Domain-Specific Languages for Embedded Systems
Portable Software Development

Vera Ivanova, Boris Sedov, Yuriy Sheynin, Alexey Syschikov
{vera.ivanova, boris.sedov, sheynin, alexey.syschikov}@guap.ru
What is DSL

Domain-Specific Language (DSL) – is a programming or modeling language designed for a particular domain area. Unlike general-purpose languages, DSLs are:

• more expressive
• easier to use
• more understandable

MATLAB Simulink Language

HTML - HyperText Markup Language
Why do we need DSL technology

Task Solution

General tools and approach

+ wide community
+ well documented tools & approaches
+ lots of legacy code
- not optimized for specific domains
- good for programmers, bad for experts

DSL-based tools and approach

- small or non-existing community
- lack of trusted tools & approaches
+ more benefits for experts
+ optimized for a particular domain(s)
+ active results reuse inside a domain

Use of DSL can significantly accelerate development process by involving both experts and programmers, but it needs right implementation
1. Decision about developing a new DSL
   *Making decision basing on patterns, knowledge, experience.*

2. Domain analysis
   *The problem domain is identifying and domain knowledge is gathering.*

3. Language design
   *New language is developing.*

4. Language implementation
   *Tools for language implementation is developing.*
**DSL development: language design**

Is existing language used?

- **yes**
  - “Base” language exploitation
    - Piggyback: Base language with new functions
      - Qt, SystemC, yacc, lex
    - Specialization: Base language restricted, removed unsafe aspects
      - Javalight, HTML
    - Extension: Base language extended with new constructs
      - Java generics, CoArray, Fortran, OpenMP

- **no**
  - Inventing DSL from scratch
1. Interpreter
2. Compiler/application generator
3. Preprocessor
   - Macro-processing
   - Source-to-source (DSL to base language) transformation
   - Pipeline of DSLs
   - Lexical processing (DSL renames base language lexemes)
4. Embedding (DSL is base language + new functions)
5. Extensible compiler/interpreter
Our proposal: visual approach

• Intuitive use of graphical notation
• Natural parallelism presentation
• Automated pipelining
• Integration of graphical and textual languages
• Various granularity levels
Our proposal: visual approach

• Intuitive use of graphical notation
• Natural parallelism presentation
• Automated pipelining
• Integration of graphical and textual languages
• Various granularity levels
Our proposal: visual approach

- Intuitive use of graphical notation
- Natural parallelism presentation
- **Automated pipelining**
- Integration of graphical and textual languages
- Various granularity levels
Our proposal: visual approach

• Intuitive use of graphical notation
• Natural parallelism presentation
• Automated pipelining
• Integration of graphical and textual languages
• Various granularity levels
Our proposal: visual approach

- Intuitive use of graphical notation
- Natural parallelism presentation
- Automated pipelining
- Integration of graphical and textual languages
- Various granularity levels
Proposed method

Domain specific languages are built on the base of general purpose graphic language VPL.

Building new elements for a coarse-grained library

Sub-scheme of low-level elements

Subroutine text on target platform language

```c
int dhGetYCbCr(DataLink *in11, DataLink *out21) {
    int p;
    memcpy(&p, in11->Data, 4);
    CImg<double>* data = (CImg<double>*)p;
    CImg<double>* res = new CImg<double>(data->get_RGBtoYCbCr());
    p = (int)(res);
    WriteReference(out21, p);
    delete data;
    return 0;
}
```
Development environment VIPE

Base elements library

Generate code and execute applications

Several regimes of showing information

Schemes validation and debugging
Example: making a DSL for image processing

1. Analyze domain of image processing

2. Create a library of new language functional elements

3. Define C++ & OpenCV functional for new elements
Example: using the DSL for image processing

Image recognition (OpenCV)

4. Build a scheme from base elements and new functional blocks
Example: using the DSL for image processing

Face/eyes detection (OpenCV)

4. Build a scheme from base elements and new functional blocks
Conclusions

We propose the DSL-based approach with:

• Advantages of visual DSLs approach
• New method of DSLs development
• Easy construction of DSLs specially for your domain
• Design tools support

We:

• use this method for DSL in image processing domain
• work on exploration of this method in other domains
• work on full tool flow and present it in the next presentation