Providing Resilience and Efficiency to the Internet of Things

Ingrid Nunes
Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre, Brazil

SEfIAS GI-Dagstuhl Seminar, 2018
About me

• Academic at UFRGS (Federal University of Rio Grande do Sul, BR) since 2012
  • Design/Implementation/Evolution Problems
  • User Agents/Decision Making (PhD)
    • Explanations in Recommender Systems (2016/2017 - sabbatical Year at TU Dortmund, DE)
      • NUNES, I.; JANNACH, D. A systematic review and taxonomy of explanations in decision support and recommender systems. User Model and User-Adapted Interaction, 2017.

21/08/2018
Ingrid Nunes <ingridnunes@inf.ufrgs.br>
About me

• Agent-oriented Software Engineering
  • Customisations of BDI Agents
    • NUNES, I.; LUCK, M. *Softgoal-based Plan Selection in Model-driven BDI Agents*, AAMAS 2014

• Agent-based Simulation

• Computer Networks

• … and self-adaptive systems
About me

21/08/2018

Ingrid Nunes <ingridnunes@inf.ufrgs.br>
From standalone to connected pieces of software

• Software components depend on one another

• Key differences from the past
  • Dynamic environment
  • Little can be assumed from the behaviour of other components
  • Intensive communication based on the internet

RESILIENCE and EFFICIENCY
Automated Management of Remediation Actions

• Remediation Action
  • An action that mitigates the consequences/effects of a problem

• Why?
  • Causes of the problem are unknown
  • Addressing the causes takes too long

Example:

```
notify_load(name, rate, link)
```

```
classify(flow)
```

```
limit(flow, rate)
```

```
start(link, samplingRate, length)
```

```
notify_detection(IPAddress)
```

```
setThreshold(t)
```

```
notify_new_record(flow)
```

```
notify_classification(label, flow)
```

```
start(IPAddress, rate, length)
```

```
limit(IPAddress, rate)
```

```
limit(IPAddress, rate)
```

```
start(IPAddress)
```

```
notify_detection(IPAddress)
```

```
notify_load(name, rate, link)
```

```
notify_detection(IPAddress)
```

```
note_threshold(t)
```

```
notify_new_record(flow)
```

```
notify_classification(label, flow)
```

```
start(IPAddress)
```

```
notify_detection(IPAddress)
```

```
notify_load(name, rate, link)
```

```
notify_detection(IPAddress)
```

```
notify_threshold(t)
```

```
notify_detection(IPAddress)
```

```
note_threshold(t)
```

```
notify_new_record(flow)
```

```
notify_classification(label, flow)
```

```
start(IPAddress)
```

```
notify_detection(IPAddress)
```

```
notify_load(name, rate, link)
```

```
notify_detection(IPAddress)
```

```
notify_threshold(t)
```

```
notify_detection(IPAddress)
```

```
note_threshold(t)
```

```
notify_new_record(flow)
```

```
notify_classification(label, flow)
```

```
start(IPAddress)
```

```
notify_detection(IPAddress)
```

```
notify_threshold(t)
```

```
notify_detection(IPAddress)
```

```
note_threshold(t)
```

```
notify_new_record(flow)
```

```
notify_classification(label, flow)
```

```
start(IPAddress)
```

```
notify_detection(IPAddress)
```

```
notify_threshold(t)
```

```
notify_detection(IPAddress)
```

```
note_threshold(t)
```

```
notify_new_record(flow)
```

```
notify_classification(label, flow)
```

```
start(IPAddress)
```

```
notify_detection(IPAddress)
```

```
notify_threshold(t)
```

```
notify_detection(IPAddress)
```

```
note_threshold(t)
```

```
notify_new_record(flow)
```

```
notify_classification(label, flow)
```

```
start(IPAddress)
```

```
notify_detection(IPAddress)
```

```
notify_threshold(t)
```

```
notify_detection(IPAddress)
```

```
note_threshold(t)
```

```
notify_new_record(flow)
```

```
notify_classification(label, flow)
```

```
start(IPAddress)
```

```
notify_detection(IPAddress)
```

```
notify_threshold(t)
```

```
notify_detection(IPAddress)
```

```
note_threshold(t)
```

```
notify_new_record(flow)
```
Automated Management of Remediation Actions

• Belief-Desire-Intention (BDI) Agents

CLASS
- att1
- att2
+ met1()
+ met2()

AGENT
- belief1
- belief2
+ goal1
+ goal2
- plan1A()
- plan2A()
- plan2B()

Reasoning cycle:
1. Update beliefs
2. Generate goals
3. Filter goals
4. Select plan
Automated Management of Remediation Actions

1. Constrained Goals/Plan Selection
   • Goal metadata to indicate restrictions of how goals can be achieved
     • constraints + utility-function

2. Cause-effect Knowledge Model
   • Used to identify possible problem causes

3. Goal Generation
   • Identify problem causes
   • Address identified causes
Automated Management of Remediation Actions

Fact
- Mandatory cause
- Optional cause
- Alternative cause

1. Build cause-effect status
2. Update status of cause factors
3. Evaluate effect goal
4. Evaluate cause
5. Generate cause factor test goals
6. Generate cause factor achievement goals
7. Update cause-effect problem end state

[no cause-effect status]
[otherwise]
[effect goal unfinished]
[effect goal finished]
[cause factor goals unfinished]
[cause factor goals finished]
[known cause]
[cause not found]
[unknown cause]
Automated Management of Remediation Actions

- **Goal**
  - \(\neg \text{water\_floor}\)
  - min time

- **Plans**
  - \text{place\_bucket}
    - Post-condition: \(\neg \text{water\_floor}\)
    - Resources: 1min
  - \text{repair\_ceiling}
    - Pre-condition: \text{have\_tools}
    - Post-conditions:
      - \(\neg \text{water\_floor}\)
      - \text{ceiling\_ok}
    - Resources: 60min
Automated Management of Remediation Actions

LimitLink
- Time = 5
- Network Availability = 0.6
- Vulnerability = 0.5

FindLimitIP
- Time = 30
- Network Availability = 0.8
- Vulnerability = 0.5

DoNothing
- Time = 0
- Network Availability = 1.0
- Vulnerability = 1.0

overUsage(link)

anomalous(IP)

threat(flow)

Fact
- Can achieve
- test goal
- Can achieve
- achievement goal

AnaliseLinkStatistics

LimitIP

ClassifyFlows

LimitFlows
Automated Management of Remediation Actions

- LimitLink
  Time = 5
  Network Availability = 0.6
  Vulnerability = 0.5

- FindLimitIP
  Time = 30
  Network Availability = 0.8
  Vulnerability = 0.5

- DoNothing
  Time = 0
  Network Availability = 1.0
  Vulnerability = 1.0

Reduction in the development effort with no impact in agent performance
Automated Management of Remediation Actions

- **LimitLink**
  - Time = 5
  - Network Availability = 0.6
  - Vulnerability = 0.5

- **FindLimitIP**
  - Time = 30
  - Network Availability = 0.8
  - Vulnerability = 0.5

- **DoNothing**
  - Time = 0
  - Network Availability = 1.0
  - Vulnerability = 1.0

Reduction in the development effort with no impact in agent performance
public class ProductService {
    /** some business logic */
    public List<Product> search(String query, boolean checkWarehouse) {
        products = ProductRepository.search(query);
        if (checkWarehouse)
            products.addAll(WarehouseAccess.searchProducts(query));
        return products;
    }
}
public class ProductService {
    /** some business logic */
    public List<Product> search(String query, boolean checkWarehouse){
        Cache cache = Cache.getInstance("productsCache");
        Cache cacheKey = "products:" + query; //key definition
        synchronized(this) {
            List<Product> products = (List<Product>)cache.get(cacheKey);
            if (products == null){
                products = ProductRepository.search(query);
                if (checkWarehouse)
                    products.addAll(WarehouseAccess.searchProducts(query));
                cache.put(cacheKey, products, 30); //TTL 30 seconds
            }
        }
        return products;
    }
}
public class ProductService {
/** some business logic */
public List<Product> search(String query, boolean checkWarehouse) {
    Cache cache = Cache.getInstance("productsCache");
    Cache cacheKey = "products:" + query; //key definition
    synchronized(this) {
        List<Product> products = (List<Product>)cache.get(cacheKey);
        if (products == null) {
            products = ProductRepository.search(query);
            if (checkWarehouse) {
                products.addAll(WarehouseAccess.searchProducts(query));
            }
        }
        cache.put(cacheKey, products, 30); //TTL 30 seconds
    }
    return products;
}
Application-level Caching

```java
public class ProductService {
    /** some business logic */
    public List<Product> search(String query, boolean checkWarehouse) {
        Cache cache = Cache.getInstance("productsCache");
        Cache cacheKey = "products:" + query; //key definition
        synchronized(this) {
            List<Product> products = (List<Product>)cache.get(cacheKey);
            if (products == null) {
                products = ProductRepository.search(query);
                if (checkWarehouse)
                    products.addAll(WarehouseAccess.searchProducts(query));
                cache.put(cacheKey, products, 30); //TTL 30 seconds
            }
            return products;
        }
    }
}
```

```java
public class OrderService {
    /** some business logic */
    public Order processOrder(Order order, Customer customer) {
        /** order processing logic */
        if (orderOk)
            Cache.getInstance("productsCache").delete("products:*");
        return products;
    }
}
```

```java
public class ProductRepository {
    /** some database-related logic */
    public void update(Product product) {
        Cache.getInstance("productsCache").delete("products:*");
        DBAccess.updateProduct(product);
    }
    public void delete(Product product) {
        Cache.getInstance("productsCache").delete("products:*");
        DBAccess.deleteProduct(product);
    }
}
```

Demands extensive knowledge

Maintenance is compromised

"21/08/2018 Ingrid Nunes <ingridnunes@inf.ufrgs.br>"
Application-level Caching

```java
public class ProductService {
    /** some business logic */
    public List<Product> search(String query, boolean checkWarehouse){
        Cache cache = Cache.getInstance("productsCache");
        Cache cacheKey = "products:" + query; //key definition
        synchronized(this) {
            List<Product> products = (List<Product>)cache.get(cacheKey);
            if (products == null){
                products = ProductRepository.search(query);
                if (checkWarehouse)
                    products.addAll(WarehouseAccess.searchProducts(query));
                cache.put(cacheKey, products, 30); //TTL 30 seconds
            }
        }
        return products;
    }
}
```

Demands extensive knowledge

Maintenance is compromised

Needs frequent revisions
Cacheability Pattern

- Is the data completely static?
  - Yes
  - No/Not Sure

- Does the data change more than it is used?
  - Yes
  - No/Not Sure

- Is the data used by many requests?
  - Yes/Not Sure

- Is the data user-specific?
  - No/Not Sure
  - Yes

- Is the data size large?
  - Yes/Not Sure

- Is the cache size large?
  - Yes
  - No

- Is the data expensive to compute?
  - Not Sure

- Uncacheable

- Cacheable
Cacheability Pattern

Does the data change more than it is used?

\[ \text{changeability}(m) > \mu_{ch} + k \times \sigma_{ch} \]

percentage of calls with the same parameters returning different values

\( k \) standard deviations above the mean of this metric considering all methods

Yes
Feedback Loop

1. Web Application

2. Monitoring Execution Traces
   - public class C1() {
     public Object proc() {
       //doing business logic...
       return content;
     }
   }

3. Cacheability Evaluation
   - 1. java.lang.Object pkg.C1.proc()
   - 2. model.Person pkg.C2.load(Long id)
   - 3. java.lang.Object pkg.C1.load()
   - 4. model.Person pkg.C3.proc(Object o)

4. Cache Management

Ingrid Nunes <ingridnunes@inf.ufrgs.br>
APLCache Framework

1. Web Application
2. Monitoring Execution Traces
3. Cacheability Evaluation
4. Cache Management

Class C1
Class C2
Class C3

Frameworks

Web Application

[JSON] JavaScript Object Notation

redis
mongoDB

Custom

Storage

Ingrid Nunes <ingridnunes@inf.ufrgs.br>
21/08/2018
Evaluation

APLCache automatically identified all methods cached by developers.
Evaluation

Increased throughput up to 17%

- Without Caching
- Human-made Caching
- APLCache

Requests handled per second with 5 simultaneous users:

- Pet Clinic: Without Caching 7.68, Human-made Caching 8.47, APLCache 9.00
- Shopizer: Without Caching 15.92, Human-made Caching 16.25, APLCache 16.96
- Cloud Store: Without Caching 22.28, Human-made Caching 22.73, APLCache 22.90

21/08/2018
Ingrid Nunes <ingridnunes@inf.ufrgs.br>
Evaluation

Increased throughput up to 17%

Requests handled per second with 5 simultaneous users

<table>
<thead>
<tr>
<th>Service</th>
<th>Without Caching</th>
<th>In-RAM Caching</th>
<th>In-RAM Caching + Cache</th>
<th>Without Caching</th>
<th>In-RAM Caching</th>
<th>In-RAM Caching + Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pet Clinic</td>
<td>7.68</td>
<td>8.47</td>
<td>9.00</td>
<td>7.68</td>
<td>8.47</td>
<td>9.00</td>
</tr>
<tr>
<td>Cloud Store</td>
<td>22.28</td>
<td>22.73</td>
<td>22.90</td>
<td>22.28</td>
<td>22.73</td>
<td>22.90</td>
</tr>
</tbody>
</table>

Jhonny Mertz  
Rômulo Meloca
Conclusion

- Automated management of remediation actions
  - Promotes resilience

- Application-level caching
  - Can be used for improving resilience in case of, e.g., communication failure
  - Promotes efficiency, avoiding unnecessary calls

- What about IoT?
  - Current application domain
    - Mitigation of CO leaking in Smart Homes
    - Use of caching in the iCasa project
      - with University of Grenoble
Key References

- Remediation Actions

- Application-level Caching

http://inf.ufrgs.br/~ingridnunes/