ev3OSEK as Robotics Use Case for mbeddr

Achim Stuy – Zwickau University of Applied Sciences – achim.stuy.051@fh-zwickau.de

Abstract

In the course of porting the nxtOSEK platform to LEGO's most recent MINDSTORMS generation, EV3, as part of a university project one task was to develop an IDE to create applications for the ev3OSEK. mbeddr seemed suitable for this task as it provides both a C IDE as well as a mechanism to create domain specific languages, e.g., for the OSEK Implementation Language.

This article describes the implementation of this task, evaluates mbeddr, explains the basic implementation of the OSEK IDE, and proposes possible further projects.

Motivation

OSEK applications consist of two inter-dependent parts: application and operating system configuration files (OIL) and the actual application source code (implemented in the C programming language). A language workbench like mbeddr provides for taking care of all the dependencies between these two OSEK development artifacts. Furthermore, through the projectional editing approach, the IDE can provide the user with additional information and metadata about the element being currently edited.

Background

mbeddr

mbeddr is a set of integrated and extensible languages for embedded software engineering, plus an IDE. mbeddr builds on JetBrains MPS, a tool for efficient building, extending and composing languages and their IDEs. MPS uses a projectional editor. Consequently, MPS supports non-textual notations such as tables, mathematical symbols, and graphics. (mbeddr)

OSEK

OSEK/VDX is a joint project of the automotive industry. It aims at an industry standard for an open-ended architecture for distributed control units in vehicles.

A real-time operating system, software interfaces and functions for communication and network management tasks are thus jointly specified.

The term OSEK stands for "Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug" (Open systems and the corresponding interfaces for automotive electronics). The term VDX means „Vehicle Distributed eXecutive“. The functionality of OSEK operating system was harmonised with VDX. For simplicity OSEK will be used instead of OSEK/VDX in the document. (OSEK/VDX)

ev3OSEK

ev3OSEK is the implementation of the OSEK operating system for the LEGO MINDSTORMS EV3 robots developed by the Informatics department of Zwickau University of Applied Sciences. ev3OSEK is partially a port of nxtOSEK, an OSEK implementation for the previous generation of LEGO robots.

Why mbeddr?

One of the requirements of the project was the development of tooling for creating OIL files. Besides mbeddr this requirement could also be solved with the help of Acceleo or Xtext.

Acceleo is a pragmatic implementation of the Object Management Group (OMG) MOF Model to Text Language (MTL) standard. (Acceleo) Thus, after the definition of the OIL in a meta model, Acceleo offers model-based creation of OIL files which are then converted to text files using Acceleo.

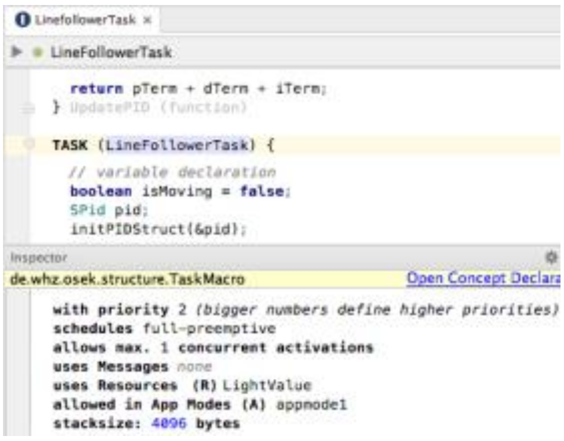
Xtext is a framework for the development of programming languages and domain-specific languages. Xtext allows to define user-defined languages using a powerful grammar language. Based on the defined language, tooling infrastructure can be automatically created, including a parser, linker, typechecker, and compiler as well as editing support for Eclipse, IntelliJ IDEA and most popular web browsers. (Xtext)

Therefore, both approaches allow the definition and text generation of OIL files. Their big disadvantage, however, is the missing integration of C source code. This is where MPS / mbeddr offers much more support: With MPS it is possible to design extensible DSLs and start using them right away to build end-user applications. (JetBrains) The OIL, thus, only needs to be implemented as a new domain-specific language. Furthermore, since C is implemented as DSL by mbeddr, a full integration of OIL and C becomes possible by simply reusing and extending mbeddr.

Implementation

The starting point is an Oil File being a Module just like Internal and External Modules. Because the System Generator converts the elements defined in the OIL file into C code, the OIL file is conceptually similar to Internal and External Modules.

One of OSEK's central concept are Tasks. As the System Generator converts them to Function Prototypes, the Task Declarations are inherited from Function Prototypes. MPS allows the definition of a so-called behavior aspect which holds methods of concepts that can be invoked on nodes of the model, just like methods of classes in OOP. The feature was used here to set the Type of the Tasks to `void` initially. The Tasks' implementations are defined by a Macro which converts the definitions to functions. Task implementation definitions are, therefore, implemented as Functions in ev3OSEK. MPS as a projectional editor supports this approach by offering IDE functionality to edit a task's OIL-properties directly from the implementation through the inspector window. A demonstration of this feature can be seen in Figure 1.



```
    return pTerm + dTerm + iTerm;
  } UpdatePID (function)

  TASK (LineFollowerTask) {
    // variable declaration
    boolean isMoving = false;
    SPid pid;
    initPIDStruct(&pid);
```

de.whz.osek.structure.TaskMacro

```
with priority 2 (bigger numbers define higher priorities)
schedules full-preemptive
allows max. 1 concurrent activations
uses Messages none
uses Resources  (R) LightValue
allowed in App Modes (A) appmode1
stacksize: 4096 bytes
```

Figure 1: Demonstration of the projectional editor

All macros defined in the `kernel.h` of ev3OSEK were decided not to be resolved, but instead MPS' generator will output the unresolved macros. This was decided for compatibility reasons, because so the System Generator and `kernel.h` can be easily replaced in order to build OSEK applications for new systems.

Further central concepts of OSEK are Events and Resources. As the System Generator converts them to Global Constant Declarations, they are implemented like them in ev3OSEK. In order to reference the Events and Resources from the implementation the OilFile needs to be referenced in the “imports” section of the Implementation Module. Through this mechanism, the links can be normal mbeddr Global Constant References and no further implementation is needed.

Using MPS’ typesystem aspect enables consistency checks across several modules. For example, OSEK allows to declare hooks for startup, shutdown, pretask, posttask and error, which are enabled in the OIL file. In the typesystem aspect a check was implemented, whether also a correspondding method was declared if the user enables one of the hooks.

Last but not least, some adjustments in the BuildConfiguration were necessary. mbeddr’s Executable requires a main function to be declared which is not necessary in OSEK applications. So an OsekExecutable was defined which drops the check for the main function. Furthermore, additional Platforms for NXT and EV3 as targets were implemented.

Remaining Issues

The OSEK mbeddr extension was built for one specific use case: ev3OSEK. Thus, the abstraction will not be flawless.

For example, the ev3OSEK directory is referenced in the final make files and so needs to be referenced in the EV3-Platform. It could be smarter to define it in the MPS settings but that is not sure.

Future Work

mbeddr offers extensions to C through components and state charts.

One interesting issue is the mapping of sender/receiver interfaces to OSEK concepts. The most obvious implementation of this appears to be OSEK messages, which are unfortunately not implemented in ev3OSEK and thus could not be tested.

A second promising issue of components is the time triggered invocation of methods which resembles to OSEK’s alarms. Unfortunately, this feature is not completely implemented in mbeddr and for that reason could not be implemented in mbeddr’s OSEK extension.

Finally, mbeddr allows the use of state machines in order to get a better overview of complex applications. These could be expanded by annotations in order to add OSEK specific information directly to affected elements.

References

Acceleo. http://www.eclipse.org/acceleo/

JetBrains. *MPS overview*. https://www.jetbrains.com/mps/

mbeddr. http://mbeddr.com/

OSEK/VDX. *Binding Specification*. Version 1.4.2. July 15th, 2004

Xtext. *Xtext - Language Engineering Made Easy!* https://eclipse.org/Xtext/