

# Certiﬁable Distributed Runtime Assurance

## Challenges

- Assure safety of distributed cyber-physical systems
- Unpredictable algorithms (machine learning)
- Multi-vehicle (distributed) coordinating to achieve mission

## Solutions

- Add simpler (verifiable) runtime enforcer to make algorithms predictable
- Formally: specify, verify, and compose multiple enforcers
- Enforcer intercepts/replaces unsafe action at right time

## Formalization (time-aware logic)

### State of system: Variable Values

#### Statespace & Actions

- $S = \{s\}$ , Safe states:  $\phi \subseteq S$
- $R_p(\alpha) \subseteq S \times S$ ;  $R_p(\alpha, s) = \{s' \mid (s, s') \in R_p(\alpha)\}$

#### Enforceable states

- $C_\phi = \{s \mid \exists \alpha: R_p(\alpha, s) \in C_\phi\}$

#### Safe actions:

- $SafeAct(s) = \{\alpha \mid R_p(\alpha, s) \in C_\phi\}$

#### Logical Enforcer: $E = (P, C_\phi, \mu)$

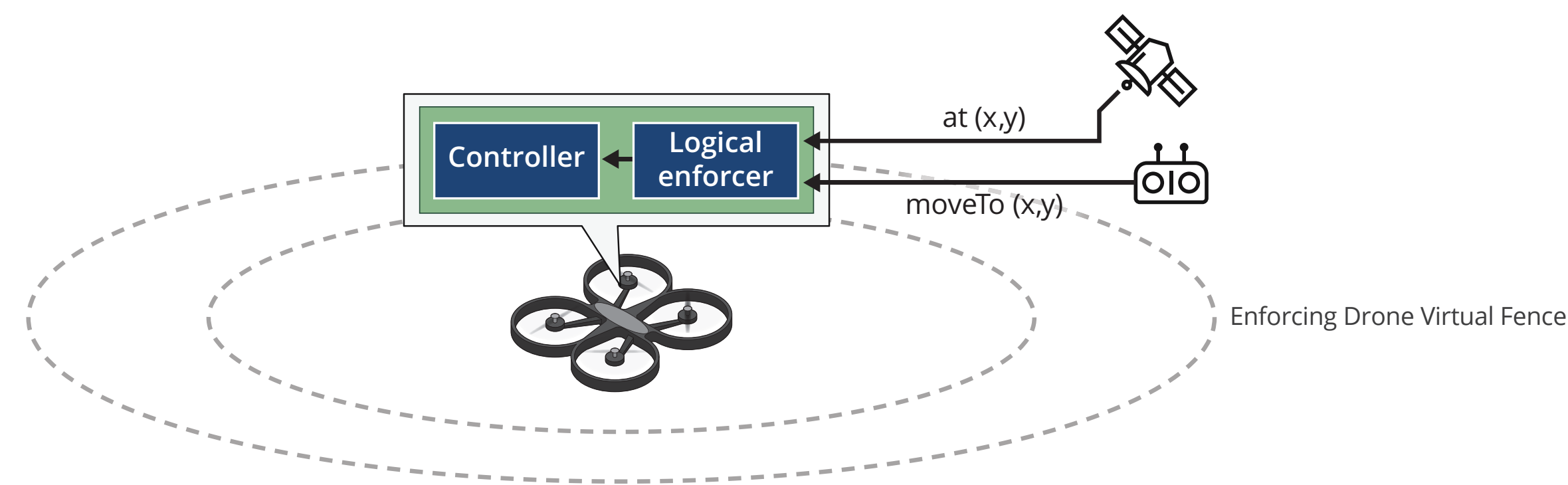
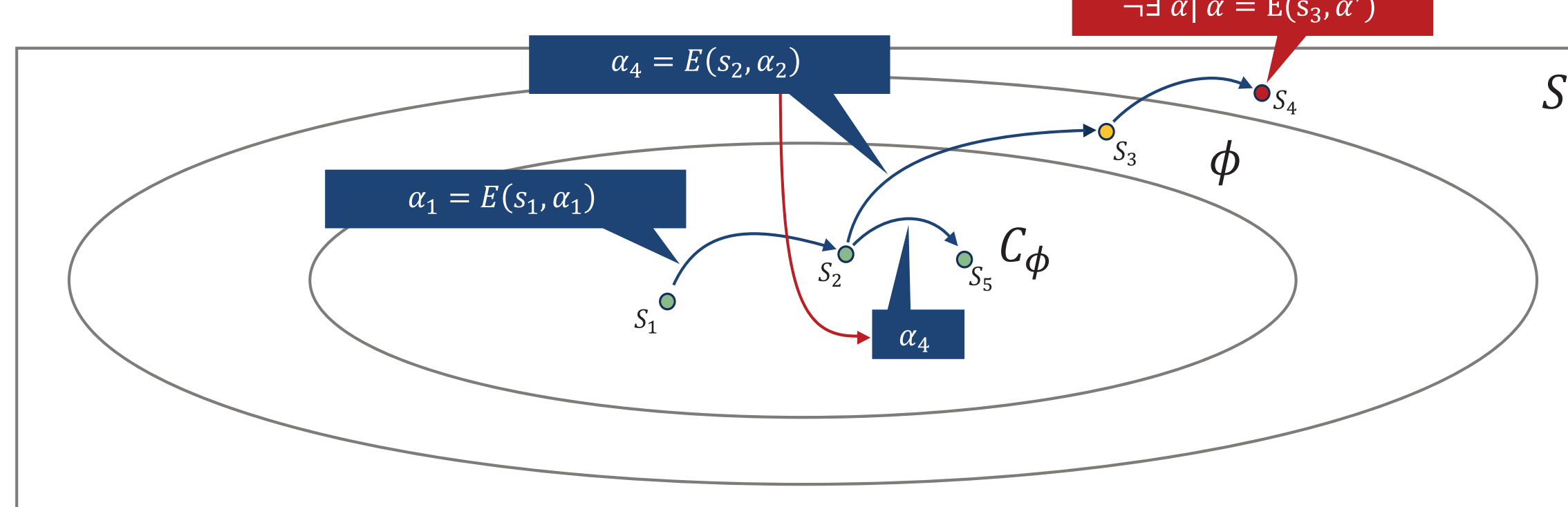
- Set of safe actions:

$$\mu(s) \subseteq SafeAct(s)$$

- Monitor and enforce safe action:

$$(\alpha =) \begin{cases} \alpha, & \alpha \in \mu(s) \\ pick(\mu(s)), & otherwise \end{cases}$$

Enforcer  $E(s, \alpha): \alpha \in SafeAct(s) ? \alpha : \alpha' \in SafeAct(s)$



## Timing Enforcement

- Unverified software may never finish!
- $\Rightarrow$  No action produced to be enforced!

## Temporal Enforcer

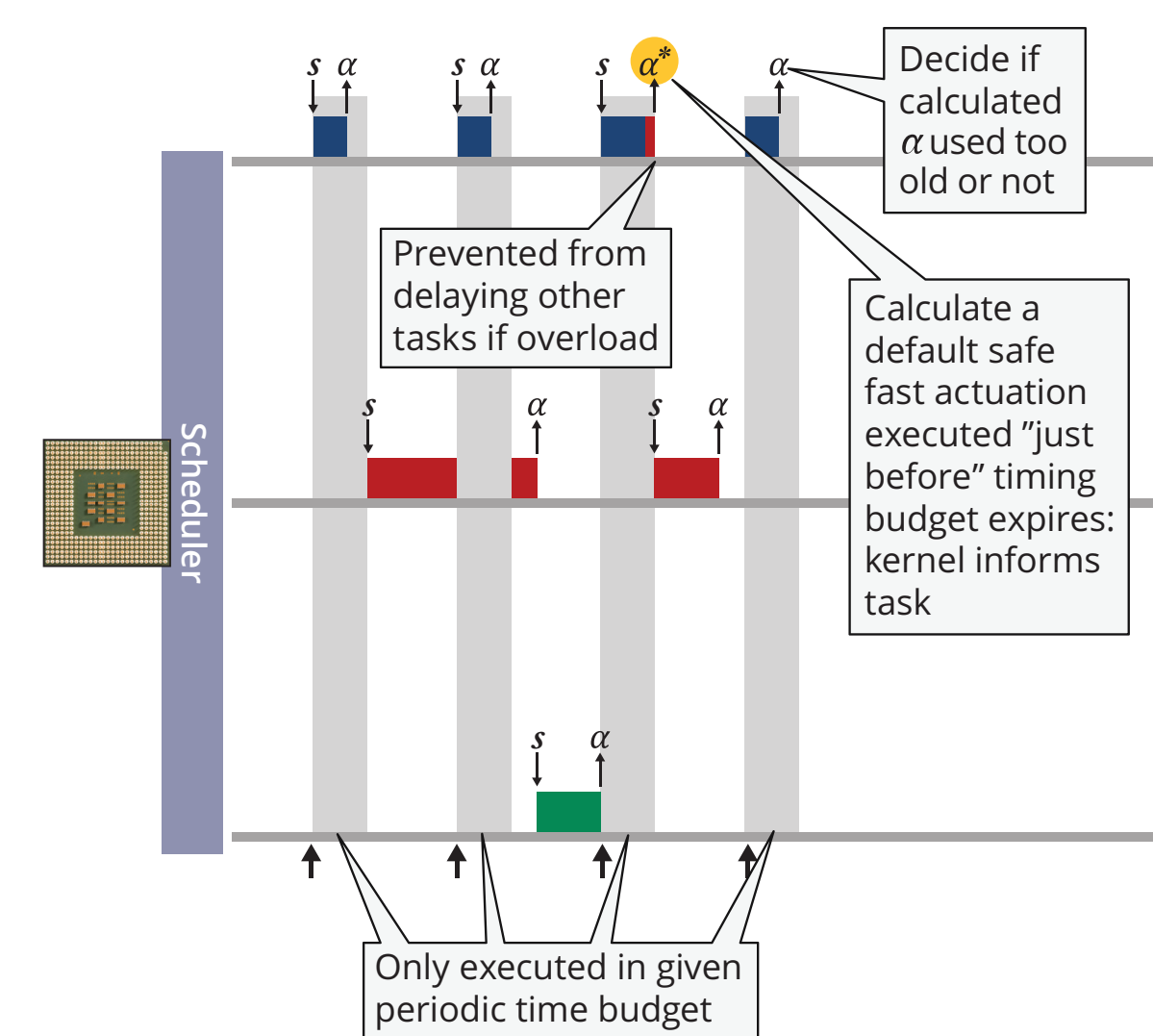
- Protect other tasks from bogus never-ending (or large) executions
- Produce default safe actuation if task takes too long

## How

- Each task gets a CPU budget stop task if budget exceeded
- If task about to exceed budget executes safe action

## Timing Guarantees

- Never allow task to exceed budget
- Always execute actuation

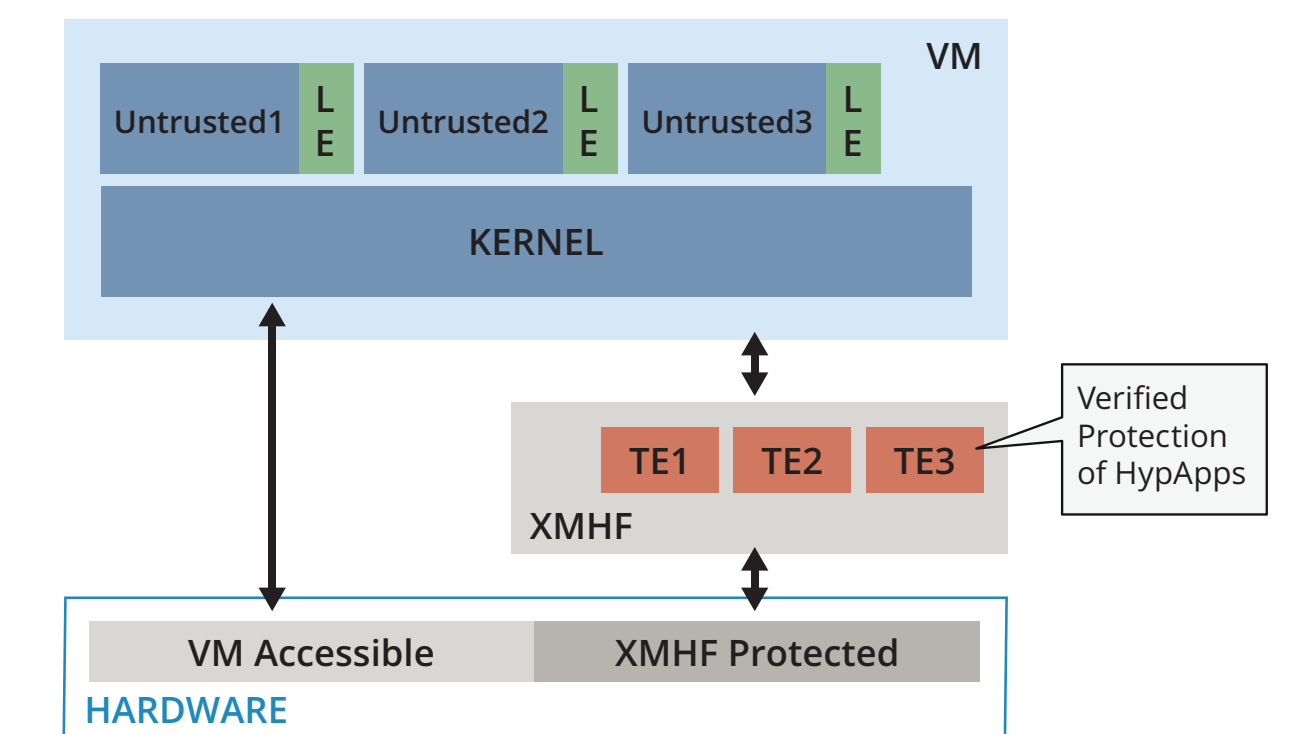


## Protecting Enforcer from Untrusted Components

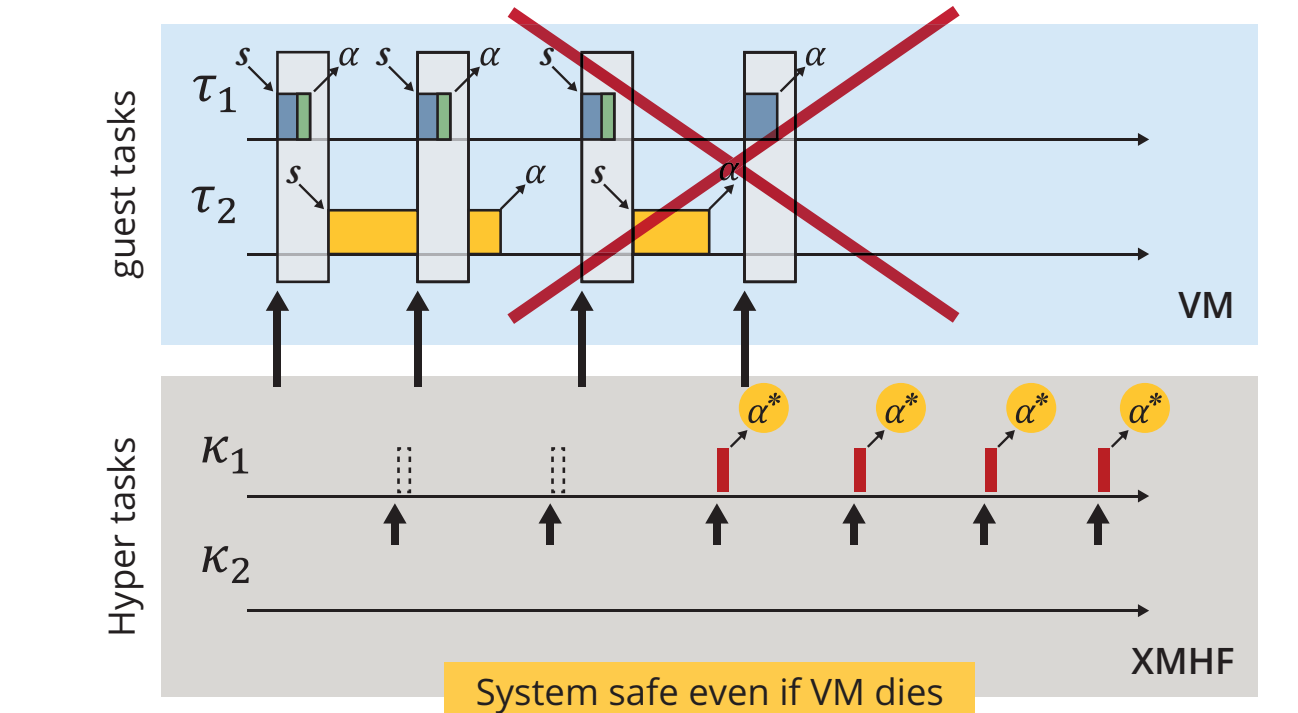
- Unverified software faults may corrupt enforcers

## Mixed Trust Computing and Scheduling

- Untrusted/unverified software for quick/cheap fielding
- Trusted enforcer to guarantee safety properties
- Micro-Hypervisor protects enforcers
- Coordinated VM+Hypervisor schedulers guarantees timing



Mixed-trust task:  $\mu_i = (\tau_i, \kappa_i)$



## Enforcers Allows Verification of Complex CPS: Autonomous Vehicles

- Limit misbehavior with verifiable enforcers
- Result: Verified whole system

## Verified: Logic, Timing, Physics!

## KEY for Assured Autonomy

Copyright 2018 Carnegie Mellon University. All Rights Reserved.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Internal use:\* Permission to reproduce this material and to prepare derivative works from this material for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use:\* This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other external and/or commercial use. Requests for permission should be directed to the Software Engineering Institute at [permission@sei.cmu.edu](mailto:permission@sei.cmu.edu).

\* These restrictions do not apply to U.S. government entities.

Carnegie Mellon® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM18-1131