Compositional Certification

John Rushby

Computer Science Laboratory
SRI International
Menlo Park CA USA

Systems, Components, and Properties

- Security, for example, is a system property
- But there is a compelling case to establish a marketplace for security-relevant components (cf. MILS)
  - Secure file systems, communications subsystems, operating system kernels
  - Filters, downgraders, authentication services
- Want the security of these components to be evaluated
- In such a way that security evaluation for a system built on these is largely based on prior evaluations of the components
- This is an example of compositional assurance
- Wanted for safety and other critical system properties as well as security

Component-Based Design and Compositional Assurance

- Component-based design
  - Take some off the shelf components
  - Build some bespoke components
  - Connect them all together with glue (components)

  To achieve the required functionality
  - We understand the functionality of the system by understanding the functions of its components
- Compositional assurance
  - This is the idea that we can provide assurance for properties of a component-based system based on preconstructed assurance for properties of its components

Why Is Compositional Assurance Hard?

- Assurance considers properties, not just function
  - Properties depend on component interactions as much as on individual component behavior
  - And must consider what must not happen
- Assurance must consider faults and malice
  - Including those that subvert the design
  - In particular, those that vitiate the separation into components and bypass the interfaces between them
  - i.e., those that create unintended interactions
- So assurance for components must anticipate this and provide very strong guarantees, and must consider interactions as well as local behavior

Frameworks for Compositional Assurance

- Assurance is about properties delivered at interfaces
- So, for compositional assurance, we need:
  - Precise properties
    - Must be meaningful at interfaces
      - So they can be evaluated locally
    - Must be meaningful in combination
      - So they compose to yield evaluable system properties
  - Precise interfaces (the paths for component interaction)
    - There must be no paths for component interaction outside the known interfaces, even in the presence of faults, or of malice in untrusted components
- Feasibility of compositional assurance depends on architectural frameworks that guarantee interfaces
  - E.g., TTA (safety), MILS (security)

Compositional Analysis

- Computer scientists have ways to do compositional verification of programs—e.g., prove
  - Program A guarantees P if environment ensures Q
  - Program B guarantees Q if environment ensures P

  Conclude that $A \,||\, B$ guarantees P and Q
- Assumes programs interact only through explicit computational mechanisms (e.g., shared variables)
- Software and systems can interact through other mechanisms
  - Computational context: shared resources
  - Noncomputational mechanisms: the controlled plant
- Need eliminate, control, and understand these paths for interaction
  - Requirement is no unintended interactions

Unintended Interaction Through Shared Resources

- This must not happen
- Need an integration framework (i.e., an architecture) that guarantees composability

  Composability: properties of a component are preserved when it is used within a larger system
- This is what partitioning is about in avionics
- Or separation in a MILS security context

Composability

Partitioning ensures **composability** of components

- Properties of a collection of interacting components are preserved when they are placed (suitably) in the environment provided by a collection of partitioning mechanisms
- Hence partitioning does not get in the way
- And the combination is itself composable
- Hence components cannot interfere with each other nor with the partitioning mechanisms

Additivity

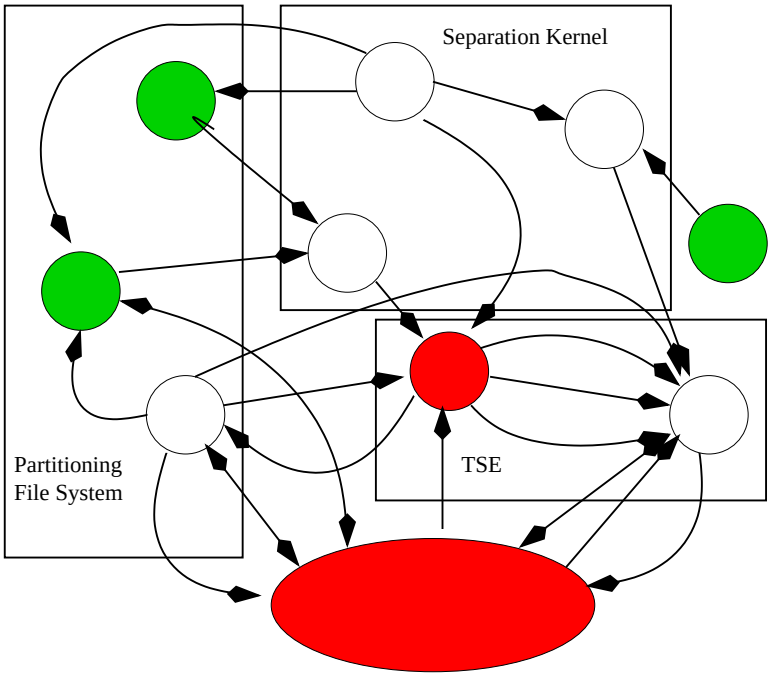
Partitioning mechanisms compose with each other **additively**

- e.g., **partitioning(kernel)** + **partitioning(network)** provides **partitioning(kernel + network)**

Partitioning (composability and additivity) make the world safe for compositional reasoning

Illustration: MILS

Security policy is enforced by trusted subjects (colored circles) interacting over known channels (arrows); prefer many small, simple trusted subjects to few complex ones; can afford this because we can efficiently and securely share physical resources among separate logical circles and arrows



Secure sharing is ensured by foundational components, which enforce partitioning/separation

Unintended Interaction Through The Plant

- The notion of interface must be expanded to include assumptions about the noncomputational environment (i.e., the plant)
  - Cf. Ariane V failure (due to differences from Ariane IV)
- Compositional reasoning must extend to take the plant into account (i.e., composition of hybrid systems)
- Control engineers do this, computer scientists are less familiar with it
- Must consider response to failures
  - Avoid domino effect
  - Control number of cases (otherwise exponential)
- And dynamic system compositions
  - Medical devices are a good case study

State of Practice in Compositional Assurance

- Not endorsed by any stringent certification regime I am familiar with
  - Because of the interaction issue: the current way to deal with this is to look at the whole system and inside every component
- E.g., the FAA certifies only airplanes, engines, propellers
  - Some weak mechanisms for components
    - ⋆ Reusable Software Components (AC 20-148)
  - And for incremental construction of certification
    - ⋆ Integrated Modular Avionics (DO-297/ED-124)
  - But the initial certification is always whole system, not compositional, and they reserve the right to look inside components
- Perhaps we need to rethink the basis for certification

Approaches to Certification

- All assurance is based on **arguments** that purport to justify certain **claims**, based on documented **evidence**
- There are two approaches to assurance: standards-based, and goal-based
- They differ in how explicit is the claims, evidence, argument structure

The Standards-Based Approach to Software Certification

- E.g., airborne s/w (DO-178B), security (Common Criteria)
- Applicant follows a prescribed method (or processes)
  - Delivers prescribed outputs
    - ⋆ e.g., documented requirements, designs, analyses, tests and outcomes, traceability among these
- Standard usually defines only the evidence to be produced
- The claims and arguments are implicit
- Hence, hard to tell whether given evidence meets the intent
- Works well in fields that are stable or change slowly
  - Can institutionalize lessons learned, best practice
    - ⋆ e.g., evolution of DO-178 from A to B to C
- But less suitable with novel problems, solutions, methods

The Goal-Based Approach to Software Certification

- E.g., air traffic management (CAP670 SW01), UK aircraft
- Applicant develops an assurance case
  - Whose outline form may be specified by standards or regulation (e.g., MOD DefStan 00-56)
  - Makes an explicit set of goals or claims
  - Provides supporting evidence for the claims
  - And arguments that link the evidence to the claims
    - ⋆ Make clear the underlying assumptions and judgments
    - ⋆ Should allow different viewpoints and levels of detail
- The case is evaluated by independent assessors
  - Explicit claims, evidence, argument

A Science of Certification

- Certification is ultimately a judgment
- But the judgment should be based on rational argument supported by adequate explicit and credible evidence
- A Science of Certification would be about ways to develop that argument and evidence
- Favor goal-based over standards-based approaches
  - At the very least, expose and examine the claims, arguments and assumptions implicit in standards
- Be wary of demands for more and more evidence, with implicit appeal to diversity and independence
  - Instead favor explicit multi-legged cases
- Use formal ("machinable") design descriptions
  - Can then use automated analysis methods

Summary

- We already do component-based design
- We urgently need methods for component-based certification
  - Compositional certification
- Crucially dependent on architectural frameworks that eliminate unintended component interactions through shared resources
  - Partitioning in avionics, separation in MILS security
- Need a scientific basis for certification that deals comprehensively with these issues
- Goal-based certification provides the best foundation for this
- A community effort is needed to move this forward