High Assurance for Distributed Cyber Physical Systems
Architecting Self-Managing Distributed Systems (ECSA/ASDS) Workshop

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Scott Hissam
September 7, 2015
DART Driving Vision

DARTs coordinate physical agents in an uncertain and changing physical world.

- Coordination – physical agents
- Timeliness – safety critical
- Resource constrained - UAVs
- Sensor rich – sensing physical world
- Intimate cyber physical interactions
- Automated adaptation to physical context
- Computationally complex decisions

Coordination, adaptation, and uncertainty pose key challenges for assuring safety and mission critical behavior of distributed cyber-physical systems.

The DART project uses develops and packages sound techniques and tools for engineering high-assurance distributed CPS.

* DART = Distributed Adaptive Real-Time
Our Current Unified Motivating Scenario

Objectives
- Search and rescue for groups of UAVs
- Protection among important assets by groups of UAVs

Threat Models, Challenges and Features
- Disrupted Communications
- Obstacles
- Emergent threats
- Data, Knowledge and Processing Overload
Application of Research to Motivating Scenario

Design Time Verification
- Guaranteed behavior
- Best-effort behavior

Autonomy Implementation
- Real-time reasoning, timing and control
- Networking
- Quality-of-service

Runtime Assurance
- Critical Timing behavior
- Coordination
- Adaptation
Engineering High Assurance for DART

- Requirements properties
- Language to specify design and behavior model
- Tools, techniques to analyze critical properties code & proofs
- Code generation runtime
- Binary code deployment

Design constraints for tractable analysis

Intent

Semantic precision

Evidence

Assurance

Traceability
DART High-Level Architecture

Software for guaranteed requirements, e.g., collision avoidance protocol must ensure absence of collisions

Software for probabilistic requirements, e.g., adaptive path-planner to maximize area coverage within deadline

High-Critical Threads (HCTs)

Low-Critical Threads (LCTs)

Environment – network, sensors, atmosphere, ground etc.

Research Thrusts
• Proactive Self-Adaptation
• Statistical Model Checking
• Real-Time Schedulability
• Functional Verification

Validation Thrusts
• Model Problem
• Workbench

Node_1

Node_k
DART Tooling and Techniques

- System model + requirements + environment
- Specification
- Verification
- Generation
- Collaborating and coordinating agents
- Timing & functional behavior & probabilistic behavior
- Traceable transformations
- Comms middleware + scheduler + monitors + adaptations
- Constituent agent
- Agent’s constrained runtime environment

High Assurance for Distributed Cyber Physical Systems
Hissam 7-SEP-2015
© 2015 Carnegie Mellon University
Research - Deterministic Behavior

Real-Time Schedulability

• Technique to guarantee deadlines among tasks with different semantic criticalities in a rate-monotonic scheduler.

Challenges:

• Mixed-Criticality Scheduling under I/O
• End-to-end Mixed-Criticality Scheduling

Functional Verification

• Technique to ensure the behavior of a distributed application that satisfies a user-specified safety specification.

Challenges:

• Unbounded Model Checking Synchronous Software
• Unbounded Model Checking Asynchronous Software

Discharges Assumptions Needed for Correctness


Research - Probabilistic Behavior

Statistical Model Checking
- Technique to compute the bounded probability that a specific event occurs during a stochastic system's execution.

Challenges:
- Importance Sampling with Heterogenous Fault Regions
- Statistical Model Checking for systems with non-determinism

Proactive Self-Adaptation
- Technique for a system to adapt to an upcoming situation given that time is needed to perform the adaptation.

Challenges:
- Adaptation decision under uncertainty
- Integration with Machine Learning Techniques


Validation Thrusts

DART Model Problem
- High assurance multi-agent DART prototype integrating deterministic and probabilistic verification techniques.

Challenges:
- Quantifying mission impact
- Spanning the gap between “the lab” and real world

DART Workbench
- Create an integrated engineering approach for developing DART systems through specifications and tooling.

Challenges:
- Semantically precise specifications
- end-to-end traceability
DART Model Problem – Elements of Analysis

- Guaranteed separation among multiple agents (MA) through compositional model checking
- Best-effort confidence in adherence to MA formation through statistical model checking
- Increased survivability of MA system from threats through proactive self-adaptation
- Coordination of sensor functions among MA with end-to-end latency (fy16)
- Coordination of mission objectives between MA systems (fy16)

Guarantee timing behavior of highly critical threads through mixed-criticality temporal protection mechanisms
DART Workbench Overview

- Specification
- Verification
- Generation

- System model + requirements + environment
- Traceable transformations
- Collaborating and coordinating agents
- Timing & functional behavior & probabilistic behavior
- Comms middleware + scheduler + monitors + adaptations

- Agent’s constrained runtime environment
- Constituent agent

- MADARA
- Sched
- OS/HW
DART Workbench Details

system model + requirements + environment

AADL, OSATE and DMPL specifications

traceable transformations

ZSRM CBMC Osmosis

timing & functional behavior & probabilistic behavior

comms middleware + scheduler + monitors + adaptations

C++ MADARA VREP

constituent Odroid U3 agent

agent’s constrained Linux/ARM runtime environment

collaborating and coordinating Odroid U3 agents

DART Workbench
DART Workbench Usage

Node Specification in a DSL

```plaintext
@HERTZ(8)
@CRITICALITY(HIGH)
@WCET_NOMINAL(2.5)
@WCET_OVERLOAD(5.0)
@BARRIER_SYNC
... void collision_avoid() {
    // Operates on X & Y
}
...
require(FORALL_NODE_PAIR
    (id1, id2,
     x@id1 != x@id2 ||
     y@id1 != y@id2));
require(InBounds(X,Y);
...
@AT_LEAST(0.8)
expect(COVER() >= 0.9)
else {
    // Adapt
};
...
```

Analysis & Verification

```plaintext
T: <other>
Period: 4
C_nom: 2.0
C_over: 2.0
Crit: Low
Pri: High
T: coll_av
Period: 8
C_nom: 2.5
C_over: 5.0
Crit: High
Pri: Low

read shared context
ASSUME (local constraints)
do collision_avoid()
ASSERT (local changes)
write shared context
log (COVER() variables)
Do collision_avoid()
    *multiple repetitions
    Perform offline statistical analysis of logged data
```

Target Code Gen.

```plaintext
attr.period_msec = 125;
attr.Cmon_msec = 2.5;
attr.Cover_msec = 5.0;
attr.criticality = HIGH;
attr.zs_instance_nsec = zsinst[“coll_avoid”];
zs_reserve(&attr);

int loc_X = ShrRead(X);
int loc_Y = ShrRead(Y);

// Do coll_avoid logic
if(!(InBounds(loc_X, loc_Y))
    // Handle Fault
    AdaptManager(COVER());
    ShrWrite(X).set(loc_X);
    ShrWrite(Y).set(loc_Y);
```

Software Engineering Institute | Carnegie Mellon University

High Assurance for Distributed Cyber Physical Systems
Hissam 7-SEP-2015
© 2015 Carnegie Mellon University
DART Workbench Screenshot

http://cps-sei.github.io/dart/
DART Demonstration Infrastructure

Odroid U3s
- Linux-RK/ARM
- ZSRM
- WiFi

Laptop
- VREP
- Logging
- Ethernet

- Odroid U3 Board
- WiFi Connectivity Between Odroids
- Ethernet connectivity with V-Rep
- Ethernet Switch
Team

Bjorn Andersson
Mark Klein
Bud Hammons
Arie Gurfinkel
Gabriel Moreno
David Kyle
Jeff Hansen
James Edmondson

Scott Hissam
Dionisio de Niz
Sagar Chaki
Contact Information Slide Format

Presenter / Point of Contact
Scott Hissam
Telephone: +1 412-268-6526
Email: shissam@sei.cmu.edu

U.S. Mail
Software Engineering Institute
Customer Relations
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

Web
www.sei.cmu.edu
www.sei.cmu.edu/contact.cfm

Customer Relations
Email: info@sei.cmu.edu
Telephone: +1 412-268-5800
SEI Phone: +1 412-268-5800
SEI Fax: +1 412-268-6257