

Specifying and Certifying Information Flow Properties in MILS Systems

*SAnToS Laboratory
Computing and Information Sciences
Kansas State University*

John Hatcliff, Professor

Torben Amtoft, Associate Professor

Anindya Banerjee, Professor

Simon Ou, Assistant Professor

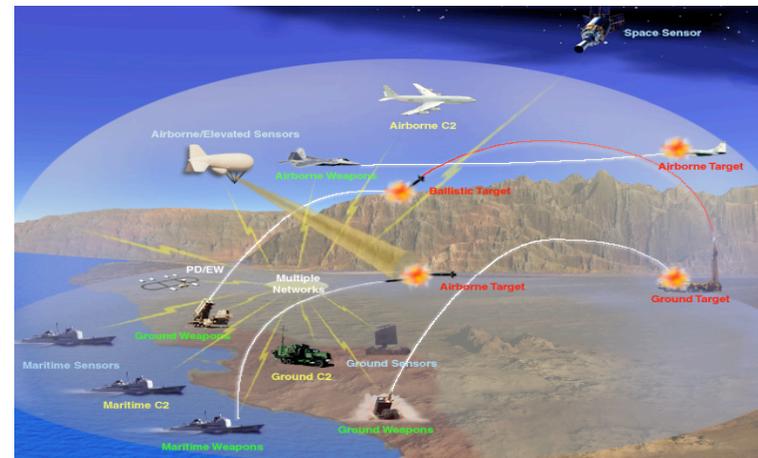
Robby, Assistant Professor

Support: *Air Force Office of Scientific Research, Rockwell Collins, National Science Foundation*

DoD Relevant Contexts

Supporting Security in Information Centric Warfare

- Access to information to warfighters across all levels of command and across forces (including, *e.g.*, NATO allies)
- Challenges
 - Data from a variety of sources with different security levels
 - weather, GIS, GPS of forces and assets, target info, battle plans, etc.
 - Data may be blended, integrated, and transformed (*e.g.*, JBI Fuselets) so that the ultimate source of the data is blurred
- Tension between
 - providing aggressive information flow
 - preventing access to unauthorized parties



Challenges...

- What architectures should be used?
- *Many* contractors -- how do you get them to agree on interfaces?
- How do you develop a market for components?
- How to plug in COTS components?
- How do you establish end-to-end assurance?

Today's Reality



CH 53K -- Chinook

- On board computer system manages mission/tactical data (high security) as well as maintenance data (low security)
- Due to the inability to implement/certify MLS systems, all data is classified as high security

- Ground crew does not have high security clearance
- Currently, pilot must manually step through all data to scrub it, and make maintenance data available to grounds crew

Chinook Pilot



*labor intensive
manual scrub of data*



Grounds Crew

Colleagues at Rockwell Collins are contracted to provide an automated solution

Previous Approaches

Monolithic Security Kernels

- All security policy enforcement performed by the security kernel
 - based on Bell-LaPadula / Trusted Computing base model
- As security policy became more complex:
 - Code grew in security kernel
 - Certification efforts become unmanageable
 - Evaluatability of kernel decreased
 - Maintainability of kernel code decreased
 - Policy decisions were based upon incomplete/unauthenticated information



(Source: MILS/MLS Architecture for Deeply Embedded Systems, Dransfield, et al.)

MILS Goals

Multiple Independent Levels of Security (MILS) Architecture

Combine best of Security and Safety Technology

FAA DO-178B Level A Safety Technology

Common Criteria EAL 7 Security Technology

to enable the provision of

MILS/MSLS/MLS Web and Network Services

to the

Deeply Embedded

Real Time

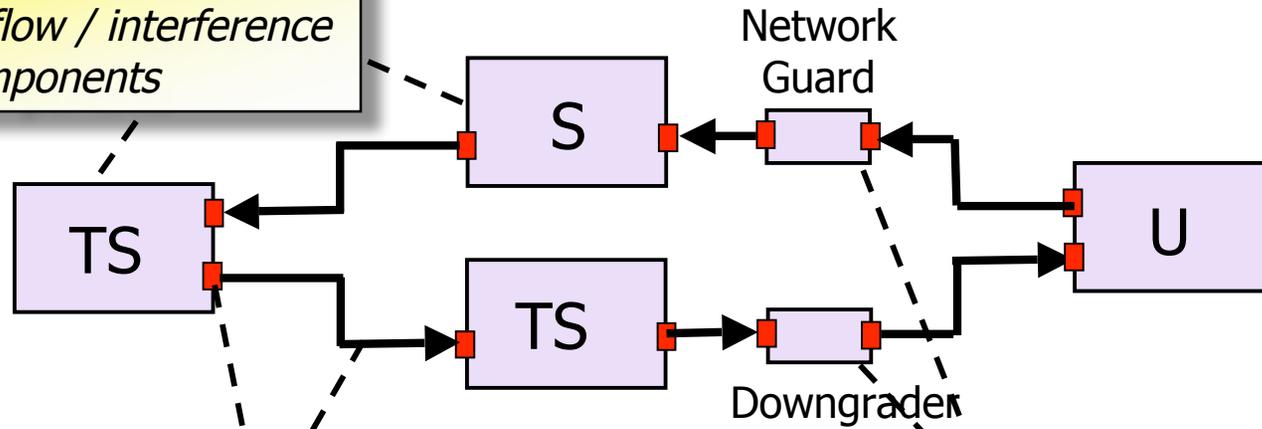
High Assurance

Weapons, Communication, and C2 Platforms

(Source: MILS/MLS Architecture for Deeply Embedded Systems, Dransfield, et al.)

MILS Principles

Separation guarantees no information flow / interference between components

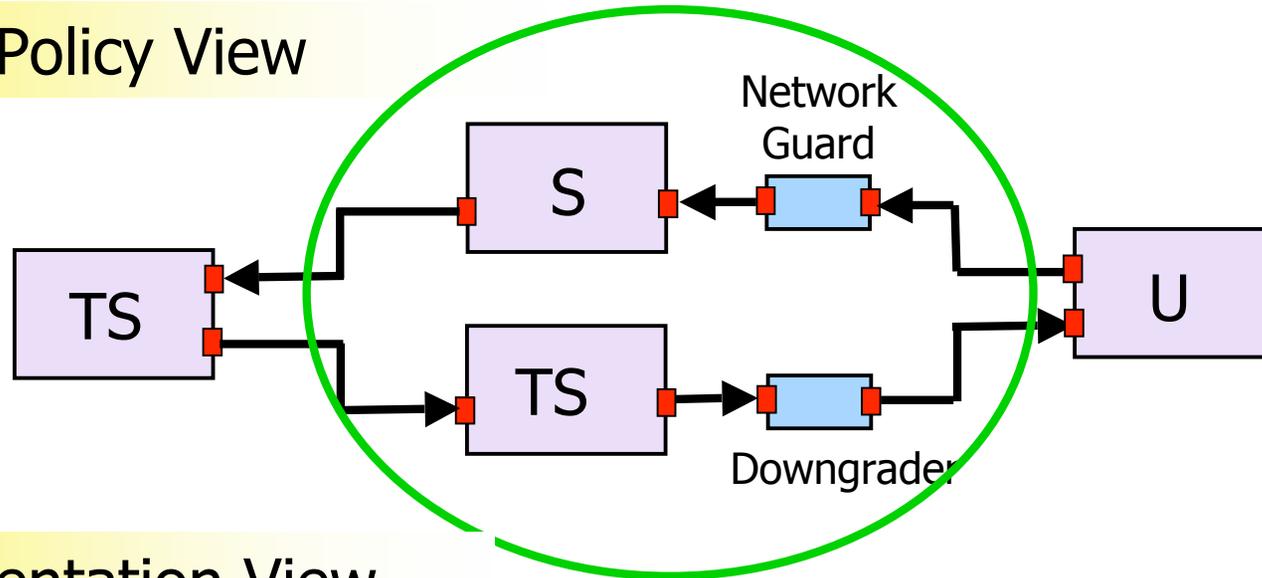


*Only interaction occurs through explicitly defined **ports** and unidirectional **channels***

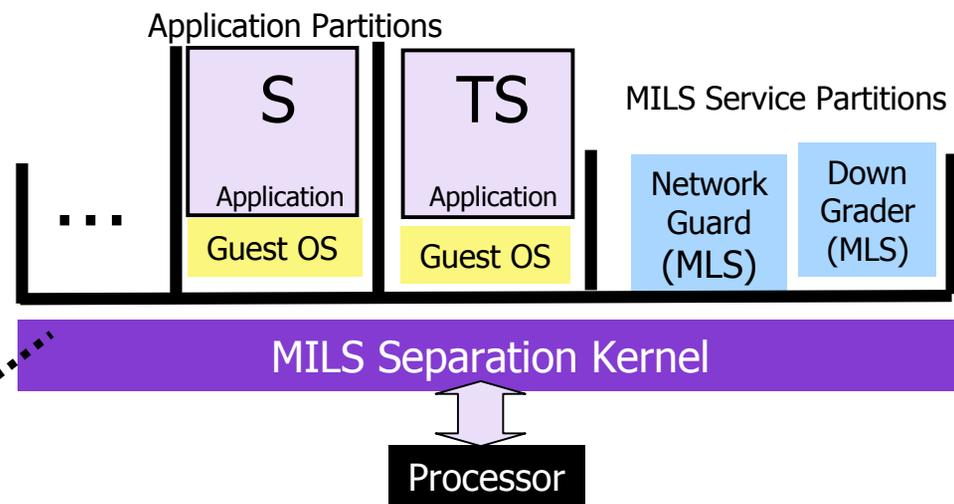
Infrastructure components mediate cross-domain flow. Built & verified once; reused in many applications.

MILS Principles

Logical/Policy View



Implementation View

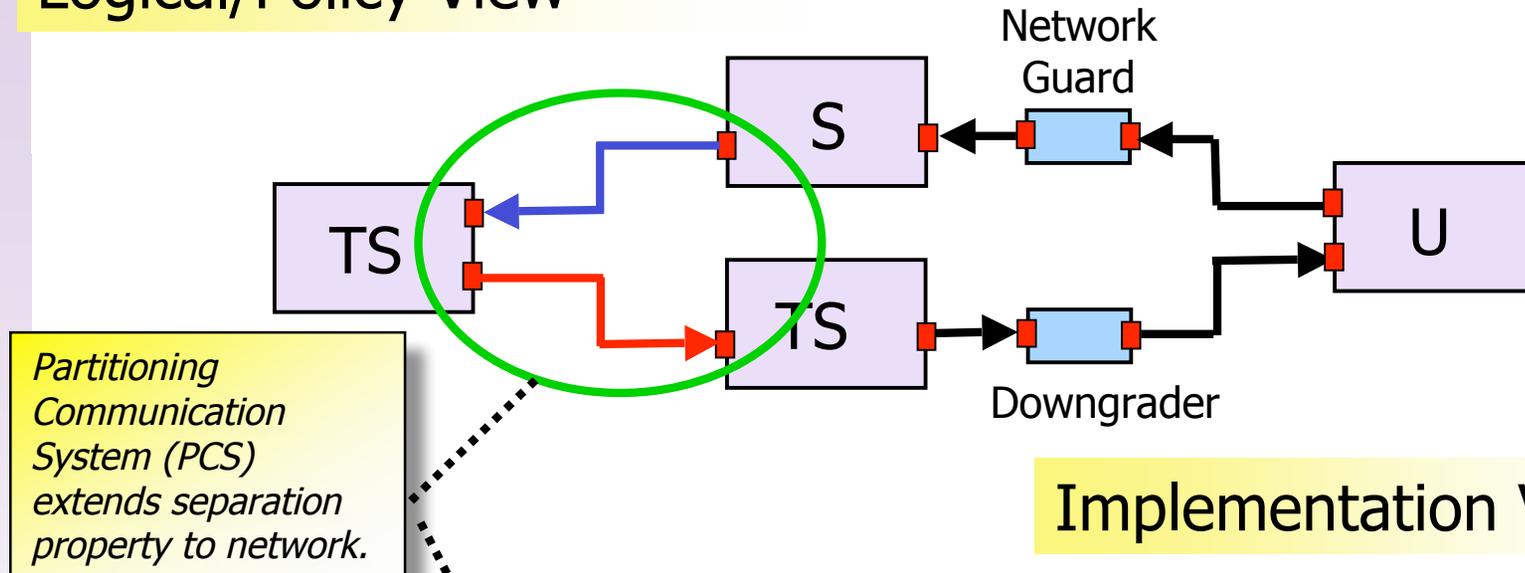


*Virtualization technology in **separation kernel** guarantees separation between partitions and confines information flow to specified channels.*

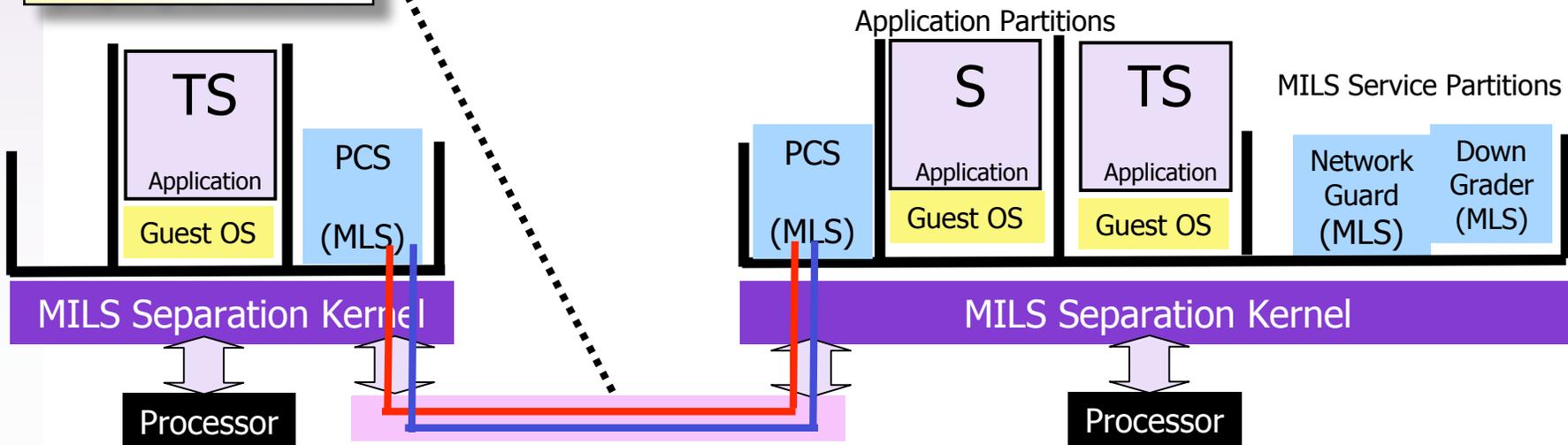
(Source: "Two-level view" due to John Rushby, SRI)

MILS Principles

Logical/Policy View



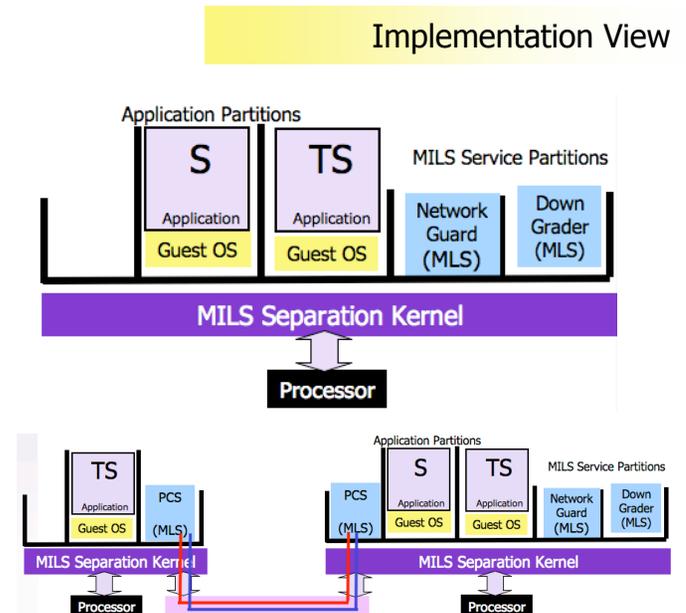
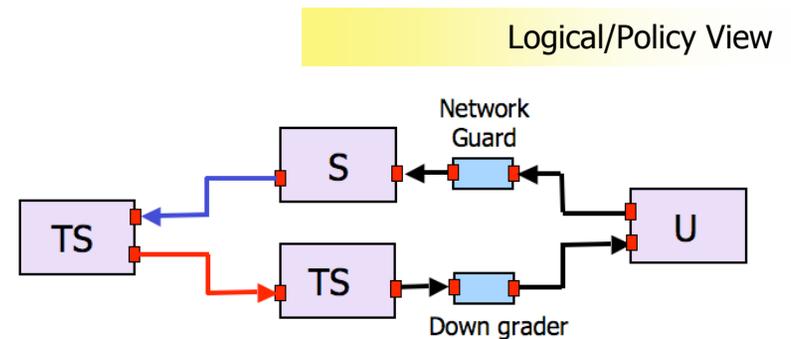
Implementation View



MILS Goals

In summary, how does MILS achieve its goals?

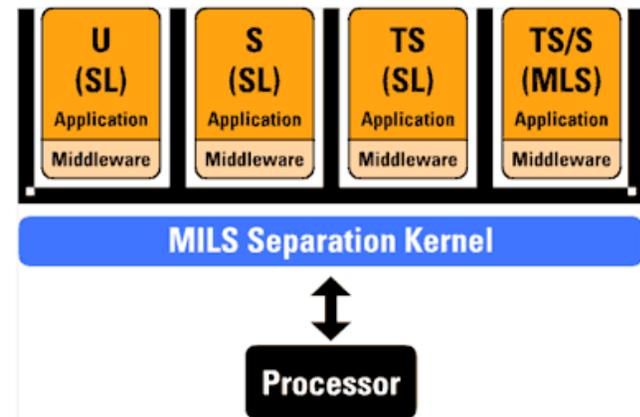
- By building/enforcing security policies based on...
 - Data Isolation
 - Information Flow Control
 - Periods Processing
 - Damage Limitation
- ...in a...
 - microprocessor centric manner (e.g., MILS separation kernel)
 - network centric manner (e.g., MILS middleware -- partitioning communication system, etc.)



MILS Architecture

MILS Security Mechanisms are “neat”

- **Non-bypassable.** Security functions cannot be circumvented.
- **Evaluatable.** Security functions are small and simple enough to enable rigorous proof of correctness through mathematical verification.
- **Always Invoked.** Security functions are invoked each and every time.
- **Tamperproof.** Security functions and their data cannot be modified without authorization, either by subversive or poorly written code.



- Factored out and detangled security concerns into smaller components -- *dramatically improves the ability to validate*
- Separation kernel typically must be verified to...
 - Common Criteria EAL 6 or EAL 7
 - Director of Central Intelligence Directive 6/3
- Requires the use of formal mathematical/logical methods

MILS Participants

Primary Sponsors

- AFRL, NSA

Separation Kernels

- AAMP7 (Rockwell Collins)
- Integrity 178B (GreenHills)
- LynxOS (LynxWorks)

Middleware and Services

- PCS (Objective Interface Systems)
- Crypto Engines (Rockwell Collins)
- Network Guard Product Line (Rockwell Collins)

(Source: MILS/MLS Architecture for Deeply Embedded Systems, Dransfield, et al.)

DoD Programs

Weapons Platforms

**F-22, C-130, UCAV,
F35 (JSF), LW,
Virginia Class,**

**Lockheed-Martin, Boeing,
GD, Raytheon, . . .**

Communications Platforms

**JTRS, Crypto MOD,
AIM, PEIP, JANIS, . . .**

**Boeing, BAE, GDDS, L-3,
NRL, Rockwell, Harris, ...**

Command and Control

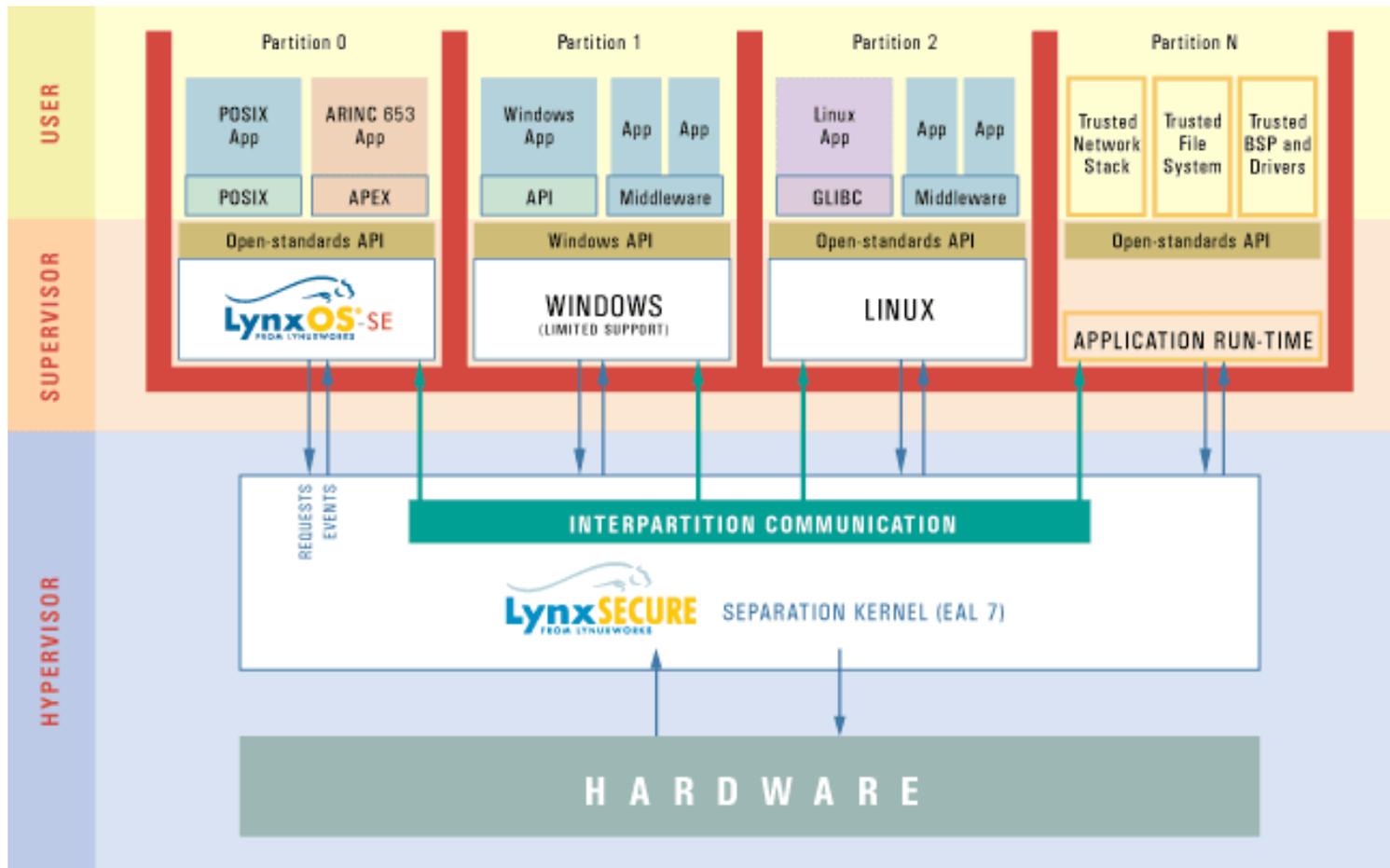
DDX, AEGIS, FCS

Boeing, Lockheed, Raytheon

(Source: MILS/MLS Architecture for Deeply Embedded Systems, Dransfield, et al.)

Example

LynxWorks LynxSecure Separation Kernel

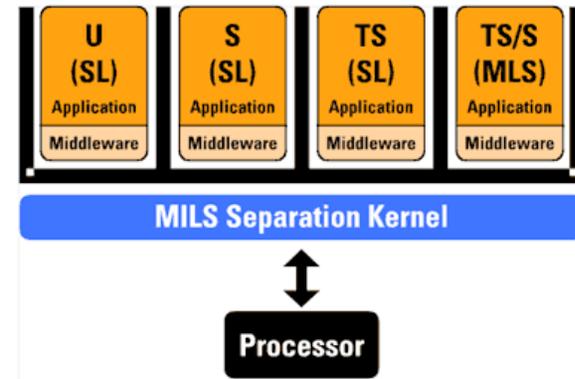


(Source: LynxWorks marketing material)

Example

Rockwell Collins -- Security Certification (NSA)

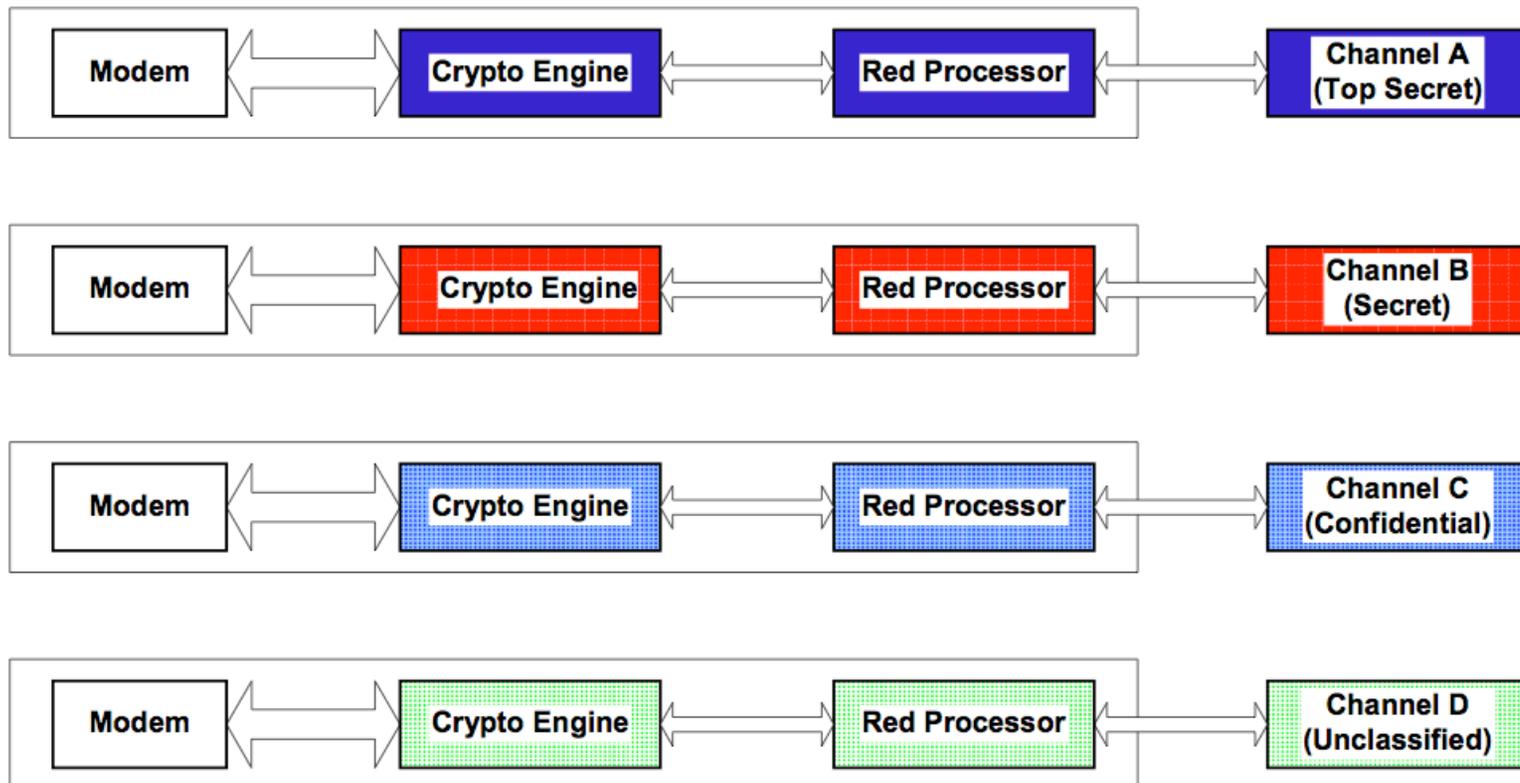
- Rockwell Collins has developed the AAMP7 processor that provides a *separation kernel*
- Separation policy of the AAMP7 has received NSA certification
 - verification using ACL2 theorem-prover
 - the only such certified processor
- Rockwell Collins has also carried out software separation kernel certification -- Green Hills INTEGRITY 178B RTOS (OS for the F-35 Joint Strike Fighter)



Rockwell
Collins

Example

Overcoming stove-piped architectures...

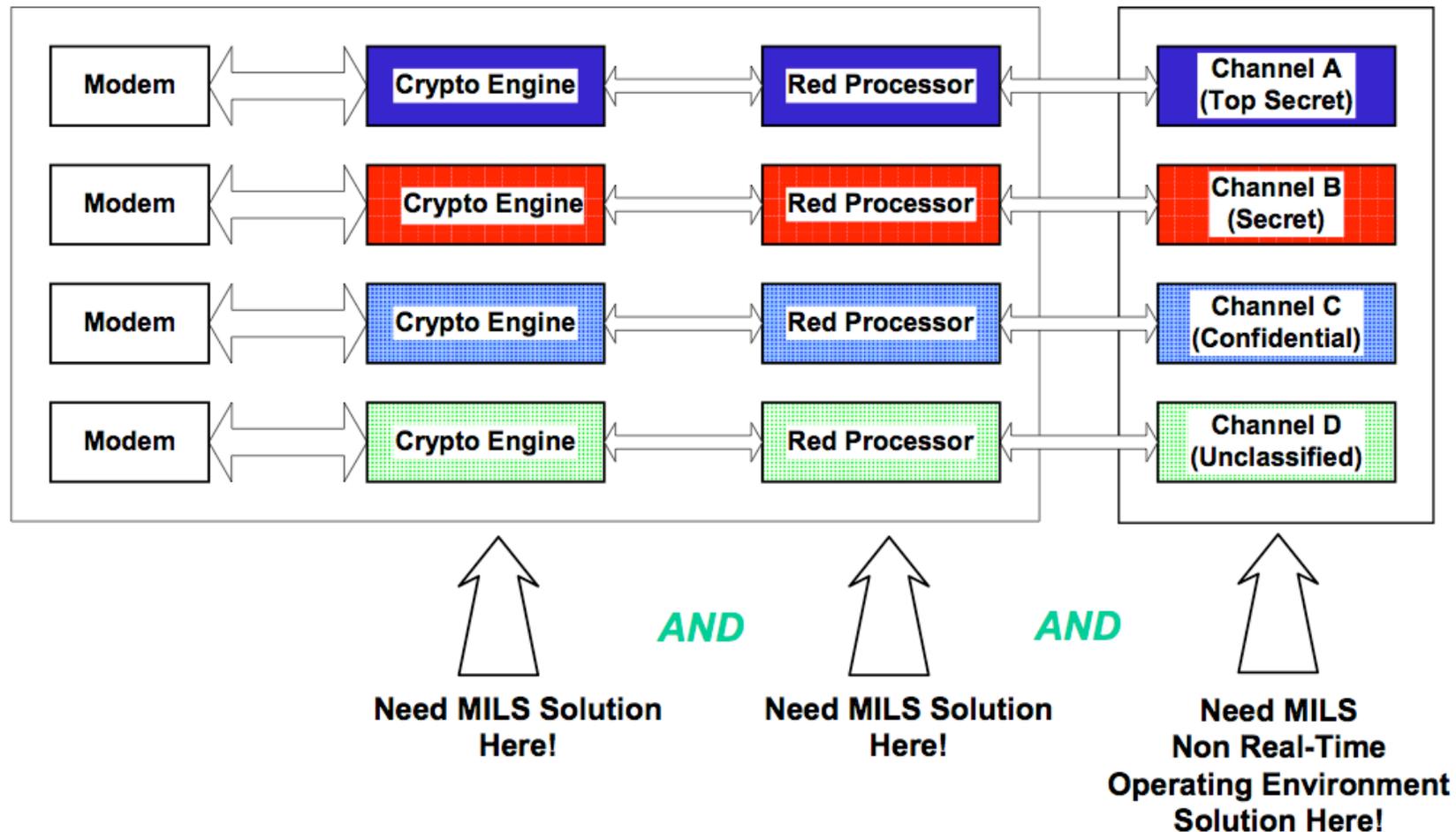


This Is Current Stovepipe Technology That Is Expensive And Inflexible

(Source: MILS/MLS Architecture for Deeply Embedded Systems, Dransfield, et al.)

Example

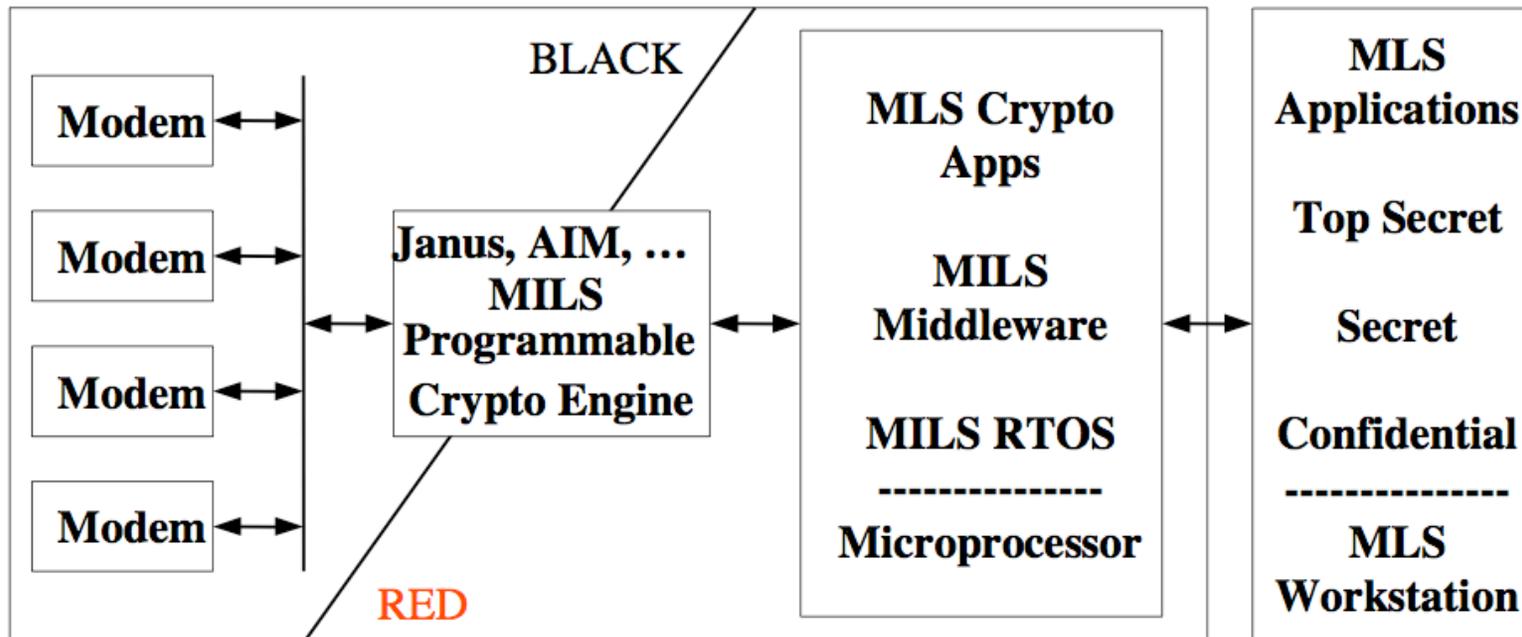
Overcoming stove-piped architectures...



(Source: MILS/MLS Architecture for Deeply Embedded Systems, Dransfield, et al.)

Example

Systematically applying MILS solutions...



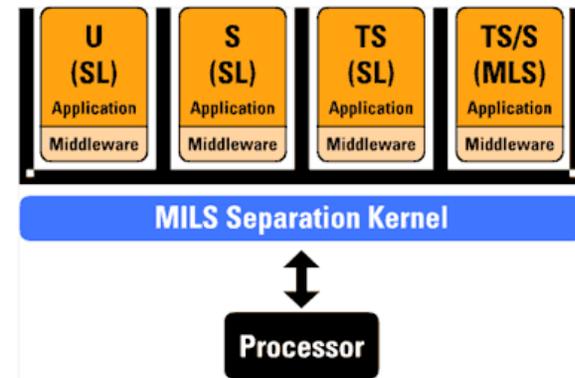
- Example MILS Applications
 - Communications Systems (JTRS, TTNT, TCS SATCOM)
 - Multi-channel, low power, low cost, flexible, open systems architectures
 - Precision Guidance/Navigation (GPS/SASSM, MUE)
 - Highly integrated, low power, low cost (system on a chip)
 - System & Platform Integration (FCS, Flight2, E6)
 - Integrated data management/fusion w/ information assurance

(**Source:** MILS/MLS Architecture for Deeply Embedded Systems, Dransfield, et al.)

Information Assurance Products

Rockwell Collins -- Multiple Products

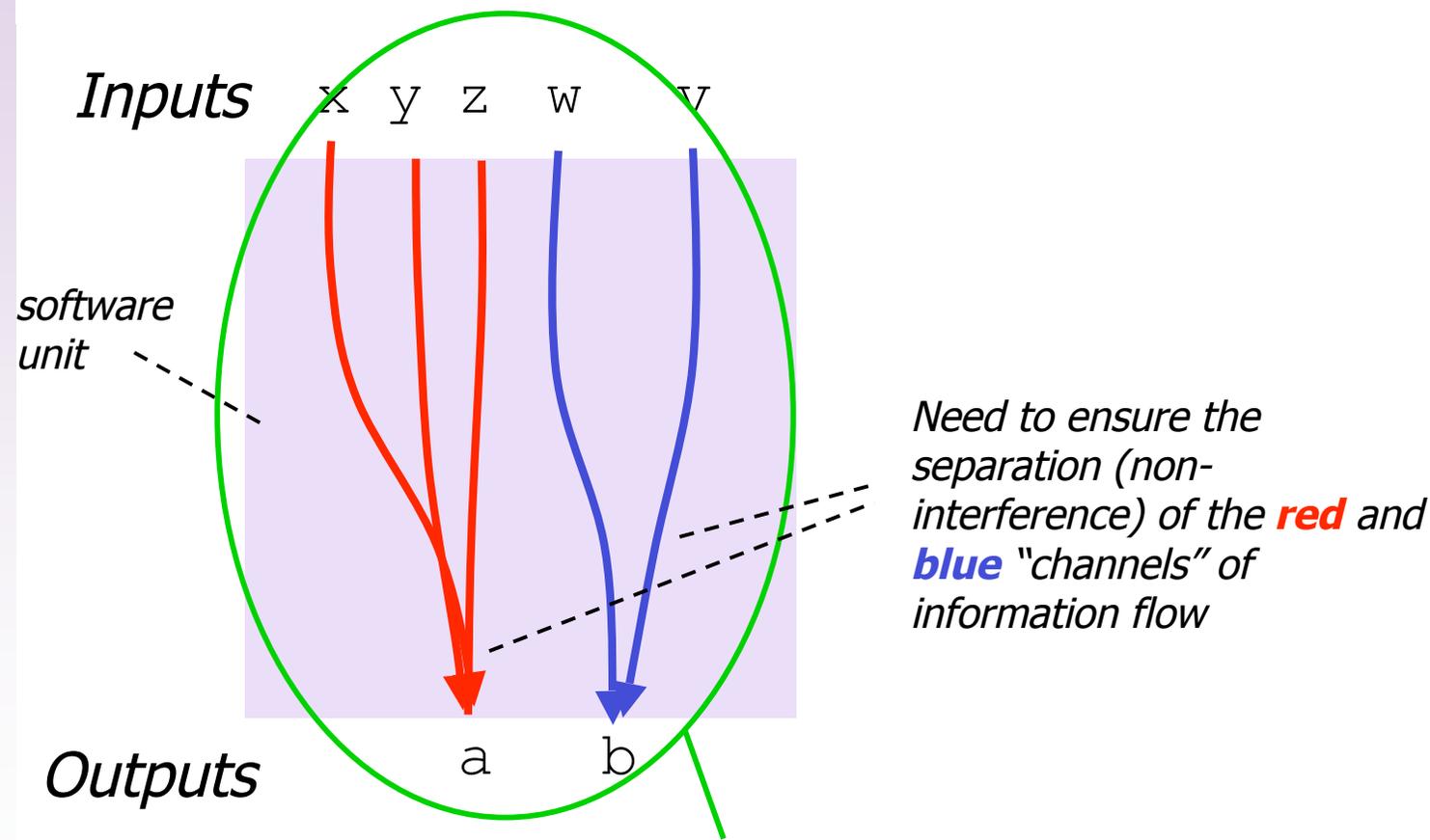
- Rockwell Collins is developing multiple information assurance on top of AAMP7 following the MILS architecture
 - Janus Crypto-Engine
 - High-Assurance Network Guard
 - product line
 - Military GPS with MLS facilities
- 200+ developers
- KSU SAnToS is working with Rockwell Collins researchers and product groups to form a vision of what MILS-specific architecture, development, and certification tools should look like
- *Almost no MILS-specific development tools exist! Solutions will likely spill over to other NSA-funded efforts*



Rockwell
Collins

Basic Issues

Specification and checking of information flows between interface inputs and outputs -- the basis of MILS security contracts



This graph (relation) between inputs and outputs is often the basis of the formal MILS security policy

Information Flow Contracts

SPARK provides "information flow" contracts that describe how a procedure causes information to flow from one variable to another



```
procedure Operate;  
--# global out KeyStore.RotorValue, Encrypted;  
--# in out KeyStore.SymmetricKey;  
--# in Clear;
```

"out"-only variables are not read

"in/out"- both read and written

"in"-only variables are not written

Start by identifying procedure inputs and outputs (both parameters and any globals used) -- provides "frame conditions"

Information Flow Contracts

SPARK provides "information flow" contracts that describe how a procedure causes information to flow from one variable to another



```
procedure Operate;  
--# global out KeyStore.RotorValue, Encrypted;  
--# in out KeyStore.SymmetricKey;  
--# in Clear;  
--# derives  
--#   KeyStore.SymmetricKey, KeyStore.RotorValue  
--# from  
--#   KeyStore.SymmetricKey  
--# &  
--#   Encrypted  
--# from  
--#   Clear, KeyStore.SymmetricKey  
--# ;
```

Spark information flow annotations...

Clear SymmetricKey



Information flows from Clear, KeyStore.SymmetricKey to Encrypted

MILS IDE Support for SPARK

Enhancements/additions to Praxis SPARK that specifically target information assurance applications

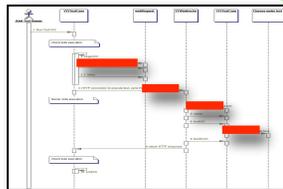


Developers

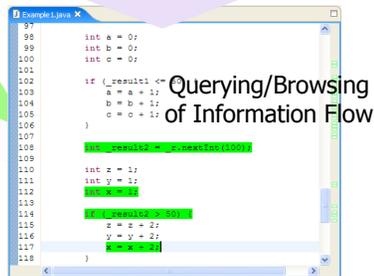
Information Flow Contracts

```
# derives
--# KeyStore.SymmetricKey, KeyStore.RotorValue
--# from
--# KeyStore.SymmetricKey
--# Encrypted
--# from
--# Clear, KeyStore.SymmetricKey
--# ;
```

Checking / Inference



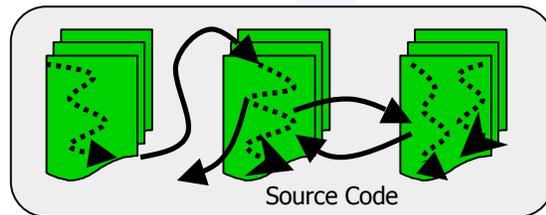
Information Flow Visualization

A screenshot of the SPARK Eclipse IDE. The main window shows a code editor with SPARK code. The code includes variable declarations and conditional logic. Some lines are highlighted in green, indicating they are the focus of the current view or analysis.

```
Example1.java X
97
98 int a = 0;
99 int b = 0;
100 int c = 0;
101
102 if (_result1 <=
103     a = a + 1;
104     b = b + 1;
105     c = c + 1;
106 }
107
108
109
110 int z = 1;
111 int y = 1;
112
113
114
115 z = z + 2;
116 y = y + 2;
117
118
```

Querying/Browsing
of Information Flow

SPARK Eclipse IDE



Source Code

Integrated development environments that help engineers specify, visualize, implement, and verify information flow policies

MILS IDE Support for SPARK

Enhancements/additions to Praxis SPARK that specifically target information assurance applications



Developers

Information Flow Contracts

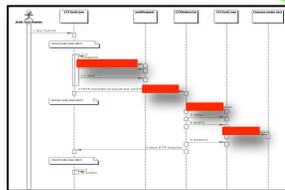
```
## derives
--# KeyStore.SymmetricKey, KeyStore.RotorValue
--# from
--# KeyStore.SymmetricKey
--# &
--# Encrypted
--# from
--# Clear, KeyStore.SymmetricKey
--# }
```

Checking / Inference

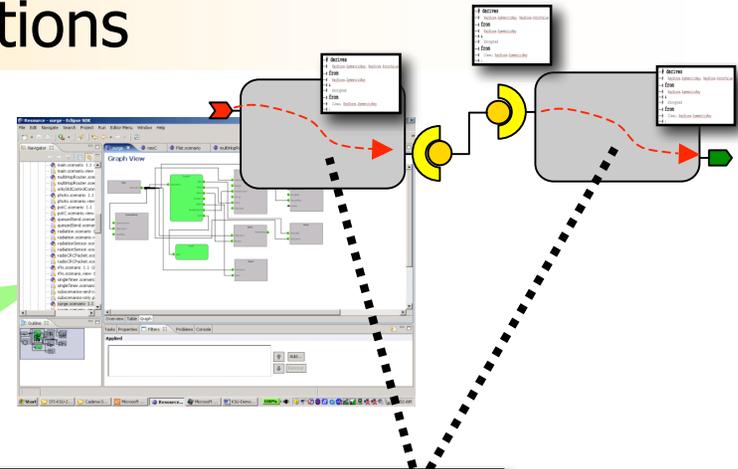
```
Example.java
97
98 int a = 0;
99 int b = 0;
100 int c = 0;
101
102 if (_result1 <= 0)
103   a = a + 1;
104   b = b + 1;
105   c = c + 1;
106
107
108
109
110 int e = 1;
111 int y = 2;
112 x = x + 1;
113
114
115 y = y + 2;
116
117
118
```

Querying/Browsing
of Information Flow

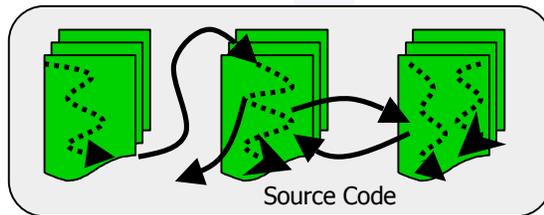
SPARK Eclipse IDE



Information Flow
Visualization



*Specify & query
information flows at the
architecture level*

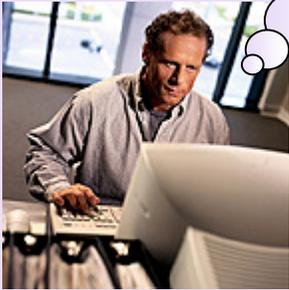


Source Code

*Provide modeling tools to specify and
evaluate MILS policy architecture*

Basic Info Flow Browsing

Where does information from parameter I3 flow?

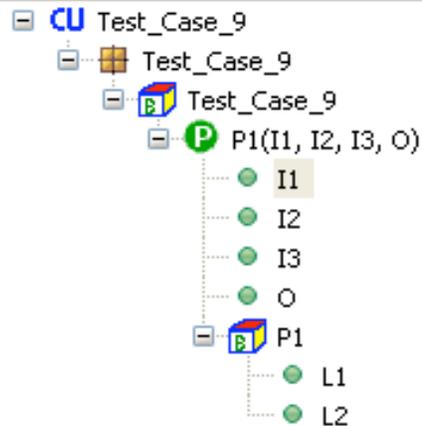


Developer highlights I3 and selects "Forward Flow" visualization.

```
package body Test_Case_9 is
  procedure P1 (I1, I2, I3 : in Integer; O : out Integer)
    --# derives O from I1, I2, I3;
  is
    L1, L2 : Integer;
  begin
    L1 := I1;
    if L1 > 5
    then
      L2 := 6;
    else
      if I2 < 0
      then
        L2 := I3;
      else
        L2 := 7;
      end if;
    end if;
    O := L2;
  end P1;
end Test_Case_9;
```

SpAda automatically marks the lines and output vars into which information from I3 flows.

Outline



Tasks Console Generator Results

Ada Build
gnatmake: objects up to date.

Unable to locate d... selected element.

Writable

Insert

3:29

Info Flow Contract (Derives Clause) Autogeneration

What should my info flow contract be for output parameter O1?



```
body Test_Case_10 is
  procedure P1 (I1, I2, I3 : in Integer; O1, O2 : out Integer)
  L1, L2 : Integer;
begin
  L2 := 5;
  L1 := I3;
  O2 := L1;
  L1 := I2; -- kill I3
  if L1 > 5
  then
    if I1 < 0
    then
      L2 := 8;
    else
      L2 := 7;
    end if;
  end if;
  O1 := L2;
end P1;
end Test_Case_10;
```

SpAda automatically infers an information flow contract for a given subprogram implementation

Outline

- CU Test_Case_10
 - Test_Case_10
 - Test_Case_10
 - P1(I1, I2, I3, O1, O2)
 - I1
 - I2
 - I3
 - O1
 - O2
 - P1
 - L1
 - L2

Generator Results

Results:

- DerivesResult-1217959368273
- DerivesResult-1218746172721
- DerivesResult-1219347293635
 - O1

Generated Code:

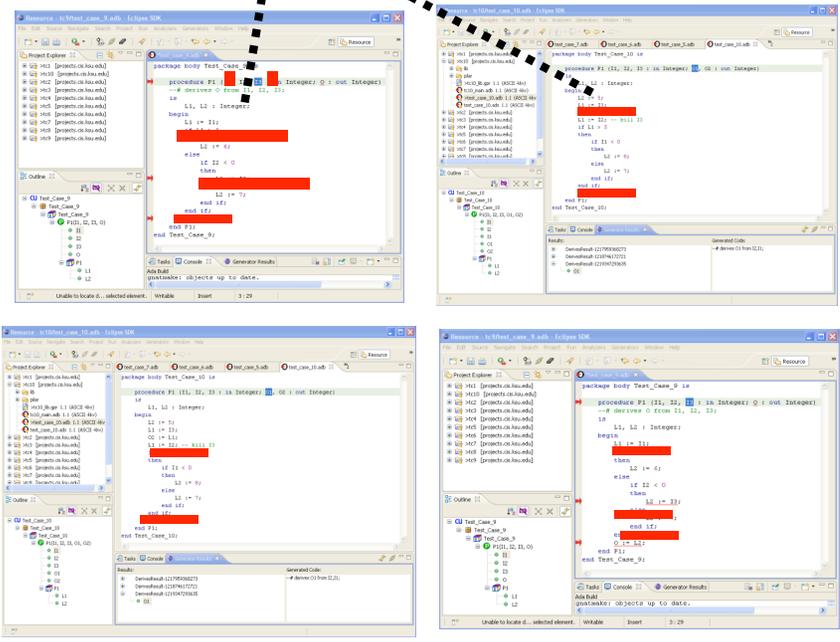
```
--# derives O1 from I2,I1;
```

MLS Flow Markup

Show me all variables and statements in my program that manipulate top secret data



Given an initial tagging of input variables according to security level, SpAda propagates and marks up source code to indicate MLS level of statement/variable



Principals	Security Lattice
A1	Top Secret
A2	High
A3	Low
...	
An	Public

MLS security lattice with color mark-up tags

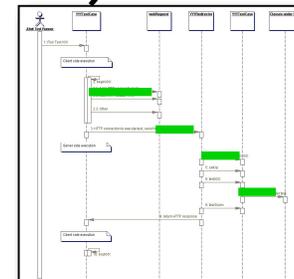
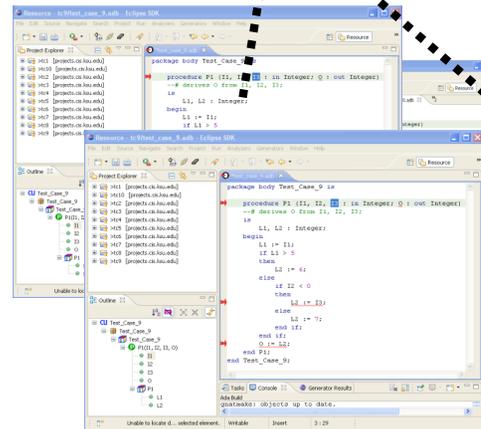
(not yet implemented -- target Dec 2008)

End-to-End Flow Visualization



Show me a sequence chart illustrate the call tree for procedure P and mark instances of **Public** data flow in parameter passing.

SpAda automatically generations visualization of a method call graph along with markups of flows of different MLS categories.



Sequence chart showing procedure call chain with markup of calls that pass **Public** data

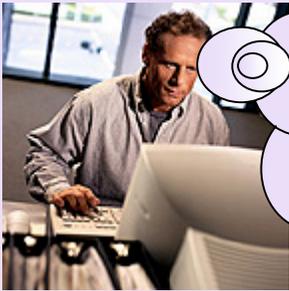
Principals Security Lattice

A1	Top Secret	█
A2	High	█
A3	Low	█
...		
An	Public	█

MLS security lattice with color mark-up tags

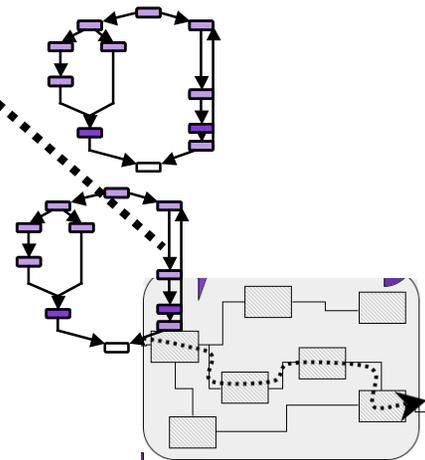
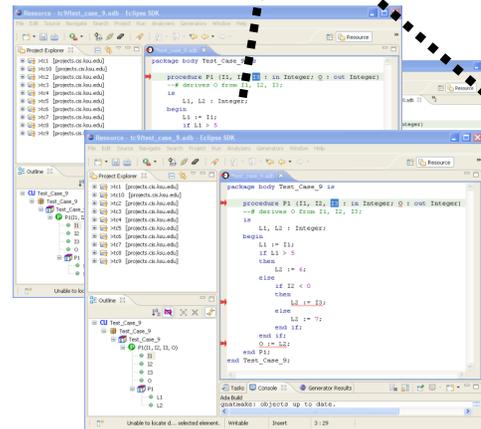
(not yet implemented -- target Dec 2008)

Model-checking of path properties through information flow graphs



Is it the case that all flows from Partition P1 to Partition P5 pass through guard procedure G3?

SpAda extracts information flow graphs from source code, and then uses model checking to check path properties through information flow graphs.



$$\begin{aligned}
 s_0 &= \perp \\
 s_1 &= \beta \vee (\alpha \wedge \text{EX} \overset{s_0}{\perp}) = \beta \\
 s_2 &= \beta \vee (\alpha \wedge \text{EX} \overset{s_1}{\beta}) \\
 s_3 &= \beta \vee (\alpha \wedge \text{EX} \overset{s_2}{(\beta \vee (\alpha \wedge \text{EX} \beta))}) \\
 &\dots
 \end{aligned}$$

High-level information flow path requirements formalized in terms of temporal logic

(not yet implemented -- target May 2009)

Conclusion

- MILS is a security architecture that...
 - provides a foundation for secure system based on notions of *separation*
 - factors out security functionality into composable units that are easier to evaluate/certify
- Security certification frameworks such as Common Criteria are being used to encourage a commodity market of MILS components
- KSU SAnToS + Rockwell Collins ATC research aims to...
 - develop mathematics/logic models of information flow
 - provide integrated tools that directly target MILS development
 - architecture modeling and information flow querying
 - code level specification/checking info flow browsing
 - reduce development time/costs, increase confidence

For More Information...



SAnToS Laboratory,
Kansas State University

<http://people.cis.ksu.edu/~hatcliff>

Please contact me if you are interested in technical papers for SPARK Ada & Java

IATAC SOAR



Information Assurance Technology Analysis Center

Software Security Assurance
(July 31, 2007)

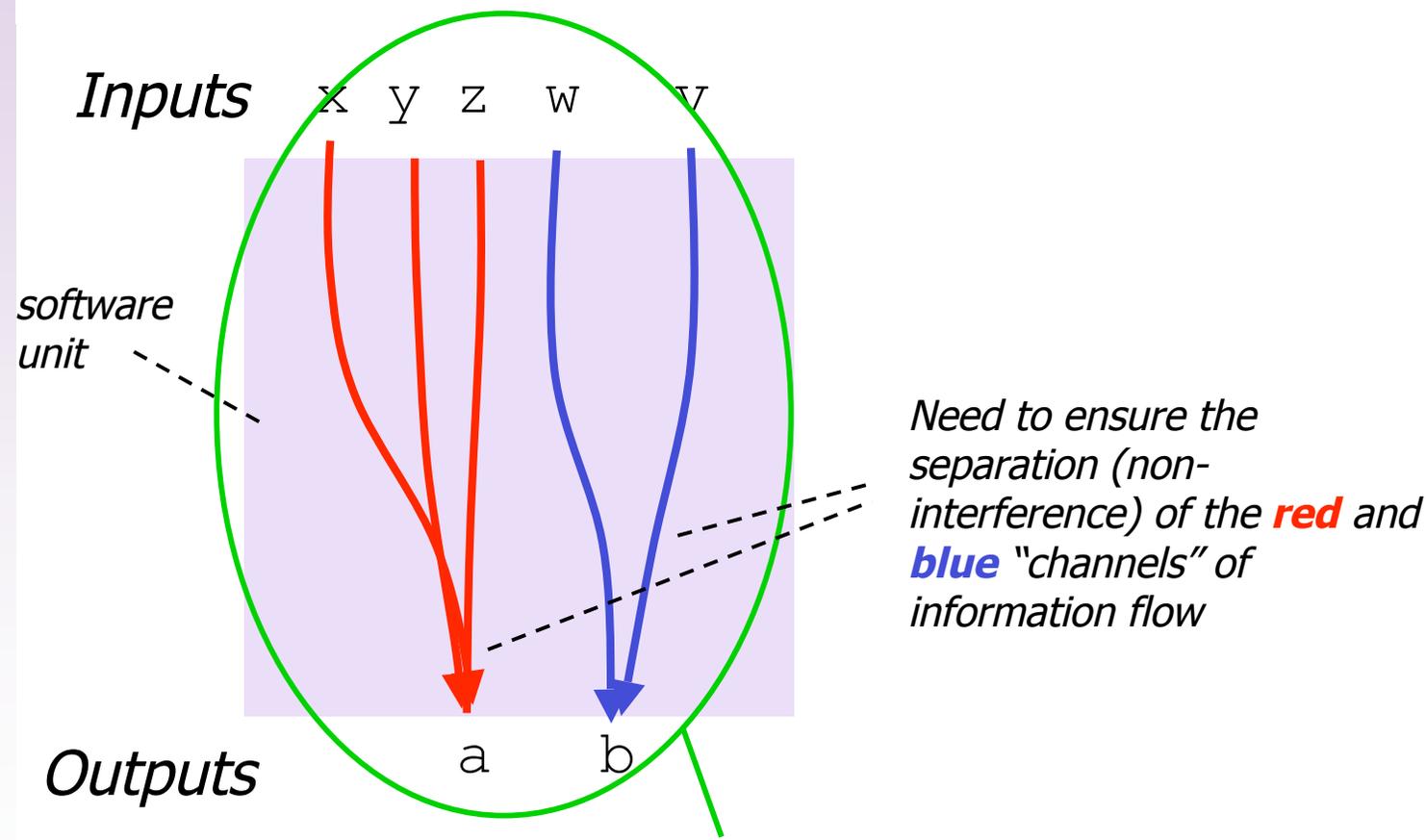
“Software security requires security to be seen as a **critical property of the software itself** — a property that is best assured if it is **specified from the very beginning of the software’s development process**. Software security assurance is addressed **holistically and systematically**, in the same way as quality and safety”.

Implies...

- Security specifications integrated throughout designs, architecture, code
- Checked/cross-checked throughout development
- Development tools (editors, compilers, debuggers, code query tools, static checkers, testing tools) extended to be aware of MILS/MLS security policies

Basic Issues

Specification and checking of information flows between interface inputs and outputs -- the basis of MILS security contracts

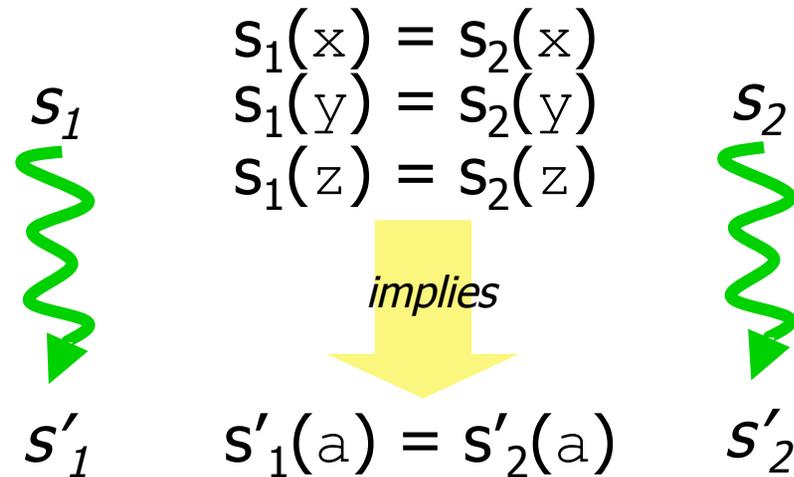
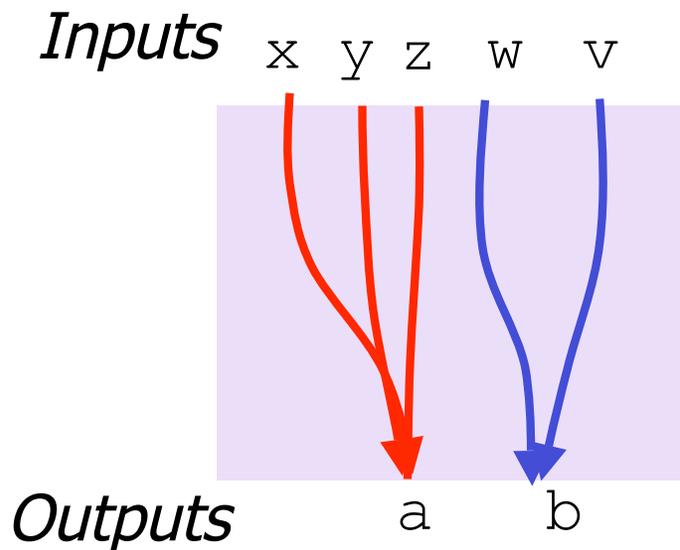


This graph (relation) between inputs and outputs is often the basis of the formal MILS security policy

Non-interference

The classical notion of *non-interference* (Goguen & Meseguer) provides semantic foundation for describing the required separation...

Proving "nothing interferes with red channel"
(i.e., a only depends on x, y and z)...



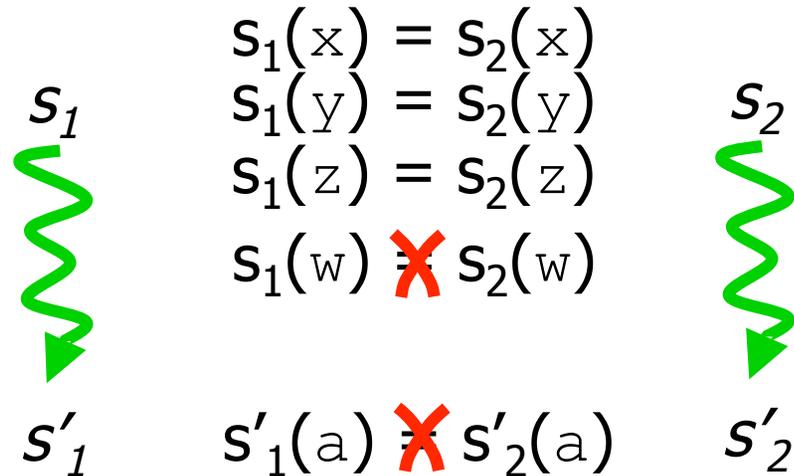
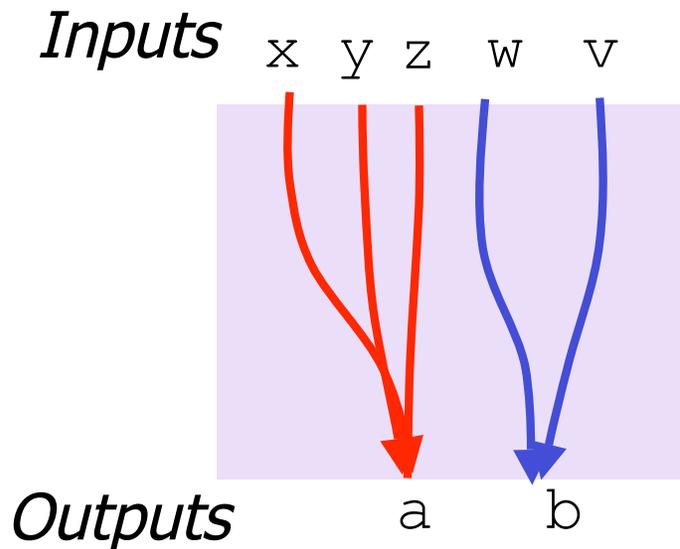
Non-interference theorem:

For any two executions with initial states s_1, s_2 ,
if s_1 and s_2 both **agree** on the values of x, y and z ,
then final states s'_1 and s'_2 **agree** on the value of a

Non-interference

The classical notion of *non-interference* provides semantic foundation for describing the required separation...

*Proving "nothing interferes with red channel"
(i.e., a only depends on x, y and z)...*

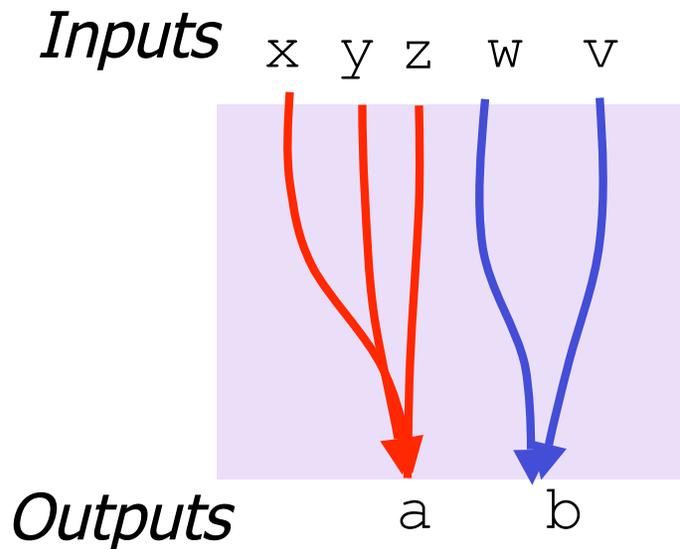


Intuition

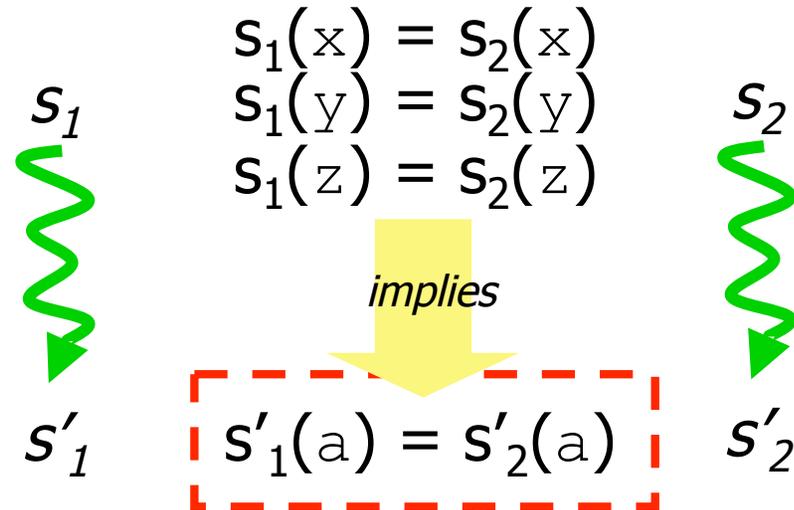
If s'_1 and s'_2 do not **agree** on the value of a
there must be some other variable upon which a depends, e.g., w
that had different values in s_1 and s_2

Non-interference

How can we lift reasoning about non-interference to code-level annotations?



Proving "nothing interferes with red channel"
(i.e., a only depends on x, y and z)...



Non-interference theorem:

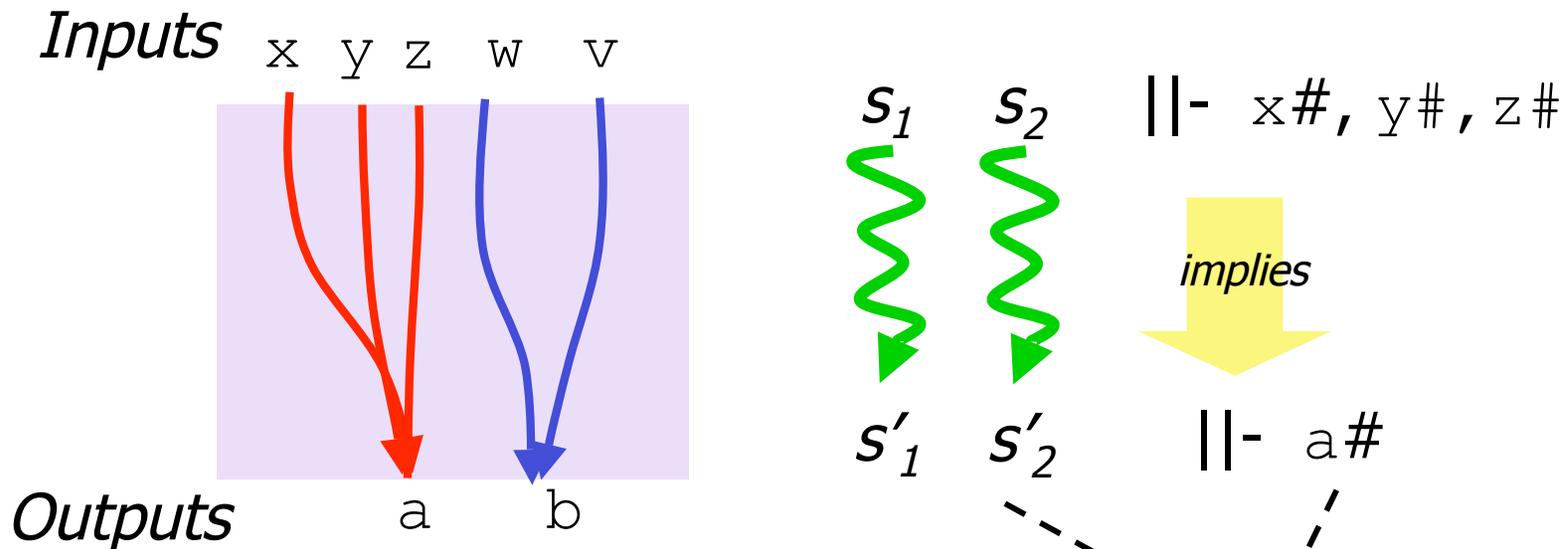
For any two executions with initial states s_1, s_2 ,
if s_1 and s_2 both **agree** on the values of x, y and z ,
then final states s'_1 and s'_2 **agree** on the value of a

Let's have an
abbreviation for the
concept of "agreement"

Non-interference

The classical notion of *non-interference* provides semantic foundation for describing the required separation...

Proving "nothing interferes with red channel"
(i.e., a only depends on x, y and z)...



Agreement Assertion:

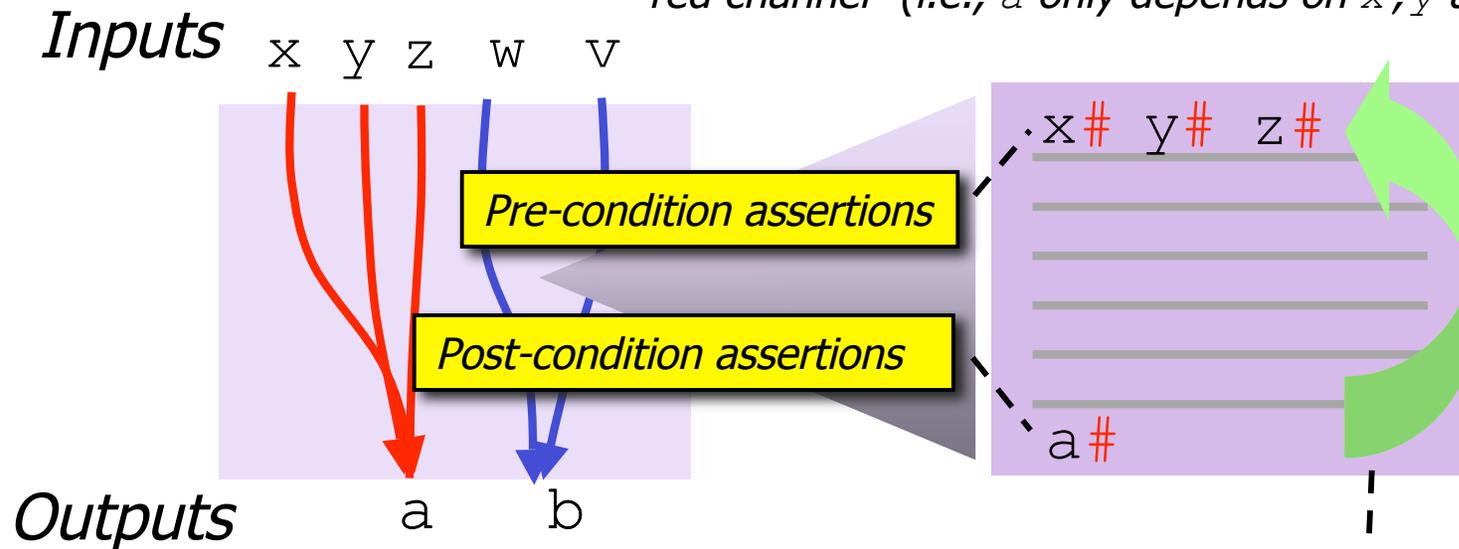
$$s_1, s_2 \Vdash x\# \text{ iff } s_1(x) = s_2(x)$$

Novel form of assertion
has semantics based on
pair of states instead of
just a single state.

Agreement Assertions

Amtoft, Bandhakavi, and Banerjee, developed a novel Hoare-style logic for agreement assertions that includes a pre-condition calculus...

Source-code contract that states "nothing interferes with red channel" (i.e., a only depends on x, y and z)...



Pre-condition calculus provides semantics and a algorithm for inferring pre-conditions from post-conditions

Example Derivation

```
if (x > 5) then
```

```
    a := y;
```

```
else
```

```
    a := z;
```

```
endif
```

```
b := w + v;
```

```
a# ----- Post-condition: agreement on a
```

Example Derivation

```
if (x > 5) then
```

```
    a := y;
```

```
else
```

```
    a := z;
```

```
endif
```

```
a#
```

```
b := w + v;
```

```
a#
```

*Irrelevant to a 's value,
so $a\#$ still required*

Example Derivation

```
if (x > 5) then
```

```
    a := y;
```

```
else
```

```
    z#
```

```
    a := z;
```

```
    a#
```

```
endif
```

```
    a#
```

```
    b := w + v;
```

```
    a#
```

a gets its value from z and so z# is required

Example Derivation

```
if (x > 5) then
```

```
  y#
```

```
  a := y;
```

```
  a#
```

```
else
```

```
  z#
```

```
  a := z;
```

```
  a#
```

```
endif
```

```
  a#
```

```
b := w + v;
```

```
  a#
```

a gets its value from y and so y# is required

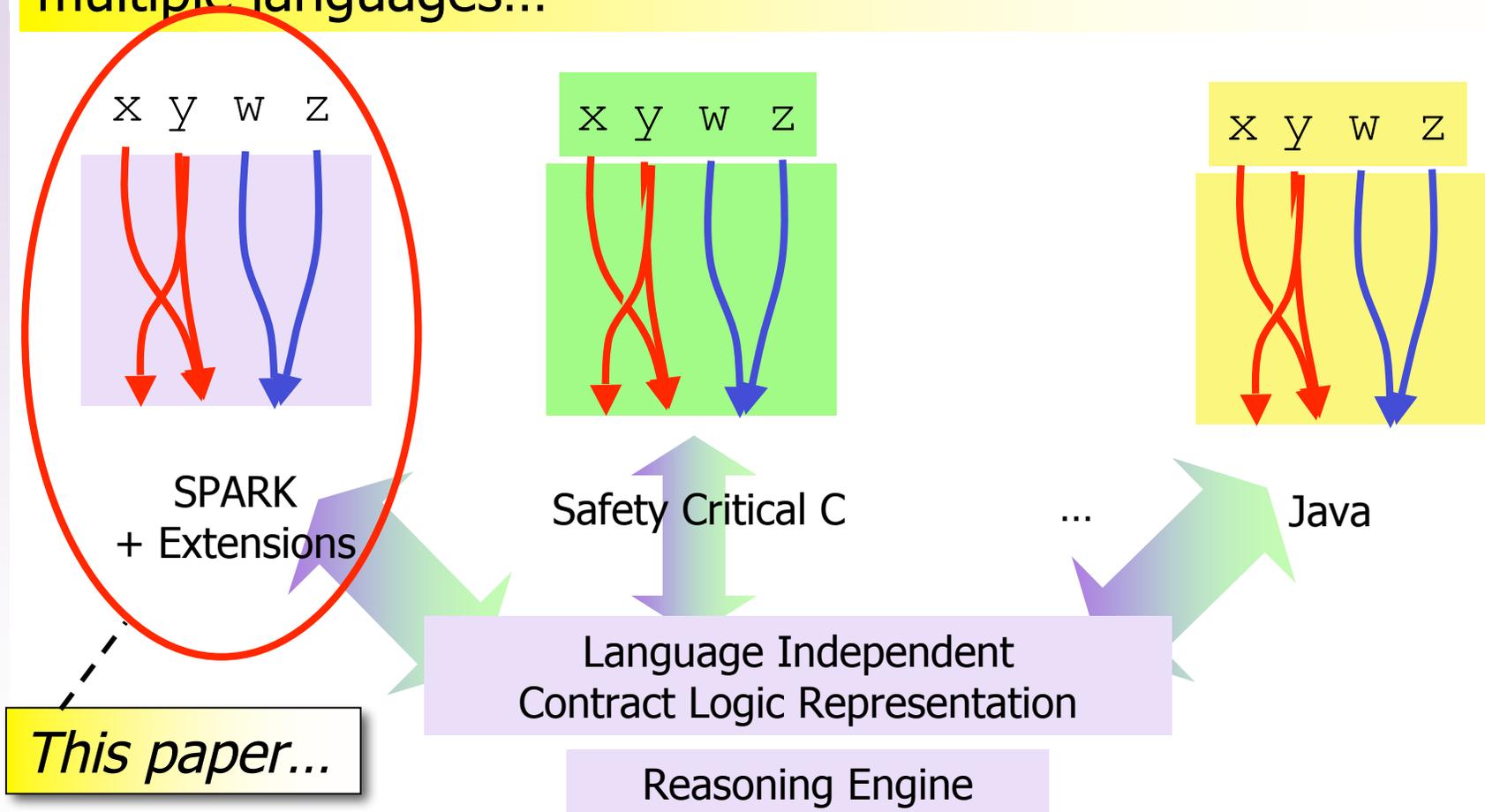
Example Derivation

```
y#, z#, x# -----  
if (x > 5) then  
    y#  
    a := y;  
    a#  
else  
    z#  
    a := z;  
    a#  
endif  
a#  
b := w + v;  
a#
```

*y# is required in true branch,
z# is required in false branch,
x# is required to ensure that
there is agreement upon which
branch is taken*

Our Intent

A language-independent core logic that provides the semantics and implementation for info flow contracts in multiple languages...



Supporting SPARK Contracts

```
--# derives a from y, z, x
--#      b from w, v
  if (x > 5) then
    a := y;
  else
    a := z;
  endif
  b := w + v;
```

Doesn't capture the fact that, e.g., a only depends on y when x > 5

```
y#a, z#a, x#a      w#b v#b
  if (x > 5) then
    y #a
    a := y;
    a#a
  else
    z#a
    a := z;
    a#a
  endif
a#a      w#b v#b
  b := w + v;
a#a      b#b
```

Mailbox Example Derivation

```
INP_1_DAT#  
OUT_0_DAT#  
INP_1_RDY#, OUT_0_RDY#  
if INP_1_RDY and not OUT_0_RDY then  
    INP_1_DAT#  
    DATA_1 := INP_1_DAT;  
    DATA_1#  
    INP_1_RDY := false;  
    DATA_1#  
    OUT_0_DAT := DATA_1;  
    OUT_0_DAT#  
    OUT_0_RDY := true;  
    OUT_0_DAT#  
fi  
OUT_0_DAT#
```

Doesn't capture the fact that OUT_0_DAT only depends on IN_1_DAT when IN_1_RDY and not OUT_0_RDY.

Conditional Agreement Assertions

We'll use a more general form of the logic described by Amtoft and Banerjee in [FMSE 07]...

$s_1, s_2 \Vdash x\#$ *iff* $[[x]]s_1 = [[x]]s_2$



$s_1, s_2 \Vdash P \Rightarrow E\#$ *iff* whenever $[[P]]s_1$ and $[[P]]s_2$ hold
then $[[E]]s_1 = [[E]]s_2$

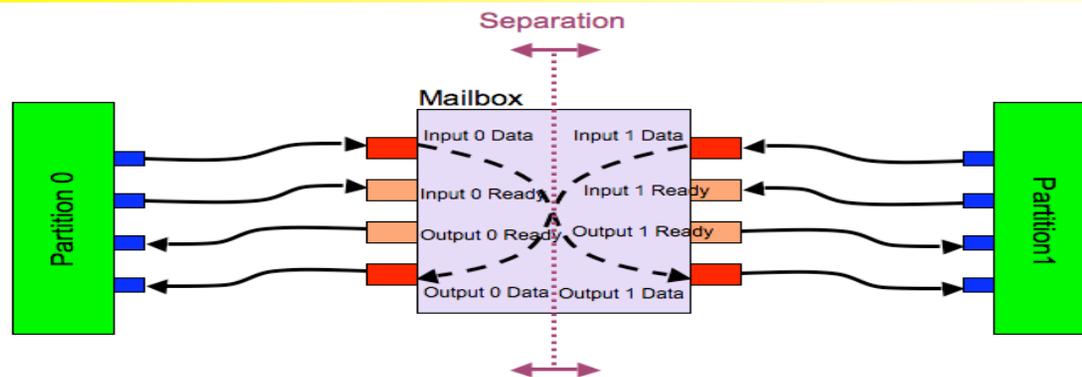
Note: $x\#$ abbreviates $\text{true} \Rightarrow x\#$

Example Derivation

```
INP_1_RDY && !OUT_0_RDY => INP_1_DAT#
!INP_1_RDY || OUT_0_RDY => OUT_0_DAT#
INP_1_RDY#, OUT_0_RDY#
if INP_1_RDY and not OUT_0_RDY then
    INP_1_DAT#
    DATA_1 := INP_1_DAT;
    DATA_1#
    INP_1_RDY := false;
    DATA_1#
    OUT_0_DAT := DATA_1;
    OUT_0_DAT#
    OUT_0_RDY := true;
    OUT_0_DAT#
fi
OUT_0_DAT#
```

Enhanced SPARK Contracts

Enhanced SPARK contract language can now capture the appropriate policy for the mailbox example...



flows exist only under certain conditions

Original SPARK

```
--# derives
--#   Output_1_Data from
--#     Input_0_Data,
--#     Output_1_Data,
--#     Input_0_Ready,
--#     Output_1_Ready
```

New conditional SPARK

```
--# derives
--#   Output_1_Data from
--#     Input_0_Data
--#     when (Input_0_Ready and
--#           not Output_1_Ready),
--#     Output_1_Data,
--#     when (not Input_0_Ready or
--#           Output_1_Ready),
--#     Input_0_Ready,
--#     Output_1_Ready
```

Current Focus -- SPARK

Enhancements/additions to Praxis SPARK that specifically target information assurance applications



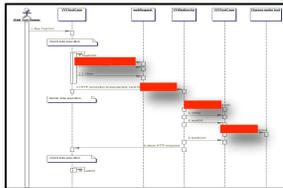
Developers

Information Flow Contracts

```

--# derives
--# KeyStore.SymmetricKey, KeyStore.RotorValue
--# from
--# KeyStore.SymmetricKey
--# &
--# Encrypted
--# from
--# Clear, KeyStore.SymmetricKey
--#
  
```

Checking / Inference



Information Flow Visualization

```

Example1.java
97
98 int a = 0;
99 int b = 0;
100 int c = 0;
101
102 if (_result1 <=
103     a = a + 1;
104     b = b + 1;
105     c = c + 1;
106 )
107
108
109
110 int e = 1;
111 int y = 2;
112 x = x + 1;
113
114
115
116 y = y + 2;
117
118
  
```

Querying/Browsing
of Information Flow

SPARK Eclipse IDE



+ extensions

- Conditional Info Flow
- Precise Array Flows
- Flows across SPARK boundaries (interpartition AAMP flows)
- MLS Security Tags

Provide underlying Hoare-logic representation that enables the production of evidence "certificates" confirming conformance to flow policies

$$\Delta \vdash e : (T, \kappa')$$



Certifiers

