# *NIST: Request for Information Relating to Cybersecurity Frameworks*

## *A Foundation Response*

## *by*

Gordon E Morrison

Gordon.Morrison@VSMerlot.com
(816-835-3071)
2951 Marina Bay Dr. 130-250
League City, TX 77573

# Cyber Secure Framework

- Can’t exist without a proper foundation
- Current Foundation
  - Spaghetti code
    - “if – then – else”
    - “case – switch”
    - Not organized
    - Not structured
  - Artistic
    - Approach will not be as secure as it could be

# Foundation Definition

- Foundation – the architecture that software uses as the core structure for organizing human written or machine generated code. Currently writing or generating code is an artistic endeavor without an engineering structure.

  In practice a software engineered foundation does not exist.

# Framework Definition

- Framework – the collection of software designed to provide services built on a nonexistent foundation. Frameworks are often combined into application specific libraries or collections.

# Software Engineering Practices

- Foundation
  - Lacking a foundation architecture
  - Lacking an engineering discipline
    - Coding is artistic
  - Requirements – Specifications – Models
    - All depend on the foundation
    - Implementations drift away form original design documentation
    - Synchronization requires manual effort
- Frameworks are built sans foundation

# From CMU-SEI

- www.sei.cmu.edu/solutions/softwaredev/

- "The quality of a system is influenced by the quality of the process used to acquire, develop, and maintain it, the analysis and forethought that goes into an architecture…"

# CMU –SEI (cont'd)

- "Using proven methods for progress and product quality, software success is predictable and achievable, and failure is avoidable."

# CMU – SEI (cont'd)

- "Once coding starts, teams trained in mature **software engineering** processes can remove defects early, when defect removal is 10 to 100 times less costly than it is during test."

# Requirements

- SEI alludes to failures due to lack of requirements.
- SEI requirements don’t correlate to the application over time.
- SEI requirements are documents that fail to stay in sync over time.
- SEI approach not as good as it could be. A good idea poorly implemented.

# No CMU-SEI Foundation Definition

- Big money is in process consulting
  - CMU-SEI sells what it knows
  - Doesn't understand lacking foundation
- No solution to spaghetti code
  - Process Management
    - Often referred to as "Software Engineering"
- Without a good foundation success is difficult at best.
- CMU-SEI – room for improvement

# SEI Template

- SEI uses a template to collect information
  - It's a fill in form approach
  - State information
  - Action information
  - Provides documentation
- Template becomes throwaway
- The template will not stay in sync
- Good idea – poorly implemented

# Herding Cats is the Standard

- Programmers want to be engineers
  - An engineering foundation is missing
- “if-then-else” and “case-switch” statements:
  - Are the cause of spaghetti code
  - Create logic that is overly complex
  - No support for temporal control flow
    - Required for correlation

# Temporal Software Engineering

- Similar to CMU-SEI template based logic
- Integrated into the application
- Uses Vector State Machine
  - Correlates to
    - Requirements
    - Specification
- Model Driven Architecture
  - Maps to IDEF ++ process
    - Improved IDEF0 & IDEF1
- Provides a solid organized foundation

# CMU-SEI Example Template

- SEI-TSP/PSP
  - Rule
  - State
  - Action
- Mixed Modes
- Does **not** correlate with solution
- Software Engineers DON'T sync this document!

| | | | |
|---|---|---|---|
| Student | J. Developer | Date | 10/27 |
| Program | LogIn | Program # | |
| Instructor | Humphrey | Language | C++ |

| State Name | Description | |
|---|---|---|
| Start | Start condition for system | |
| CheckD | The state of the system after a user ID is requested | |
| CheckPW | The state of the system after a user password is requested | |
| End | The final state: LogIn either logs in or cuts off the user. | |
| **Function/Parameter** | **Description** | |
| ID | User identification: ID is Valid or !Valid | |
| PW | User password: PW is Valid or !Valid | |
| n | Integer count of ID and password errors | |
| nMax | Maximum value of ID and password errors: n >= nMax is rejected. | |
| Fail | Error count or timeout error indicator: Fail = true is failure, Fail = false is ok. | |
| **States/Next States** | **Transition Condition** | **Action** |
| Start | | |
| Start | No transitions from Start to Start | |
| CheckID | True | Get ID, n := 0; ID and PW !Valid |
| CheckPW | No transitions from Start to CheckPW | |
| End | No transitions from Start to End | |
| CheckID | | |
| Start | No transitions from CheckID to Start | |
| CheckID | No transitions from CheckID to CheckID | |
| CheckPW | Valid ID | Get password |
| CheckPW | !Valid ID | Get password |
| End | Timeout | Fail := true |
| CheckPW | | |
| Start | No transitions from CheckPW to Start | |
| CheckID | (!Valid PW ∨ !Valid ID) ∧ n < nMax ∧ !Timeout | Get ID, n := n + 1 |
| CheckPW | No transitions from CheckPW to CheckPW | |
| End | Valid PW ∧ Valid ID | Fail := false, login user |
| End | (n >= nMax ∨ Timeout) ∧ (!Valid PW ∨ !Valid ID) | Fail := true, cut off user |
| End | | |
| | No transitions from End to any state | |

# CMU-SEI - Stopwatch Example

- Work is not part of implementation
- Must be **converted** to if/else or switch-case logic

| State Name | Description | |
|---|---|---|
| Zero | Start condition for system | |
| Running | Stopwatch running and displaying | |
| On-hold | Stopwatch running with display on hold | |
| Stopped | Stopwatch stopped | |
| **States/Next States** | **Transition Condition** | **Action** |
| Zero | | |
| Zero | reset $\vee$ hold | Stop clock, reset clock, clear display |
| Running | start/stop | Start clock, display clock |
| Running | | |
| Zero | reset | Stop clock, reset clock, clear display |
| On-hold | hold | Hold display |
| Stopped | start/stop | Stop clock, hold display |
| On-hold | | |
| Zero | reset | Stop clock, reset clock, clear display |
| Running | hold | Start clock, display clock |
| Stopped | start/stop | Stop clock, hold display |
| Stopped | | |
| Zero | reset | Stop clock, reset clock, clear display |
| Running | start/stop | Start clock, display clock |
| Stopped | hold | Stop clock, hold display |

# Proposed Foundation

- COSA – based on US Patent 6,345,387
  - Free of License – Free of License!
  - Template based executable logic table
  - Table based Vector State Machine (VSM)
- Temporal Engineering – the use of COSA, correlating all aspects of the software development life cycle.
- Temporal Engineering – improves the CMU-SEI management paradigm
  - Everything stays in sync!!!
- This is a good idea – good implementation

# SEI vs. COSA

- Work going into the SEI template is not directly used, i.e. it's wasted.
- Work put into a COSA table is used
  - The table is a logic template
  - The table is executed with COSA Engine
  - Testable with populated member functions or stubs
  - Does one thing and does it well
  - Includes trace debugging

# No Foundation vs. Foundation

- No Foundation Today
  - Bucket-of-Bolts
    - Spaghetti code
- Foundation – COSA
  - An Engineering Discipline
  - Not an artistic approach
  - Not just writing code
  - Organized
  - Standardized
- COSA
  - No License required
  - Patent definition open disclosure
  - Book available on Amazon.com

# What's Missing

- "**Software engineering**" mentioned on slide 8 does not refer to a foundation architecture.
- It refers to the process in which the code is developed.
- The fundamental "if-then-else" structure i.e. "spaghetti" code that SEI teaches.
- Compare the next two slides…

# Traditional Software

**What Is Wanted**

- Engineering Discipline
- Uniformity
- Consistency
- Preemptablity
- Single Point Logic Testing
- Trace - True
  - True Behavior Logic
  - True Logic Trace
  - True Logic Temporal Path
- Trace - False
  - False Behavior Logic
  - False Logic Trace
  - False Logic Temporal Path
- Well Defined
  - Rules
  - Specification
  - Analysis
- Orthogonal
  - Logic
  - Data

**What is Delivered**

- Authors that are like Herding Cats
- The style of the author
- Inconsistent development styles
- Control and Preemptablity an after thought
- Multiple if-then-else logic dispersed everywhere
- Spaghetti Logic
- Trace - True
  - Numerously inserted trace logic
- NONE
- Trace – False
  - Numerously inserted trace logic
- Spaghetti Logic
- NONE
- NONE
- Rarely Well Defined
  - No Rules
  - Independent Specification
  - Inconsistent Analysis
- Never Orthogonal
  - Spaghetti Logic
  - Spaghetti Data

# COSA Engineering

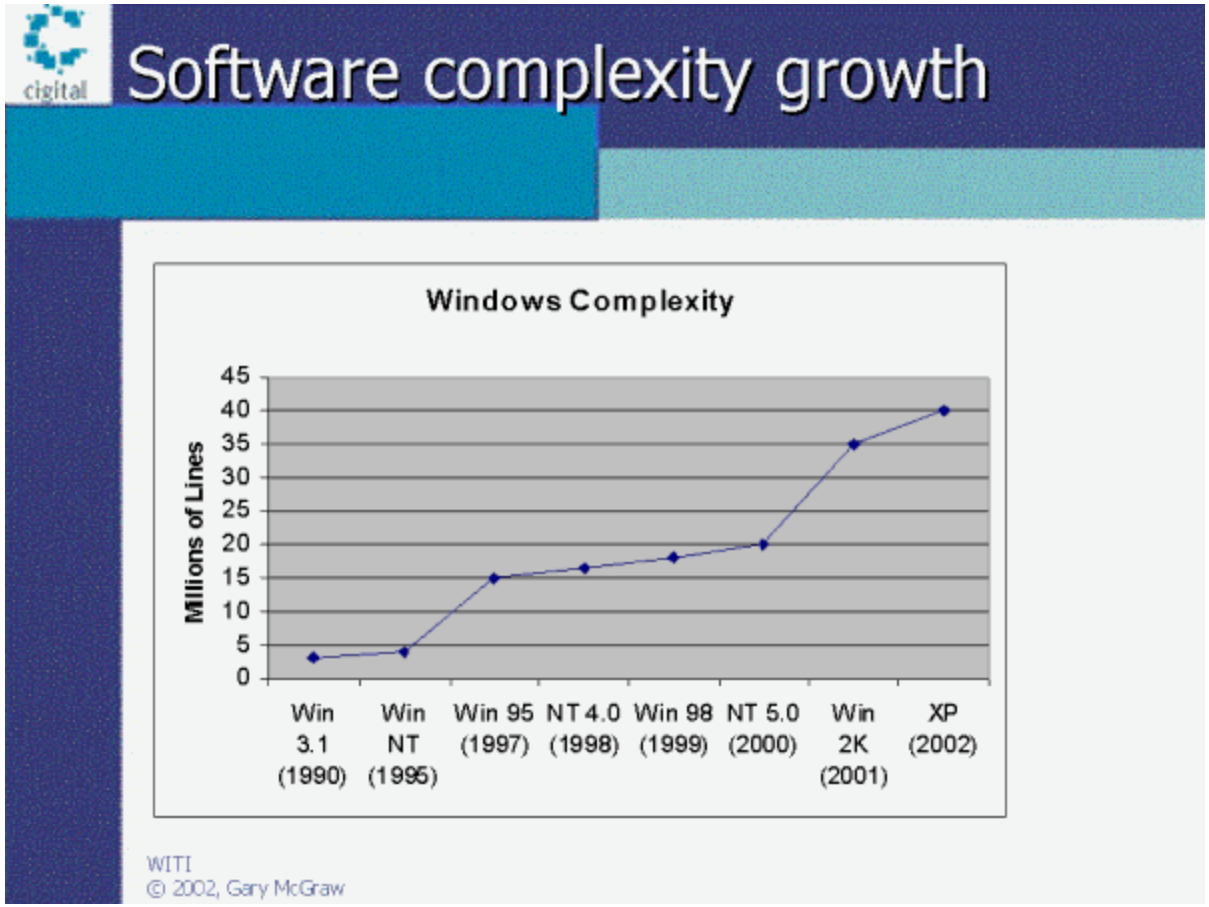
**What Is Wanted**

- Engineering Discipline
- Uniformity
- Consistency
- Preemptablity
- Single Point Logic Testing
- Trace - True
  - True Behavior Logic
  - True Logic Trace
  - True Logic Temporal Path
- Trace - False
  - False Behavior Logic
  - False Logic Trace
  - False Logic Temporal Path
- Well Defined
  - Rules
  - Specification
  - Analysis
- Orthogonal
  - Logic
  - Data

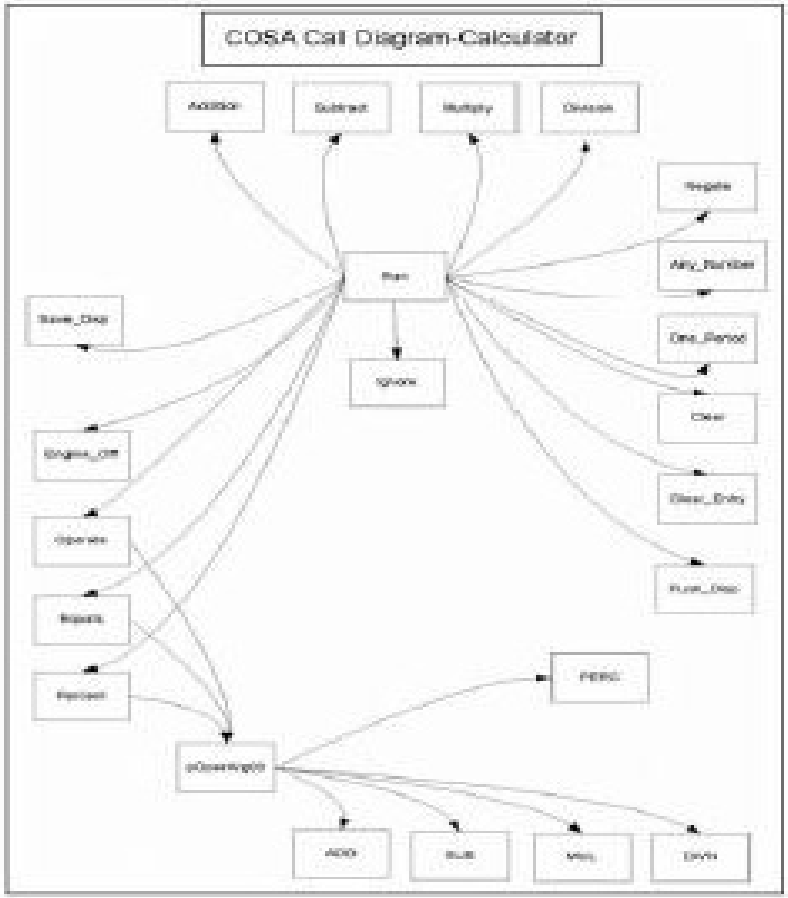
**COSA Delivers**

- Engineering Discipline
- Uniformity
- Consistency
- Preemptablity
- Single Point Logic Testing
- Trace - True
  - Static Document Trace
  - Dynamic Logic Trace
- Trace – False
  - Static Document Trace
  - Dynamic Logic Trace
- Well Defined
  - Template Rules
  - Specification
  - Traced Spec to Application
- Orthogonal
  - Logic
  - Data

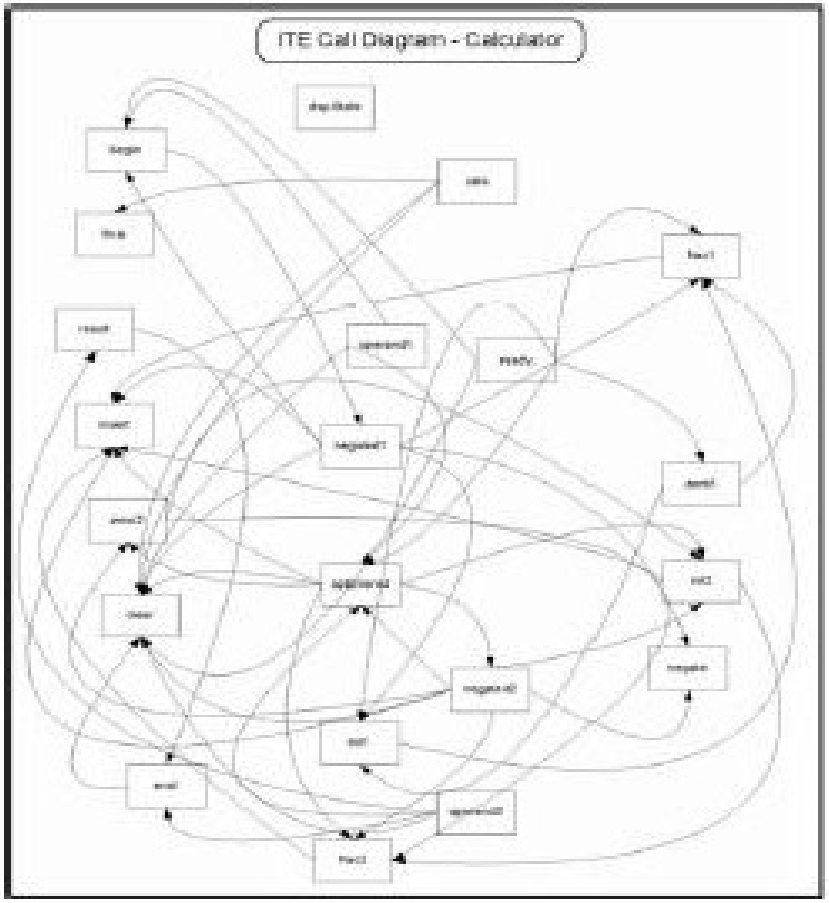
# Complexity is out of control



# Compare Complexity

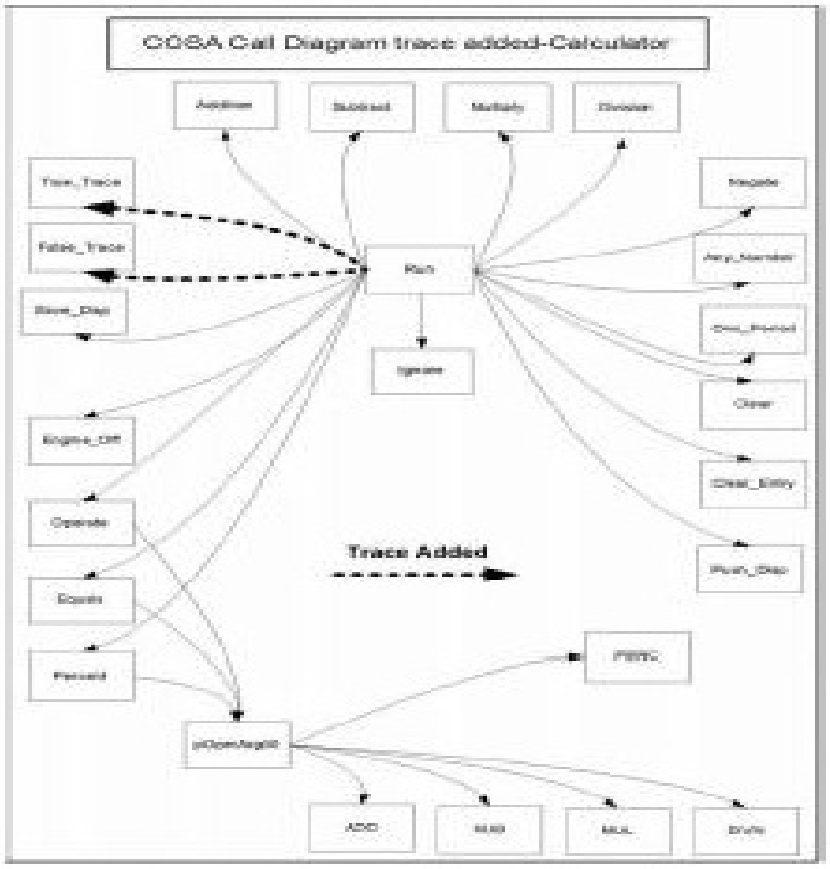



COSA Temporal Engineering

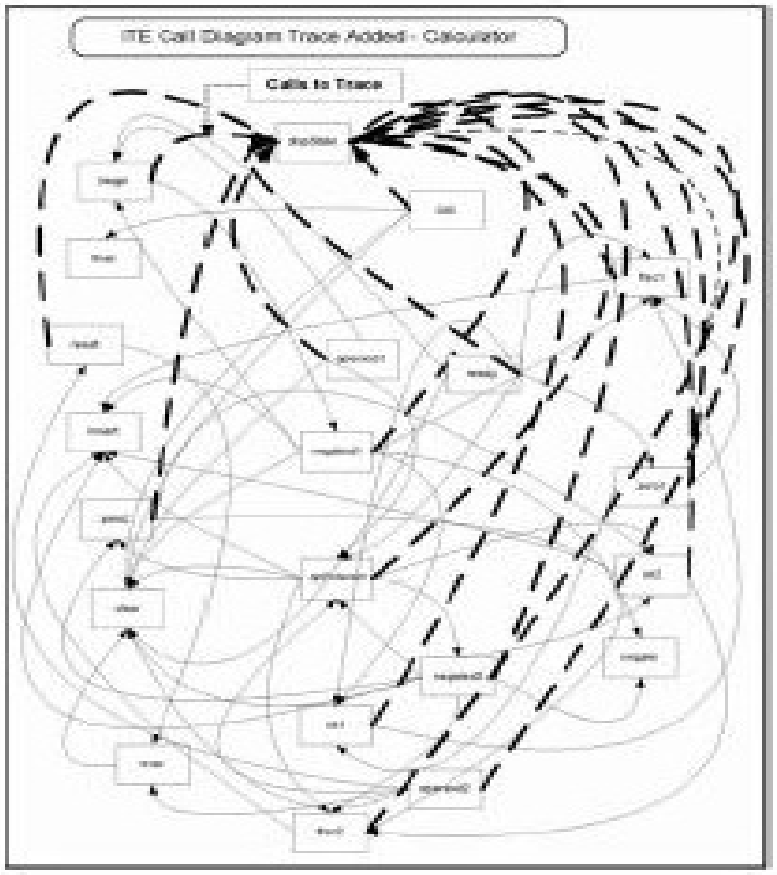



ITE Spatial Engineering

# Now with Trace




COSA Temporal Engineering
With Trace




ITE Spatial Engineering
With Trace

# ITE Trace Complexity

- 3 columns of trace
  - 4 columns info in each
- Little info
- Embedded throughout program
- Side effects
- 107 states

| State | Routine | Sig | Value |
|---|---|---|---|
| 1 | calc | 0 | 0 |
| 2 | calc | 0 | 0 |
| 3 | calc | 1 | 0 |
| 4 | ready | 0 | 0 |
| 5 | ready | 2 | 0 |
| 6 | ready | | |
| 7 | ready | 1 | 0 |
| 8 | begin | 0 | 0 |
| 9 | begin | 2 | 0 |
| 10 | begin | | |
| 11 | begin | 1 | 0 |
| 12 | begin | 1107 | 0 |
| 13 | negaged1 | 0 | 0 |
| 14 | begin | 0 | 0 |
| 15 | calc | 0 | 0 |
| 16 | begin | 3 | 0 |
| 17 | ready | 3 | 0 |
| 18 | ready | 0 | 0 |
| 19 | negaged1 | 2 | 0 |
| 20 | negated1 | | |
| 21 | negaged1 | 1 | 0 |
| 22 | negaged1 | 1010 | 0 |
| 23 | int1 | 0 | 0 |
| 24 | negaged1 | 0 | 0 |
| 25 | operand1 | 0 | 0 |
| 26 | calc | 0 | 0 |
| 27 | negaged1 | 3 | 0 |
| 28 | operand1 | 2 | 0 |
| 29 | operand1 | | |
| 30 | int1 | 2 | 0 |
| 31 | int1 | | |
| 32 | int1 | 1 | 0 |
| 33 | int1 | 1101 | 0 |
| 34 | frac1 | 0 | 0 |
| 35 | int1 | 0 | 0 |
| 36 | int1 | 3 | 0 |

| State | Routine | Sig | Value |
|---|---|---|---|
| 37 | frac1 | 2 | 0 |
| 38 | frac1 | | |
| 39 | frac1 | 1 | 0 |
| 40 | frac1 | 1010 | 0 |
| 41 | frac1 | 1010 | 0 |
| 42 | frac1 | 1010 | 0 |
| 43 | frac1 | 1010 | 0 |
| 44 | frac1 | 1010 | 0 |
| 45 | frac1 | 1107 | 0 |
| 46 | operand1 | 1107 | 0 |
| 47 | frac1 | 3 | -3.14159 |
| 48 | opEntered | 0 | -3.14159 |
| 49 | operand1 | 0 | -3.14159 |
| 50 | operand1 | 3 | -3.14159 |
| 51 | opEntered | 2 | -3.14159 |
| 52 | opEntered | | |
| 53 | opEntered | 1 | -3.14159 |
| 54 | opEntered | 1107 | -3.14159 |
| 55 | negated2 | 0 | -3.14159 |
| 56 | opEntered | 0 | -3.14159 |
| 57 | opEntered | 3 | -3.14159 |
| 58 | negated2 | 2 | -3.14159 |
| 59 | negated2 | | |
| 60 | negated2 | 1 | -3.14159 |
| 61 | negated2 | 1010 | -3.14159 |
| 62 | int2 | 0 | -3.14159 |
| 63 | negated2 | 0 | -3.14159 |
| 64 | operand2 | 0 | -3.14159 |
| 65 | calc | 0 | 0 |
| 66 | negated2 | 3 | -3.14159 |
| 67 | operand2 | 2 | -3.14159 |
| 68 | operand2 | | |
| 69 | int2 | 2 | -3.14159 |
| 70 | int2 | | |
| 71 | int2 | 1 | -3.14159 |

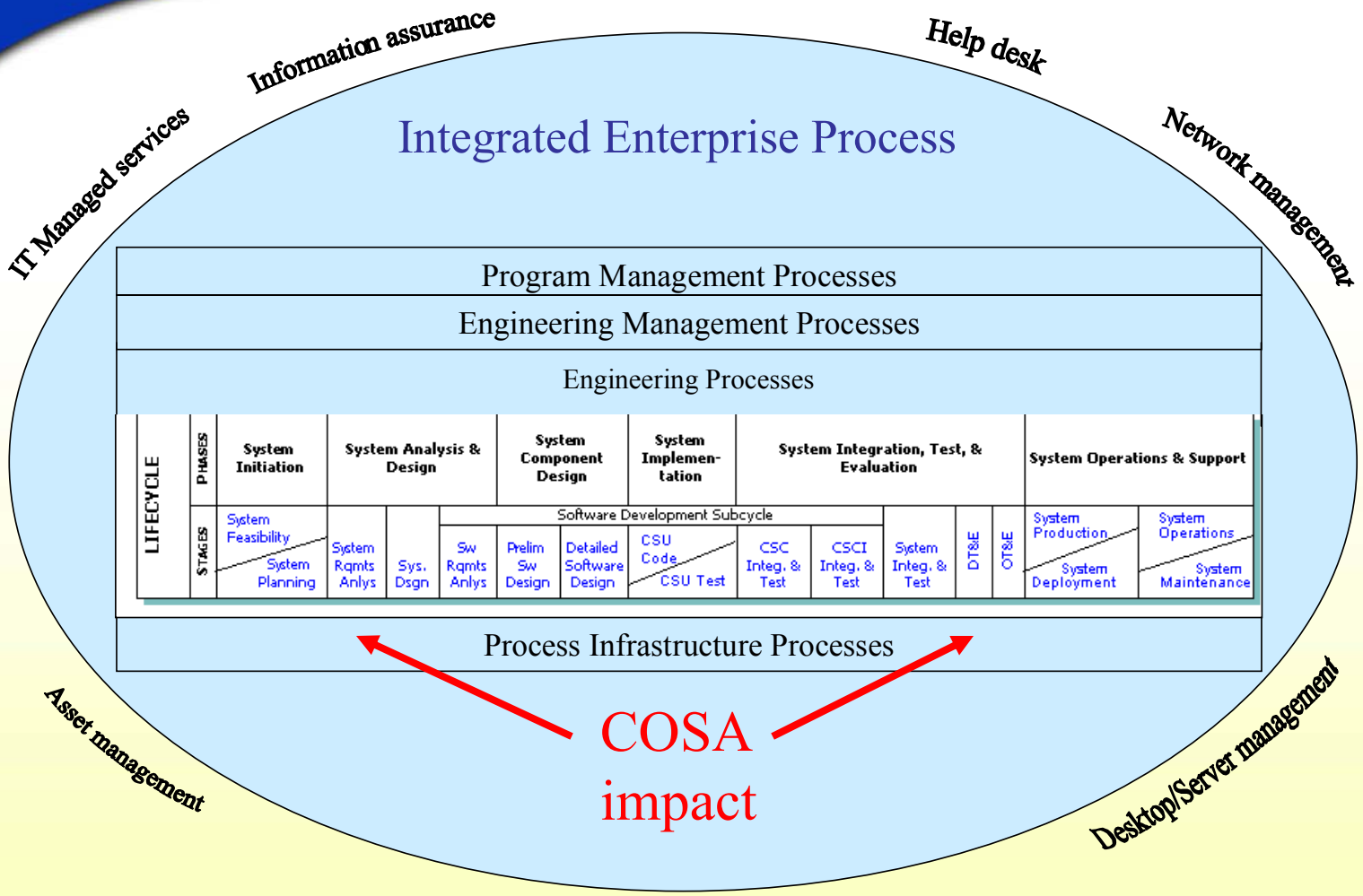
| State | Routine | Sig | Value |
|---|---|---|---|
| 72 | int2 | 1101 | -3.14159 |
| 73 | frac2 | 0 | -3.14159 |
| 74 | int2 | 0 | -3.14159 |
| 75 | int2 | 3 | -3.14159 |
| 76 | frac2 | 2 | -3.14159 |
| 77 | frac2 | | |
| 78 | frac2 | 1 | -3.14159 |
| 79 | frac2 | 1010 | -3.14159 |
| 80 | frac2 | 1010 | -3.14159 |
| 81 | frac2 | 1010 | -3.14159 |
| 82 | frac2 | 1010 | -3.14159 |
| 83 | frac2 | 1010 | -3.14159 |
| 84 | frac2 | 1102 | -3.14159 |
| 85 | operand2 | 1102 | -3.14159 |
| 86 | frac2 | 3 | -3.14159 |
| 87 | result | 0 | 0 |
| 88 | operand2 | 0 | -3.14159 |
| 89 | ready | 0 | 0 |
| 90 | calc | 0 | 0 |
| 91 | operand2 | 3 | -3.14159 |
| 92 | ready | 2 | 0 |
| 93 | ready | | |
| 94 | result | 2 | 0 |
| 95 | result | | |
| 96 | eval | 1104 | 0 |
| 97 | result | 1 | -1 |
| 98 | result | 100 | -1 |
| 99 | ready | 100 | -1 |
| 100 | calc | 100 | -1 |
| 101 | result | 3 | -1 |
| 102 | ready | 3 | -1 |
| 103 | final | 0 | -1 |
| 104 | calc | 0 | -1 |
| 105 | calc | 3 | -1 |
| 106 | final | 2 | -1 |
| 107 | final | 1 | -1 |

# COSA Trace More Information

- 1 Column Trace
  - 8 Columns Info
- Reduced Complexity
- More Information
- Dynamic On-Off
- Minimal side effects
- 30 states vs. 107

| Count | Step | Trace | Eng | Static | Dynamic | Behavior | Value |
|---|---|---|---|---|---|---|---|
| 1 | +T= 0; | 100 | Off; | 44; | 44; | Negate; | N= - |
| 2 | +T= 1; | 101 | Off; | 1; | 1; | Any_Number; | N= -3 |
| 3 | ĞF= 1; | 101 | On; | 1; | 59; | Ignore; | N= |
| 4 | +T= 2; | 102 | Off; | 59; | 59; | One_Period; | N= -3. |
| 5 | +T= 3; | 103 | Off; | 1; | 1; | Any_Number; | N= -3.1 |
| 6 | +T= 3; | 103 | Off; | 1; | 1; | Any_Number; | N= -3.14 |
| 7 | +T= 3; | 103 | Off; | 1; | 1; | Any_Number; | N= -3.141 |
| 8 | +T= 3; | 103 | Off; | 1; | 1; | Any_Number; | N= -3.1415 |
| 9 | +T= 3; | 103 | Off; | 1; | 1; | Any_Number; | N= -3.14159 |
| 10 | ĞF= 3; | 103 | On; | 1; | 44; | Ignore; | N= |
| 11 | ĞF= 4; | 104 | On; | 12; | 44; | Ignore; | N= |
| 12 | ĞF= 5; | 105 | On; | 11; | 44; | Ignore; | N= |
| 13 | ĞF= 6; | 106 | On; | 1; | 44; | Push_Disp; | N= |
| 14 | ĞF= 7; | 500 | On; | 43; | 44; | Ignore; | N= |
| 15 | +T= 8; | 501 | On; | 44; | 1; | Subtraction; | N= -3.14159 |
| 16 | +T= 12; | 700 | Off; | 1; | 1; | Engine_Off; | N= -3.14159 |
| 17 | +T= 13; | 701 | Off; | 44; | 44; | Negate; | N= - |
| 18 | +T= 14; | 702 | Off; | 1; | 1; | Any_Number; | N= -2 |
| 19 | ĞF= 14; | 702 | Off; | 1; | 59; | Ignore; | N= |
| 20 | +T= 15; | 703 | Off; | 59; | 59; | One_Period; | N= -2. |
| 21 | +T= 16; | 704 | Off; | 1; | 1; | Any_Number; | N= -2.1 |
| 22 | +T= 16; | 704 | Off; | 1; | 1; | Any_Number; | N= -2.14 |
| 23 | +T= 16; | 704 | Off; | 1; | 1; | Any_Number; | N= -2.141 |
| 24 | +T= 16; | 704 | Off; | 1; | 1; | Any_Number; | N= -2.1415 |
| 25 | +T= 16; | 704 | Off; | 1; | 1; | Any_Number; | N= -2.14159 |
| 26 | ĞF= 16; | 705 | On; | 1; | 13; | Ignore; | N= |
| 27 | ĞF= 18; | 706 | On; | 12; | 13; | Ignore; | N= |
| 28 | ĞF= 17; | 707 | On; | 1; | 13; | Save_Disp; | N= |
| 29 | ĞF= 19; | 900 | On; | 11; | 13; | Ignore; | N= |
| 30 | +T= 20; | 901 | Off; | 13; | 13; | Equals; | N= -1 |

# CMMI Extensibility and Flexibility




 Copyright Northrop Grumman Company 2003

# A 4 Step COSA Solution

1) Well Defined Core Foundation
   - COSA – Table Drive Vector State Machine
     - Temporal Software Engineering
2) Model Driven Architecture
   - WYSIWYG BNF model to application
   - Rules / Logic can be tested on boundary values
3) Re-manufactured Applications
   - Legacy Integrated Forward Engineering (LIFE)
   - Replaces & Reduces Maintenance Costs
4) System Level Integration
   - Focus on top-down organization
   - Reduce failures, improve quality, reduce costs

# The End