ARD

Preparation of a software infrastructure for research in Lego-based software engineering

Chapter 1: Introduction

1.1. The Problem Domain

In BGU university there are software engineers who research reactive systems. They gathered into lab dedicated to Lego and raspberry-pi hardware experiments.

Our project designed give them comfort software infrastructure for the research.

An API should be created in java and python for working with sensors and operators of Lego and Grove companies and create an infrastructure for adding additional hardware in the future.

1.2. Context

How the system work?

The system gets from the user JS file. In the file there is the code of the user written in BPjs.

The file is parsed by the system to events of commands that send to the robot, and the robot actuate according to the commands.

1.3. Vision

The goals of the projects are:

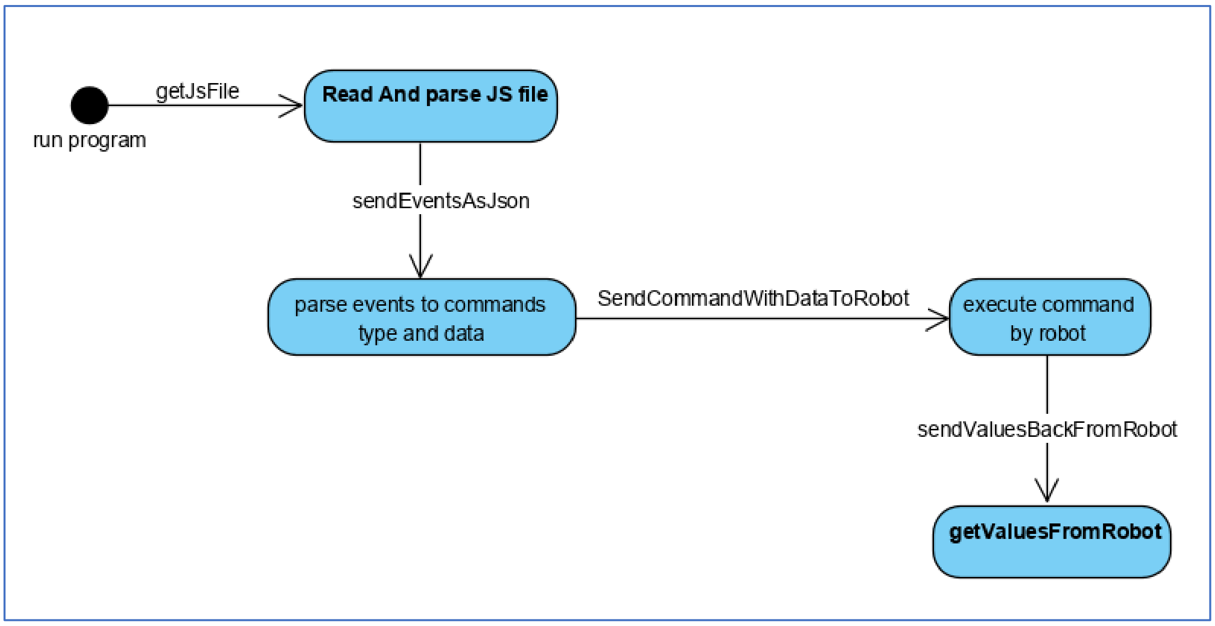
- Expanding support for the behavioral programming paradigm
- Create a comfortable programming infrastructure in behavioral programming in a simple way.
- Creating an adaptive infrastructure that adapts to add new hardware and software for future development.

1.4. Stakeholders

The researchers in the robotics laboratory, graduate students at Ben-Gurion University.

1.5. Software context

In high-level description of the software system, we can see the chart below:



The input is the BPjs program and the output is the values that return from the robot and the side effects from the actions of the robot.

Some of the use-cases in the system: BPjs program that contain events who can cause the robot: drive, rotate his parts, set the values of his sensors etc.

Chapter 2: Usage Scenarios

2.1. Use cases

All the use-cases in the system are detailed in the ADD file.

2.2. Special usage consideration

- Knowledge of robotics
- Programming knowledge in BPjs
- Robot based on raspberry pi, branded with the Raspberry Pi Image operating system, with the option of adding a Lego EV3 hardware system and a Grove Pi kit.
- Installation of RabbitMQ communication infrastructure used to transfer commands in the system and configure user permissions.

Chapter 3: Functional Requirements

- The programmer can create event to register all the boards and ports used by the sensors in the system.
- The programmer can create event to get update values from the sensors by the ports they connected to.
- The programmer can create event to stop getting update values from the sensors by the ports they connected to.

- The programmer can create event to set the value of the sensors of the robot.
- The programmer can create event cause robot drive with the desired speed
- The programmer can create event cause robot rotate his motors with the desired speed and angel.
- The programmer can use with number of boards.

Chapter 4: Non-Functional Requirements

- Simple installation.
- Provide efficient communication in aspect of speed.
- Add hardware in a convenient way that does not require redesign.
- Add software in a convenient way that does not require redesign.

4.1. Implementation constraints

Performance

This is reactive system, the reactions of the robot to the commands which sent him need to be faster. The system needs to save on sending unnecessary messages and based on fast communication between processes. Yet the EV3 have hardware restriction, it is slow in relation to the speed at which events are sent.

Reliability & Stability

The user needs to know exactly what he is doing wrong in writing the code. Therefore, specific errors will be thrown out for incorrect code.

Adaptation

The system should be adapted for expansion into additional programming languages and support for adding software and hardware. Therefore interfaces should be used for the classes and functions supported through it

Usability

Users are expected to have programming knowledge in BP and knowledge in robotics. They need to know about threading to understand how the program could be running.

4.2. Platform constraints

In reactive system the communication must be faster. After researching the different ways of communication, we found the RabbitMQ as message broker, who pass massages between the producer and consumer, and balances very well between high performance and comfort.

4.2.1. SE Project constraints

The input to the system will be given in advance by the file that contain the BPjs code. The output is influenced by the physical boards and the environment in which they run.

Therefore, the system is limited to specific input structures and individual types of boards (raspberry, grove and EV3). That restrictions are extensively detailed in the code documentation.

Because the system actuates physical robot, we don't want the robot to be connected at any running of the system in the development processing. In addition, the unit tests will test the code without be depend on the physical robot. As a result, we use mock and fake classes to imitate the performance of the robot.

4.3. Special restrictions & limitations

There may be several issues if one of the installations is not performed properly. In addition, the code may be used incorrectly by the programmer and as a result the program may get stuck or behave differently than expected.

Chapter 5: Risk assessment & Plan for the proof of concept

Plan for the proof of concept

Proving the correctness of the system in the eyes of the customer and meeting the requirements for the outgoing prototype, will be measured by a practical examination of the robot's response for the required triggers and compliance with it as agreed in each round of development and publication of the prototype.

Risk assessment

- An update of the Lego company to its kits that will disrupt the interface we created
- Defective hardware in the developers
- Slow development in relation to the given time
- Construction of incorrect communication configuration

Chapter 6: Appendences

Glossary

API

Interface for the user, which represents the library of which functions can be called that directly support the operations of the robot.

Behavioral programing

Behavioral Programming (BP) is an approach and technique for software development, which enables incremental development in a natural way. A behavioral application consists of threads of behavior each of which represents an independent scenario that the system should and shouldn't follow. These independent behavior threads are interwoven at run-time yielding integrated system behavior. For example, in a game-playing application, each game rule, and each playing strategy would be programmed separately and independently with little or no awareness of other modules.

BPjs

BPjs is a library for executing behavioral programs written in Javascript. It can be used from the commandline, or embedded in a Java application.

Raspberry Pi

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python.

EV3

Lego Mindstorms EV3 (evolution 3) is the third generation robotics kit in Lego's Mindstorms line. The EV3 Home (31313) set consists of: 1 EV3 programmable brick, 2 Large Motors, 1 Medium Motor, 1 Touch Sensor, 1 Color Sensor, 1 Infrared Sensor, 1 Remote Control, cables, USB cable, and 585 TECHNIC elements.

Grove Pi

GrovePi is an add-on board that brings Grove Sensors to the Raspberry Pi. It is consists of: 7 digital Ports, 3 analoge Ports, 3 I2C ports, 1 Serial port connect to GrovePi, 1 Serial port connect to Raspberry Pi and Grove header Vcc output Voltage: 5Vdc

RabbitMQ

RabbitMQ is a messaging broker - an intermediary for messaging. It gives your applications a common platform to send and receive messages, and your messages a safe place to live until received.