Responsible Objects: Towards Self-healing Internet of Things Applications

Rafael Angarita

LAMSADE
Université Paris-Dauphine
rafael.angarita@lamsade.dauphine.fr

3rd International Workshop on Self-aware Internet of Things 2015
Overview

1 Introduction

2 The Concept of Responsible Objects

3 Summary and Future Work
Section 1

Introduction
Introduction

The Complexity Crisis:

Introduction

The Complexity Crisis:

Introduction

The Complexity Crisis:

- 2011: the Internet of Things Strategic Research Roadmap [Vermesan et al., 2011].
Introduction

The Complexity Crisis:

- 2011: the Internet of Things Strategic Research Roadmap [Vermesan et al., 2011].
Application Example

Resource constrained application; for example, constrained by execution time.

Abstractions of physical objects or software components.

Failures and connectivity disruptions are expected.

Call Emergency

Notify Contact

Notify Doctor

Display Message
Research question and challenges

Research question

How can we ensure the correct, efficient, fault tolerant execution of critical IoT applications \(^a\) with minimal human intervention?

\(^a\)for example, e-health or industry 4.0 applications
Research question and challenges

Research question

How can we ensure the correct, efficient, fault tolerant execution of critical IoT applications with minimal human intervention?

for example, e-health or industry 4.0 applications

Challenges:

- The dynamic nature of IoT application environments.
- Autonomic object design to support self-healing decisions.
Solution

We propose the concept of responsible objects to support self-healing IoT applications, which is an evolution of ideas taken from the research of transactional and self-healing services.
Section 2

The Concept of Responsible Objects
The Concept of Responsible Objects

In our context, the duty of an object is to serve in critical applications while facing unreliable and changing environments. We say that a smart object/thing is responsible if:
The Concept of Responsible Objects

In our context, the duty of an object is to serve in critical applications while facing unreliable and changing environments. We say that a smart object/thing is responsible if:

- It is transactional (deep-seated notion of acceptable behavior);
The Concept of Responsible Objects

In our context, the duty of an object is to serve in critical applications while facing unreliable and changing environments. We say that a smart object/thing is responsible if:

- It is transactional (deep-seated notion of acceptable behavior);
- it has self-healing capabilities (more sophisticated behavior);
The Concept of Responsible Objects

In our context, the duty of an object is to serve in critical applications while facing unreliable and changing environments. We say that a smart object/thing is responsible if:

- It is transactional (deep-seated notion of acceptable behavior);
- it has self-healing capabilities (more sophisticated behavior);
- we can talk with it through its API.
Transactional Model

Transactional properties [El Haddad et al., 2010].

- **pivot** ($p$): an operation is pivot, if once it successfully completes, its effects remain forever and cannot be undone; if it fails, it has no effect at all;
Transactional properties [El Haddad et al., 2010].

- **pivot** ($p$): an operation is pivot, if once it successfully completes, its effects remain forever and cannot be undone; if it fails, it has no effect at all;

- **compensable** ($c$): an operation is compensable, if it exists another operation which can undo its execution;
Transactional Model

Transactional properties [El Haddad et al., 2010].

- **pivot** \((p)\): an operation is pivot, if once it successfully completes, its effects remain forever and cannot be undone; if it fails, it has no effect at all;

- **compensable** \((c)\): an operation is compensable, if it exists another operation which can undo its execution;

- **retrievable** \((r)\): an operation is retrievable, if it guarantees a successful termination after a finite number of invocations.
Transactional properties [El Haddad et al., 2010].

- **pivot** \((p)\): an operation is pivot, if once it successfully completes, its effects remain forever and cannot be undone; if it fails, it has no effect at all;
- **compensable** \((c)\): an operation is compensable, if it exists another operation which can undo its execution;
- **retrievable** \((r)\): an operation is retrievable, if it guarantees a successful termination after a finite number of invocations.

The **retrievable** property can be combined with properties \(p\) and \(c\) defining **pivot retrievable** \((pr)\) and **compensable retrievable** \((cr)\) operations.
Transactional Model

Recovery mechanisms based on transactional properties:

Figure: Recovery Mechanisms
Self-healing states

We propose to use the following self-healing model as a framework:

Figure: Self-healing States (from [Ghosh et al., 2007])
Self-healing Staged Loop

Figure: Self-healing loop: detection, diagnosis, and recovery.
Self- and Context-knowledge

Only possible by being self- and context-aware:
General Architecture

This leads us to an agent-like architecture;

![Virtual Object General Architecture Diagram]

Figure: Virtual Object General Architecture
Self-healing State Transition Rules

Now, we can just tell the following health related rules to objects:

\[ \exists QoS_i \in QoS | \circ QoS_i > \circ \approx QoS_i \]

\[ \frac{\text{Degraded}_{QoS}}{\text{Degraded}_{QoS}} \]

\[ \exists QoS_i \in QoS | \circ QoS_i > \theta QoS_i \]

\[ \frac{\text{Timeout}_{QoS}}{\text{Timeout}_{QoS}} \]

\[ \mathcal{R}_{state^1} : \frac{\text{Timeout}_{QoS}}{\text{BROKEN}} \]

\[ \mathcal{R}_{state^2} : \frac{\neg \text{Timeout}_{QoS}}{\text{DEGRADED}} \]

\[ \mathcal{R}_{state^3} : \frac{\text{Degraded}_{QoS}}{\text{DEGRADED}} \]

\[ \gamma_{state} = C \]

\[ \mathcal{R}_{state^4} : \frac{\gamma_{state} = C}{\text{NORMAL}} \]

\[ \gamma_{state} = D \]

\[ \mathcal{R}_{state^5} : \frac{\neg \text{Degraded}_{QoS}}{\text{NORMAL}} \]

...but who tells those rules to each individual object?

---

1Rules to take actions were left out of this presentation
Application Supervisor

- Global knowledge and requirements for specific applications
- Knows application specific rules and high-level objectives
- Implements “disaster mode” protocol

**Call Emergency**

- \(d_{11}\)
- \(d_{12}\)
- \(d_{13}\)
- \(d_{14}\)

**Sugar Analysis**

- \(d_{1}\)
- \(d_{3}\)
- \(d_{5}\)

**Diagnoser**

- \(d_{8}\)

**Notify Contact**

- \(d_{10}\)

**Notify Doctor**

- \(d_{2}\)
- \(d_{4}\)

**Vital Signs Implant**

- \(d_{6}\)
- \(d_{7}\)

**Vital Signs Analysis**

- \(d_{9}\)

When a new object is introduced to a system, it will learn and take into account the specific composition and configuration of the application.

Application specific object knowledge base (another kind of context to take into account)
Section 3

Summary and Future Work
Summary

- We presented our motivation for building self-healing IoT applications.
We presented our motivation for building self-healing IoT applications.
Our interest lies specially in the context of critical systems.
We presented our motivation for building self-healing IoT applications. Our interest lies specially in the context of critical systems. We introduced the concept of responsible objects and its architecture. This concepts combine transactional and self-healing properties, and an interface to talk with the world.
Future Work

- First of all, perform experimental evaluation, though it may be difficult.
Future Work

- First of all, perform experimental evaluation, though it may be difficult.
- Refine and extend presented ideas.
Future Work

- First of all, perform experimental evaluation, though it may be difficult.
- Refine and extend presented ideas.
- Consider other types of problem-diagnosis components such as Bayesian networks.
Future Work

- First of all, perform experimental evaluation, though it may be difficult.
- Refine and extend presented ideas.
- Consider other types of problem-diagnosis components such as Bayesian networks.
- Take advantage of generated data to implement other self-* capabilities; for example, self-configuration.
Future Work

- First of all, perform experimental evaluation, though it may be difficult.
- Refine and extend presented ideas.
- Consider other types of problem-diagnosis components such as Bayesian networks.
- Take advantage of generated data to implement other self-* capabilities; for example, self-configuration.
- Given a high-level global objective, automatically derive individual local goals and rules.
The End
References I

J. Kephart and D. Chess,
The vision of autonomic computing.

O. Vermesan, P. Friess, P. Guillemin, S. Gusmeroli, H. Sundmaeker, A. Bassi, I. S. Jubert, M. Mazura, M. Harrison, M. Eisenhauer et al.,
Internet of things strategic research roadmap.

Atzori, Luigi, Antonio Iera, and Giacomo Morabito.
The internet of things: A survey.

El Haddad, Joyce and Manouvrier, Maude and Rukoz, Marta.
TQoS: Transactional and QoS-aware selection algorithm for automatic Web service composition
Cardinale, Yudith and Rukoz, Marta.
Fault Tolerant Execution of Transactional Composite Web Services: An Approach
In In Proceedings UBICOMM, pages 1–6, Lisbon, 2011.

Ghosh, Debanjan and Sharman, Raj and Rao, H. Raghav and Upadhyaya, Shambhu.
Self-healing systems - survey and synthesis.
Turtle/N3 Rules

@prefix int:  
<http://www.w3.org/2001/XMLSchema#integer>.
@prefix :  
<http://jena.hpl.hp.com/prefix#>.
@prefix sh:  
<http://www.lamsade.dauphine.fr/~angarita/selfhealing#>.

[ (?d int:hasValue ?a) greaterThan(?a,18) −> 
(sh:state sh:isA sh:normal)]
[ (?d int:hasValue ?a) lessThan(?a,18) −> 
(sh:state sh:isA sh:broken)]
[ (sh:state sh:isA sh:broken) −> 
(sh:action sh:isA sh:compensate)]
[ (sh:state sh:isA sh:normal) −> 
(sh:action sh:isA sh:continue)]
PREFIX sh: <http://www.lamsade.dauphine.fr/~angarita/selfhealing#>

SELECT ?action WHERE {sh:action sh:isA ?action}
Self-healing State Action Rules from Broken State

\[ \gamma_{sh-state} = \text{BROKEN} \]
\[ \neg \gamma_{tp} = r \]
\[ \frac{\gamma_{compensable}}{\text{COMPENSATE}} \]

\[ \gamma_{sh-state} = \text{BROKEN} \]
\[ \neg \gamma_{compensable} \]
\[ \frac{\gamma_{sh-state} = \text{BROKEN}}{\text{REPLICATE}} \]

\[ \gamma_{sh-state} = \text{BROKEN} \]
\[ \gamma(\circ)_{\text{progress}} \geq \theta_{\text{progress}} \]
\[ \gamma_{time} > 0 \]
\[ \frac{\gamma_{sh-state} = \text{BROKEN}}{\text{RETRY}} \]

\[ \gamma_{sh-state} = \text{BROKEN} \]
\[ \gamma(\circ)_{\text{progress}} < \theta_{\text{progress}} \]
\[ \frac{\gamma_{sh-state} = \text{BROKEN}}{\text{COMPENSATE}} \]
Self-healing State Action Rules from Normal State

\[ \gamma_{sh-state} = \text{NORMAL} \]
\[ \gamma_{availability} < \theta(\gamma)_{availability} \]
\[ \text{REPLICATE} \]

\[ \gamma_{sh-state} = \text{NORMAL} \]
\[ \neg \gamma^+_\text{time} > 0 \]
\[ \text{REPLICATE} \]

\[ \gamma_{sh-state} = \text{NORMAL} \]
\[ \gamma^+_\text{time} > 0 \]
\[ \gamma_{availability} \geq \theta(\gamma)_{availability} \]
\[ \text{CONTINUE} \]
Self-healing State Action Rules from Degraded State

\[
\begin{align*}
\gamma_{sh-state} &= \text{DEGRADED} \\
\gamma_{state} &= F \\
\neg \gamma_{tp} &= r \\
\frac{}{\text{COMPENSATE}}
\end{align*}
\]

\[
\begin{align*}
\gamma_{sh-state} &= \text{DEGRADED} \\
\gamma_{availability} &< \theta(\gamma)_{availability} \\
\frac{}{\text{REPLICATE}}
\end{align*}
\]

\[
\begin{align*}
\gamma_{sh-state} &= \text{DEGRADED} \\
\neg \gamma_{state} &= X \\
\gamma^+_\text{time} &> 0 \\
\gamma_{availability} &\geq \theta(\gamma)_{availability} \quad \text{CONTINUE}
\end{align*}
\]

\[
\begin{align*}
\gamma_{sh-state} &= \text{DEGRADED} \\
\gamma_{state} &= X \\
\gamma^+_\text{time} &> 0 \\
\gamma_{availability} &\geq \theta(\gamma)_{availability} \quad \text{RETRY}
\end{align*}
\]