16th FRUCT Conference

Domain-Specific Languages for Embedded Systems Portable Software Development

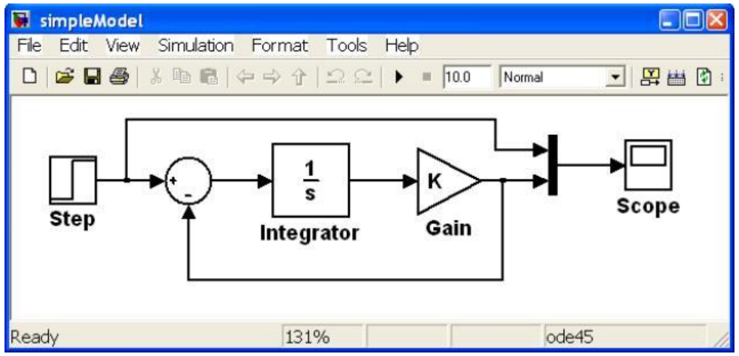
*Vera Ivanova, Boris Sedov, Yuriy Sheynin, **Alexey Syschikov***

{vera.ivanova, boris.sedov, sheynin, alexey.syschikov}@guap.ru

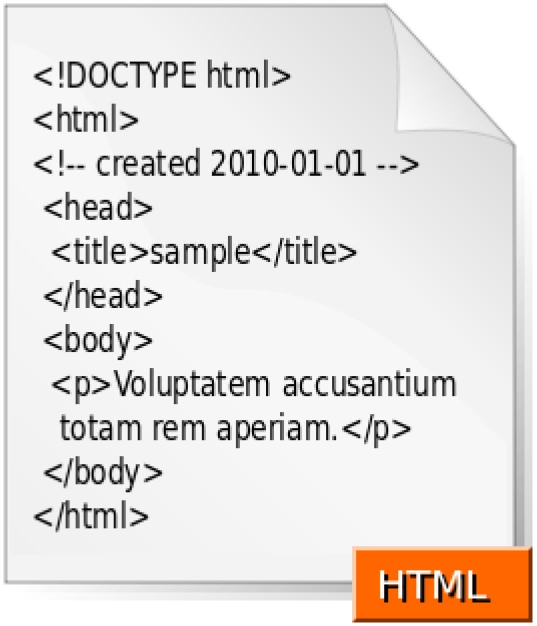
What is DSL

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



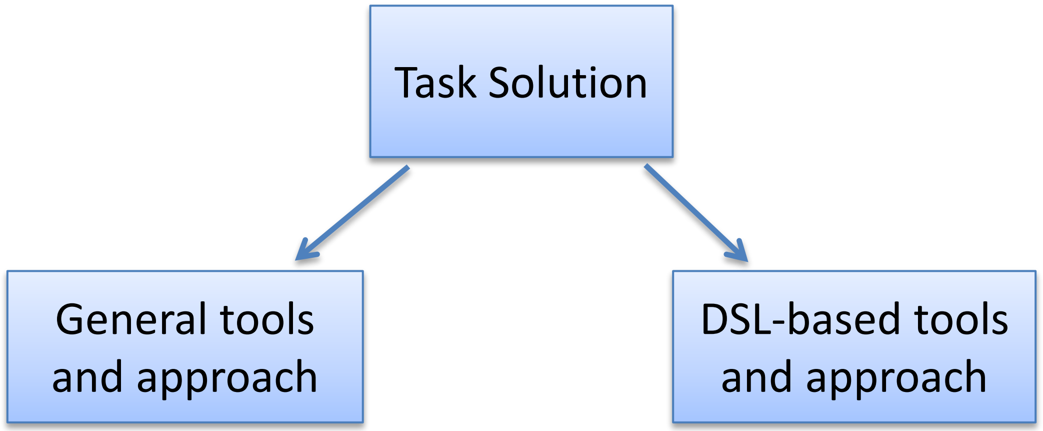
MATLAB Simulink Language



```
<!DOCTYPE html>
<html>
<!-- created 2010-01-01 -->
 <head>
  <title>sample</title>
 </head>
 <body>
  <p>Voluptatem accusantium
   totam rem aperiam.</p>
 </body>
</html>
```

HTML - HyperText Markup Language

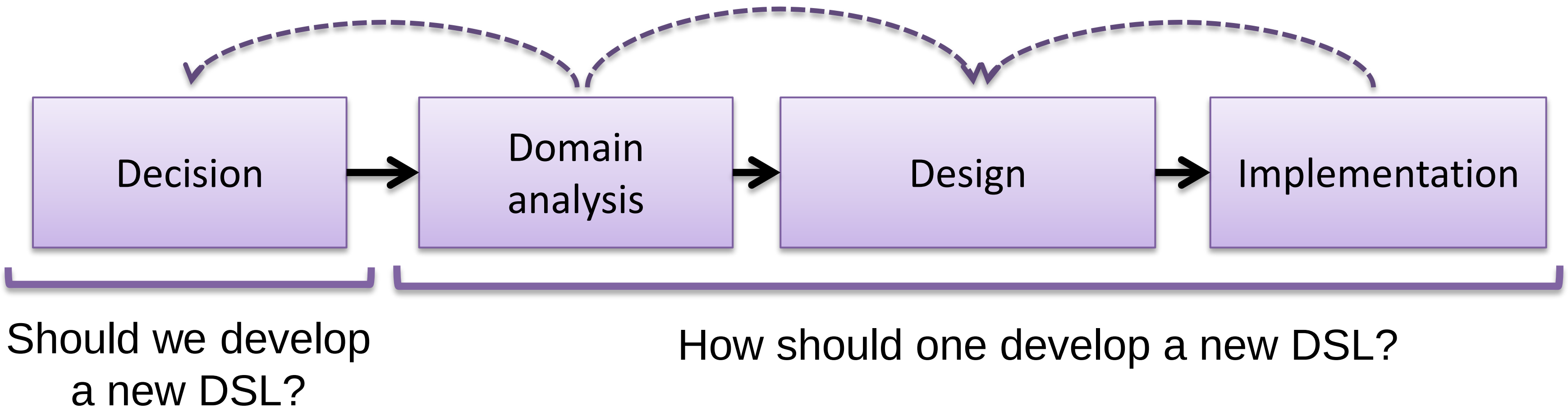
Why do we need DSL technology



General tools and approach	DSL-based tools and approach
+ wide community	- small or non-existing community
+ well documented tools & approaches	- lack of trusted tools & approaches
+ lots of legacy code	+ more benefits for experts
- not optimized for specific domains	+ optimized for a particular domain(s)
- good for programmers, bad for experts	+ active results reuse inside a domain

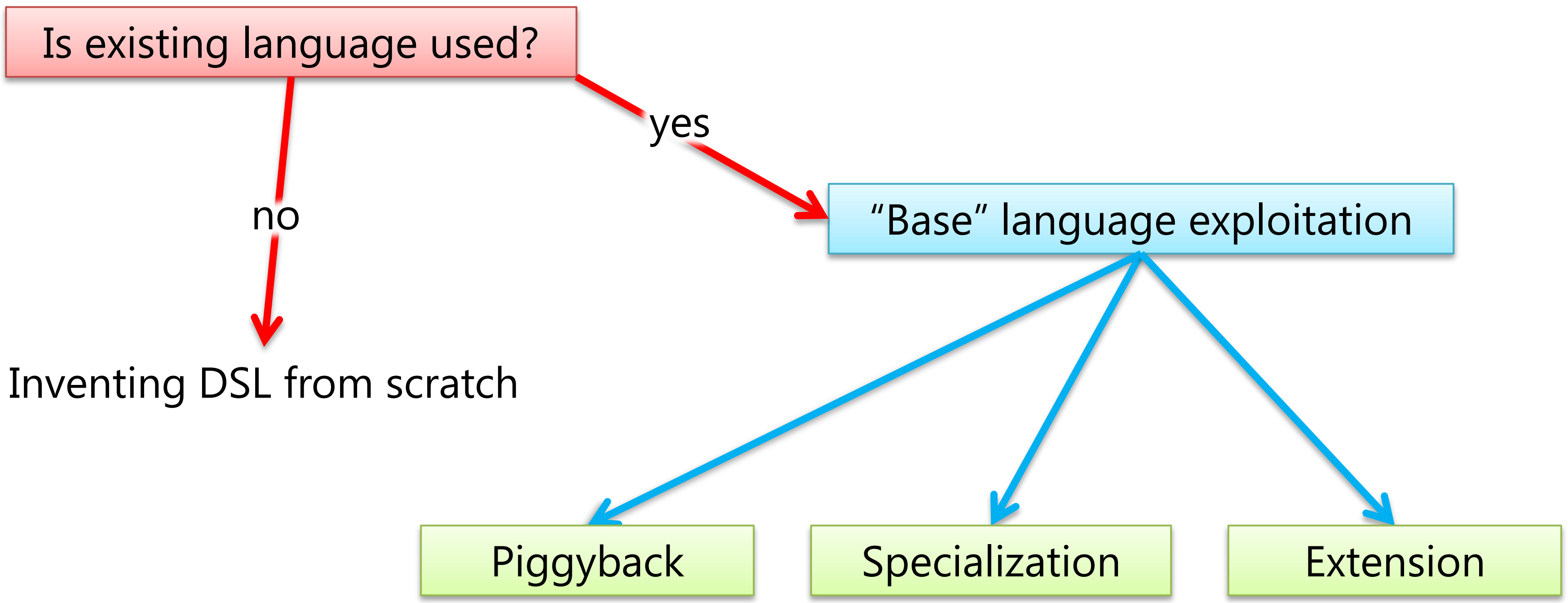
Use of DSL can significantly accelerate development process by involving both experts and programmers, but it needs right implementation

DSL development cycle


Decision
Domain analysis
Design
Implementation
Should we develop a new DSL?
How should one develop a new DSL?

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
no
Inventing DSL from scratch
"Base" language exploitation
Piggyback
Specialization
Extension

Base language with new functions

Base language restricted, removed unsafe aspects

Base language extended with new constructs

Qt, SystemC, yacc, lex

Javalight, HTML

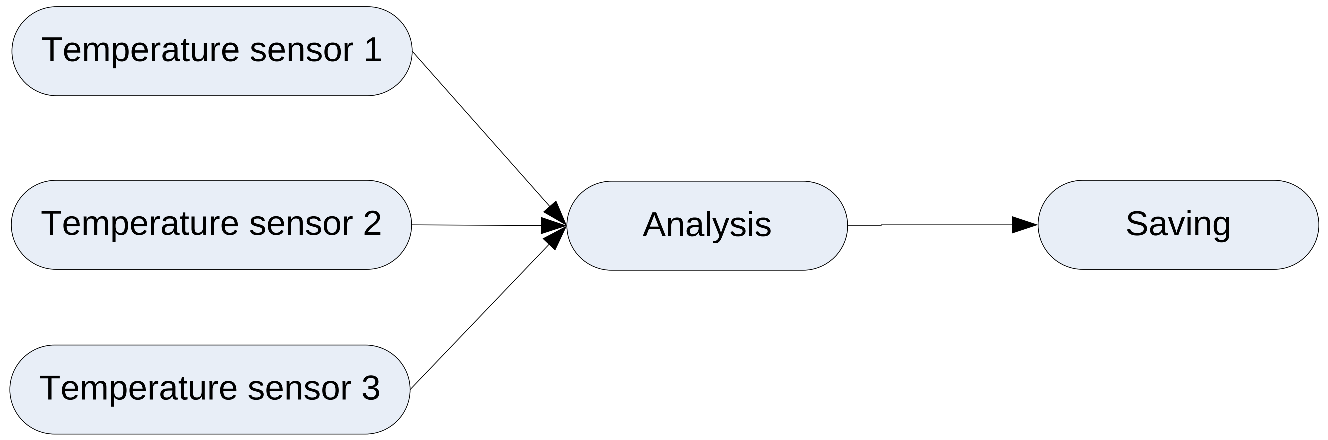
Java generics. CoArray Fortran, OpenMP

DSL development: language implementation

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

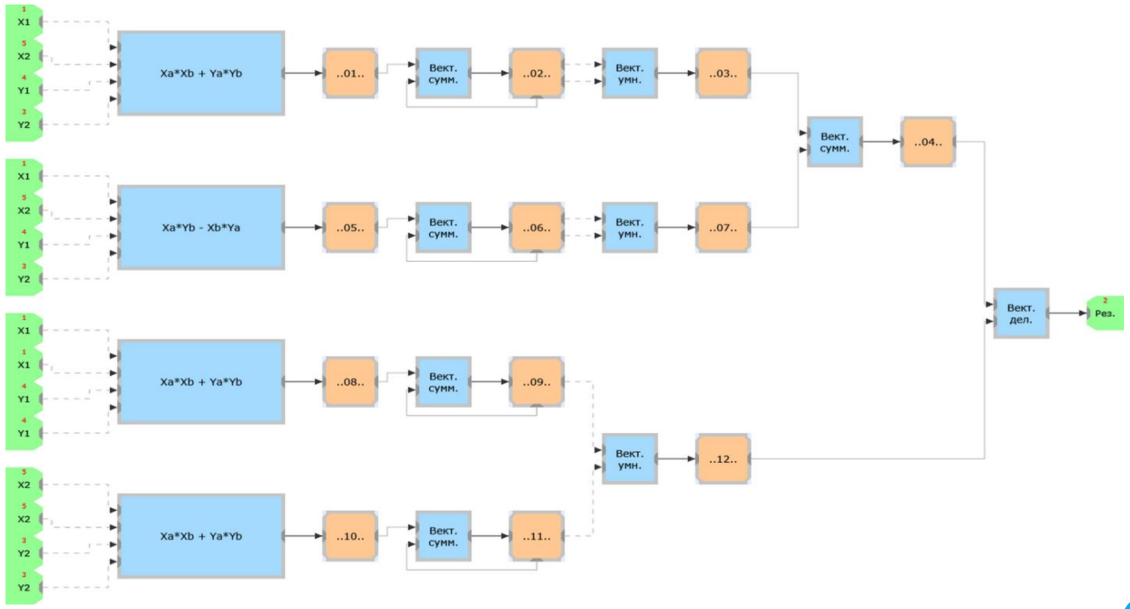
Our proposal: visual approach

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



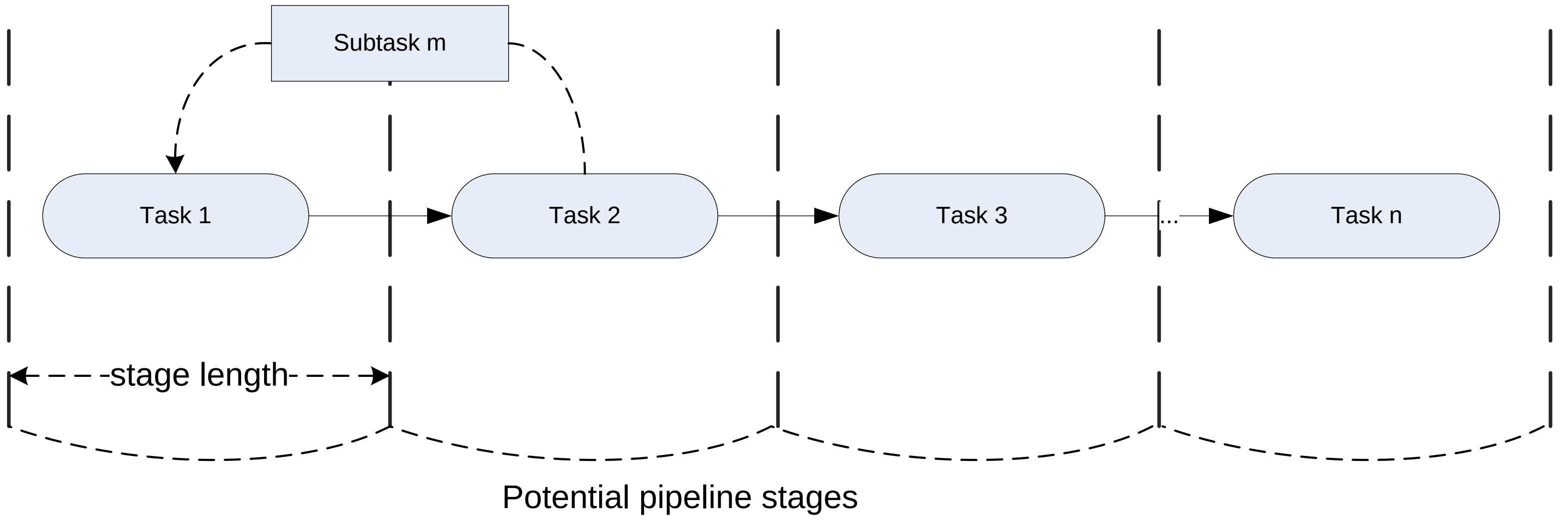
Our proposal: visual approach

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



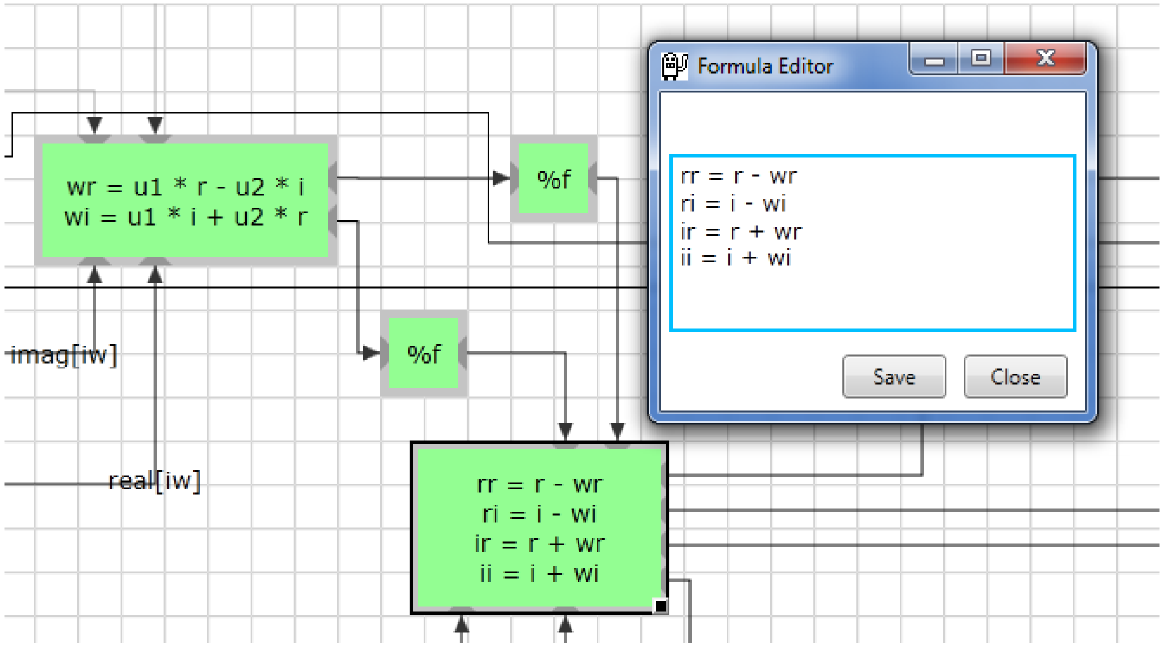
Our proposal: visual approach

- 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

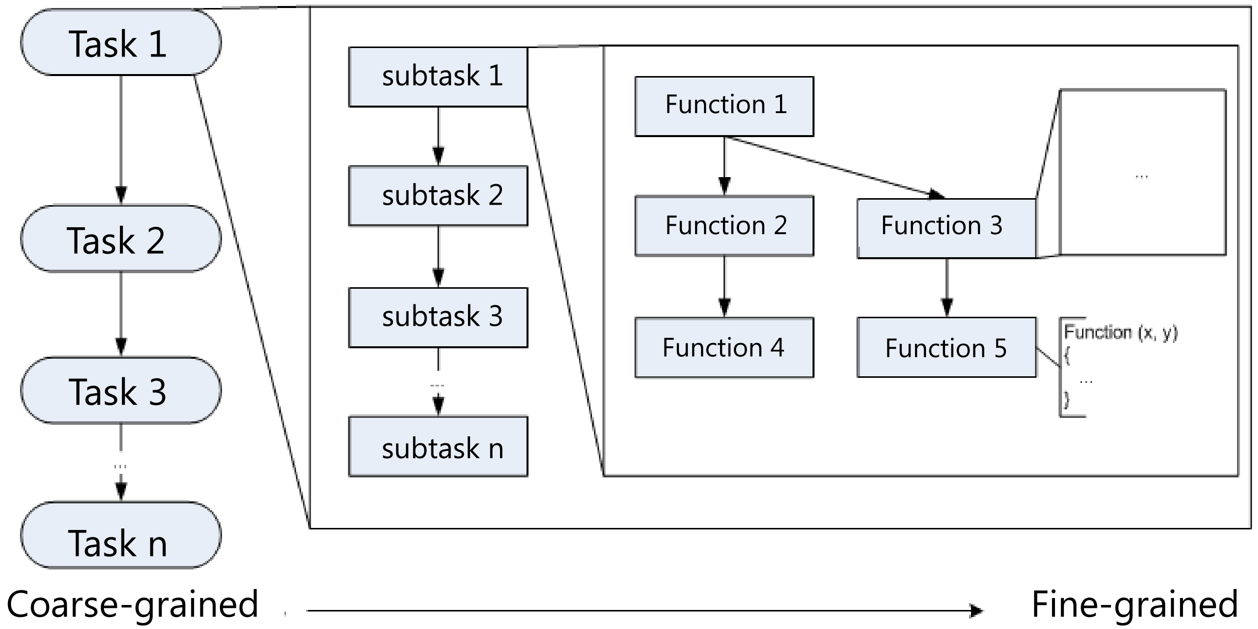


```
wr = u1 * r - u2 * i
wi = u1 * i + u2 * r
```

```
rr = r - wr
ri = i - wi
ir = r + wr
ii = i + wi
```

Our proposal: visual approach

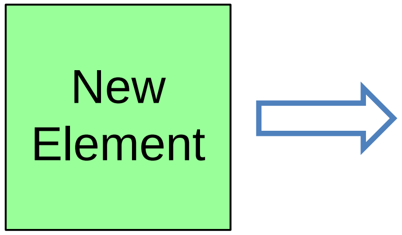
- Intuitive use of graphical notation
- Natural parallelism presentation
- Automated pipelining
- Integration of graphical and textual languages
- Various granularity levels



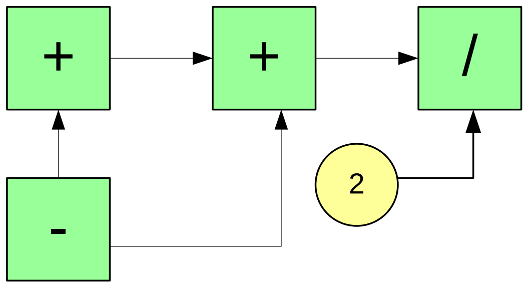
Proposed method

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

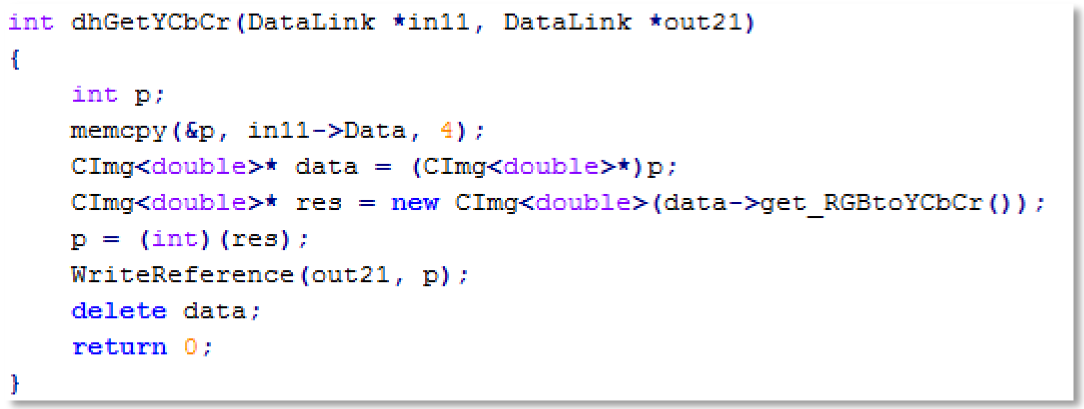
Building new elements for a coarse-grained library



Sub-scheme of low-level elements

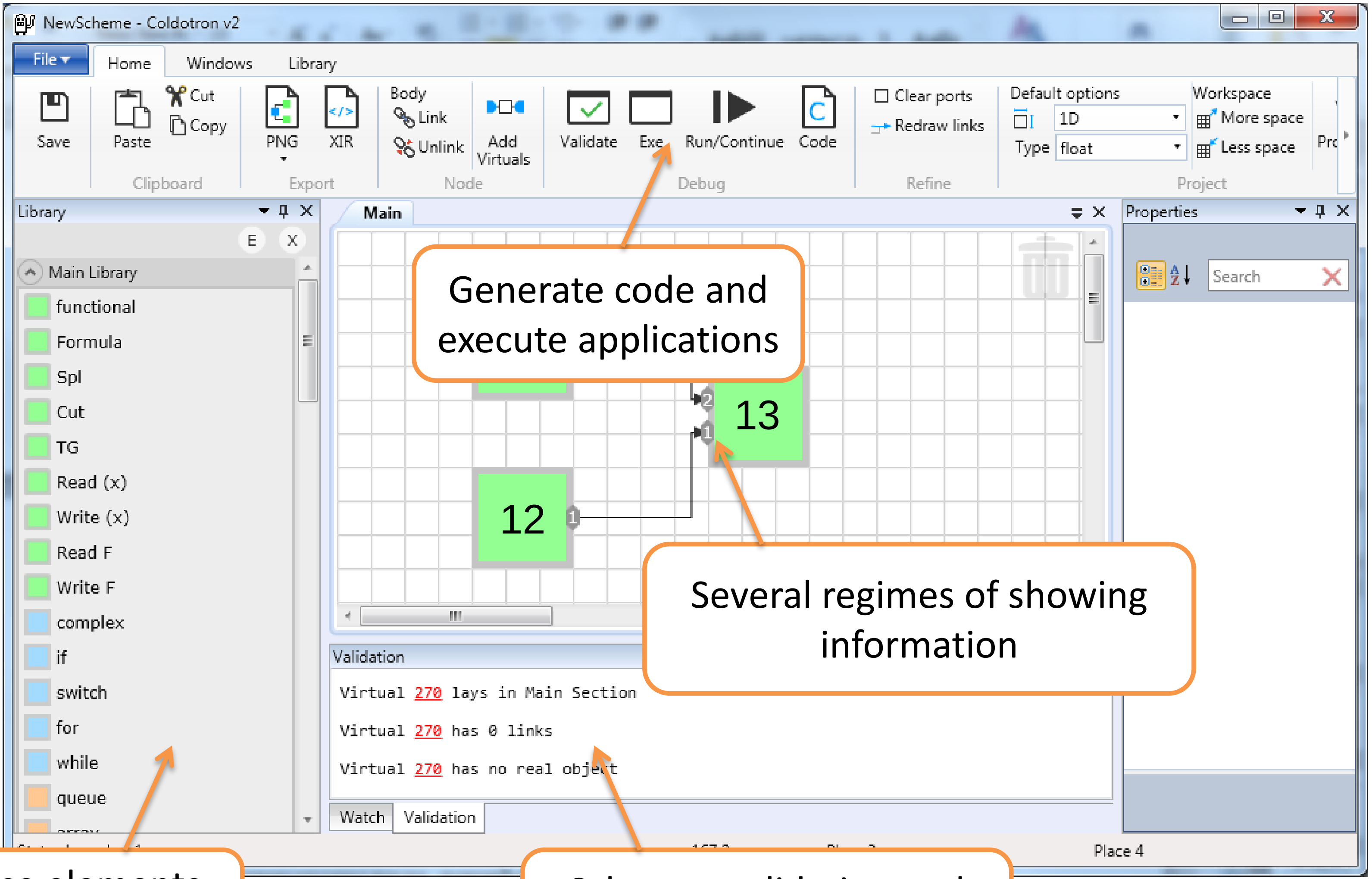


Subroutine text on target platform language



```
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



Validation

```
Virtual 270 lays in Main Section
Virtual 270 has 0 links
Virtual 270 has no real object
```

Generate code and execute applications

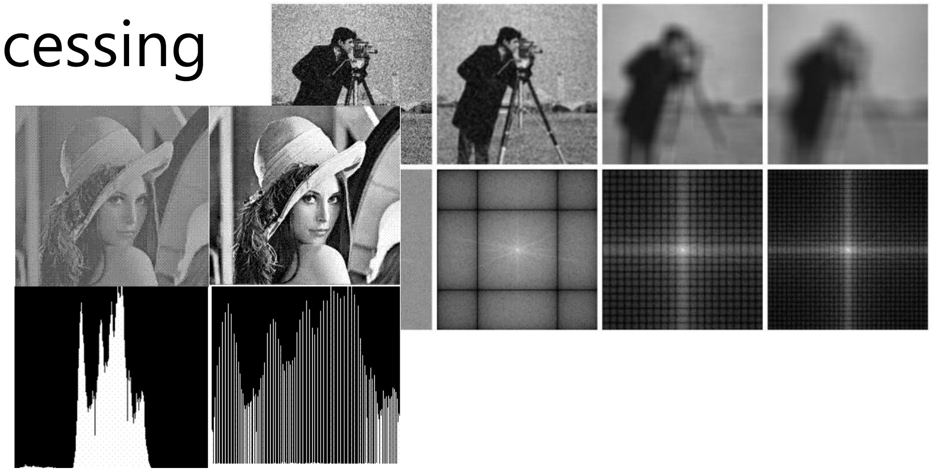
Several regimes of showing information

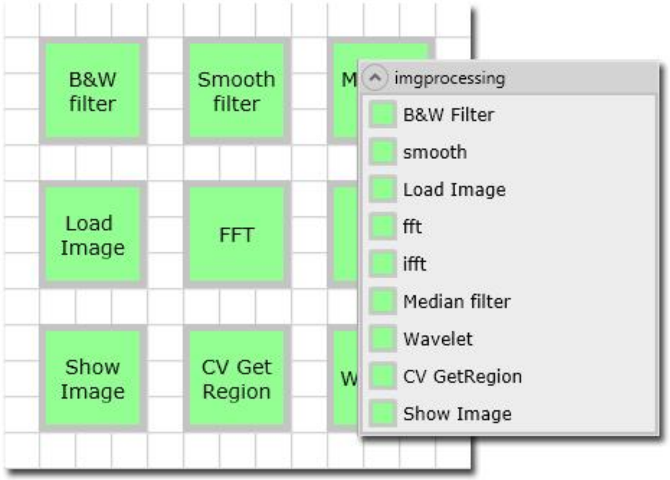
Base elements library

Schemes validation and debugging

Example: making a DSL for image processing

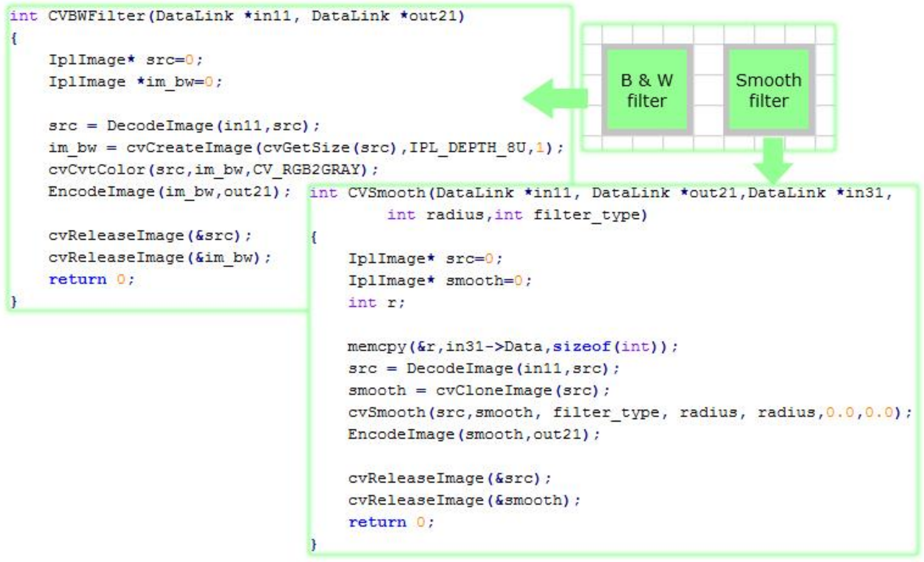
1. Analyze domain of image processing





2. Create a library of new language functional elements

3. Define C++ & OpenCV functional for new elements



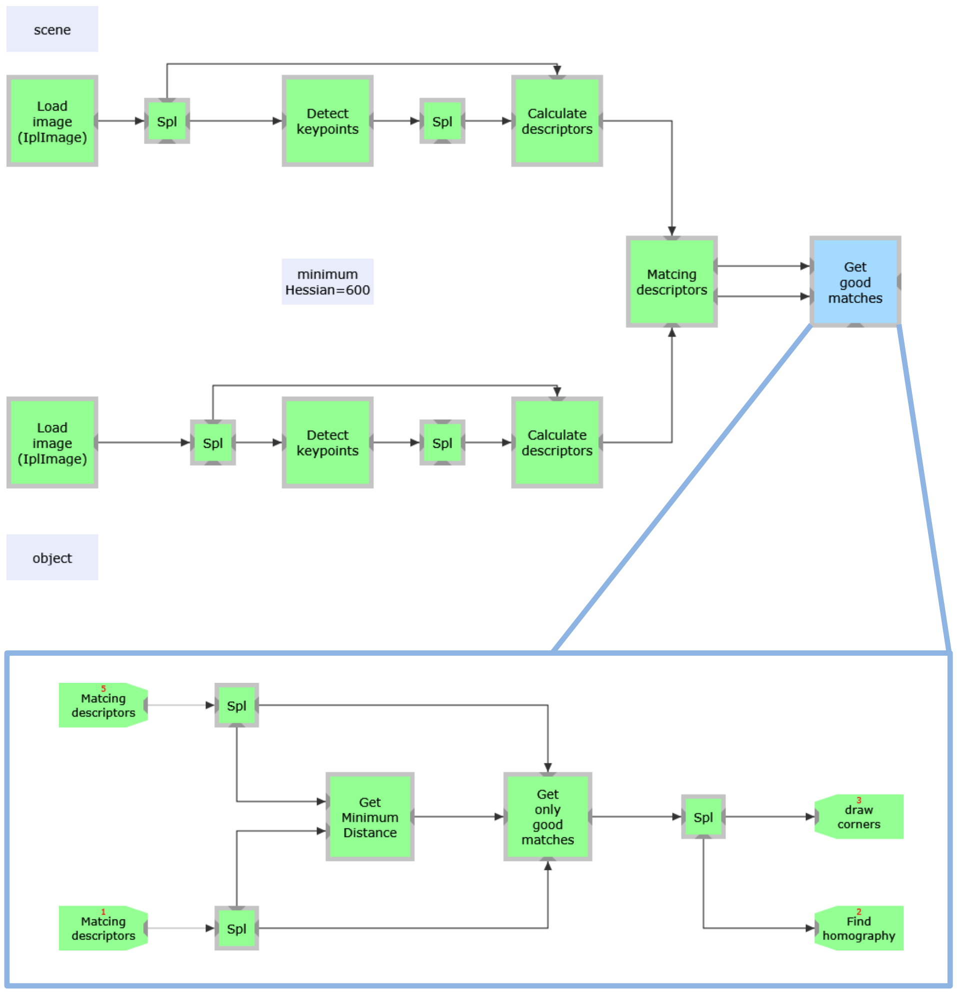
```
int CVBWFilter(DataLink *in11, DataLink *out21)
{
    IplImage* src=0;
    IplImage *im_bw=0;

    src = DecodeImage(in11,src);
    im_bw = cvCreateImage(cvGetSize(src),IPL_DEPTH_8U,1);
    cvCvtColor(src,im_bw,CV_RGB2GRAY);
    EncodeImage(im_bw,out21);

    cvReleaseImage(&src);
    cvReleaseImage(&im_bw);
    return 0;
}
```

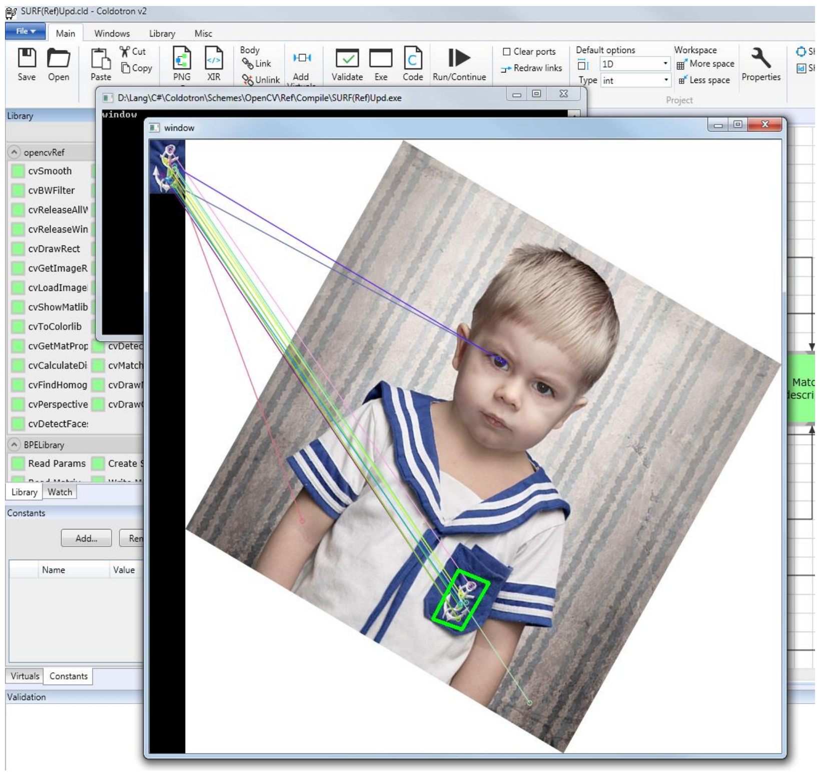
```
int CVSmooth(DataLink *in11, DataLink *out21,DataLink *in31,
        int radius,int filter_type)
{
    IplImage* src=0;
    IplImage* smooth=0;
    int r;

    memcpy(&r,in31->Data,sizeof(int));
    src = DecodeImage(in11,src);
    smooth = cvCloneImage(src);
    cvSmooth(src,smooth, filter_type, radius, radius,0.0,0.0);
    EncodeImage(smooth,out21);

    cvReleaseImage(&src);
    cvReleaseImage(&smooth);
    return 0;
}
```

Example: using the DSL for image processing

Image recognition (OpenCV)

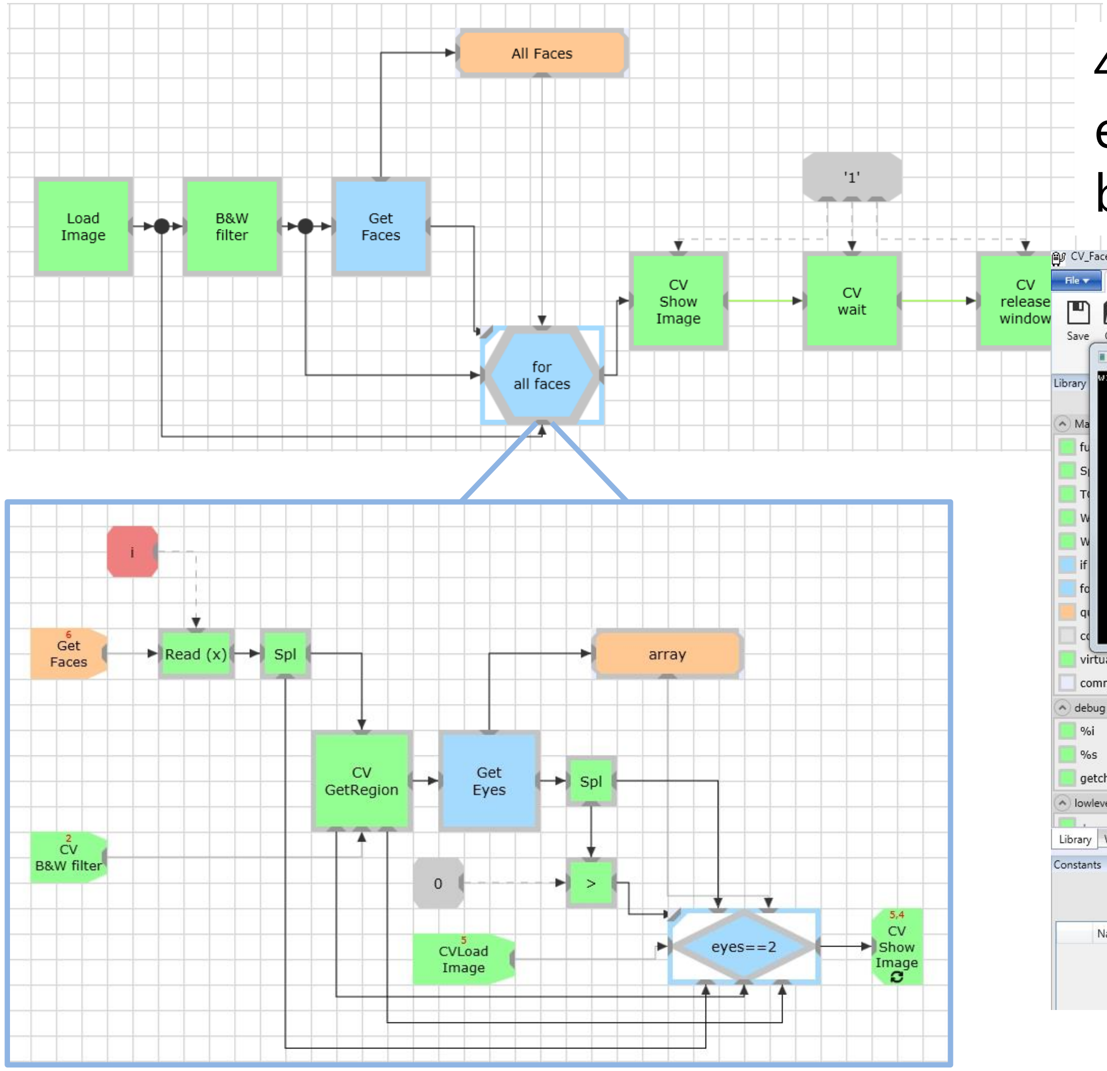
4. Build a scheme from base elements and new functional blocks



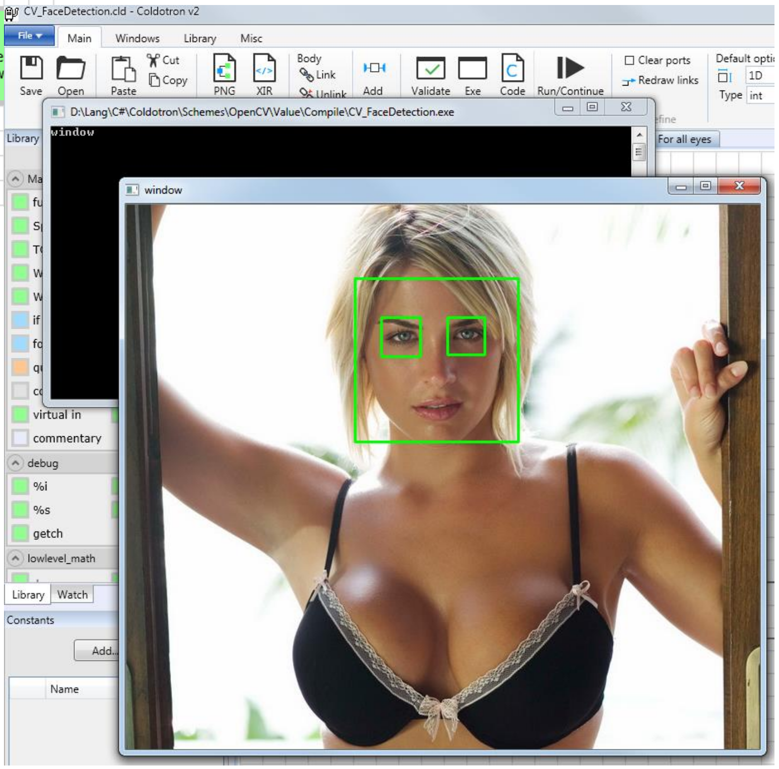


Example: using the DSL for image processing

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