소프트웨어 안전성 분석
Software Safety Analysis: Framework and Process

최 윤 자

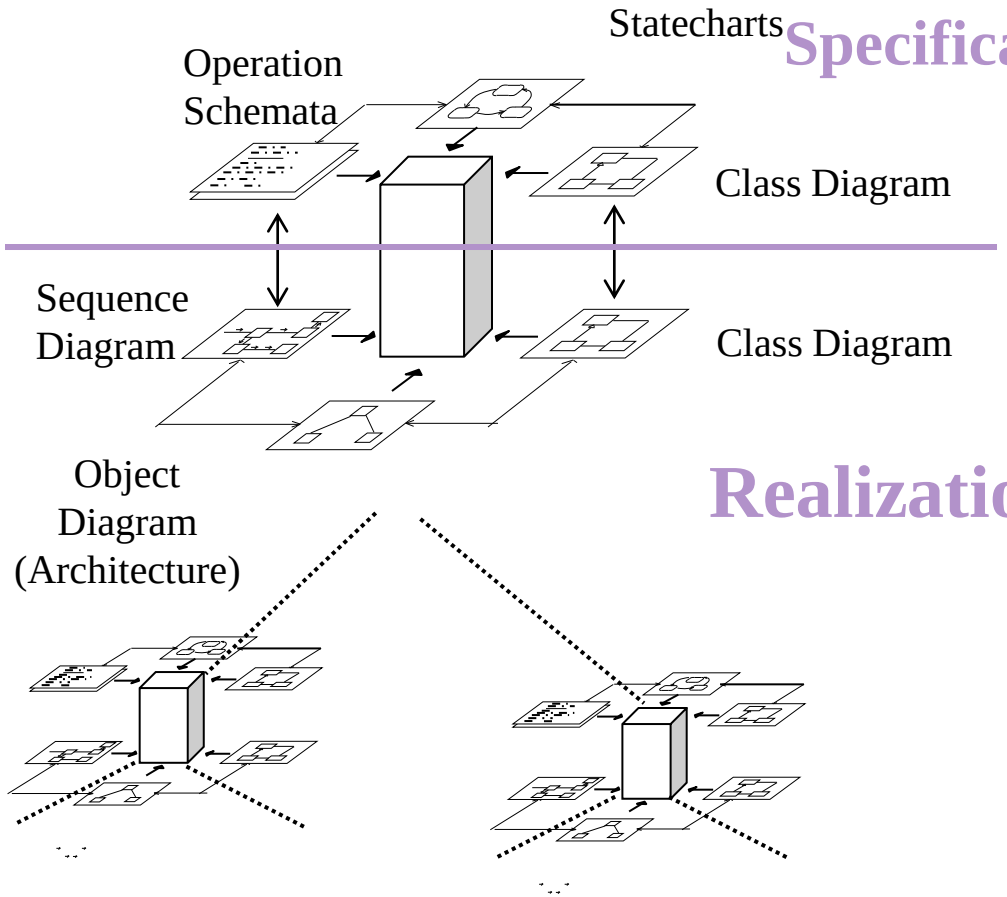
소프트웨어 안전공학 연구실
Software Safety Engineering Lab.
경북대학교

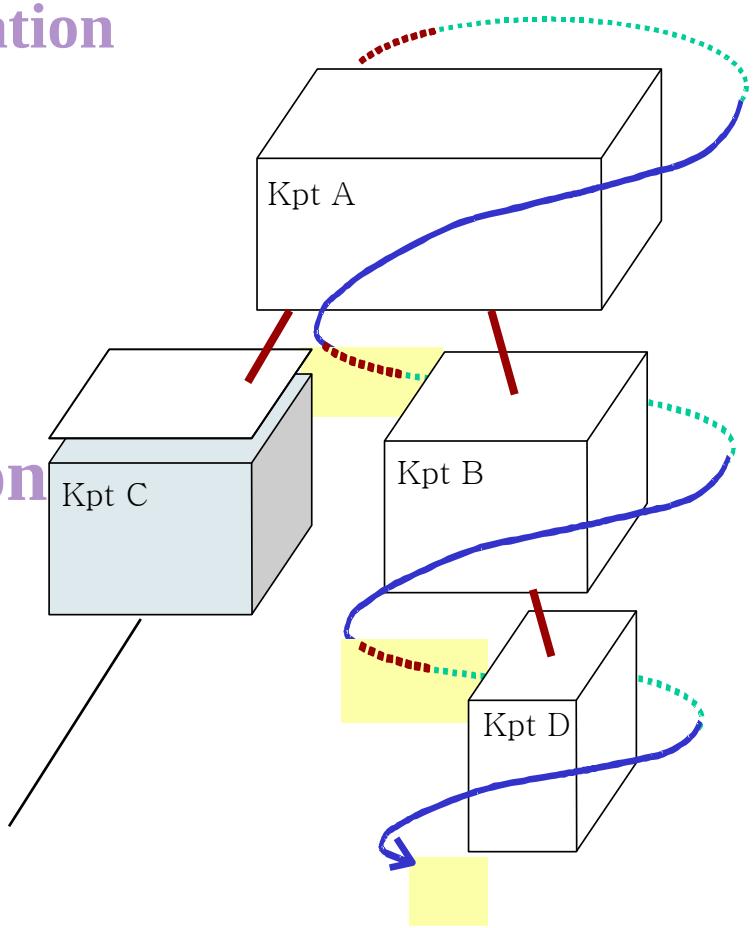
진행상황

- MARMOT Framework
- 자동차 전장용 소프트웨어를 위한 안전성 분석연구

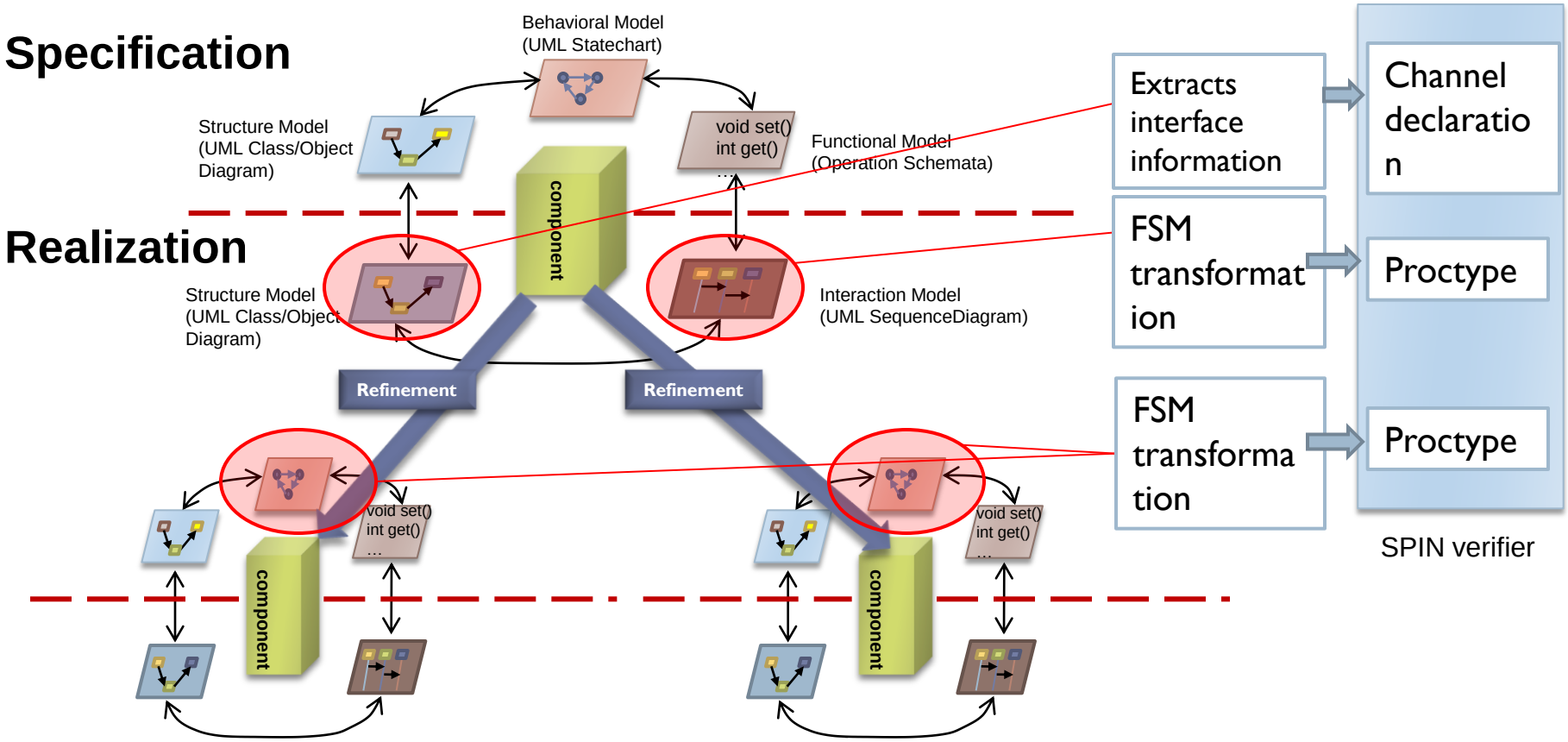
MARMOT : 진행되고 있는 일들

모델기반, 컴포넌트 중심 개발 방법론 MARMOT

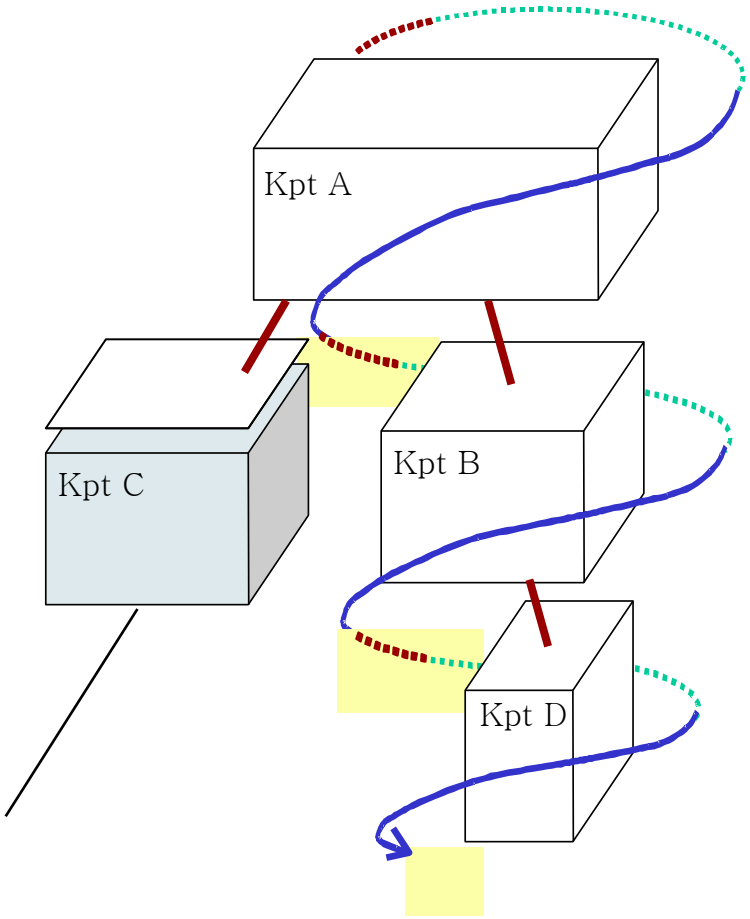




MARMOT 변환기


Specification
Realization
Behavioral Model
(UML Statechart)
Structure Model
(UML Class/Object
Diagram)
void set()
int get()
…
Functional Model
(Operation Schemata)
component
Structure Model
(UML Class/Object
Diagram)
Interaction Model
(UML SequenceDiagram)
Refinement
Refinement
void set()
int get()
…
component
void set()
int get()
…
component
Extracts
interface
information
FSM
transformat
ion
FSM
transforma
tion
Channel
declaratio
n
Proctype
Proctype
SPIN verifier

하향식 개발방식과 상향식 조합방식의 결합



Model-driven development

Bottom-up composition

Synchronized reduction

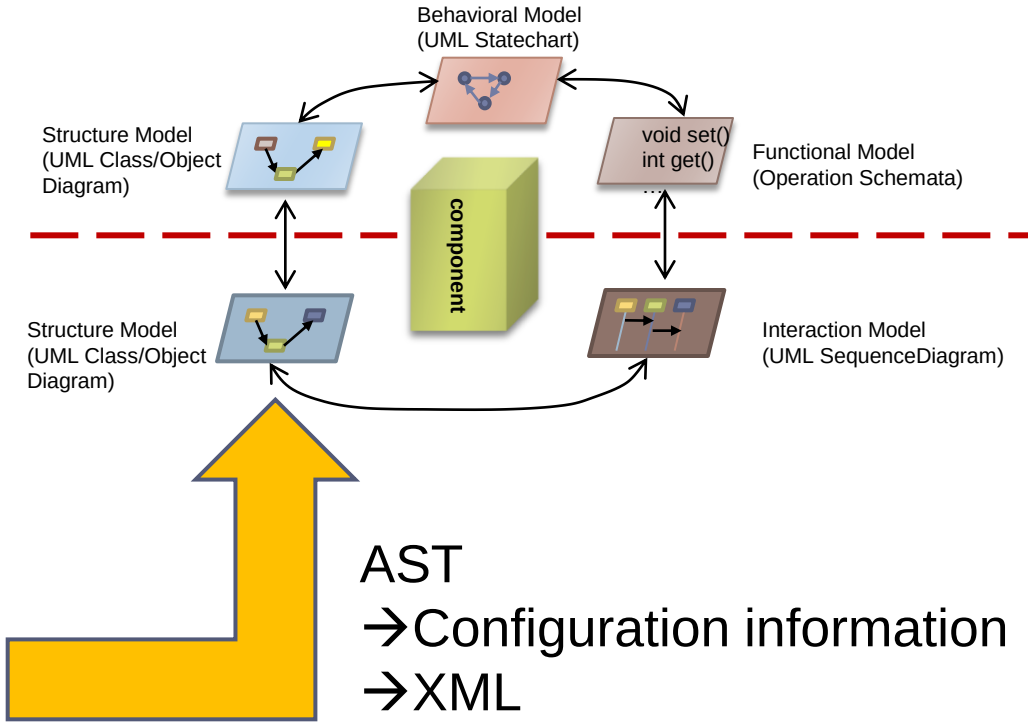
Projection

현재 진행상황

- 마무리 단계 작업들
  - MARMOT Framework (Software Systems and Modeling, to appear)
  - Abstraction techniques and experiments (COMPSAC 2010)
- 더 향상되거나 구현되어야 할 작업들
  - MARMOT 변환기 안정화
  - Implementation of synchronized reduction
- 진행 중인 작업들
  - Reverse engineering abstract components (HASE 2010, to appear)
  - Automation of Model extraction (code to abstract components)

코드 추출: Code to abstract components

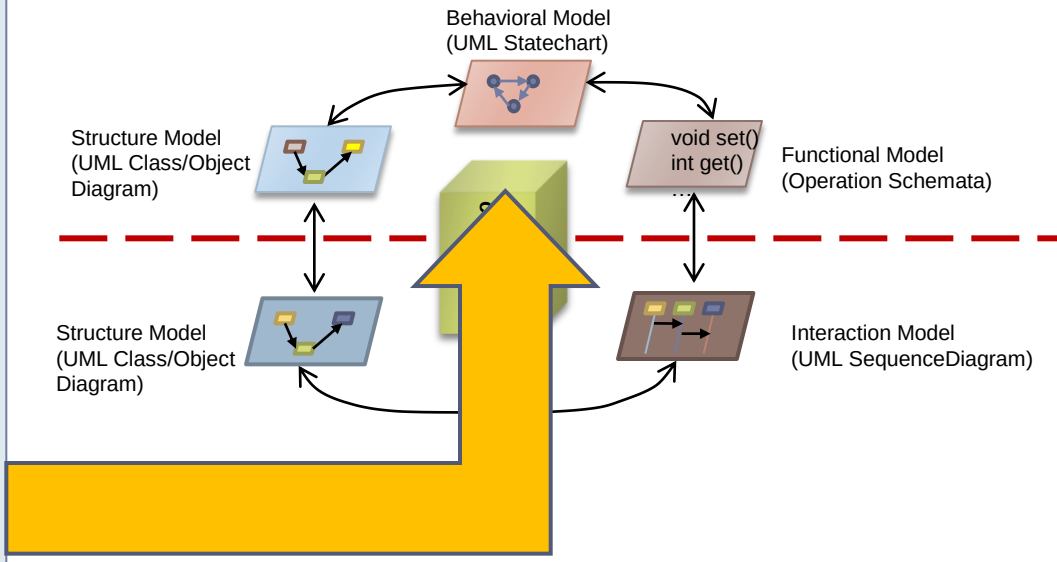
```
generic configuration AlarmMilli32C()
{
  provides interface Init;
  provides interface Alarm<TMilli,uint32_t>;
}
implementation
{
  components new Alarm32khz16C() as
AlarmFrom;
  components CounterMilli32C as Counter;
  components new
TransformAlarmC(TMilli,uint32_t,T32khz,uint16
_t,5) as Transform;

  Init = AlarmFrom;
  Alarm = Transform;

  Transform.AlarmFrom -> AlarmFrom;
  Transform.Counter -> Counter;
}
```



AST
→Configuration information
→XML
→MARMOT component structure

코드 추출: Code to abstract components

```
module LedsP {
  provides {
    interface Init;
    interface Leds;
  }
  uses {
    interface GeneralIO as Led0;
    interface GeneralIO as Led1;
    interface GeneralIO as Led2;
  }
}
implementation {
  command error_t Init.init() {
    atomic {
      call Led0.makeOutput();
      call Led1.makeOutput();
      call Led2.makeOutput();
      call Led0.set();
      call Led1.set();
      call Led2.set();
    }
    return SUCCESS;
  }
….
}
```



AST
→Implementation information
→XML
→MARMOT component behavior

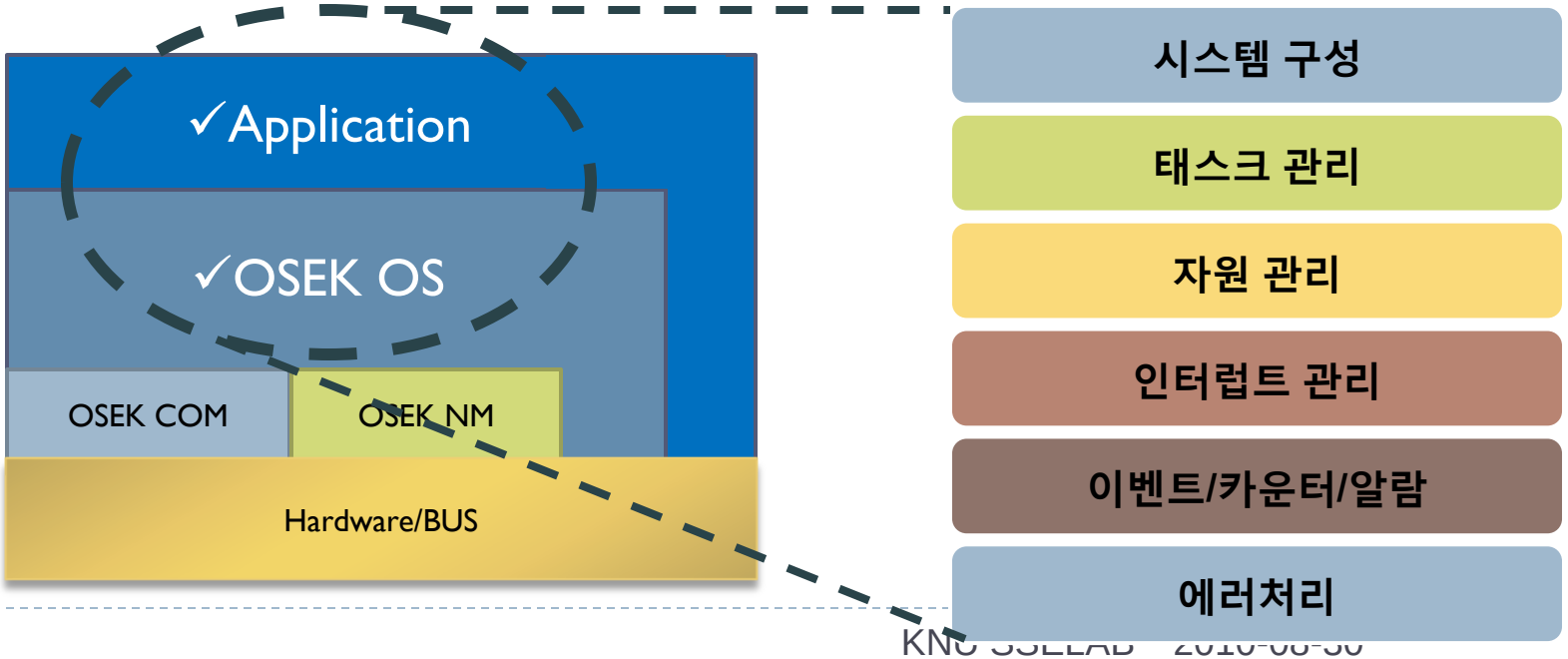
차량 전장용 시스템을 위한 소프트웨어 안전성 분석

OSEK/VDX 기반 소프트웨어 안전성 분석

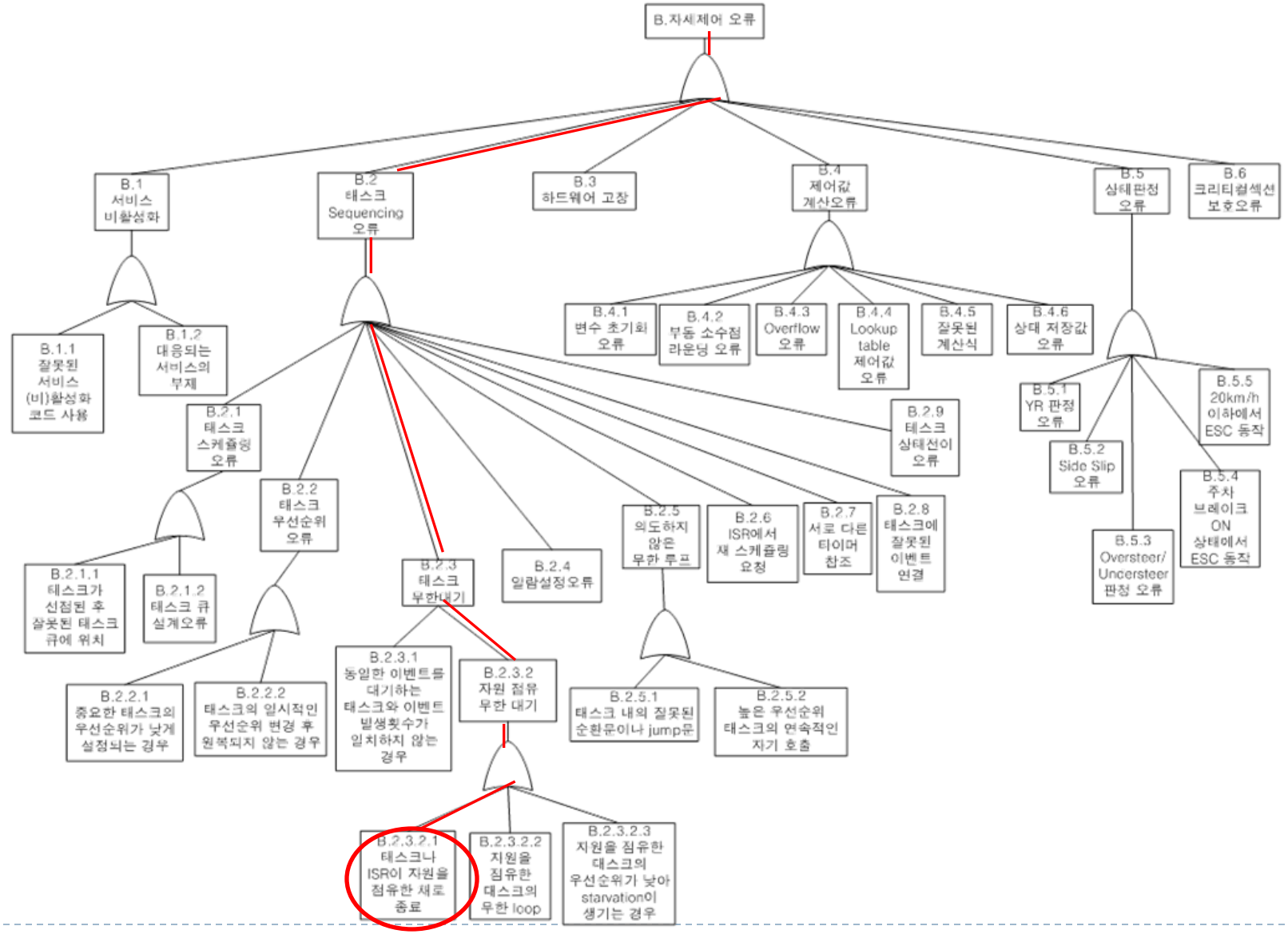
- **차량용 실시간 운영체제 관련 안전요건 분석**

실시간운영체제의 자체 안전성 요건

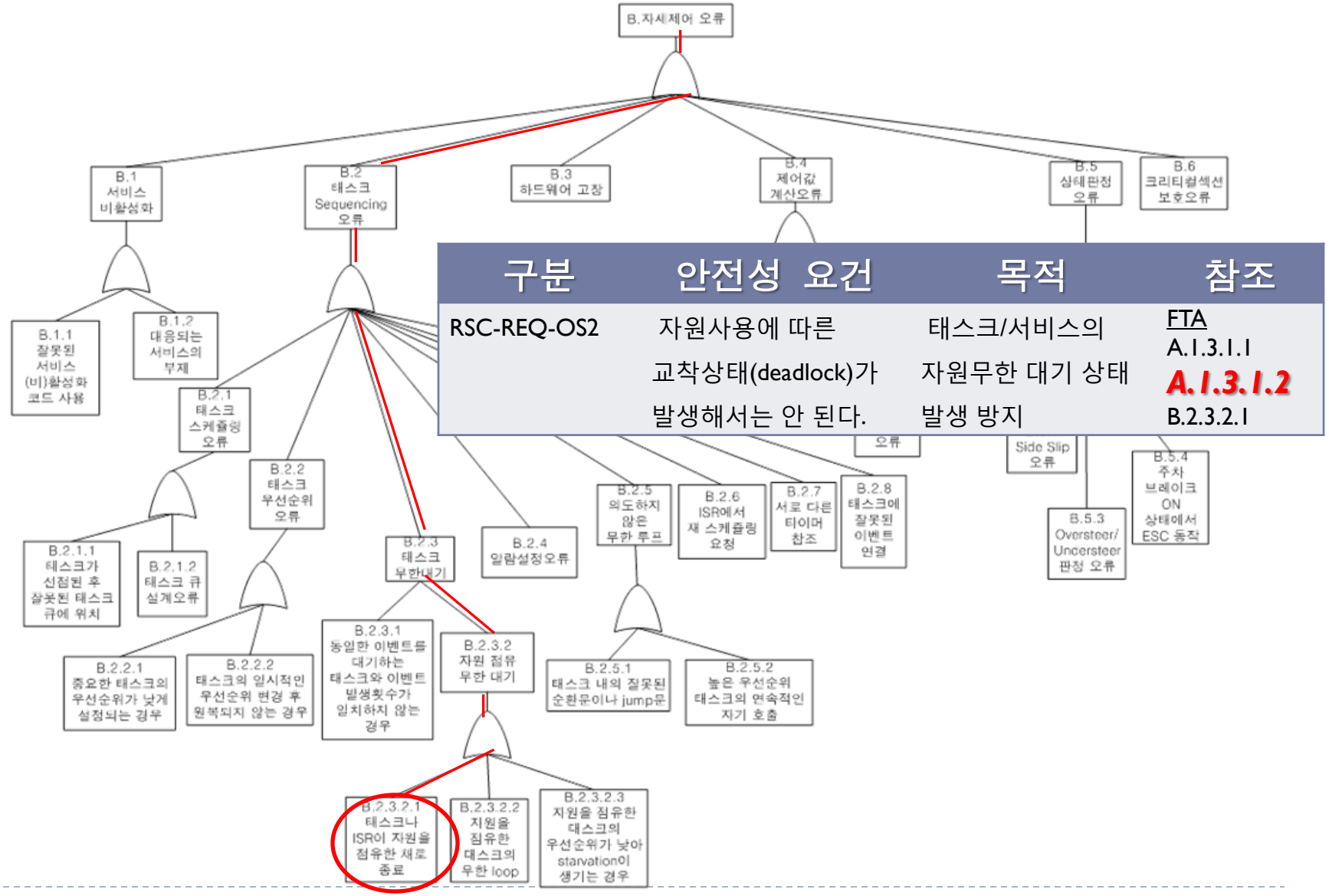
실시간운영체제와 관련된 응용소프트웨어의 안전성 요건



소프트웨어 안전성 요구사항 분석사례



소프트웨어 안전성 요구사항 분석사례



구분	안전성 요건	목적	참조
RSC-REQ-OS2	자원사용에 따른 교착상태(deadlock)가 발생해서는 안 된다.	태스크/서비스의 자원무한 대기 상태 발생 방지	FTA A.1.3.1.1 **A.1.3.1.2** B.2.3.2.1

안전 요구사항 분석결과

- ✓ **IEC 61508에 부합하는 안전성 요건 분석 절차 소개**
- ✓**시스템 수준 및 소프트웨어 수준의 안전성 요건 분석 과정 예시**
  - ✓ 가상 전장장비(차량주행제어통합시스템) 위험분석 및 안전기능 식별
  - ✓ 가상 안전관련시스템(횡방향자세제어시스템 및 소프트웨어) 개념 설계
  - ✓ 횡방향자세제어소프트웨어에 대한 FTA 수행(81개의 위험요소 식별)
- ✓**OSEK OS규격, FTA결과, 기타 표준(AUTOSAR, ISO CD 26262)를 이용하여 차량용 실시간 운영체제의 안전요건 분석**

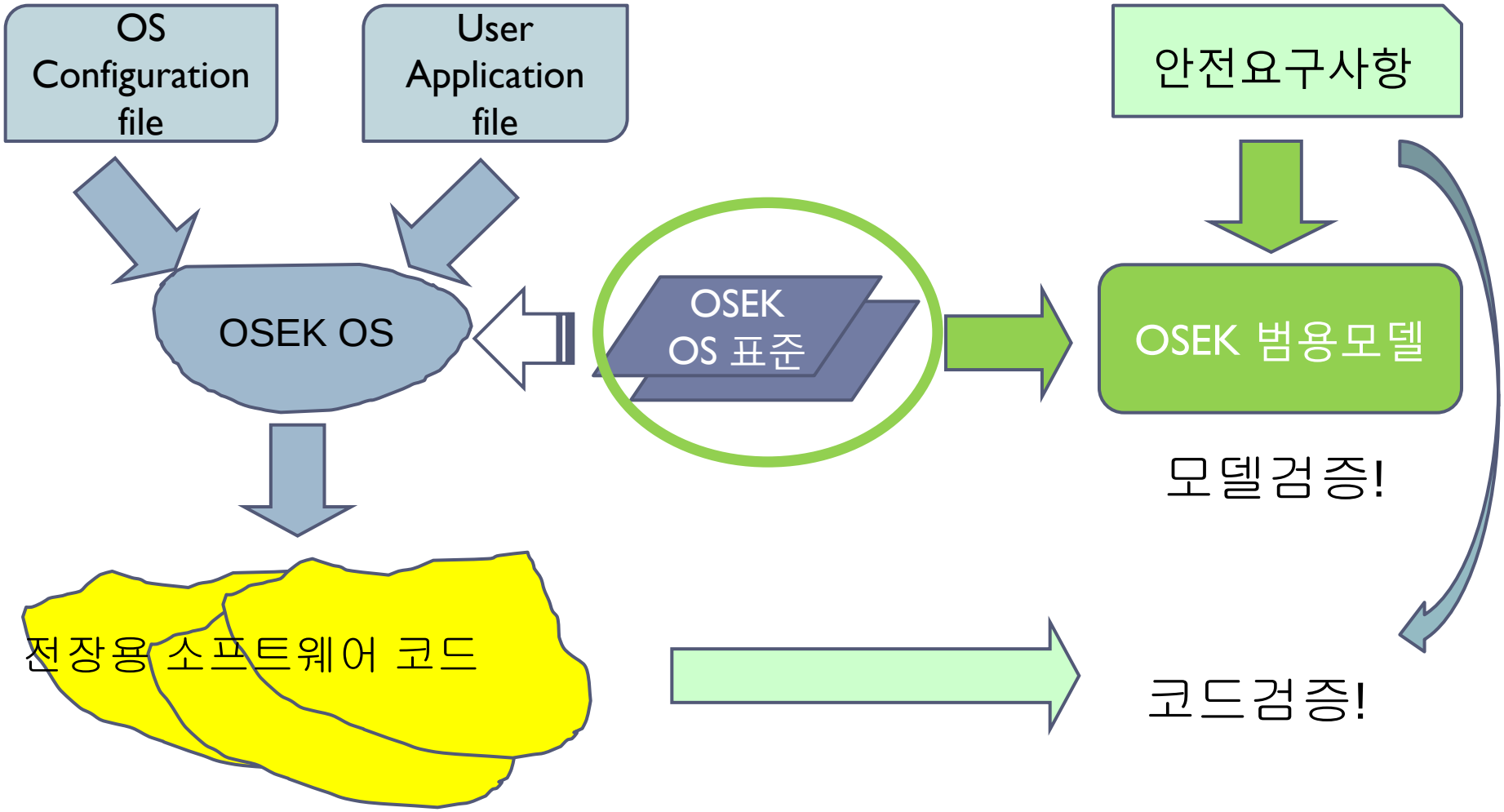
구분	시스템 구성	태스크 관리	자원 관리	인터럽트	이벤트/ 카운터/알람	에러 처리	합
운영체제 (하위요건 포함)	5 (11)	7 (11)	4 (6)	2 (5)	1 (1)	6 (21)	**25 (55)**
응용SW (하위요건 포함)	11 (3)	6 (10)	5 (10)	4 (8)	4 (4)	1 (2)	**31 (65)**

소프트웨어 안전성 검증

구분	안전성 요건	목적	참조
RSC-REQ-OS2	자원사용에 따른 교착상태(deadlock)가 발생해서는 안 된다.	태스크/서비스의 자원무한 대기 상태 발생 방지	FTA A.1.3.1.1 **A.1.3.1.2** B.2.3.2.1

- 무엇을 어떻게 검증할 것인가?
  - OSEK 표준을 검증?
  - 프로그램 코드를 검증?

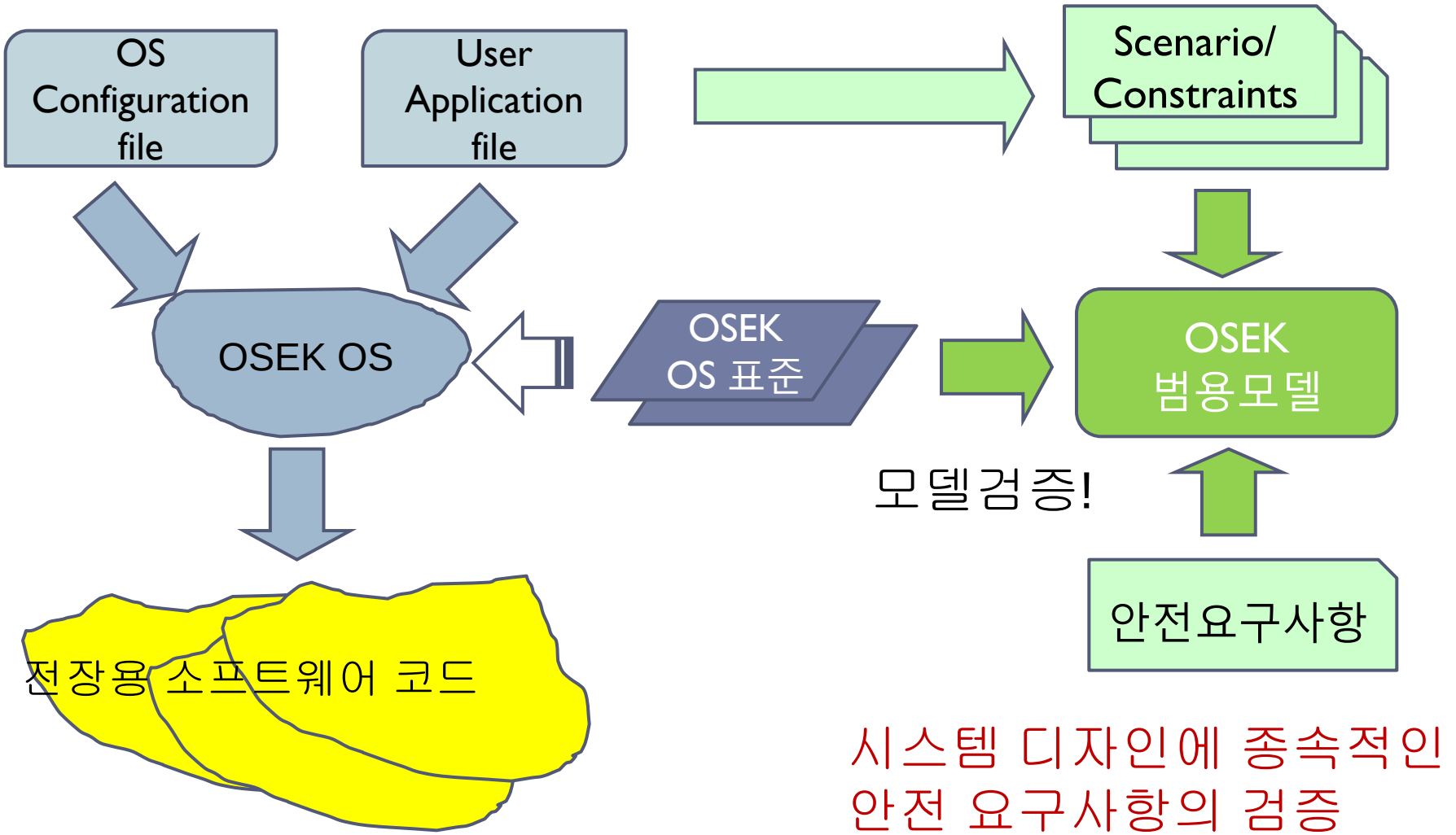
접근방식


OS Configuration file
User Application file
안전요구사항
OSEK OS
OSEK OS 표준
OSEK 범용모델
모델검증!
전장용 소프트웨어 코드
코드검증!

범용모델의 안전성 검증

- 시스템 구성요소나 응용 소프트웨어의 시나리오에 독립적인 안전 요구사항을 검증
  - 예:
    - 자원사용에 따른 교착상태가 발생하지 않아야 한다
    - 실행 중 태스크의 우선순위가 변경되는 경우, 태스크 우선순위의 무결성이 보장되어야 한다
    - 자원을 점유한 상태에서 재스케쥴링이 발생해서는 안된다
- 시스템 디자인에 종속적인 안전 요구사항의 완전 검증은 너무 많은 false alarm 의 존재로 불가능
  - 모델 시뮬레이션 수준의 검증

범용 모델을 응용소프트웨어 설계 테스터로 활용



시스템 디자인에 종속적인 안전 요구사항의 검증

문제점들

- 모델 검증의 결과를 코드 검증에 활용할 수 있는 방안 필요
  - 모델에서 코드를 생성하지 않는 경우
- 시스템 코드 무결성 $\subseteq$ OS 코드 무결성 $\times$ 디자인 무결성 $\times$ 응용 프로그램 무결성
  - 특정 시스템 코드의 무결성은 단지 하나의 시나리오에 대한 무결성을 의미할 뿐
  - 무결성 검증을 위한 프레임워크의 개발이 더 중요
- 코드 검증의 어려움
  - 추상화?? (검증 범위, 하드웨어 모델, loop unwinding 등등)
  - 도구의 비효율성
  - 거짓 경보 파악의 어려움

진행상황

- OSEK/VDX 기반 범용 OS 모델 구축
  - SPIN 을 이용한 응용 소프트웨어에 비종속적인 안전 요구 사항 검증 작업을 진행 중
- 코드 검증
  - Trampoline OSEK 코드의 정형검증
    - SPIN
    - CBMC

앞으로 더 필요한 일들

- 검증 과정의 체계화 : 각 단계별로 검증이 필요한 안전성 요구사항의 분류
  - 범용 모델 검증 단계
  - 설계 종속적인 모델 검증 단계
  - 코드 검증 단계
- 코드검증???
  - 모델 검증의 결과를 코드검증에서 재사용 할 수 있는 방안
- 검증 지원도구의 개발
  - 범용 모델을 이용한 응용 소프트웨어 검증 도구
  - 코드 추상화와 거짓 경보 식별기