Security Impact Analysis for Common Criteria Evaluations

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

The Challenge

- 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

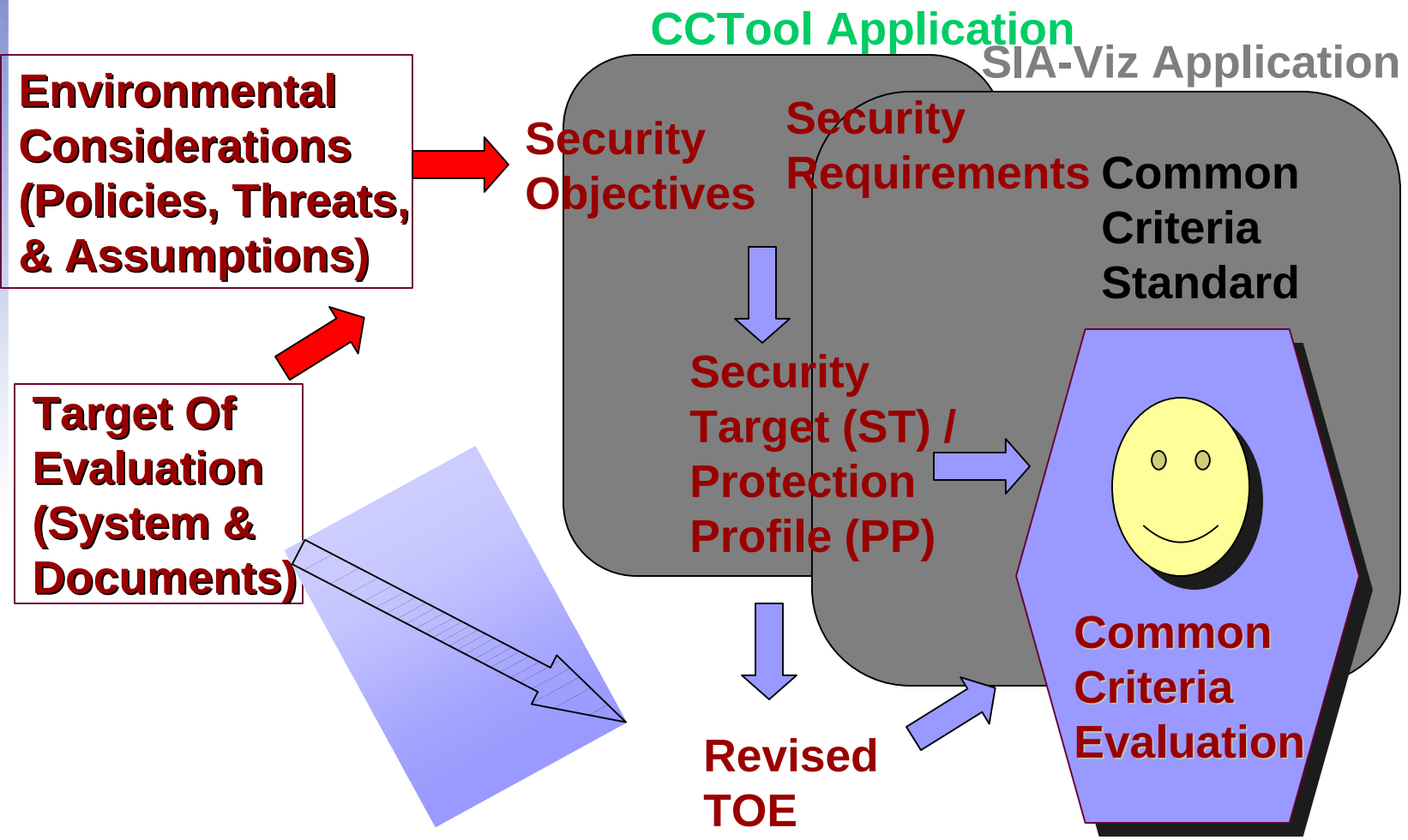
Goal: Quicken and Clarify CCITSE

- 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

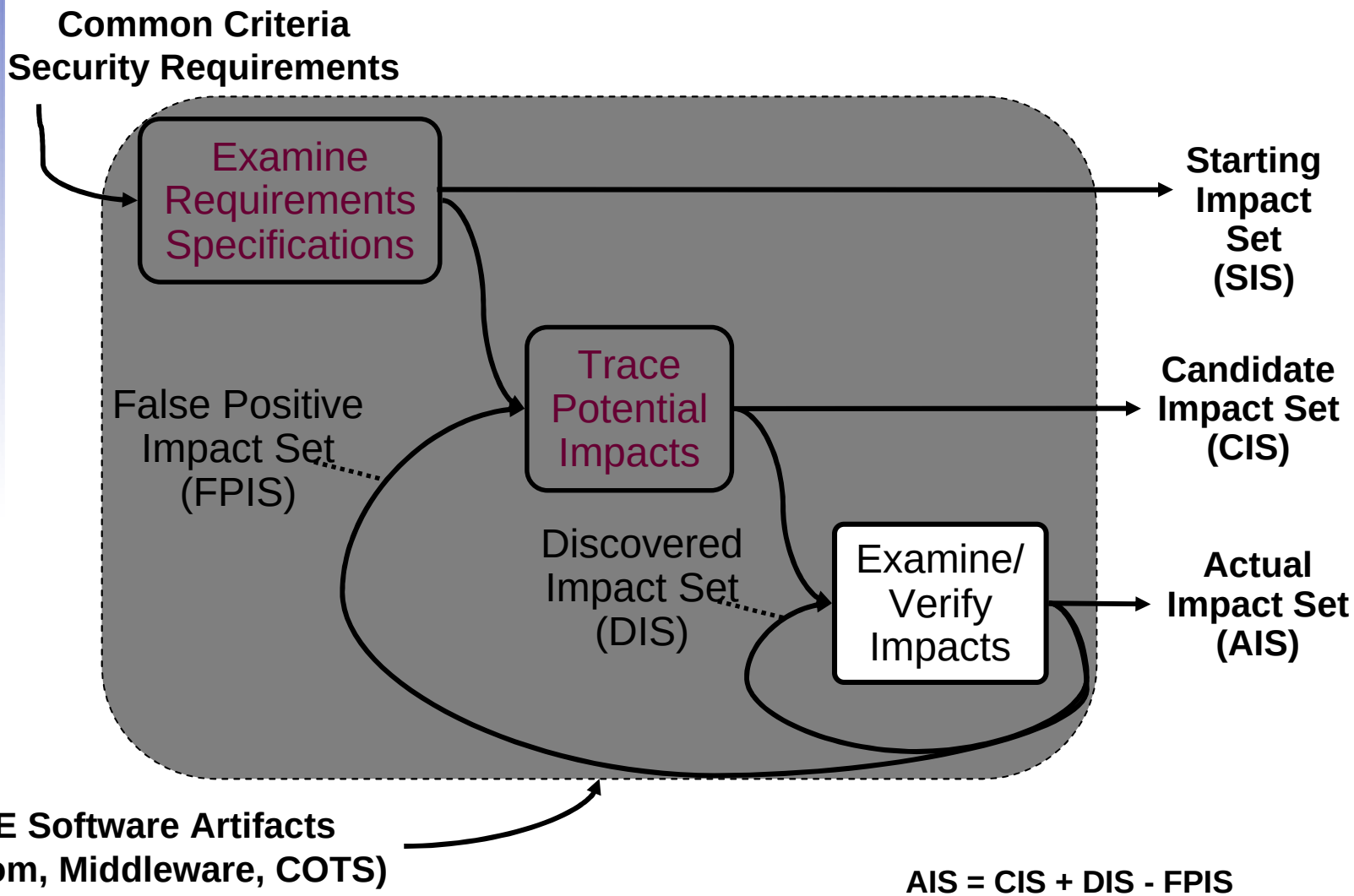
Technical Approach

- 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

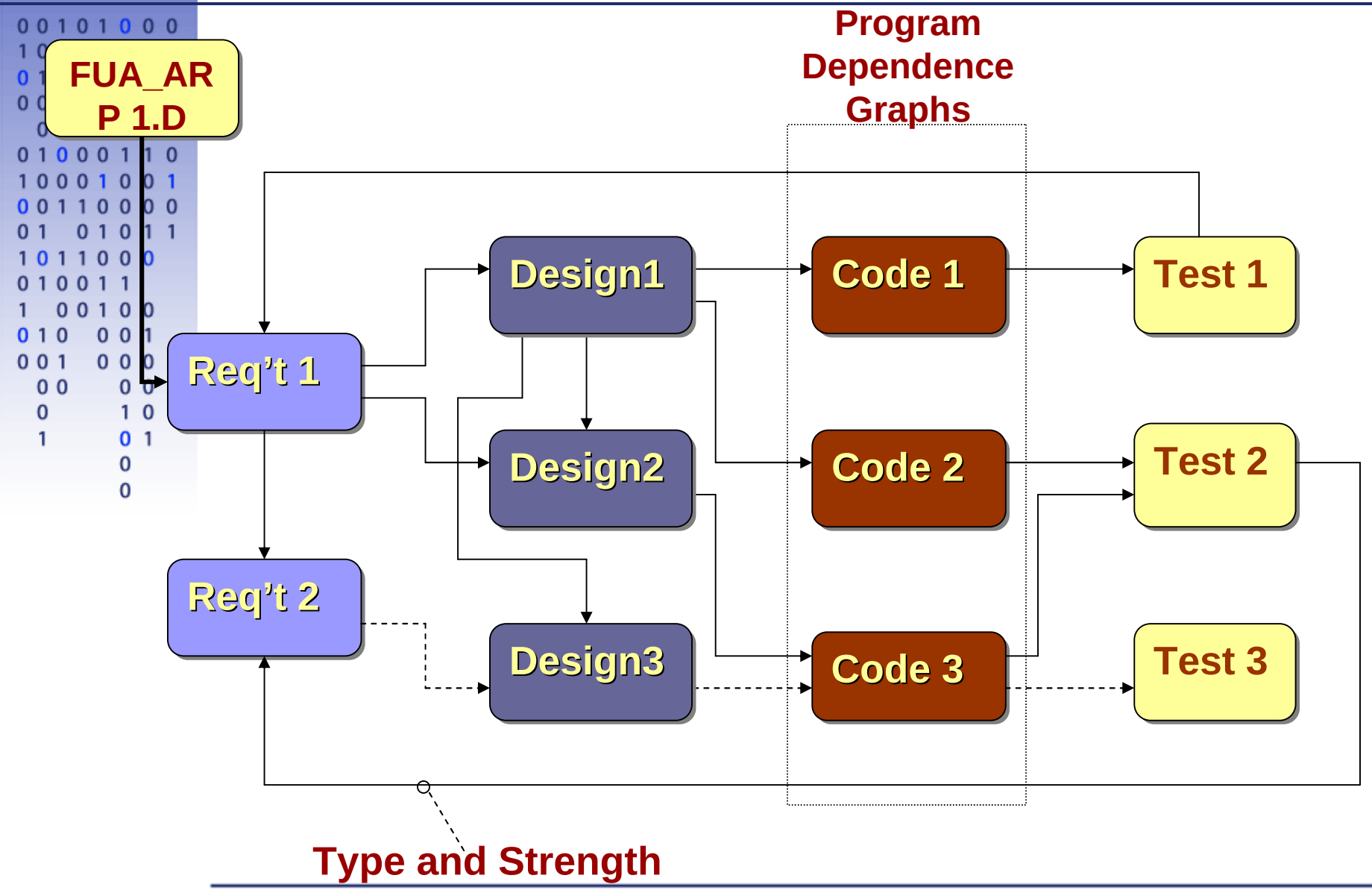


Basic Security Impact Analysis

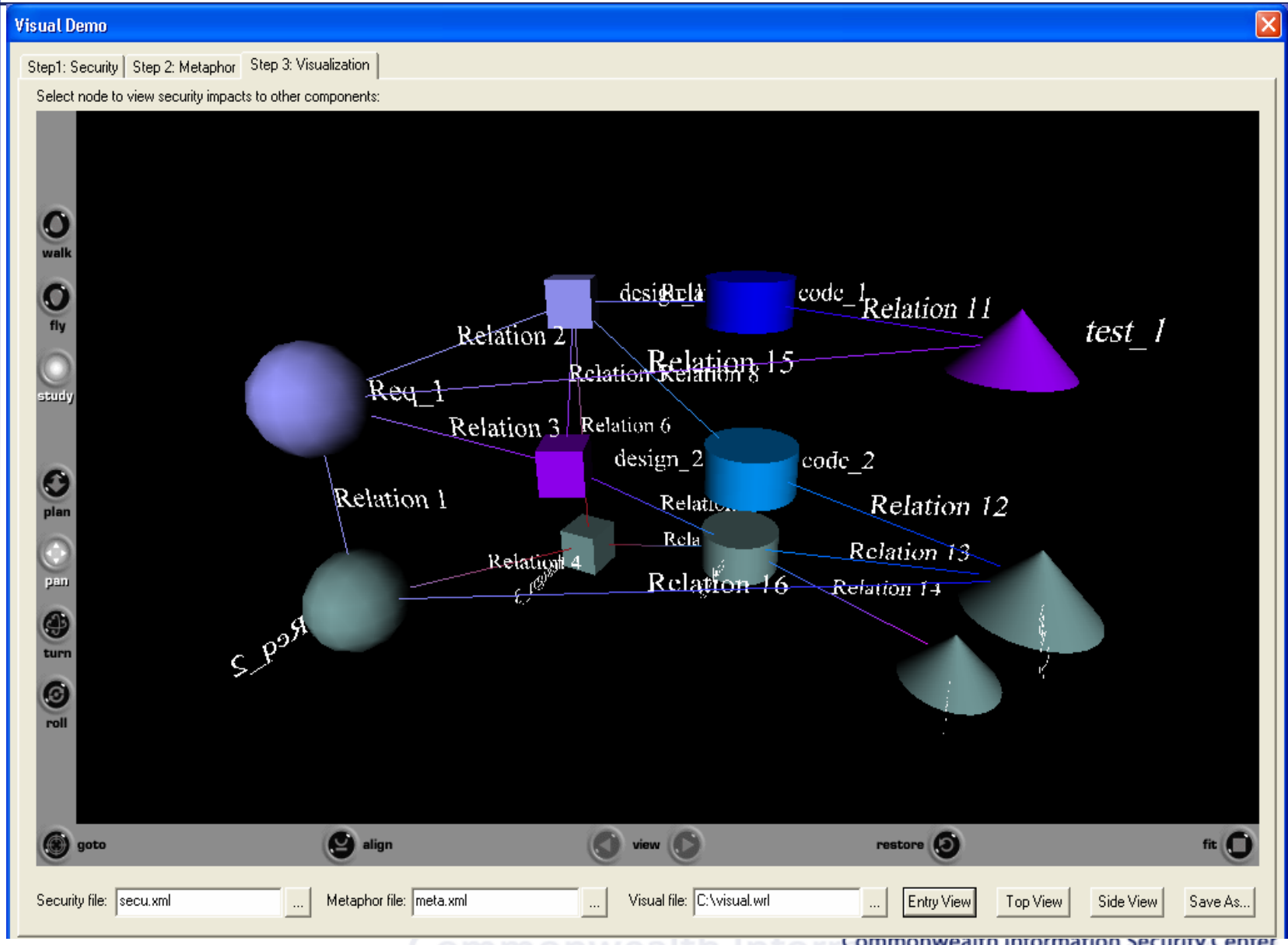


AIS = CIS + DIS - FPIS

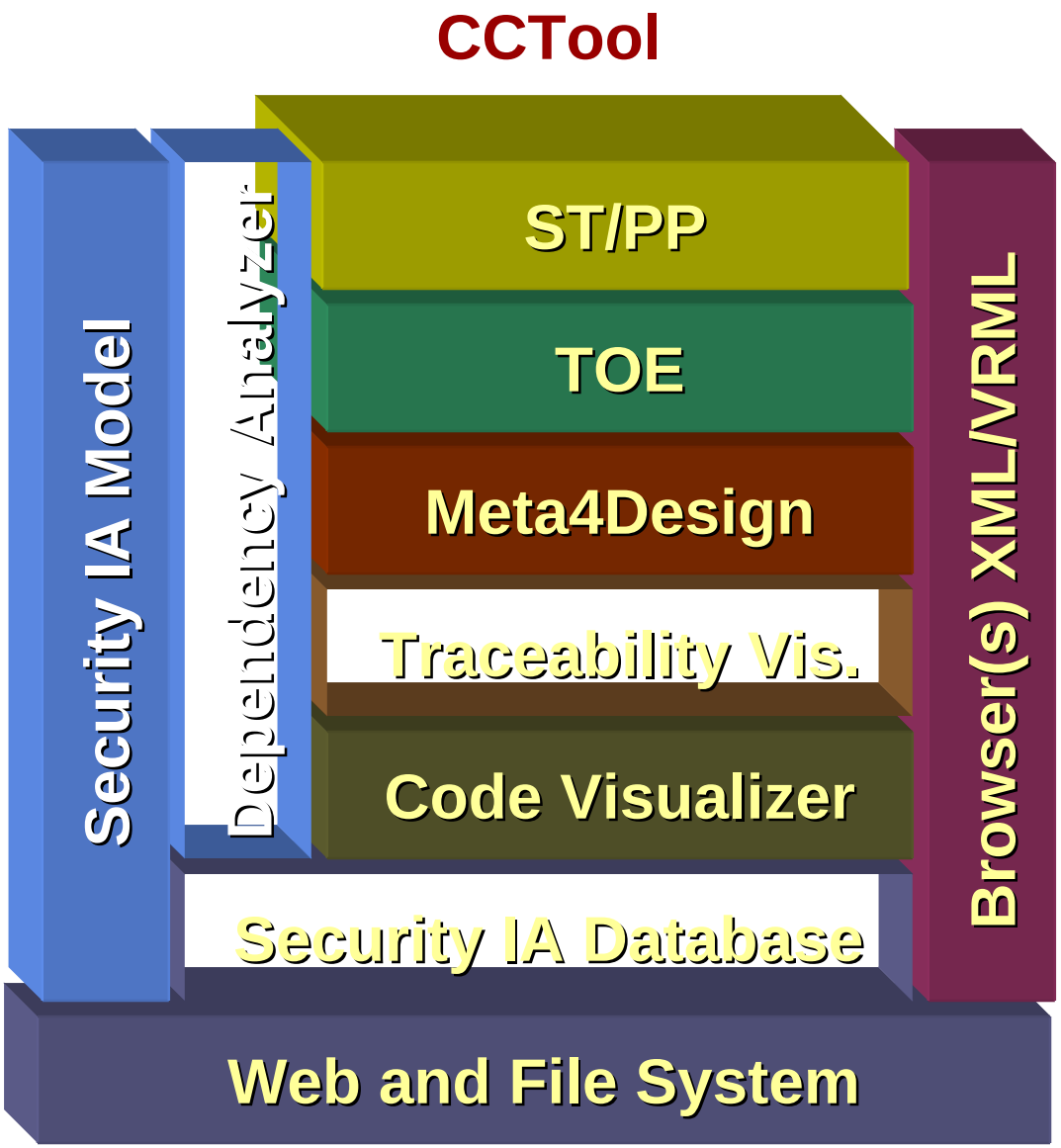
Traceability View of Life Cycle Objects



Visualizing and Navigating the Impacts

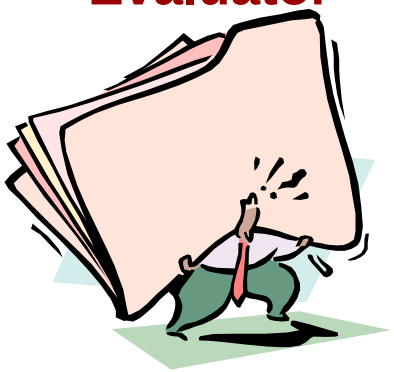


Architecture of SIA-Viz Environment



Common Criteria Evaluation Methodology

CCITSE Evaluator



Prototype Assumptions

- 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

- 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