Based on "Formal Modeling and Verification of Safety Critical Software", IEEE Software, 2009.

정형 요구사항 명세 및 검증기반 원자력 소프트웨어 개발 방법론

JUNBEOM YOO Dependable Software Laboratory

> KONKUK University http://dslab.konkuk.ac.kr

2010.09.09

Contents

- Introduction
 - Safety-Critical Software in Nuclear Power Plants
 - Software Development Process (Existing vs. Proposed)
- Software Development Process for NPPs
 - Development Process
 - Verification Process
 - Safety Analysis Process
- Conclusion and Future Work

Introduction

- 1. Safety-Critical Software in Nuclear Power Plants
- 2. Software Development Process (Existing vs. Proposed)

Safety-Critical Software in Nuclear Power Plants

- **RPS** (Reactor Protection System)
- ESF-CCS (Engineering Safety Features Component Control System)



Existing Software Development Process

• For Most NPPs in Korea (e.g. Wolsung NPP)



Proposed Software Development Process

- For KNICS RPS for APR-1400 [1] (http://www.knics.re.kr)
 - APR-1400 : Next generation nuclear reactor being developed in Korea



Software Development Process for NPPs

- 1. Development Process
- 2. Verification Process
- 3. Safety-Analysis Process

Development Process

- 1. Formal Requirements Specification
- 2. Automatic Design Synthesis
- 3. FBD Testing



1. Formal Requirements Specification

• NuSCR [3]

- Formal requirements specification language
- Customized SCR [2] for nuclear applications
 - Listened to opinions offered by domain experts
- 4 constructs
 - SDT (Structured Decision Table)
 - FSM (Finite State Machine)
 - TTS (Timed Transition System)
 - FOD (Function Overview Diagram)

Conditions		
$k_X_MIN <= f_X <= k_X_MAX$	Т	F
Actions		
$f_X_Valid := 0$	X	
f_X_Valid := 1		X

SDT



1. Formal Requirements Specification

- NuSRS (ver 2.0)
 - CASE tool supporting
 - NuSCR specification
 - Self-Checking (on-going)
 - SMV program translation (NuSCR → SMV)
 - SMV verification (CTL Model Checking)
 - Case Study
 - KNICS-RPS-SRS101, Rev,00, 2003.
 (by NuSRS 1.0)
 - KNICS-RPS-SVR131-01, Rev.00, 2005. (by NuSRS 2.0)



2. Automatic Design Synthesis

- NuSCRtoFBD Synthesis Procedure [8]
 - Synthesizes FBD programs from NuSCR specification automatically
 - More than twice FBD blocks than manually coded and optimized ones
 - Unused in the project, because unable to develop CASE tools in advance
 - However, can be used as a baseline for FBD programming in design phase

- NuSCRtoFBD (ver 1.0)
 - CASE tool supporting
 - Automatic FBD synthesis from NuSCR
 - Reads NuSCR specification in XML format
 - Stores FBD programs in standard XML format (on-going)
 - Algorithm is being optimized



NuSCRtoFBD (ver. 1.0)

- Synthesized from KNICS RPS BP SRS (KNICS-RPS-SVR131-01, Rev.00, 2005)



NuSCRtoFBD (ver. 2.0) - Synthesized from KNICS RPS BP SRS (KNICS-RPS-SVR131-01, Rev.00, 2005)

3. FBD Testing

- Direct FBD Testing
 - Eunkyoung Jee, Junbeom Yoo, Sungdeok Cha, and Doohwan Bae, "A Data Flow-based Structural Testing Technique for FBD Programs," Information & Software Technology, Vol.51, No.7, July, pp.1131-1139, 2009.



FBD Tester (ver. 1.0)

Verification Process

- 1. Model Checking Requirements
- 2. Model Checking Design
- 3. Equivalence Checking Designs



1. Model Checking Requirements

- Formal verification for requirements specification
 - Target : NuSCR formal specification
 - Tool : Cadence SMV [5]
 - Technique : CTL model checking
- NuSRS (ver. 2.0)
 - Automatic translation from NuSCR into SMV programs [10]
 - Seamless execution of SMV
 - Case Study
 - KNICS-RPS-SVR131-01, Rev.00, 2005
 - Found 157 errors (25 critical)

SMV						
- SMV Input for f_VAR_OVER	_PWR_Val_Out					
MODULE m f VAR OVER P	WR Val Out/f \	AR OVER	PWR PV f	VAR OVER	PWR Ma	inu Tes
/AR					_	
f_VAR_OVER_PWR_Val_Ou	it : 0100;					
inputs						
STATE:{ init_s0_s1};						
, , , , , , , , , , , , , , , , , , ,						
4SSIGN						
nit(STATE) := _init_;						
iext(STATE) := case						
FROMinitTO-s0-taken : s	30;					
FROM-s0-TO-s0-taken : s0;						
FROM- init -TO-s1-taken : SU;	:1:		INI	JSF	S	2.1
FROM-s0-TO-s1-taken : s1;						
FROM-s1-TO-s1-taken : s1;						
1 : STATE;	_					
•						
Save		Close			Execu	tion
PROPERTY						
🖌 default, smv						
Madefault,smv Eile <u>P</u> rop <u>V</u> iew <u>G</u> o	to H <u>i</u> story	Abstract	tion			
Videfault,smv Eile Prop View Go Browser Propertie	to H <u>i</u> story s Results	<u>A</u> bstract	tion Using	Groups	Ì	
Mdetault.smv Eile <u>P</u> rop <u>V</u> iew <u>G</u> o <u>B</u> rowser Pr <u>o</u> pertie	to H <u>i</u> story s <u>R</u> esults	<u>A</u> bstract	tion Usi <u>ng</u>	<u>G</u> roups		
Midetault, smv File Prop View Go Browser Propertie Name	to H <u>i</u> story s <u>R</u> esults	<u>A</u> bstract <u>C</u> one Layer	tion Usi <u>ng</u>	<u>G</u> roups		
Elle Prop View Go Browser Propertie Name ⊡:((op.leve)):	to H <u>i</u> story s <u>R</u> esults	<u>A</u> bstract <u>C</u> one Layer	tion Usi <u>ng</u>	<u>G</u> roups		
Elle Prop View Qo Browser Propertie Name Cycle Pr_Mod_Err	to H <u>i</u> story s <u>R</u> esults	<u>A</u> bstract <u>C</u> one Layer	tion Usi <u>ng</u>	<u>G</u> roups	Ì	
detault smv File Prop Yiew Qo Browser Propertie Name Image: (top level): -cycle -cycle D+f, Mod, Err D+f, Mod, Err D+f, Mod, Correr, PW	to History s <u>R</u> esults /R_Chan_Err	<u>A</u> bstract <u>C</u> one Layer	tion Usi <u>ng</u>	<u>G</u> roups		
Idetault smv Eile Prop Yiew Qo Browser Propertie Name Image: market state sta	to History s <u>Results</u> /R_Chan_Err /R_MT_Query Manu Test	<u>A</u> bstract	tion Usi <u>ng</u>	<u>G</u> roups		
Elle Prop View Go Browser Propertie Name E: (ftolswa): F-cycle D+f_Mod_Err D+f_VAR_OVER_PW D+f_VAR_OVER_PW D+f_VAR_OVER_PW	to History s <u>Results</u> /R_Chan_Err /R_MT_Query /R_Manu_Test	<u>A</u> bstract	tion Usi <u>ng</u>	Groups		
detault smv Eile Prop Yiew Go Browser Propertie Name - Crycle - Crycle - Crycle - H. Mod. Err + T. VAR_OVER_PW + T. VAR_OVER_PW + T. VAR_OVER_PW Source Trace	to History Results /R_Chan_Err /R_Manu_Test g	Abstract	tion Usi <u>ng</u>	Groups		
detault smv File Prop Yiew Go Browser Propertie Name : (too level): - crycle + + f. VAR OVER PW • Gource Irace Lo File Show	to History Results /R_Chan_Err /R_MT_Query /R_Manu_Test s	Abstraci	tion Using	Groups		
detault.smv Eile Prop Yiew Go Browser Propertie Name Image: Constraint of the second se	to History s <u>Results</u> /R_Chan_Err /R_MT_Query /R_Manu_Test g VAR_OVER_PR	Abstraci	tion Using	Groups		
Cetault smv Eile Prop View Go Browser Propertie Name Constraints Constrain	to History s Results /R_Chan_Err /R_MT_Query s vAr_over_Pt vAr_over_Pt	Abstract	don Usi <u>ng</u>	Groups		
detault smv Elle Prop Browser Propertie Name Name Hoolewell: Froycle Hoolewell: Froycle Hoolewell: Froycle Hoolewell: Froycle How AR OVER PW Froycle How AR OVER PW Frowce Source Trace Lo File ShV Input for g SMV Input for f MODULE m f VAR OVER for AR	to History s Results /R_Chan_Err /R_MT_Query R_Manu_Test VAR_OVER_P1 VAR_OVER_P1 VAR_OVER_P1 . PUR Val OL	Abstract	using	<u>Groups</u>		R PWR
defaultsmv Eile Prop Yiew Go Browser Propertie Name B: (top level): - crycle #f.Mod_Err #f.VAR_OVER_PW Seurce Irace Loo File ShOW SHV Input for f_ MODULE m_f_VAR_OVER m_Test, f_VAR_OVER	to History S Results R Chan Err R Manu_Test S VAR_OVER_P1 VAR_OVER_P1 VAR_OVER_P1 *_PUR_Val_OC =P8R_MT_OUR	Abstract <u>Cone</u> Layer JR JR JR JR JR JR JR JR JR JR	ut OVER PW It, sec)	<u>G</u> roups	VAR_OVE	R_PWR
Idefault.smv Eile Prop Yiew Go Browser Propertie Name B: (top.lsys)): - cycle Df.Mod_Err Df.Mod_Err Df.VAR_OVER_PW Df.VAR_OVER OVDULE Input for f_ NODULE m_f_VAR_OVER VAR VAR	to History s Results R_Chan Err R_Manu_Test s var_over_pr var_ov	Abstract <u>Cone</u> Layer	ut OVER PW Le, sec)	<u>G</u> roups	VAR_OVEI	R_PWR
detault smv Eile Prop Yiew Go Browser Propertie Name	to History s Results R_Chan Err R_Manu_Test s var_over_pr var_over_pr var_over_pr var_over_pr var_over_pr to to t	Abstract	ut over pw Le, sec)	<u>G</u> roups	VAR_OVEI	R_PWR
defaultsmv Eile Prop Yiew Go Browser Propertie Itopleyelli Ccycle Hod_Err Hr.Mod_Err Hr.Mod_Err Hr.Wat, OVER_PW Hr.Wat, OVER_PW Hr.Wat, OVER_PW Hr.Wat, OVER_PW Hr.Wat, OVER_PW File Show SHV Input for f_ MODULE m 1 VAR_OVER in Unput for f_ MODULE m 1 VAR_OVER in Unput for f_ MODULE m 1 VAR_OVER in Unput STATE : (_init_, s0	to History s Results R_Chan Err R_Manu_Test s vAR_OVER_PI vAR_OVER_PI vAR_OVER_PI vAR_OVER_PI .PVR_Val_OU .PVR_TT_OUC 	Abstract	Using	<u>G</u> roups R_PV, 1_'	VAR_OVE	R_PWR,
Idetault smv Elle Prop Yiew Go Browser Propertie Name Hoolevell:	to History s Results /R_Chan_Err /R_MT_Query R_Manu_Test s VAR_OVER_PI VAR_OVER_PI VAR_OVER_PI VAR_OVER_PI L_PUR_Val_OU _PUR_BT_Que L_OUT : 0	Abstraci	ut OVER_PW Le, sec)	<u>G</u> roups R_PV, <u>±</u> '	U U U U U U U U U U U U U U U U U U U	R_PWR
defaultsmv Elle Prop Yiew Go Browser Propertie Name B: (top level):	to History S Results RChan Err RCMT_OUER RManu_Test S VAR_OVER_PH VAR_OVER_PH VAR_OVER_PH VAR_OVER_PH L_PUR_VAL_OU L_PUR_VAL_OU L_OUT : 0 J, sl);	Abstract	ut OVER PW Le, sec)	<u>G</u> roups	VAR_OVEI	R_PWR
defaultsmv Eile Prop Yiew Go Browser Propertie Tcycle Trace Lo File Show SNV Input for f Mobule matterst, f_VAR_OVER VAR Tcycle, f_VAR_OVER Tcycle, f_V	to History s Results R_Chan_Err R_MAT_Query R_Manu_Test s var_over_Pi var_over_Pi var_over_Pi var_over_Pi i_PWR_Val_ov PWR_TT_ouc i_Out : 0 i, s1); c;	Abstract Cone Layer JR JR JR JR JR JR JR JR JR JR JR JR JR	at OVER_PW Ie, sec)	<u>Groups</u>	VAR_OVEI	R_PWR.
default smv Eile Prop View Go Browser Propertie Name D:	to History s Results R_Chan_Err R_Manu_Test g VAR_OVER_PI VAR_OVER_PI VAR_OVER_PI PUR_VAI_OU _PUR_VAI _PUR_VAI	Abstraci	ut OVER_PW Ie, sec)	<u>G</u> roups	VAR_OVEI	R_PWR.
defaultsmv Elle Prop Yiew Go Browser Propertie Image: State of the state	to History s Results R Chan Err R Manu_Test yur_over_pi _pwr_wr_over_pi _pwr_wr_ou_ov _i_out : 0 i, sl); _t_;	Abstract	ut OVER PW Le, Sec)	<u>G</u> roups	VAR_OVEI	R_PWR.
defaultsmv Eile Prop View Go Browser Propertie Comparison of the second seco	to History s Results R Chan Err R Manu_Toury R Manu_Toury s s var_over_pi var_over_pi var_over_pi i_pvr_val_ov pvr_v	Abstract Cone Layer Layer IR IR IR IR UR UR UR UR UR UR UR UR UR UR UR UR UR	ut OVER_PW Ie, sec)	<u>Groups</u>	VAR_OVEI	R_PWR.
default smv Elle Prop Ylew Go Browser Propertie Name Name El. (100 Usyn):	to History s Results R_Chan Err R_Manu_Test s var_OVER_PI var_OVER_PI var_OVER_PI var_OVER_PI .pur_nt_Out .pur_nt_Out .cout	Abstract Cone Layer IR IR_Val_O It (f_VAR Ity, Cyc 100;	ut Using	<u>G</u> roups R_PV, <u>f</u> _'	UVAR_OVET	R_PWR
detault smv Elle Prop View Go Browser Propertie Name Name Browser Propertie Prover Propertie Browser Propertie Browser Propertie Browser Propertie Browser Propertie Browser Trace Source Trace File Show SNV Input for f NOPULE f VAR_OVER_PWR_Ve inputs STATE : (_init_, s0 ASSION init (STATE) := init (STATE) := FROM Prov FROM Prov FROM De FROM De Inct (STATE) = State : FROM De FROM De FROM De FROM De FROM De	to History s Results /R_Chan_Err /R_Manu_Test s VAR_OVER_PI VAR_OVER_PI VAR_OVER_PI VAR_OVER_PI (1, s1); (1, s1); (Abstraci	ut OVER_PW le, sec)	<u>Groups</u> R_PV, <u>f_</u> '	VAR_OVET	R_PWR
detault smv Elle Prop View Go Browser Propertie Name B: (top level): 	to History s Results R Chan Err R MT_Ouery R Manu_Test s VAR_OVER_PH VAR_OVER_PH VAR_OVER_PH VAR_OVER_PH UAR_OVER_PH (Abstract	ut OVER PW Le, sec)	<u>Groups</u>	VAR_OVE	R_PWR
detault smv Elle Prop View Go Browser Propertie Name E: (toolsyn):	to History s Results R Chan Err R MT_Query R Manu_Test s vAR_OVER_P1 VAR_OVER_P1 VAR_OVER_P1 vAR_OVER_P1 vAR_OVER_P1 is is result is to is is is is is is is is is is	Abstract Cone Layer IR IR IR IR IR IR IR IR IR IR IR IR IR	Using Using	<u>G</u> roups	VAR_OVE	R_PWR
default smv Ele Prop View Go Browser Propertie Name Browser Propertie Direction Name Browser Propertie Direction Propertie Browser Propertie Direction Propertie Direction Propertie Browser Trace Lo File Source Trace File Show SHV Input for f MODULE m f VAR_OVER PWR_Ve inputs STATE : (_init_, s0 ASSIGN Shit (STATE) := _insi init (STATE) := _insi rease FROM-initTO-s0 rewor.sourcosurcase FROM-s1-TO-s0-tak FROM-s1-TO-s1-tak 1: STATE; esac;	to History s Results R_Chan_Err R_Manu_Test g VAR_OVER_PI VAR_OVER_PI VAR_OVER_PI PUR_Val_OC _PUR_TI_OUE L_DUR_t 0 i, s1); taken : s0; taken : s1; ren : s1;	Abstract Cone Layer IR IR_Val_O It (f_VAR ILy, Cyc 100;		Groups R_PV, f_	VAR_OVEI	R_PWR



FBD Verification using - SMV model checking & VIS Equivalence checking

2. Model Checking Design

- Formal verification for design specification
 - Target : FBD program
 - Tool : Cadence SMV [5]
 - Technique : LTL model checking
- FBD Verifier (ver. 1.0 / 2.0)
 - Automatic translation from FBD programs into Verilog programs [11]
 - Seamless execution of SMV
 - Case Study
 - KNICS-RPS-SDS231, Rev.01, 2006
 - Found 60 errors (13 critical)



	6 ¹⁰	MAL (NO		- ARC	.001. (N	k			. (s - 3	L.D.NT	80
PY_00	181	LE.REK			#82.803	<u> </u>			-	98.	Just -PTRP.
8128	182	_	PTRIPLUSIC	- ne			0		191		
						B 430	2181 290	5 1	182		
					PTRIP_ON		ADD2_D1WT	_			
						-182					
	p.	201 (NO	k					-	051.00X D	a' i	
19.00-		GE_DINT			<u> </u>			-	380	PTRIP.	TIRKC .
wor.	162						PIRIPLINIC-	181			
							14.6	182		<u> </u>	
								(n)	10. MA. 0	a' i	
					۰L				5090	-mp	
					en 100	UR. m	PEP-				
			PTSP-			A002,904		182			
			PHS-		182						
	_				-						

Engineering Tools by PLC vendors

3. Perform SMV model checking

T FD	LFIISING, V										10
Elle	Prop View	Goto I	History	Abstrac	ction						He
					Links						
Die	wser prop	ercies	Seance	Toue	Using	GLO	ups				
	Name	La	yer								
8	PTRP_LOG PTRP_LOG PTRP_LOG PTRP_LOG PTRP_LOG PTRP_LOG PTRP_LOG PTRP_LOG PTSP_1 PTSP_1 PTSP_2 PTSP_3 PTSP_3 PTSP_3	IC IC_1 IC_2 IC_3 IC_out									
Sou	arce Trace	Log									
FIR	e show										
hod	ule main (c)	lk, PV_OU	T, TRI	P_LOGIC_	out, T	SP_out	, PTP	IP_LOG	IC_ow	t, PTSF	, ont) 1
	//constants	,									
	define	HY5	0								
	define	PHYS	0								
	define	PTSP_K	0								
	`define	TSP_K	0								
	input clk;										
	reg		PIRIP	_rooic;							
	reg	[710]	PISPI								
	input	[710]	PV_00	TJ							
	reg		TRIP	LOGIC;							
	reg	[7:0]	1251								
			TRTR	LOSIC 11							
	wire	17:01	THP 1		·						
	wine	[TRYR	icerc a.							
	wire	17:01	TRP 2	1							
	wire		TOTO	LOATE A.							
	WILE	(7:0)	TSP 3		,						
	ONEDNE	[1.0]	TRTR	LOGIC or							
	out hat	10.00	TOD -	www.su_ot	M V P						
1	oucput	17101	101_0	957							
									1	-search	

SMV

ID Verifier								-					o' 🖬 🗟
Verity Help													
C Robrant to	0 4	Verleg	CE from	P2									
t of FDC_RISINO	• Disp	Displaying veriable: Input 🗹 Reg 🗹 Output 🗹											121 #
ompiete S1)	0	riginal CE	Redu	ced form	Timing	graph	Sliced vars	Warn	ings				
TIMERIN >=													
0(PA	OUT TS	ø										
xe (90	,	90	90	90	90	90	90	90	90	90	
N1 >=	TI	P. THER	0 7	RIP_1.00	IC out								
N2 =			_	-	- I								1
1997,		-	-	-	-	-	-	-		-	-	-	-
ATTER LOOKS	ШĘ	tout											
and store adversely			_	-	-	-	_	-	-	-	-	-	
_L00iC_1_out =	16	OP_TIMER	_IN	_	-	-	_	-	-	-	-	-	- II
LL													
THP_THER_O	TP	P_LOGIC	1,00									-	
) ;=													
HIP_LOOIC,	TS	P_1_out											
0	91	90		90	90	90	90	90	90	90	90	89	69
t out =	TS	Post											
LC			_										
3	1	90)	90	90	90	90	90	90	90	90	89	90
a land	2	-	-										
top,	Πř	001341	50.	-	-	-	_	-		-	-	-	-
2													
NT #													
TEP,	•												
N2 30	. Mor	itoring var	iable P	_OUT >= 1	sp								-
nded with code 0													

Counterexample viewer in FBD Verifier 1.0

3. Equivalence Checking Designs

- Formal verification for design specifications
 - Target : Two FBD programs
 - Tool : VIS Verification System [4]
 - Technique : Equivalence checking, Simulation
- VIS Analyzer (ver. 1.0)
 - Seamless execution of VIS (VIS has no GUI)
 - Visualization of VIS's process and verification results [12]
 - Unused in the project, because unable to develop CASE tools in advance
 - Case Study
 - KNICS-RPS-SDS101, Rev.00, 2005
 - No official result

Trip Logic	Error Type	Compared FBD (Num. of Errors)	Original FBD (Num. of Errors)
Fixed Set-Point Rising Trip	Syntactic	0	0
without Operating Bypass	Logical	0	1
Manual Reset Variable Set-Point Trip	Syntactic	0	3
without Operating Bypass	Logical	6	2 20

Engineering Tools by PLC vendors



1. Read two FBD programs in XML format

FBD Verifier 1.0



3. Read two Verilog programs

VIS Analyzer 1.0 (Source)

Verilog sources Result Result Table			-
Verilog 1		Verilog 2	u
KC2007(TeXhomelVIShis-2.0texamples)RPSIFBD_Verifierth_X_Pretrip_Manual	l.v	C.KC2007/TeXhomelVIShis-2.0lexamples/RPSIFBD_Verifierth_X_Pretrip_Mech.v	
<pre>ypedef enum (80, 01) th_z Pretrip_state; ypedef enum (70, 71, 72, 73, 74, 75, 76, 77, 78, 9, 710, 711, 712, 713, 714, 715, 716, 717, 710, 19, 720) time_state; define k_Dratrip_Marpaint 30 define k_zPretrip_Mys 10</pre>	•	typedef enum (80, 81, 82) th, X Pretrip_state; typedef enum (80, 71, 72, 73, 74, 75, 76, 77, 70, 79, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720) timer_state; 'define & Dratrip_state; 30 'define & Z Pretrip_Mys 10	
/ define k_Trip_Delay 20		//`define k_Trip_Delay 20	Ц
<pre>/ th_%_Pretrip module odule th_%_Pretrip(clk, f_%, th_%_Pretrip);</pre>	-	<pre>// th_X_Pretrip module module th_X_Pretrip(clk, f_X, th_X_Pretrip);</pre>	
nput clk; nput[0:6] f_X;		<pre>input clk; input[0:6] f_X;</pre>	
utput th_X_Pretrip;		output th_X_Pretrip;	e
/integer wire f_X;		//integer wire f_X;	
/integer f_%;		//integer f_%;	
ire th_X_Pretrip;		wire th_X_Pretrip;	
h_X_Pretrip_state reg state;		th X Pretrip state reg state;	
ag th Prav X Pratrip;		reg th Prev X Pretrip;	
imer_state reg timer;		timer_state reg timer;	
nitial state = 31;		initial state = 20;	
nitial th_Prev_X_Pretrip = 1;		initial th_Prev_X_Pretrip = 1;	
nitial timer = 20;		initial timer = 20;	
ssion th X Pretrin = (state==20 && f X <=			
Bartain Artaniat - 15 W Bartain War 31.		wire Cond a 1;	

VIS Analyzer 1.0 (Result)





Automatic Vis E	iquivalence Checker				• ⁴ ⊠" ⊇
Verilog sour	ces Result Result	Table			ſ
# state	input	File1Output	File2Output	File1State	File2State
0	Initial	Initial	Initial	S1 1 T0	S0 1 T0
1	61	1	1	S1 1 T1	S1 1 T1
2	61	1	1	S1 1 T2	S1 1 T2
3	61	1	1	S1 1 T3	S1 1 T3
4	61	1	1	S1 1 T4	S1 1 T4
5	61	1	1	S1 1 T5	S1 1 T5
6	61	0	0	S0 0 T5	S2 0 T5
7	52	0	0	S0 0 T0	S2 0 T0
8	52	1	0	Null	Null
eadv					

VIS Analyzer (ver. 1.0)

- Visualized and reorganized result - counterexample



VIS Analyzer (ver. 3.0)

- Visualized and reorganized result - counterexample

Safety Analysis Process

- 1. Fault Tree Analysis for Requirements
- 2. Fault Tree Analysis for Design



1. Fault Tree Analysis for Requirements

- Fault Tree Analysis
 - Performed manually
 - Totally depends on analyst's experience and ability
- We provided FTA templates and CASE tool (NuFTA) for NuSCR [13]



2. Fault Tree Analysis for Design

We provided FTA templates and FBD [15]



Conclusion and Future Work

Conclusion

- We proposed software development processes using formal methods
 - Target: KNICS RPS for APR-1400
 - Development process
 - NuSCR formal requirements specification
 - Automatic FBD design synthesis
 - Verification process
 - Model checking NuSCR requirements
 - Model checking FBD design
 - Equivalence checking FBD designs
 - Safety analysis process
 - FTA templates for NuSCR requirements
 - FTA templates for FBD programs
 - Case Study
 - KNICS-RPS-SVR131-01, Rev.00, 2005
 - KNICS-RPS-SDS231, Rev.01, 2006



Future Work

- 1. Integrated Tool-set
- 2. Tool Enhancement
 - Self-checking : completeness & consistency (NuSRS)
 - Synchronous Verilog issue in model checking FBD programs using SMV (FBD Verifier)
 - Optimization of FBD synthesis algorithm (NuSCRtoFBD)
 - Add other functions to VIS Analyzer (VIS Analyzer)
- 3. Traceability Analysis
 - From requirements to design
 - From requirements' FTA to design's FTA
- 4. FBD Testing
 - Measures (coverage criteria)
 - Testing tool support
- 5. Application to Other Domains

정형 요구사항명세 기반 원자력 소프트웨어 개발 방법론





References

- [1] KNICS (Korea Nuclear Instrumentation & Control System R&D Center). http://www.knics.re.kr.
- [2] Kathryn L. Heninger, "Specifying Software Requirements for Complex Systems: New Techniques and Their Application," *IEEE Trans* actions on Software Engineering, SE Vol.6, No.1, pp2-13, 1980.
- [3] Junbeom Yoo, Taihyo Kim, Sungdeok Cha, Jang-Su Lee, Han Seong Son, "A Formal Software Requirements Specification Method f or Digital Nuclear Plants Protection Systems," *Journal of Systems and Software*, Vol.74, No.1, pp.73-83, 2005.
- [4] VIS (Verification Interacting with Synthesis), http:// http://embedded.eecs.berkeley.edu/research/vis.
- [5] SMV (Symbolic Model Verifier), http://www.kenmcmil.com/smv.html.
- [6] Sungdeok Cha, "Pet Formalisms versus Industry-Proven Survivors: Issues on Formal Methods Education," *Journal of Research and Practice in Information Technology*, Vol.32, No.1, pp39-46, 2000.
- [7] Mats P.E. Heimdahl and Nancy G. Leveson, "Completeness and Consistency in Hierarchical State-Based Requirements," IEEE Tran sactions on Software Engineering, Vol.22, No.6, pp363-377, 1996.
- [8] Junbeom Yoo, Sungdeok Cha, Chang Hwoi Kim, Duck Yong Song, "Synthesis of FBD-based PLC Design from NuSCR Formal Speci fication," *Reliability Engineering and System Safety*, Vol.87, No.2, pp287-294, 2005.
- [9] US NRC, Digital Instrumentation and Control Systems in Nuclear Power Plants: Safety and Reliability Issues, *National Academy Press*, 1997
- [10] Jaemyung Cho, Junbeom Yoo, Sungdeok Cha, "NuEditor A Tool Suite for Specification and Verification of NuSCR," In proceeding of Second ACIS International Conference on Software Engineering Research, Management and Applications (SERA2004), pp298-30 4, LA, USA, May 5-7, 2004.
- [11] Junbeom Yoo, Sungdeok Cha, and Eunkyoung Jee, "A Verification Framework for FBD based Software in Nuclear Power Plants," I n the proceeding of 15th Asia Pacific Software Engineering Conference (APSEC), pp.385-392, Beijing, China, Dec. 3-5, 2008.
- [12] Junbeom Yoo, Sungdeok Cha, and Eunkyoung Jee, "Verification of PLC Programs written in FBD with VIS," *Nuclear Engineering a nd Technology*, Vol.41, No.2, 2009, to be published.
- [13] Taeho Kim, Junbeom Yoo, Sungdeok Cha, "A Synthesis Method of Software Fault Tree from NuSCR Formal Specification using Te mplates," *Journal of Korea Institute of Information Scientists and Engineers (in Korean)*, SE Vol.32, No.12, pp1178-1192, 2005.
- [14] Younju Oh, Junbeom Yoo, Sungdeok Cha, Han Seong Son, "Software Safety Analysis of Function Block Diagrams using Fault Tree s," *Reliability Engineering and System Safety*, Vol.88, No.3, pp215-228, 2005.
- [15] Gee-Yong Park, Kwang Yong Koh, Eunkyoung Jee, Poong Hyun Seong, Kee-Choon Kwon and Dae Hyung Lee, "Fault Tree Analys is of KNICS RPS Software," *Nuclear Engineering and Technology*, Vol.40, No.5, pp397-408, 2008.