The Impact of SOA Policy-Based Computing on C2 Interoperation and Computing

R. Paul, W. T. Tsai, Jay Bayne

Table of Content

- Introduction
- Service-Oriented Computing
  - Acceptance of SOA within DOD
- Policy-based Computing
  - What is a “Policy”?
  - Policy Specification Languages
- Two different Schools of Thoughts on SOA for Enterprise C2 Systems
- Other Consideration Issues and Open Issues

Features and Challenges in New NC C2 Systems.

- Network-centric nature
- Service-Oriented architecture (SOA) and associated the Service-Oriented Computing (SOC)
- Policy-based computing
    - Policies are specified as a part of the system
    - Policies can be changed and updated at runtime in real time during warfighting
- These features distinguish new C2 systems

from the traditional C2 systems.

Motivation

- Addressing the impact of SOA on modern network-centric enterprise and policy-based C2 systems.
- Network-centric operation is already a difficult issue, but SOA involves additional issues such as C2 policy specifications, V&V and enforcement.
- In an SOA system, a service can be:
  - Discovered, bound, executed, verified and validated at runtime and in real-time; and
  - The entire system is organized as a loosely coupled system of services.

  While this kind of loose coupling provides flexibility and adaptability, it is difficult to enforce C2 policies in such a network as it will involve policy synchronization.

- Each computation unit in such a system is actually a collection of services possibly composed in real time and at runtime. So, how can one enforce C2 policies in such an environment as it is not known what service will be used before application?

Consensus on SOA-based EC2 Systems

- Policies must be specified and embedded in the enterprise C2 system because these policies may be dynamically composed and changed in real time and at runtime;
- Policies should not be hard coded into the system – this will make changes very difficult for agile warfighitng;
- Policies must be enforced in a distributed manner involving policy synchronization and coordination; and
- Actions and tasks must be checked and enforced against stated C2 policies in real time and at runtime.

Service-Oriented Computing

The key concepts of SOC are as follows:

- Each computing unit is considered as a service, and each service is considered self-contained and independent of other services;
- A service publishes its interfaces using a standard language such as WSDL or OWL-S;
- The service provider registers its services with an intermediate broker such as a UDDI server.
- A client that wishes to obtain a service inquires the UDDI server and requests specific services that it needs;
- After receiving the user request, the UDDI matches the need with the most appropriate service in its repository, and sends the ID of the service to the client;
- The client then makes a call to the service requesting specific kind of tasks to be performed; and
- All communications among all parties are through standard protocols such as XML and SOAP.

Difference between SOC & OOC

- SOC may looks similar in many ways to OOC, but it is a different computing paradigm;
- Like OOC is much different from procedural computing, SOC is fundamentally different from OOC;
- Designers are thinking in terms of services, runtime service discovery and composition, runtime system re-composition, and runtime system verification and validations. These are different from the OOC concepts.

Difference between SOC & OOC

(Continued)

- Services are available and accessible on-line. A user or a software agent can use runtime search to discover new services. A user may not need to buy and install;
- SOC also has a significant impact on the system structure and dependability such as system reliability, system composition (and re-composition), and security; and
- The SOA is robust and survivable because most of binding is done at *runtime*, thus services can be removed and/or added to the service pool without changing the overall architecture and protocols among all the parties.

SOA Development Process

- Essentially, SOC will change the whole landscaping in system and software development from requirement to V&V. Specifically, in the **requirement** stage, knowledge of existing services is critical as reusability will be the key enabler. Thus, rather than construction of new requirements, reuse of existing requirements including existing service specifications will be critical.
- In the **design** stage, the loosely coupled SOA allows dynamic composition, dynamic re-composition, and dynamic reconfiguration. In the sense, the design is never completed because new service may arrive after the current design is completed, and the new service may replace the already selected services at runtime in real time.

SOA Development Process (continued)

- In the **implementation** phase, majority of work will be composition and linking rather than code development as services will be reused.
- In the **V&V** phase, **CV&V** (Collaborative Verification and Validation) will be used rather than **IV&V** as the source code of many services may not be available. Service specification will be extensively used because often a service customer has access to specification and URL only, no source code or even binary code would be available.

Differences (continued)

	Traditional IV&V (OO)	Service-Oriented CV&V
Definition	By independent team	Collaboration among multi-parties
Approach	Off-line field testing	On-line just-in-time testing
Regression	Off-line regression	On-line regression
Integration	Static configuration & linking	Dynamic reconfiguration & binding
Testing coverage	Structural & functional	Specification& Usage based
Profiling	Static and centralized	Dynamic and distributed
Model checking	Based source code or states	Just-in-time dynamic model checking
Reliability model	Input domain & Reliability growth models	Dynamic profiling and group testing
Certification	Static certification center	Dynamic certification based on history

Available SOA System Engineering Techniques

- Automated completeness and consistency checking.
- Verification and analysis techniques including both static and dynamic verification techniques;
- Automated state model generation;
- Formal model checking;
- Runtime verification and constraint checking with simulation;
- Automated test case generation;
- Test coverage analysis based on the service specification and risk analysis;
- Automated dependency identification and analysis;
- Automated concurrency analysis; and
- Autonomous distributed test execution via remote agents.

What is a Policy?

- A policy is a statement of the intent of the controller of some computing resources, specifying how he wants them to be used.
  - Policies are ubiquitous in most, if not all, computing systems.
  - However, most policies implemented are hard coded into a system by functional requirements, language features, and design decisions.
  - For example, if a policy says, “passwords must be at least 8 characters long”, there must exist a segment of code in the system that checks the length of passwords.

What is a Policy? (Continued)

- A policy allows or denies subjects (such as processes) that satisfy some conditions to perform designated actions on objects (such as files).
- ***Policy***,
  - A course or general plan of action.
  - A contract of insurance governing a plan of action.
  - The rules or constraints governing a general plan of action.

What is a Policy? (Cont'd)

- The common pattern of a policy takes the form of:

{allow | deny | require} {subject} performs {action} on {object} when {conditions}

- *Policy specification*, *enforcement* and *revision* are the three basic mechanisms a policy-handling system must provide.

Limitations of Traditional Policy Handling Systems

- It does not separate policy specification from policy implementation.
- Policies are difficult and expensive to change. Adding new policies or updating / removing existing policies requires modifying the policies, recompiling and redeploying these policies into the system.

Benefits of Policy Specification Languages in Enterprise C2 (EC2)

- Policy specification languages enable policies to be defined, independent from a concrete system implementation;
- Policy specification languages are to be interpreted by a policy engine at runtime, which makes dynamical policy changes possible;
- Policy specification languages formalize the intent of the commander into a form that can be analyzed and interpreted by systems; and
- Policy specification languages are high-level languages, which makes it easy to learn and use by policy makers who are normally non-programmers.

Policy-Based Computing in GIG

- **Policy-Based Networking**: various network operations will be possible but they will be controlled by policies in an EC2 SOA.
- **Common Open Policy Service**: this is a query and response protocol that can be used to exchange policy information between a policy server and its clients.
- **Routing Protocol Specification Language**: this allows a network operator to specify routing policies at various levels.
- **Internet Protocol (IP) Security Policy**: this is a repository-independent information model for supporting IP security policies.

Policy-based Computing

- Each computing unit has a formal and self consistent set of C2 policies,
- It has a set of C2 policy execution processes (mechanisms),
- It has a set of C2 process performance measures and measurement processes, and
- It has an accountability structure governing allocation of assets required in and consumed by the execution of any given policy or set of policies
- Each policy will be applicable to a policy domain. The EC2 accountability structure defines a policy tree spanning selected activities of the joint field combatant commanders up through the President of the United States. This “nested” set of interdependent policy domains constitutes a *containment hierarchy*. Generally, enterprise policy trees include high-level strategic, mid-level operational and low-level tactical policy domains.

Executable Policies in EC2 SOA

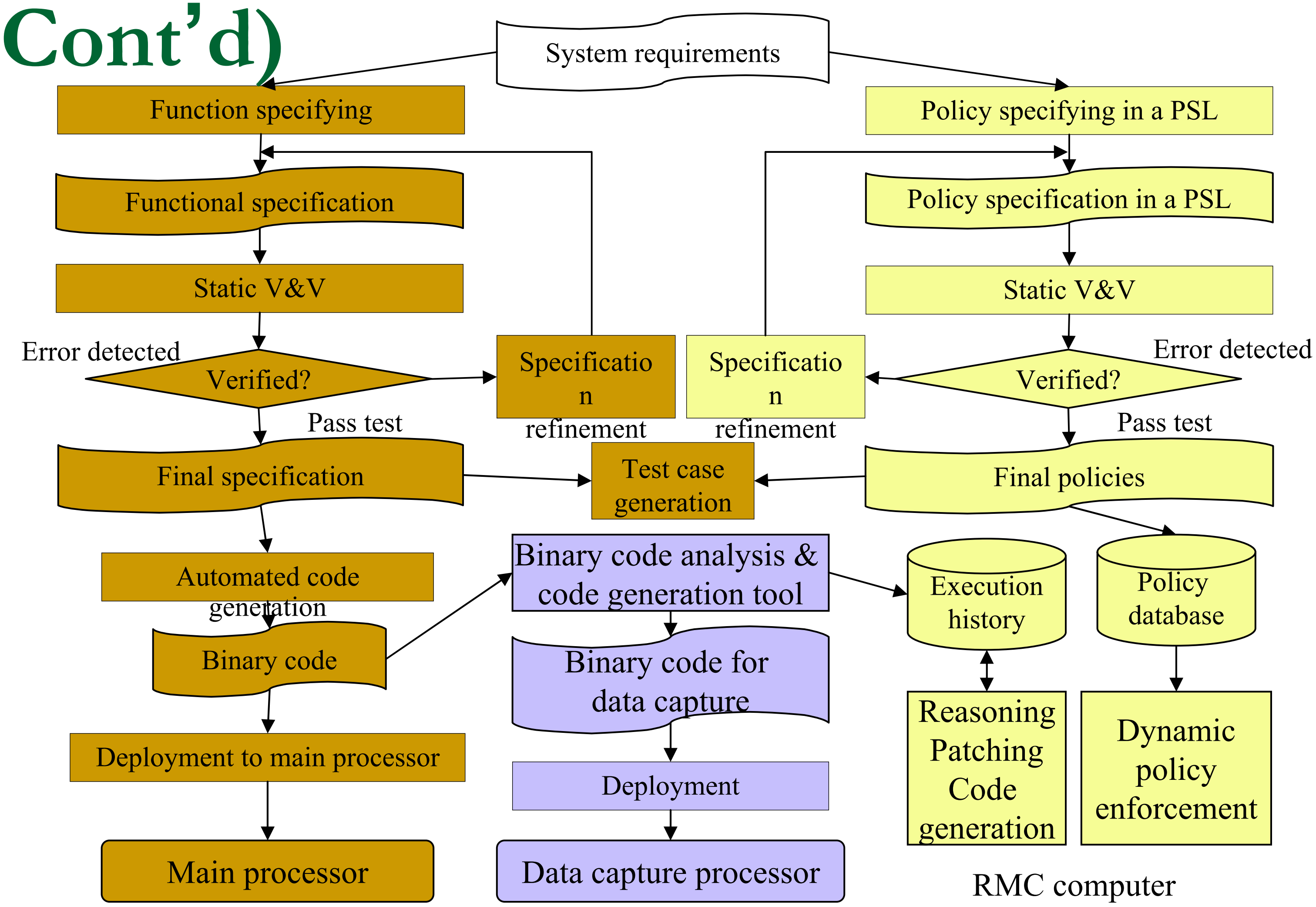
- A policy in an EC2 SOA must be executable as policies will be enforced at runtime in real time, and thus it is essential policies can be executed at runtime and in real time to enforce their constraints.
- The execution capability of a policy language support simulation of adding or modifying policy prior to their deployment throughout relevant policy domains, and
- Provide a formal means of *verification and validation* that policies are "correct" with respect to quality of service (QOS) and service level agreements (SLA) established for their respective containment domains at runtime and in real time.

Policy Enforcement and Operations

A policy language must allow policies specified to be enforced either statically or at runtime. Both static and runtime policy enforcement follows a 3-step process:

- Policy C&C checking: This ensures that policies specified are complete with external requirements and regulations, as well as consistent with each other before application of these policies to evaluate the concerned system. Sample security policies can be Bell-LaPadular security policies or Chinese Wall security policies;
- Embed policies with the system specification: Once the policies specified are determined to be of reasonable quality, the policies specified using PSEL can now be embedded into the system specification for evaluation; and
- Evaluation of the system with respect to policies specified: This step evaluates the system to see if it satisfies the policies specified.

Policy Enforcement and Operations (Cont'd)



Different Schools of Thoughts

- One school of thought (School A):
  - SOA is simply an *implementation* technology just like programming languages or design techniques such as OOC;
  - Emphasize that fundamental C2 policies remain the same, and thus focus on generic EC2 architecture independent of implementation technology.
  - Policies are *passive*, and treated as static XML text for constraint checking and enforcement.
  - Processes used to enforce policies must be synchronized.
  - Frequent changes in policies can be problematic, and thus policies will change but a slower pace. Updating policies may involve updating related processes (services).

Different Schools of Thoughts (Cont'd)

- Another school of thoughts (School B)
  - Treat policies as services in an SOA. In other words, policies are *active* processes, and can be published, disseminated, executed and monitored in real time and at runtime.
  - Change of policies will be done similar to change in services in a uniform way. The SOA already has the overhead of service discovery and execution, and this overhead can be used to support policy-based computing.
  - The flexibility and adaptability will be better.

Comparisons

	Policies as passive objects (School A)	**Policies as active services (School B)**
Policy Specification	Policies specified as data such as XML files.	Policies specified as services
Policy Enforcement	Policies will be enforced by execution services, different enforcement algorithms and software can be used.	Policy services are active services that can perform enforcements. Multiple services may cooperate together to accomplish service enforcement.
Policy Overhead	As policies are treated as static objects, the overhead of policy is the overhead of accompanied computation in managing and enforcement of policies.	Policy services is just like another services. The overhead is the added complexity of activating policy services while doing regular computing.
Policy Update	Policy can be updated by updating the related policy files, and possibly policy processing services as well.	Policy can be updated by replacing the existing policy service by a new policy service.
Policy V&V	Various static and dynamic V&V mechanisms can be used, e.g., completeness and consistency checking, and policy simulation.	Policy V&V will be equivalent to service V&V, a variety of static and dynamic V&V techniques can be used including test case generation, simulation, and runtime monitoring.

Comparison (Cont'd)

	Policies as passive objects (School A)	**Policies as active services (School B)**
Policy Simulation	Policy can be simulated by running the system using policy data.	As a policy service is just another service in an SOA, policy simulation can be carried using any SOA simulation.
Policy communication	Two systems can send policies to each other like sending a regular electronic message as policies are treated as data.	Two systems can communicate their policies by sending service specifications or the URL of the related services.
Policy synchronization	Policy synchronization is needed whenever a policy is updated or the system is reconfigured. Formal synchronization protocols need to be employed to ensure completeness and consistency.	Policy synchronization is needed whenever a policy service is updated or system is reconfigured. Policy services may actively pursue their own service synchronization protocols.
System Structure	Enterprise C2 systems will be organized in an SOA, but policies will be treated as objects or data used by services.	Enterprise C2 systems will be organized in an SOA-manner and policy services will be treated like a regular service.
System Reconfiguration	Systems can be easily reconfigured using a DRS (dynamic reconfiguration service) and policy files need to be updated to ensure that new policies are consistent with the new system structure.	System can be easily reconfigured using a DRS, and policy services will be a part of reconfiguration process.

Other Consideration Issues and Open Issues

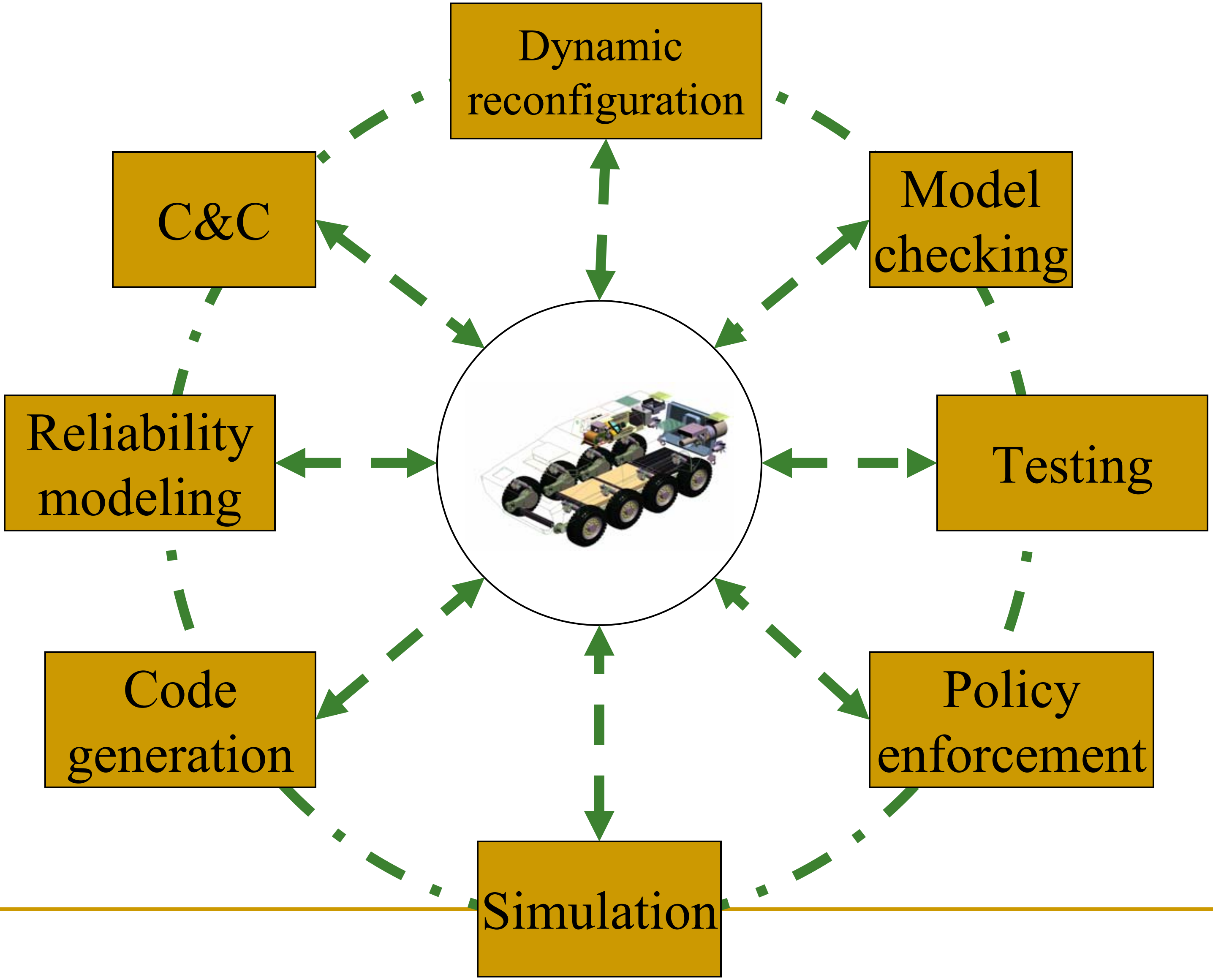
The EC2 SOA still face many open problems such as:

- A formal yet easy to use policy specification language that can be used to specify, analyze, simulate and enforce distributed policies in an EC2 SOA federated systems;
- Distributed policy synchronization and enforcement; and
- Policy dependency analysis to assist real-time policy evolution;
- These and other related issues must be addressed before policy-based computing can be realized.

Demo

- A network-centric policy enforcement on an embedded system is available.
- This is to demonstrate that it is possible to do real-time code generation, dynamic reconfiguration, verification, policy specification, policy enforcement, simulation at runtime. These usually take days if not weeks or months to do, now they can be performed on-the-fly using an SOA technology in seconds automatically.

Real-Time and Network-Centric V&V



Dynamic reconfiguration

C&C

Model checking

Reliability modeling

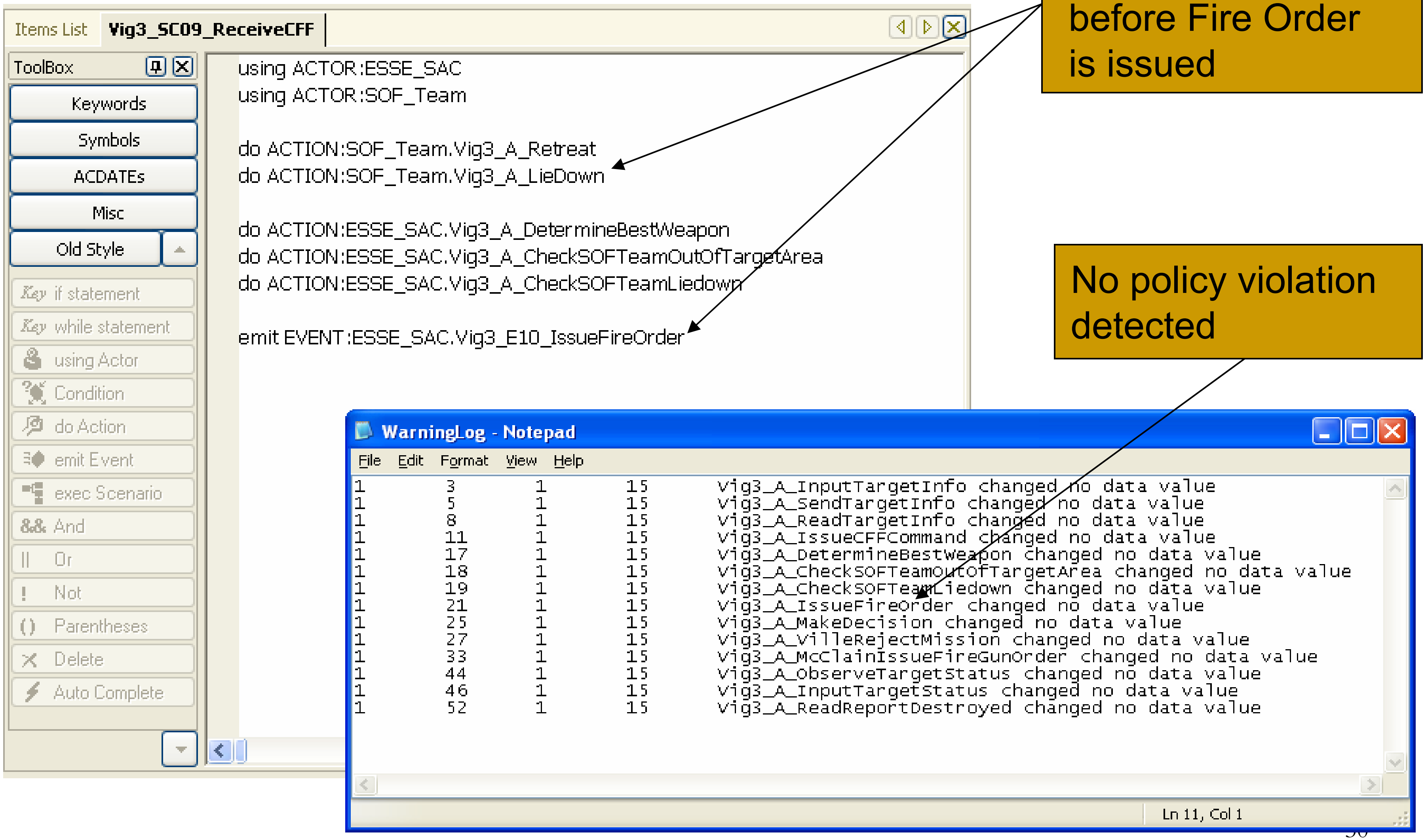
Testing

Code generation

Policy enforcement

Simulation

Acceptable Scenario



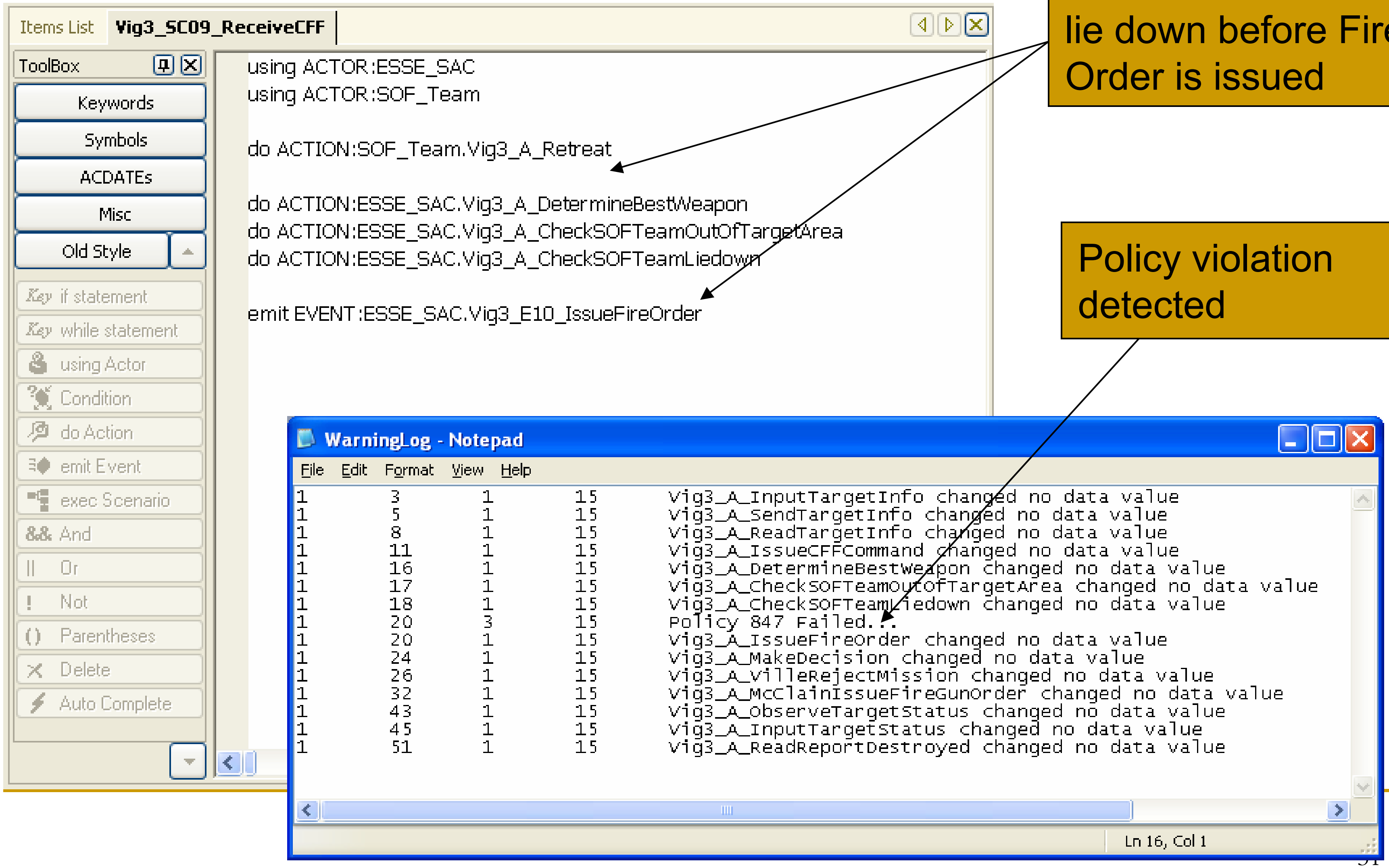
```
using ACTOR:ESSE_SAC
using ACTOR:SOF_Team

do ACTION:SOF_Team.Vig3_A_Retreat
do ACTION:SOF_Team.Vig3_A_LieDown

do ACTION:ESSE_SAC.Vig3_A_DetermineBestWeapon
do ACTION:ESSE_SAC.Vig3_A_CheckSOFTeamOutOfTargetArea
do ACTION:ESSE_SAC.Vig3_A_CheckSOFTeamLiedown

emit EVENT:ESSE_SAC.Vig3_E10_IssueFireOrder
```

```
1       3       1       15      Vig3_A_InputTargetInfo changed no data value
1       5       1       15      Vig3_A_SendTargetInfo changed no data value
1       8       1       15      Vig3_A_ReadTargetInfo changed no data value
1       11      1       15      Vig3_A_IssueCFFCommand changed no data value
1       17      1       15      Vig3_A_DetermineBestWeapon changed no data value
1       18      1       15      Vig3_A_CheckSOFTeamOutOfTargetArea changed no data value
1       19      1       15      Vig3_A_CheckSOFTeamLiedown changed no data value
1       21      1       15      Vig3_A_IssueFireOrder changed no data value
1       25      1       15      Vig3_A_MakeDecision changed no data value
1       27      1       15      Vig3_A_VilleRejectMission changed no data value
1       33      1       15      Vig3_A_McClainIssueFireGunOrder changed no data value
1       44      1       15      Vig3_A_ObserveTargetStatus changed no data value
1       46      1       15      Vig3_A_InputTargetStatus changed no data value
1       52      1       15      Vig3_A_ReadReportDestroyed changed no data value
```

SOF Team lies down before Fire Order is issued

No policy violation detected

Scenario Violating Policy



```
using ACTOR:ESSE_SAC
using ACTOR:SOF_Team

do ACTION:SOF_Team.Vig3_A_Retreat

do ACTION:ESSE_SAC.Vig3_A_DetermineBestWeapon
do ACTION:ESSE_SAC.Vig3_A_CheckSOFTeamOutOfTargetArea
do ACTION:ESSE_SAC.Vig3_A_CheckSOFTeamLiedown

emit EVENT:ESSE_SAC.Vig3_E10_IssueFireOrder
```

```
1       3       1       15      Vig3_A_InputTargetInfo changed no data value
1       5       1       15      Vig3_A_SendTargetInfo changed no data value
1       8       1       15      Vig3_A_ReadTargetInfo changed no data value
1       11      1       15      Vig3_A_IssueCFFCommand changed no data value
1       16      1       15      Vig3_A_DetermineBestWeapon changed no data value
1       17      1       15      Vig3_A_CheckSOFTeamOutOfTargetArea changed no data value
1       18      1       15      Vig3_A_CheckSOFTeamLiedown changed no data value
1       20      3       15      Policy 847 Failed...
1       20      1       15      Vig3_A_IssueFireOrder changed no data value
1       24      1       15      Vig3_A_MakeDecision changed no data value
1       26      1       15      Vig3_A_VilleRejectMission changed no data value
1       32      1       15      Vig3_A_McClainIssueFireGunOrder changed no data value
1       43      1       15      Vig3_A_ObserveTargetStatus changed no data value
1       45      1       15      Vig3_A_InputTargetStatus changed no data value
1       51      1       15      Vig3_A_ReadReportDestroyed changed no data value
```

SOF Team doesn't lie down before Fire Order is issued

Policy violation detected