


0 0 1 0 1 0 0 0
1 0 0 0 1 0 0
0 1 0 0 0 0 1
0 0 0 1 1 0 1
0 1 0 0 0 0
0 1 0 0 0 1 1 0
1 0 0 0 1 0 0 1
0 0 1 1 0 0 0 0
0 1 0 1 0 1 1
1 0 1 1 0 0 0
0 1 0 0 1 1
1 0 0 1 0 0
0 1 0 0 0 1
0 0 1 0 0 0
0 0 0 0
0 1 0
1 0 1
0
0

Security Impact Analysis for Common Criteria Evaluations

Shawn Bohner, Ph.D. and Denis Gracanin, Ph.D.
Department of Computer Science
Virginia Tech

The Challenge

```
0 0 1 0 1 0 0 0
1 0 0 0 1 0 0
0 1 0 0 0 0 1
0 0 0 1 1 0 1
  0 1 0 0 0 0
0 1 0 0 0 1 1 0
1 0 0 0 1 0 0 1
0 0 1 1 0 0 0 0
0 1 0 1 0 1 1
1 0 1 1 0 0 0
0 1 0 0 1 1
1 0 0 1 0 0
0 1 0 0 0 1
0 0 1 0 0 0
  0 0 0 0
  0 1 0
1 0 1
  0
  0
```

- Demand for Common Criteria Information Technology Security Evaluations exceeding supply of Evaluators
 - Labor intensive CCITSE process
 - Effort in Weeks and Calendar time in Months
 - National Information Assurance Acquisition Policy (NSTISSP #11) July 2002 mandate for SW security evaluations
 - Limited Number of Testing Labs
 - And then there are all the software updates...
- How can this situation be alleviated?
 - Relax policy & allow lesser/non-evaluated systems
 - Increase supply of Evaluators
 - Increase the productivity of Evaluators 

Goal: Quicken and Clarify CCITSE

```
0 0 1 0 1 0 0 0
1 0 0 0 1 0 0
0 1 0 0 0 0 1
0 0 0 1 1 0 1
  0 1 0 0 0 0
0 1 0 0 0 1 1 0
1 0 0 0 1 0 0 1
0 0 1 1 0 0 0 0
0 1 0 1 0 1 1
1 0 1 1 0 0 0
0 1 0 0 1 1
1 0 0 1 0 0
0 1 0 0 0 1
0 0 1 0 0 0
  0 0 0 0
  0 1 0
1 0 1
  0
  0
```

- Improve Efficiency of CCITSE Process through Better Navigation
 - Reduce time in navigating the documentation (shorten the conceptual distances)
 - Reduce effort and time by identifying failing evaluations early
 - Reduce time for key time consuming activities
- Improve Effectiveness of CCITSE Process through Better Visibility
 - Increase confidence of evaluations
 - Better decisions
- Bottom Line: Automate Key Evaluation Steps

Technical Approach

```
0 0 1 0 1 0 0 0
1 0 0 0 1 0 0
0 1 0 0 0 0 1
0 0 0 1 1 0 1
  0 1 0 0 0 0
0 1 0 0 0 1 1 0
1 0 0 0 1 0 0 1
0 0 1 1 0 0 0 0
0 1 0 1 0 1 1
1 0 1 1 0 0 0
0 1 0 0 1 1
1 0 0 1 0 0
0 1 0 0 0 1
0 0 1 0 0 0
  0 0 0 0
  0 1 0
  1 0 1
    0
    0
```

- Employ Complementary Technologies
 - Software Impact Analysis
 - Software Visualization / Virtual Environments
- Develop Software Security Impacts Model to Analyze Inherent Dependencies
 - Relevant to CC* structure and semantics
 - Provides traceability framework for revising TOE
 - Readily depicted in a visual context
- Develop Virtual Environment for Evaluators
 - Analyze the Target Of Evaluation Artifacts
 - Navigate the TOE Artifacts during Evaluation

SIA-Viz Builds on NIST's CCTool

```
00101000  
1000100  
0100001  
0001101  
010000  
01000110  
10001001  
00110000  
0101011  
1011000  
010011  
100100  
010001  
001000  
0000  
010  
000  
0
```

Environmental Considerations (Policies, Threats, & Assumptions)

Target Of Evaluation (System & Documents)

Security Objectives

Security Requirements

Common Criteria Standard

Security Target (ST) / Protection Profile (PP)

Common Criteria Evaluation

Revised TOE

CCTool Application

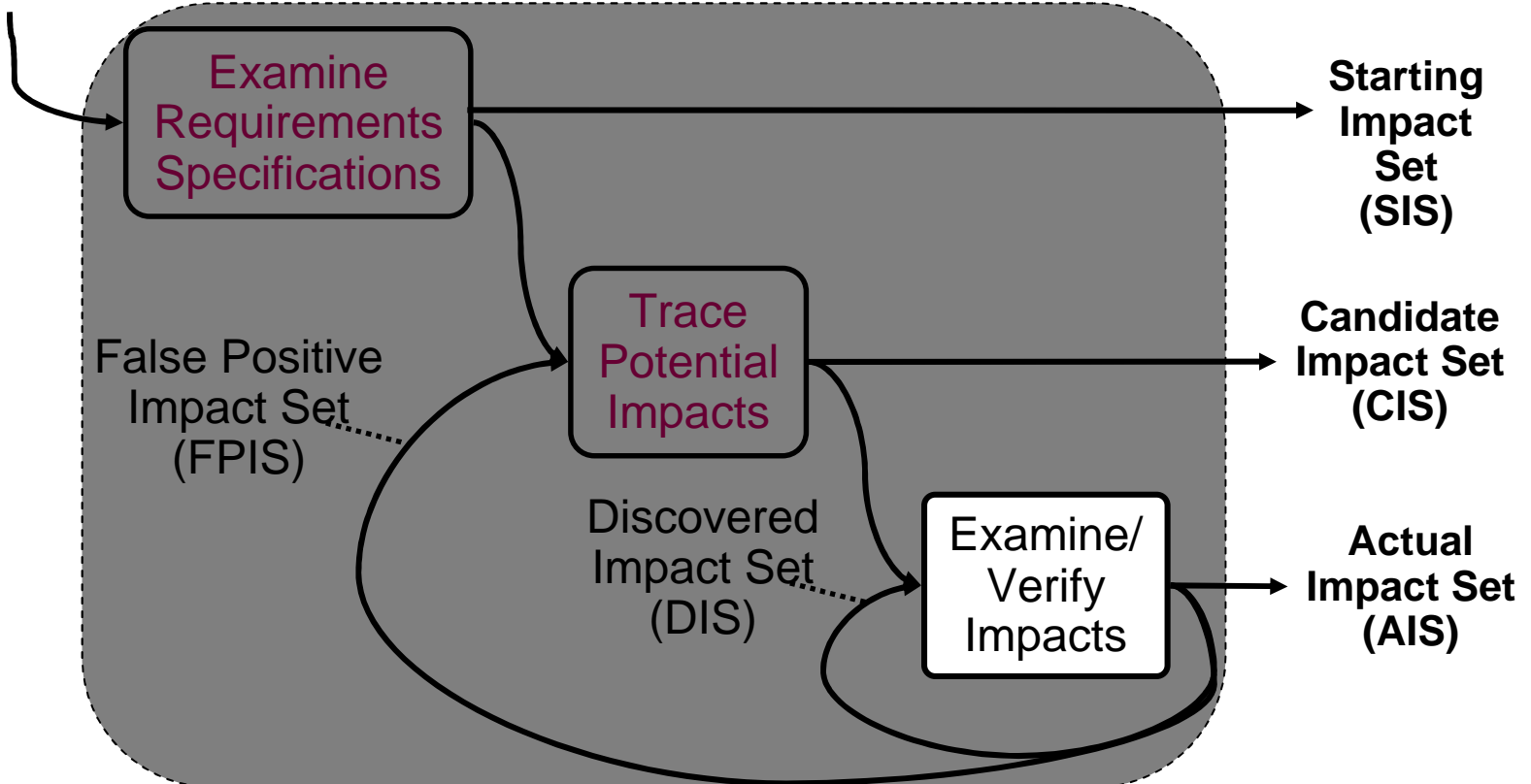
SIA-Viz Application

Basic Security Impact Analysis

```

00101000
100 0100
0100001
0001 101
 01000 0
01000110
10001001
00110000
01 01011
1011000
010011
1 00100
010 001
001 000
 00 00
 0 10
1 01
 0
 0
  
```

**Common Criteria
Security Requirements**



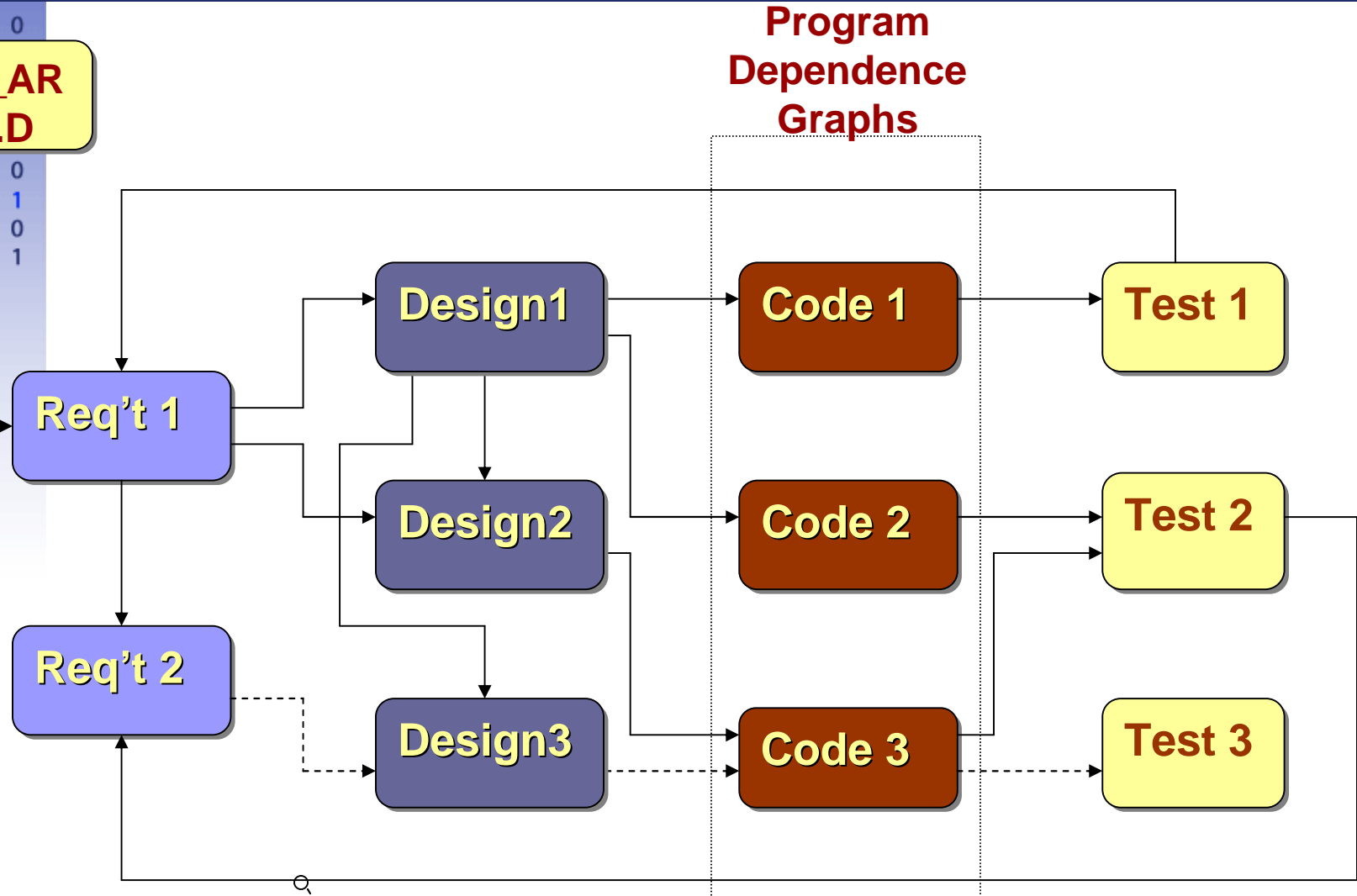
**TOE Software Artifacts
(Custom, Middleware, COTS)**

$$AIS = CIS + DIS - FPIS$$

Traceability View of Life Cycle Objects

```
00101000
10
01
00
01000110
10001001
00110000
01010111
10110000
010011
1001000
010001
0010000
00
010
001010
000
101
000
```

**FUA_AR
P 1.D**



Type and Strength

Visualizing and Navigating the Impacts

```

0 0 1 0 1 0 0 0
1 0 0 0 1 0 0
0 1 0 0 0 0 1
0 0 0 1 1 0 1
0 1 0 0 0 0
0 1 0 0 0 1 1 0
1 0 0 0 1 0 0 1
0 0 1 1 0 0 0 0
0 1 0 1 0 1 1
1 0 1 1 0 0 0
0 1 0 0 1 1
1 0 0 1 0 0
0 1 0 0 0 1
0 0 1 0 0 0
0 0 0 0 0
0 1 0
1 0 1
0
0

```

Visual Demo ✖

Step 1: Security | Step 2: Metaphor | Step 3: Visualization

Select node to view security impacts to other components:

walk

fly

study

plan

pan

turn

roll

goto
align
view
restore
fit

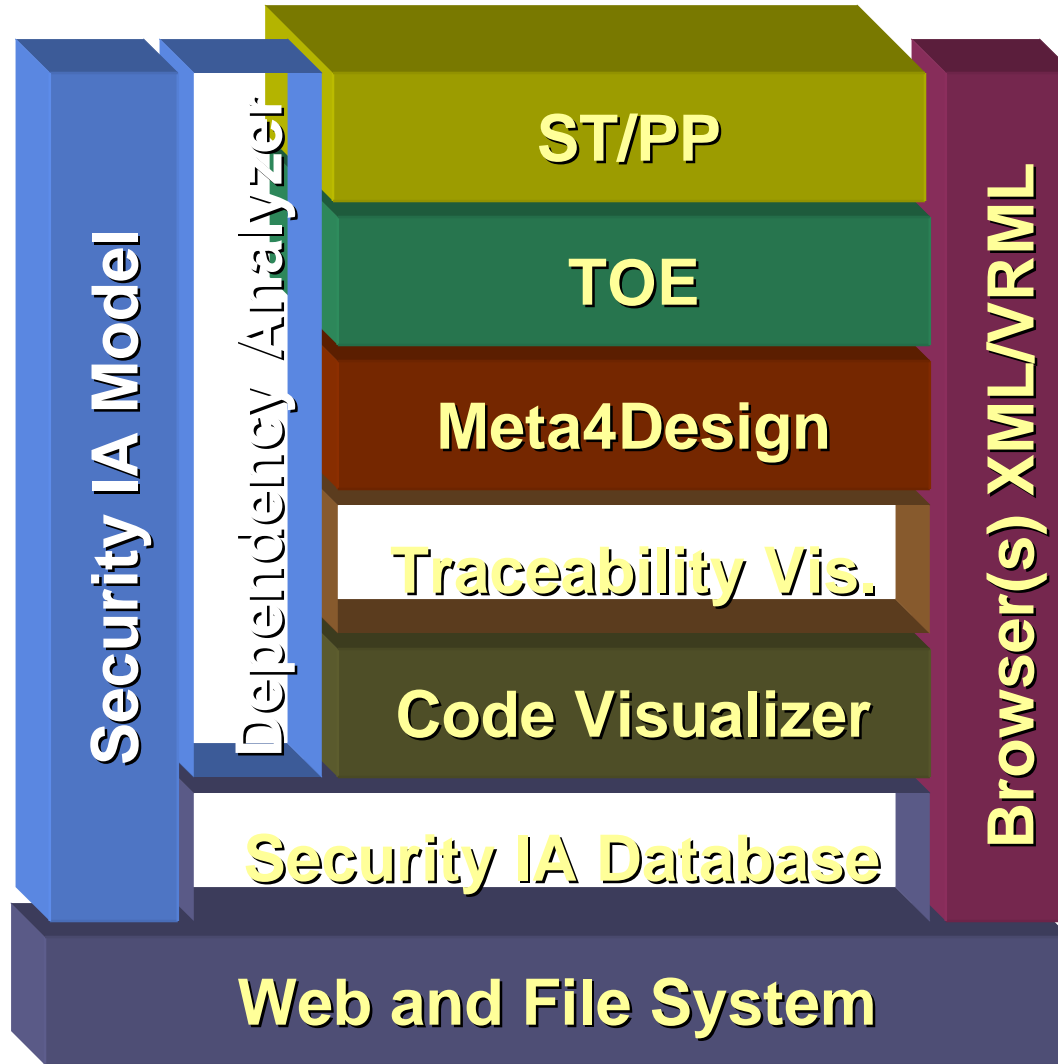
Security file: ...
Metaphor file: ...
Visual file: ...
Entry View
Top View
Side View
Save As...

Architecture of SIA-Viz Environment

```

00101000
100 0100
0100001
0001 101
 01000 0
01000110
10001001
00110000
01 01011
1011000
010011
1 00100
010 001
001 000
 00 000
 0 10
 1 001
    0
  
```

CCTool



**CCITSE
Evaluator**



**Common
Criteria
Evaluation
Methodology**

Prototype Assumptions

```
00101000
100 0100
0100001
0001 101
 01000 0
01000110
10001001
00110000
01 01011
1011000
010011
1 00100
010 001
001 000
 00 00
 0 10
 1 01
  0
  0
```

- Proof of Concept Prototype
 - Navigation and Analysis
 - Assume XML and Java (for now)
- Evaluate Dependency Analysis Models
 - Software Engineering + Security
- Experiment with Appropriate Metaphors
 - Investigator/Explorer
 - Universe/Geographic Space
 - Immersion in Virtual Environment
- Establish Foundation for More Aggressive Automation Opportunities

Status and Next Steps

```
00101000
100 0100
0100001
0001 101
 01000 0
01000110
10001001
00110000
01 01011
1011000
010011
1 00100
010 001
001 000
 00 00
 0 10
 1 01
    0
    0
```

- Basic Security Impact Model in Place
- Developing Overall SIA-Viz User Interface
 - Solidifying Metaphor for Universal Objects
 - Developing Metaphors for Specific Contexts
- Developing Security Impact Analysis Database
- Integrating Traceability and Program Dependency Analysis Elements
- Talking with Common Criteria Evaluation Vendors for Validation on Real Examples