A Component Architecture for the Internet of Things

Written by: Christopher Brooks, Chadlia Jerad, Hokeun Kim, Edward A. Lee, Marten Lohstroh, Victor Nouvellet, Beth Osyk, and Matt Weber

Presented by: Eric Wendt

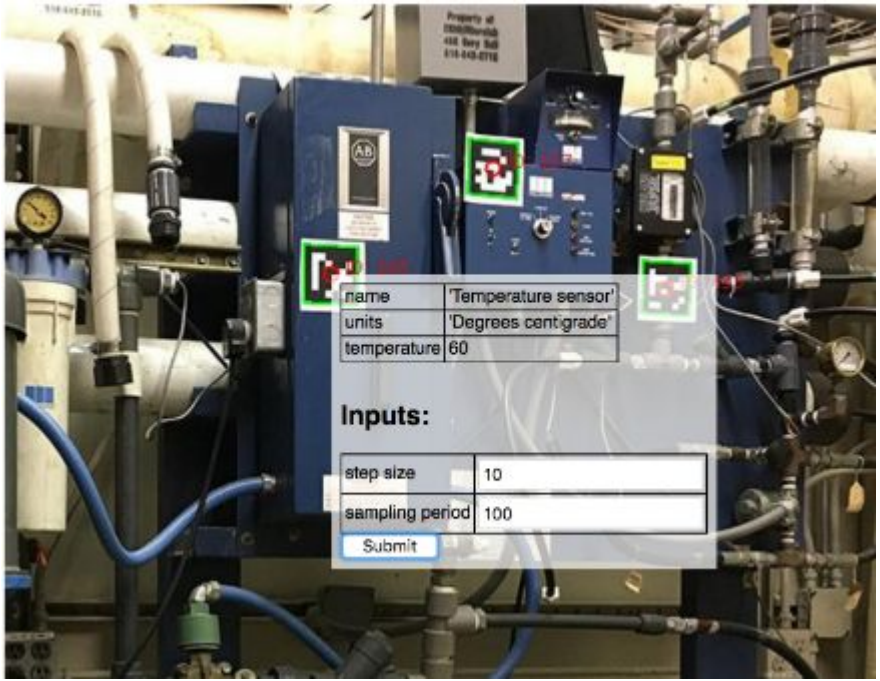
CapeCode - The Purpose

- Provides a level of determinism to counter unpredictable latencies
- Intricate IoT systems can be ‘modularized’
- State-flow does not have to be written by one vendor

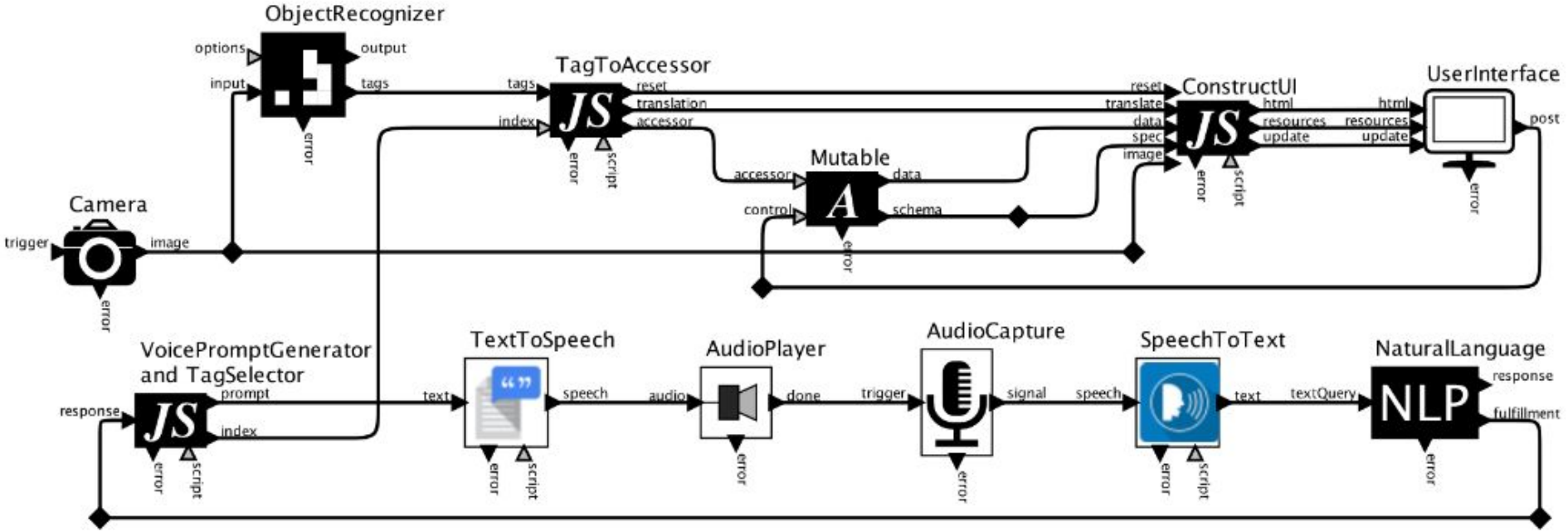
The Example

The Example

AR goggles show heads-up display



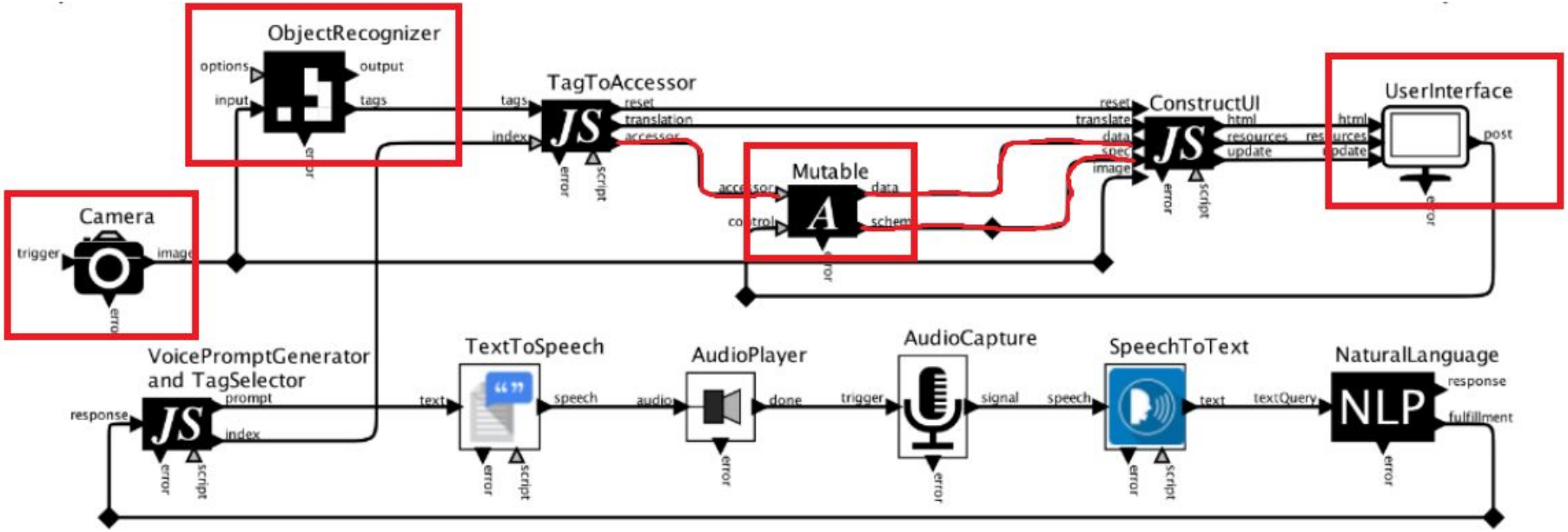
The System


ObjectRecognizer
options
output
input
tags
error
TagToAccessor
tags
reset
translation
index
accessor
error
script
Mutable
accessor
data
control
schema
error
ConstructUI
reset
translate
data
spec
image
html
resources
update
error
script
UserInterface
html
resources
update
post
error
Camera
trigger
image
error
VoicePromptGenerator
and TagSelector
response
prompt
index
error
script
TextToSpeech
text
speech
error
script
AudioPlayer
audio
done
error
AudioCapture
trigger
signal
error
SpeechToText
speech
text
error
script
NaturalLanguage
textQuery
response
fulfillment
NLP
error

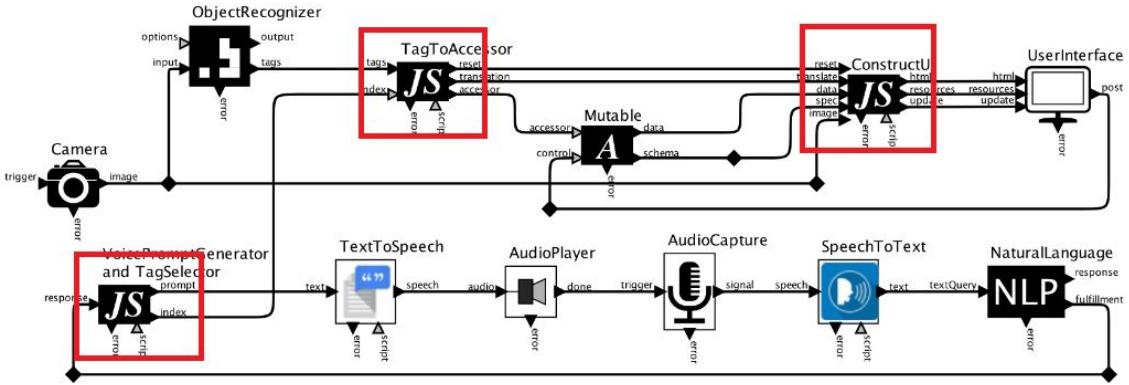
Assessors

- Code for processing and transforming data
- proxies/services that interact with IoT devices, strictly NOT human users.
- The “reusable” component of the architecture

Assessors


ObjectRecognizer
options
output
input
tags
error
TagToAccessor
tags
reset
translation
index
accessor
error
script
Mutable
accessor
data
control
schem
error
ConstructUI
reset
translate
data
spec
image
html
resources
update
error
script
UserInterface
html
resources
update
post
error
Camera
trigger
image
error
VoicePromptGenerator
and TagSelector
response
prompt
index
error
script
TextToSpeech
text
speech
error
script
AudioPlayer
audio
done
error
AudioCapture
trigger
signal
error
SpeechToText
speech
text
error
script
NaturalLanguage
textQuery
NLP
response
fulfillment
error

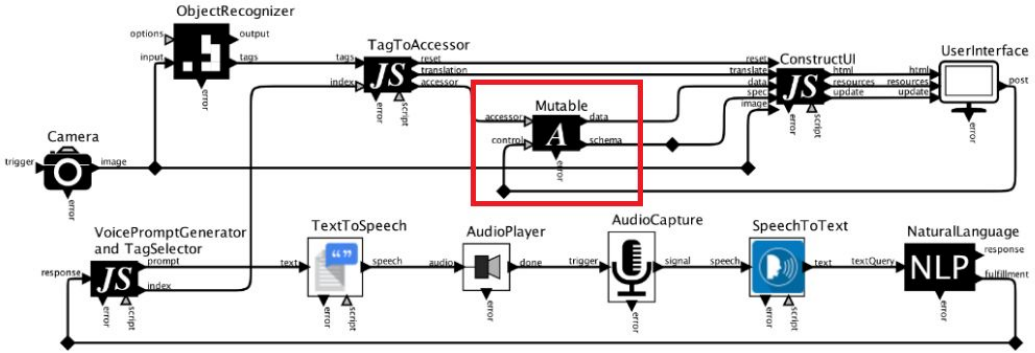
JS nodes



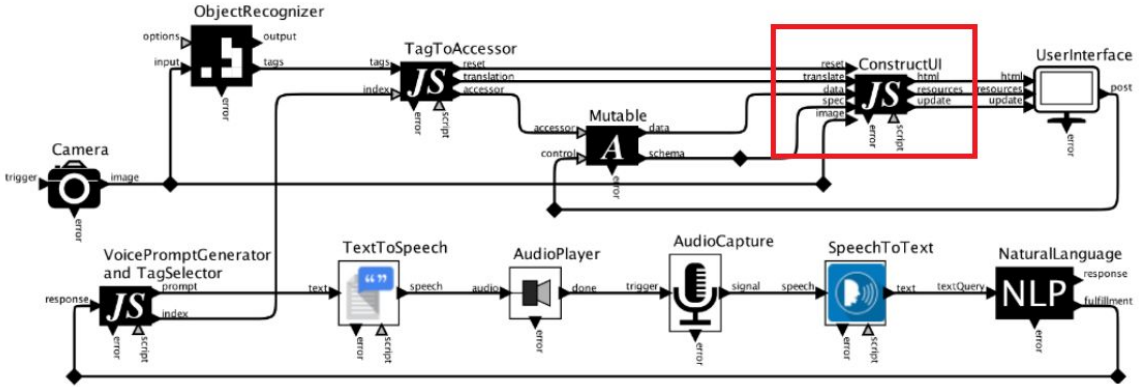
- Non-reusable scripts written in JavaScript
  - Not portable between vendors
  - Device/hardware-specific

Mutable

- Source code “template” for accessor
- Can be thought of as an abstract class
- When an accessor is passed to the Mutable node, it is reified and replaces it.



Endpoint



- The reified accessor is passed to another human-interactive component, such as the UI
- Any further input or data can be forwarded back to the accessor
- ConstructUI is JUST data and JS code, NOT an accessor

Discrete-Event Systems

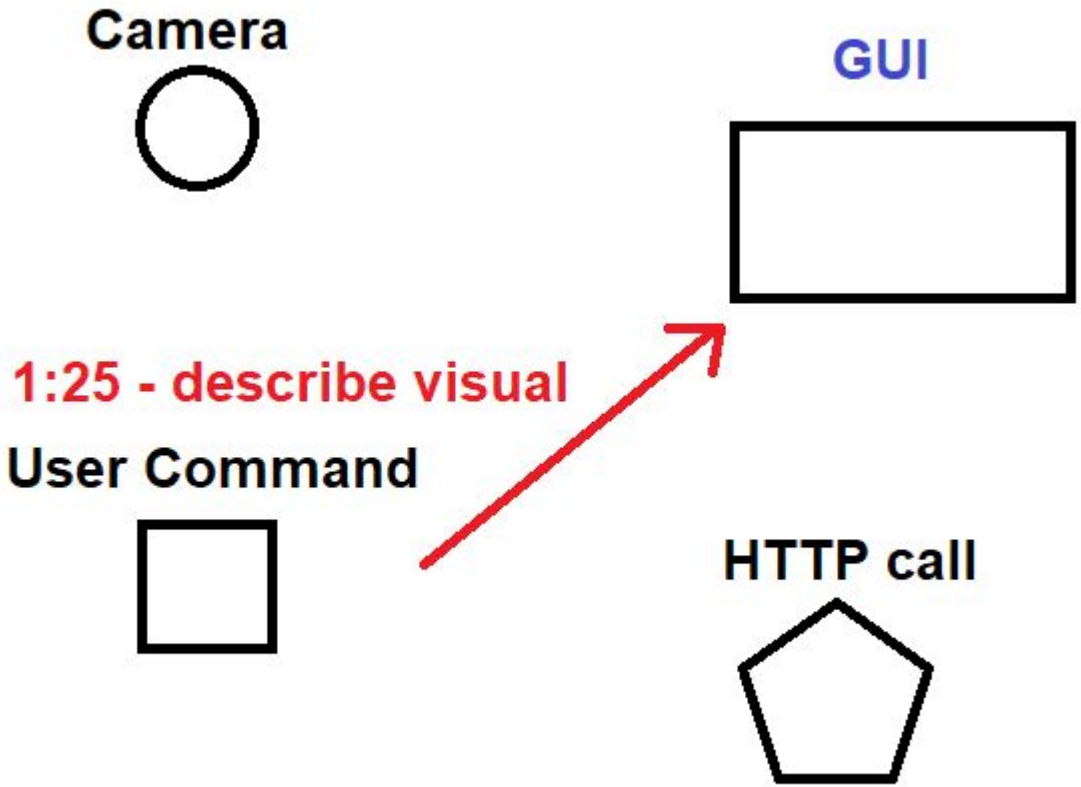
- Goal is to reduce non-determinism
- Relies on a system of timestamps to determine flow of execution
- Coupled with atomic callbacks, provides ease of programming and scalability

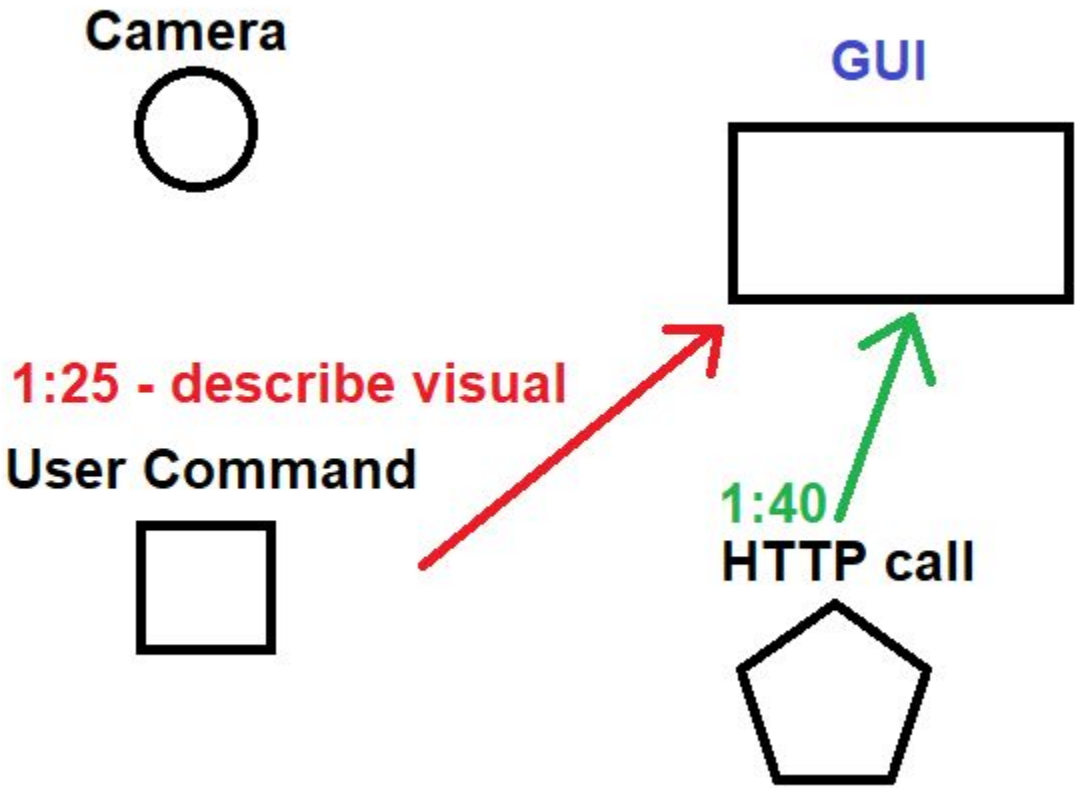
Camera

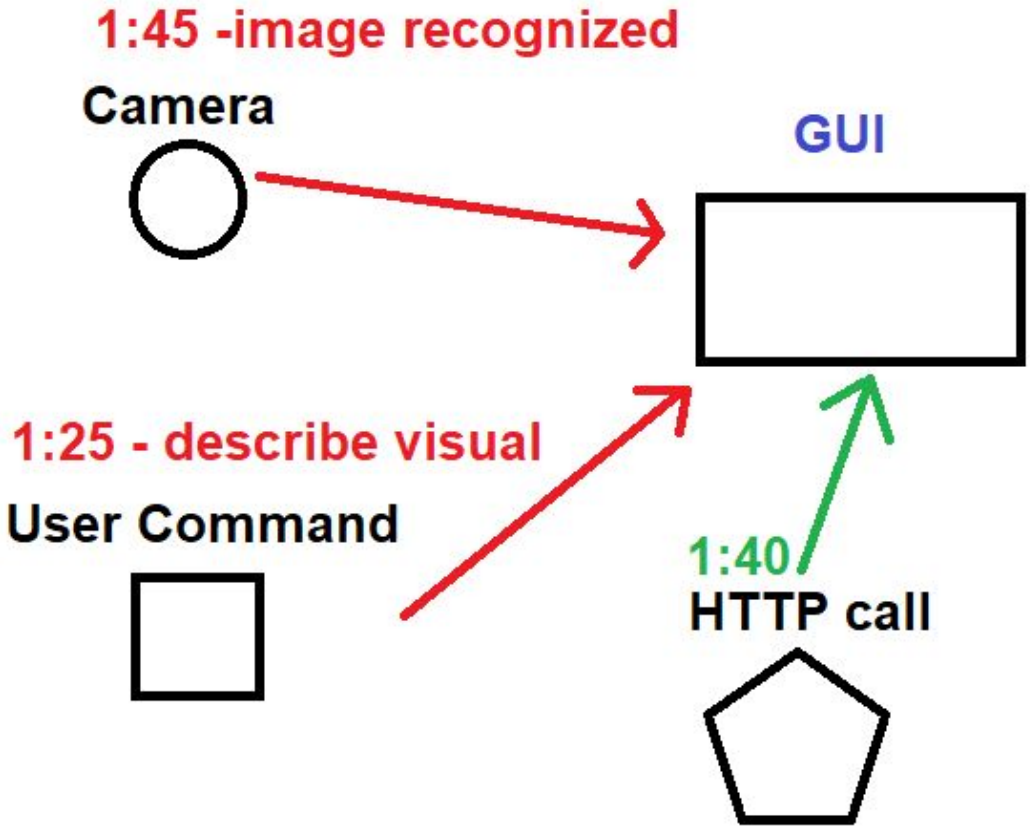
GUI

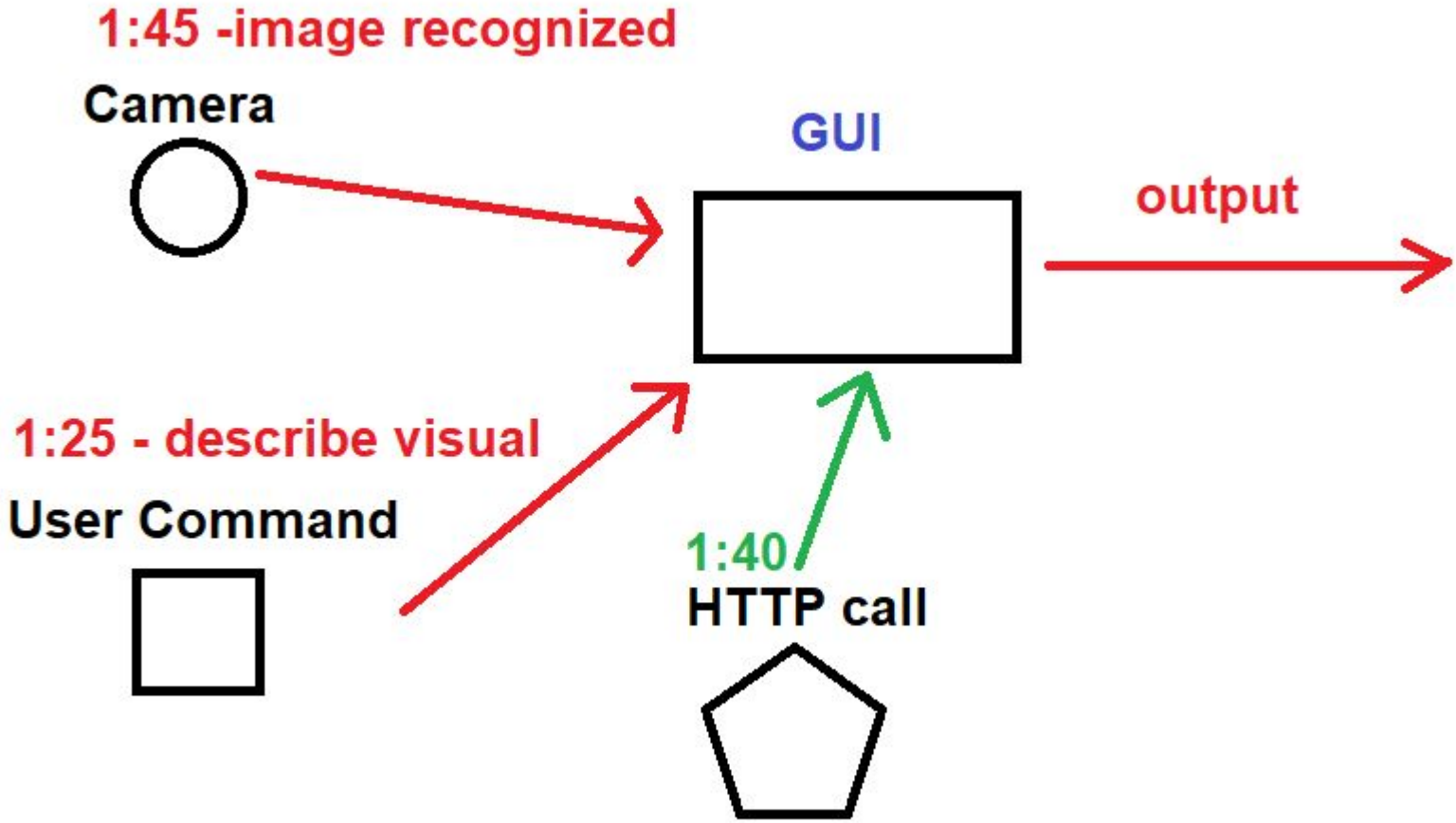
User Command

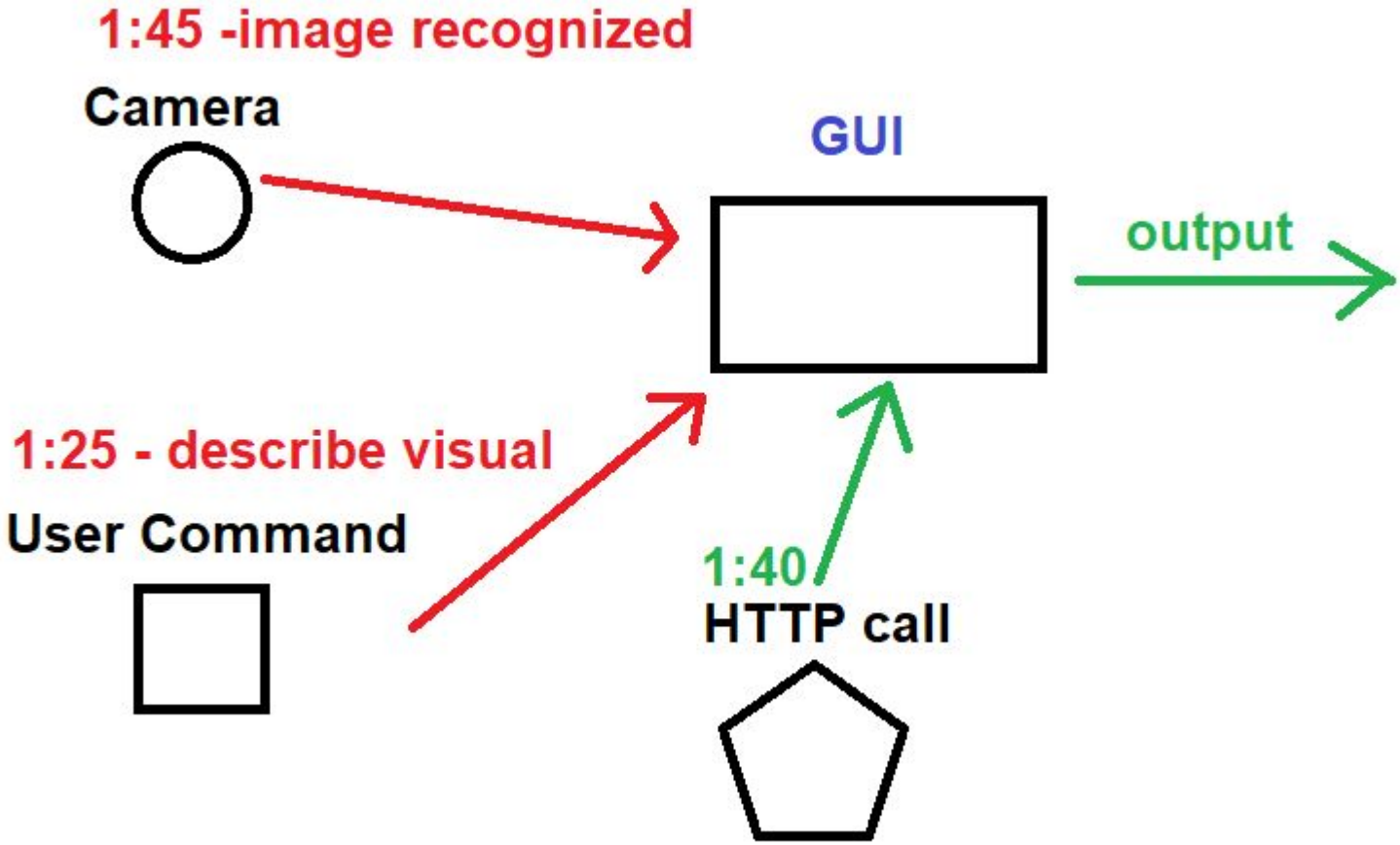
HTTP call


Camera
GUI
1:25 - describe visual
User Command
HTTP call


Camera
GUI
1:25 - describe visual
User Command
1:40
HTTP call

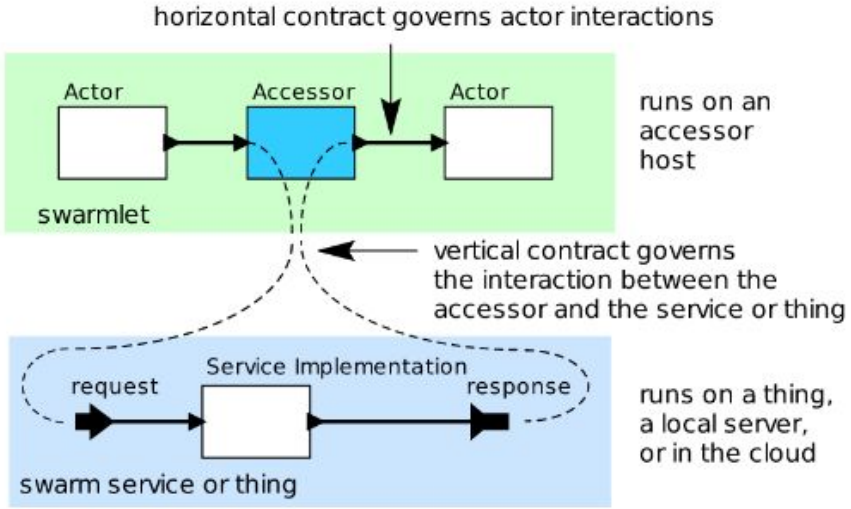

1:45 -image recognized
Camera
GUI
1:25 - describe visual
User Command
1:40
HTTP call


1:45 -image recognized
Camera
GUI
output
1:25 - describe visual
User Command
1:40
HTTP call


1:45 -image recognized
Camera
GUI
output
1:25 - describe visual
User Command
1:40
HTTP call

Contracts

- Think of contracts as APIs
- Actors are data components that communicate with a more generic Accessor -> **horizontal contract**
- The Accessor then communicates with a “Thing.” -> **vertical contract**
- Could be any service or physical device.
- Vertical contract is device-specific



Asynchronous Atomic Callbacks

- Non-blocking behaviour
- Services can activate callback functions atomically at their leisure
- Functions cannot activate while another is active

Callback functions

```
exports.setup = function() {
  this.input('trigger');
  this.output('data', {
    'type': 'JSON',
    'spontaneous': true
  });
}
var httpClient = require('http-client');

exports.initialize = function() {
  var self = this;
  this.addInputHandler('trigger', function() {
     httpClient.get(
          'http://accessors.org',
          function(data) {
            self.send('data', JSON.parse(data.body));
          }
     );
  });
}
```

Callback functions

```
exports.setup = function() {
  this.input('trigger');
  this.output('data', {
    'type': 'JSON',
    'spontaneous': true
  });
}
var httpClient = require('http-client');

exports.initialize = function() {
  var self = this;
  this.addInputHandler('trigger', function() {
     httpClient.get(
         'http://accessors.org',
         function(data) {
           self.send('data', JSON.parse(data.body));
         }
     );
  });
}
```

Callback functions

```
exports.setup = function() {
  this.input('trigger');
  this.output('data', {
    'type': 'JSON',
    'spontaneous': true
  });
}
var httpClient = require('http-client');

exports.initialize = function() {
  var self = this;
  this.addInputHandler('trigger', function() {
     httpClient.get(
          'http://accessors.org',
          function(data) {
            self.send('data', JSON.parse(data.body));
          }
     );
  });
}
```

Callback functions

```
exports.setup = function() {
  this.input('trigger');
  this.output('data', {
    'type': 'JSON',
    'spontaneous': true
  });
}
var httpClient = require('http-client');

exports.initialize = function() {
  var self = this;
  this.addInputHandler('trigger', function() {
     httpClient.get(
          'http://accessors.org',
          function(data) {
            self.send('data', JSON.parse(data.body));
          }
     );
  });
}
```

Callback functions

```
exports.setup = function() {
  this.input('trigger');
  this.output('data', {
    'type': 'JSON',
    'spontaneous': true
  });
}
var httpClient = require('http-client');

exports.initialize = function() {
  var self = this;
  this.addInputHandler('trigger', function() {
     httpClient.get(
          'http://accessors.org',
          function(data) {
            self.send('data', JSON.parse(data.body));
          }
     );
  });
}
```

Callback functions

```
exports.setup = function() {
  this.input('trigger');
  this.output('data', {
    'type': 'JSON',
    'spontaneous': true
  });
}
var httpClient = require('http-client');

exports.initialize = function() {
  var self = this;
  this.addInputHandler('trigger', function() {
    httpClient.get(
        'http://accessors.org',
        function(data) {
          self.send('data', JSON.parse(data.body));
        }
    );
  });
}
```

What’s the point?

- Workflow of many Iot devices can be standardized in a high-level framework.
  - Better outreach to developers
- Ensures determinism, multithreading and interrupts do not exist.
- Provides bridge between device-specific code and a general data flow model.

Security

- Accessors are untrusted code
- They run in a virtual environment usually in a browser
- Secure Swarm Toolkit uses Auth on edge devices to handle authentication and authorization

Critiques

- Robustness to network failures is not guaranteed.
- Discrete Events seem less efficient that a standard multithreading model.
- Few examples

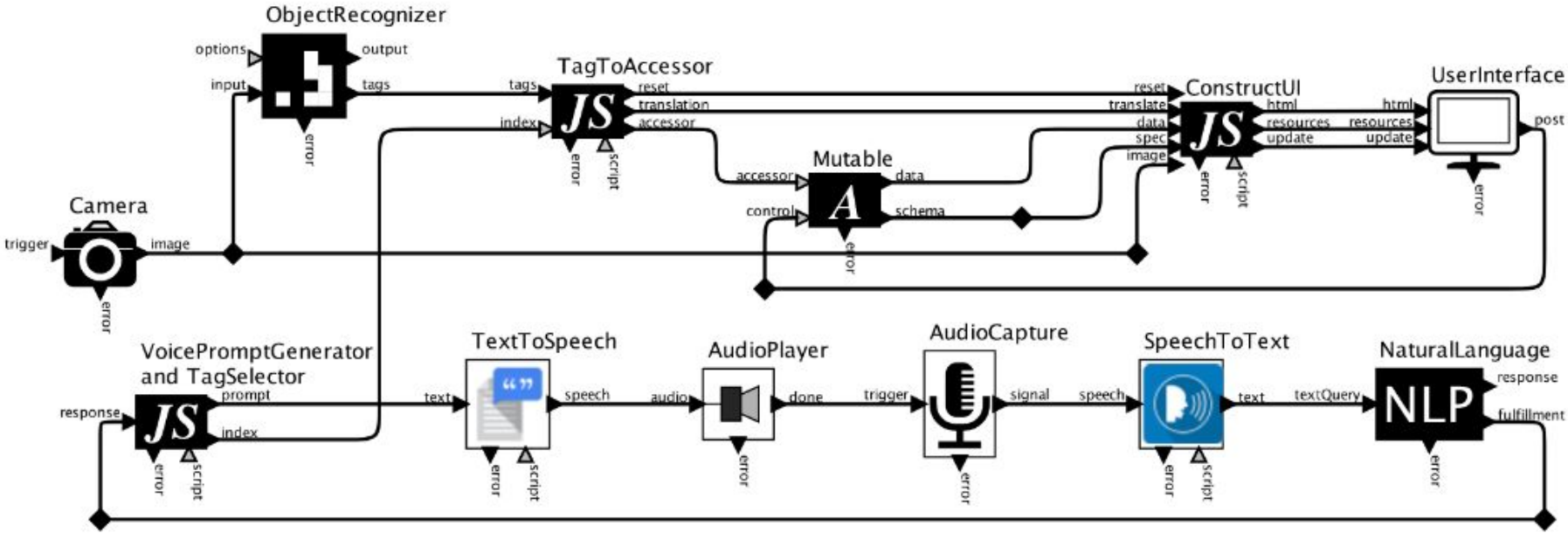
Questions

- **@Others, Gregor Peach** - I was a little confused whether or not they extended this system beyond one "device" or not. How would they handle cross device communication?
- **@samfrey99, Sam Frey -** The paper says that a properly written AAC implementation doesn't use locks and cannot deadlock, but if their implementation requires all actions to be atomic, isn't there now a far greater chance of live lock under heavy load?
- **@lrshpak, Lily Shpak -** Does the accessor add another layer of unneeded complexity?
- **@grahamschock, Graham Schock -** In the section that details coordination between these devices on the network it says unexpected non determinism is a problem. What does that mean? What is non determinism? How does it affect the system?

Conclusion

- Idea is a step towards faster development
- Descriptions were confusing, few examples provided
- Scalability and Robustness claims made are questionable

Conclusion


ObjectRecognizer
options
input
output
tags
error
TagToAccessor
tags
index
reset
translation
accessor
error
script
Mutable
accessor
control
data
schema
error
ConstructUI
reset
translate
data
spec
image
html
resources
update
error
script
UserInterface
html
resources
update
post
error
Camera
trigger
image
error
VoicePromptGenerator
and TagSelector
response
prompt
index
error
script
TextToSpeech
text
speech
error
script
AudioPlayer
audio
done
error
AudioCapture
trigger
signal
error
SpeechToText
speech
text
error
script
NaturalLanguage
textQuery
response
fulfillment
error