

Using the trees to find the forest

Trustworthy computing as a systems-level issue

John James

John-James@usma.edu

Frank Mabry

Frank.Mabry@usma.edu

Trustworthy Computing Efforts

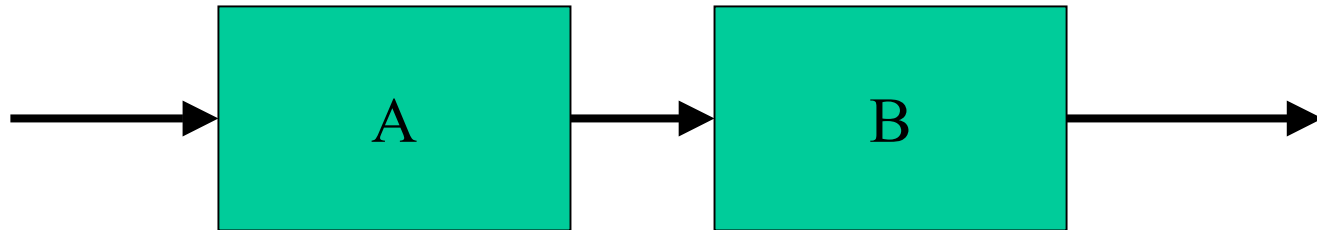
- NASA-Langley Goals (Chuck Meissner mid-80s):
 - Provably correct computer
 - Provably correct algorithms
- DARPA DSSA Program (Erik Mettala/Barry Boehm '90-'95):
 - Enable reuse through composition of components
 - Domain-specific models and architectures
 - Architectural Definition Languages (ADL)
 - Interface Definition Languages (IDL)
- NIST ATP on Component-Based Software (Barbara Cuthill '95-'99)
 - Foster an industry based on composition of components
 - Productization of components
- DARPA project on Trustworthy Computing (Anup Ghosh '03)
 - Creating and maintaining trustworthy components
 - Related to reliability and software safety issues

Trustworthy Computing Challenges

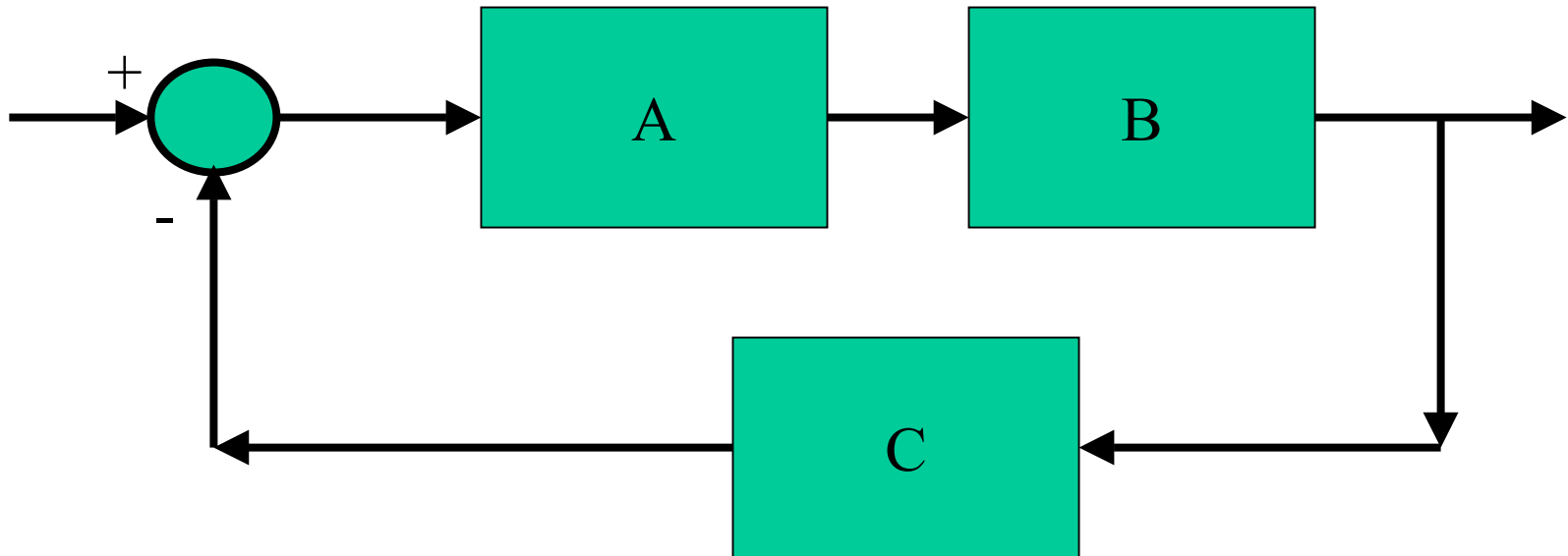
- We build more complex systems than we can understand
 - We do not have accurate domain models for many systems
 - We do not know the failure modes of many existing systems
- We “move the aiming point” of existing systems:
 - Verification – “Am I building the system right?”
 - Validation – “Am I building the right system?”
 - Accreditation – “Do I *trust* that the system as built meets the requirements as stated?”
- We are focused on the trustworthiness of components
 - “Denial of service” attacks do not affect enterprise processes
 - Compromise of individual system components does not affect enterprise processes

Open-Loop and Closed-Loop Systems

Open-Loop (Model-View-Controller/Publish-And-Subscribe)



Closed-Loop (components are *not* independent)



One Approach: Intermediate levels of abstraction

- No ‘silver bullets’ are available for solving the problem of trusting that the systems we build are working properly
- Focus on the system *purpose* (i.e. enable some enterprise process(es))
- System partitioning and evaluations help:
 - Maintain domain models
 - Use intermediate levels of aggregation
 - Use multiple sources of evaluation:
 - Network monitoring (i.e. intrusion detection and reaction)
 - Signals intelligence (i.e. jamming or signals intercept)
 - Human intelligence (i.e. greatest threat is internal)

Back-up

Attack Plans

Attack manufacturing

- Buffer overflow exploit
- Root kit for admin privileges
- Expand to other network components
- Modify part orders
- Modify shipping times
- Modify delivery locations
- Modify manufacturing tolerances
- Modify interface requirements
- Indicate errors arise from faulty received parts

Attack Control

- Insider access from gambling debts
- Insider provides root access
- Expand to other network components
- Modify control plans
- Modify subordinate unit tasks
- Modify subordinate unit commanded locations
- Modify status reports to higher headquarters
- Modify timing of operations
- Indicate errors arise from faulty orders

Attack Transportation

- Buffer overflow exploit
- Root kit for admin privileges
- Expand to other network components
- Modify delivery schedules
- Modify available units
- Modify indicated truck capacities
- Modify pick up locations
- Modify delivery locations
- Indicate errors arise from faulty requests

Some Environments for complex systems

Architecture Description Language (ADL)

Interface Definition Language (IDL)

- Mentor Graphics : VHDL-AMS as an ADL
- Mathworks: Matlab-Simulink (Hybrid state as an ADL)
- U.C. Berkeley: Ptolemy (Hybrid state as an ADL)

Real Time Support

- POSIX.13 - Shorthand for the POSIX Realtime Application Support (AEP), IEEE Std.1003.13-1998.
- DARPA Open Control Platform (OCP) – based on a lightweight ORB
- Artisan – Real time UML (finite state only)