  SIMPLIFIED 18-JUN-2009, 14:25:50
             (Simplified by SPADE Simplifier, Demo Version)

          procedure Card_SS_GroupStatus.changeOnlyKeyCardSSGroup



For path(s) from start to finish:

procedure_changeonlykeycardssgroup_1.
*** true .          /* all conclusions proved */
```

결론 및 향후 연구

결론

- 명세 단계에서 검증된 속성을 코드 단계에서 유지
- 인증에 필요한 평가물로써 각 단계에서 도출된 산출물 개발

향후 연구

- SPM을 명세하여 명세부터 코드까지 검증
- 테스팅을 통한 고무결성 속성 확인
- 보안/안전 필수 시스템에 적용

Why Spark ADA?

- Sound
  "No false negatives" / "There are no Bugs"
  SPARK을 사용하면 Bug가 없다는 것을 주장할 수 있음.
- Low False Alarm Rate
  SPARK은 false alarm이 거의 발생하지 않는다고 주장.
  다른 정적 도구 및 언어에 비해 좋다고 합니다.
- Depth
  SPARK 분석기는 복잡한 코드 및 요소들을 검증할 수 있음
- Fast
  Toolset은 개발하는 과정에서 쌍방으로 사용될 수 있음.
- 이것은 Testing과 Compiling을 보다 빠르게 하기 위해 디자인 됨
- Modular
  결과들을 얻는데 있어서 전체 프로그램을 분석할 필요가 없다고 함.

- Analysis tool을 사용하면 다음과 같은 프로그래밍 에러들에 대항할 수 있다고 함.
  - Improper input validation
  - Buffer overflow errors
  - Improper initialization
  - Information leak