Towards A Holistic Software Systems Engineering Approach for Dependable Autonomous Systems

Authored by Aniculaesei, Adina et. al.
Presented by Samantha McDonald

PROBLEM DOMAIN

Autonomous systems!

Not reliable, or transparent.

Increasing in success, because of

- ★ The growing demand for smart, interconnected systems
- ★ Improvement in the area of data acquisition
- ★ Improvement of self-* (driving, steering, etc) features based on metadata extraction
- ★ Expansion of AI concepts through open-source frameworks

But, not secure!

THE ISSUES

SAFETY

Mitigation of personal danger when using autonomous systems

SECURITY

Reliability of a system, and ability to navigate failures

PRIVACY

Guarding of personal data and information

DEPENDABILITY

Safety, security, and privacy

SYSTEM DEVELOPMENT CHALLENGES

UN(CERTAIN/KNOWN) ENVIRONMENTS

Can't predict every encounter possible

SELF-LEARNING SYSTEMS

Adapt to a new problem domain, unpredictably

OPEN/INCOMPLETE ARTIFACTS

Stability in autonomous systems is not possible

Uncertain / Unknown Environments

Operational environments can't be fully defined

Correspondingly, all reactions can't be defined

Example: Tesla crashes into white truck in 2016, 2020

Self-learning Systems

Expected to adhere to their specifications

Also expected to adapt to new problem environments

Example: Microsoft's Twitter bot that was meant to learn to interact with users, learned to curse instead

Open / Incomplete Artifacts

An environment has infinite situations, that can't all be defined

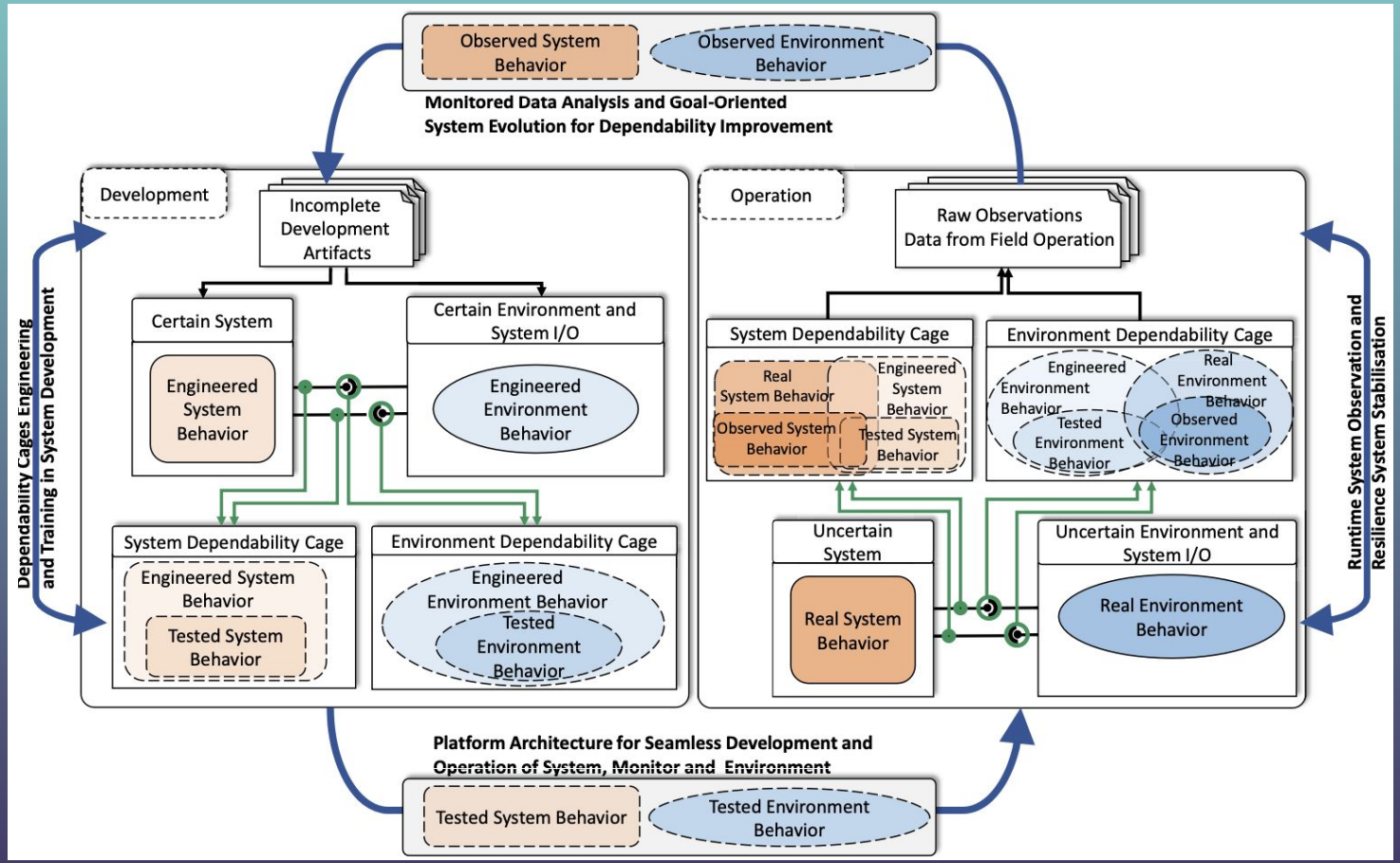
Systems will adapt to their situation, and are never stable

Systems can never be fully tested

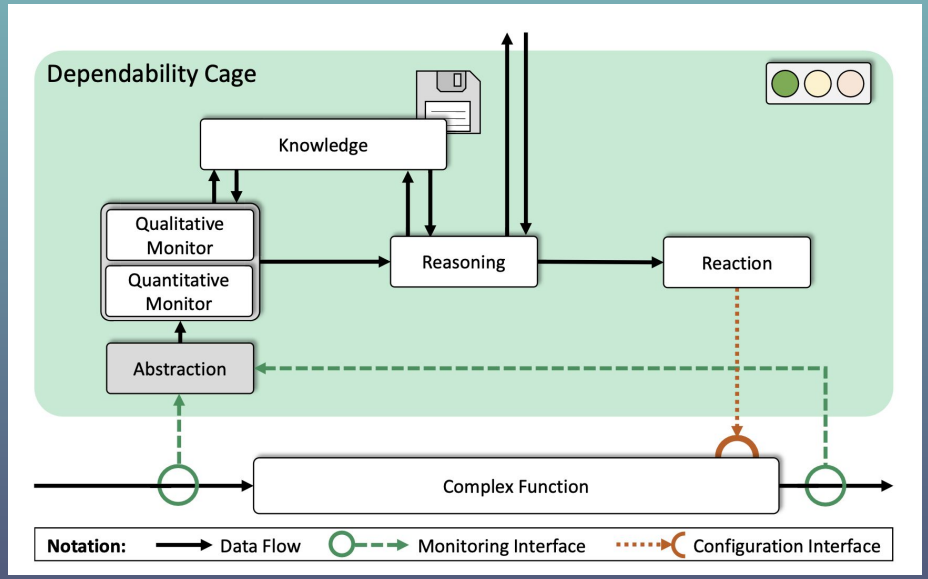
SOLUTION

Change the autonomous system development paradigm.
Rather than designing test environments, use **dependability cages**

DEPENDABILITY CAGE



DEPENDABILITY CAGE TYPES



01

INTERNAL SYSTEM

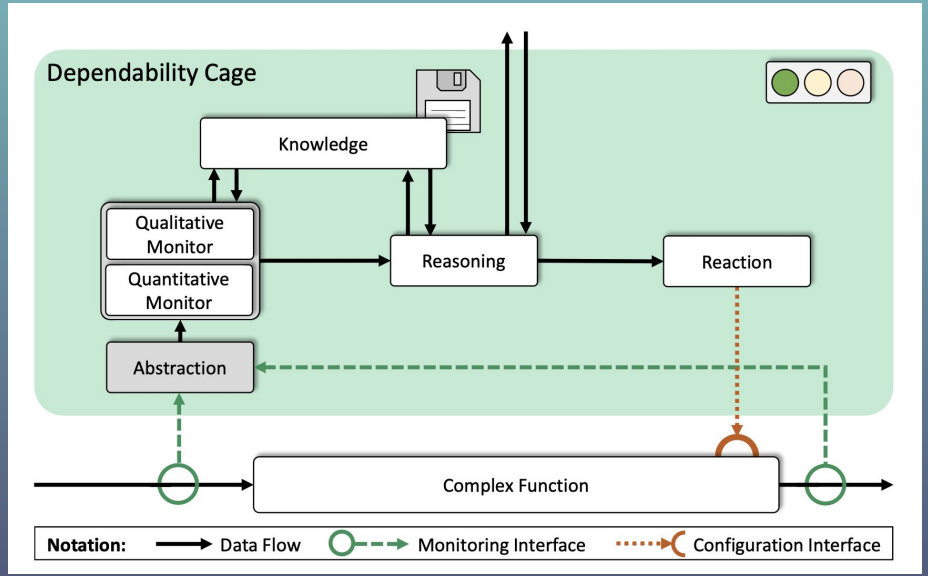
Ensure the behavior of the system components, system function, and the autonomous system as a whole
Brought up from the changing behavior of the internal system

02

EXTERNAL ENVIRONMENT

Ensure the environment in which the vehicle is operating in can be assessed.
Brought up from the uncertainties in the system's real environment

DEPENDABILITY CAGE DATA



01

QUALITATIVE DATA

* Compares the system's observed behavior to the tested one.
* Recognizes when calculations no longer provide specified outputs.
* Checks the functional correctness of the system.

02

QUANTITATIVE DATA

*Detects if the context of the situation is potentially unsafe or untested
*Verifies if the system is operating in a secure environmental context

RESILIENCE SYSTEM STABILIZATION

* Untested environments will be encountered, findings must be fed back into development artifacts

* Dependability cages attempt to ensure that the observed behavior stays inside the tested behavior

* Combining data collected from a variety of individual widespread dependability cages / data sources ⟶ improved decision making

SYSTEM DEVELOPMENT ARCHITECTURE IDEALS

* Software architecture must recognize that
    system must be in a safe state at all times
    system must be able to adapt to new situations at operation time

* Dependability cages should be integrated
    without problems or side-effects
    with a fallback if observed conditions can't be handled

* As test and operation modes are distinct, switching should be done between simulated / test hardware and real environments

FUTURE WORK / CONCLUSIONS

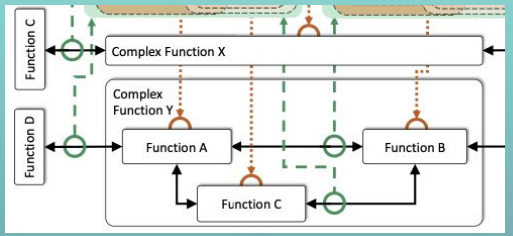
FUTURE WORK

Integrated dependability cages (eg. sensors, functions, and actuators for embedded systems)

CONCLUSIONS

Introduced concept of dependability cages examine runtime behavior, and analyze new samples

MY THOUGHTS



- "Dependability cage" is never given a clear definition
- Half of the paper is spent on background information
- Some/most concepts described are already in use
    - Using collected data to re-train models
    - Define operational bounds to test within
- Unlikely that a single dependability approach will be used in all autonomous systems

Thanks for Listening!

Any Questions?

DISCUSSION

- Should there be a standard to determine or ensure the trustworthiness of a system?
- How can / should such a system be developed?
- What are the drawbacks of such a system?

slidesgo