Carnegie Mellon University
Software Engineering Institute

Self-Adaptive Systems

SEPTEMBER 26, 2023

Gabriel Moreno, PhD
Principal Researcher

DM23-1014

Self-Adaptive Software evaluates its own behavior and changes behavior when the evaluation indicates that it is not accomplishing what the software is intended to do, or when better functionality or performance is possible.

[DARPA BAA-98-12]

Self-Adaptation Characteristics

Self-configuration

- Ability to install, replace and configure components and to modify the structure of the system

Self-optimization

- Ability to allocate resources in the most efficient way to achieve the system's goals

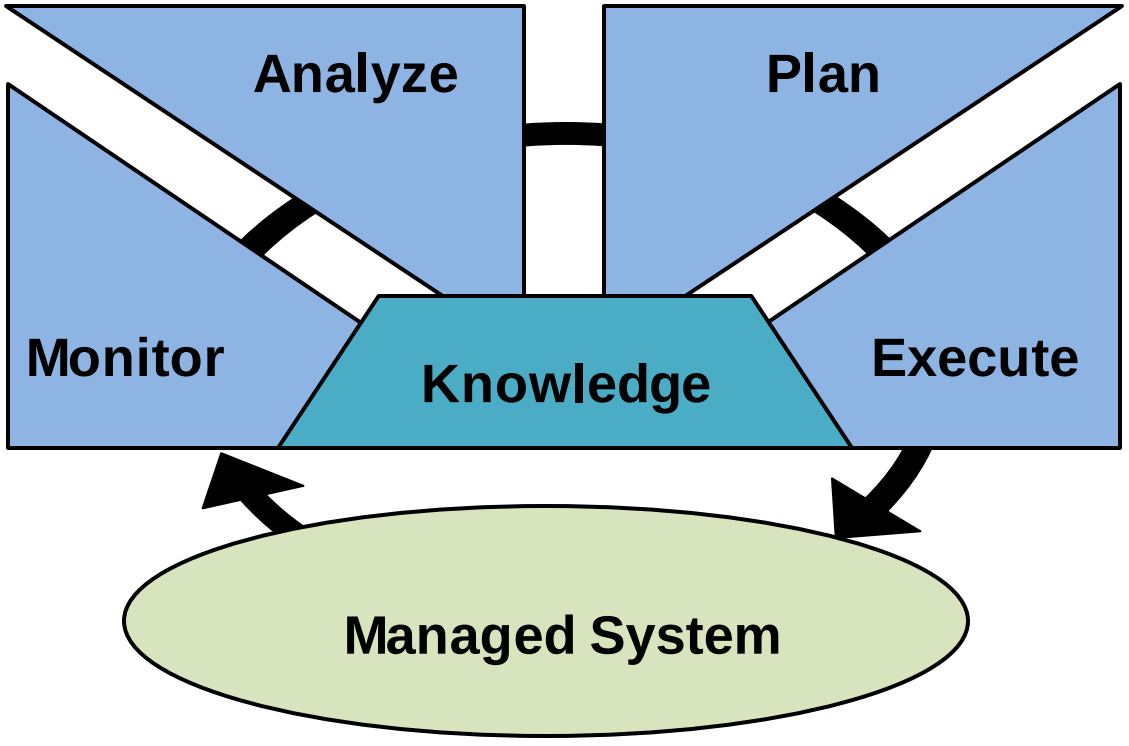
Self-healing

- Ability to detect, diagnose and recover from problems

Self-protection

- Ability to defend against cyber-attacks and to avoid or mitigate security problems anticipated by early warnings

MAPE-K Loop



J.O. Kephart, and D.M. Chess. "The vision of autonomic computing." *Computer* 36, no. 1, 2003

Self-Adaptation Approaches

Event-Condition-Action (ECA) Rules

- Simple method for analysis and planning.
- Rules manually created based on heuristics.

When average CPU utilization goes over 60%,	← event
if maximum number of servers not reached ,	← condition
add a server	← action

- Example: Amazon EC2 auto-scaling

Pros and Cons of ECA Rules

Carnegie Mellon University Software Engineering Institute

Pros

- Each individual rule is easy to understand
- Rules can be added incrementally

Cons

- The management of a goal can be scattered over multiple rules, hindering comprehension
- Multiple rules can interact and have conflicts
- It's difficult to guarantee coverage of the whole adaptation space

H. Ghanbari, B. Simmons, M. Litoiu, & G. Iszlai. "Exploring alternative approaches to implement an elasticity policy." *Proceedings of the International Conference on Cloud Computing*, 2011

Utility-Based Adaptation Decisions

- In general, more than one adaptation option is applicable at a given time.
- The adaptation decision consists in selecting the best.

Utility-Based Decision

Utility: metric that quantifies the goodness of an adaptation alternative.

Usually a function of how the resulting configuration would perform in the current environment.

$$U = f(c, e)$$

Algorithm: filter out not-applicable alternatives; quantify the utility for each applicable alternative; select the adaptation with the highest utility

Architecture-Based Self-Adaptation

- Software architecture
  - provides a global perspective of the system
  - exposes system properties
    - response time, availability, etc.
  - with the use of architectural styles, imposes restrictions on the possible changes (allows checking the integrity of the adaptation)
- Focuses on the dynamic structure of the architecture (components and connectors)
- Allows using different existing architecture analyses

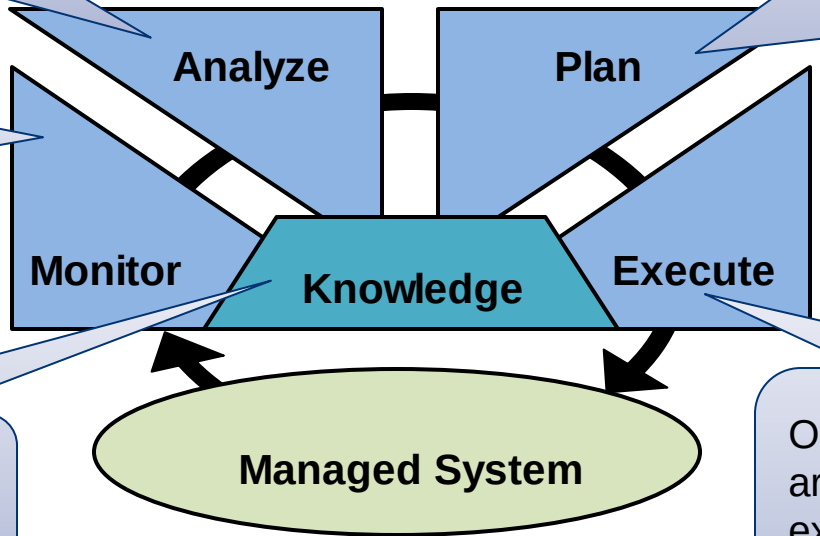
D. Garlan, B. Schmerl, & S.-W. Cheng. "Software architecture-based self-adaptation." In *Autonomic Computing and Networking*, 2009

Architecture-Based Self-Adaptation

Adaptation goals are typically functions of the properties of the architecture or its components

- Architectural style guides possible changes
- Architecture analyses are used to evaluate the impact of possible changes

Keeps the architecture model (structure and properties) up-to-date

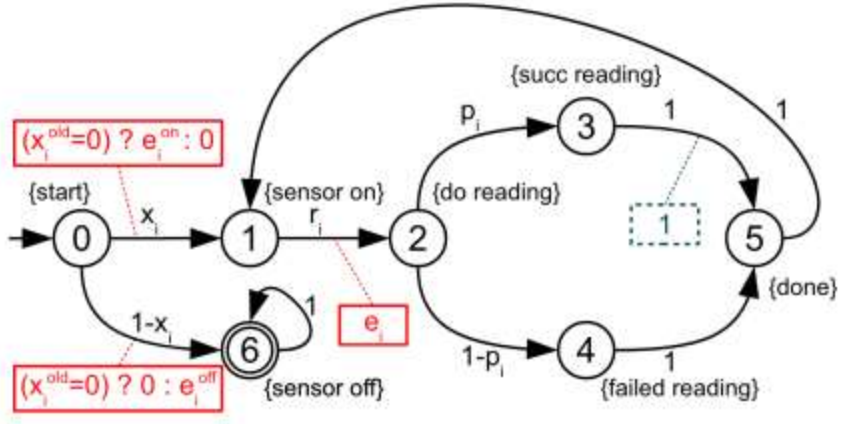


Model of the software architecture in a machine-processable format

Operators defined over the architecture make it easier to express changes (e.g., connect components)

Runtime Quantitative Verification

- The system is mathematically modeled using probabilistic formalisms
  - Discrete-Time Markov Chain (DTMC), Continuous Time Markov Chain (CTMC)
- System requirements are formally expressed
  - Probabilistic Computation Tree Logic (PCTL)
  - Continuous Stochastic Logic (CSL)



The energy consumption of the sensors should not exceed 120 Joules per 10 surveyed meters [at a speed of s]

$$R^{energy}_{\leq 120}\left[C^{\leq 10/s}\right]$$

source: [2]

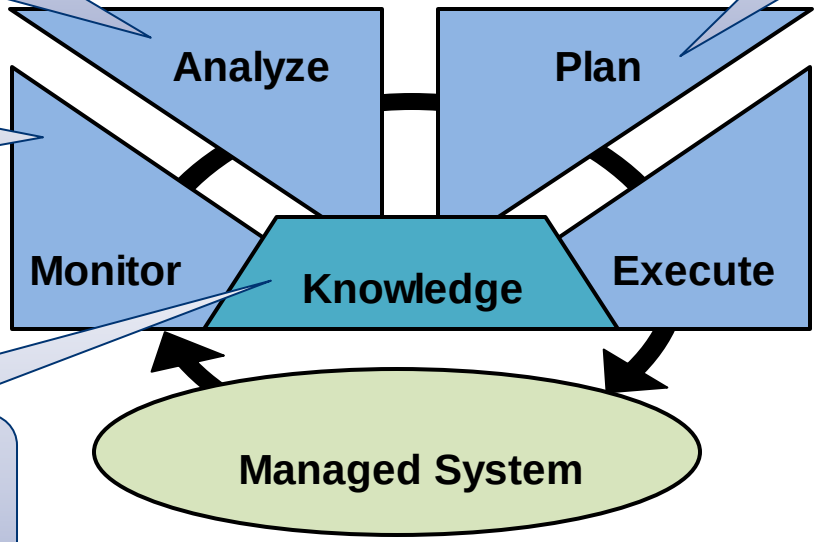
[1] Calinescu, R., C. Ghezzi, M. Kwiatkowska, and R. Mirandola. "Self-Adaptive Software Needs Quantitative Verification at Runtime." *Communications of the ACM*, 2012.

[2] Gerasimou, S., R. Calinescu, and A. Banks. "Efficient Runtime Quantitative Verification Using Caching, Lookahead, and Nearly-Optimal Reconfiguration." *International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, 2014.

Runtime Quantitative Verification

Runtime verification is used to check if requirements are being or will be violated

Runtime verification is used to select a new system configuration that satisfies requirements

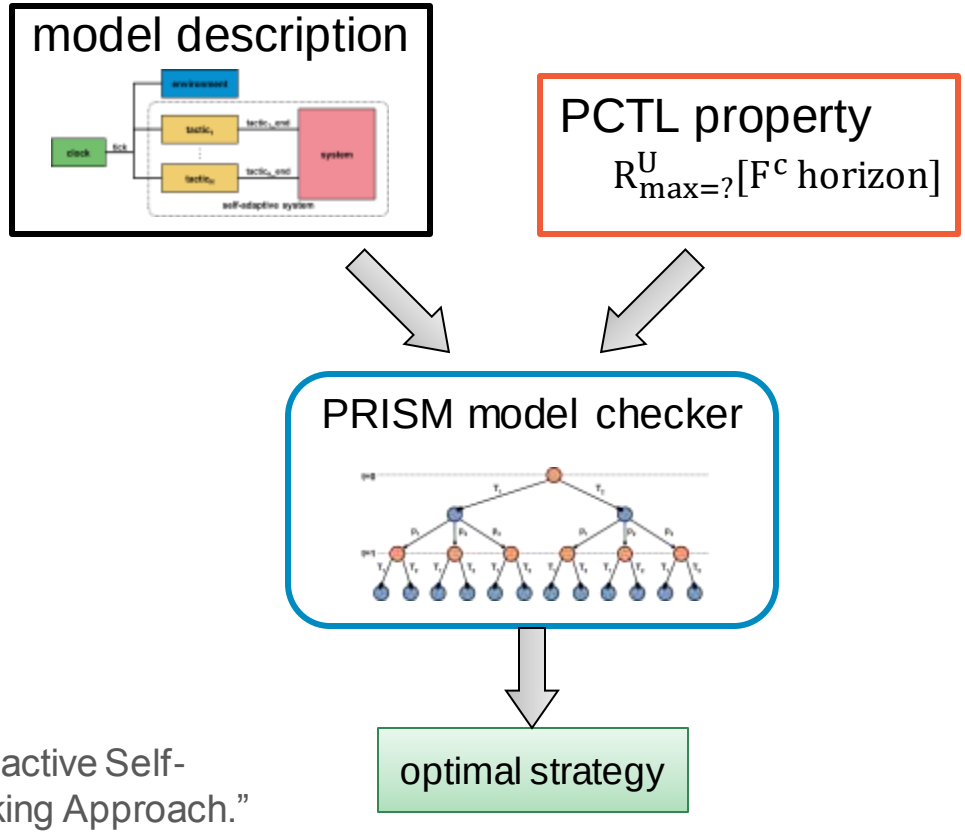


Keeps parameters of the formal model up-to-date

Formal model of the system (DTMC or CTMC)

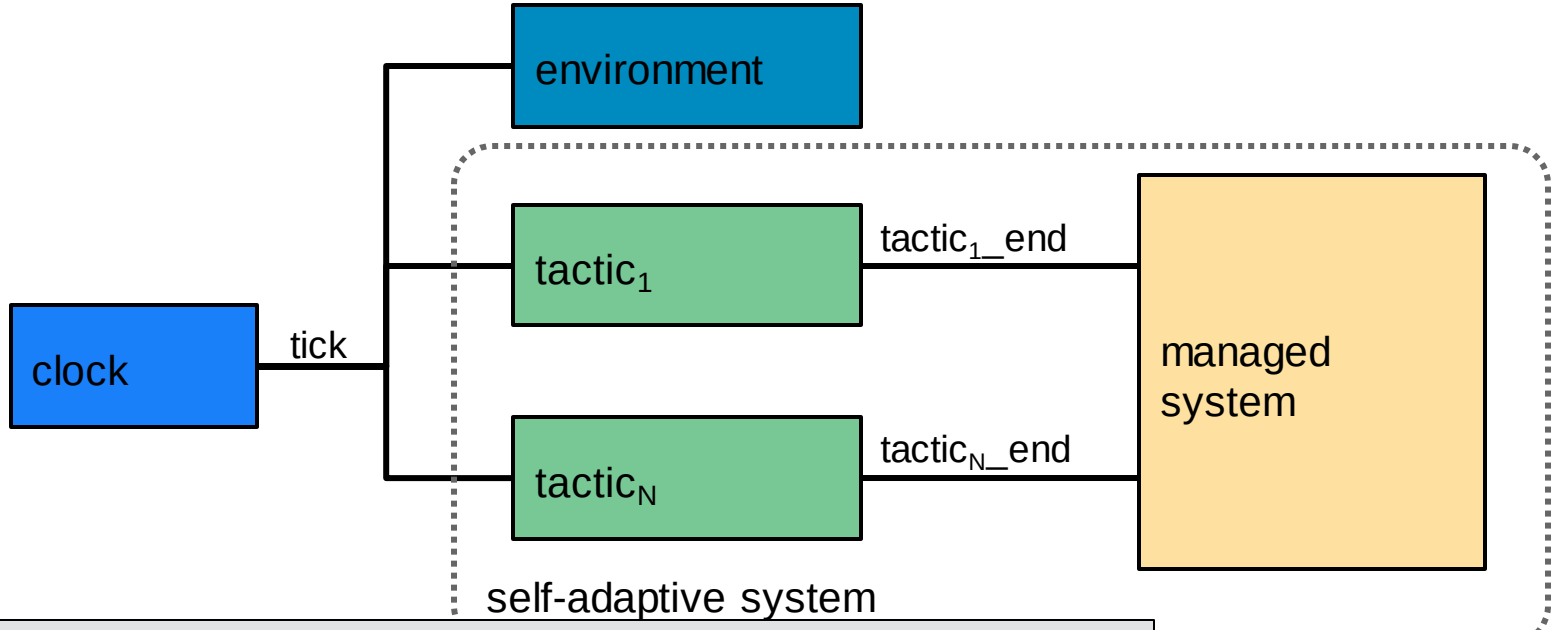
Proactive Latency-Aware Adaptation with Probabilistic Model Checking (PLA-PMC)

- The system *and* the adaptation tactics mathematically modeled using a Markov Decision Process (MDP)
- Utility function formally expressed as a reward property PCTL
- Probabilistic model checker used to synthesize optimal adaptation strategy



Moreno, G.A., J. Cámara, D. Garlan, and B. Schmerl. "Proactive Self-Adaptation under Uncertainty: A Probabilistic Model Checking Approach." *Joint Meeting on Foundations of Software Engineering ESEC/FSE*, 2015.

PLA-PMC Model Overview



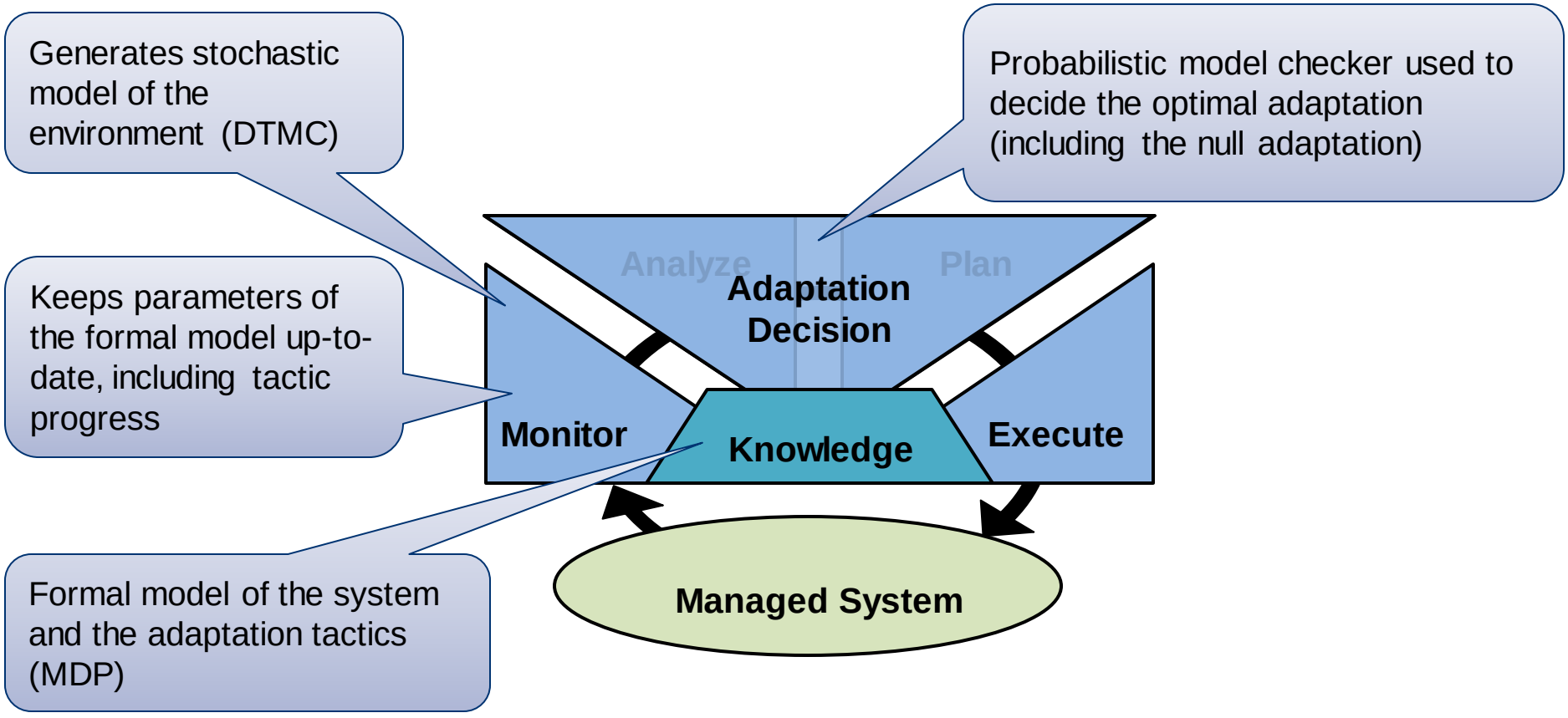
Key Idea

Leave the decision to start a tactic underspecified through nondeterminism, and have the model checker resolve the nondeterministic choices

module

shared action

PLA-PMC Adaptation Loop



Smart Cyber-Physical Systems

Carnegie Mellon University
Software Engineering Institute

Smart Cyber-Physical Systems

"Cyber-physical systems (CPS) are engineered systems that are built from, and depend upon, the seamless integration of computation and physical components." [NSF]

Smart Cyber-Physical Systems (sCPS)

Modern CPS need to be smart to be able to operate in dynamic and uncertain environments, be scalable, tolerate threats and perform as good as possible given a situation.

These needs can be fulfilled with self-adaptation techniques.

Challenges in Smart Cyber-Physical Systems

Carnegie Mellon University Software Engineering Institute

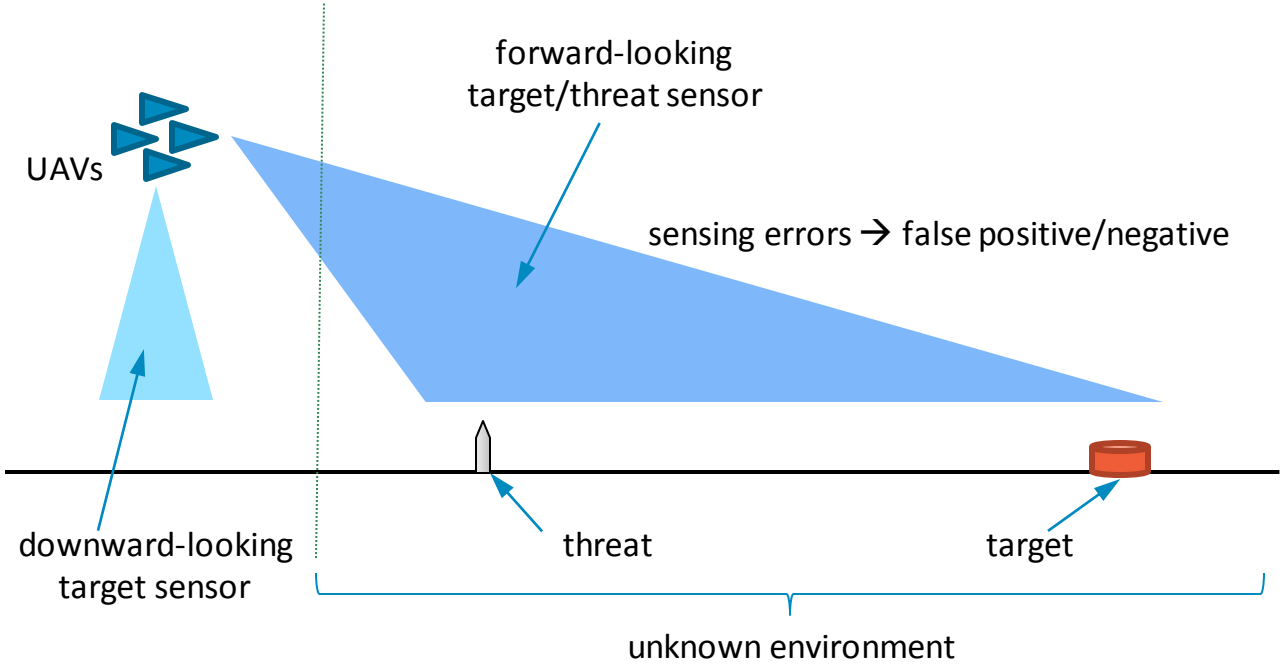
Self-adaptation for sCPS faces particular challenges:

- environment monitoring subject to sensing errors
  - e.g., noisy sensors, limited precision
- adaptation actions that take time
  - e.g., due to physical movement, sensors that recalibrate on power-up
- dire consequences for not adapting in a timely manner
  - e.g., physical damage to system or third party
- incomparable objectives that cannot be conflated into a single utility metric
  - e.g., avoiding an accident vs. providing good service

DARTSim

Simulated autonomous team of unmanned aerial vehicles (UAVs)

Reconnaissance mission in hostile and unknown environment: *fly predefined route to detect targets on the ground while avoiding threats*



*DART = Distributed Adaptive Real-Time

Moreno, G., C. Kinneer, A. Pandey, and D. Garlan. "DARTSim: An Exemplar for Evaluation and Comparison of Self-Adaptation Approaches for Smart Cyber-Physical Systems." In *International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. 2019.

Self-Adaptation in DARTSim

Adaptation Tactics

- climb/descend to change flight level (slow tactic)
- change formation loose/tight (fast tactic)

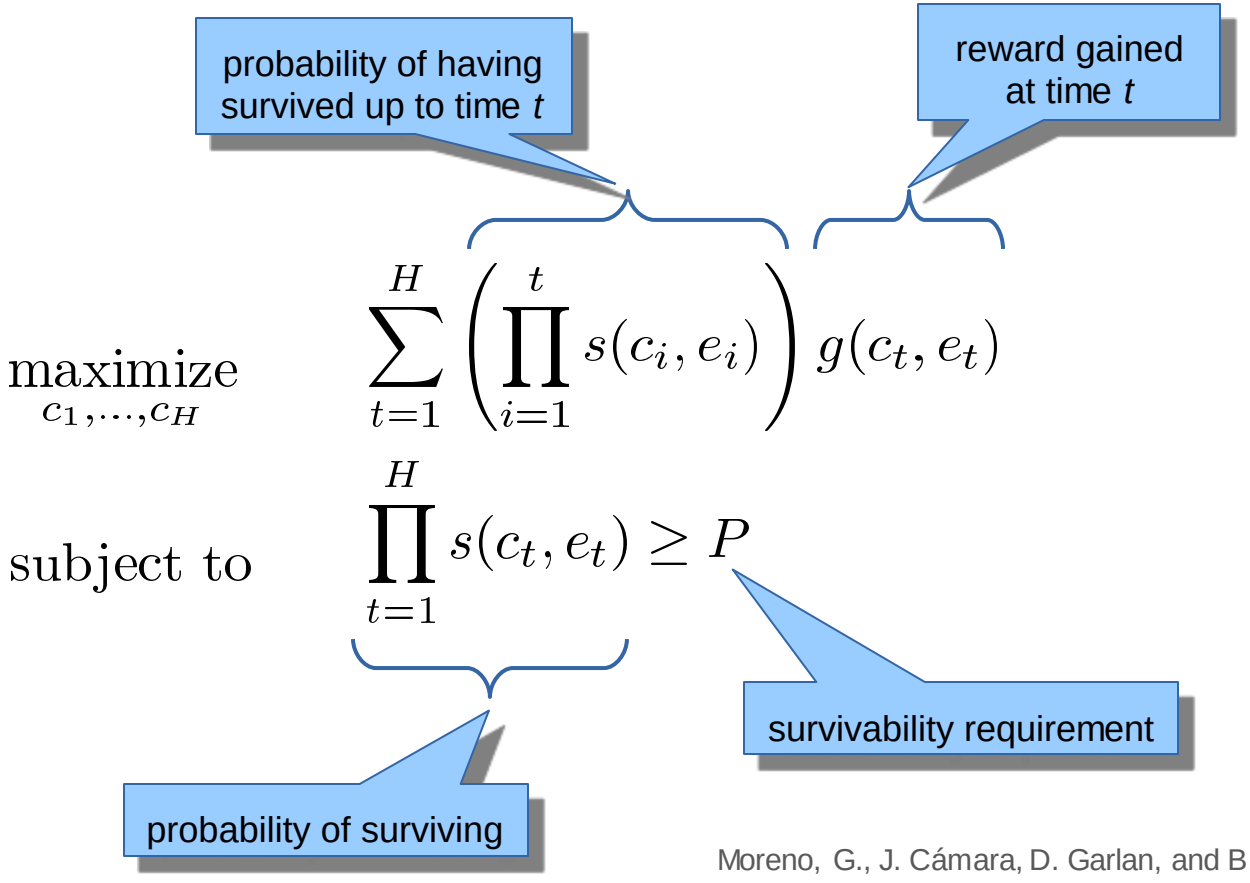
Adaptation Goal

- detect as many targets as possible
- survive the mission with probability >= 90%

Realization of sCPS Self-Adaptation Challenges

- Dynamically discovered and uncertain environment
  - Targets and threats localization/number unknown
- Forward-looking sensor is subject to errors
  - The sensor reduces uncertainty but does not eliminate it
- Adaptation actions that require large physical movement take time
  - Limits on climb and descend rate result in adaptation latency
- Not adapting in a timely manner may be catastrophic
  - Loss of UAVs to threat
- Mission survival and target detection maximization not trivial to combine
  - Not possible to assign value to targets to balance with survival (due to unknown number of targets)
  - Simply prioritizing survival results in very conservative behavior (not detecting enough targets)

A Different Form of Utility



Moreno, G., J. Cámara, D. Garlan, and B. Schmerl. "Flexible and Efficient Decision-Making for Proactive Latency-Aware Self-Adaptation." *ACM Transactions on Autonomous and Adaptive Systems.* 2018.

Concluding Remarks

Carnegie Mellon University Software Engineering Institute

Self-adaptation techniques endow systems with useful autonomous properties

- self-configuration, self-optimization, self-healing, self-*

Although originated in information systems, most current research efforts target cyber-physical systems.

Several decision-making approaches are available to tackle different adaptation decision problems.

Model-based automated analyses are key to realizing advanced forms of self-adaptation.