KOREA UNIVERSITY
College of Informatics

Software Safety and Dependable Systems

Sungdeok (Steve) Cha
Dept of Computer Science and Engineering
Korea University
Seoul, Korea

http://dependable.korea.ac.kr/

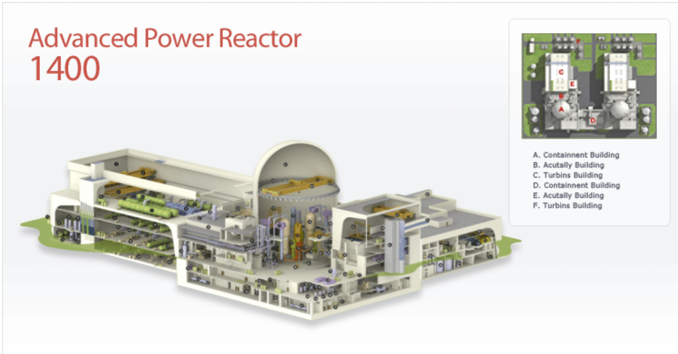
1981~1991

1994~2008

2008~

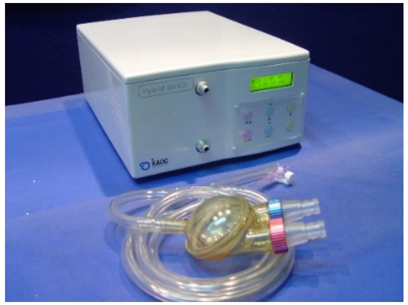
UCI, 1981

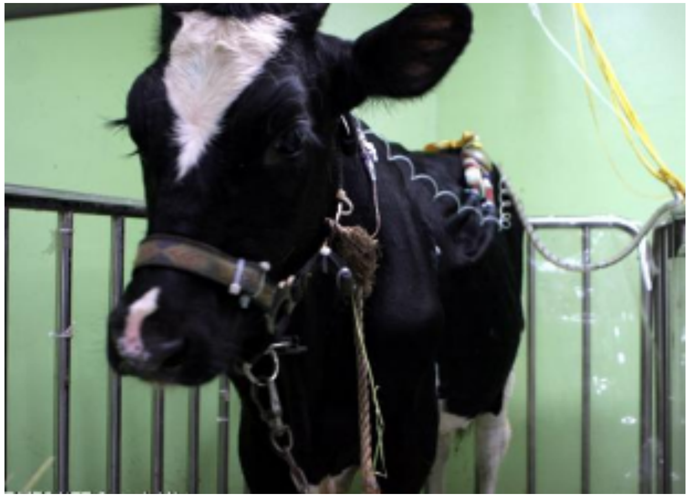
MIT, 2013



KNICS (Korea Nuclear I&C System)

Reactor Protection Software

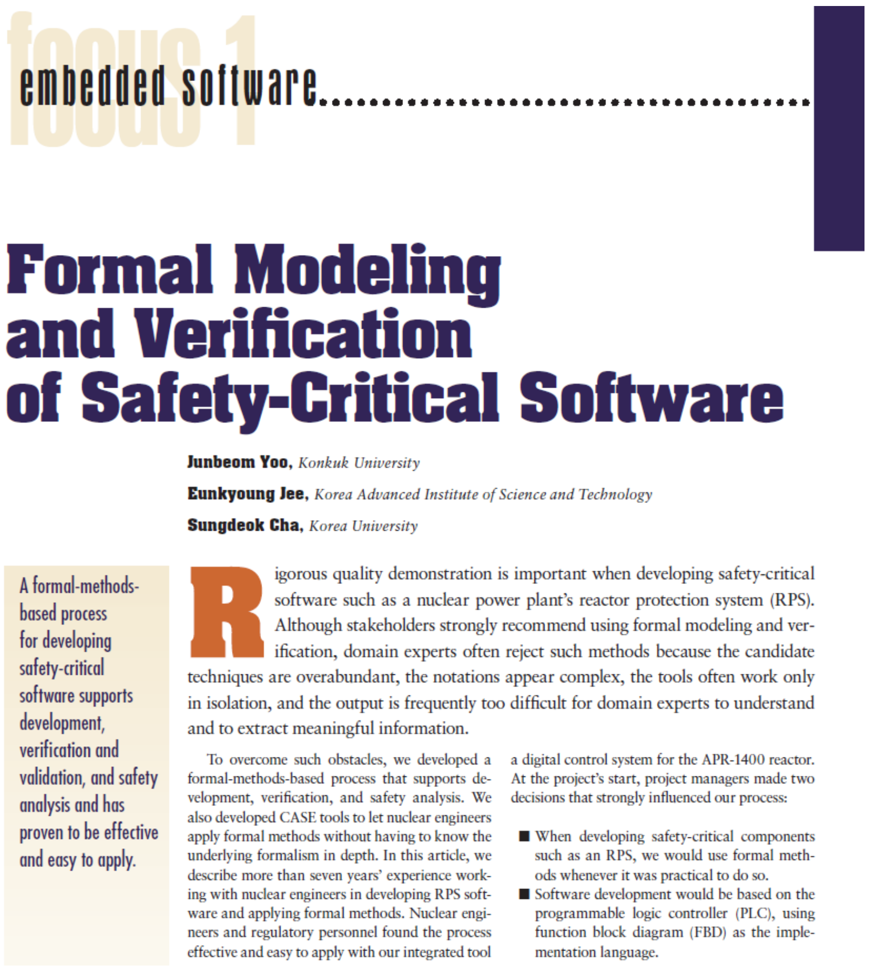




Artificial Heart (HVAD)

KNICS Project

- **Software qualification for digital safety system (2001~2007)**



focus 1

embedded software

Formal Modeling and Verification of Safety-Critical Software

Junbeom Yoo, *Konkuk University*

Eunkyoung Jee, *Korea Advanced Institute of Science and Technology*

Sungdeok Cha, *Korea University*

A formal-methods-based process for developing safety-critical software supports development, verification and validation, and safety analysis and has proven to be effective and easy to apply.

Rigorous quality demonstration is important when developing safety-critical software such as a nuclear power plant's reactor protection system (RPS). Although stakeholders strongly recommend using formal modeling and verification, domain experts often reject such methods because the candidate techniques are overabundant, the notations appear complex, the tools often work only in isolation, and the output is frequently too difficult for domain experts to understand and to extract meaningful information.

To overcome such obstacles, we developed a formal-methods-based process that supports development, verification, and safety analysis. We also developed CASE tools to let nuclear engineers apply formal methods without having to know the underlying formalism in depth. In this article, we describe more than seven years' experience working with nuclear engineers in developing RPS software and applying formal methods. Nuclear engineers and regulatory personnel found the process effective and easy to apply with our integrated tool a digital control system for the APR-1400 reactor. At the project's start, project managers made two decisions that strongly influenced our process:

- When developing safety-critical components such as an RPS, we would use formal methods whenever it was practical to do so.
- Software development would be based on the programmable logic controller (PLC), using function block diagram (FBD) as the implementation language.

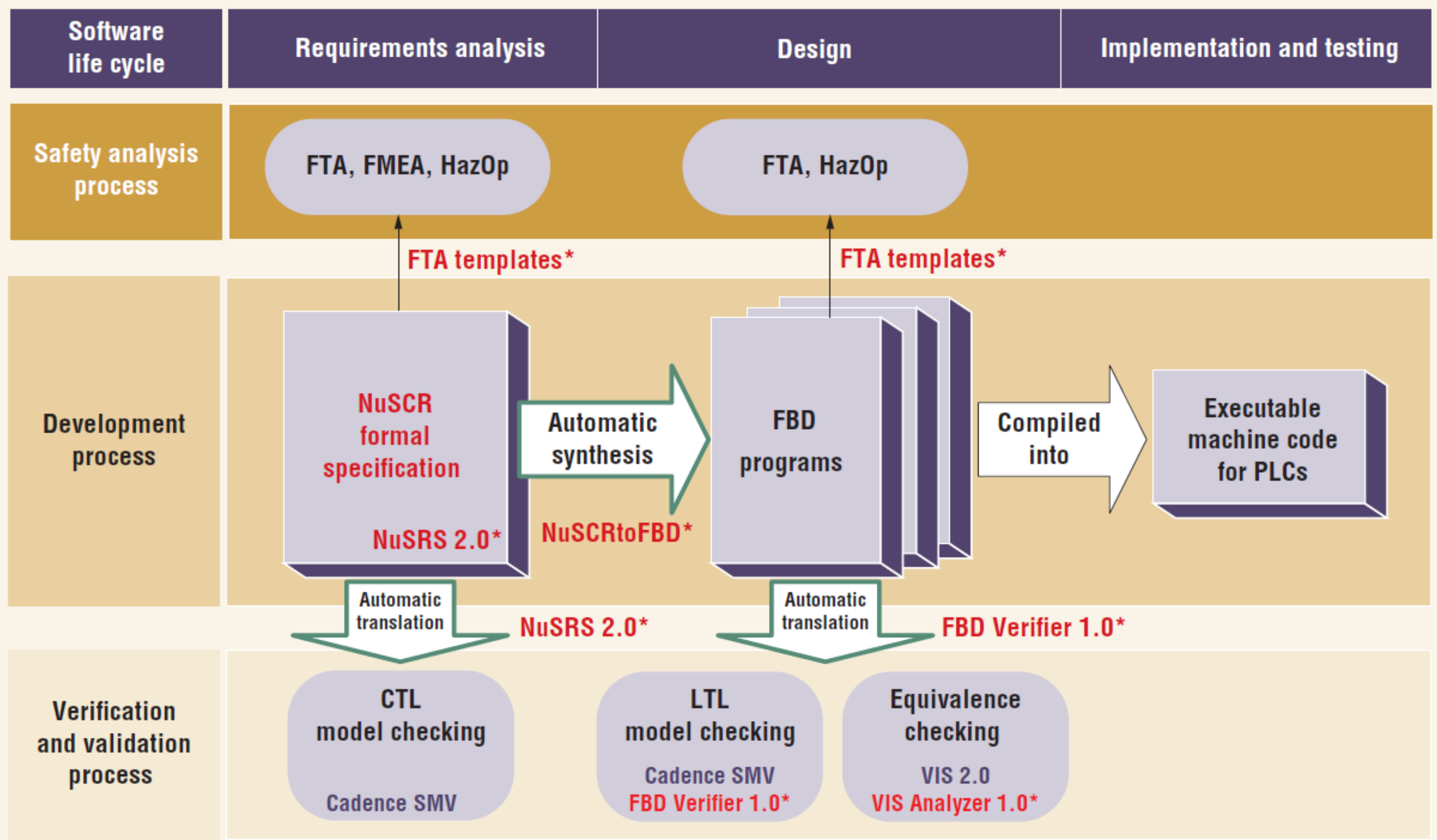
IEEE Software, May/June 2009

More on KNICS Project

- **Programmable Logic Controller(PLC)-based software development**
  - Using function block diagram (FBD) as the implementation language
  - "Project environment" to our group

- **Formal methods were used whenever practical**
  - To automate as much analysis as possible
  - To reduce human errors
  - To provide greater safety assurance
  - Our group's own decision

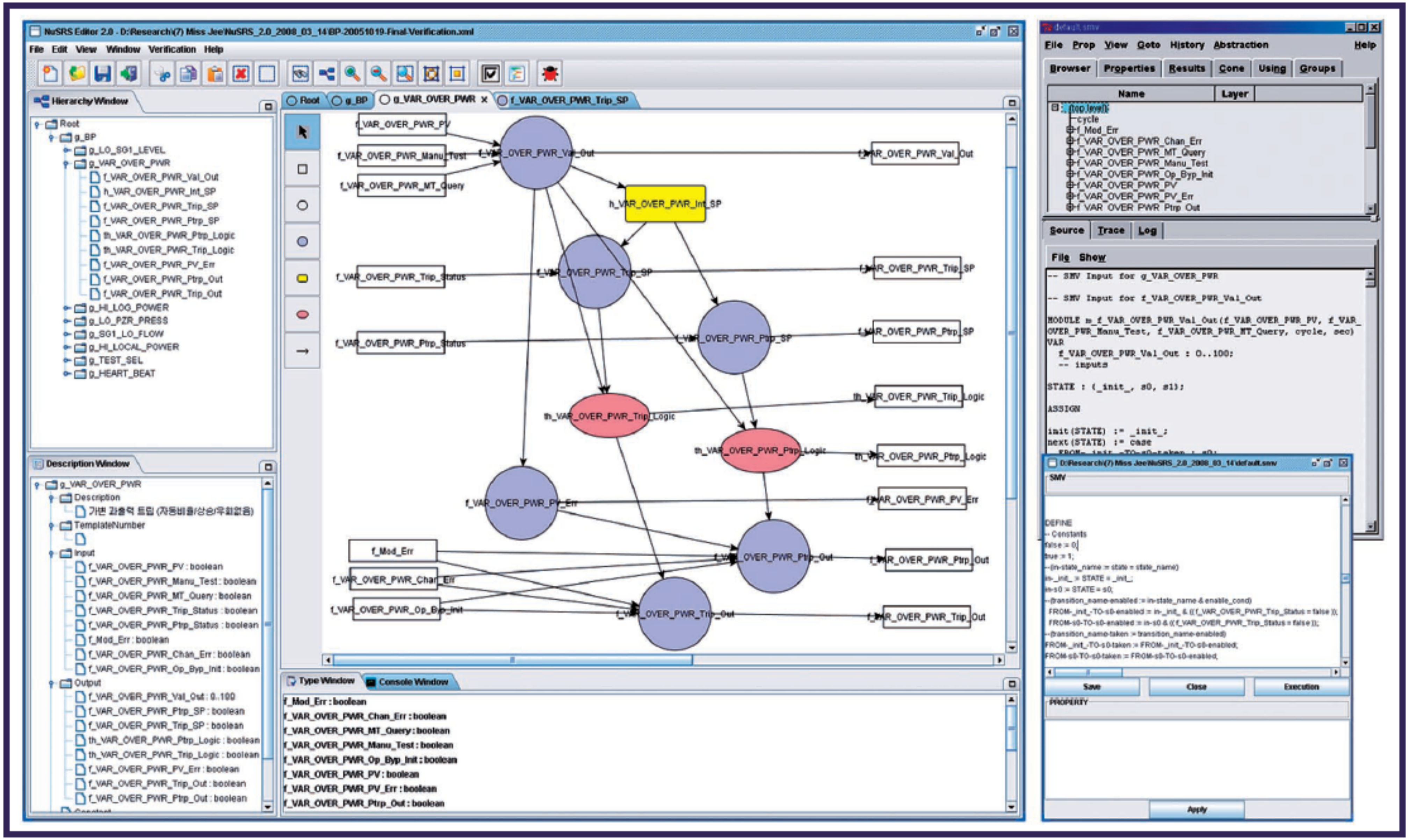
Our Approach



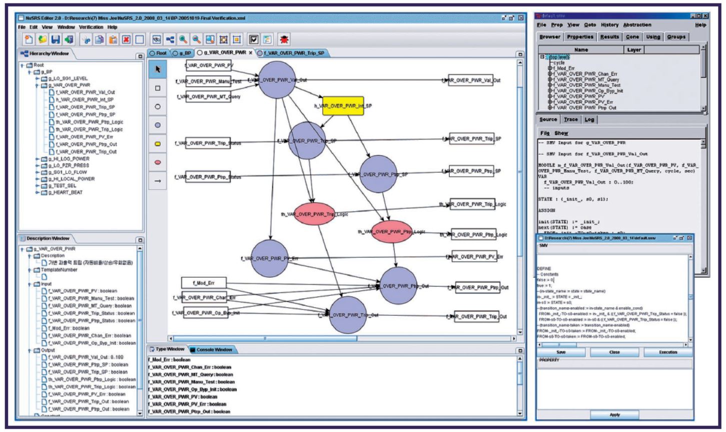
CTL	Computation tree logic	NuSCR	Software cost reduction for nuclear applications
FBD	Function block diagram	NuSRS	CASE tool for NuSCR specification and verification
FMEA	Failure modes and effects analysis	PLC	Programmable logic controller
FTA	Fault tree analysis	SMV	Symbolic model verifier
HazOp	Hazard and operability study	VIS	Verification Interacting with Synthesis
LTL	Linear temporal logic		

KOREA UNIVERSITY
College of Informatics

Requirements Engieering



Requirements Modeling



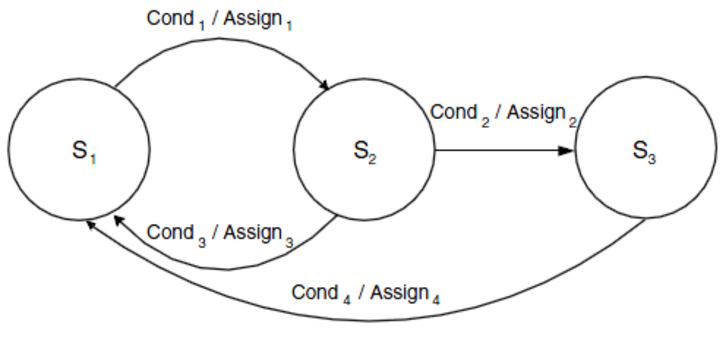


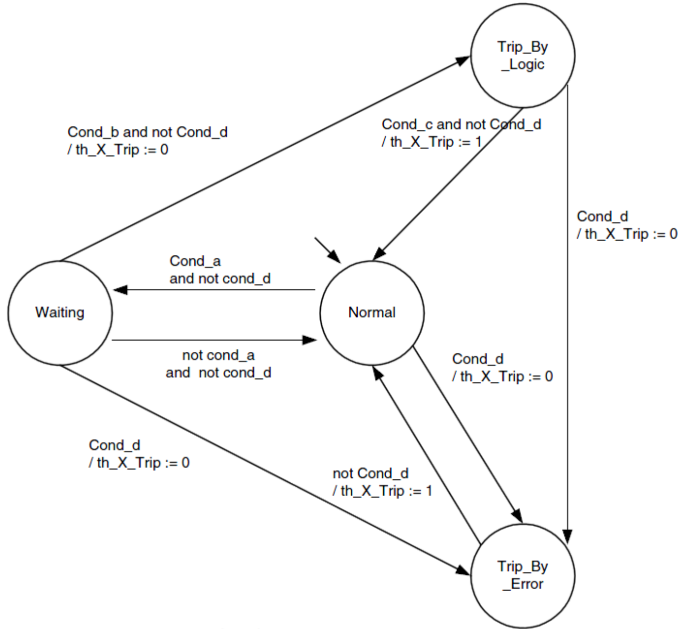
Fig. 7. Finite state machine for history variable node.

History node

Conditions			
$Cond_1$	T	-	T
$Cond_2$	F	T	-
$Cond_3$	-	F	T
Actions			
$Assign_1$	X		
$Assign_2$		X	
$Assign_3$			X

Fig. 6. Structured decision table for a function variable.

Function node



Cond_a : f_X >= k_X_Trip_Setpoint
Cond_b : [k_Trip_Delay, k_Trip_Delay] (f_X >= k_X_Trip_Setpoint and h_X_OB_Sta = 0)
Cond_c : f_X < k_X_Trip_Setpoint - k_X_Trip_Hys
Cond_d : f_X_Valid = 1 or f_Module_Error = 1 or f_Channel_Error = 1

Fig. 3. Timed transition system for *th_X_Trip*.

Timed-History node

Requirements Modeling

- **Attempted to balance between readability, expressiveness, and analyzability**
  - Preference of stakeholder, in particular the regulatory body, was taken into consideration
  - Approval experience on Wolsung NPP 2-3-4 shutdown system (1995~1997)

Conditions			
$Cond_1$	T	-	T
$Cond_2$	F	T	-
$Cond_3$	-	F	T
Actions			
$Assign_1$	X		
$Assign_2$		X	
$Assign_3$			X

Fig. 6. Structured decision table for a function variable.

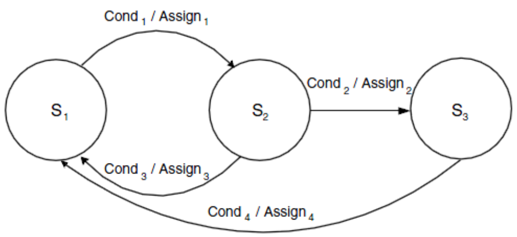


Fig. 7. Finite state machine for history variable node.

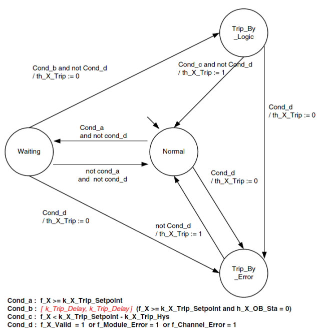
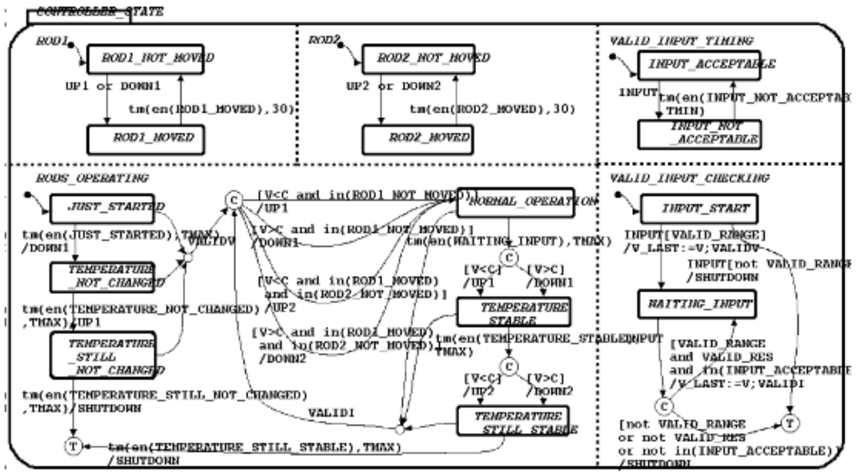


Fig. 3. Timed transition system for th_X_Trip.

VS



KOREA UNIVERSITY
College of Informatics

Requirements Modeling

- **Defined notations AND formal semantics**
- **Automated much of requirements analysis**
  - Completeness, consistency, …
  - Model checking

ELSEVIER

Available online at www.sciencedirect.com

SCIENCE DIRECT®

The Journal of Systems and Software 74 (2005) 73–83

The Journal of Systems and Software

www.elsevier.com/locate/jss

A formal software requirements specification method for digital nuclear plant protection systems

Junbeom Yoo [a,*], Taihyo Kim [a], Sungdeok Cha [a], Jang-Soo Lee [b], Han Seong Son [b]

[a] *Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology (KAIST) and AITrc/SPIC, 373-1, Kusong-dong, Yusong-gu, Taejon 305701, South Korea*

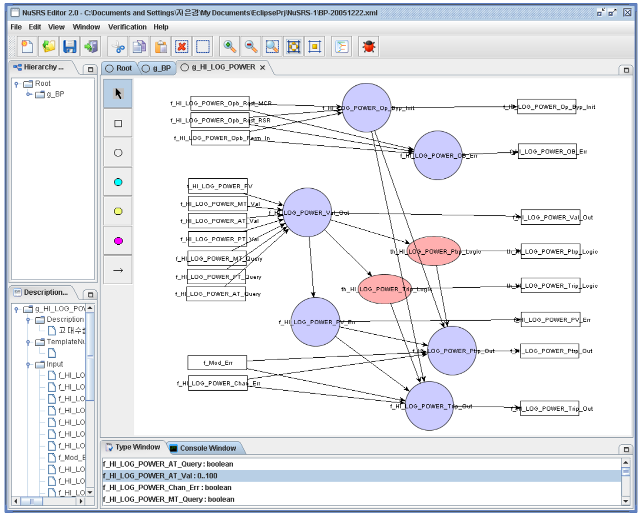
[b] *Korea Atomic Energy Research Institute (KAERI), MMIS team, 150, Deokjin-dong, Yusong-gu, Taejon, South Korea*

Received 18 March 2003; received in revised form 29 September 2003; accepted 5 October 2003
Available online 30 December 2003

Abstract

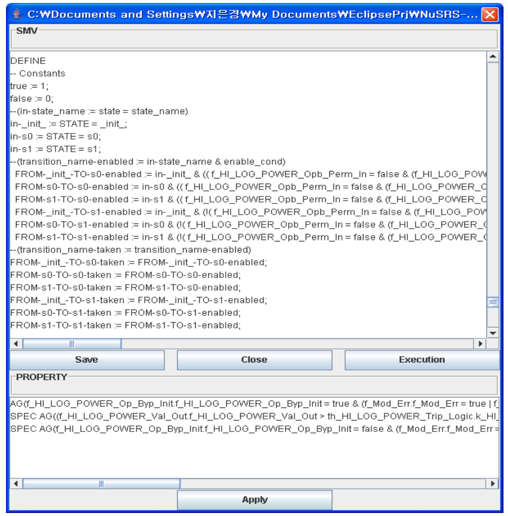
This article describes NuSCR, a formal software requirements specification method for digital plant protection system in nuclear power plants. NuSCR improves the readability and specifiability by providing graphical or tabular notations depending on the type of operations. NuSCR specifications can be formally analyzed for completeness, consistency, and against the properties specified in temporal logic. We introduce the syntax and semantics of NuSCR and demonstrate the effectiveness of the approach using reactor protection system, digital protection system being developed in Korea, as a case study.

SRS Model Checking

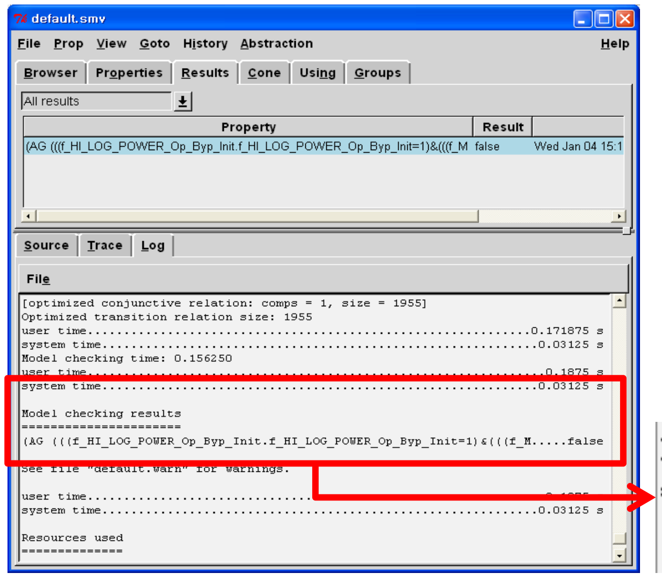


①SMV input generation from NuSCR spec.

NuSCR2SMV translator

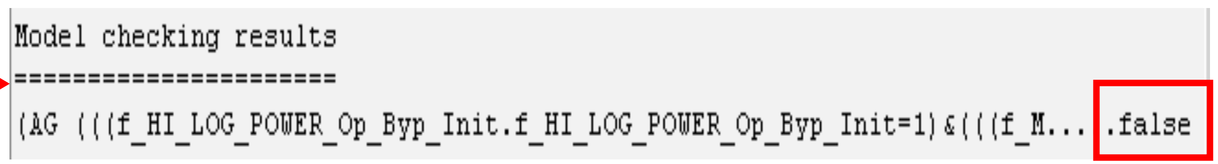


↑②Insert properties



←③Model checking using SMV

↓④Verification Result Analysis



FBD Synthesis

- **Was feasible due to relatively small (and intended) semantic gap**

Conditions			
$Cond_1$	T	-	T
$Cond_2$	F	T	-
$Cond_3$	-	F	T
Actions			
$Assign_1$	X		
$Assign_2$		X	
$Assign_3$			X

Fig. 6. Structured decision table for a function variable.

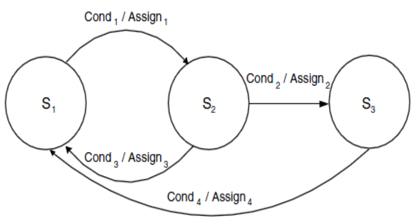
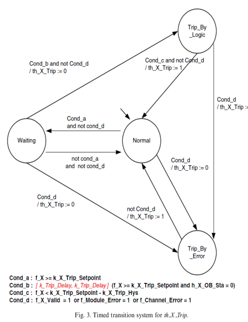
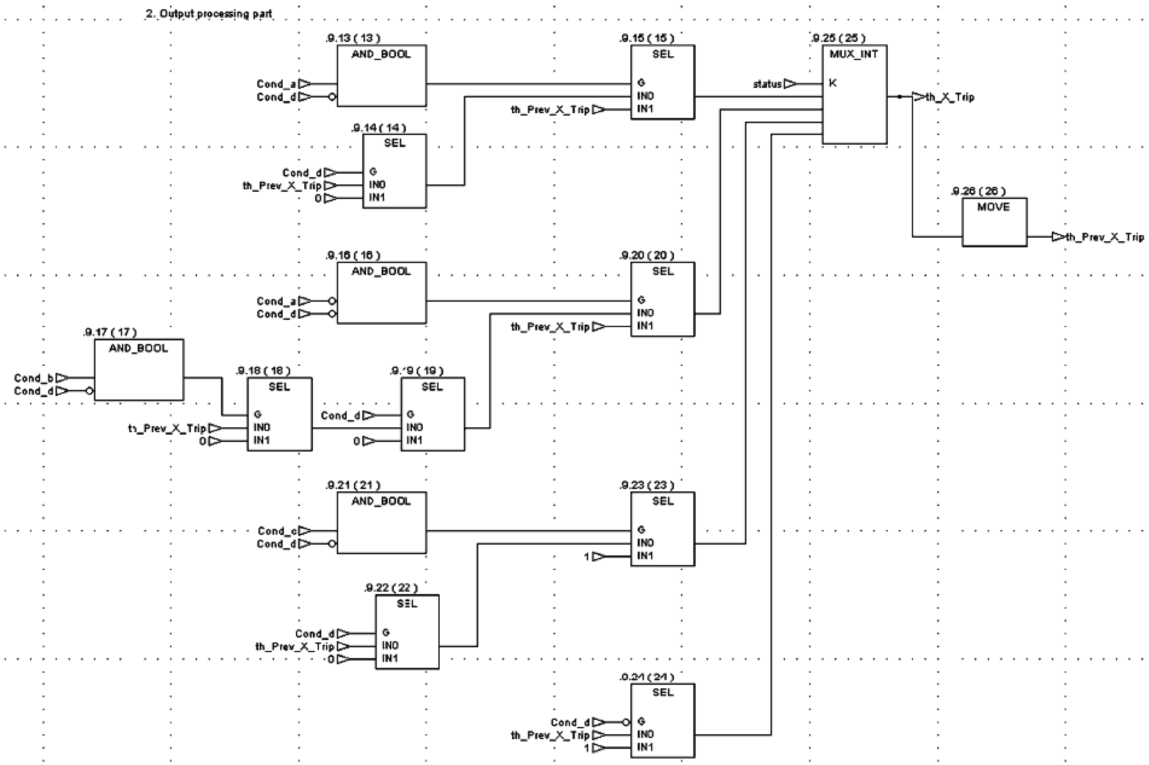


Fig. 7. Finite state machine for history variable node.





KOREA UNIVERSITY
College of Informatics

Synthesized FBD was NOT Optimized

- **state- and history-dependent nodes posed challenge**

Table 1
Comparison of the number of FBD blocks included in the fixed set-point rising trip logic

	f_X_Valid	*th_X_Trip*	*th_X_Pretrip*	*F_X_OB_Perm*	*h_X_OB_Sta*	Total
System atically generated from NuSCR	3	39	16	2	11	71
Manually generated by experts	3	12	8	9		32

Number of function blocks used.

Table 2
Comparison of the number of function blocks used for the representative trip logics in BP

Trip logic for BP	Mechanically generated from NuSCR	Manually generated by experts
Fixed set-point rising trip with operating bypass	71	32
Fixed set-point rising trip without operating bypass	53	24
Auto-limited rate variable set point trip without operating bypass	95	40
Manual reset variable set point trip with operating bypass	117	67
Total	336 2.06:1	163

Number of function blocks used.

ELSEVIER
Available online at www.sciencedirect.com
SCIENCE DIRECT
Reliability Engineering and System Safety 87 (2005) 287–294
RELIABILITY ENGINEERING & SYSTEM SAFETY
www.elsevier.com/locate/ress

Synthesis of FBD-based PLC design from NuSCR formal specification

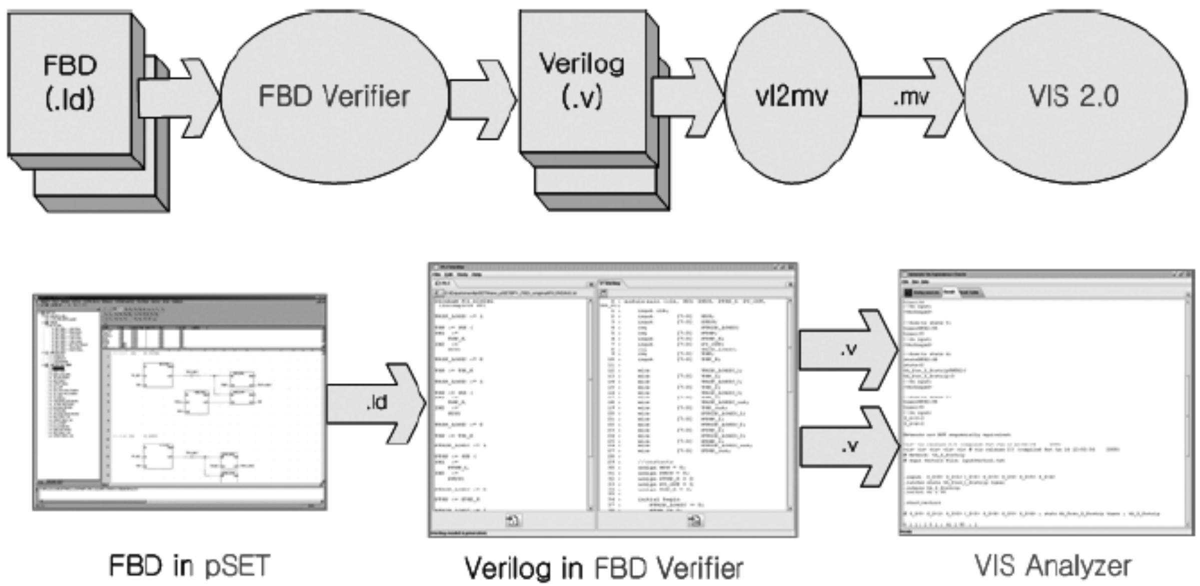
Junbeom Yoo[a,*], Sungdeok Cha[a], Chang Hwoi Kim[b], Duck Yong Song[c]

[a]*Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology (KAIST) and AITrc/SPIC/IIRTRC, 373-1, Kusong-dong, Yusong-gu, Taejon, South Korea*
[b]*I&C-HMI team, Korea Atomic Energy Research Institute (KAERI), 150, Deokjin-dong, Yusong-gu, Taejon, South Korea*
[c]*Nuclear Research Division, Atomic Creative Technology Ltd, 1688-5, Sinil-dong, Daedeok-gu, Taejon, South Korea*

Received 15 August 2003; accepted 21 May 2004

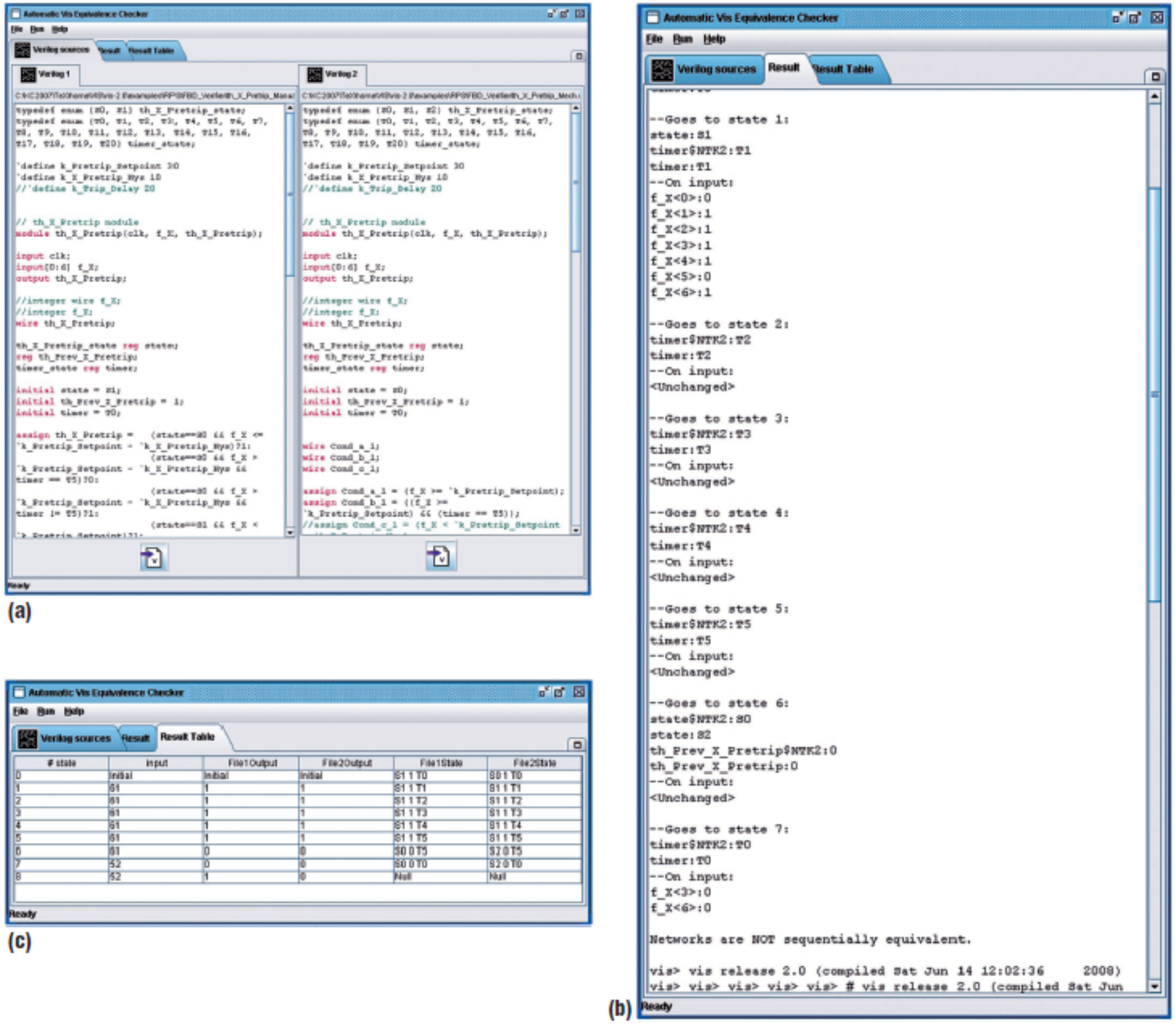
FBD Verification and VIS

- **Manual optimization of FBD code was inevitable**
- **Used VIS (Verification Interacting with Synthesis) to support subsequent behavioral equivalence**
  - Defined FBD translation rules into Verilog



Intuitive and Visual Analysis

- **To help domain experts better understand results**

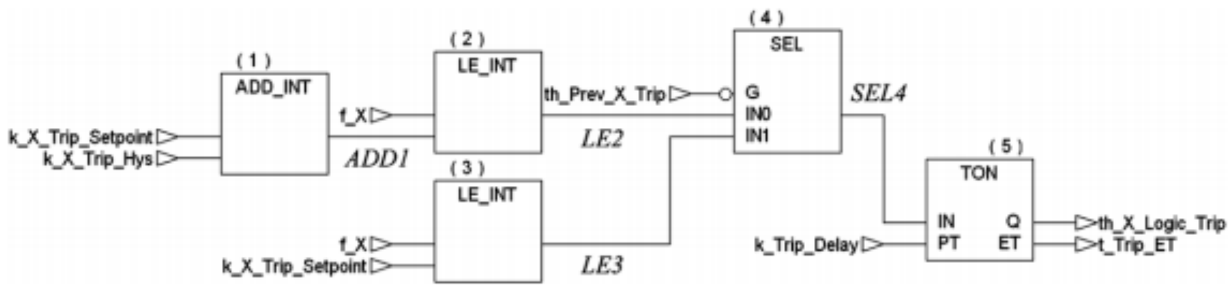


(a) (b) (c)

FBD Testing

- **FBD, based on its data-flow model, posed particular challenge**
  - Any test case would satisfy 100% coverage with simplistic definition
- **Had to define a customized coverage criteria to satisfy regulatory requirements**

…The two aspects of test coverage that are particularly important for the unit testing of safety system software are *coverage of requirements* and *coverage of the internal structure of the code.* … For safety system software, the unit test coverage criteria to be employed should be identified and justified. … [USNRC Regulation Guide 1.171] [4]



KOREA UNIVERSITY
College of Informatics

Obama co-author? ;)

Information and Software Technology 51 (2009) 1131–1139

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

ELSEVIER

A data flow-based structural testing technique for FBD programs

Eunkyoung Jee [a], Junbeom Yoo [b], Sungdeok Cha [c,*], Doohwan Bae [a]

[a] Div. of Computer Science, EECS Department, KAIST, 335 Gwahangno, Yuseong-gu, Daejeon 305-701, Republic of Korea
[b] Div. of Computer Science and Engineering, Konkuk University, 1 Hwayang-dong, Gwangjin-gu, Seoul 143-701, Republic of Korea
[c] Dept. of Computer Science and Engineering, Korea University, Anam-dong, Seongbuk-Gu, Seoul 136-701, Republic of Korea

A R T I C L E I N F O

Article history:
Received 4 August 2008
Received in revised form 22 January 2009
Accepted 24 January 2009
Available online 10 March 2009

Keywords:
Software testing
Structural testing
Test coverage criteria
Programmable logic controller
Function block diagram

A B S T R A C T

With increased use of programmable lo
assurance became an important issue. R
ical systems by identifying coverage crit
erage criteria, based on control flow g
function block diagram (FBD) which is a
three structural coverage criteria for FB
their effectiveness using a real-world re
prepared by FBD testing professionals, o
tested sufficiently. Domain experts, ha
effective.

FBD-customized coverage criteria

- **Defined conditions under which specific input played direct role in determining the output**
  - d-path condition
- **Defined various coverage criteria under which various FBD unit testing could be performed**
  - Basic coverage, input condition coverage, complex condition coverage
  - Similar to statement coverage, branch coverage, condition coverage in traditional (procedural) software testing

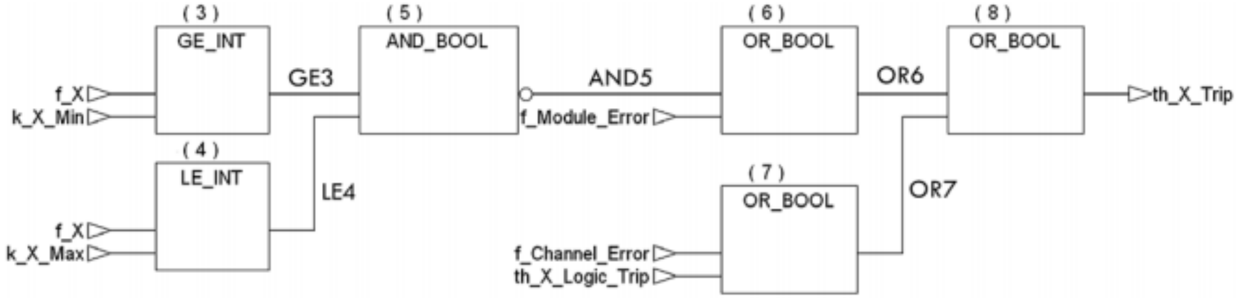


Fig. 5. A simplified FBD program for calculating *th_X_Trip*.

(a) Coverage Result for **Basic Coverage Criterion**

Test Set	Test Requirements							#Test Reqs (Satisfied/ Total)	Coverage (%)
	DPC(p_{31})	DPC(p_{32})	DPC(p_{33})	DPC(p_{51})	DPC(p_{52})	DPC(p_{53})	DPC(p_{54})		
TS1	O	O	O	O	O	O	O	7/7	100%

(b) Coverage Result for **Input Condition Coverage Criterion**

Test Set	Test Requirements										#Test Reqs (Satisfied/ Total)	Coverage (%)
	DPC(p_{31}) ∧ ME	DPC(p_{31}) ∧ ¬ME	DPC(p_{32}) ∧ CE	DPC(p_{32}) ∧ ¬CE	DPC(p_{33}) ∧ LT	DPC(p_{33}) ∧ ¬LT	DPC (p_{51})	DPC (p_{52})	DPC (p_{53})	DPC (p_{54})		
TS1	X	O	X	O	X	O	O	O	O	O	7/10	70%
TS2	O	O	O	O	O	O	O	O	O	O	10/10	100%

(c) Coverage Result for **Complex Condition Coverage Criterion**

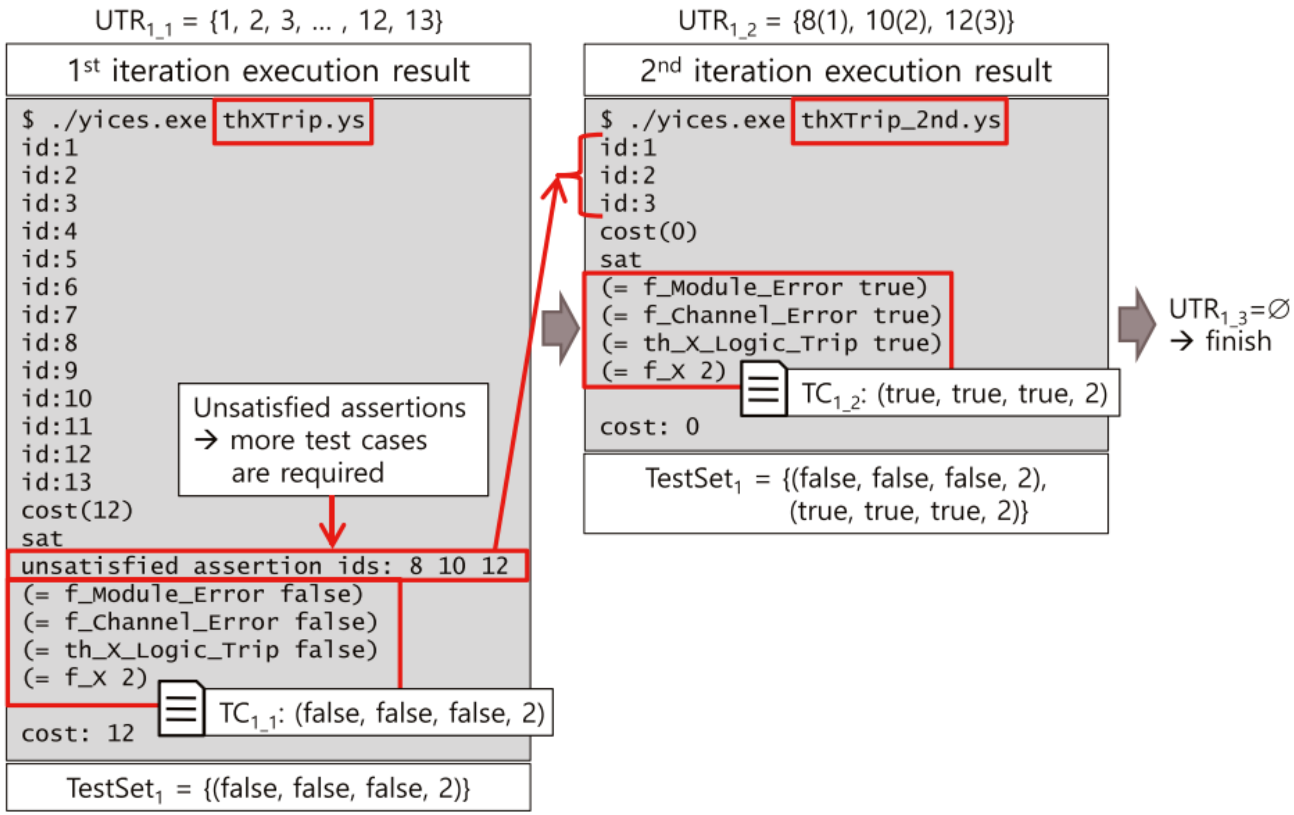
Test Set	Test Requirements																#Test Reqs (Satisfied/ Total)	Coverage (%)
	DPC(p_{31}) ∧ ME	DPC(p_{31}) ∧ ¬ME	DPC(p_{31}) ∧ OR6	DPC(p_{31}) ∧ ¬OR6	DPC(p_{31}) ∧ TR	DPC(p_{31}) ∧ ¬TR	• • •	DPC(p_{54}) ∧ LE4	DPC(p_{54}) ∧ ¬LE4	DPC(p_{54}) ∧ AND5	DPC(p_{54}) ∧ ¬AND5	DPC(p_{54}) ∧ OR6	DPC(p_{54}) ∧ ¬OR6	DPC(p_{54}) ∧ TR	DPC(p_{54}) ∧ ¬TR		
TS1	X	O	X	O	X	O		O	X	X	O	X	O	X	O	25/50	50%
TS2	O	O	O	O	O	O		O	X	X	O	X	O	X	O	34/50	68%
TS3	O	O	O	O	O	O		O	O	O	O	O	O	O	O	50/50	100%

For the input vector (f_X, f_Module_Error, f_Channel_Error, f_X_Logic_Trip),
TS1 = { (2, 0, 0, 0) }
TS2 = { (2, 0, 0, 0), (2, 1, 1, 1) }
TS3 = { (2, 0, 0, 0), (2, 1, 1, 1), (0, 0, 1, 1), (99, 0, 1, 1) }

ME: f_Module_Error
CE: f_Channel_Error
LT: th_X_Logic_Trip
TR: th_X_Trip

Fig. 6. Coverage assessment result for the FBD program in Fig. 5.

- **Developed a tool to automate test case generation (Jee et at., STVR, 2014)**
  - Used Yices, an SMT solver developed by SRI International



KOREA UNIVERSITY
College of Informatics

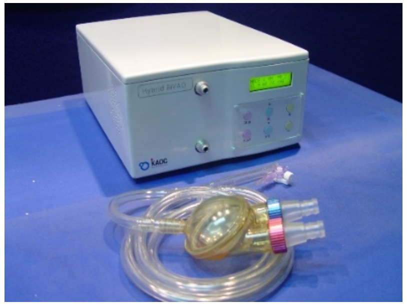
KNICS Lessons

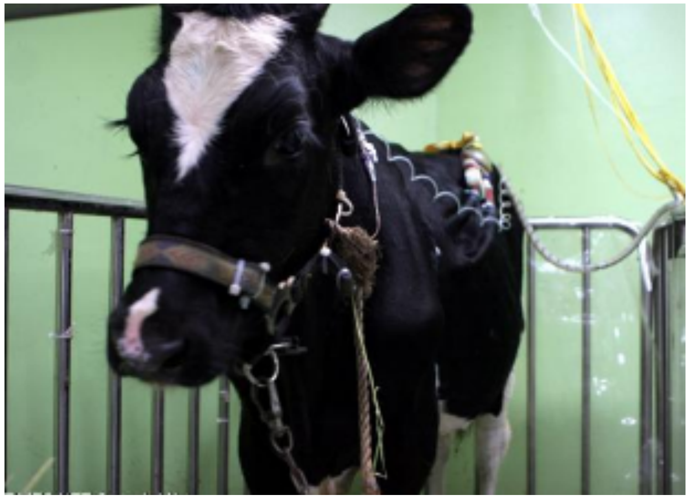
- **Be flexible and attentive to stakeholders' needs**
  - Including regulatory personnel when relevant
- **"Let them do the work" ;)**
  - Extremely important that SE professionals communicate and work well with domain experts
  - It is NEVER as easy as it seems
- **Do not reinvent wheels. SMV, VIS, Yices, …**
- **Provide data visualization/interpretation tools**
- **Domain-specific problems can become interesting SE challenges (e.g., FBD testing criteria)**

KOREA UNIVERSITY
College of Informatics

Hybrid Ventricular Assist Device

- **Korea Artificial Organ Center (KAOC) project**
- **Animal-tested for 183 days on a calf, exceeding the FDA regulations on long-term experiment**
  - No anomaly was detected





뉴스홈 > 의료

등록날짜 2007년08월13일 00시00분 글자크기

국산 인공심장 이식 송아지, 국내 최장 70일 생존

高大 한국인공장기센터 연구팀, 인공장기상용화 전기마련

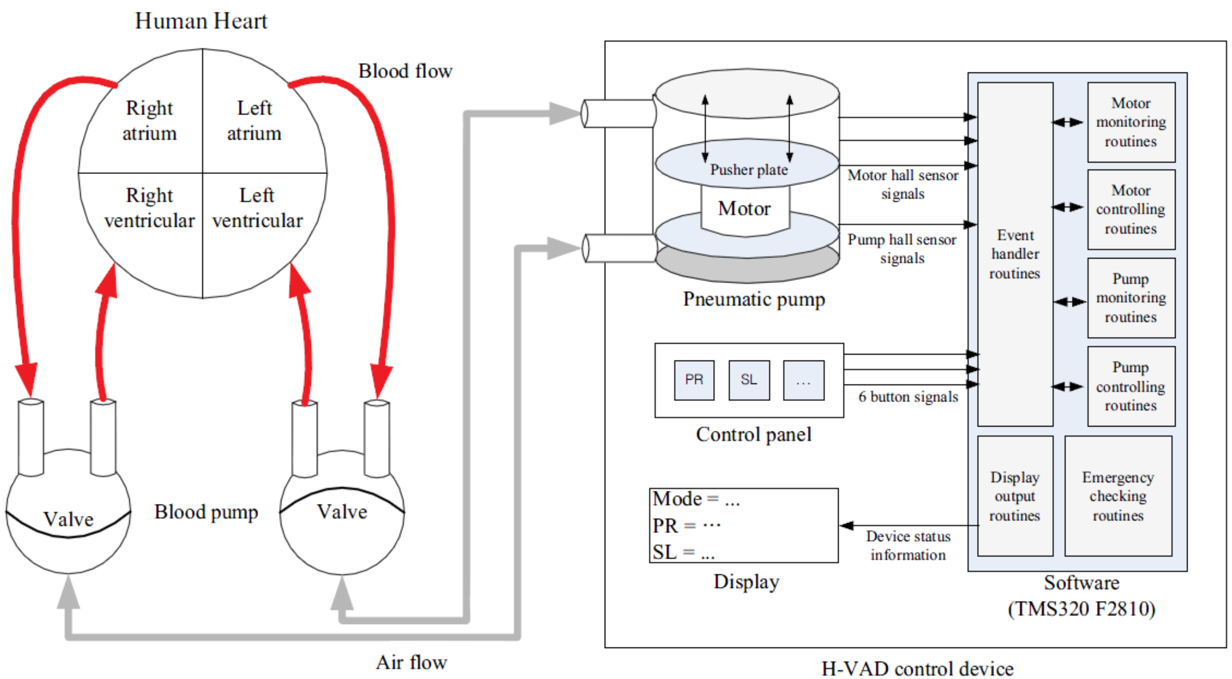
고려대 한국인공장기센터(소장 선경, 안암병원 흉부외과)가 인공심장을 이식한 송아지가 국내 최장 생존기록 70일을 세워, 의학계의 관심이 쏠리고 있다.

종전 45일보다 25일 이상 오래 생존한 기록이다.

이번에 이식한 인공심장 H-VAD는 고려대 한국인공장기센터가 자체개발한 것이다.이로써 국내기술로 만든 인공심장으로는 최장 생존기록을 달성했다는 점이 의미가 깊다.

H-VAD Software

- **Event-driven architecture**
  - Pumping Rate (PR), Stroke Length (SL), Start / Stop button
  - 211 probing statements were added
- **Control logic in ~3,800 LoC in C**
  - Relatively simple branch conditions




```
switch(CHKBIT(Control_Flag, BIT01)) {
    case B1SET :    // Brake-time Process
        /*****************/
        visited(1, 1, 11);
        /*****************/
        if(CHKBIT(Control_Flag, BIT02) == B2SET) {  // if left_end_position,
            /*****************/
            visited(1, 1, 12);
            /*****************/
            if( Ref_Left_Brake_Time < Real_Left_Brake_Time ) {
                /*****************/
                visited(1, 1, 13);
                /*****************/
                if( Op_Status != EMERGENCY_OP_2 ) { // if abnormal stop not occurred,
                    /*****************/
                    visited(1, 1, 14);
                    /*****************/
                    // PR_Calculation
                    // 0.001 * PR_Counter (sec) : 1 = 60 (sec) : Pump_Rate
```

Time Interrupt Function

KOREA UNIVERSITY
College of Informatics

In-Vitro Testing

- **Each button was pressed at least once**
- **All permitted parameter values were covered (e.g., SL 30~90, default 60)**
- **Stop button was pressed at arbitrary and random moments**

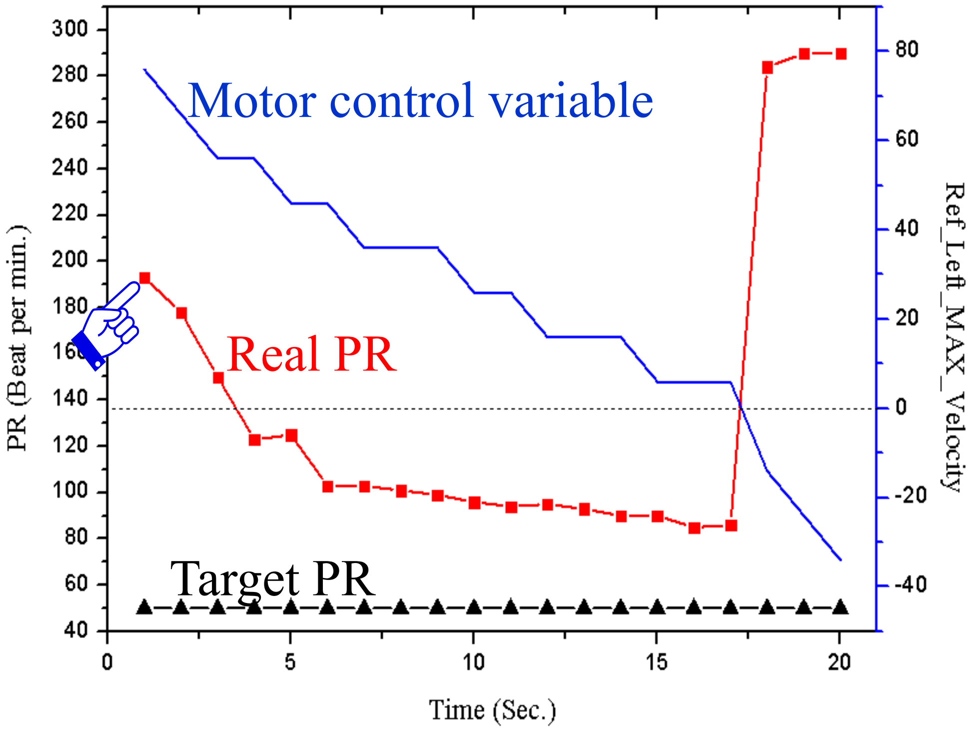
In-Vitro Testing

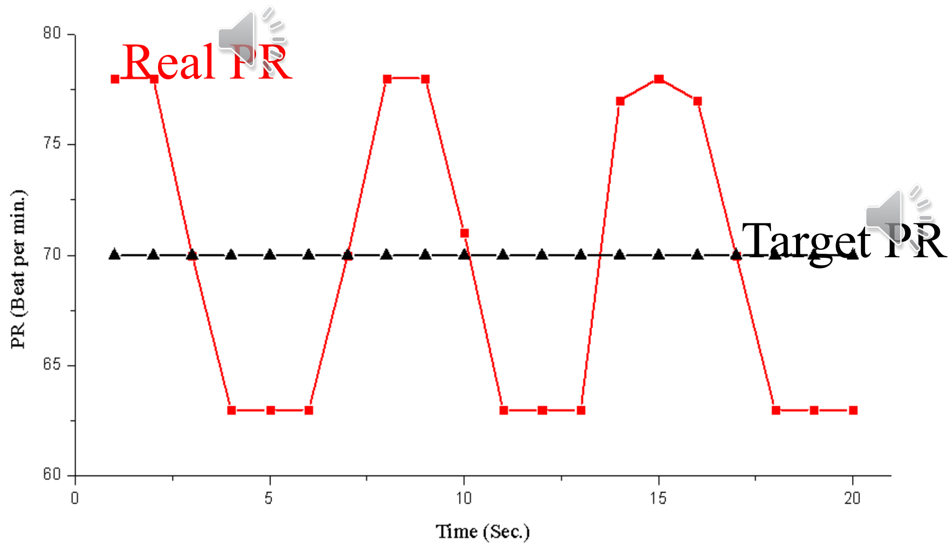
- **Tried to force the system to engage in predefined emergency modes**
- **Achieved 80.6%, 170 out of 211 probes, coverage**

ID	Test inputs	Precondition						Related Guide line	Description
		PR	SL	Control mode	Running mode	Pump position	Pump direction		
TC1	START	50	60	PR	STOP	Center	N/A	1	Baseline
TC2	Plus (10 times)	50	60	PR	RUNNING	N/A	N/A	2	Increasing pump rate
TC3	Plus	150	60	PR	RUNNING	N/A	N/A	3	Upper boundary checking for pump rate
TC4	Minus (10 times)	50	60	PR	RUNNING	N/A	N/A	2	Decreasing pump rate
TC5	Minus	5	60	PR	RUNNING	N/A	N/A	3	Lower boundary checking for pump rate
TC6	SL	50	60	PR	RUNNING	N/A	N/A	1	Changing the control mode
TC7	Plus (10 times)	50	60	SL	RUNNING	N/A	N/A	2	Increasing stroke length
TC8	Plus	50	80	SL	RUNNING	N/A	N/A	3	Upper boundary checking for stroke length
TC9	Minus (10 times)	50	60	SL	RUNNING	N/A	N/A	2	Decreasing stroke length
TC10	Minus	50	30	SL	RUNNING	N/A	N/A	3	Lower boundary checking for stroke length
TC11	PR	50	60	SL	RUNNING	N/A	N/A	1	Changing the control mode
TC12	Plus (10 times)	50	60	PR	STOP	N/A	N/A	2	Same with TC2 but in stopped mode
TC13	Plus	150	60	PR	STOP	N/A	N/A	3	Same with TC3 but in stopped mode
TC14	Minus (10 times)	50	60	PR	STOP	N/A	N/A	2	Same with TC4 but in stopped mode
TC15	Minus	5	60	PR	STOP	N/A	N/A	3	Same with TC5 but in stopped mode
TC16	SL	50	60	PR	STOP	N/A	N/A	1	Same with TC6 but in stopped mode
TC17	Plus (10 times)	50	60	SL	STOP	N/A	N/A	2	Same with TC7 but in stopped mode
TC18	Plus	50	80	SL	STOP	N/A	N/A	3	Same with TC8 but in stopped mode
TC19	Minus (10 times)	50	60	SL	STOP	N/A	N/A	2	Same with TC9 but in stopped mode
TC20	Minus	50	30	SL	STOP	N/A	N/A	3	Same with TC10 but in stopped mode
TC21	PR	50	60	SL	STOP	N/A	N/A	1	Same with TC11 but in stopped mode
TC22	STOP	N/A	N/A	N/A	RUNNING	Top	Up	4	Pressing STOP button when the pump is going up from the center position
TC23	STOP	N/A	N/A	N/A	RUNNING	Center	Up	4	Pressing STOP button when the pump is passing the center position and the direction is up.
TC24	STOP	N/A	N/A	N/A	RUNNING	Bottom	Up	4	Pressing STOP button when the pump is going up from the bottom
TC25	STOP	N/A	N/A	N/A	RUNNING	Top	Down	4	Pressing STOP button when the pump is going down from the top
TC26	STOP	N/A	N/A	N/A	RUNNING	Center	Down	4	Pressing STOP button when the pump is passing the center position and the direction is down.
TC27	STOP	N/A	N/A	N/A	RUNNING	Bottom	Down	4	Pressing STOP button when the pump is going down from the center position
TC28	Disable center hall sensor	N/A	N/A	N/A	RUNNING	N/A	N/A	5	Triggering emergency mode 1
TC29	Block air valve	N/A	N/A	N/A	RUNNING	N/A	N/A	5	Triggering emergency mode 2

In-Vitro Testing: Results

- **Found two behavior patterns, previously unknown to KAOC staff, that appeared abnormal**



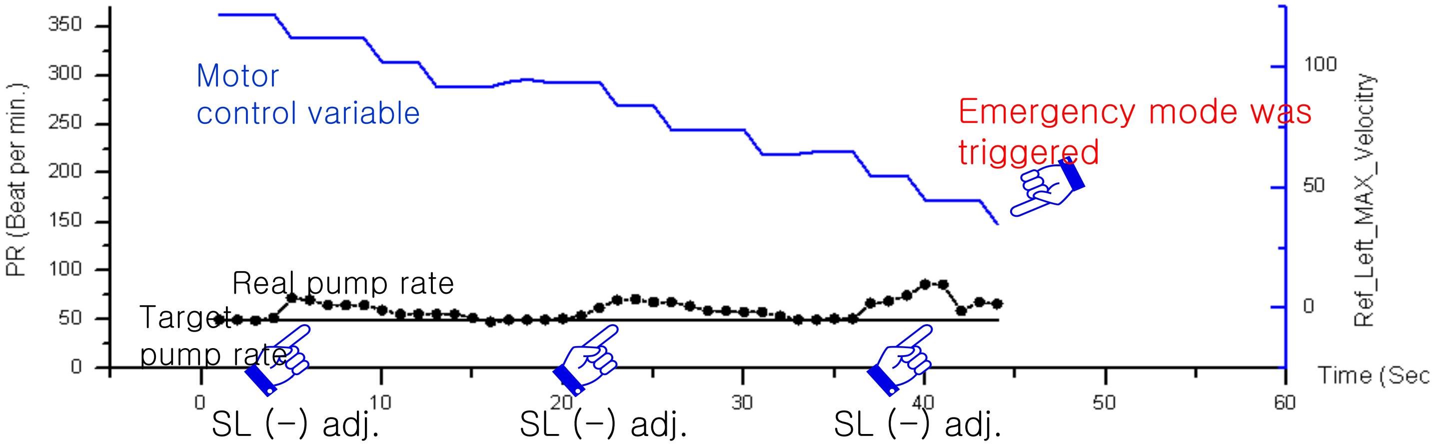


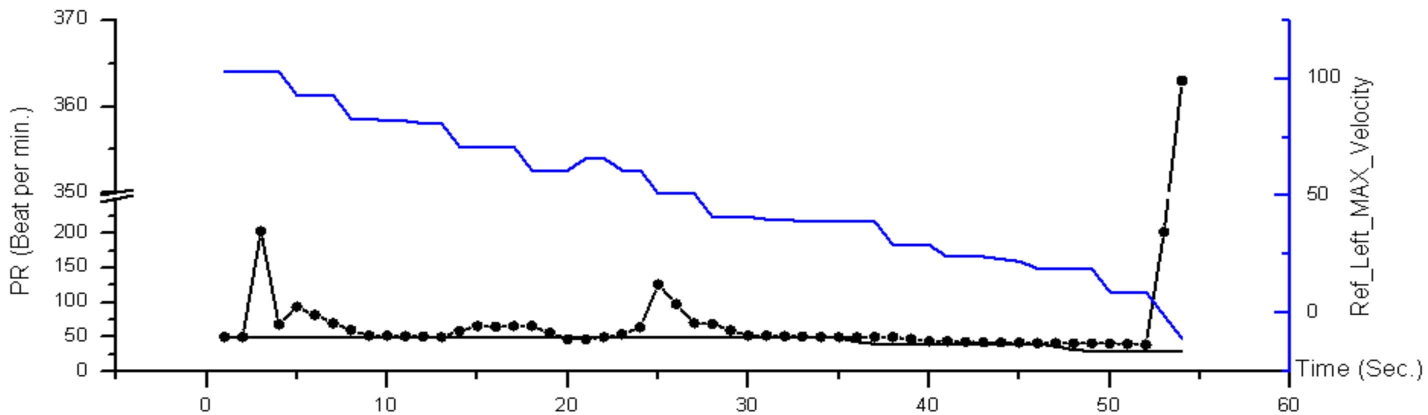
Animal Testing and H-VAD Software

- **Used two 3-months old piglets**
- **78.7% code coverage (cf 80.6%)**
  - Due to our inability to force/repeat certain test cases without endangering the test animal's life

Animal Testing: Results

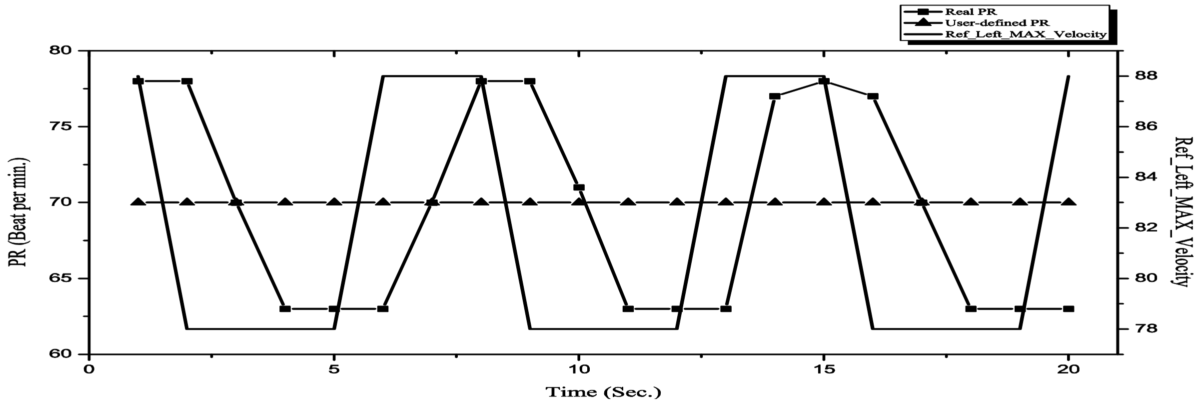
- **Could NOT recreate in-vitro testing result**





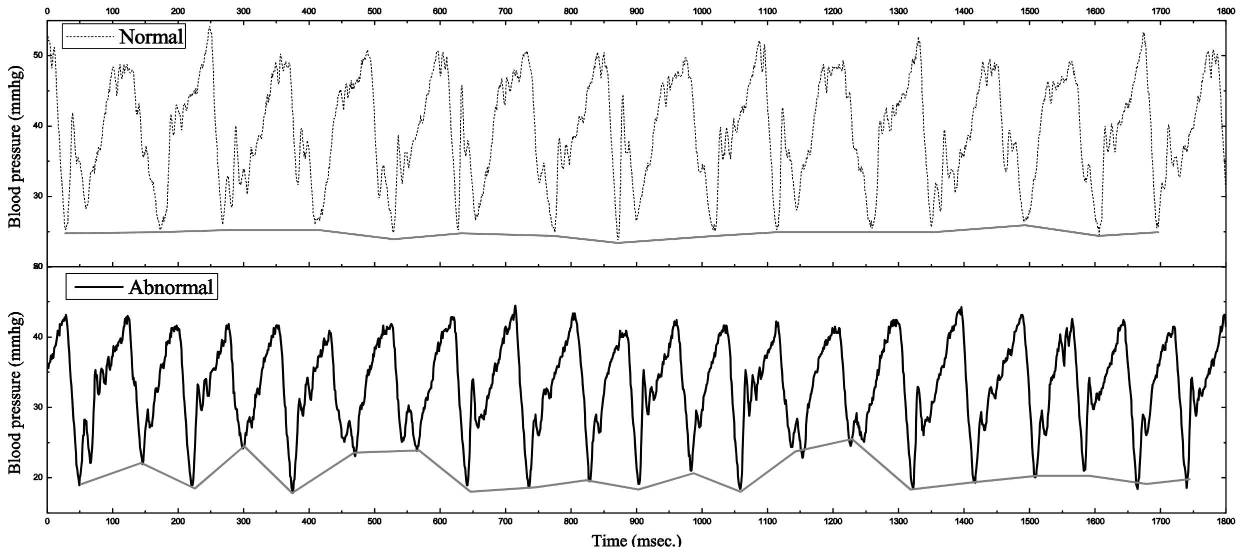
Animal Testing: Results

- **Abnormal pumping pattern, off by 7, repeated**



Normal

Abnormal



H-VAD Lessons

- **SE techniques, although common sense and trivial to us, are not always applied in practice**
- **"Live" testing is expensive, difficult, time-consuming, …**
  - It is extremely difficult to make credible claims on software quality in safety-critical setting
- **"All NEW H-VAD" project could not be launched**
  - Guide hardware design so as to simplify software design and enhance software safety assurance
- **Dedicated and continuous involvement of domain experts are crucial to the success**

- **Safety-Critical Systems Symposium, Feb 2011, Southampton, UK**

Testing of Safety-Critical Software Embedded in an Artificial Heart

Sungdeok Cha[1], Sehun Jeong[1], Junbeom Yoo[2] and Young-Gab Kim[1]

[1]Korea University, Seoul, Korea

[2]Konkuk University, Seoul, Korea

Abstract Software is being used more frequently to control medical devices such as artificial heart or robotic surgery system. While much of software safety issues in such systems are similar to other safety-critical systems (e.g., nuclear power plants), domain-specific properties may warrant development of customized techniques to demonstrate fitness of the system on patients. In this paper, we report results of a preliminary analysis done on software controlling a Hybrid Ventricular Assist Device (H-VAD) developed by Korea Artificial Organ Centre (KAOC). It is a state-of-the-art artificial heart which completed animal testing phase. We performed software testing in in-vitro experiments and animal experiments. An abnormal behaviour, never detected during extensive in-vitro analysis and animal testing, was found.

KOREA UNIVERSITY
College of Informatics

Conclusions

- **Interdisciplinary research is important and do-able, but difficult**
- **Software engineering can and should play important roles in software-driven and software-intensive society**
- **Support domain experts to do their work well**
  - We must learn to work with domain experts