



16th FRUCT Conference

Technology and Design Tools for Portable Software Development for Embedded Systems

Boris Sedov, Alexey Syschikov, Vera Ivanova

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

Why do we need such technology?

1. “Two-in-one” developer is required:



skilled domain experts +



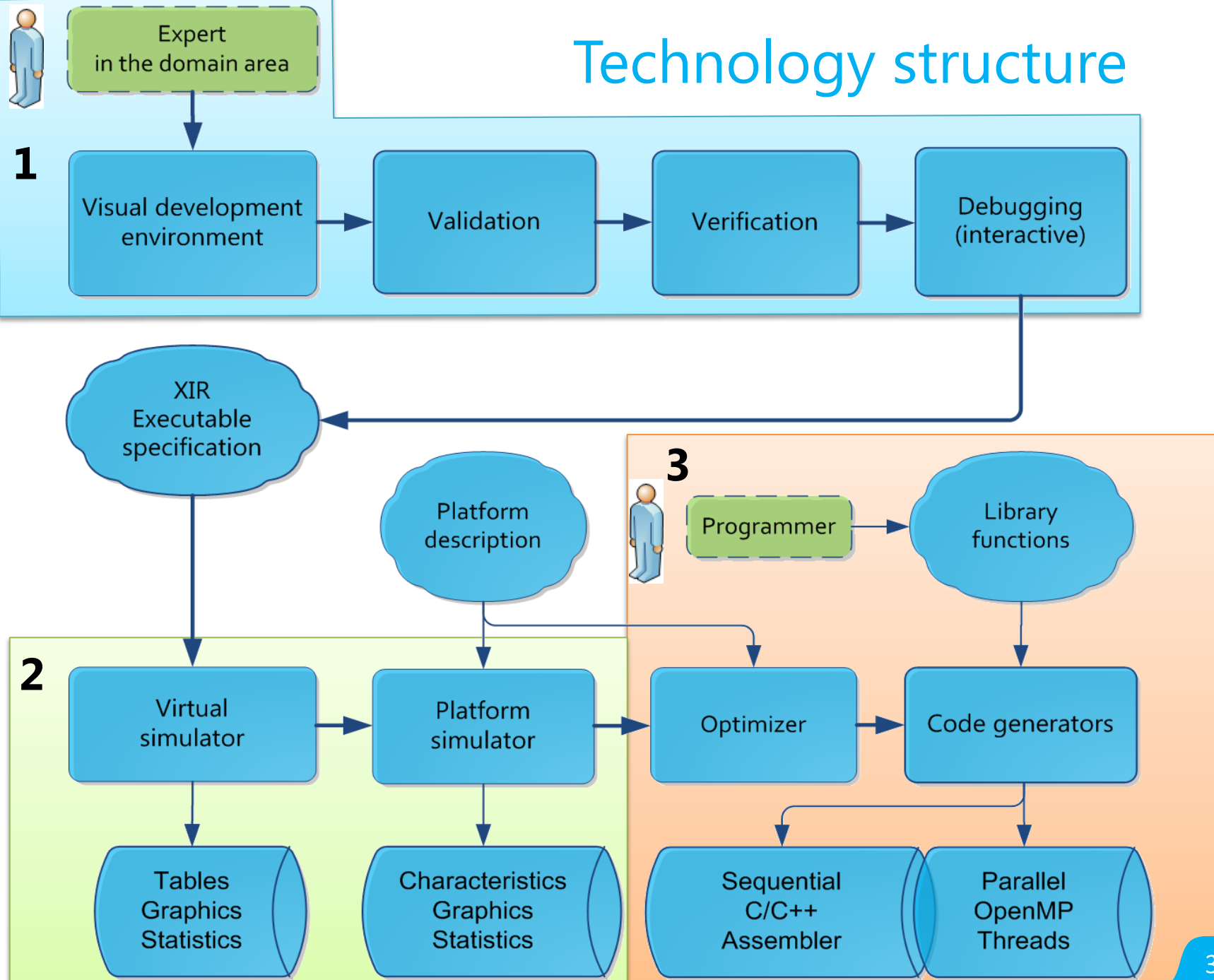
skilled programmer

2. Contradictive requirements to hardware platforms

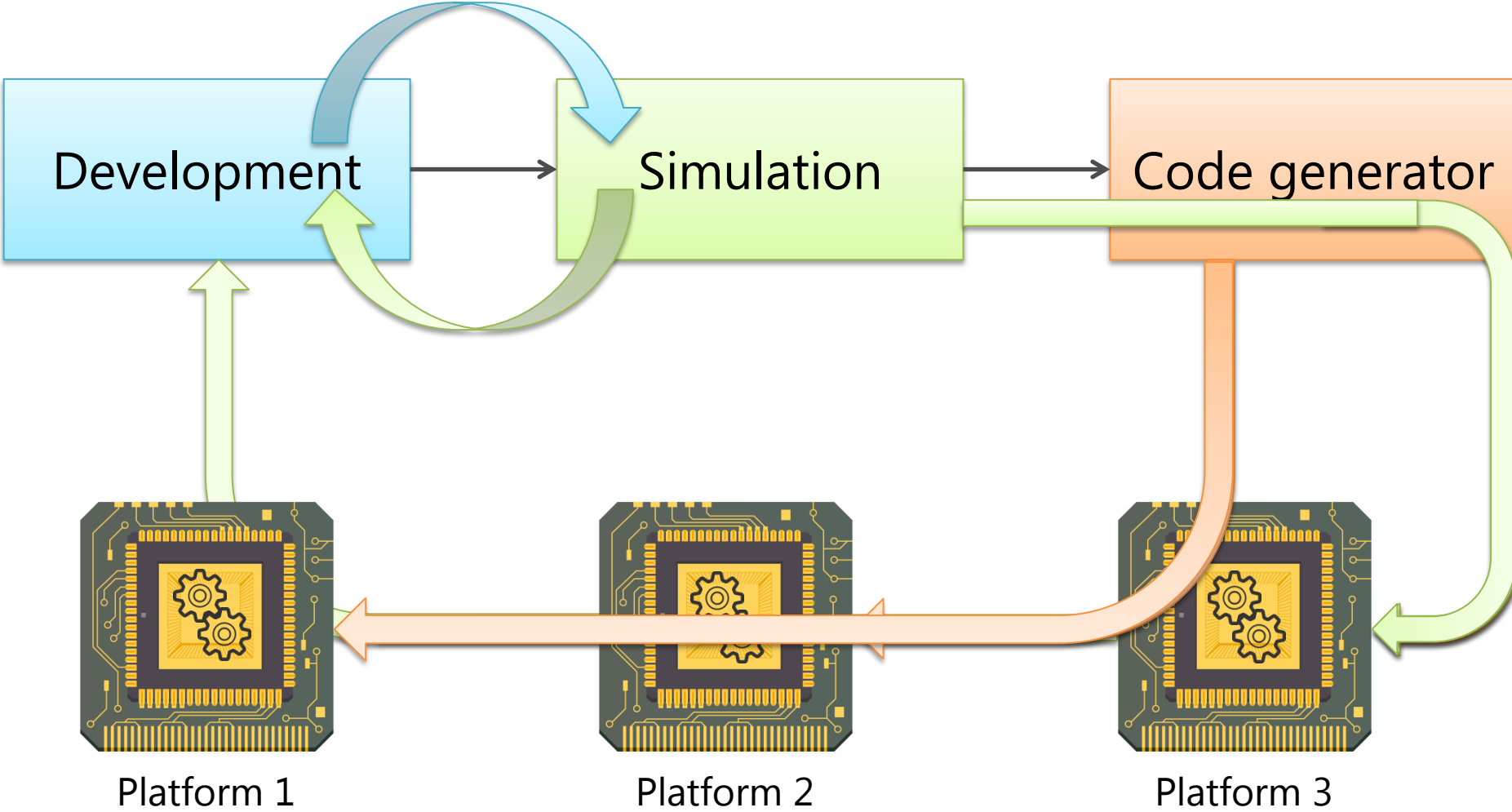
3. The development of an algorithm and program should be started before the selection of a specific platform configuration

4. Hardware platforms become out-of-date rapidly, but computation tasks that are executed on them changes rarely

Technology structure



The life cycle of programs for embedded systems





Expert
in the domain area

Visual Development Environment

Visual development
environment

Validation

Verification

Debugging
(interactive)

XIR
Executable
specification

Platform
description



Programmer

Library
functions

Virtual
simulator

Platform
simulator

Optimizer

Code generators

Tables
Graphics
Statistics

Characteristics
Graphics
Statistics

Sequential
C/C++
Assembler

Parallel
OpenMP
Threads

Visual Development Environment (VIPE)

The screenshot displays the SURF.cld - Coldotron v2 Visual Development Environment (VIPE) interface. The main workspace shows a flowchart for image processing and feature matching. The flowchart consists of the following nodes and connections:

- scene** (light blue box) connects to **Load image** (green box).
- Load image** connects to **Spl** (green box).
- Spl** connects to **Detect keypoints** (green box).
- Detect keypoints** connects to **Spl** (green box).
- Spl** connects to **Calculate descriptors** (green box).
- Calculate descriptors** connects to **Matching descriptors** (green box).
- Matching descriptors** connects to **Get good matches** (blue box).
- Get good matches** connects to **object** (light blue box).

The flowchart is displayed on a grid. A dashed line with the value "600" is visible between the two "Detect keypoints" nodes. The interface includes a menu bar (File, Main, Windows, Library, Misc), a toolbar with various icons (Save, Open, Paste, Copy, PNG, XIR, Body Link, Unlink, Add Virtuals, Validate, Exe, Code, Run/Continue, Clear ports, Redraw links, Default options, Workspace, Properties, Show ports, Show info, Flip ports, To back, To front, Prev, Next, Level up, Real size), and several panels:

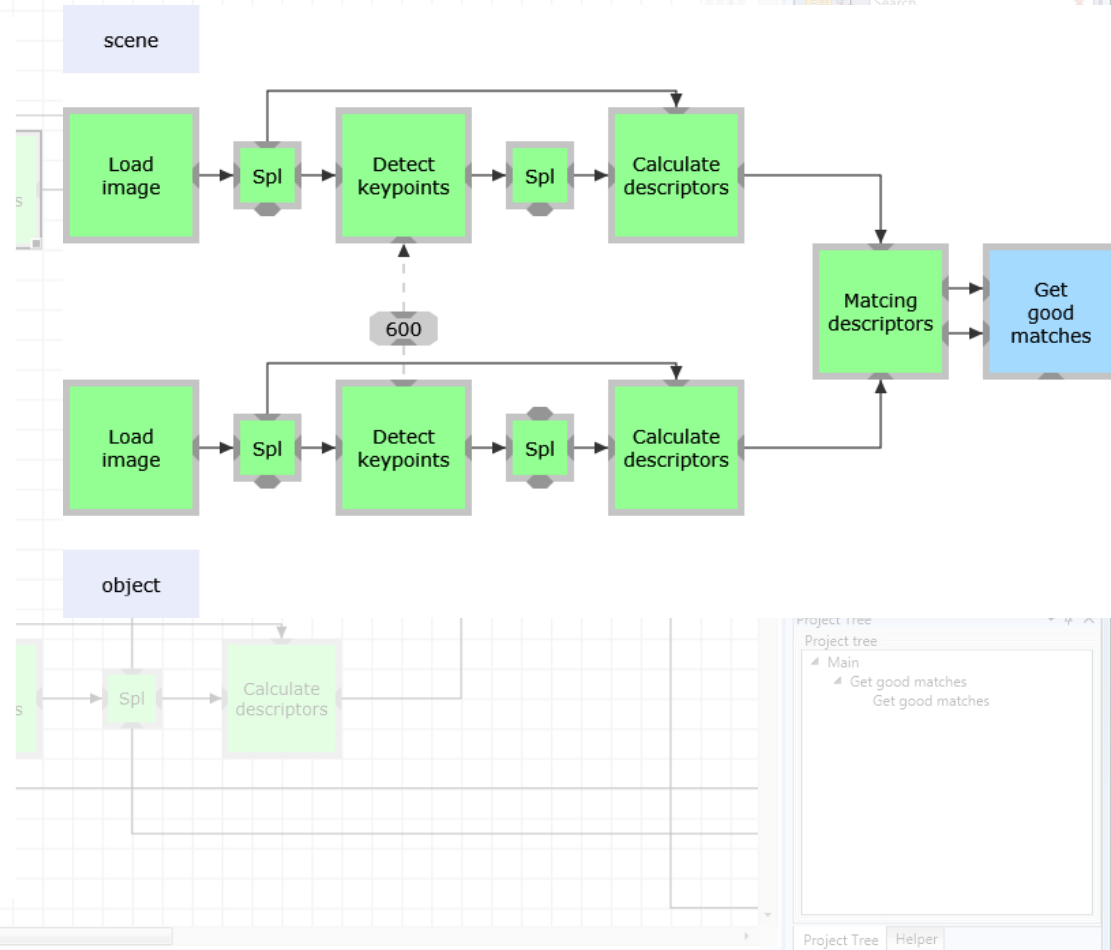
- Library:** Main Library (functional, Spl, TG, Write (x), Write F, if, for, queue, constant, virtual in, commentary), debug (%i, %c, %s, %f, getch), powerLib (Fractional Pov, Power, Int power, Log 2, max(bvte), max(double)).
- Properties:** NodeController (Search, Additional, Comment: Detect keypoints, Main: Execution cost: 0, Implement function: CVDetectKeypoi, Is permanent: [], Operation name, Misc: ID: 320, Subtype: functional, Type: terminal).
- Project Tree:** Project tree (Main: Get good matches, Get good matches).
- Constants:** Add..., Remove, Name, Value table.

Status bar: place1 703.98055289435 Place 3 Place 4

Visual Programming Language (VPL)

Cognitive advantages:

- clear view of development process
- traceability of the dependency graph
- calculations management structure
- natural parallelism
- potential pipelining



Asynchronous Growing Processes (AGP) formal computational model

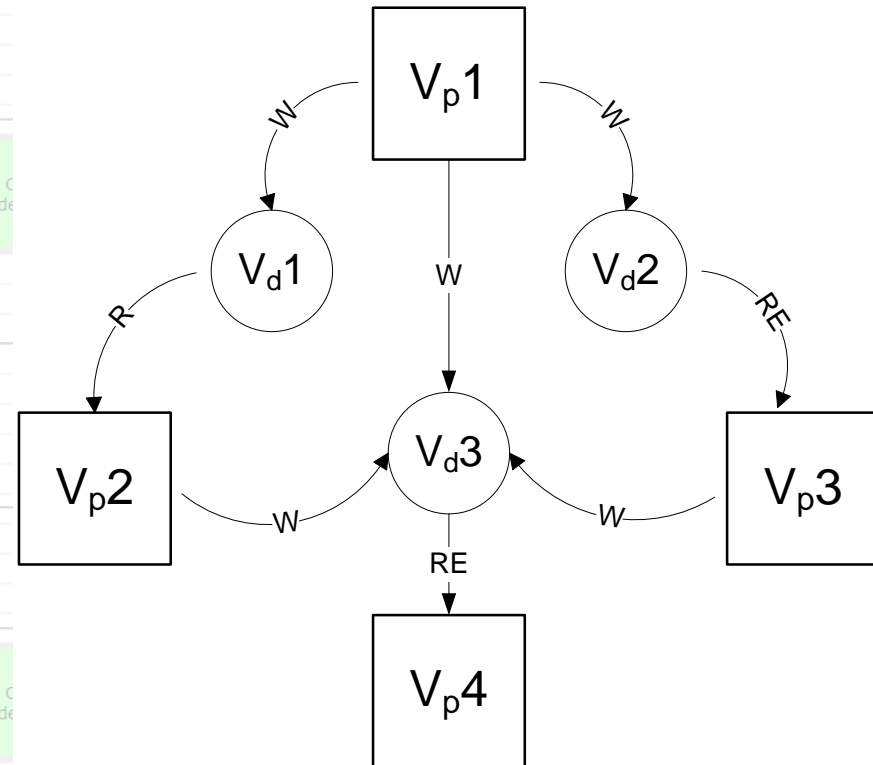
AGP defines:

- language syntax
- semantic of language objects
- control units

AGP provides:

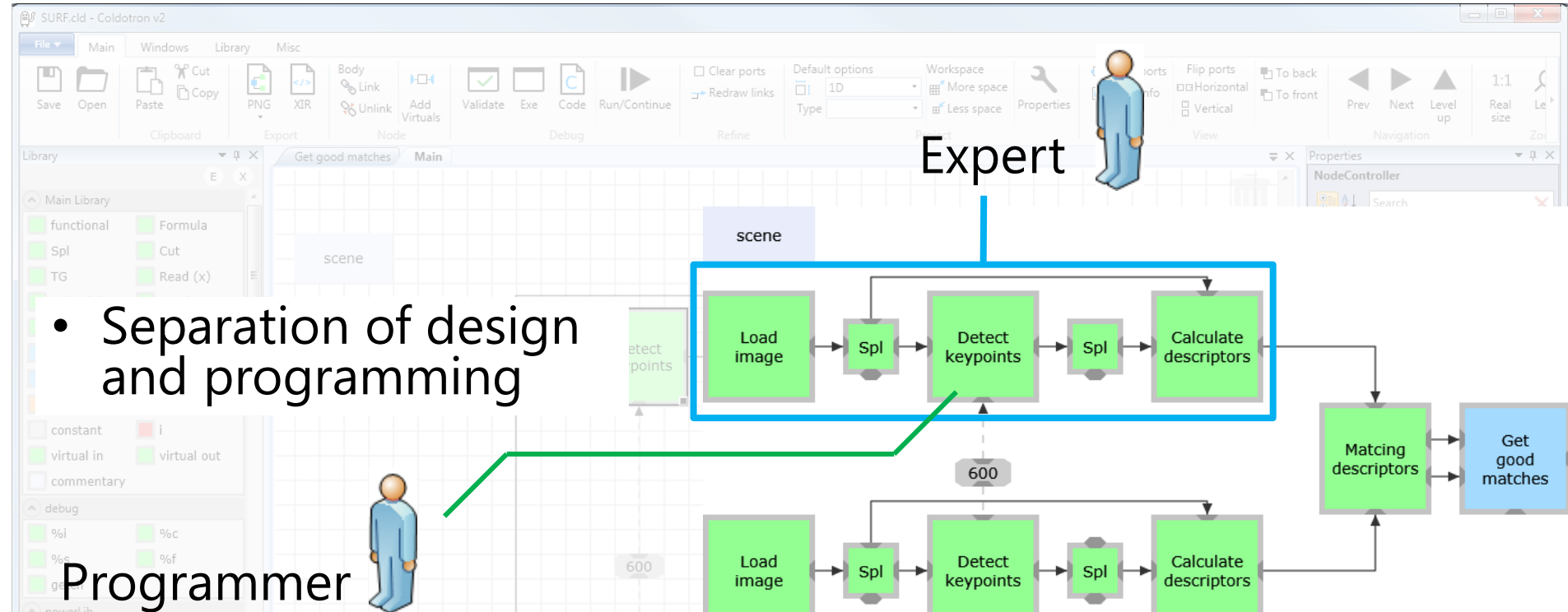
- formal verification
- debugging
- portability

Program scheme —
oriented graph



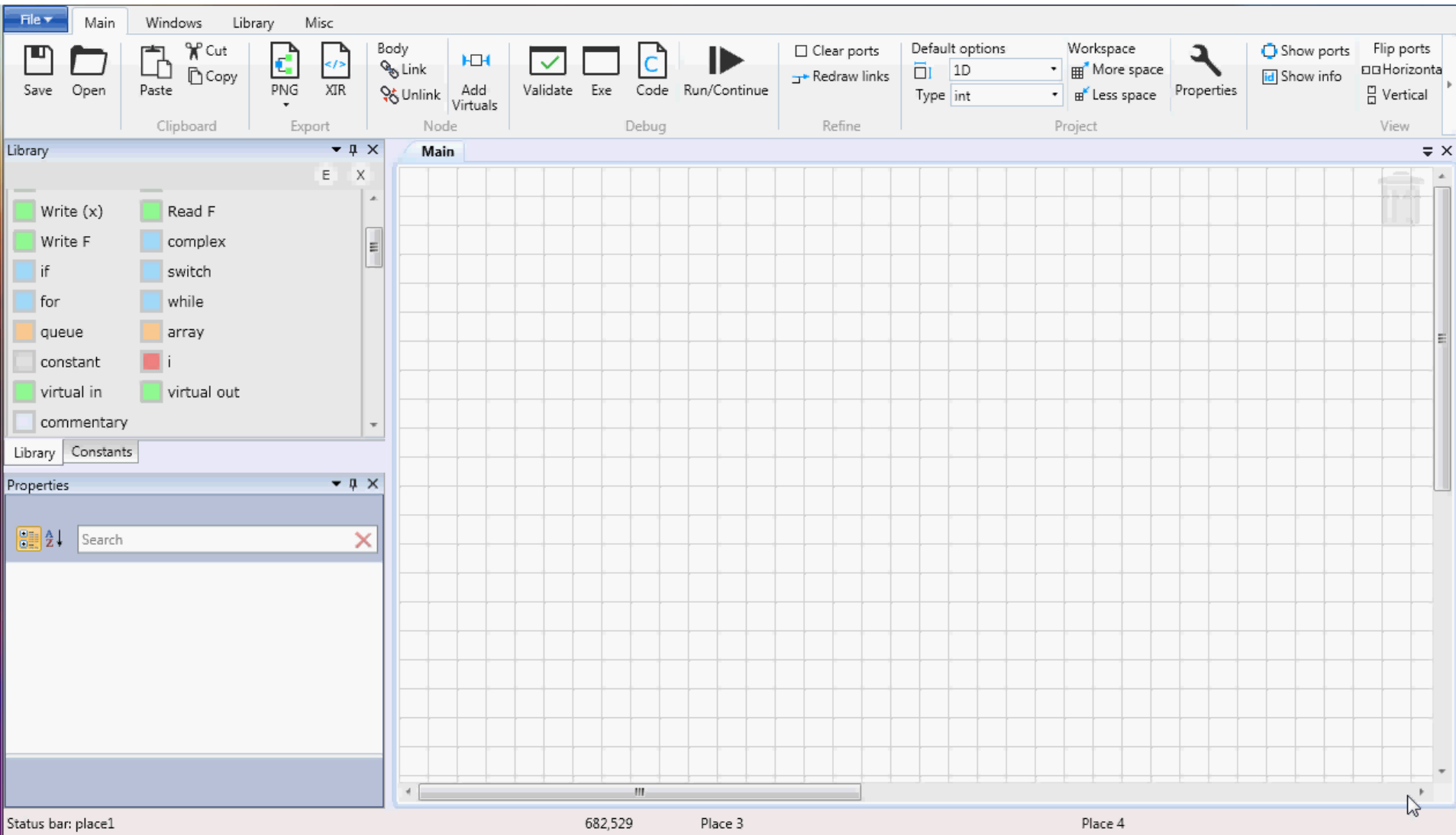
V_p – operator vertex,
 V_d – data-object vertex,
 $W/R/RE$ – arcs (links) marking

Visual Programming Language (VPL)



- Easy program development
- Easy changes in program structure
- No direct programmer influence on a program scheme
- Local appearance of code errors
- Possibility of auto-documenting of the program scheme
- Effective program maintenance during the whole lifecycle
- Decreasing of errors possibility without sacrificing program obviousness
- Flexibility and ease-of-change on any design stage

Visual Programming Language (VPL)



Domain specific programming

The screenshot shows the SURF.cld IDE interface. On the left, a 'Library' panel lists various DSL components under categories like 'powerLib', 'lowlevel_math', 'lowlevel_bitwise', 'opencvRef', and 'opencv'. The main workspace displays a flow diagram with nodes: 'Spl', 'Detect keypoints', 'Spl', and 'Calculate descriptors'. The 'Detect keypoints' node is highlighted in green. The 'Properties' panel on the right shows the configuration for a 'NodeController' with a search field and sections for 'Additional' and 'Main'. The 'Project Tree' panel at the bottom right shows a tree structure with 'Main' and sub-items 'Get good matches'.

- DSL libraries provide convenience of design within an application domain
- Easy re-use of development results



Expert
in the domain area

Interactive tools

Visual development
environment

Validation

Verification

Debugging
(interactive)

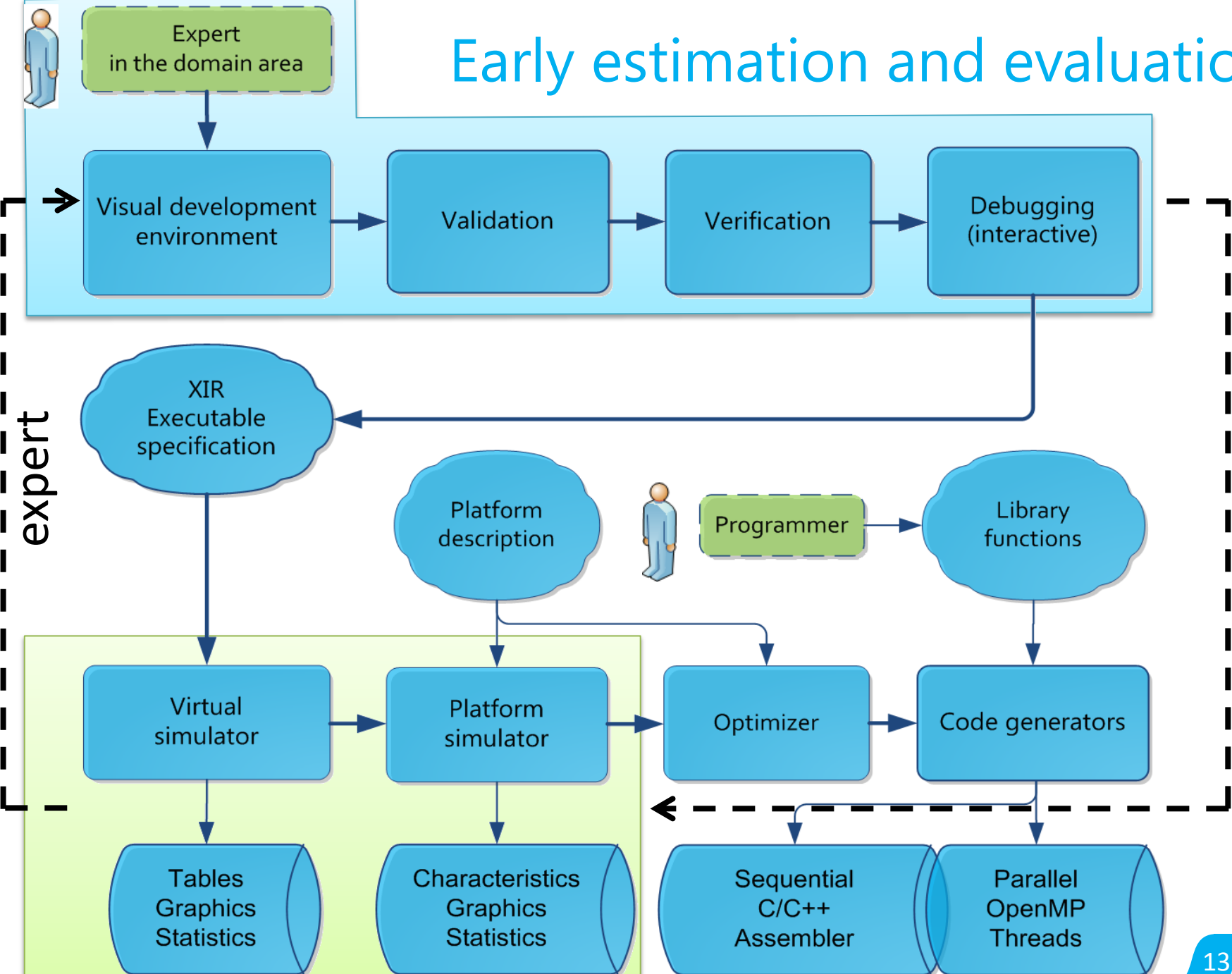
Development process support tools:

- scheme validation
- verification
- interactive debugging etc.

The screenshot shows the SURF.cld - Coldotron v2 development environment. On the left is a library of components including functional, Formula, Spl, Cut, TG, Read (x), Write (x), Read F, Write F, complex, if, for, que, cor, vir, cor, debu, %l, %s, get, pow, Fra, Int, ma. The main workspace displays a visual programming flowchart with components like 'Mod', 'Spl', and 'I=' connected by arrows. A 'cond' component is highlighted with a red box. Below the flowchart, a 'Validation' status bar shows 'Virtual 182 has 0 links'. On the right, a code editor window shows a 'my for' loop with the code 'i=9; i >= 0; i=i-1' and a 'my for' component with value 321. Below the code editor is a 'Watch' window with the following table:

Name	Values
Link 425	[X]
Link 3651	[X]
Link 3652	[10]

Early estimation and evaluation

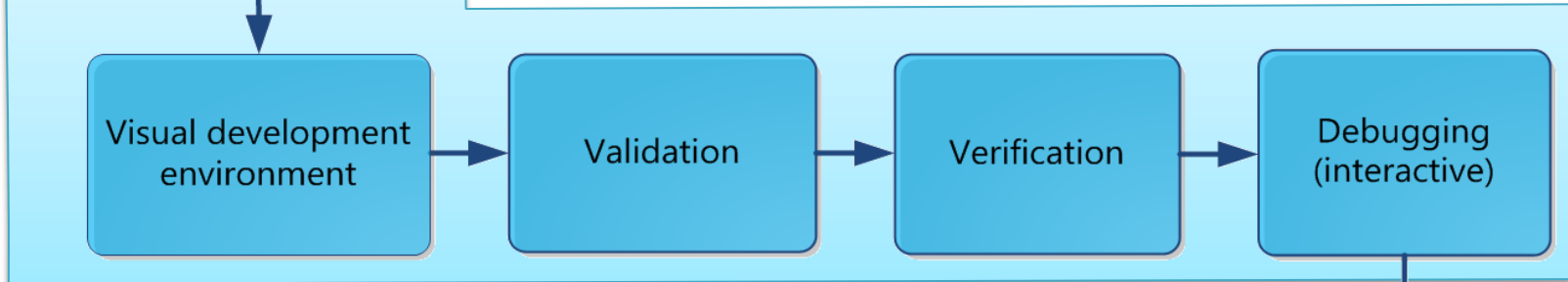


expert



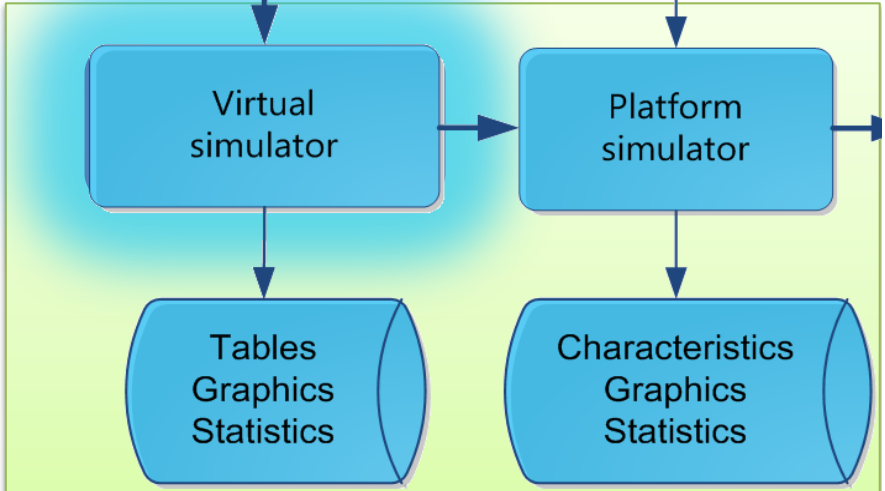
Expert in the domain area

Virtual simulator



XIR Executable specification

Platform description



Allows estimate:

- maximal possible parallelism
- computation space (amount of computations, used memory etc.)
- computation time
- amount and intensity of data exchange



Expert
in the domain area

Visual development
environment

Validation

Platform simulator

Allows estimate:

- requirements to embedded system cores performance
- requirements to embedded system cores memory
- computation cores occupation for the different allocation variants and occupation balance
- amount and intensity of data exchange
- effectiveness of hardware loading
- bottlenecks of hardware platform, program and tasks allocation

XIR
Executable
specification

Platform
description

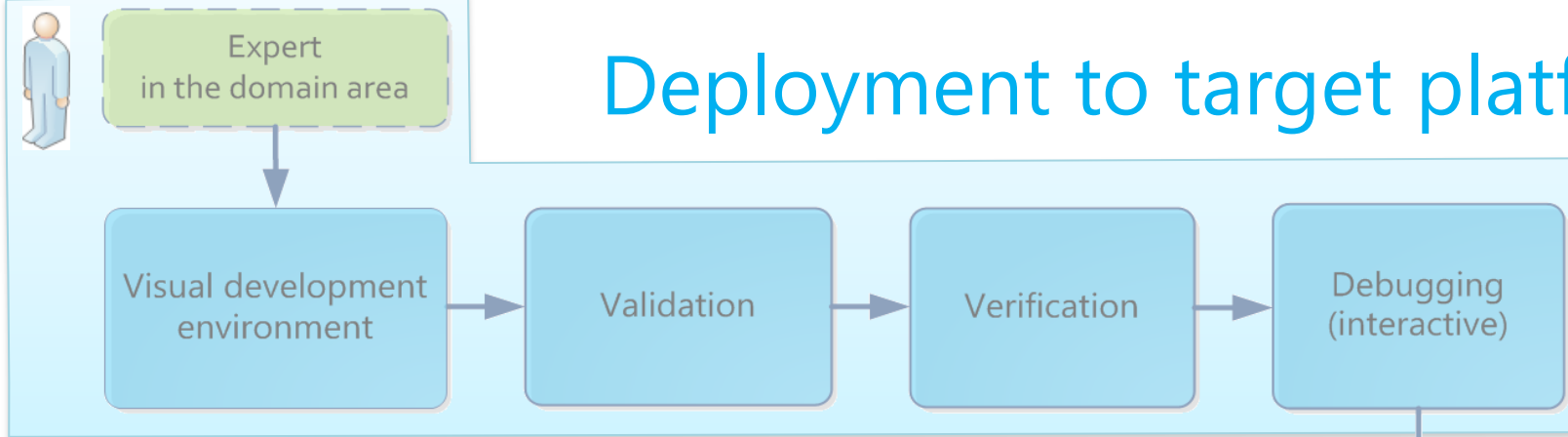
Virtual
simulator

Platform
simulator

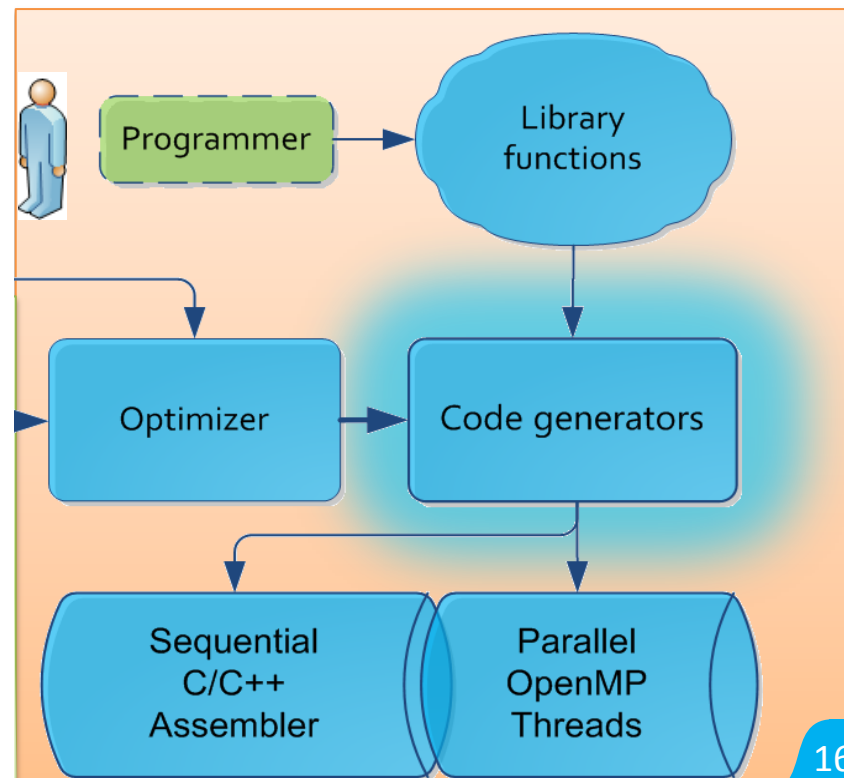
Tables
Graphics
Statistics

Characteristics
Graphics
Statistics

Deployment to target platforms



- ANSI C
- C++
- Parallel OpenMP
- Parallel threads
- MPI
- Assembler MIPS, DSP
- Inhomogeneous systems





Expert
in the domain area

Conclusions

Visual development
environment

Validation

Verification

Debugging
(interactive)

- Supporting of a whole embedded software lifecycle
- The formal model based development ensures the equal execution of a debugged program in any runtime environment
- An algorithm is created and debugged once Parallel threads
- Scheme optimization for platform granularity
- Supporting of heterogeneous embedded systems (hardware blocks, accelerators, DSP)

Tables
Graphics
Statistics

Characteristics
Graphics
Statistics

Sequential
C/C++
Assembler

Parallel
OpenMP
Threads



Thank you!