A Model-Driven Approach to Develop Adaptive Firmwares

Franck Fleurey, Brice Morin and Arnor Solberg
franck.fleurey@sintef.no
SINTEF, Oslo, Norway

This work has been funded by the DiVA (EU FP7 STREP) and MODERATES (SINTEF) projects.
Context

- Internet of things / Sensor networks
  - Resource constrained systems
  - Increasingly complex and dynamic applications

- Current practice
  - Trial and error development of C modules
  - Tangling of different concerns
    - Data processing, communication and networking, adaptation, ...
  - No maintenance, reuse or evolutions considerations

- In the need for software engineering techniques
  - Suited for resource-constrained environments
  - Taking the hardware and physical systems into account
Motivating example

**ZigBee Clients**
- Data Logging server
- Other Sensor Nodes
- ZigBee Appliances

**Bluetooth clients**
- Smartphones
- Tablet applications
- Laptop computers

**Gateway**
Bridge between low power / low range radio communication to standard radio technologies

**Adaptive Wireless Temperature Sensor Nodes**
- Adapt to the situation
- Battery powered
- Limited resources
- Low power radio link
The target platform

- MSP430 F2274
  - 16 bits RISC
  - 32 ko Flash
  - 1 ko SDRAM
  - 16 MHz
- USART
- Digital I/Os
- ADC Inputs (10-bits)
- Wireless Transmitter and Receiver
- 2 LEDs
- 1 Push-button
How complicated can it be?

- Very simple functional specification
  - Provide temperature data to the applications
    ```java
    while (true) {
        sampleTemperature();
        sendTemperature();
        delay(5000);
    }
    ```

- But... in a resource-constrained environment
  - Limited CPU and memory, battery powered, limited bandwidth

- But... Different applications have different needs
  - Update rate, notifications, accuracy, response time, ...
How complicated can it be?

- Very simple functional specification
  - Provide temperature data to the applications
    - Adapt sampling frequency and measurement accuracy
    - Have a sensor keep track of minimum and maximum
    - Provide an “alarm service” to clients
    - Gracefully degrades on low battery
    - Etc...

- But...
  - Limited CPU and memory, battery powered, limited bandwidth

- But... Different applications have different needs
  - Update rate, notifications, accuracy, response time, ...
More generally

- Include more logic in the sensors
  - Push the logic as close as possible to the data source
    - Save unnecessary communication which are costly in terms of power and bandwidth.
  - Have the sensor adapt to its environment
    - Find QoS trade-offs according to available resources.
  - Have the sensors adapt the applications needs
    - Adjust the QoS trade-offs to the needs and priority of the applications

- Makes producing the firmware a lot more challenging
- Becomes a self-adaptive system
- Need for software engineering techniques
Proposed Approach

- Model the system expected behaviour
  - Using event driven state machines and asynchronous messages
- Model the runtime variability in the system
  - Using aspect-oriented modelling techniques
- Model the adaptation policies
  - Using a domain specific modelling language

- Perform exhaustive analysis of these models
- Produce an optimized state machine which compiles behaviour, variability and adaptation together
- Produce firmware code which runs on the target platform
Adaptation Model

Context

Rules

Constraints

Variants (V1, ..., Vn)

Constraints

Base Model

State machine

Devices & Messages

Aspect Models

Pointcut

Advice

Exhaustive Simulation (for all context instances)

Weaving of all reachable configurations

Composition & minimization of the configuration models

Optimized Adaptive Firmware code (in C)

Compilation and loading

Tools

Editor

Simulator

Editor

Weaver

Editor

Simulator

Compilation Chain

(1) (2) (3) (4) (5)
Adaptation Model

Context

Rules

Constraints

Variants (V1, ..., Vn)

Base Model

State machine

Devices & Messages

Aspect Models

Pointcut

Advice

Configurations

Weaving of all reachable configurations

Composed model

State machine

Devices & Messages

C Code Generation

Optimized Adaptive Firmware code (in C)

Compilation and loading

Tools

Editor

Simulator

Editor

Weaver

Editor

Simulator

Models

Exhaustive Simulation (for all context instances)

Composition & minimization of the configuration models

Compilation Chain

(1) Exhaustive Simulation (for all context instances)

(2) Weaving of all reachable configurations

(3) Composition & minimization of the configuration models

(4) C Code Generation

(5) Compilation and loading

Ctx Instance Configuration

<table>
<thead>
<tr>
<th>Ctx Instance</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context 1</td>
<td>V1, V3, V4</td>
</tr>
<tr>
<td>Context 2</td>
<td>V2, V4</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Context N</td>
<td>Vi, ..., Vj</td>
</tr>
</tbody>
</table>
The Adaptation Model

- Variability in the system
  - Features, dependencies, constraints
- Variability in the environment
  - What are the elements the system should adapt to
- Adaptation policy
  - What feature should be used in a given context

- The DiVA approach combines
  - Adaptation hard-constraints
  - High-level adaptation rules
    - Expressed on properties of the system
Adaptation model

### Context Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>ID</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enum Power Status</td>
<td>power</td>
<td>{normal, low, critical}</td>
</tr>
<tr>
<td>Enum Sensor Mode</td>
<td>mode</td>
<td>{passive, active}</td>
</tr>
<tr>
<td>Boolean Has Subscribers</td>
<td>subscribers</td>
<td>-</td>
</tr>
<tr>
<td>Boolean Reliable Network</td>
<td>reliablenet</td>
<td>-</td>
</tr>
<tr>
<td>Boolean RF Signal Available</td>
<td>rfsignal</td>
<td>-</td>
</tr>
<tr>
<td>Boolean Alarm</td>
<td>alarm</td>
<td>-</td>
</tr>
</tbody>
</table>

### System Variability and Constraints

<table>
<thead>
<tr>
<th>Name</th>
<th>ID</th>
<th>Lower</th>
<th>Upper</th>
<th>dependency</th>
<th>available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension Transmission</td>
<td>T</td>
<td>0</td>
<td>-1</td>
<td>-</td>
<td>subscribers</td>
</tr>
<tr>
<td>Variant Unicast</td>
<td>UNI</td>
<td>-</td>
<td>-</td>
<td>MM and (BRO or UNI)</td>
<td>mode=scribers</td>
</tr>
<tr>
<td>Variant Broadcast</td>
<td>BRO</td>
<td>-</td>
<td>-</td>
<td>UNI</td>
<td>not power=critical</td>
</tr>
<tr>
<td>Variant SendMinMax</td>
<td>SMM</td>
<td>-</td>
<td>-</td>
<td>TED</td>
<td></td>
</tr>
<tr>
<td>Variant Transmission Errors Detection</td>
<td>TED</td>
<td>-</td>
<td>-</td>
<td>TED or ROE</td>
<td></td>
</tr>
<tr>
<td>Variant Retransmit On Error</td>
<td>ROE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Variant Buffer On Error</td>
<td>BOE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Dimension Power Management</td>
<td>P</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Variant Intermittent Operation</td>
<td>INTER</td>
<td>-</td>
<td>-</td>
<td>BRO</td>
<td>power=low</td>
</tr>
<tr>
<td>Variant Critical Operation</td>
<td>SLEEP</td>
<td>-</td>
<td>-</td>
<td>UNI</td>
<td>power=critical</td>
</tr>
<tr>
<td>Dimension Data Collection Options</td>
<td>D</td>
<td>0</td>
<td>-1</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Variant Compute Min/Max values</td>
<td>MM</td>
<td>-</td>
<td>-</td>
<td>SMM or not (UNI or BRO)</td>
<td>power=normal</td>
</tr>
<tr>
<td>Variant Average 3 Samples</td>
<td>AVG</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
Adaptation model (2)

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Load</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Negligible</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>Low</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>High</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>Sensor Data Accuracy</td>
<td>1</td>
<td>-</td>
</tr>
</tbody>
</table>

Transmission (T)

<table>
<thead>
<tr>
<th>CPU Load</th>
<th>Sensor Data Accuracy</th>
<th>Responsivness</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Unicast (UNI) | Low | No Impact
Broadcast (BRO) | Low | No Impact
SendMinMax (SMM) | Negligible | No Impact
Transmission Errors Detection (TED) | Low | No Impact
Retransmit On Error (ROE) | High | Improves
Buffer On Error (BOE) | Low | Improves

<table>
<thead>
<tr>
<th>Name</th>
<th>ID</th>
<th>context</th>
<th>CPU Load</th>
<th>Sensor Data Accuracy</th>
<th>Responsivness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule Max Accuracy</td>
<td>MA</td>
<td>subscribers and power=normal</td>
<td>Low</td>
<td>High</td>
<td>-</td>
</tr>
<tr>
<td>Rule Alarm</td>
<td>AL</td>
<td>alarm</td>
<td>-</td>
<td>High</td>
<td>-</td>
</tr>
<tr>
<td>Rule Not Alarm</td>
<td>SP</td>
<td>not alarm</td>
<td>High</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Rule Save power</td>
<td>SP</td>
<td>not power = normal</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Rule Bad Network</td>
<td>BN</td>
<td>not reliablenet</td>
<td>-</td>
<td>-</td>
<td>High</td>
</tr>
</tbody>
</table>
### Exhaustive simulation

#### Exhaustive enumeration of the contexts

<table>
<thead>
<tr>
<th>power</th>
<th>mode</th>
<th>subscribers</th>
<th>reliablenet</th>
<th>rfsignal</th>
<th>alarm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context</td>
<td>Normal</td>
<td>Active</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>Context</td>
<td>Normal</td>
<td>Active</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>Context</td>
<td>Normal</td>
<td>Active</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>Context</td>
<td>Normal</td>
<td>Active</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>Context</td>
<td>Normal</td>
<td>Active</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>Context</td>
<td>Low</td>
<td>Passive</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>Context</td>
<td>Low</td>
<td>Passive</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>Context</td>
<td>Low</td>
<td>Passive</td>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>Context</td>
<td>Low</td>
<td>Passive</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

#### Simulation results

<table>
<thead>
<tr>
<th>Scenario EXHAUSTIVE</th>
<th>CPU</th>
<th>ACC</th>
<th>RES</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Configuration (34)</td>
<td>-48</td>
<td>80</td>
<td>2</td>
<td>34</td>
</tr>
<tr>
<td>Unicast (-8)</td>
<td>-8</td>
<td>0</td>
<td>0</td>
<td>-8</td>
</tr>
<tr>
<td>SendMinMax (0)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Transmission Error</td>
<td>-8</td>
<td>0</td>
<td>0</td>
<td>-8</td>
</tr>
<tr>
<td>Buffer On Error (10)</td>
<td>-8</td>
<td>16</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>Compute Min/Max</td>
<td>-8</td>
<td>32</td>
<td>0</td>
<td>24</td>
</tr>
<tr>
<td>Average 3 Sample</td>
<td>-16</td>
<td>32</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>Context</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Configuration (-32)</td>
<td>-64</td>
<td>32</td>
<td>0</td>
<td>-32</td>
</tr>
<tr>
<td>Unicast (-32)</td>
<td>-32</td>
<td>0</td>
<td>0</td>
<td>-32</td>
</tr>
<tr>
<td>SendMinMax (0)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Compute Min/Max</td>
<td>-32</td>
<td>32</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Context</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Configuration (34)</td>
<td>-48</td>
<td>80</td>
<td>2</td>
<td>34</td>
</tr>
<tr>
<td>Unicast (-8)</td>
<td>-8</td>
<td>0</td>
<td>0</td>
<td>-8</td>
</tr>
<tr>
<td>SendMinMax (0)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Exhaustive simulation (2)

17 configurations
Exhaustive simulation (2)
Adaptation Model

Context

Rules

Variants (V1, ..., Vn)

Constraints

Constraints

Context

Context

Rules

Variants (V1, ..., Vn)

Exhaustive Simulation
(for all context instances)

(1)

Weaving of all reachable configurations

(2)

Configuration Models

State machine

Devices & Messages

Configuration Models

State machine

Devices & Messages

Composition & minimization of the configuration models

(3)

Optimized Adaptive Firmware code (in C)

(4)

Compilation and loading

(5)
Base model with state machines

State Machine

Device model

```
device Timer
{
    // Start the Timer
    message start(timer_id : Integer, delay : Integer);
    // Cancel the Timer
    message cancel(timer_id : Integer);
    // Notification that the timer has expired
    message timeout(timer_id : Integer);

    sends timeout
    receives start, cancel
}
```
Aspect to specify variability

Base model

Idle → Sample
Transmit

Sleep Aspect

Sleep

<<global>>

Min/Max Aspect

Idle
get Min
get Max

Sample

T = ?

Min = (T < Min) ? T : Min
Max = (T > Max) ? T : Max

SINTEF
Adaptation Model
- Context
- Rules
- Variants (V1, ..., Vn)
- Constraints

Aspect Models
- Pointcut
- Advice

Base Model
- State machine
- Devices & Messages

Exhaustive Simulation (for all context instances)

Weaving of all reachable configurations

Composition & minimization of the configuration models

Composed model
- State machine
- Devices & Messages

Optimized Adaptive Firmware code (in C)

Compilation and loading

Compilation Chain
1. Editor
2. Simulator
3. Editor
4. Weaver
5. Editor
6. Simulator
Composed model: Strategy 1

- One composite state per configuration
- Good readability but far from optimal
Composed model: Strategy 2

- Equivalent to the previous state machine
- Obtained by flattening and minimization
- Much more efficient but not as readable
- Used for code generation
Adaptation Model

- Context
- Rules
- Constraints
- Variants (V1, …, Vn)

Aspect Models

- Pointcut
- Advice

Base Model

- State machine
- Devices & Messages

Exhaustive Simulation (for all context instances)

<table>
<thead>
<tr>
<th>Ctx Instance</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context 1</td>
<td>V1, V3, V4</td>
</tr>
<tr>
<td>Context 2</td>
<td>V2, V4</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Context N</td>
<td>Vi, …, Vj</td>
</tr>
</tbody>
</table>

Weaving of all reachable configurations

Composition & minimization of the configuration models

Composed model

- State machine
- Devices & Messages

Optimized Adaptive Firmware code (in C)

Compilation and loading
Summary

- Sensor networks and Internet of things
  - Highly dynamic adaptive and distributed systems
  - Need proper software engineering techniques
- Even the smallest nodes need to adapt
  - Challenge: resource-constrained environment
  - Opportunity: reasonable scale
- The approach uses domain specific modelling languages
  - Adaptation model combining constraints and goals
  - Behaviour model using state machines and aspects
- Exhaustive simulation and optimisation
- Generation of firmware in plain platform specific C
Limitations and Future work

- Automated detection of quiescent states
  - Allow reconfigurations as soon as they are safe
- Validation with model checking
  - Of the adaptation model and combined state machines
- Dealing with “unforeseen” adaptations
  - Firmware live updates
  - Finer grained reconfigurations
- Coordinating the adaptation of a set of devices
  - Connect the adaptation of a set of devices
  - Distribute the adaptation logic
- Integrations of a complete tool chain
  - Individual tools are already available as open-source
Thank you!

- Questions?
Sate Machine model

```
composite state ACTIVE init ReadSensorValues {

    state WaitingForCmd {

        on entry {
            send Timer.start('1', '10000')
        }

        transition update_data -> ReadSensorValues {
            event Timer#timeout
            guard 'timer_id == 1'
        }

        transition GetData -> WaitingForCmd {
            event WTSClient#GetData
            action send WTSClient.SensorData('temp','0','0','batt')
        }

        transition GetStatus -> WaitingForCmd {
            event WTSClient#GetStatus
            action send WTSClient.SensorStatus('interval','0','0','0')
        }

        transition GetName -> WaitingForCmd {
            event WTSClient#GetName
            action send WTSClient.SensorName('name')
        }

        transition SetName -> WaitingForCmd {
            event WTSClient#SetName
            action 'strcpy(name, new_name);'
        }
    }

    state ReadSensorValues {

        on entry {
            send LED.light_on('2') // GREEN
            send MSP430Sensor.measure_temperature()
        }

        on exit {
            send LED.light_off('2')
        }
    }
```
Aspect implementing Min/Max

```plaintext
pointcut {
    // The system root component in which to store the min/max info
    component WTS <PC_SystemRoot> {}
    // The idle state where we add the component
    state WaitingForCmd <PC_Idle_State>{}
    // The sampling state where to add the min and max
    state ReadSensorValues <PC_Sample_State>{}

    transition temperature <PC_setData> {
        // Detect all places where data is sent
        <PC_message_to_change> {
            send WTSClient.SensorData('temp', '0')
            // Things which we refer to but are already in the base
            device WTSClient {
                message SensorData(temp : ?), min : ?, max : ?, batt : ?)
                receive SensorData
            }
        }
    }

    advice {
        // add the min/max variables
        component WTS <AD_SystemRoot> {
            property min : Integer
            property max : Integer
        } state WaitingForCmd <AD_Idle_State> {
            transition ResetMinMax <AD_ResetMinMax> -> WaitingForCmd {
                event WTSClient#ResetMinMax
                action 'min = temp; max = temp;'
            }
        }

        // add the actions to compute the min/max
        state ReadSensorValues <PC_Sample_State> {
            transition temperature <PC_setData> ? {
                action {
                    if (temp > max) max = temp;
                    if (temp < min) min = temp;
                }
            }
        }

        // add the data to the SensorData messages
        <AD_message_to_change> {
            send WTSClient.SensorData('temp', 'min', 'max', 'batt')
            // Things which we refer to but are already in the base
            datatype Integer;
            device WTSClient {
                message SensorData(temp : ?, min : ?, max : ?, batt : ?);
                sends ResetMinMax
                receives SensorData
            }
        }
    }
}```