Pivot

- The Language for Home Automation -

Project Report

D410f19

Aalborg University
Computer Science

Hannah Lockey

Asger Gammelgaard Bertel

Philip Irming Holler

Magnus Kirkegård Jensen

Mads Schou Faber

Mads Kristian Bau-Madsen

AALBORG UNIVERSITY
STUDENT REPORT

Computer Science
Selma Lagerløfs Vej 300
9220 Aalborg Ø
http://www.cs.aau.dk/

Title:
Pivot - The language for home automation

Theme:
Computer Science, Domain Specific Language

Project Period:
Spring Semester 2019

Project Group:
D410f19

Participant(s):
Asger Gammelgaard Bertel
Philip Irming Holler
Hannah Lockey
Magnus Kirkegård Jensen
Mads Schou Faber
Mads Kristian Bau-Madsen

Supervisor(s):
Michele Albano

Copies: 1

Page Numbers: 22

Date of Completion:
March 12, 2019

Abstract:

Here is the abstract

Contents

Abstract

Preface

This report is written by six 4th semester computer science students from Aalborg University in the period from February 1st to May 28th. The project assignment involved designing, defining and implementing a new programming language using syntax and semantics, as well as using implementation techniques learned in the courses in parallel with the project process.

Chapter 1

Introduction

The network and communication between electronic devices have in the past years become more popular[7], and to describe the connectivity between the various devices in our homes, the term "Internet of Things" (IoT) is used. Smart devices such as TVs, thermostates, lamps, Google home and various other devices are being developed and distributed to regular households, and with the growth of communication devices, the potential of IoT also becomes greater. Currently, the software for IoT is being programmed in general purpose programming languages such as Java, C or Python[7]. The purpose of this project is to create a specialised domain language for IoT, to offer easier options to the user and to customise the behaviour of their own devices in their homes.

Chapter 2

Problem Analysis

In this chapter the problem will be described in detail. This should lead to a problem description which will be the basis for the solution. The initial idea started, when it was found out, that almost all home automation software was written using a general purpose language. This begged the question of whether or not the process of writing home automation could be made easier and or faster with a domain specific language.

Home automation is defined by a system scheduling e.g. time based events to other systems in the household such as lights or other appliances[14]. You can make your own events with home automation, but only in the scope that the manufacturer decides. Using a programming language to define rules and events would widen the scope and allow the user a great degree of freedom in creating home automation systems.

Currently there are several existing systems for managing home automation. Some popular systems for smart lightning include Philips Hue[13], IKEA Trådfri[8]. Other systems for home automation includes Siemens connect, which allows smart managing and automation of home appliances. These systems enable consumers to automate their homes by setting for example timers for their lighting. The systems, however, require the user to use software either written by these companies or written using API's from these companies. Also the software will for the most part only work with hardware from that specific company. This kind of specialized hardware might be expensive and or not fitting the specific need from the consumer. For example a regular Philips Hue Bulb for the E26 socket costs 29.99USD[13]. This means that a cheaper and more flexible solution could be desired by some users.

The stakeholders for this project, would be not only the users wishing to create software for IoT, but also potentially the producers of hardware and software used in home automation. The best case for this project would be, that the development of IoT software through our programming language, can be applied to all kinds of devices without any change in hardware, but this is fairly optimistic. Furthermore, the stakeholders that can influence this project, would be the ones producing products that are the target devices for our programming language, due to them controlling the procedures for communicating with the devices.

Most current home automation programs are written in general purpose languages such as Java, Python, C and C++ [9]. Except for Python, all of these languages are of the C-family of programming language, meaning that they have many similarities in syntax. All of these languages are general purpose languages, which can lead to some solutions being less than optimal due to needing some workarounds. Also this can lead to many lines of unnecessary coding, which can increase the code complexity and the needed to develop and test it. For example in Java, if a programmer wanted to create an event based program, he or she must first create either an infinite loop inside the main method or listeners for events. This main method would then also need to be in a class, since java is strictly object oriented. Both these things add lines code, that does not actually contribute to the functionality of the program. A programmer trying to implement an event based home automation program to manage domotic devices would need to create an event handler and possibly a queue for sorting the events after time, depending on the specific implementation. This will further increase the amount of programming needed, even though these things will probably stay the same for most implementations. Some of this could also be solved using some libraries for Java, C# etc, but this would still lead to more work to design, implement and debug these.

The extra lines of code and additional programming leads to the need for a higher level language, that would automatically take care of timing, queue, loops, listeners etc. This would free up time for the programmer to focus on implementing the logic for the events. In this report we will limit our research to event based programming as a solution.

Another problem with the current solutions for home automation is, that there are many different ways of communicating through the network to the smart devices. In some cases, proprietary software and hardware is used for this purpose. This leads to a fragmentation in the market between the different brands and approaches. For this reason a new higher level programming language would also either need to compile to a program with predefined communication protocols for some of the current standards, if possible, or some structures inside the language to customise these protocols. This would either add support for current hardware by customising, or create a new standard, that the hardware then needs to be adapted to.

2.1 Existing Solutions

In the search for existing systems we limit our search to programming languages for implementing home automation. We do not consider, for example the IKEA Trådfri software a competitor, since it is not a way of creating home automation software, but instead a software for managing an existing smart lightning system with only IKEA hardware.

Tools such as Calaos[2] and MisterHome[10] already exists for creating custom home automation solutions for a variety of devices. Both of these tools limit themselves entirely to configuration through a user interface with a limited amount of predefined customisation options. This again distances them from a programming language for home automation.

Domoticz is a tool which allows further customisation using a drag and drop programming solution. Domoticz also allows adding new types of devices by defining which signals they can receive, there is however only a limited amount of signals to choose from and no way to define new signals. This limits the amount of devices that the system can interact with. [5]

Home Assistant is an interface-based tool like Calaos and MisterHome but it also allows scripting in python [6]. But python is a general purpose language which, as previously discussed, may lead to suboptimal programming when compared to a domain specific language. Similarly OpenHAB[12] allows scripting but it has its own domain-specific programming language. One downside to OpenHAB is that it only works with a set of predefined devices.

Another group at Aalborg University created a 4th semester project regarding a programming language for home automation in 2014[3]. This group implemented a language called Home, that could be used to create event based home automation programs, that take input from hardware to trigger an event. They did, however, not manage to implement a timer for their home automation. This means, that an event cannot be triggered at a certain time or date, but instead only when some hardware sends an input to the program. The focus of the Home language was also to enable novice programmers to code their own home automation software using a simple high level language.

It appears that existing languages for home automation limit the user, either by only supporting a predefined set of devices, or by lacking crucial features such as timed events.

2.2 Problem description

The problem analysis led to the following problem description which will be used for creating a solution to the problem.

- **Problem:**
  How can a higher level language be designed, that enables easier development of an event based program to control home automation?
  - **Subproblems:**
    How should this language be compiled to run a server device?
    - How would the compiled program communicate with the smart devices?
  - How should the syntax look in order to optimise the code for the specified user?
    - Who would be the user of this language?
    - Language evaluation criteria

2.3 Solution criteria

In order to make sure that the proposed language can be considered a solution, a number of criteria has been made. These will be used to evaluate the final language. Additionally they will be used to conclude which characteristics are the most important for the language in section 3.3. The following criteria has been established based on the problem analysis:

- The language must enable the user to easily express and solve home automation problems.
- The language must enable the user to control devices based on incoming signals and time-based events.
- The language must support addition of new types of devices.

The first criteria is very broad, but serves as a guideline designing and creating the proposed language. The first criteria reflects on the ease of use as well as how easy it is to understand, specifically in relation to home automation. This is important, since being able to easily write and understand the code, will allow developers to be more productive. The second criteria revolves around the requirement to allow interaction with sensors and enable time-based events to be created. The third requirement specifies that the language must enable support of new devices that are not supported by default. This could be accommodated by allowing the user to define basic custom communication protocols.

Chapter 4

Language Design

In the previous Chapter, Chapter 3, seven steps to create a DSL were listed. In this chapter, steps 3 and 4 will be considered. It will be discussed how the Pivot language will be designed. Initially, some of the knowledge gathered in the previous chapters will be used to briefly describe Pivot. Following this, an example program will be discussed with reasoning for the syntax.

4.1 Initial Description of Pivot - Not Finished

Pivot has two new types. The Signal type and the Device type. The Signal type is for declaring which types of signals a sensor should emit. A Device can be two things. Either a Device is a smart device of some kind, like a light bulb, or it can be a sensor.

4.2 Pivot program example

In order to show how a program, that solves the problems described in the problem analysis in section 2, would look like, an example was created. Several example programs were created, since they allow for finding a way to express a solution to a given problem in the Pivot language. This is also useful for finding the abstract syntax of the Pivot language.

```
// Define signals
#define Signal toggle: On = 1, Off = 0;
#define Signal celsius:  1..60;

// Define device types
#define Device Bulb input: toggle;
#define Device Window input: open;
// Define sensors
#define Device MovementSensor output: toggle & input: celsius;
```

```
#define Device Thermometer output: celsius;

// Variable definitions of every device in the house
Bulb frontDoorLight = 129.67.198.1:12565;
Window bedRoomWindow = 129.67.198.65:12565;

MovementSensor frontDoorSensor = 129.67.198.65:12565;

Thermometer mainThermometer = 129.67.198.65:12565;

void init(){
    // Optional code for initialization
    // Automatically run when the program is run for the first time
}

// Events
When frontDoorSensor toggle: on {
    if(now > 17:00 && now < 21:00){
        set frontDoorLight toggle: on;
    }
    wait 30 seconds;

    while(get frontDoorSensor output toggle != on){
        set frontDoorLight toggle: off;
    }

    every 10 seconds until now + 2 minutes {
        // Do something
    }
}

When mainThermometer celsius exceeds 22{
    setBedroomWindowsOpen(50); // Calling function defined further down
}

// Function
void setBedroomWindowsOpen(int percentage){
    set bedRoomWindow open: percentage;
}

// Repeating event
every 2 hours{
    // Do something
}
```

Code snippet 4.1: Pivot example program

4.3 Abstract syntax

The program structure of this language is described by the following abstract syntax. This abstract syntax was created to give a user an overview of the structure of a program written in Pivot. The example programs were used to find this structure.

Syntaktiske Kategorier

a ∈ Arithmetic expressions
b ∈ bool
e ∈ event
f ∈ function
d ∈ define
i ∈ inst
b ∈ block

Primitiver:

t ∈ time
now ∈ now
w ∈ wait
n ∈ num (Float | Int)
s ∈ String

Opbygningsregler:

a ::= n | x | a1 + a2 | a1 * a2 | a1 - a2 | a1 / a2 | (a1)
b ::= var signal: n | var signal: toggle | a1 == a2 | a1 < a2 | a1 > a2 | -b1
| b1 && b2 | b1 | | b2 | now > t | now == t | now < t
e ::= 'when' b block | every time B
f ::= returnType ID (param) B
d ::= #define (device | signal) typeID input ((toggleID = value)*) | range
i ::= typeID varID = IP:Port
B ::= If(b) B else B1 | if(b) B | while(b) B | assignment | funcCall(param)

```
P ∈ Prog - Program
D ∈ Decl - Declaration
P ::= D
D ::= Type T; D
        | var x; D
        | event bS; D
        | func f(x) = e; D
T::= int | float | bool | string
       | Signal(T)
       | device (Tvektor)
```

A program consist of one or more declarations. These declarations consists of either type, var, event or function declarations. A type can be of either int, float, bool, string, signal or device. The b in the events symbols a boolean expression. The S inside the event brackets symbolises a statement. A statement consists of variable declaration, assignments, control structures and function calls. The x inside the func represents the arguments passed to the function. The e at the end is the return value of the function.

~~4.4 Target language~~

~~In this section it will be analysed which target language will be chosen and why. Furthermore it will be analysed how the language should be compiled from the source language to the target language.~~

~~4.4.1 Compiling the code~~

~~We have decided to compile our program to Java, which has its own just in time compiler, that compiles to Java Bytecode and then to machine code for all systems, that has support for the java virtual machine. This way the Java compiler, which is known to work, can be used to get to the desired machine code.~~

~~Another good reason for choosing Java as the target language is that it is one of the popular general purpose languages currently used for writing home automation software[9]. This means that programmers using the Pivot language will likely also understand and be able to manually modify the output program written in Java. C#, C++, Python and other general purpose languages could also have been used as target language, but was decided against due to this group having more experience using Java.~~

Bibliography

[1] emory college of arts and sciences. *The dangling else ambiguity*. `http://www.mathcs.emory.edu/~cheung/Courses/561/Syllabus/2-C/dangling-else.html`. 2019.

[2] Calaos. *Calaos official website*. `https://calaos.fr/en/`. 2019.

[3] Jacob Bach Hansen Søren Jensen Kenneth Toft Rasmussen Marc Kjær Deleuran Michael Tarp Dennis Vestergaard Værum Frederik Madsen Halberg. *Home Automation*. `https://projekter.aau.dk/projekter/files/198203443/Report_Home_Automation.pdf`. 2014.

[4] Thom Dietrich. *openHABian hassle-free openHAB Setup*. `https://community.openhab.org/t/openhabian-hassle-free-openhab-setup/13379/670`. 2017.

[5] Domoticz. *Domoticz official manual*. `https://www.domoticz.com/DomoticzManual.pdf`. 2019.

[6] HomeAssistant. *HomeAssistant official website*. `https://www.home-assistant.io/`. 2019.

[7] IBM. *The Internet Of Things (IOT) Will Be Massive In 2018*. `https://www.forbes.com/sites/bernardmarr/2018/01/04/the-internet-of-things-iot-will-be-massive-in-2018-here-are-the-4-predictions-from-ibm/#22dad192edd3`. 2018.

[8] Ikea. *Smart belysning*. `https://www.ikea.com/dk/da/catalog/categories/departments/lighting/36812/`. 2019.

[9] IoTforall. *Top 3 programming languages*. `https://www.iotforall.com/2018-top-3-programming-languages-iot-development/`. 2018.

[10] MisterHouse. *MisterHouse official website*. `http://misterhouse.sourceforge.net/`. 2019.

[11] Office for National Statistics. *Families and Households: 2017*. `https://www.ons.gov.uk/peoplepopulationandcommunity/birthsdeathsandmarriages/families/bulletins/familiesandhouseholds/2017`. 2017.

[12] OpenHAB. *OpenHAB official website*. `https://www.openhab.org/`. 2019.

[13] Philips. *Personal wireless lightning*. https://www2.meethue.com. 2019.

[14] safewise. *What is Home Automatiion*. https://www.safewise.com/home-security-faq/how-does-home-automation-work/. 2018.

[15] Robert W. Sebesta. *Concepts of Programming Languages*. 10th. Pearson, 2012. ISBN: 0273769103, 9780273769101.

[16] Statista. *Number of Smart Homes forecast in the United Kingdom from 2017 to 2023 (in millions)*. https://www.statista.com/statistics/477134/number-of-smart-homes-in-the-smart-home-market-in-the-united-kingdom/. 2019.

[17] Arie Van Deursen, Paul Klint, and Joost Visser. "Domain-specific languages". English. In: 35 (2000), pp. 26–36. ISSN: 0362-1340. DOI: 10.1145/352029.352035. URL: https://sfx.aub.aau.dk/sfxaub?sid=google&auinit=A&aulast=Van+Deursen&atitle=Domain-specific+languages:+An+annotated+bibliography&id=doi:10.1145/352029.352035&title=ACM+SIGPLAN+Notices&volume=35&issue=6&date=2000&spage=26.