Motivation

The Internet of Things (IoT) poses **unprecedented challenges** to designers:

- Vast **heterogeneity**;
- Variable **utility**;
- “Perishable” **environment assumptions**.

Goal: **Adaptive Contracts**

- Integrate **dynamic re-configuration** and **re-purposing** in the design process;
- Extend **Contract Algebra**, by defining an **approximation relation**.

Means: **Accessors, Platform-based Design, Contracts**
Diverse Requirements
Unanticipated Use

TerraSwarm Research Center

05/06/16

BROKEN IF SEALED!
Fluid Environments

- Mobility
- Evolving Infrastructure
A Tower of Babel?
Standardization?
Accessors provide **access** to **any resource** that is reachable through an **arbitrary protocol** and exposes **some interface**.

**Accessors**

- Wrap an existing thing or service
- Export an actor interface
- Are **composable** with other actors
Platform-Based Design (PBD)

The application space includes the specification for the current mapping process. A specification can be provided by the designer or be the result of another PBD iteration.

The mapping process consists in the selection of a specific architectural instance, evaluating costs and functional/architectural constraints.

The architectural space includes platform components (libraries) abstracted from lower levels, connection rules and other properties such as component cost and timing properties.

Specs

C1

Cn
Platform-Based Design (PBD)

- Application space
- Architectural space
- Mapping and optimization
- Refinement
- Abstraction

Specs

Contracts to formally reason about horizontal and vertical relations defined by PBD!
The accessor bridges two semantic domains: Discrete Events (DE) and the accessor runtime (AR). Each domain has its own rules. Most importantly, the AR uses the asynchronous atomic callback (AAC) pattern due to its implementation in JavaScript, in DE a component may only react when it is fired.

“An Interface Theory for the Internet of Things”, Lohstroh and Lee (SEFM’15)
Other useful properties...

- For example, have responses from a Web server appear at the accessor’s output port in the same order as the corresponding requests arrived at its input port (in LTL):

\[
\forall a, b, c \in L : \\
G\{in = a \land t_{in} = x_1 \land F(in = b \land t_{in} = x_2)\} \Rightarrow \\
F\{out = f(a) \land t_{out} \geq x_1 \land F(out = f(b) \land t_{out} \geq x_2 \geq x_1)\}
\]
A contract $C=(A, G)$ is characterized by:

- A set of variables, or **ports**
- A set $A$ of **assumptions**
- A set $G$ of **guarantees**

$A$ and $G$ represent **sets of environment and system behaviors**

For a component $M$ (also defined as a set of behaviors) we have that

$$M \models C \iff A \cap M \subseteq G$$

A contract is saturated if in the form:

$$C = (A, G \cup \neg A)$$
A/G Contract Theory specifies a number of operations to operate on contracts.

We can recall a few of them:

- **Composition**
  For $C_1 = (A_1, G_1), C_2 = (A_2, G_2)$:
  $$C_1 \otimes C_2 = (A_1 \cap A_2 \cup \neg(G_1 \cap G_2), G_1 \cap G_2)$$

- **Refinement**
  For $C_1 = (A_1, G_1), C_2 = (A_2, G_2)$:
  $$C_1 \preceq C_2 \iff A_1 \supseteq A_2 \text{ and } G_1 \subseteq G_2$$

- **Compatibility**
  $C = (A, G)$ is compatible iff $A \neq \emptyset$

- **Consistency**
  $C = (A, G)$ is consistent iff $G \neq \emptyset$
Concrete representation of contracts using Linear Temporal Logic formulas

- Assumptions and guarantees of a contract represented by a pair of LTL formulas \((\varphi_A, \varphi_G)\)

Given contracts \(C_1 = (\varphi_{A1}, \varphi_{G1}), C_2 = (\varphi_{A2}, \varphi_{G2})\)

- Composition is
  \[
  C_1 \otimes C_2 = (\varphi_{A1} \land \varphi_{A2} \lor \neg(\varphi_{G1} \land \varphi_{G2}), \varphi_{G1} \land \varphi_{G2})
  \]

- Refinement is
  \[
  C_1 \preceq C_2 \quad \text{iff} \quad \varphi_{A2} \Rightarrow \varphi_{A1} \quad \text{and} \quad \varphi_{G1} \Rightarrow \varphi_{G2}
  \]
Contracts in IoT systems

Contracts are well suited for traditional systems design, but they can be problematic in more dynamic contexts...

Example: security camera with remote analysis of the footage

Contract C describes one aspect of the controller for the camera.

C
Inputs: x (network bandwidth)
Outputs: y (alarm notification delay)
A: x ≥ 1 Mbps
G: y < 10 sec

In saturated form, guarantees are x < 1 Mbps ∨ y < 10 sec

If the network is too slow, then the component is allowed to expose any behavior!
Gradually degrade performance, according to current environment conditions

Example: security camera with remote analysis of the footage
If the network cannot satisfy the assumptions, then the component should guarantee lower performance.

C
Inputs: x (network bandwidth)
Outputs: y (alarm notification delay)
A: x ≥ 1 Mbps
G: y < 1 sec

C*
Inputs: x (network bandwidth)
Outputs: y (alarm notification delay)
A: x ≥ 512 Kbps
G: y < 20 sec

C* represents an approximation of the original contract C
We introduce the notion of **Contract Approximation**:

\[ C_2 \preceq C_1 \text{ iff } A_2 \supseteq A_1 \text{ and } G_2 \supseteq G_1 \]

- This relation allows for the definition of a partial order on reliability levels of systems.
- A system implementing an approximate contract will be able to work with a wider set of assumptions at a cost of degraded guarantees.
Charging Station for Electric Cars
A charging station for electric cars is able to optimize power delivery to connected cars. Knowing how many cars are charging, the controller delivers a certain amount of power to every car to optimize charging time...

A typical requirement for this scenario could be:

- ...
- If a single car is connected to the charging station, its charging time cannot exceed 3h.
- ...
Charging Station for Electric Cars

A charging station for electric cars is able to optimize power delivery to connected cars. Knowing how many cars are charging, the controller delivers a certain amount of power to every car to optimize charging time...

C1: Charging module
- **Input**: n: # of cars (unitless)
- **Output**: p: charging power (W)
- **Assume**: 0 ≤ n ≤ 2
- **Guarantee**: 1500W ≤ p ≤ 5000W

C2: Battery module
- **Input**: p: charging power (W)
- **Output**: t: charging time (hours)
- **Assume**: 1000W ≤ p ≤ 5000W
- **Guarantee**: 0 ≤ t ≤ 2h

Spec:
- **Input**: n: # of cars (unitless)
- **Output**: t: charging time (hours)
- **Assume**: n = 1
- **Guarantee**: 0 ≤ t ≤ 3h

With Jun Jie Ng and Lucas Servén
Is $C_1 \otimes C_2$ compatible? Is $C_1 \otimes C_2$ consistent?

$C_1 \otimes C_2$ Assumptions:

$$(0 \leq n \leq 2 \land 1000W \leq p \leq 5000W) \lor \neg\{
[1500W \leq p \leq 5000W \lor \neg(0 \leq n \leq 2)] \land
[(0 \leq t \leq 2h) \lor \neg(1000W \leq p \leq 5000W)]\}$$

$C_1 \otimes C_2$ Guarantees:

$$[1500W \leq p \leq 5000W \lor \neg(0 \leq n \leq 2)] \land
[(0 \leq t \leq 2h) \lor \neg(1000W \leq p \leq 5000W)]$$

✓ Compatible

✓ Consistent
Does $C_1 \otimes C_2$ refine Spec?

Refinement Check: Assumptions

$C_1 \otimes C_2$ Assumptions:

\[(0 \leq n \leq 2 \land 1000W \leq p \leq 5000W) \lor \neg \{ [1500W \leq p \leq 5000W \lor \neg (0 \leq n \leq 2)] \land ([0 \leq t \leq 2h] \lor \neg (1000W \leq p \leq 5000W)) \}\]

Spec Assumptions:

\[n = 1\]
Refinement Check: Guarantees

C1⊗C2 Guarantees:

\[1500W \leq p \leq 5000W \lor \neg(0 \leq n \leq 2)\] \land
\[(0 \leq t \leq 2h) \lor \neg(1000W \leq p \leq 5000W)\]

Spec Guarantees:

\[0 \leq t \leq 3h \lor \neg(n = 1)\]

✓ Yes, C1⊗C2 ⊲ Spec
What happens if a charging station needs to charge more cars? (i.e. the requirement requires a guarantee for 3 cars)

We need to approximate C1 and degrade system performance.

C1: Charging module
Input: n: # of cars (unitless)
Output: p: charging power (W)
Assume: 0 ≤ n ≤ 2
Guarantee: 1500W ≤ p ≤ 5000W

C2: Battery module
Input: p: charging power (W)
Output: t: charging time (hours)
Assume: 1000W ≤ p ≤ 5000W
Guarantee: 0 ≤ t ≤ 2h

Spec:
Input: n: # of cars (unitless)
Output: t: charging time (hours)
Assume: n = 3
Guarantee: 0 ≤ t ≤ 3h
The System is Repurposed

What happens if a charging station needs to charge more cars? (i.e. the requirement requires a guarantee for 3 cars)

We need to approximate C1 and degrade system performance.

C1*: Charging module

Input: n: # of cars (unitless)
Output: p: charging power (W)
Assume: 0 ≤ n ≤ 3
 Guarantee: 1000W ≤ p ≤ 5000W

C2: Battery module

Input: p: charging power (W)
Output: t: charging time (hours)
Assume: 1000W ≤ p ≤ 5000W
 Guarantee: 0 ≤ t ≤ 2h

Spec:

Input: n: # of cars (unitless)
Output: t: charging time (hours)
Assume: n = 3
 Guarantee: 0 ≤ t ≤ 3h

Diagram:

C1* ⊗ C2

What happens if a charging station needs to charge more cars? (i.e. the requirement requires a guarantee for 3 cars)

We need to approximate C1 and degrade system performance.
Horizontal Composition: \( C_1^* \otimes C_2 \)

Is \( C_1^* \otimes C_2 \) compatible? Is \( C_1^* \otimes C_2 \) consistent?

**C1*⊗C2 Assumptions:**

\begin{align*}
(0 \leq n \leq 3 \land 1000W \leq p \leq 5000W) \lor \neg \\
\{ [1000W \leq p \leq 5000W \lor \neg(0 \leq n \leq 3)] \land \\
[(0 \leq t \leq 2h) \lor \neg(1000W \leq p \leq 5000W)] \}
\end{align*}

**C1*⊗C2 Guarantees:**

\begin{align*}
[1000W \leq p \leq 5000W \lor \neg(0 \leq n \leq 3)] \land \\
[(0 \leq t \leq 2h) \lor \neg(1000W \leq p \leq 5000W)]
\end{align*}

\( \checkmark \) Compatible

\( \checkmark \) Consistent
Refinement Check: Assumptions

C1*⊗C2 Assumptions:

\((0 \leq n \leq 3 \land 1000W \leq p \leq 5000W) \lor \neg\)
\(\left\{\left(1000W \leq p \leq 5000W \lor \neg (0 \leq n \leq 3)\right) \land \right.\)
\(\left(0 \leq t \leq 2h \lor \neg (1000W \leq p \leq 5000W)\right)\}\)

Spec Assumptions:

\(n = 3\)
Does $C_1^* \otimes C_2$ refine Spec?

Refinement Check: Guarantees

$C_1^* \otimes C_2$ Guarantees:

\[
[1000W \leq p \leq 5000W] \lor \neg(0 \leq n \leq 3) \land
[(0 \leq t \leq 2h) \lor \neg(1000W \leq p \leq 5000W)]
\]

Spec Guarantees:

\[
0 \leq t \leq 3h \lor \neg(n = 3)
\]

✓ $C_1^* \otimes C_2 \preceq$ Spec
IoT Design Methodology

**Design**

**Contracts** to formalize:
- actor **composition**
- specification **refinement**

**Validation** of:
*Functional requirements*
- through **verification** and **synthesis**

*Non-functional requirements*
- through **simulation** and **co-simulation**

**Component composition model**

*horizontal contract*
*refines*
*upper level requirements*
*(vertical contracts)*

---

**TerraSwarm Research Center**

05/06/16
IoT Design Methodology

Contracts to formalize:
- actor composition
- specification refinement

Validation of:
Functional requirements
- through verification and synthesis

Non-functional requirements
- through simulation and co-simulation

Component composition model
horizontal contract refines
upper level requirements (vertical contracts)

Contract Approximation to:
- increase reliability;
- enable model adaptiveness.

Execution monitors to:
- validate run-time behavior;
- identify specification violations;
- perform assumption mining.

Accessor (proxy)

Assumption violation triggers contract approximation

If no valid approximation is available.
Immediate Goals

● Develop an online monitor for Linear Temporal Logic Contracts on top of the "accessor" framework for IoT design that is available in Ptolemy II.

● Develop a mechanism to automatically identify specifications for a set of components such that a global system specification is met, maximizing (non-)functional requirements.

Future Work

● Fault localization & assumption mining
● Contract learning & optimization
Thank you!

V&V!

I&F?