Formal Methods for System/Software Engineering: NASA & Army Experiences

NAS



Dr. Mike Hinchey/GSFC Caroline Wang/MSFC Josh McNeil/ARMY

Introduction



- What are Formal Methods?
- Problem/Approach
- Challenges
- Recommendations
- Future Plans



Formal Methods - Dr. Mike Hinchey

Formal Methods

- Formal methods are mathematically based techniques for specification, development and verification of systems, both hardware and software.
- The use of formal methods approaches can help to eliminate errors early in the design process.
- Practitioners have also recognized that they can make searching for reusable components more effective by having formal specifications of components.

Current Formal Methods activities within NASA/Army, and International Formal Methods community.

- Pockets of expertise within NASA (specifically ARC, JPL, LaRC) and Army.
- Tools and techniques in use within NASA and Army but not widely used on projects and missions.
- International Formal Methods Community

Problem/Approach

_

General Problem	Approach
System/Hardware/Software complexity	 Provide accurate and appropriate specifications of required system behavior using Formal Methods
 Inadequate requirements specifications / misinterpretation of natural language Significant number of problems introduced due to vague requirements 	 Develop requirement specification as Formal Specification (using formal semantics) to eliminate misinterpretation of vague and incomplete natural language requirements
Significant number of safety and reliability problems are traced to incorrect performance or behavior specifications, or incorrect interfaces	 Use Formal methods to prove safety properties derived from safety analyses Use Formal Methods and deductive apparatus to prove correctness of system behavior and interfaces

Problem/Approach

			——————————————————————————————————————
S	pecific Problem	Ap	oproach
•	Formal Methods Learning Process Difficult for new users	•	Develop specific project related case studies and provide examples for potential users
•	Select development tools No time to learn all the tools Inadequate resource	•	Based on the project size and resources available, select appropriate Formal Methods development techniques and tools
•	Budget and Schedule constraints	•	Support program development and in parallel prove potential savings
•	Differences in priorities between Research and Production environments	•	Many researchers focus on development of new techniques and tools Production or development programs are concerned with delivery of a product Need to build bridges between the research and production environments

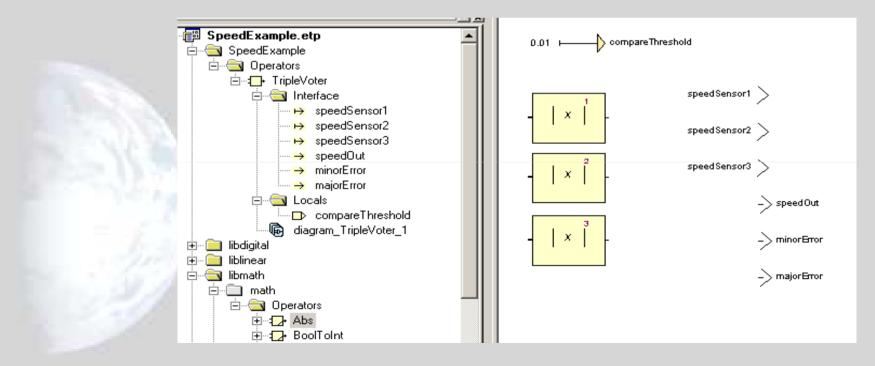
Challenges

- High cost of some commercial development tools.
- Open source free tools do not have adequate training material and support.
- Formal Methods tools require extensive learning process.
- Die-hard Systems and Hardware Engineers are not convinced of the importance of software.



Developing TripleVoter Model

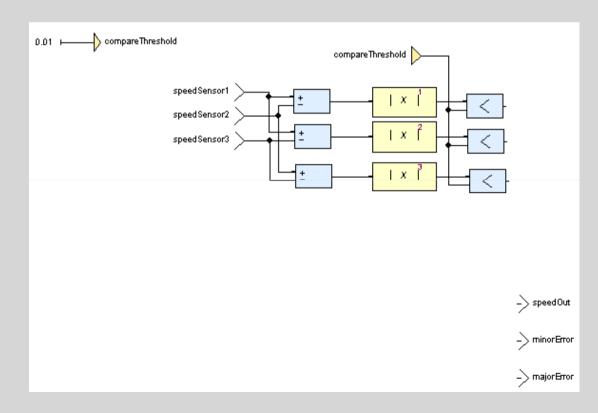
- Double-click the TripleVoter operator to begin modeling.
- Select all variables (speedSensor1, speedSensor2, speedSensor3, speedOut, minorError, majorError, and compareThreshold). Drag them onto the diagram.
- Select the compareThreshold local variable, modify it through Properties → Use, and change its use to Out.



Implementing Model Logic

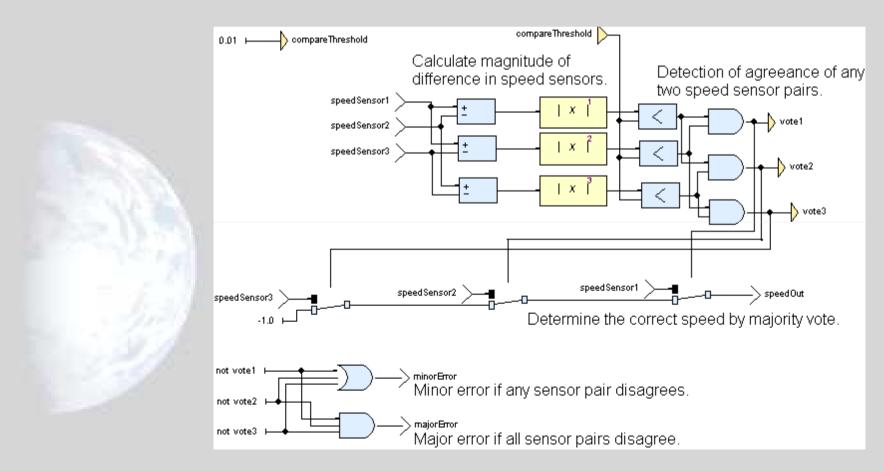
- Connect speedSensorX to the "+" input and speedSensorX to the "-" input of the New Minus operator.
- Connect speedSensor1, speedSensor2, and speedSensor3 to the first input of each New Minus operator.
- Connect all outputs of the New Minus operators to the inputs of the Abs operators.





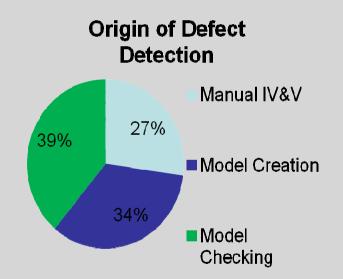
Completing The Model Logic

- Complete other logic components by drag and drop or connections.
- Add new If..Then..Else operators () to the diagram.
- Add comments to model for readability
- **Design Verification** Design Verifier can be used to develop properties that can be proven by formal methods.



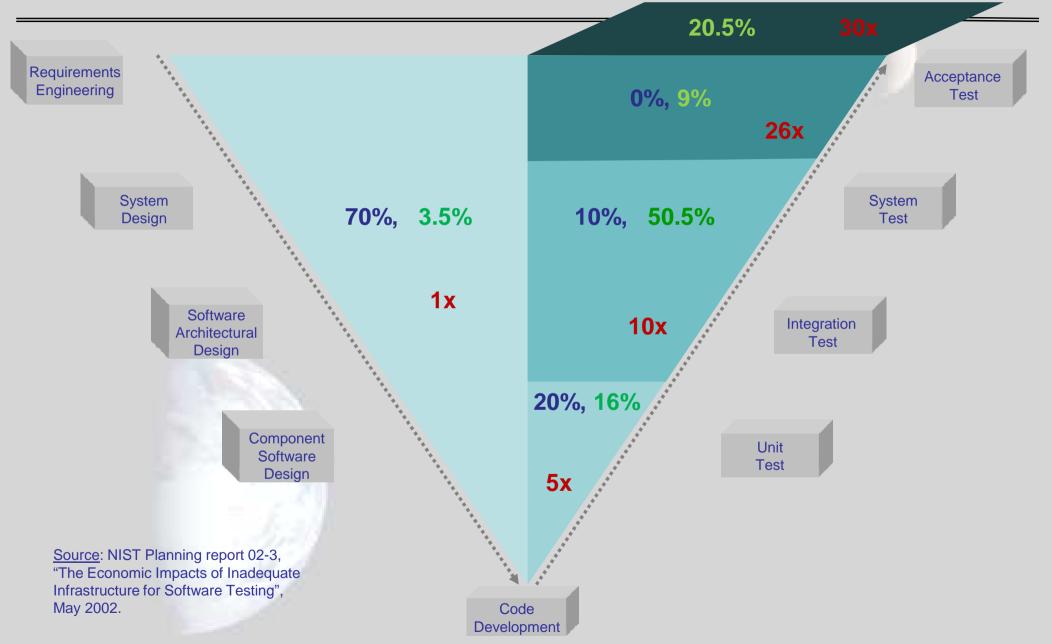
Army's experience and Return on Investment

- Formal methods approach using SCADE method found 144 defects *their* traditional IV&V would miss (73% of all defects found)
- Estimating it would cost approximately 3500 man hours at \$100 per man hour to fix the 144 defects later in the lifecycle
- Early defect removal savings is \$350K
- The cost to perform formal methods analysis: -\$137K
- Net savings of \$213K or 5% of the total project

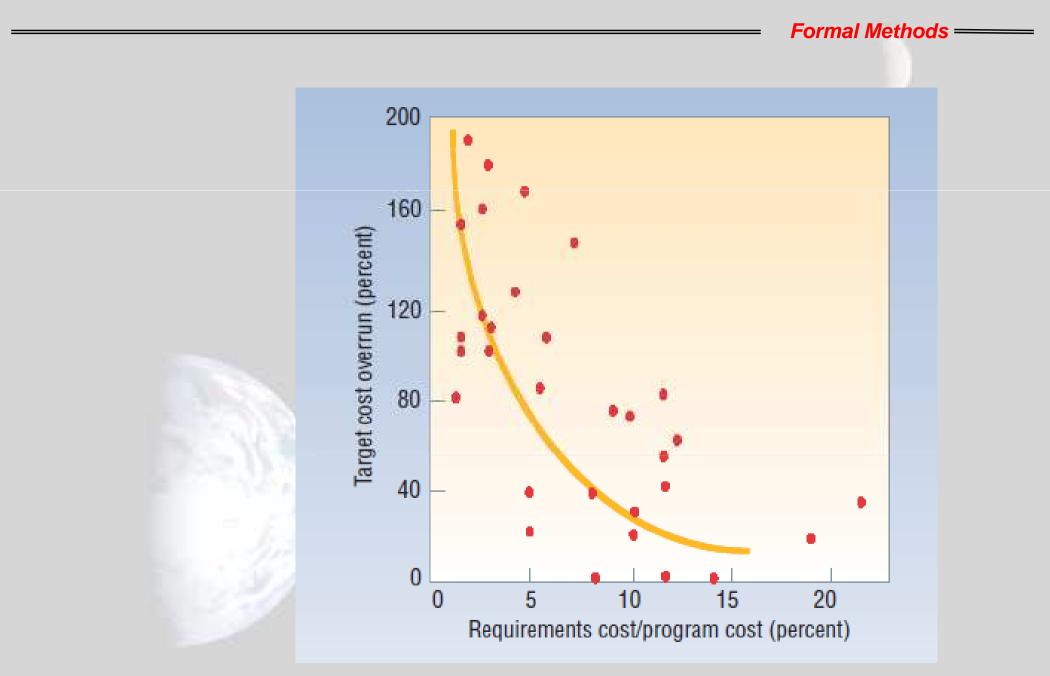


Savings could be even higher if defect detected earlier

The Army "V" concept Where are faults introduced, discovered and cost for removal



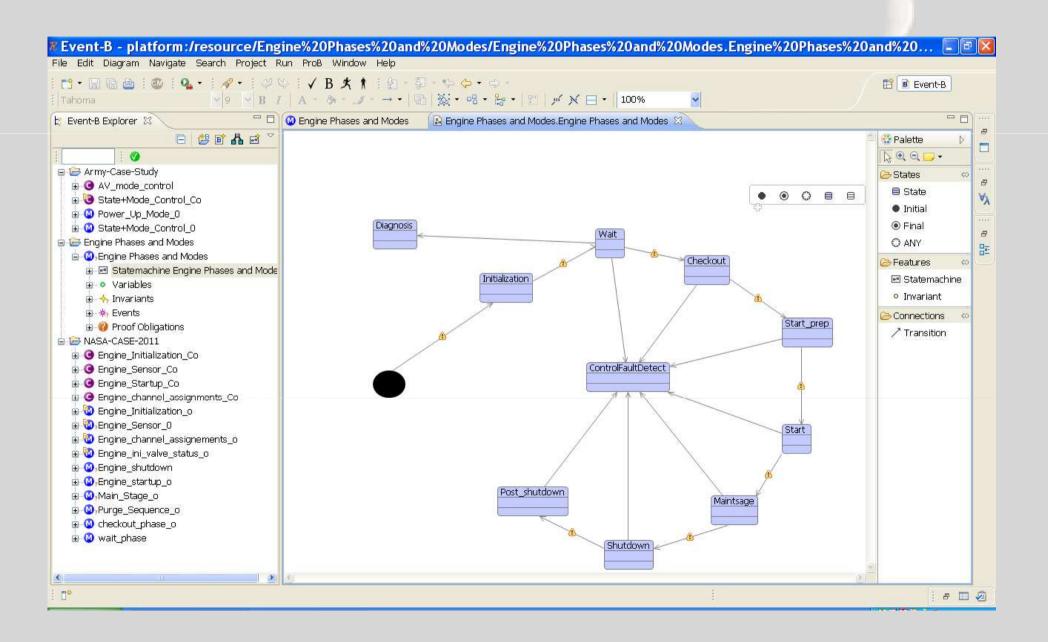
NASA Cost overruns



NASA MSFC Experience in this study

- Using open source development environments
 - B-Tool kit
 - Rodin Event B
 - EA UML
 - Integrated Rodin Event B and UML B
- Currently migrating all the work to the integrated Rodin Event B and UML B.
- Developed top level diagram and state machine in UML B, and used auto translator to translate into Rodin Event B.
- Using Rodin Event B platform for detailed refinement.
- The community is working on auto coding from Event B.

UML-B Statemachine



Auto Translation to Event B

_

Image: State-Mode_Control_Control_Control_Control Factors Image: State-Mode_ControlFaultDetect = ControlFaultDetect = ControlFaultDe	Image: State Control State St		es/Engine Phases and Modes.bum - Rodin Platform	đ
Event & Explore S C engine Phases and Modes S Engine Phases and Modes Engine Phases and Modes Wards and the set of the set o	Evente Explore ?			
<pre>MACHINE Engine Phases and Modes VARIABLES Initialization State-Mode_Control.Co Stat</pre>	Image: State-Mode Control			-21
<pre>Multiple White Mode Control Co Was Academy Control Co Was Con</pre>	<pre>Maintsage.of Maintsage.o</pre>		😡 Engine Phases and Modes 🖄 📓 Engine Phases and Modes.Engine Phases and Modes 👘 🖓	3
<pre>Engine Phases and Modes # Arrw-Case-Sudy # @ AV_mode_control @ @ Power_Up_Mode_0 @ @ Power_Up_Mode_0 @ @ Power_Up_Mode @ @ Power_Up_Mode @ @ Pomer_Bases and Modes # @ Power_Distance # @ Power_Subt # @ Prof Obligations # @ Prof Obligati</pre>	<pre>Engine Phases and Modes # Army-Case-Study # Army-Case-Study # Army-Case-Study # O State+Mode_Control_Co # O Former_Up_Mode_0 # O State+Mode_Control_Co # O Former_States and Modes # O Former_Up_Mode_0 # O Former_States and Modes # O Former_Up_Mode_0 # O Former_States # O Form</pre>	🕒 🙋 🖻 🔒 🖻 🎽	MACHINE	~
<pre>VARIABLES VARIABLES V</pre>	<pre>VARIABLES VARIABLES V</pre>	÷ 📀		
 A V_mode_control A V_mode_control State+Mode_Control_Co State+Mode_Co State+Mode State+Mode_Co <	 A v_mode_control A v_mode_control B state-Mode_Control_Co C horev_top_Mode_0 G State-Mode_Contol_O S state-Mode_Contol_O S	Army-Case-Study		
Initialization Initi	 State+Mode_Control_Co Prower_Up_Mode_O State+Mode_Control_Co Prower_Up_Mode_O Start_prep Figne Phases and Modes Engine Phases and Modes Start_prep <l< td=""><td></td><td></td><td></td></l<>			
• Orwer_Up_Mode_0 • Wall · Startpre • Orage Statemachine Engine Phases and Modes · Wall · Wa	 Power_Up_Mode_0 Power_Up_Mode_0 Power_Up_Mode_0 Pengne Phases and Modes Pengne Phases and Modes Pengne Phases and Modes Post_shutdown Post_shutdown = PALSE Maintsage_init : Maintsage = FALSE 			
 G State-Mode_Control_0 G Engine Phases and Modes G Engine Statu-D_O G Engine_Startup_O G OrdeoLout_Phase_O G Main_Stage_O G Main	 State-Mode_Control_D Engine Phases and Modes Start_prep Statemachine Engine Phases and Modes Start_prep Statemachine Engine Phases and Modes Start Maintsage Shutdown Variables Proof Colligations Proof Colligations Engine_Initialization_Co Engine_Initialization_Co Engine_Initialization_Co Engine_Initialization_Co Engine_Sensor_Co Engine_Startup_Co Engine_Initialization_co Engine_Initialization_co Engine_Startup_Co Engine_Startup_Co Engine_Statup_Co 			
<pre>Engine Phases and Modes ■ Engine Phases and Modes ■ Engine Phases and Modes ■ Engine Phases and Modes ■ Variables ■ A transis ■ A transi</pre>	<pre>Engine Phases and Modes Gengine Phases Gengine Phases and Modes Gengine Phases Geng</pre>			
 ¹ Brighe Phases and Modes ¹ B Statemachine Engine Phases and Modes ¹ • Variables ¹ • Post_shutdown ¹ Diagnosis ¹ • Crof Obligations ¹ • Post_shutdown ¹ Diagnosis ¹ • Crof Obligations ¹ • Post_shutdown ¹ Diagnosis ¹ • Crof Obligations ¹ • Post_shutdown ¹ Diagnosis ¹ • Crof Obligations ¹ • Post_shutdown ¹ Diagnosis ¹ • Crof Obligations ¹ • Post_shutdown ¹ Diagnosis ¹ • Crof Obligations ¹ • Post_shutdown ¹ Diagnosis ¹ • Crof Obligations ¹ • Post_shutdown ¹ • Crof Obligations ¹ • Post_shutdown ¹ • Post_shu	• Bingine Phases and Modes • Bingine Phases and Modes • Bingine Phases and Modes • O Variables • O Variables • Privariants • Prof Obligations • Privariants • O Prof Obligations • Descience_Initialization_Co • Engine_Initialization_Co • Engine_Sensor_Co • Engine_Sensor_Co • Display_Entralization_o • Display_Entraliza			
Statemachine Engine Phases and Mode Shutdown Post_Shutdown Post_Shutdown Post_Shutdown Post_Shutdown Post_Shutdown Diagnosis ControlFaultbetect INVARIANTS Post_Shutdown Initialization_type : Initialization < BOOL	 B Statemachine Engine Phases and Mode C Variables A Variables<td></td><td></td><td></td>			
 Post_shutdown Diagnosis Post_shutdown Diagnosis Post_shutdown Diagnosis Post_shutdown Diagnosis Post_shutdown Diagnosis Post_shutdown Post_shu	 Post_shutdown Diagnosis Post_shutdown Post_shutdown Post_shutdown ∈ BOOL Post_shutdown = FALSE Maintsage.init : Maintsage = FALSE Post_shutdown = FALSE 			
Bignosis ControlFaultDetect Bignosis ControlFaultDetect Bignosis Bignosis ControlFaultDetect INVARIANTS Bignosis Bignosis Bignosis Bignosis ControlFaultDetect INVARIANTS Bignosis Initialization.co Bignosis Bignosis Checkout Spect Bignosis Start_type	 b) Variants c) Variants c) Prof Obligations Diagnosis c) Prof Obligations NASA-CASE-2011 c) Engine_Sensor_Co c) Engine_Sensor_Co c) Engine_Sensor_Co c) Engine_Sensor_Co c) Engine_channel_assignments_Co c) Engine_channel_assignments_o c) Engine_channel_assignments_o c) Engine_sensor_O c	-		
Invariants ControlFaultDetect Initialization Initialization \ E Bool Initialization_Co Initialization.type : Initialization \ E BOOL Initialization_Co Checkout.type : Checkout \ E BOOL Initialization_Co Start_prep.type : Start_prep \ E BOOL Initialization_co Start.type : Start \ E BOOL Initialization_o Start.type : Doi_shutdown \ E BOOL Initialization_o Start.type : Doi_shutdown \ E BOOL Initialization_o Diagnosis.type : Diagnosis \ E BOOL Initialization_o ControlFaultDetect \ E BOOL Initialization Events Initialization Events Initialization Initialization \ I	 ControlFaultDetect ControlFaultDetect MASA-CASE-2011 Engine_Sensor_Co Engine_		-	
<pre>INVARIANTS INVARIANTS INVARIANTS INVARIANTS INVARIANTS INVARIANTS INVARIANTS INVARIANTS INVARIANTS INVARIANTS INITIALIZATION # O Proge Sequence_O O O Control Fault Detect. type : Initialization</pre>	<pre>INVARIANTS INVARIANTS INITIALIZATION INVARIANTS INITIALISATION INVARIANTS INV</pre>			
<pre>B ● Proof Obligations NASA-CASE-2011 B ● Engine_Initialization_CO B ● Engine_Sensor_CO B ● Engine_Startup_CO B ● Engine_Channel_assignments_CO B ● Engine_channel_assignments_CO B ● Engine_channel_assignments_CO B ● Engine_channel_assignments_O B ● Engine_initvake_status_O B ● Engine_shutdown B ● Engine_shutdown B ● Engine_shutdown B ● Engine_shutdown B ● Engine_startup_O B ● Engine_startup_O B ● Engine_startup_O B ● Engine_startup_O B ● Engine_startup_O B ● Engine_startup_O B ● O Proof Obligations B ● Engine_startup_O B ● O Proof Obligations B ● Engine_startup_O B ● O Proof Obligations B ● O Proof Obligation</pre>	 Proof Obligations Initialization.type : Initialization ∈ BOOL Wait.type : Wait ∈ BOOL Engine_Initialization_Co Engine_Sensor_Co Engine_Startup_Co Engine_dhannel_assignments_Co Engine_initialization_o Start.type : Start ∈ BOOL Start.type : OntrolFaultDetect ∈ BOOL Start.type : ControlFaultDetect ∈ BOOL Start.type : Start ∈ BOOL Start.type : ControlFaultDetect ∈ BOOL Start.type : Start ∈ BOOL Start.type : Start ∈ BOOL Start.type : Start ∈ BOOL Start.type : ControlFaultDetect ∈ BOOL Start.type : Start Start.type : Start Start.type : Start Start.type : Start Start.type : Start.type : ControlFaultDetect ∈ BOOL Start.type : Start Start.type : Start.type : St	-		
<pre>Waix_cAss=-2011 Waix_type : Wait € BOOL Checkout.type : Checkout € BOOL Start_prep.type : Start_prep € BOOL Start.type : Start_prep.type : Start_eBOOL Start.type : Start € BOOL Start.type : Start € BOOL Shutdown.type : Shutdown € BOOL Shutdown.type : Shutdown € BOOL Dot_Bion_sin_valve_status_0 Dot_Bion_e_shutdown Dot_Bi</pre>	<pre>WaSA-CASE-2011 Wait.type : Wait € BOOL Checkout.type : Checkout € BOOL Start.prep.type : Start € BOOL Start.type : Diagnosis € BOOL Diagnosis.type : Diagnosis € BOOL ControlFaultDetect.type : ControlFaultDetect € BOOL Engine_shutdown Start.type Start.type : ControlFaultDetect € BOOL Engine_startup_o Superg_Sequence_o Superg</pre>			
 G Engine_Initialization_Co G Engine_Sensor_Co G Engine_Startup_Co G Engine_Startup_Co G Engine_Linitalization_o G Engine_Initialization_o G Engine_Sensor_0 G Engine_Sensor_0 G Engine_Sensor_0 D Engine_Sensor_0 D Engine_Channel_assignments_o G Engine_ini_valve_status_o G Engine_shutdown G Start_Type G W wait_phase G W wait_phase G Engine_Shutdown.init : Post_shutdown = FALSE 	 a Genge_Initialization_Co b Engine_Sensor_Co c Engine_Startup_Co c Engine_channel_assignments_Co c Engine_Initialization_o c Engine_Initialization_o c Engine_channel_assignments_o c Engine_channel_assignments_o c Engine_channel_assignments_o c Engine_sensor_O c Engine_channel_assignments_o c Engine_sensor_O c Engine_sensor_O c Engine_sensor_O c Engine_sensor_O c Engine_initialization_o c Engine_channel_assignments_o c Engine_sensor_O c Diagnosis.type : Diagnosis ∈ BOOL c OntrolFaultDetect.type : ControlFaultDetect ∈ BOOL c Engine_startup_O c OntrolFaultDetect.type : ControlFaultDetect ∈ BOOL c Events c OntrolFaultDetect.type : ControlFaultDetect ∈ BOOL c Events c OntrolFaultDetect.type : ControlFaultDetect.type (TRUE}, {Unitialization} ∩ {TRUE}, {Wait} ∩ {TRUE}, {Check c StartUS c ordinary b BeGIN Post_shutdown.init : Post_shutdown = FALSE Maintsage.init : Maintsage = FALSE 	- 🥭 NASA-CASE-2011		
 G Engine_Sensor_Co Start_prep.type : Start_prep ∈ BOOL Start.type : Start ∈ BOOL Shutdown.type : Shutdown ∈ BOOL Shutdown.type : Post_shutdown ∈ BOOL Shutdown.type : Diagnosis ∈ BOOL Start.type : ControlFaultDetect ∈ BOOL Start.type : ControlFaultDetect ∈ BOOL Start.type : ControlFaultDetect ∈ BOOL Shutdown Shutdow	<pre>Start_prep.type : Start_prep Start_p</pre>	🗉 🕒 Engine_Initialization_Co		
 G Engine_Startup_Co S Engine_channel_assignments_Co B Engine_Initialization_o B Engine_Sensor_0 B Engine_channel_assignments_o B Engine_channel_assignments_o B Engine_channel_assignments_o B Engine_shutdown B E Engine_shutdown.init C Post_shutdown = FALSE 	 G Engine_Startup_Co G Engine_channel_assignments_Co G Engine_initialization_o G Engine_shutdown G Events G ControlFaultIstication = ({TRUE}, {Initialization} n {TRUE}, {Wait} n {TRUE}, {Check EVENTS G Checkout_phase_o G Checkout_phase G Checkout_phase<!--</td--><td>🗐 🕒 Engine_Sensor_Co</td><td>21</td><td></td>	🗐 🕒 Engine_Sensor_Co	21	
 G Engine_channel_assignments_Co Maintsage.type : Maintsage € BOOL Shutdown.type : Shutdown € BOOL Post_shutdown.type : Post_shutdown € BOOL Diagnosis.type : Diagnosis € BOOL ControlFaultDetect.type : ControlFaultDetect € BOOL Status ControlFaultDetect.type : ControlFaultDetect € BOOL ControlFaultDetect.type : ControlFaultDetect € BOOL ControlFaultDetect.type : ControlFaultDetect € BOOL Status ControlFaultDetect.type : ControlFaultDetect € BOOL Status ControlFaultDetect.type : ControlFaultDetect € BOOL ControlFaultDetect.type : ControlFaultDetect € BOOL ControlFaultDetect.type : ControlFaultDetect € BOOL Status ControlFaultDetect.type : ControlFaultDetect € BOOL ControlFaultDetect.type	 G Engine_channel_assignments_Co Maintsage.type : Maintsage & BOOL Shutdown.type : Shutdown & BOOL Post_shutdown & BOOL Diagnosis.type : Diagnosis & BOOL Diagnosis.type : Diagnosis & BOOL ControlFaultDetect.type : ControlFaultDetect & BOOL ControlFaultDetect.type : ControlFaultDetect & BOOL ControlFaultDetect.type : ControlFaultDetect & BOOL Engine_shutdown Diagnosis.type : ControlFaultDetect & BOOL ControlFaultDetect.type : ControlFaultDetect & BOOL Engine_shutdown Diagnosis.type : ControlFaultDetect & BOOL Engine_startup_o Maintsage_o Maintsage_o Mintsage.o O checkout_phase_o O checkout_phase O checkout_phase O st_shutdown.init : Post_shutdown = FALSE Maintsage.init : Maintsage = FALSE 	🗄 🧿 Engine_Startup_Co		
 Brigne_Initialization_o Brigne_Sensor_0 Brigne_Sensor_0 Brigne_channel_assignements_o Brigne_channel_assignements_o Brigne_ini_valve_status_o Brigne_shutdown Brigne_shutdow	 Bengine_Initialization_o Bengine_Initialization_o Bengine_Sensor_0 Bengine_channel_assignements_o Bengine_ini_valve_status_o Bengine_shutdown Bengine_shutdown Bengine_shutdown Bengine_startup_o Bengine_Sequence_o Checkout_phase_o Bengines Bengines Bengines Bengines Bengine_ini_valve_status_o Bengine_shutdown B	🗉 🧿 Engine channel assignments Co		
• ①, Engine_Sensor_0 Post_shutdown.type : Post_shutdown ∈ BOOL • ② Engine_channel_assignements_o Diagnosis.type : Diagnosis ∈ BOOL • ③, Engine_ini_valve_status_o Diagnosis.type : ControlFaultDetect ∈ BOOL • ④, Engine_shutdown Engine_thate = assignements_o • ④, Engine_shutdown Engine_thate = assignements_o • ⊕, Engine_startup_o Events • ⊕, Main_Stage_o INITIALISATION • @ otheckout_phase_o otheckout_phase • @ wait_phase BEGIN	 Dispine_Sensor_0 Post_shutdown.type : Post_shutdown ∈ BOOL Diagnosis.type : Diagnosis ∈ BOOL ControlFaultDetect.type : ControlFaultDetect ∈ BOOL Status ControlFaultDetect.type : ControlFaultDetect ∈ BOOL ControlFaultDetect.type : ControlFaultDetect ∈ BOOL ControlFaultDetect.type : ControlFaultDetect ∈ BOOL Status Status Ordinary Waintsage.init : Post_shutdown = FALSE Maintsage.init : Maintsage = FALSE 	Engine Initialization o		
<pre>Diagnosis.type : Diagnosis ∈ BOOL ControlFaultDetect.type : ControlFaultDetect ∈ BOOL Diagnosis.type : Diagnosis ∈ BOOL ControlFaultDetect.type : ControlFaultDetect ∈ BOOL Engine Phases and Modes.partitionedStates.1 : partition ({TRUE}, {Initialization} ∩ {TRUE}, {Wait} ∩ {TRUE}, {Check EVENTS INITIALISATION ≜ STATUS Ordinary BEGIN Post_shutdown = FALSE</pre>	<pre>Diagnosis type : Diagnosis € BOOL ControlFaultDetect.type : ControlFaultDetect € BOOL Diagnosis.type : ControlFaultDetect € BOOL ControlFaultDetect.type : ControlFaultDetect € BOOL Diagnosis.type : ControlFaultDetect € BOOL ControlFaultDetect.type : ControlFaultDetect € BOOL Engine Phases and Modes.partitionedStates.1 : ({TRUE}, {Initialization} ∩ {TRUE}, {Wait} ∩ {TRUE}, {Check UNMain_Stage_0 Wintialization} ∩ {TRUE}, {Wait} ∩ {TRUE}, {Check UNTS UNITIALISATION ≜ STATUS Ordinary BEGIN Post_shutdown.init : Post_shutdown = FALSE Maintsage = FALSE</pre>		Post shutdown.type : Post shutdown ∈ BOOL	
<pre>© Engine_ini_valve_status_o © @ @;Engine_shutdown @ @;Engine_shutdown @ @;Engine_startup_o @ @;Main_Stage_o @ @;Nain_Stage_o @ @;Nurge_Sequence_o @ @;Purge_Sequence_o @ @;Purge_Sequence_o @ @;Purge_Sequence_o @ @;Nurge_Sequence_o @ @;Nurge_Sequence_o @ @;Purge_Sequence_o @ @;Nurge_Sequence_o @ @;Purge_Sequence_o @ @;</pre>	<pre>© Engine_ini_valve_status_o @ 0;Engine_shutdown @ 0;Engine_status_o @ 0;Main_Stage_o @ 0;Nurge_Sequence_o @ 0;Purge_Sequence_o @ 0; engine_status_o @</pre>			
<pre>Begine shutdown Begine shutdown Begine Phases and Modes.partitionedStates.1 Begine Phases and Begi</pre>	<pre>e @,Engine_shutdown @ @,Engine_startup_0 @ @,Main_Stage_0 @ @,Purge_Sequence_0 @ @ neckout_phase_0 @ @ wait_phase BEGIN Post_shutdown.init : Post_shutdown = FALSE Maintsage.init : Maintsage = FALSE</pre>			
<pre> @ Wain_starge_o @ Wain_stage_o @ Wain_stage_o @ Wain_stage_o @ Wain_stage_o @ Operation of the startus @ Operation of the startus @ Wain_phase @ Wain_phase @ Wain_phase @ Wain_phase @ Wain_starge_o @ Wain_st</pre>	<pre>@ Wain_Stage_0 @ Wain_Stage_0 @ Wain_Stage_0 @ Wain_phase</pre>			
● ①,Main_Stage_0 INITIALISATION ≜ ● ②,Purge_Sequence_0 STATUS ● ② checkout_phase_0 ordinary ● ③ wait_phase BEGIN Post_shutdown.init Post_shutdown = FALSE	 W,Main_Stage_o W,Purge_Sequence_o W checkout_phase_o W wait_phase W wait_phase BEGIN Post_shutdown.init : Post_shutdown = FALSE Maintsage : FALSE 		({IRUE}, {INITIALIZATION} n {IRUE}, {WAIT} n {IRUE}, {CHECK	
■ ③,Purge_Sequence_o STATUS ■ ④ checkout_phase_o ordinary ■ ④ wait_phase BEGIN Post_shutdown.init Post_shutdown = FALSE	<pre> Work of the constraint of the constra</pre>		EVENTS	
	■ ③ checkout_phase_o ■ ③ wait_phase BEGIN Post_shutdown.init : Post_shutdown = FALSE Maintsage = FALSE			
Wait_phase BEGIN Post_shutdown.init : Post_shutdown = FALSE	W wait_phase BEGIN Post_shutdown.init : Post_shutdown = FALSE Maintsage.init : Maintsage = FALSE ✓			
Post_shutdown.init : Post_shutdown = FALSE	Post_shutdown.init : Post_shutdown = FALSE Maintsage.init : Maintsage = FALSE			
	Maintsage.init : Maintsage ≔ FALSE	🗄 🥥 wait_phase		
				*

Event B Editor

_

File Edit Navigate Search Project Run Ren	ne_ini_valve_status_o.bum - Rodin Platform
	2 ♀ 1 √ B 大 † 1 約 - 印 · 中 ♀ · 1 ✔ げ ✔ ✔
🖞 Event-B Explorer 🕱 📃 🗖	
🖹 🏙 🖻 🔒 🖻 🎽	z 🚸 internal 📓 ggfv_open : not extended 🛃 ordinary 📓 //
1 🕐	
🖻 🗁 Army-Case-Study 🤷	REFINES
AV_mode_control	
🗉 😼 State+Mode_Control_Co	ANY
🕀 🥝 Power_Up_Mode_0	Alti
🗄 🚳 State+Mode_Control_0	
🖻 😂 Engine Phases and Modes	◊
🖻 🙆 Engine Phases and Modes	
💼 📼 Statemachine Engine Phases and Mc	● 仓 令
🗉 💿 Variables	VHERE
🗉 - 😽 Invariants	\$500000 000-000-000
	• ÷ ÷
🛓 🔞 Proof Obligations	□ 🖕 grd1 : ggfv = N theorem //
E NASA-CASE-2011	
Engine_Initialization_Co	Image: grd2 : ggfv ≥ 95 theorem //
Gengine_Sensor_Co	
	● ŷ \$
Engine_channel_assignments_co Engine_Initialization_o	∀WITH
Engine_Initialization_0	● {}
Engine_sensor_o Engine_channel_assignements_o	
Engine_ini_valve_status_o	⇒ THEN
	● ☆ ₽
	A
🗄 🤫 Proof Obligations	 · ·
Engine_shutdown	END
🖶 💿 Variables	
🗊 🔶 Invariants	▽ * 🗢 internal 👱 ggfv_close : not extended ⊻ ordinary 🕑 //

Event B prettyprint

_

le Edit Navigate Search Project Run Renai	me Event-B ProB Window H	lelp			
📬 • 🔛 🕼 🖆 🛯 🚳 🛛 🌯 • 🗄 🛷 • 🗄 🤣 🧐	◇ ✓ B 大 † 約 - 答] - 😉 🗇 • 🤤 - 🕴 🖍 📌 📌 🕐		🖺 Ever	nt-B
Event-B Explorer 🛛 🗖 🗖	🕲 Engine Phases and Modes 👘	🚯 Engine Phases and Modes.Engine Phases and	😡 Engine_shutdown	🕼 Engine_ini_valve_status_o 🛛	
Army-Case-Study Army-Case-Study Av_mode_control A	act4 mov_sta act5 otbv_sta act6 bv_sta act7 ftplos act8 asiov_s act9 main_lo act10 : act11 gg_lo: act12 gg_fu act14 otpt_s act14 : act14 : act14 : act14 : ggfv_open = STATUS ordinary ANY ggfv WHERE grd1 : grd1 : ggfv ≤ THEN : :	<pre>atus = CLOSE tatus = CLOSE tatus = CLOSE _status = SEAL status = CLOSE ox_sys_prg_status = CLOSE fuel_sys_prg_status = PRG_OFF x_prg_status = PRG_OFF el_prg_status = PRG_OFF seal_prg_status = PRG_OFF seal_prg_status = PRG_OFF</pre>			
•••••••••••••••••••••••••••••••••	ANY ggfv WHERE grd1 : ggfv∈	N			
• Variables • Invariants • Fivents • • Proof Obligations	grd2 : ggfv ≤ THEN act1 : ggfv_s ⁻ FND	5 tatus ≔ CLOSE			

NASA/Army Experience -Learning curve

- Unlike other tools, Formal Methods requires serious study
 - Formal Methods Language (B, Z...)
 - Formal Methods Development platform (Rodin, Event B...UML, UML-B...)
 - Mathematical symbols, rules, logic...
- Training on Formal Methods is necessary
 - Engineers with better understanding of the project
 - Eliminate errors
 - Reduce Design complications and time
 - Encourage Engineers with better mathematics and science
- Easy is not the best solution for NASA and Army
 - Easy tools are easy to sell, but not able to solve our real problems

Recommendations

Formal Methods =

• High cost tool

- Powerful, but not affordable to most of the organizations
- Army used SCADE and Simulink with Design Verifier as a modeling tool.

Open Source

- No cost, but high learning curve and lack of support
- Training program will significantly reduce the learning curve, this can be used for large community.

Recommendations:

- Project requiring immediate results may need to use high cost tools.
- Continue monitoring open source tools (e.g. Integrated Rodin Event B and UML B) which will likely become more advanced in the future.



Results

- Formal methods can have significant cost savings.
- Defects can be found earlier when easier and cheaper to fix (cf. Army experience).
- While FMs are difficult to use and learn, a typical engineer can use them successfully when given appropriate support.
- Numerous tools are available. Choice is determined by:
 - Cost
 - Support
 - Deadlines
- Free (or cheap) is not necessarily best.

Future Plans

- Continue monitoring new and emerging Formal Methods techniques for practical usefulness and applicability to critical NASA/Army systems and software development activities.
- Complete Case study for both NASA/Army subsystems.
- Army is utilizing Formal Methods techniques for current programs.
- Complete Guidebook with road maps for future users.
- Pursue training opportunities with NASA STEP training office.
- Continue to emphasize awareness in Formal Methods and related training program



Contact Information

- Caroline K. Wang
 - <u>Caroline.k.wang@nasa.gov</u>
- Dr. Mike Hinchey
 - mike.hinchey@lero.ie
- Josh McNeil
 - Josh.McNeil@us.army.mil

