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

- intent
- semantic precision
- evidence
- assurance

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

Environment – network, sensors, atmosphere, ground etc.

High-Critical Threads (HCTs)  Low-Critical Threads (LCTs)

MADARA Middleware
ZSRM Mixed-Criticality Scheduler
OS/Hardware

Node$_1$

Node$_k$

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

Validation Thrusts
- Model Problem
- Workbench
DART Tooling and Techniques

- Specification
- Verification
- Generation

- System model + requirements + environment
- Timing & functional behavior & probabilistic behavior
- Collaborating and coordinating agents

- Traceable transformations
- Comms middleware + scheduler + monitors + adaptations

- Agent’s constrained runtime environment
- Constituent agent

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

Discharges Assumptions Needed for Correctness

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


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

Measures the Effectiveness of Adaptation


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

Drives Implementation
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
- timing & functional behavior & probabilistic behavior
- collaborating and coordinating agents
- comms middleware + scheduler + monitors + adaptations
- constituent agent
- agent’s constrained runtime environment

- traceable transformations
- constituent agent
- MADARA
- Sched
- OS/HW
DART Workbench Details

- AADL, OSATE and DMPL specifications
- ZSRM, CBMC, Osmosis
- C++, MADARA, VREP
- System model + requirements + environment
- Timing & functional behavior & probabilistic behavior
- Traceable transformations
- Comms middleware + scheduler + monitors + adaptations
- Collaborating and coordinating Odroid U3 agents
- Constituent Odroid U3 agent
- Agent’s constrained Linux/ARM runtime environment
- DART Workbench
DART Workbench Usage

Node Specification in a DSL

```c
@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

```c
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.

```c
attr.period_msec = 125;
attr.Cmon_msec = 2.5;
attr.Cover_msec = 5.0;
attr.criticality = HIGH;
attr.zs_instant_nsec = zsinst["coll_avoid"];
zs_reserve(&attr);

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

// Do collision_avoid logic
if(!(InBounds(loc_X, loc_Y))
    // Handle Fault
    AdaptManager(COVER());

ShrWrite(X).set(loc_X);
ShrWrite(Y).set(loc_Y);
```
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 Switch
Ethernet connectivity with V-Rep

Laptop running V-Rep simulator
## Contact Information Slide Format

<table>
<thead>
<tr>
<th>Presenter / Point of Contact</th>
<th>U.S. Mail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scott Hissam</td>
<td>Software Engineering Institute</td>
</tr>
<tr>
<td>Telephone: +1 412-268-6526</td>
<td>Customer Relations</td>
</tr>
<tr>
<td>Email: <a href="mailto:shissam@sei.cmu.edu">shissam@sei.cmu.edu</a></td>
<td>4500 Fifth Avenue</td>
</tr>
<tr>
<td></td>
<td>Pittsburgh, PA 15213-2612</td>
</tr>
<tr>
<td></td>
<td>USA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Web</th>
<th>Customer Relations</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.sei.cmu.edu">www.sei.cmu.edu</a></td>
<td>Email: <a href="mailto:info@sei.cmu.edu">info@sei.cmu.edu</a></td>
</tr>
<tr>
<td><a href="http://www.sei.cmu.edu/contact.cfm">www.sei.cmu.edu/contact.cfm</a></td>
<td>Telephone: +1 412-268-5800</td>
</tr>
<tr>
<td></td>
<td>SEI Phone: +1 412-268-5800</td>
</tr>
<tr>
<td></td>
<td>SEI Fax: +1 412-268-6257</td>
</tr>
</tbody>
</table>