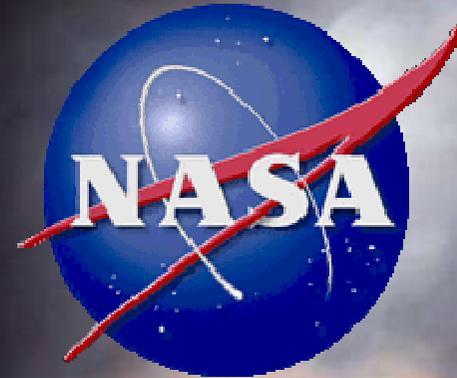


# *Formal Methods for System/Software Engineering: NASA & Army Experiences*



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

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



# Formal Methods

## - Dr. Mike Hinchey

*Formal Methods*

### 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

## *Formal Methods*

General Problem	Approach
<ul style="list-style-type: none"><li>• <b>System/Hardware/Software complexity</b></li></ul>	<ul style="list-style-type: none"><li>• <b>Provide accurate and appropriate specifications of required system behavior using Formal Methods</b></li></ul>
<ul style="list-style-type: none"><li>• <b>Inadequate requirements specifications / misinterpretation of natural language</b></li><li>• <b>Significant number of problems introduced due to vague requirements</b></li></ul>	<ul style="list-style-type: none"><li>• <b>Develop requirement specification as Formal Specification (using formal semantics) to eliminate misinterpretation of vague and incomplete natural language requirements</b></li></ul>
<ul style="list-style-type: none"><li>• <b>Significant number of safety and reliability problems are traced to incorrect performance or behavior specifications, or incorrect interfaces</b></li></ul>	<ul style="list-style-type: none"><li>• <b>Use Formal methods to prove safety properties derived from safety analyses</b></li><li>• <b>Use Formal Methods and deductive apparatus to prove correctness of system behavior and interfaces</b></li></ul>

# Problem/Approach

## *Formal Methods*

Specific Problem	Approach
<ul style="list-style-type: none"><li>• <b>Formal Methods Learning Process</b> <b>Difficult for new users</b></li></ul>	<ul style="list-style-type: none"><li>• <b>Develop specific project related case studies and provide examples for potential users</b></li></ul>
<ul style="list-style-type: none"><li>• <b>Select development tools</b> <b>No time to learn all the tools</b> <b>Inadequate resource</b></li></ul>	<ul style="list-style-type: none"><li>• <b>Based on the project size and resources available, select appropriate Formal Methods development techniques and tools</b></li></ul>
<ul style="list-style-type: none"><li>• <b>Budget and Schedule constraints</b></li></ul>	<ul style="list-style-type: none"><li>• <b>Support program development and in parallel prove potential savings</b></li></ul>
<ul style="list-style-type: none"><li>• <b>Differences in priorities between Research and Production environments</b></li></ul>	<ul style="list-style-type: none"><li>• <b>Many researchers focus on development of new techniques and tools</b></li><li>• <b>Production or development programs are concerned with delivery of a product</b></li><li>• <b>Need to build bridges between the research and production environments</b></li></ul>

# Challenges

---

---

## *Formal Methods*

---

---

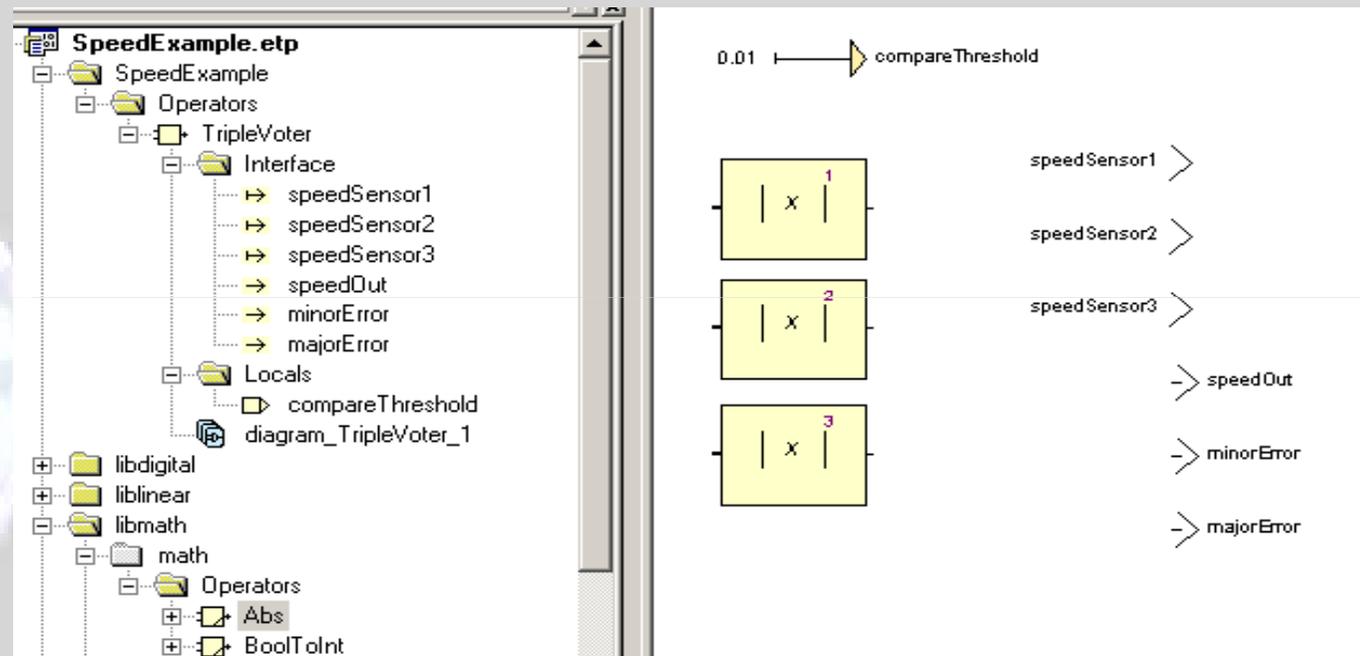
- High cost of some commercial development tools.
- Open source free tools do not have adequate training material and support.
- Formal Methods tools require extensive learning process.
- Die-hard Systems and Hardware Engineers are not convinced of the importance of software.



# Developing TripleVoter Model

*Formal Methods*

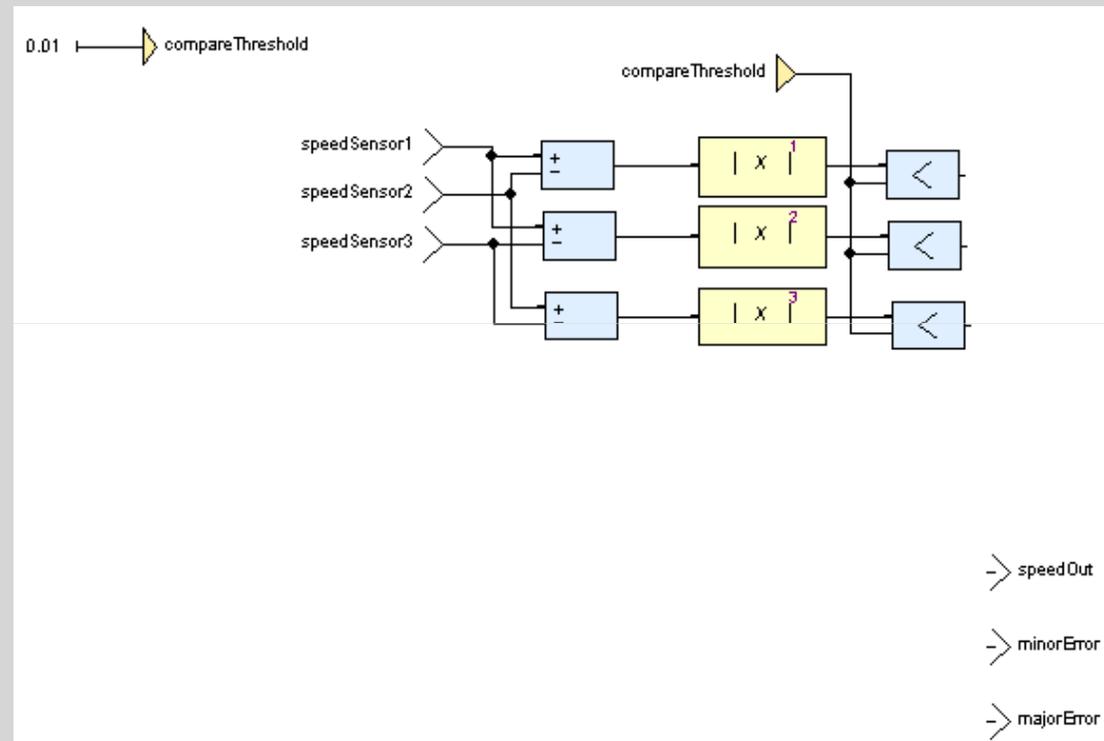
- Double-click the TripleVoter operator to begin modeling.
- Select all variables (speedSensor1, speedSensor2, speedSensor3, speedOut, minorError, majorError, and compareThreshold). Drag them onto the diagram.
- Select the compareThreshold local variable, modify it through Properties → Use, and change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



# Implementing Model Logic

## Formal Methods

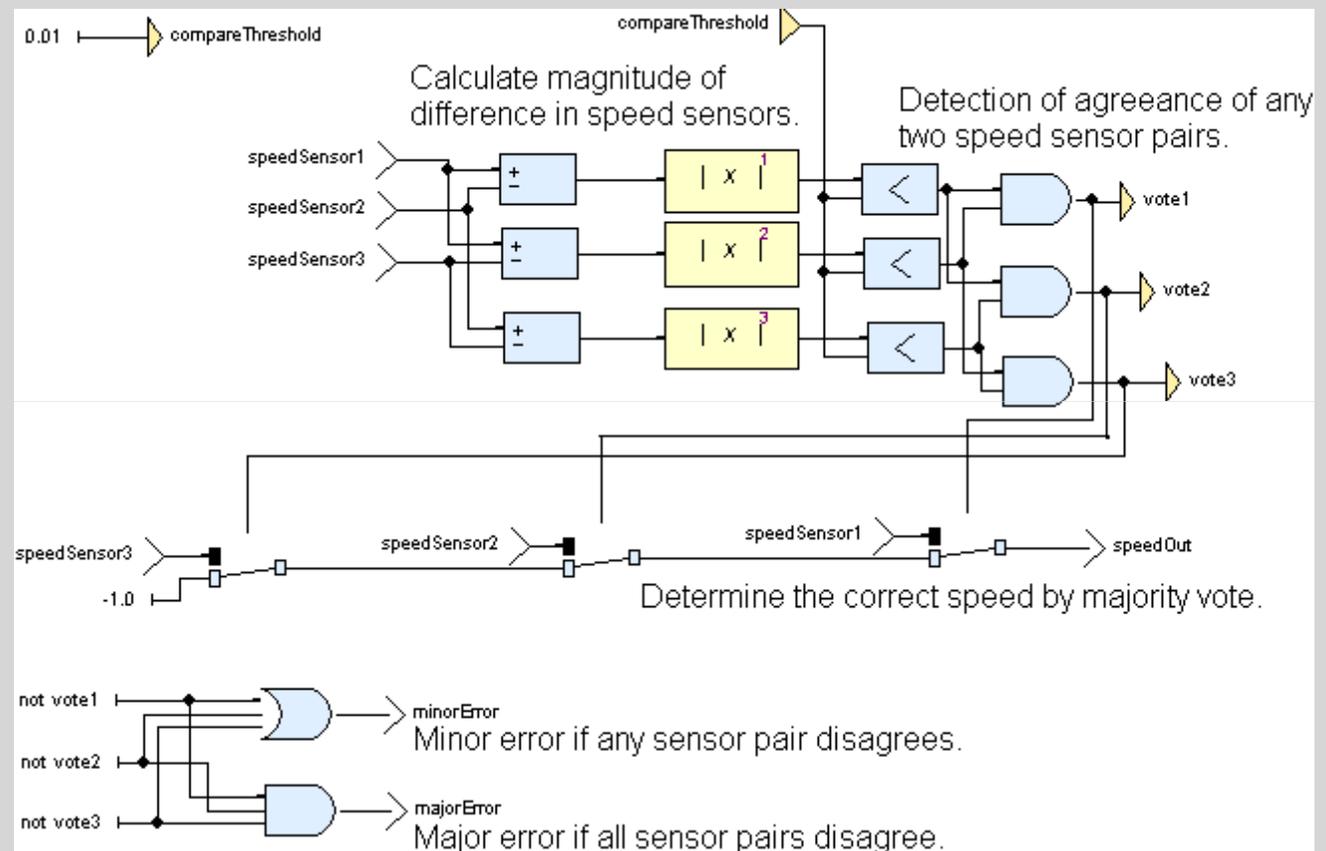
- Connect speedSensorX to the “+” input and speedSensorX to the “-” input of the New Minus operator.
- Connect speedSensor1, speedSensor2, and speedSensor3 to the first input of each New Minus operator.
- Connect all outputs of the New Minus operators to the inputs of the Abs operators.



# Completing The Model Logic

*Formal Methods*

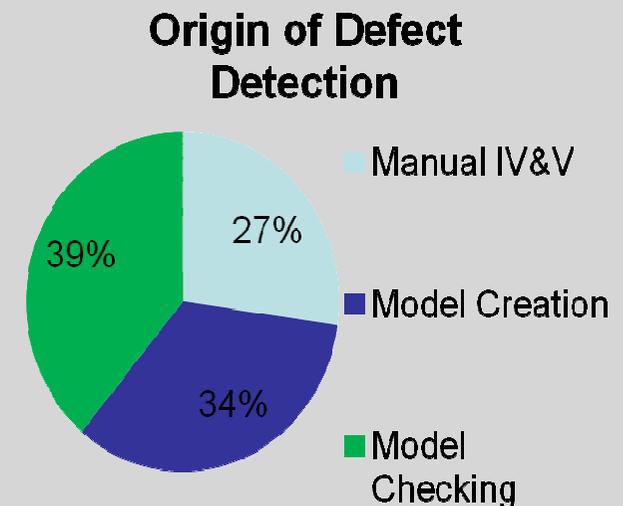
- Complete other logic components by drag and drop or connections.
- Add new If..Then..Else operators ( ) to the diagram.
- Add comments to model for readability
- **Design Verification** – Design Verifier can be used to develop properties that can be proven by formal methods.



# Army's experience and Return on Investment

Formal Methods

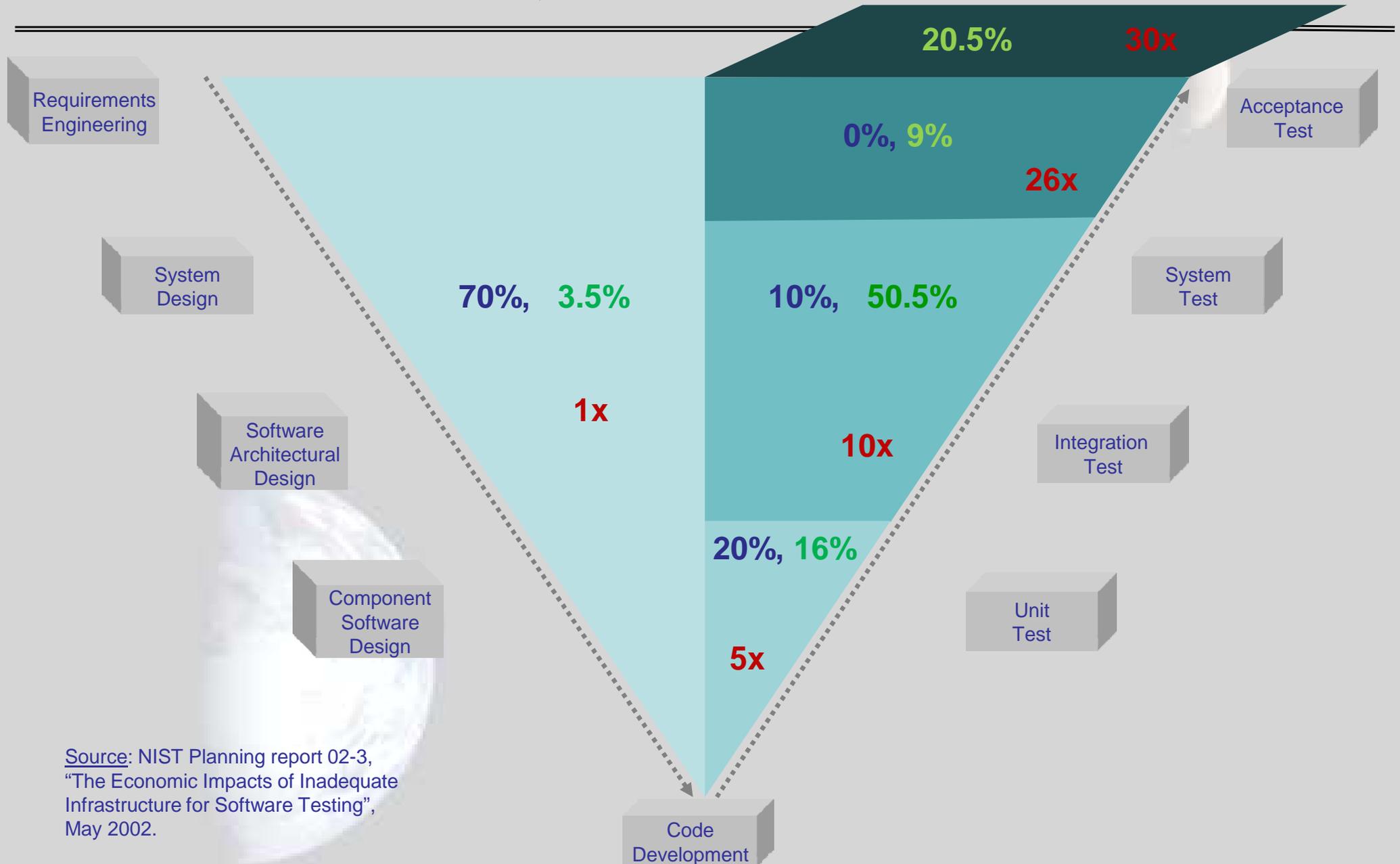
- Formal methods approach using SCADE method found 144 defects **their** traditional IV&V would miss (73% of all defects found)
- Estimating it would cost approximately 3500 man hours at \$100 per man hour to fix the 144 defects later in the lifecycle
- Early defect removal savings is \$350K
- The cost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

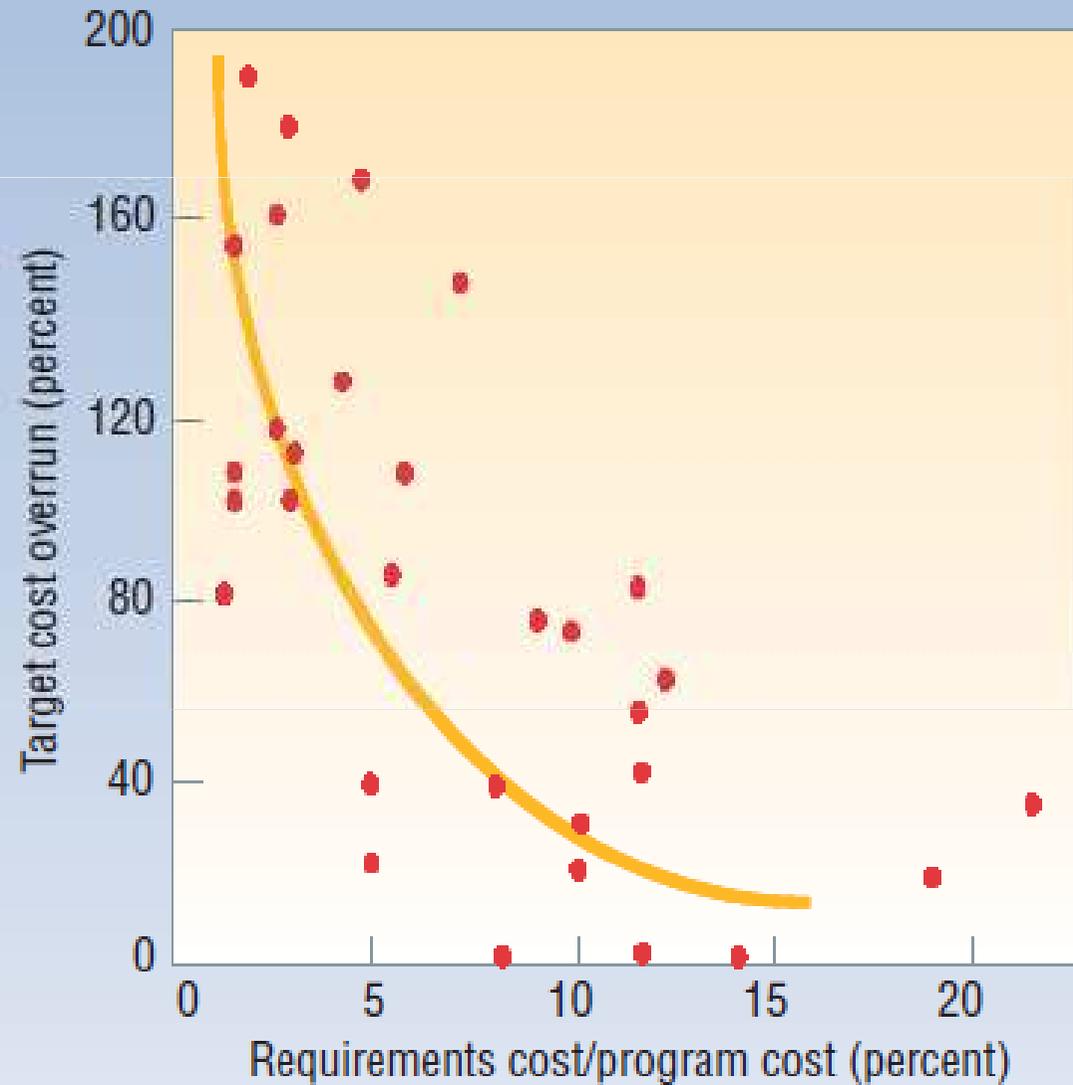
Where are faults introduced, **discovered** and **cost for removal**



Source: NIST Planning report 02-3, "The Economic Impacts of Inadequate Infrastructure for Software Testing", May 2002.

# NASA Cost overruns

*Formal Methods*



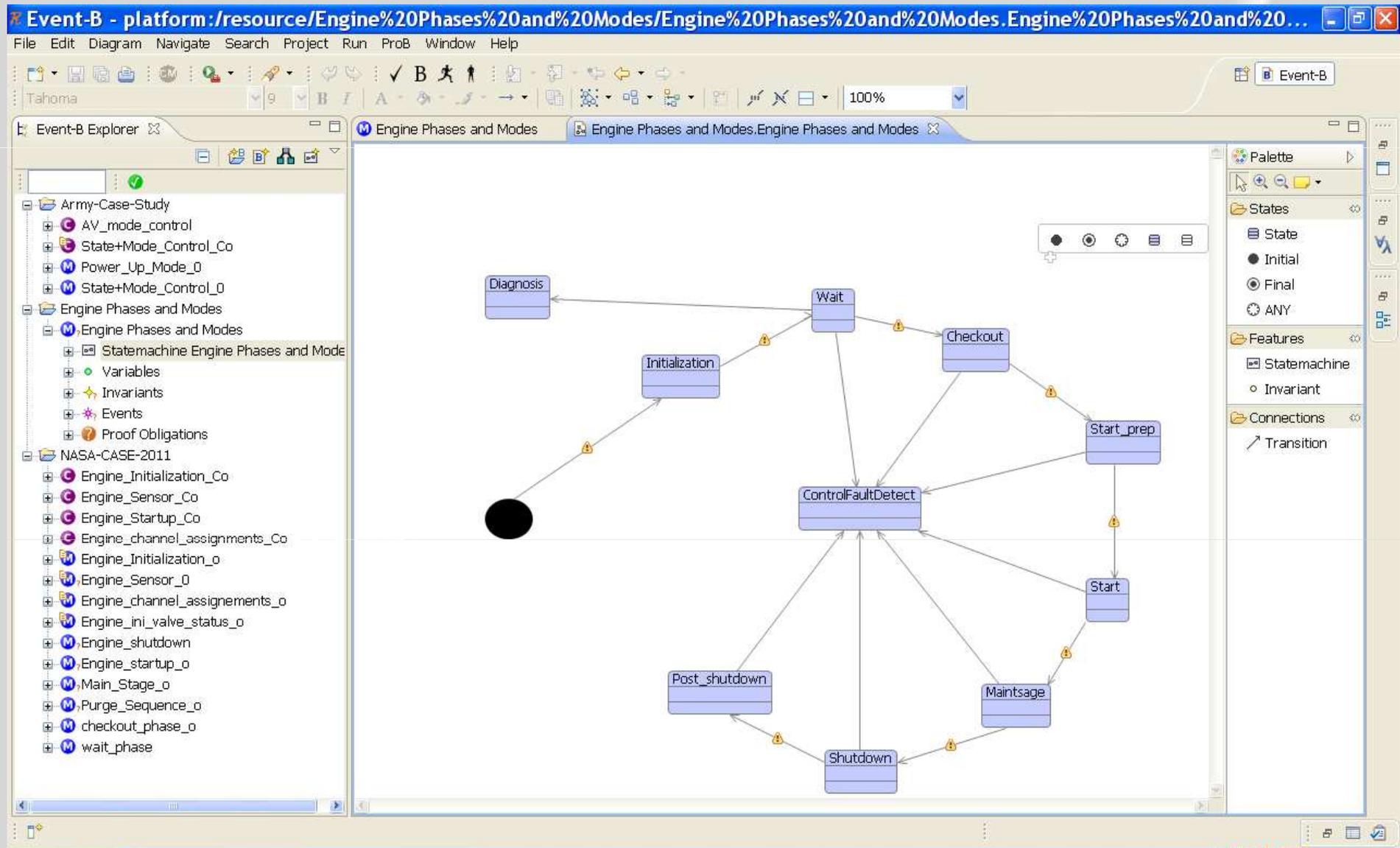
# NASA MSFC Experience in this study

*Formal Methods*

- 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

*Formal Methods*



# Auto Translation to Event B

*Formal Methods*

The screenshot displays the Rodin Platform Event-B IDE. The main window shows the source code for a machine named "Engine Phases and Modes". The code is structured as follows:

```
MACHINE
Engine Phases and Modes
VARIABLES
Initialization
Wait
Checkout
Start_prep
Start
Maintsage
Shutdown
Post_shutdown
Diagnosis
ControlFaultDetect
INVARIANTS
Initialization.type : Initialization ∈ BOOL
Wait.type : Wait ∈ BOOL
Checkout.type : Checkout ∈ BOOL
Start_prep.type : Start_prep ∈ BOOL
Start.type : Start ∈ BOOL
Maintsage.type : Maintsage ∈ BOOL
Shutdown.type : Shutdown ∈ BOOL
Post_shutdown.type : Post_shutdown ∈ BOOL
Diagnosis.type : Diagnosis ∈ BOOL
ControlFaultDetect.type : ControlFaultDetect ∈ BOOL
Engine Phases and Modes.partitionedStates.1 : partition
({TRUE}, {Initialization} n {TRUE}, {Wait} n {TRUE}, {Check
EVENTS
INITIALISATION
STATUS
ordinary
BEGIN
Post_shutdown.init : Post_shutdown = FALSE
Maintsage.init : Maintsage = FALSE
```

The left sidebar shows a project tree with folders for "Army-Case-Study", "Engine Phases and Modes", and "NASA-CASE-2011". The "Engine Phases and Modes" folder is expanded, showing sub-elements like "State+Mode\_Control", "Variables", "Invariants", "Events", and "Proof Obligations".

# Event B Editor

*Formal Methods*

The screenshot displays the Event-B Editor interface. The title bar reads "Event-B - NASA-CASE-2011/Engine\_ini\_valve\_status\_o.bum - Rodin Platform". The menu bar includes File, Edit, Navigate, Search, Project, Run, Rename, Event-B, ProB, Window, and Help. The toolbar contains various icons for file operations and editing.

The left pane, "Event-B Explorer", shows a project tree with the following structure:

- Army-Case-Study
  - AV\_mode\_control
  - State+Mode\_Control\_Co
  - Power\_Up\_Mode\_0
  - State+Mode\_Control\_0
- Engine Phases and Modes
  - Engine Phases and Modes
    - Statemachine Engine Phases and Mc
    - Variables
    - Invariants
    - Events
    - Proof Obligations
  - NASA-CASE-2011
    - Engine\_Initialization\_Co
    - Engine\_Sensor\_Co
    - Engine\_Startup\_Co
    - Engine\_channel\_assignments\_Co
    - Engine\_Initialization\_o
    - Engine\_Sensor\_0
    - Engine\_channel\_assignments\_o
    - Engine\_ini\_valve\_status\_o
      - Variables
      - Invariants
      - Events
      - Proof Obligations
    - Engine\_shutdown
      - Variables
      - Invariants
      - Events
      - Proof Obligations

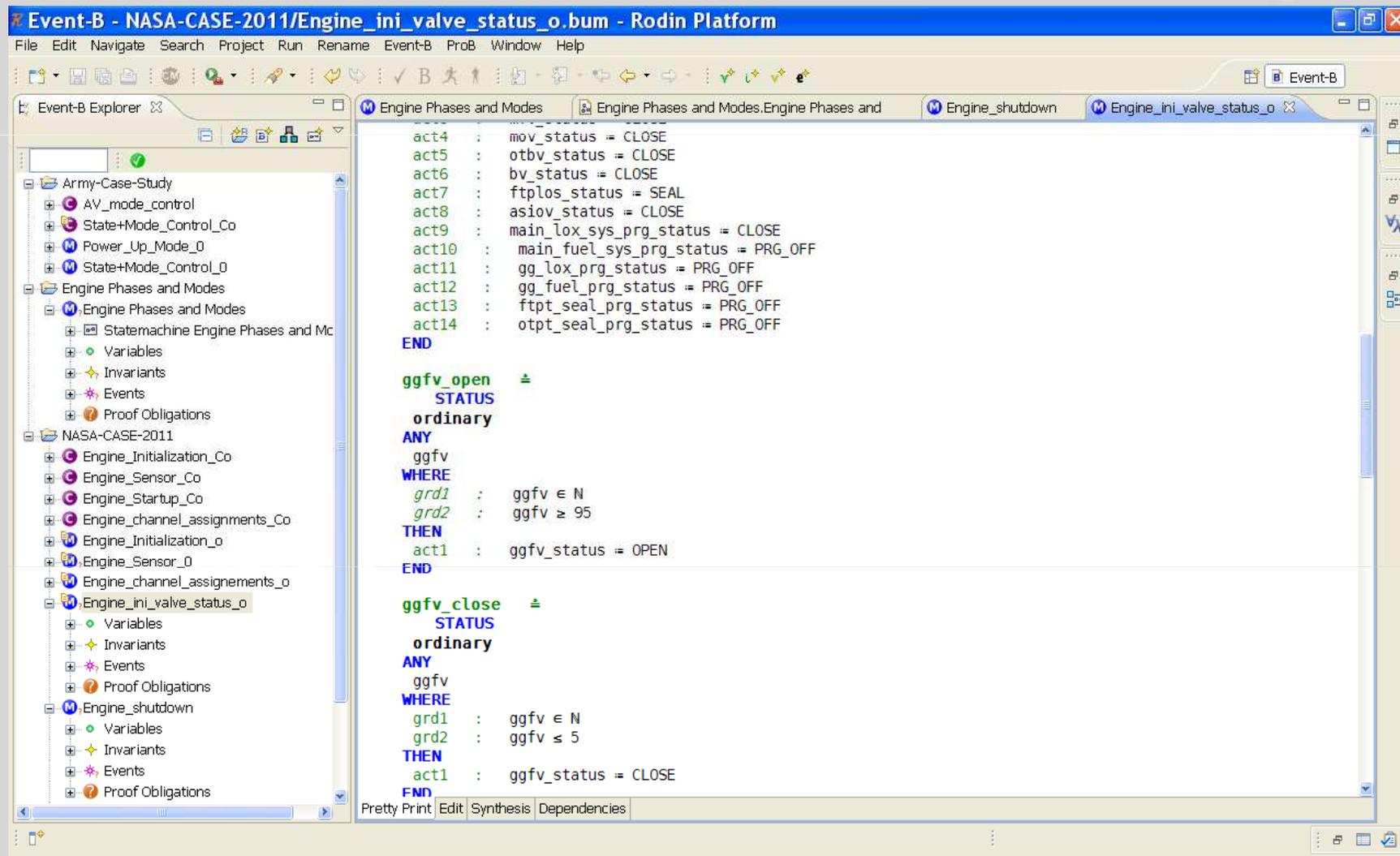
The main editor shows the formal specification for the event `ggfv_open`. The specification is as follows:

```
internal ggfv_open : not extended ordinary //  
  
REFINES  
  
ANY  
  
ggfv //  
  
WHERE  
  
grd1 : ggfv ∈ ℕ theorem //  
grd2 : ggfv ≥ 95 theorem //  
  
WITH  
  
THEN  
  
act1 : ggfv_status = OPEN //  
  
END  
  
internal ggfv_close : not extended ordinary //
```

At the bottom of the editor, there are buttons for "Pretty Print", "Edit", "Synthesis", and "Dependencies".

# Event B prettyprint

*Formal Methods*



```
act4 : mov_status = CLOSE
act5 : otbv_status = CLOSE
act6 : bv_status = CLOSE
act7 : ftplos_status = SEAL
act8 : asiov_status = CLOSE
act9 : main_lox_sys_prg_status = CLOSE
act10 : main_fuel_sys_prg_status = PRG_OFF
act11 : gg_lox_prg_status = PRG_OFF
act12 : gg_fuel_prg_status = PRG_OFF
act13 : ftpt_seal_prg_status = PRG_OFF
act14 : optp_seal_prg_status = PRG_OFF
END

ggfv_open ≐
STATUS
ordinary
ANY
ggfv
WHERE
  grd1 : ggfv ∈ N
  grd2 : ggfv ≥ 95
THEN
  act1 : ggfv_status = OPEN
END

ggfv_close ≐
STATUS
ordinary
ANY
ggfv
WHERE
  grd1 : ggfv ∈ N
  grd2 : ggfv ≤ 5
THEN
  act1 : ggfv_status = CLOSE
END
```

# NASA/Army Experience

## -Learning curve

---

### *Formal Methods*

---

- 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

---

---

- 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

---

## *Formal Methods*

---

- 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

*Formal Methods*

- Caroline K. Wang
  - [Caroline.k.wang@nasa.gov](mailto:Caroline.k.wang@nasa.gov)
- Dr. Mike Hinchey
  - [mike.hinchey@lero.ie](mailto:mike.hinchey@lero.ie)
- Josh McNeil
  - [Josh.McNeil@us.army.mil](mailto:Josh.McNeil@us.army.mil)

