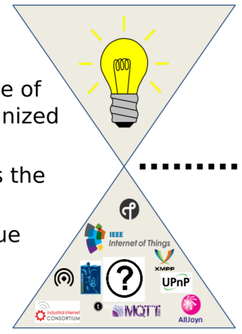
Building IoT Applications with Accessors in CapeCode

Marten Lohstroh Ilge Akkaya Christopher Brooks Edward A. Lee **UC Berkeley**

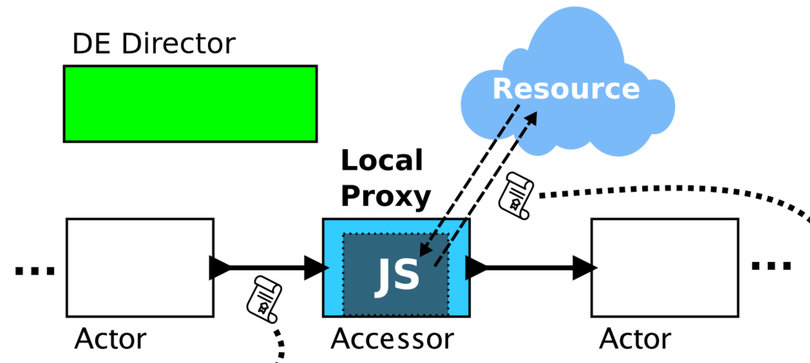
ICCPS 2016

http://terraswarm.org/

Composability in the IoT



- IoT faces **heterogeneity**, an overwhelming diversity in device characteristics, and a jungle of interfaces, APIs, and middle ware, but all organized in disjoint "ecosystems."
- The de facto standard for device interaction is the "app," but apps are **not composable**.
- **Composability** is the key to unlocking the true potential of the Internet of Things beyond isolated and proprietary "smart islands."
- What is missing is a universal platform for composition that **backward *and* forward compatible**.



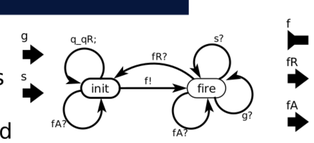
Accessors

- Accessors encapsulate "things" and services by endowing them with an **actor interface**;
  - They embrace the vast heterogeneity of the IoT space.
- This pattern provides a universal **platform for composition**;
  - It does not impose any over-the-wire standards.
- We leverage disciplined **models of computation** to facilitate analyzability and improve predictability;
  - The approach is flexible enough to embed a multitude of interaction paradigms such as: REST, RPC, Publish-subscribe...
- Accessors allow breaking up the classic divide between **"design time"** and **"run time"**;
  - They enable integration of deployed and simulated subsystems.

Host Capabilities

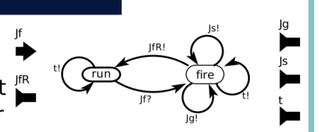
- The execution platform for accessors is called the **host**.
- The host has capabilities that are exposed through the **modules** it implements.
- An accessor may **require** particular modules in order to function.
- Different implementations of the host are underway:
  - Ptolemy II, Node.js, Web browser, Vert.x, Duktape

Horizontal Contract



- An **actor abstract semantics** specifies the behavior of the accessor as observed by other actors that it may be composed with. Interface functions: *initialize()*, *fire()*, *wrapup()*.
- Other aspects to consider:
  - Ordering of requests and responses;
  - Deadlines, timeouts;
  - Responses to internal faults.

Vertical Contract



- Interactions with the run time environment that lets the accessor connect to things or services occur through **asynchronous atomic callbacks (AACs)**.
- Other aspects to consider:
  - Throttling, network delays, and buffer sizes;
  - Fault handling, adapting to changing environmental circumstances.

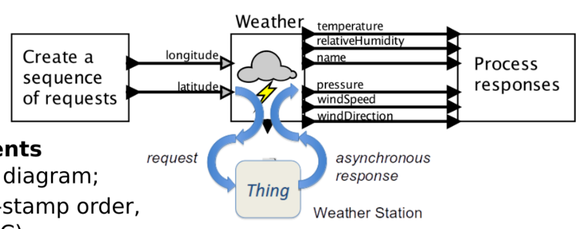
JavaScript

Easily-adapted, interpreted, and sand-boxed code.

```
// Import a module providing network services
var http = require("http");
// Construct a URL encoding a request
var url = "http://foo.com/deviceID/...";
// Issue the request and provide a callback
http.get(url, function(response) {
    // ... handle the response ...
});
```

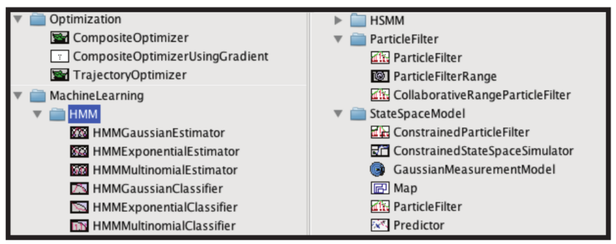
CapeCode

- CapeCode is a host based on the actor-oriented modeling environment **Ptolemy II**, it:
  - Provides a rich **library of functional components** that can be composed with accessors in a block diagram;
  - Uses **time-stamped events** processed in time-stamp order, a discrete-event (DE) model of computation (MoC);
  - Features a **graphical user interface** for wiring together actors and accessors in a block diagram;
  - Is **under development,** and progress is made toward: *code generation for small-footprint platforms, virtual devices that model resources, aspect-oriented resource assignment, and resource discovery.*



With accessors, CapeCode embraces concurrency, asynchrony, and atomicity.

Plug & Play Machine Learning: PILOT



Ptolemy **I**nference, **L**earning, and **O**ptimization **T**oolkit

- As a part of the CapeCode component library, PILOT offers a range of machine learning tools:
  - Bayesian inference. Hidden Markov Model and Gaussian Mixture Model based inference: parameter estimation and classification;
  - Constrained optimization;
  - State estimation, particle filtering;
  - Model-predictive control.

Sponsored by the TerraSwarm Research Center, one of six centers administered by the STARnet phase of the Focus Center Research Program (FCRP) a Semiconductor Research Corporation program sponsored by MARCO and DARPA.