Extensible Languages
DSLs, and Embedded

Bernhard Merkle
SICK AG
bernhard.merkle@gmail.com

Markus Voelter
Independent/Itemis
voelter@acm.org
Two Classes in Embedded Systems Development

(yes, this is a slight simplification)
Plan Oriented
Top Down
Big Systems
Big Companies

Modeling

Grown
Bottom Up
Small/Medium Systems
Smaller Companies

C Code
programming?
modelling?

programming? vs. modelling?
different?
the same?

Why
the difference?
History?

Why
the difference?
It is time for ...

... a Different Perspective
We don’t want to model, we want to program!

... at different levels of abstraction
... from different viewpoints
... integrated!
We don’t want to model, we want to program!

... with different degrees of **domain-specificity**
... with suitable **notations**
... with suitable **expressiveness**

We don’t want to model, we want to program!

And always: **precise and tool processable**
Modular Languages
Big Language?

with many first class concepts!

Small Language?

with a few, orthogonal and powerful concepts
Modular Language

with many optional, composable concepts

Modular Language

Like frameworks and libraries,
Modular Language

Like frameworks and libraries, but with syntax and IDE support.
Incremental Extension of C

Components
Tasks
State Machines
Physical Units
Special Data Types
Incremental Extension of C syntactically and semantically integrated

Incremental Extension of C extensible with domain-specific constructs (DSLs)
Projectional Editing
Parser-based

- text
  - ... to tree
  - ... to text

Projectional

- tree
  - ... to text-lookalike (editor)
  - ... to other trees ... [:]
  - ... to text
Programming as Modeling

... (Mostly) Graphical Notations
... Abstract Syntax Storage
... Projecting Editors
... Different editable views for model
Language Composition

There’s no parsing.
Unique Language Element Identity.
Unlimited language composition.

Flexible Notations

Textual like ASCII
Graphical box & line
Semi-Graphical mathematical

} treated the same can be mixed
Automatic IDE Extension

Tool support is inherent for languages build with projectional tools.

Multiple Notations

... for the same concepts
e.g. in different contexts or for different tasks
Partial Projections

... different views
... for different roles/people
... only a particular variant

Storage != Schema

... store arbitrary meta data
  change log
  conflicting information
  variability annotations

... independent of language schema!
... „aspects“, overlay
Live Programs

think: spreadsheet

a change to one part of program can lead to (dependent) changes in other parts

Tree Editing

... is different from editing text

... try to make it feel like text

... takes some getting used to

but: for more flexible notations a more general editing paradigm is needed
Infrastructure Integration

... storage is not text
... diff/merge must be in tool
... existing text tools don’t work

Proprietary Tools

... no standards
... no interop
also do...

IntelliJ IDEA
Resharper

released in
Q3 2009
currently
1.5.1
April 2011
2.0
Open Source
licensed under
Apache 2.0
The Showcase: Target Platform
NXT Brick
32-bit ARM7@48MHz
256kB Flash
64kB RAM

NXT Brick
100x64 LCD
USB 2.0
Bluetooth
OSEK/VDX

Embedded OS used in (small) automotive systems

OSEK/VDX

– Communication
  (data exchange within and between control units)

– Operating System
  (real-time execution of ECU software and base for the other OSEK/VDX modules)

– Network Management
  (Configuration determination and monitoring)
helloworld.c

```c
/* helloworld.c */
#include "kernel.h"
#include "ecrobot_interface.h"

/* nxOSEK hook to be invoked from an ISR in category 2 */
void user_icall_isr_type2(void){ /* do nothing */ }

TASK(OSEK_Task_Background)
{
    while(1){
        ecrobot_status_monitor("Hello, World!");
        systick_wait_ms(500); /* 500msec wait */
    }
}
```

implementation.oil

```c
#include "implementation.oil"

CPU ATMEL_AT91SAM7S256
{
    OS LEOS_OSEK
    {
        STATUS = EXTENDED
        STARTUPHOOK = FALSE
        ERRORHOOK = FALSE
        SHUTDOWNHOOK = FALSE
        PRETASKHOOK = FALSE
        POSTTASKHOOK = FALSE
        USBGETSERVICEID = 0
        USEPARAMETERACCESS = FALSE
        USEPSSCHEDULER = TRUE
    }

    /* Definition of application mode */
    APPMODE appmode();

    /* Definition of OSEK_Task.Background */
    TASK OSEK_Task_Background
    {
        AUTOSTART = TRUE
        {
            APPMODE = appmode();
        }
        PRIORITY = 1; /* Lowest priority */
        ACTIVATION = 1;
        SCHEDULE = NULL;
        STACKSIZE = 512;
    }
};
```
Lejos-OSEK

nxtOSEK: OSEK platform for Lego NXT
– realtime, fast execution, memory 10k
– device driver of leJOS NXJ C/Assembly
– Features:
  • ANSI C/C++ programming env using GCC tool chain
  • C/C++ API for NXT Sensors, Motor, and other devices
  • TOPPERS/ATK/JSP real-time multi tasking features
– main devs: Takashi Chikamasa, Masaaki Mizuno

http://lejos-osek.sourceforge.net/
The Showcase:
The Demo
The End.

coordinates

web www.voelter.de
email voelter@acm.org
skype schogglad
xing http://www.xing.com/profile/Markus_Voelter
linkedin http://www.linkedin.com/pub/0/377/a31

web www.bmerkle.de
email bernhard.merkle@gmail.com
xing http://www.xing.com/profile/Bernhard_Merkle
linkedin http://www.linkedin.com/pub/bernhard-merkle/0/b73/738