XII международная конференция
CEE-SECR / РАЗРАБОТКА ПО
28 - 29 октября, Москва

ГУАП

Технология предметно-ориентированного программирования для неоднородных многоядерных платформ

Борис Седов, Юрий Шейнин, Алексей Сыщиков

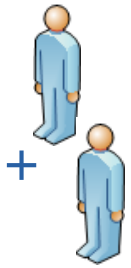
boris.sedov@guap.ru, sheynin@aanet.ru

Санкт-Петербургский государственный
университет аэрокосмического приборостроения

ГУАП

Зачем нужна такая технология?

1. Требуется разработчик “два в одном”:



квалифицированный специалист в предметной области
+ квалифицированный программист

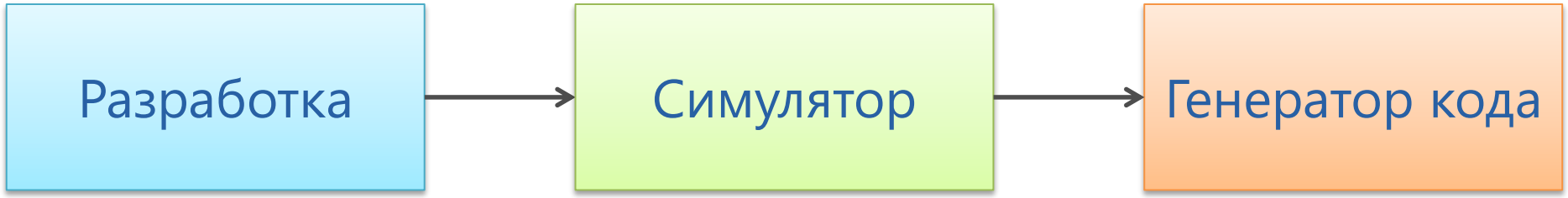
2. Сложность разработки параллельного встраиваемого ПО

3. Неоднородность встраиваемых систем

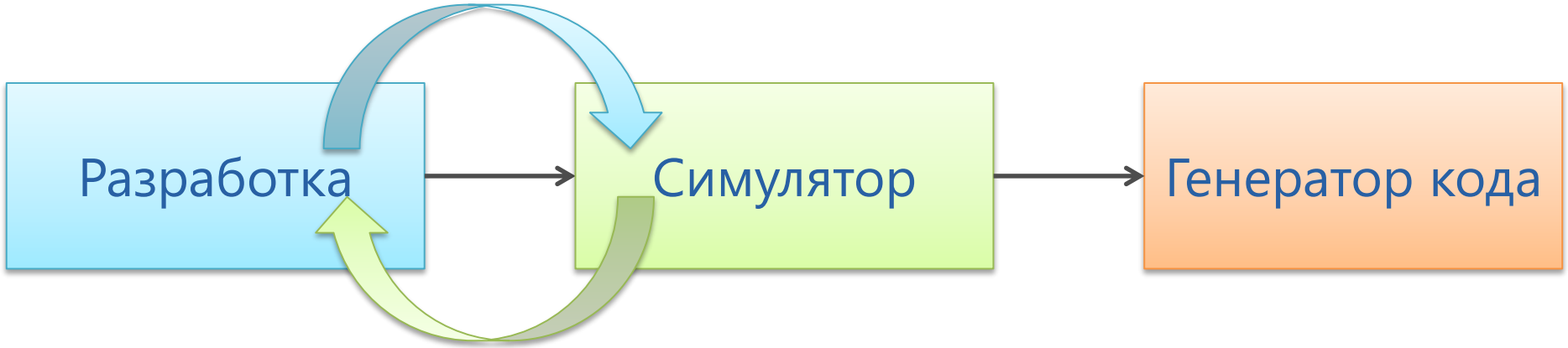
4. Аппаратные платформы быстро устаревают, но решаемые на них задачи и алгоритмы устаревают редко

5. Для достижения требований необходима адаптация алгоритмов под платформы и платформ под алгоритмы

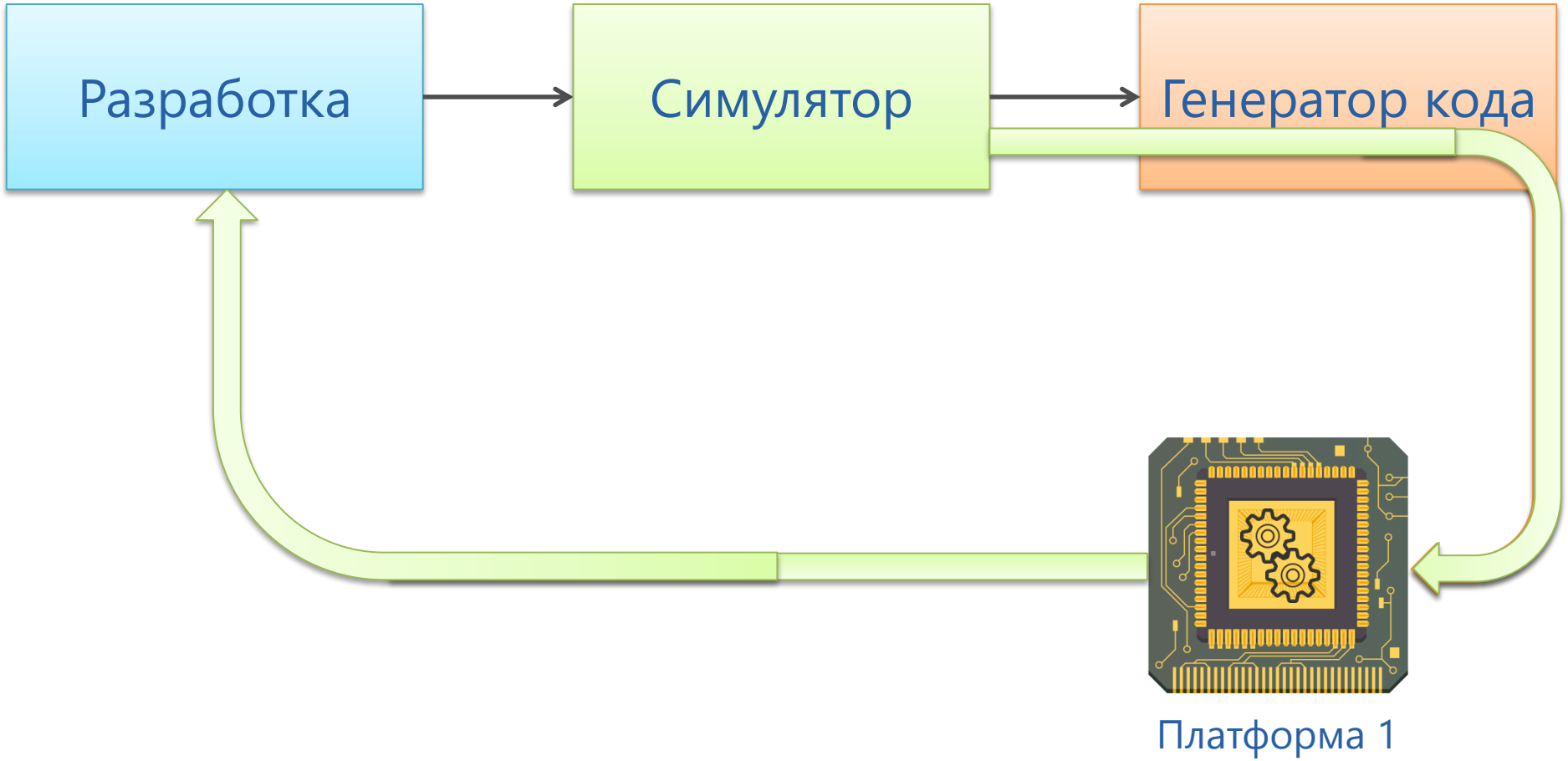
Жизненный цикл программы в рамках технологии



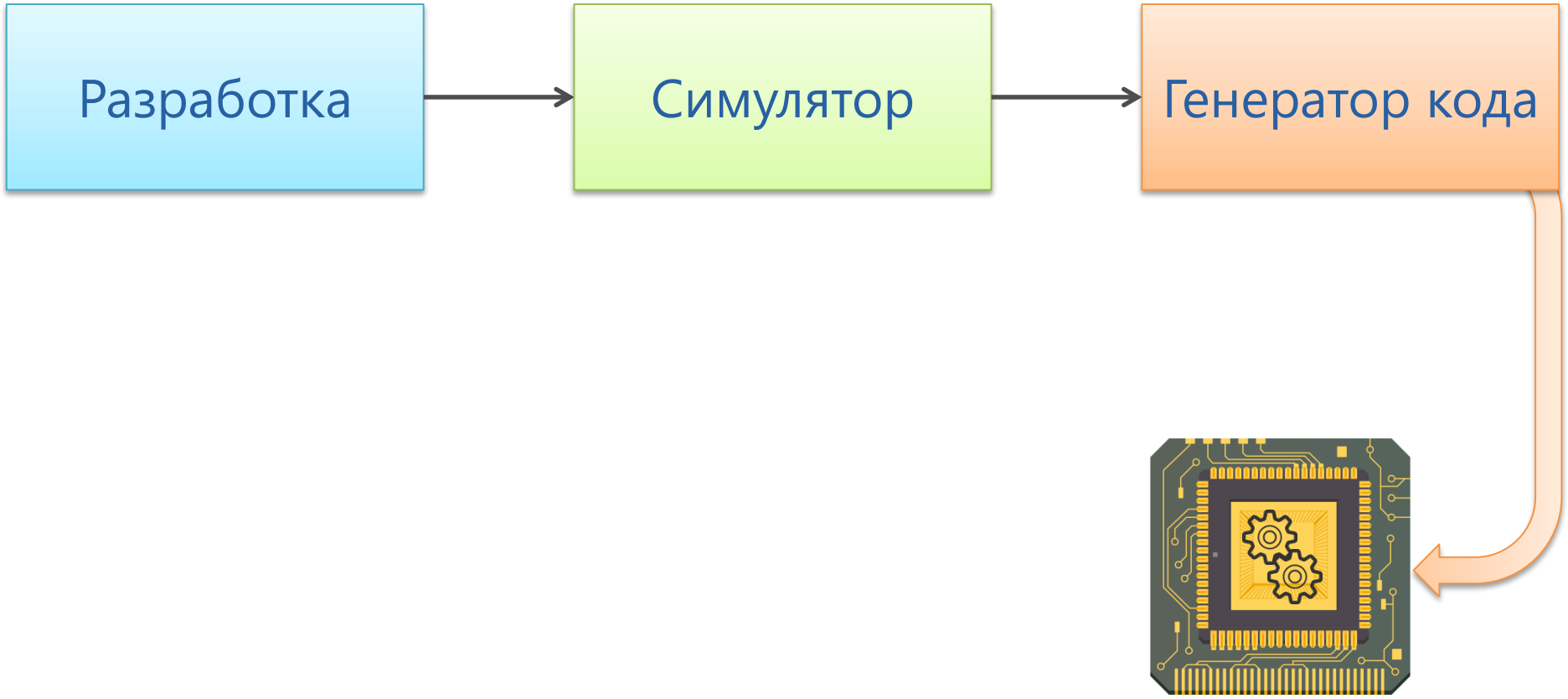
Жизненный цикл программы в рамках технологии



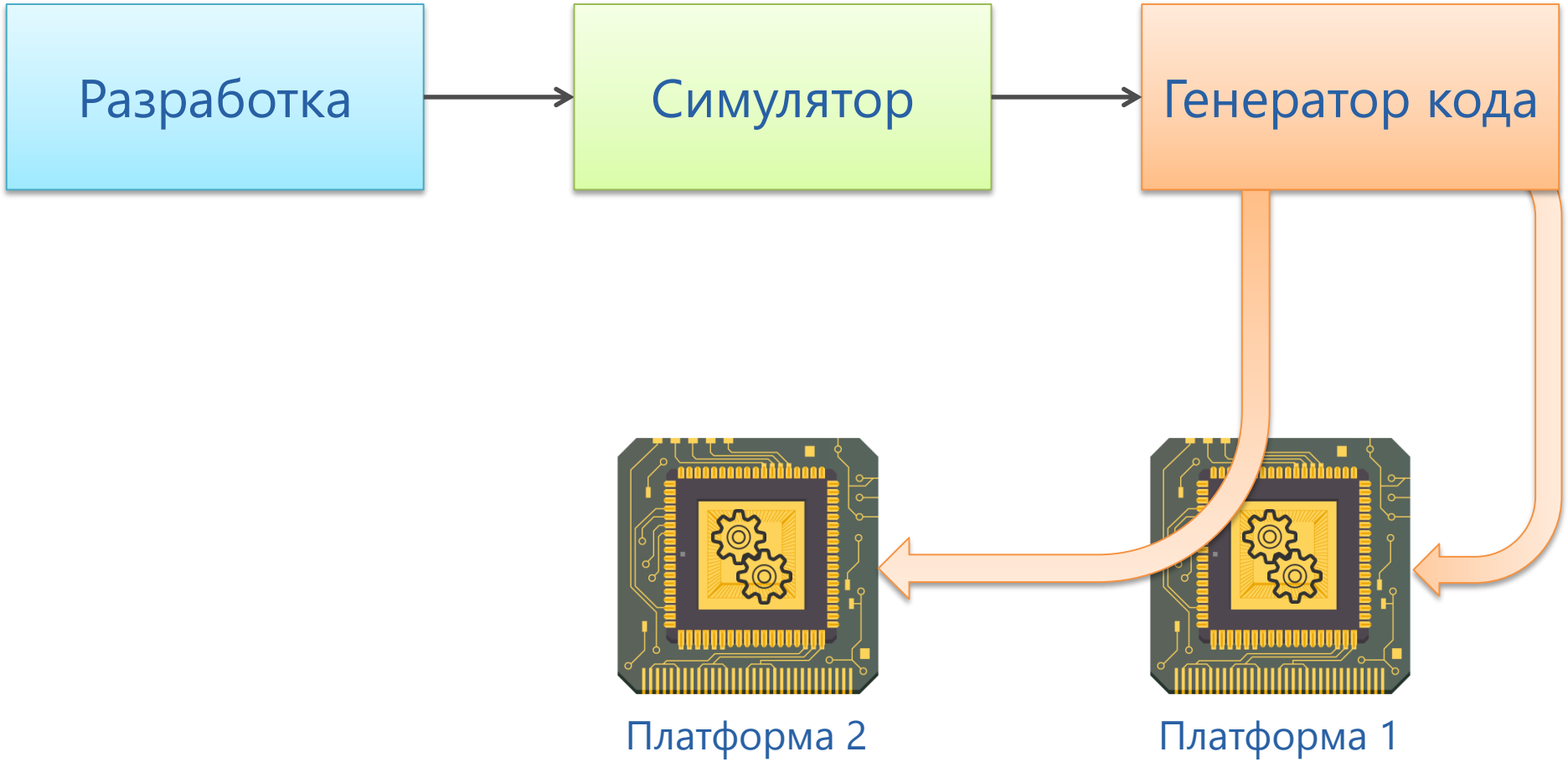
Жизненный цикл программы в рамках технологии



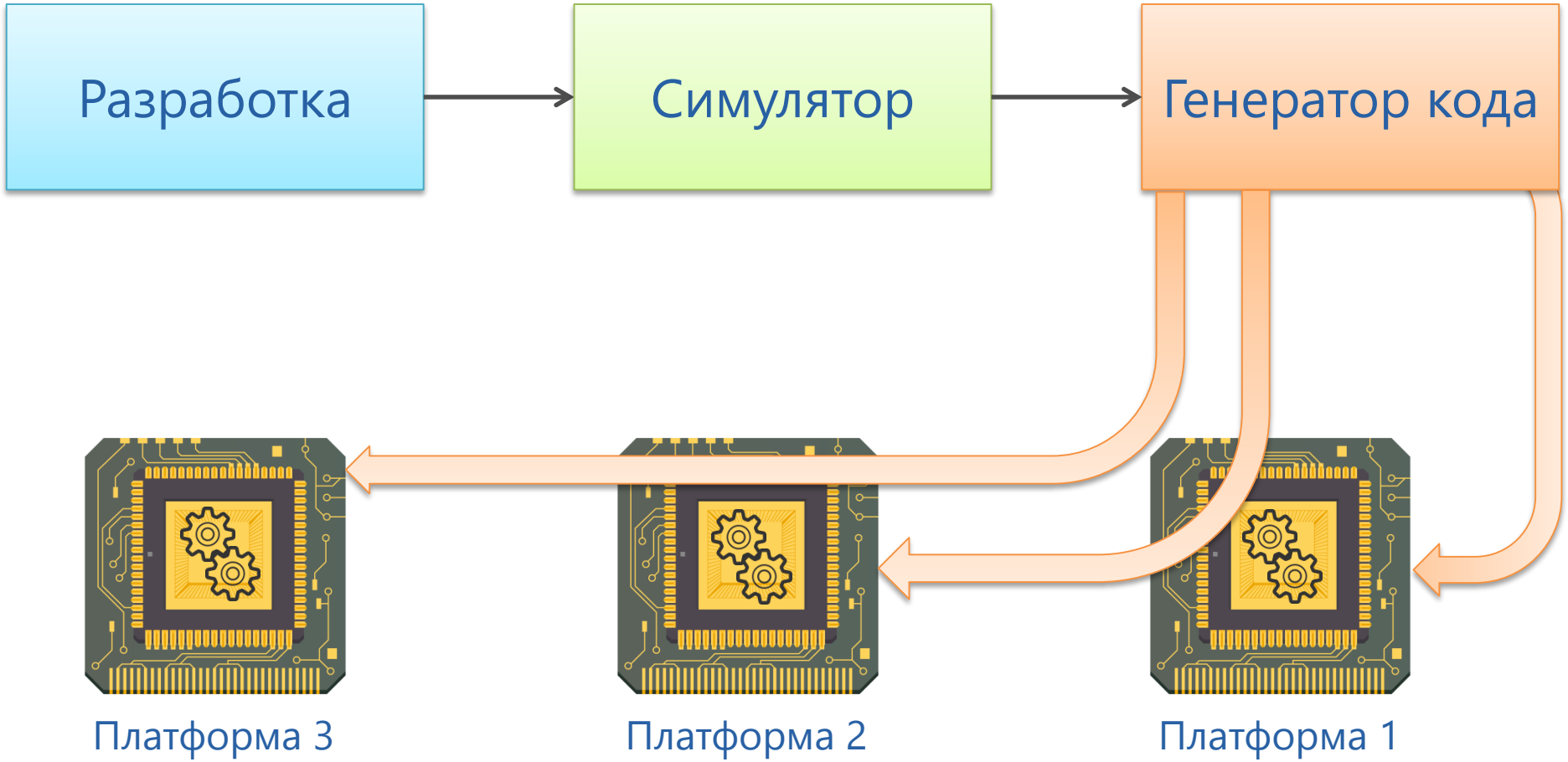
Жизненный цикл программы в рамках технологии



Жизненный цикл программы в рамках технологии



Жизненный цикл программы в рамках технологии



Детальная схема технологии

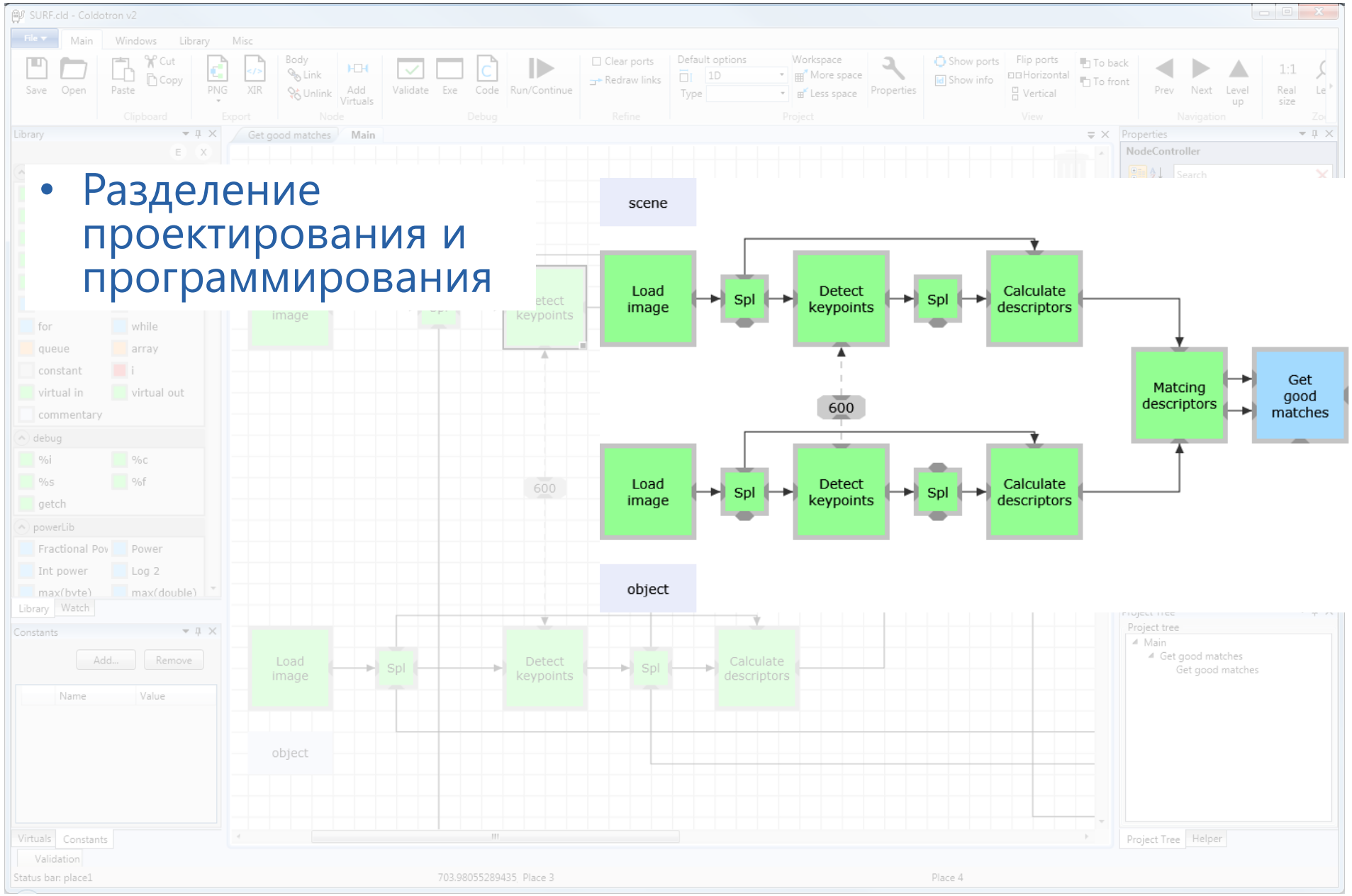
Детальная схема технологии

Детальная схема технологии

Детальная схема технологии

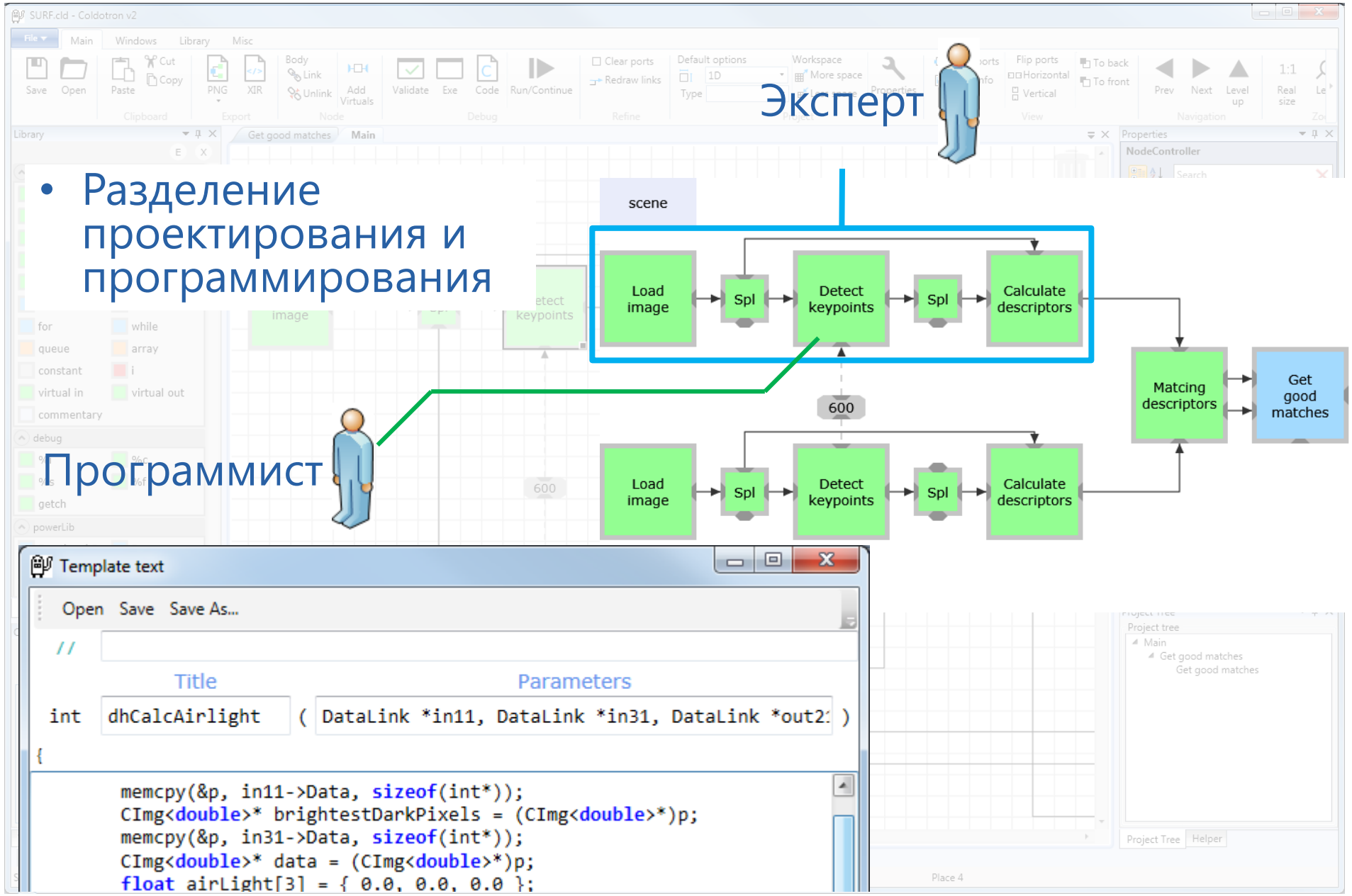
Графический язык программирования VPL

- Разделение проектирования и программирования

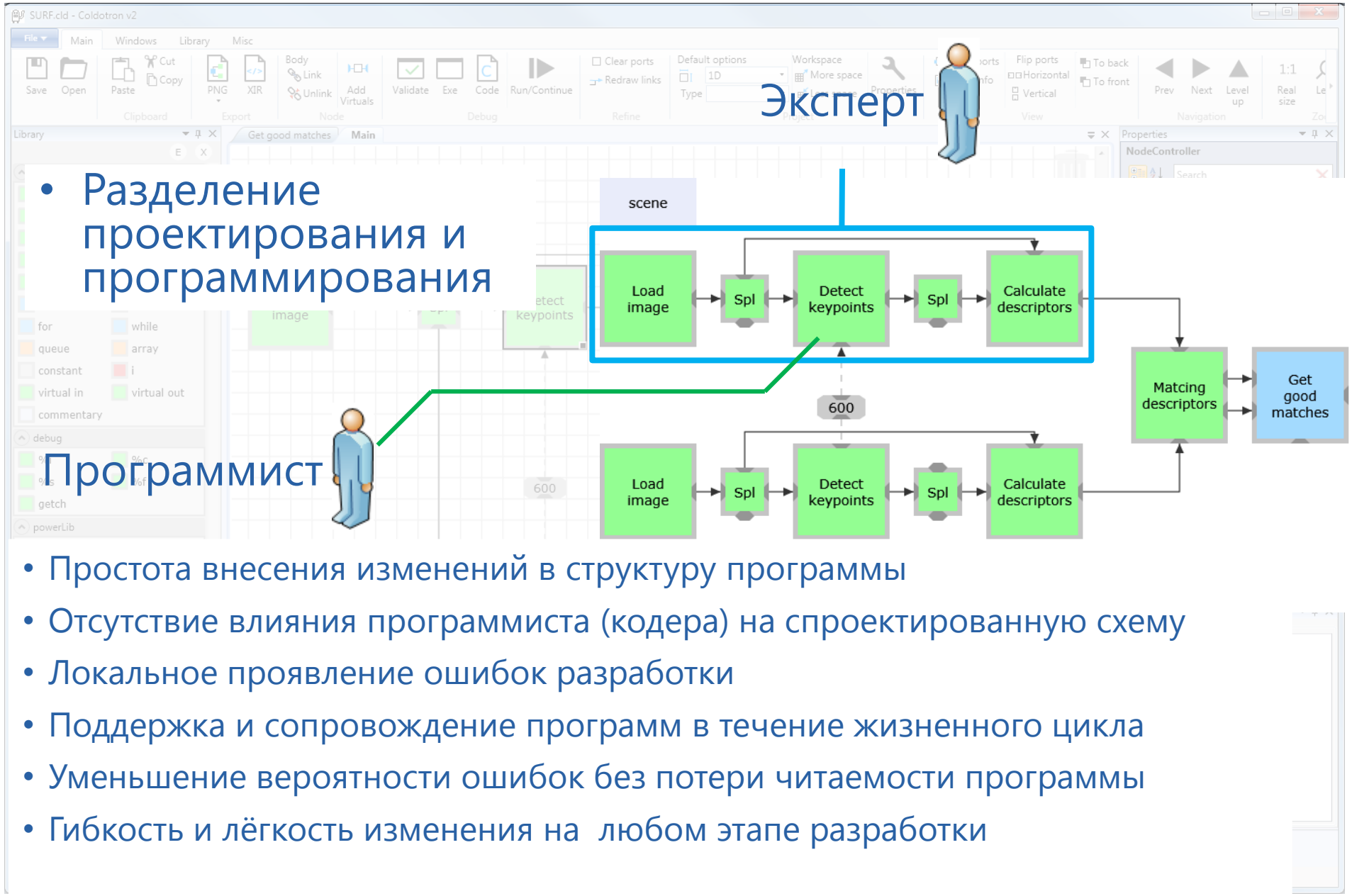


Графический язык программирования VPL

- Разделение проектирования и программирования



Графический язык программирования VPL

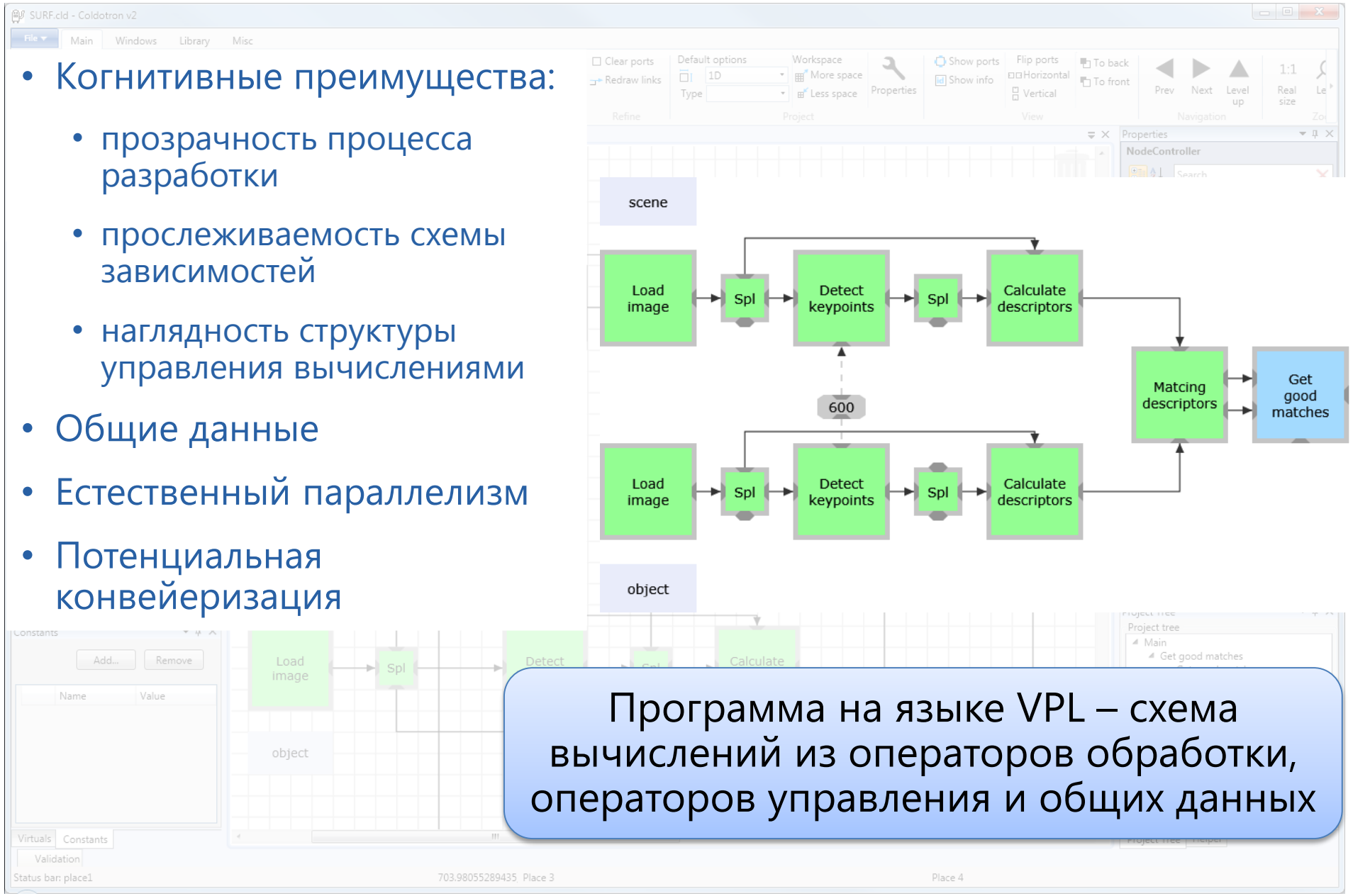


- Разделение проектирования и программирования

- Простота внесения изменений в структуру программы
- Отсутствие влияния программиста (кодера) на спроектированную схему
- Локальное проявление ошибок разработки
- Поддержка и сопровождение программ в течение жизненного цикла
- Уменьшение вероятности ошибок без потери читаемости программы
- Гибкость и лёгкость изменения на любом этапе разработки

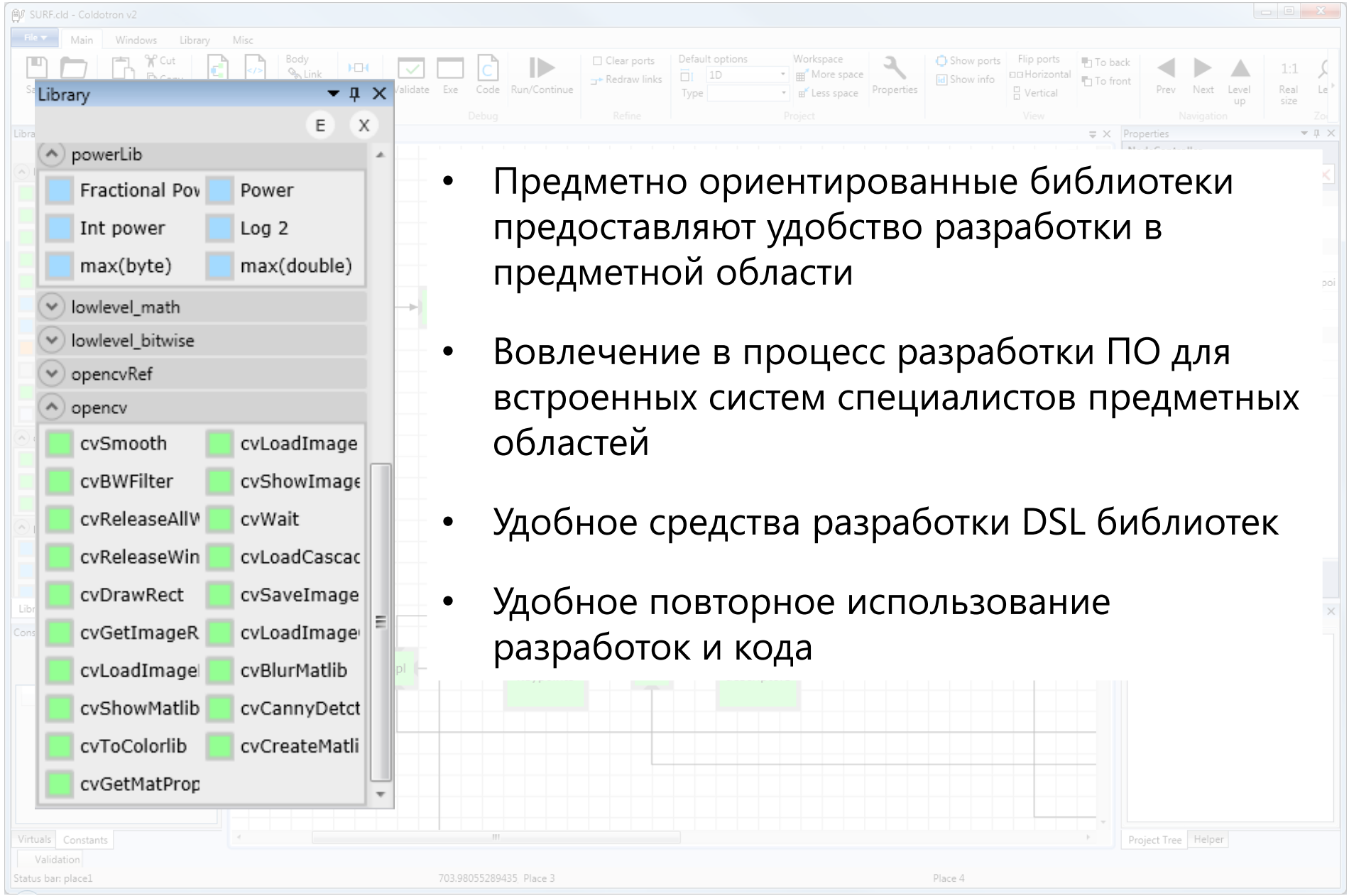
Графический язык программирования VPL

- Когнитивные преимущества:
  - прозрачность процесса разработки
  - прослеживаемость схемы зависимостей
  - наглядность структуры управления вычислениями
- Общие данные
- Естественный параллелизм
- Потенциальная конвейеризация



Программа на языке VPL – схема вычислений из операторов обработки, операторов управления и общих данных

Предметно-ориентированное программирование



- Предметно ориентированные библиотеки предоставляют удобство разработки в предметной области
- Вовлечение в процесс разработки ПО для встроенных систем специалистов предметных областей
- Удобное средства разработки DSL библиотек
- Удобное повторное использование разработок и кода

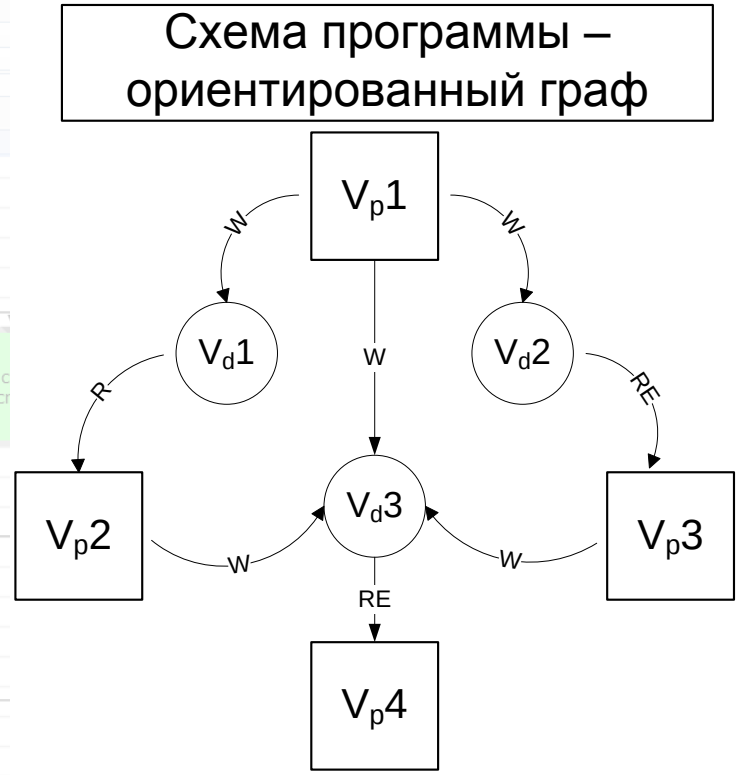
Формальная модель вычислений: Асинхронные Развивающиеся Процессы

АРП-модель определяет

- синтаксис языка
- семантику объектов языка
- управляющие конструкции

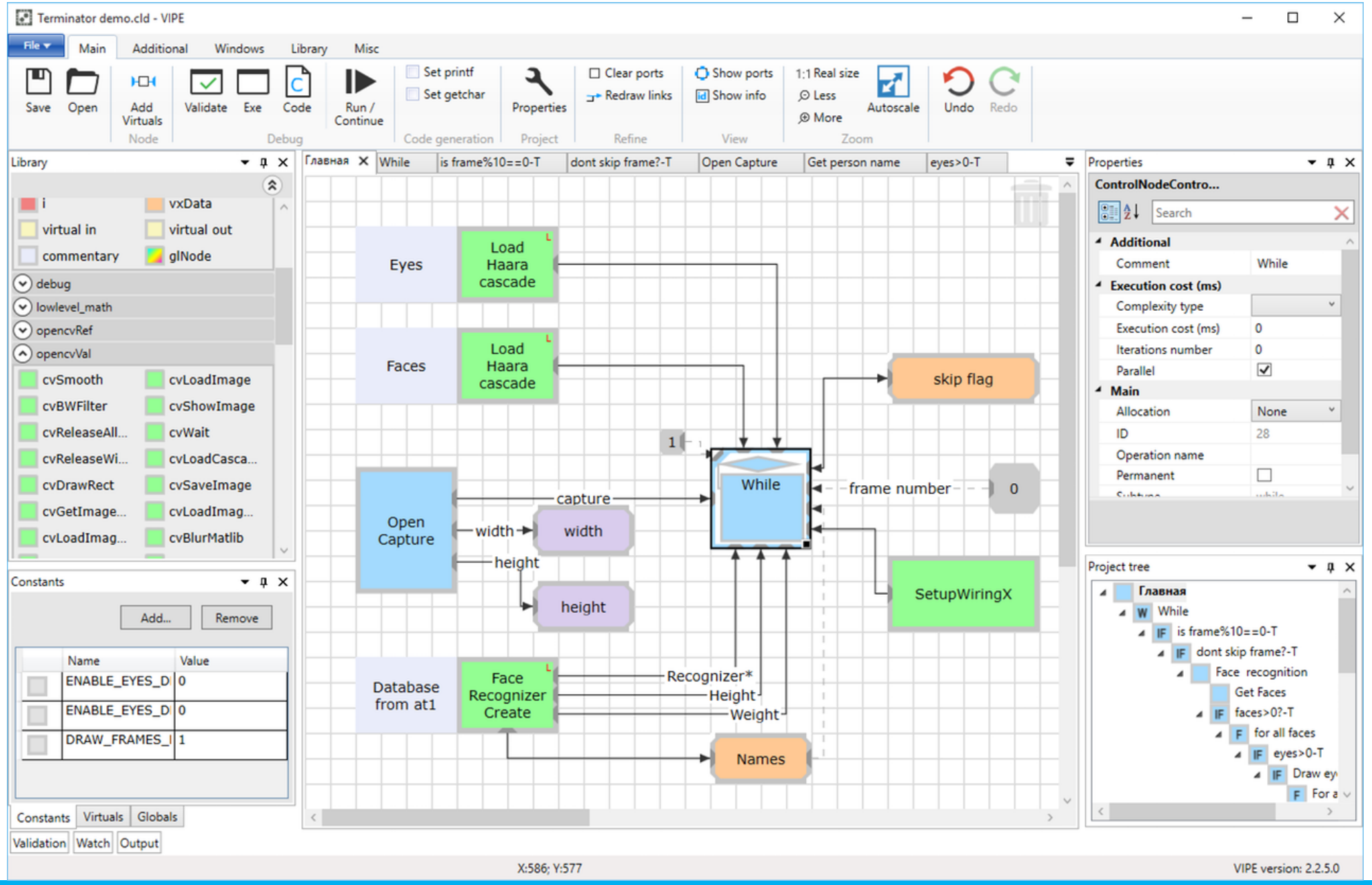
АРП-модель предоставляет

- методы формальной верификации
- идентичность результатов при работе в разных окружениях
- возможности для переносимости вычислений
- представление динамики параллельного вычисления
- совмещение работы с общими данными и модели распределенной памяти



V_p – вершина-оператор,
V_d – вершина-данное,
W/R/RE – дуги (связи) и их разметка

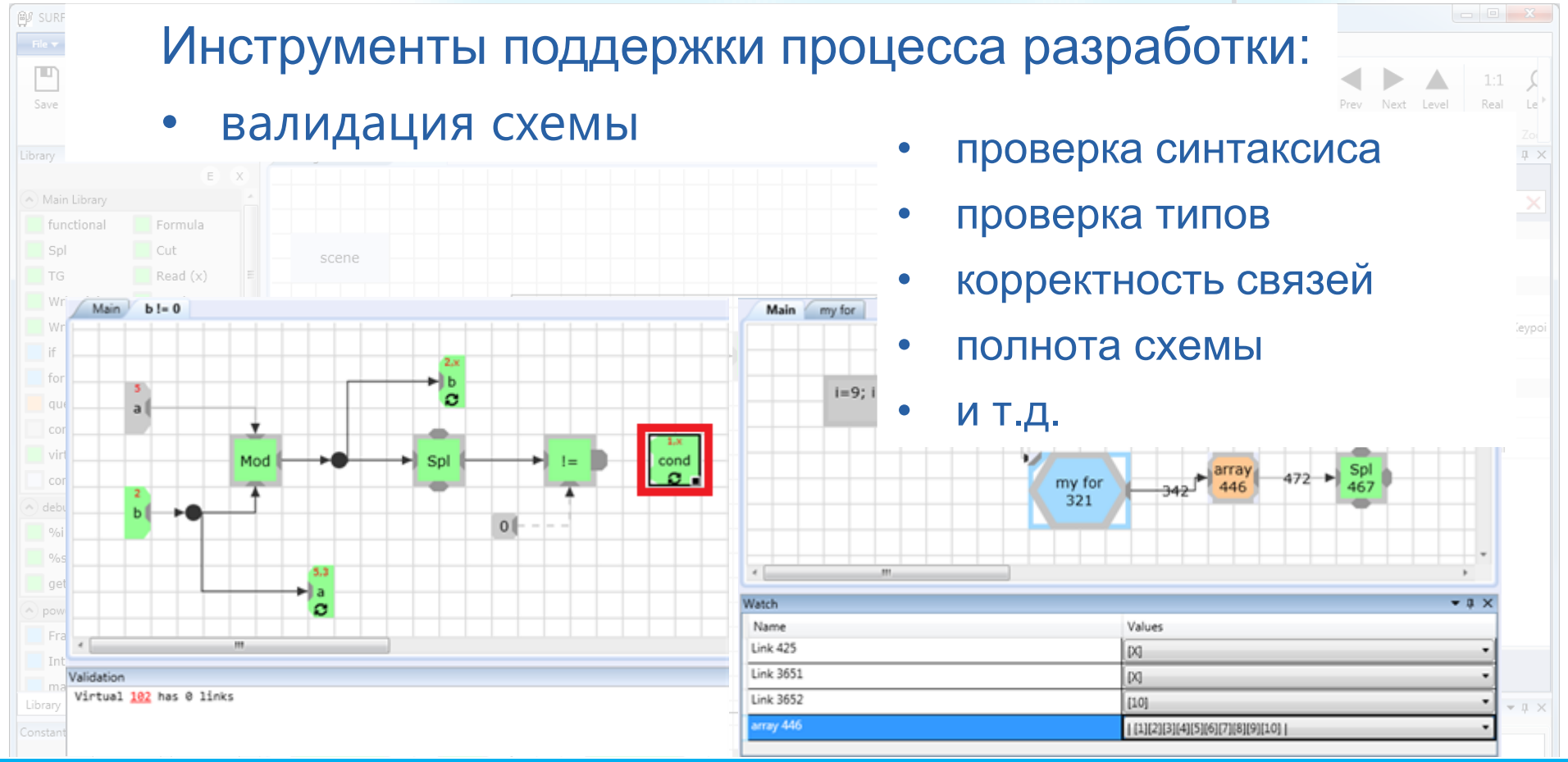
VIPE: визуальная среда проектирования



Интерактивные инструменты

Инструменты поддержки процесса разработки:

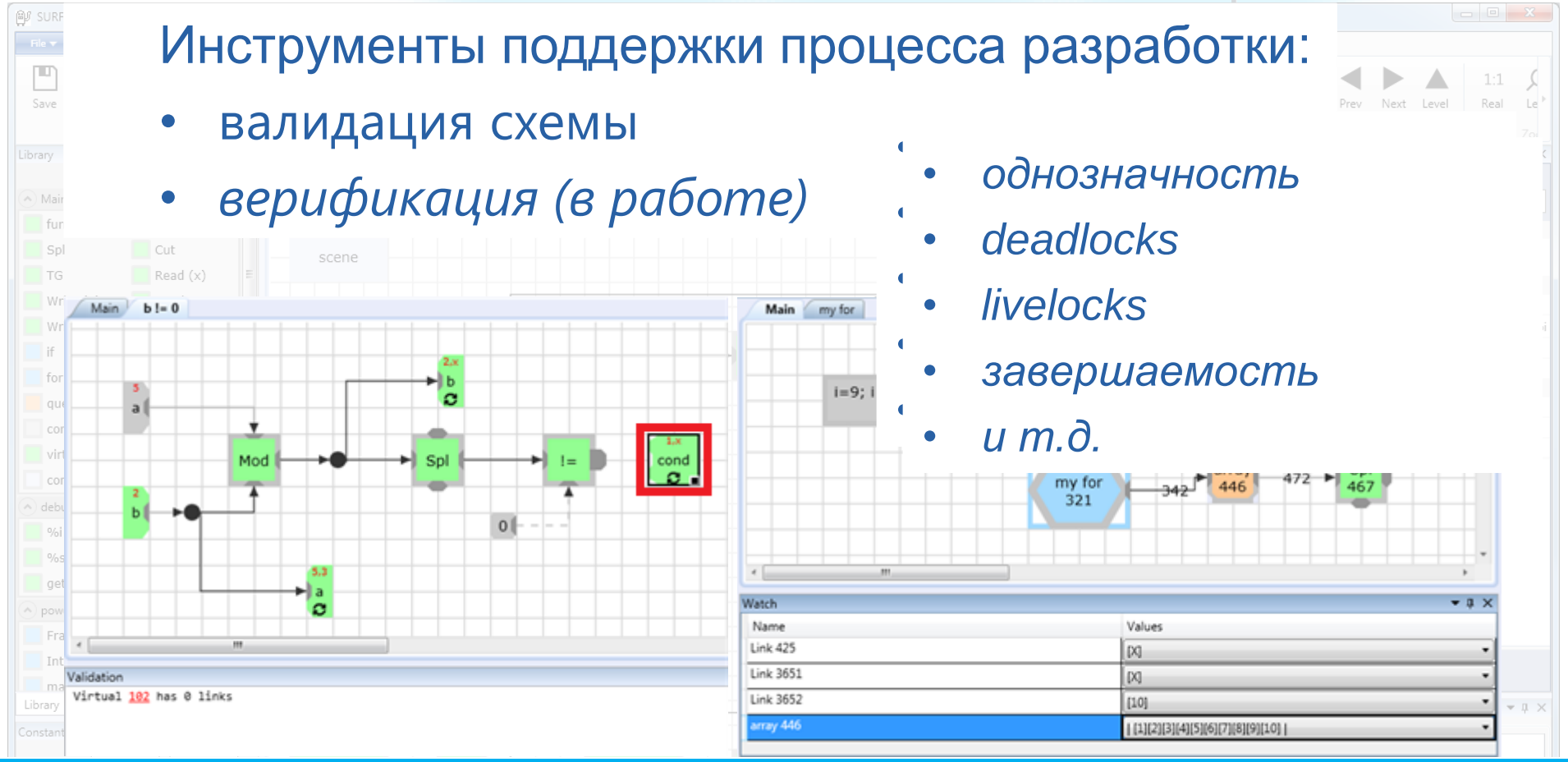
- валидация схемы
- проверка синтаксиса
- проверка типов
- корректность связей
- полнота схемы
- и т.д.



Интерактивные инструменты

Инструменты поддержки процесса разработки:

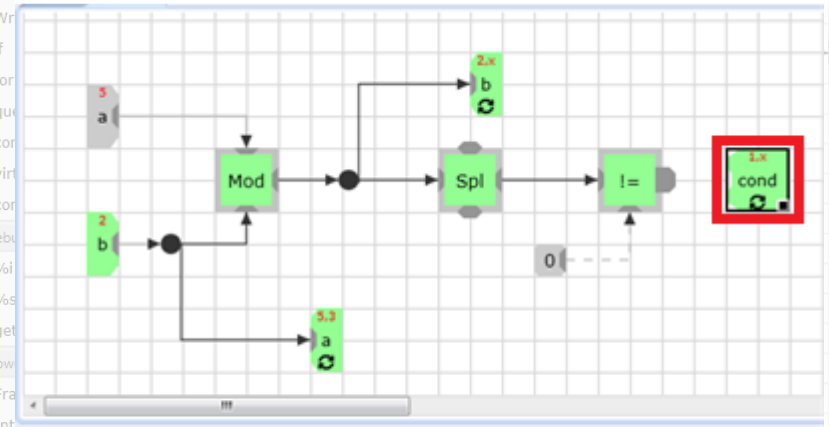
- валидация схемы
- *верификация (в работе)*
  - *однозначность*
  - *deadlocks*
  - *livelocks*
  - *завершаемость*
  - *и т.д.*



Интерактивные инструменты

Инструменты поддержки процесса разработки:

- валидация схемы
- *верификация (в работе)*
- интерактивная отладка и т.д.

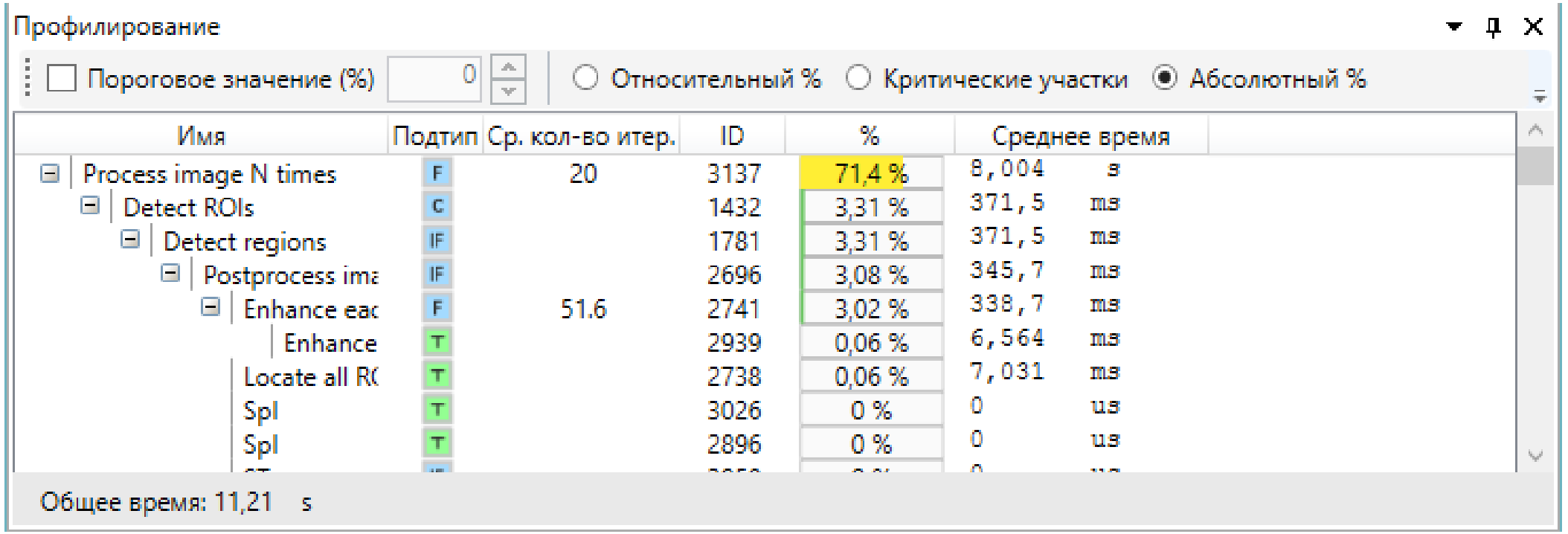


- пошаговая отладка
- breakpoints
- watches
- передача данных
- трассы вычислений
- выполнение операторов
- функциональная отладка при последовательном выполнении

Средства проектирования параллельных программ

Инструмент анализа «визуальный профилировщик»

Поиск критических участков (hotspots) в программе

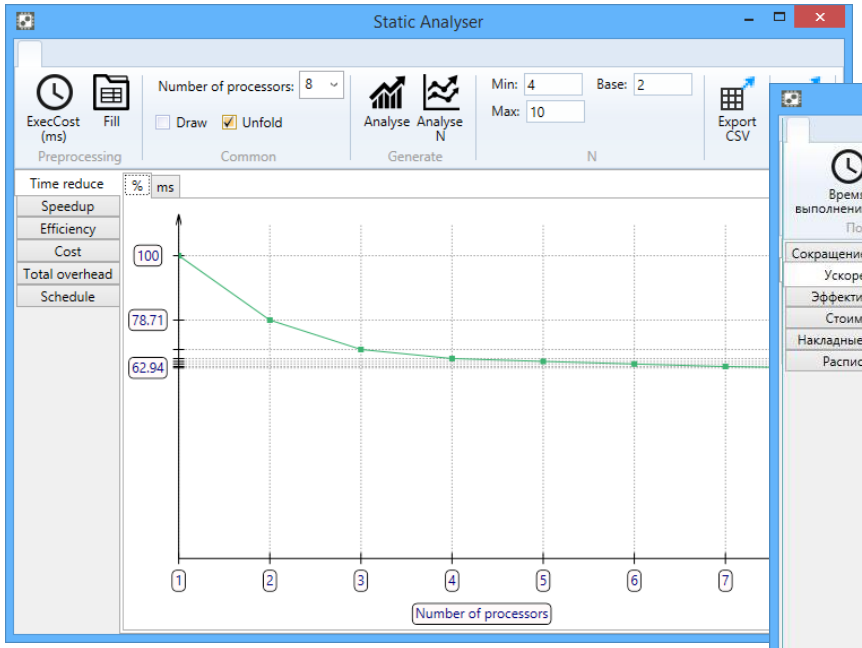


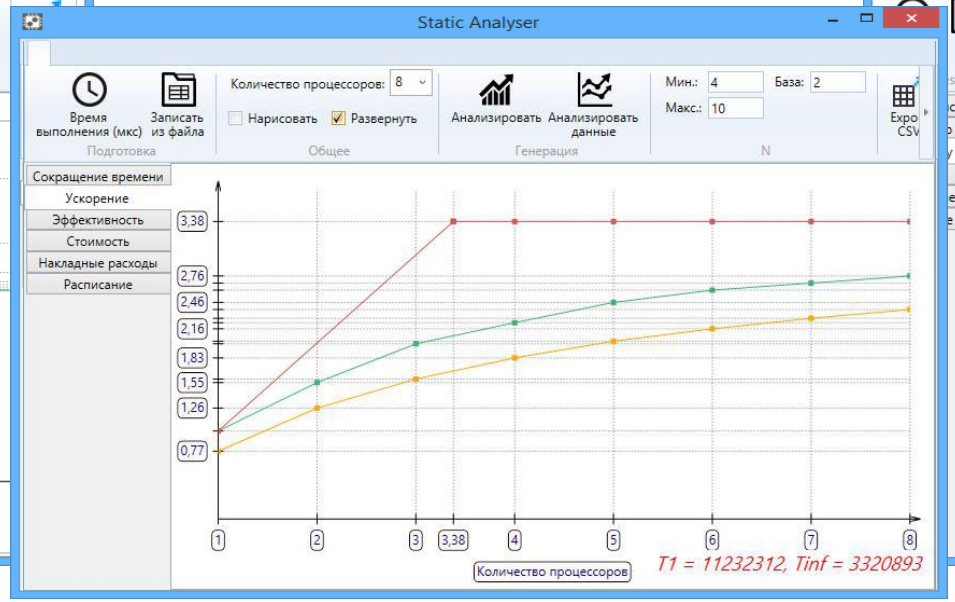
Различные режимы работы:

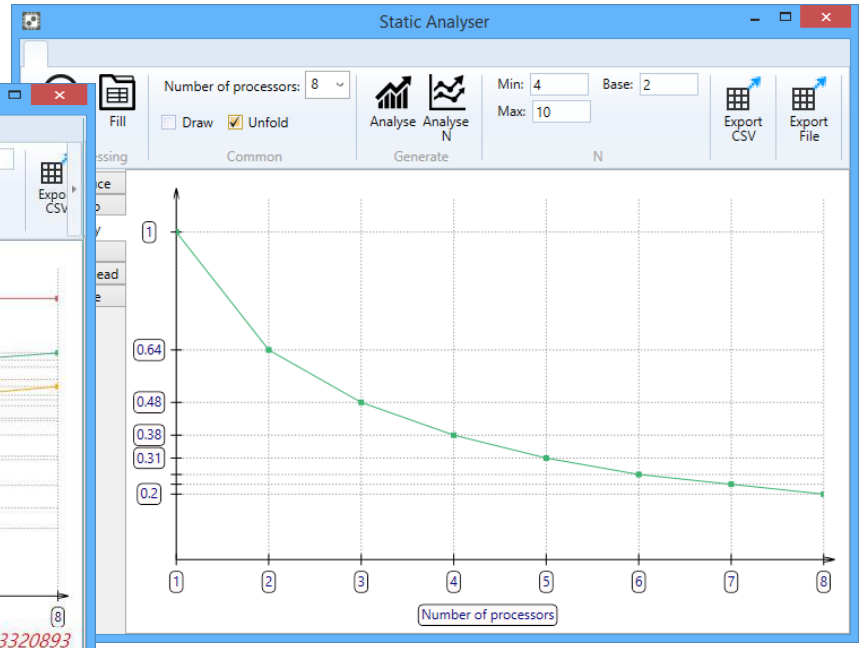
- Абсолютные времена выполнения операторов схемы
- Относительные времена согласно структуре вложенностей
- Критические участки с фильтрацией малых величин

Инструмент статического анализа

Быстрая оценка работы ПО на модели многоядерной платформы







Время

$$R = \frac{T_n}{T_1} * 100\%$$

Ускорение

$$S = \frac{T_1}{T_n}$$

Коэффициент эффективности

$$E = \frac{T_1}{N * T_n}$$

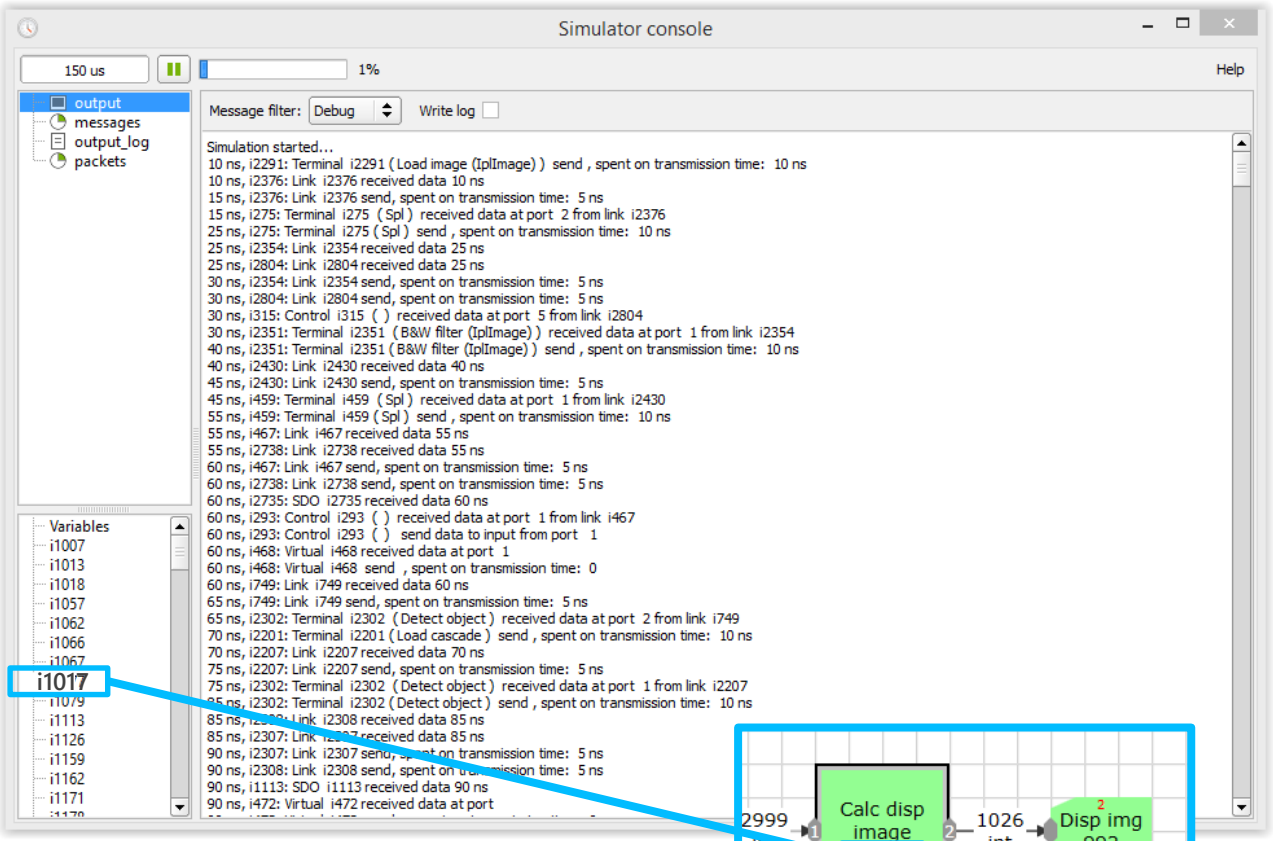
T_1 – время выполнения программы на 1 процессоре

T_n – время выполнения программы на N процессорах

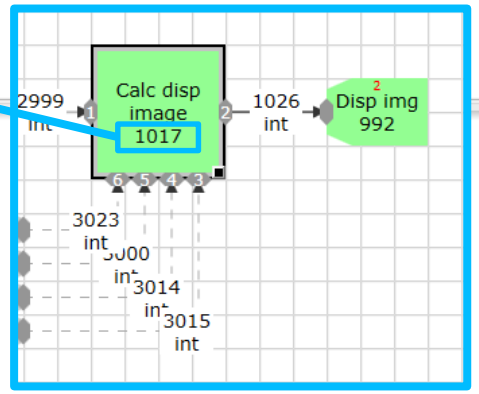
N – количество процессоров

ГУАП

VPL-симулятор



VPL-симулятор

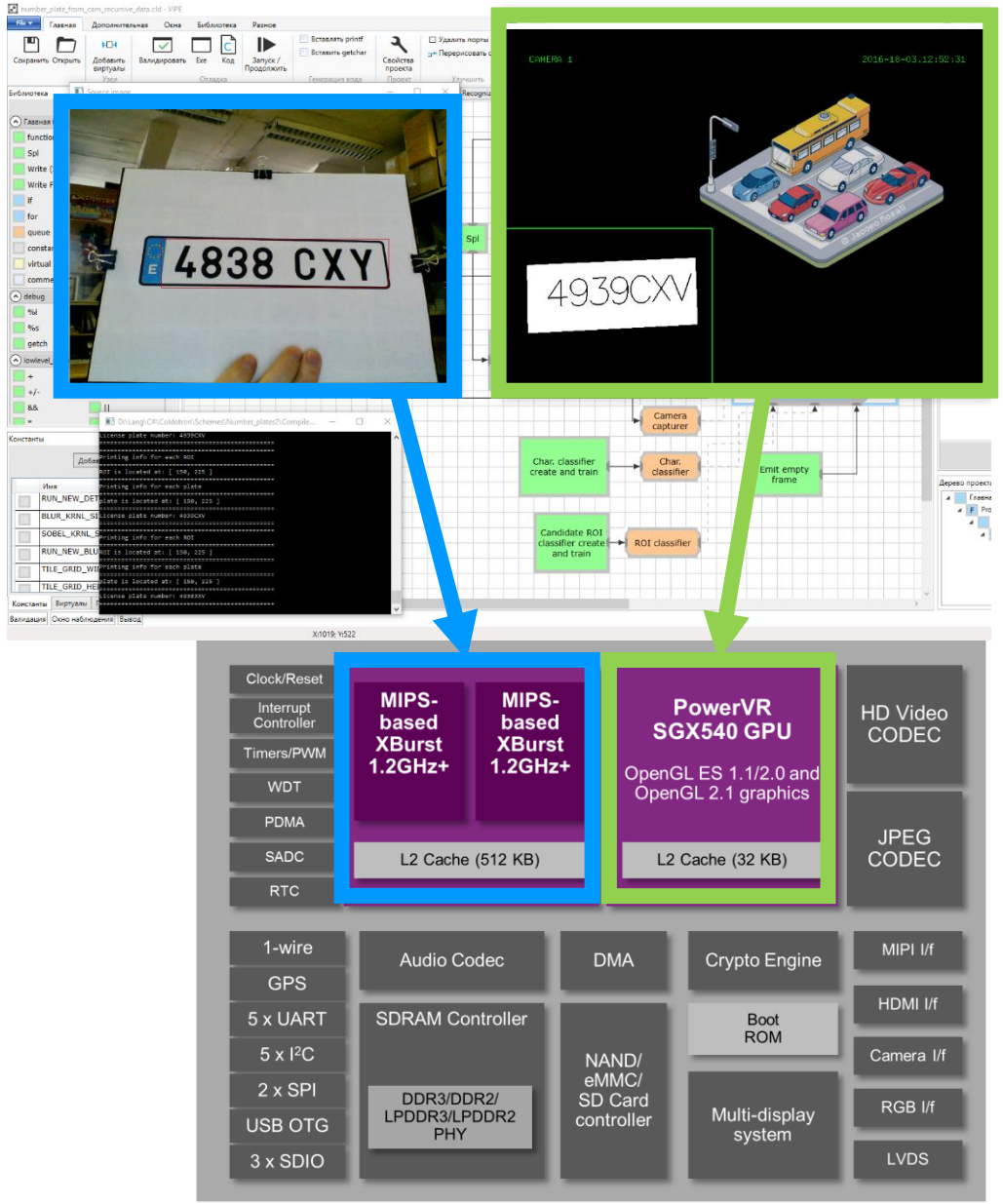


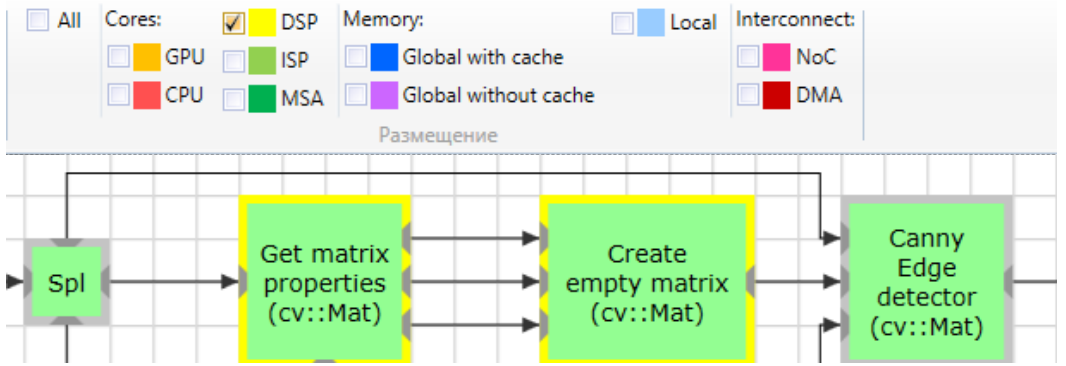
VPL программа

Позволяет оценить:

1. требования к производительности ядер встроенной системы
2. требования к памяти ядер встроенной системы
3. занятость вычислительных ядер для различных вариантов размещения и баланс занятости
4. количество и интенсивность обменов данными
5. эффективность загрузки аппаратного обеспечения
6. узкие места аппаратной платформы, программы и распределения задач

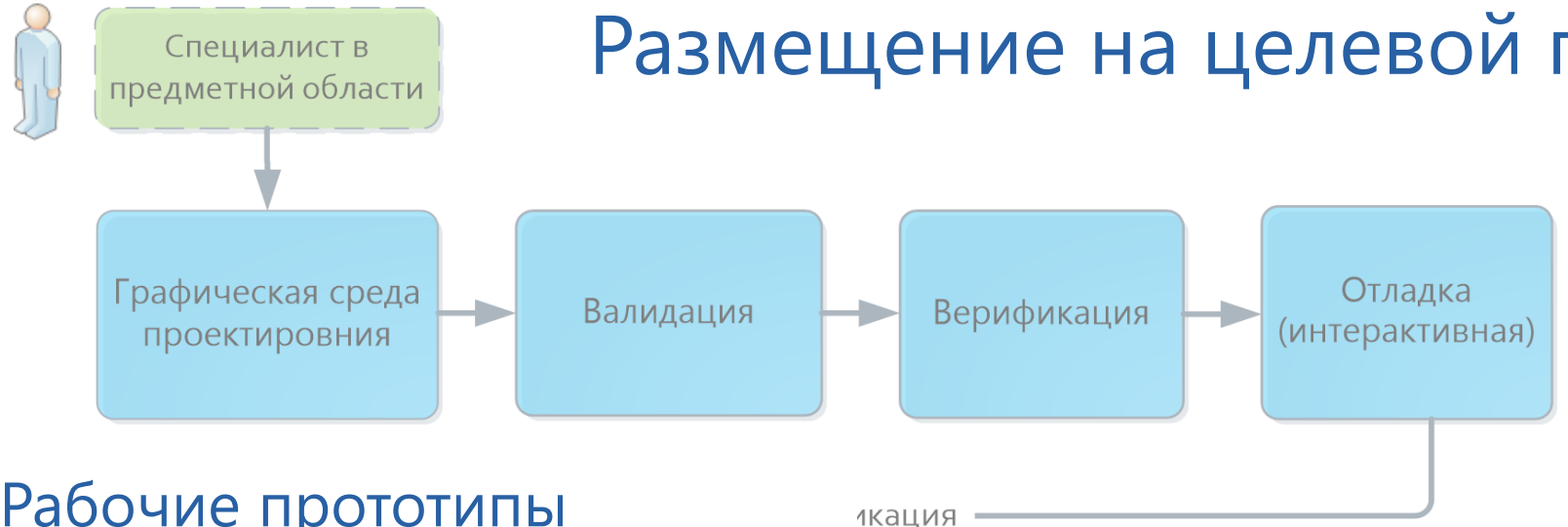
Поддержка неоднородных платформ в VIPE





- Назначение элементов схемы на компоненты аппаратной платформы (CPU, GPU, DSP, DMA)
  - операторов на различные типы ядер
  - данных на разные типы памяти
  - обменов на разные типы соединений
- Выбор подходящей реализации операторов обработки данных
- Подготовка исходных данных и отвод результатов работы оператора программы с учетом специфики различных механизмов обмена данными

Размещение на целевой платформе

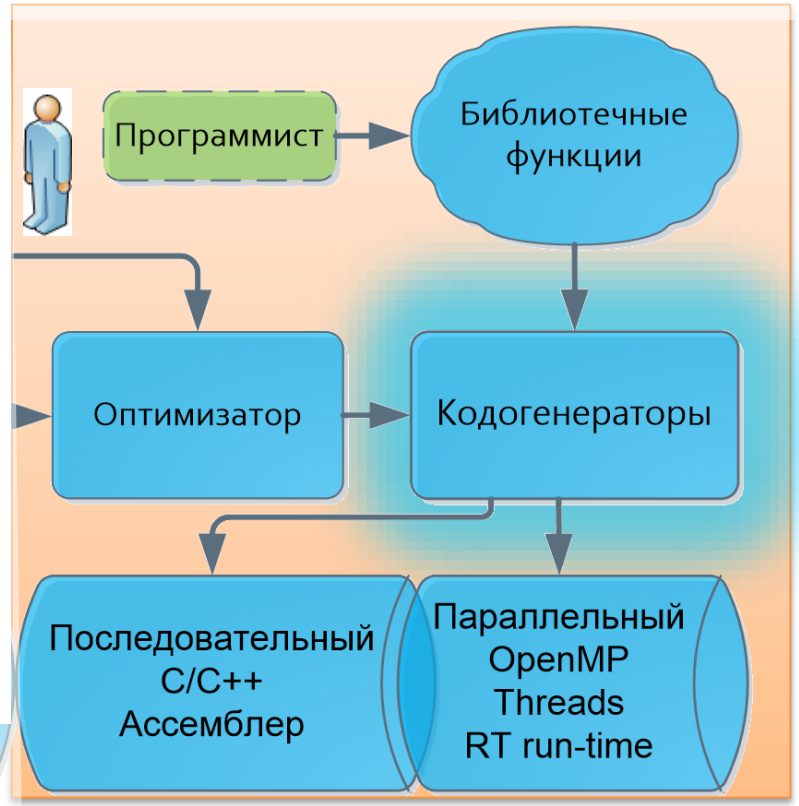


Рабочие прототипы

- ANSI C
- C++
- Параллельный OpenMP
- RT-run-time под платформу *(в работе)*

Proof of concept

- Параллельные потоки
- MPI
- Ассемблер MIPS, DSP



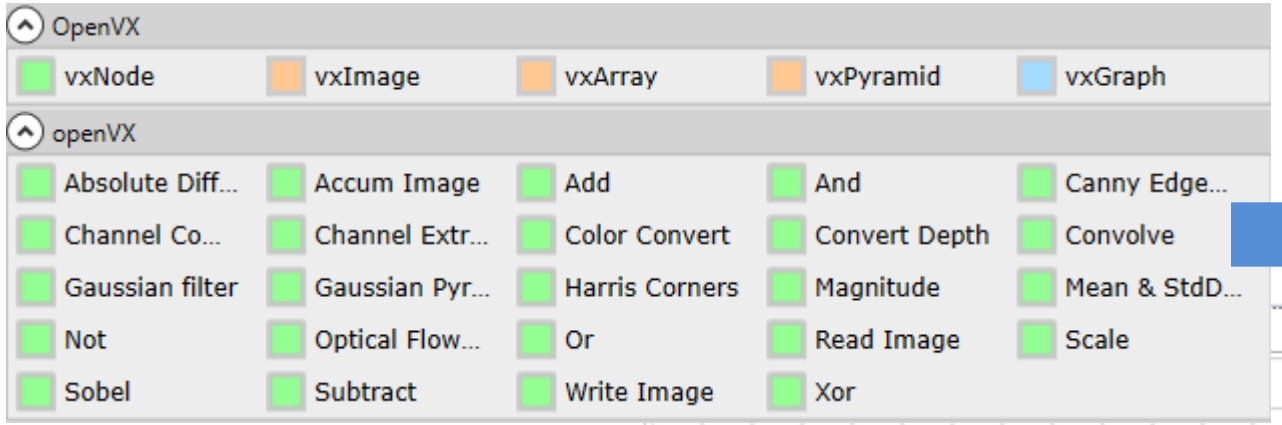
Выводы

- Технология покрывает различные требования к разработке встроенного ПО
- DSL позволяют вовлекать экспертов непосредственно в процесс разработки
- Инструменты оценки производительности помогает разработчику достичь лучших программных характеристик, соблюсти ограничения и требования
- Базовая формальная модель обеспечивает тождественность результатов исполнения программы в любом окружении, в том числе в режиме отладки
- Подход позволяет создавать и отлаживать алгоритм однократно, чтобы потом повторно использовать его в другом окружении и на других платформах
- Технология позволяет быстро подключать новые платформы
- Технология позволяет быстро создавать прототипы для демонстрации их потенциальным клиентам

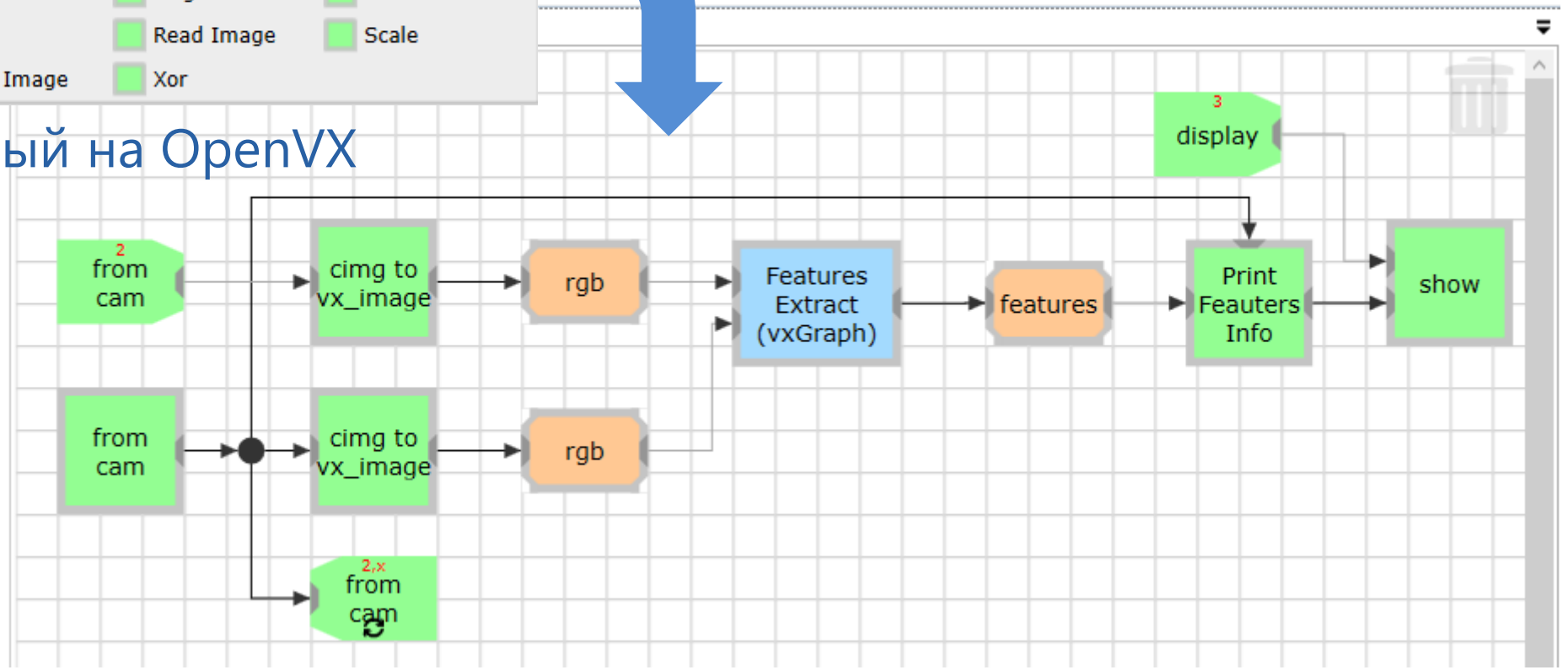
boris.sedov@guap.ru, sheynin@aanet.ru

Примеры применения

Слежение за особенностями сцены
DSL и проектирование

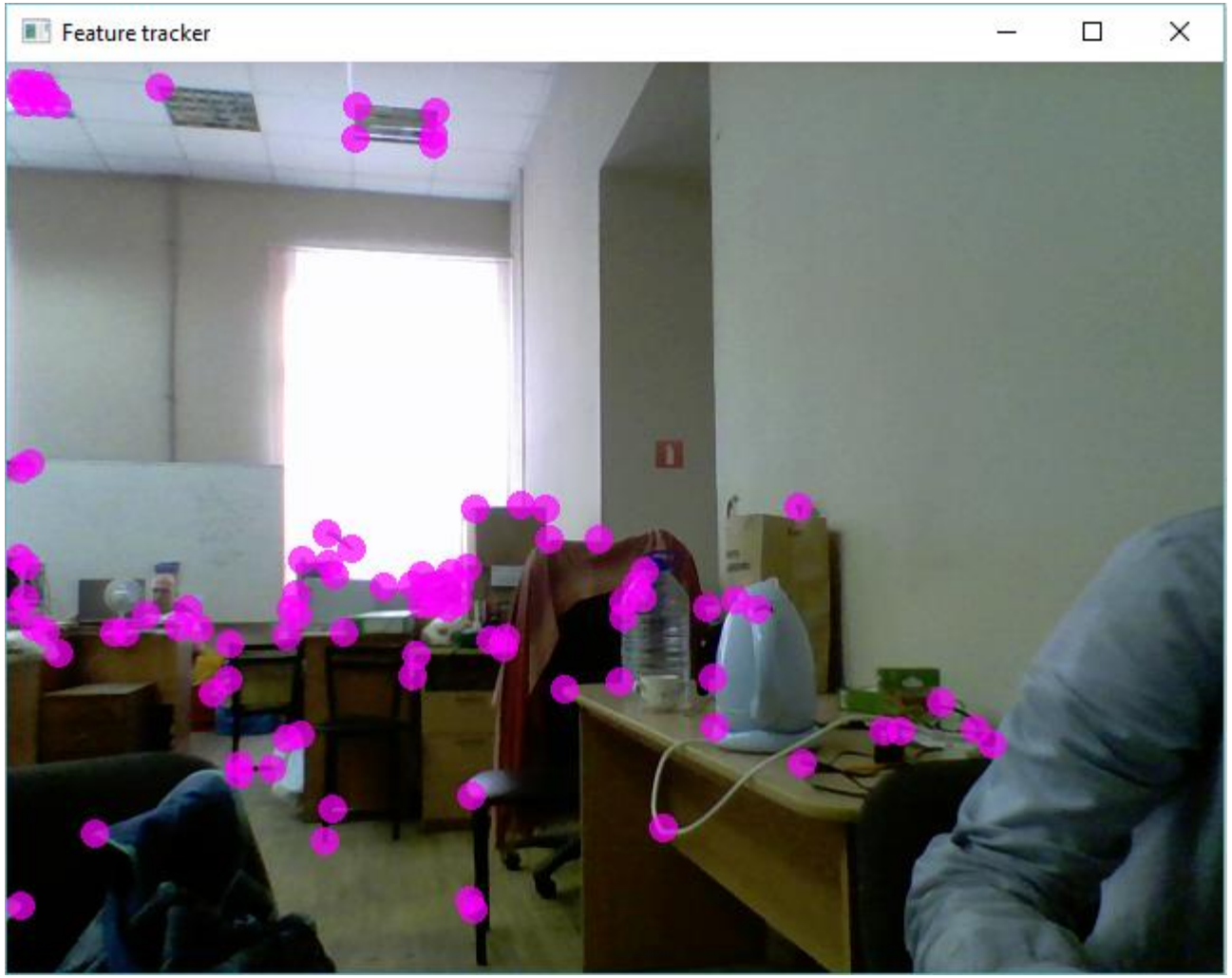


DSL, основанный на OpenVX



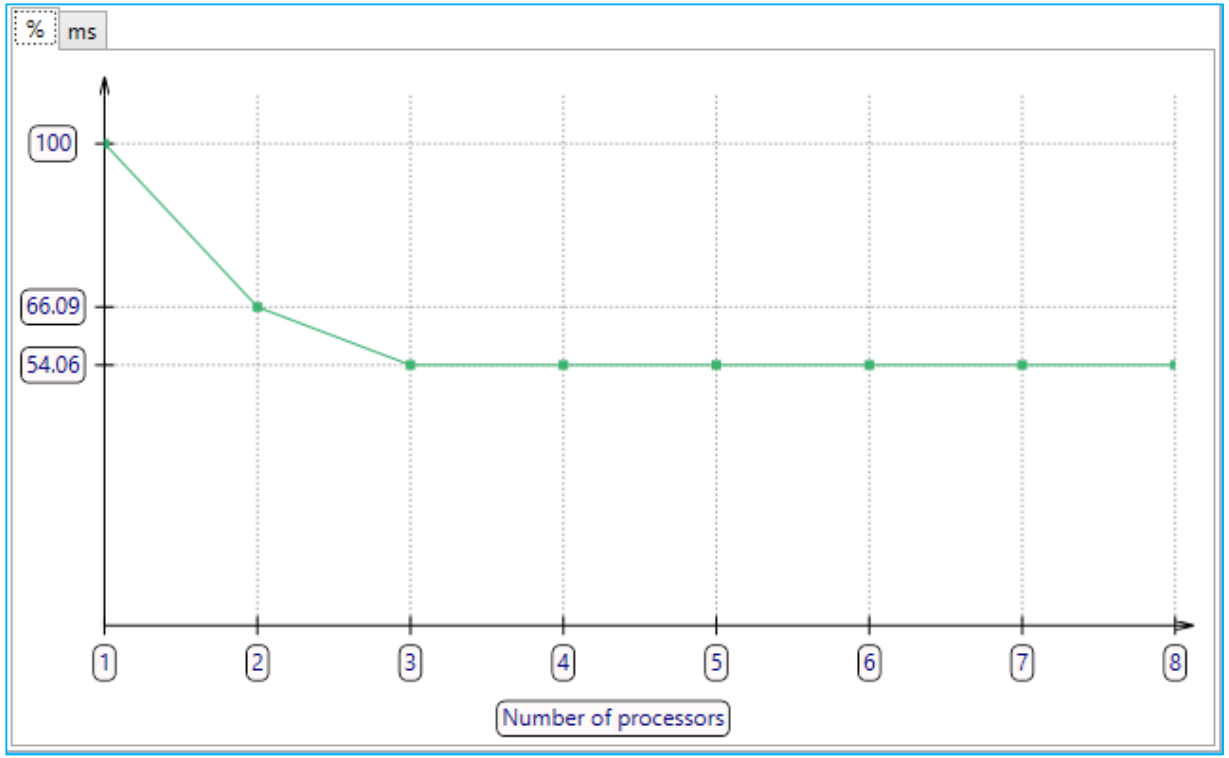
Программа отслеживания особенностей сцены в VIPE

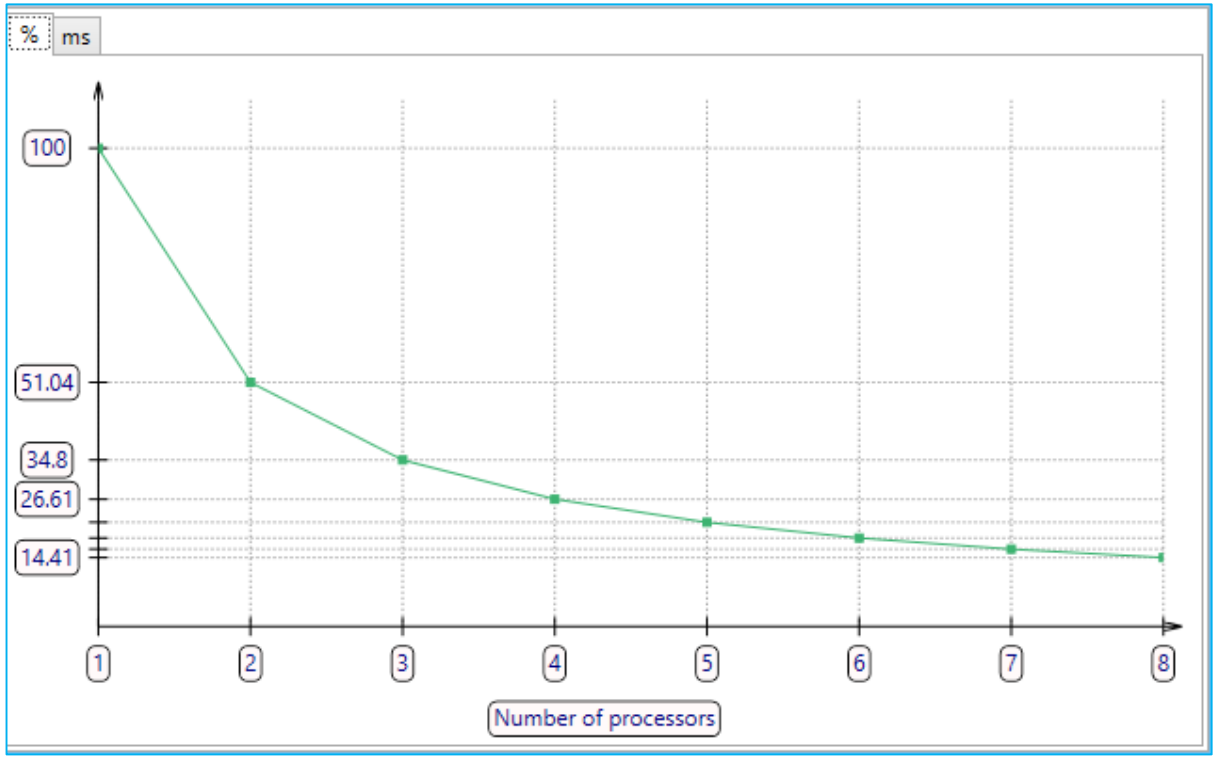
Слежение за особенностями сцены
Результаты выполнения



Программа отслеживания особенностей схемы запускалась на платформе x86 с использованием реализации от Khronos

Слежение за особенностями сцены
Статический анализ



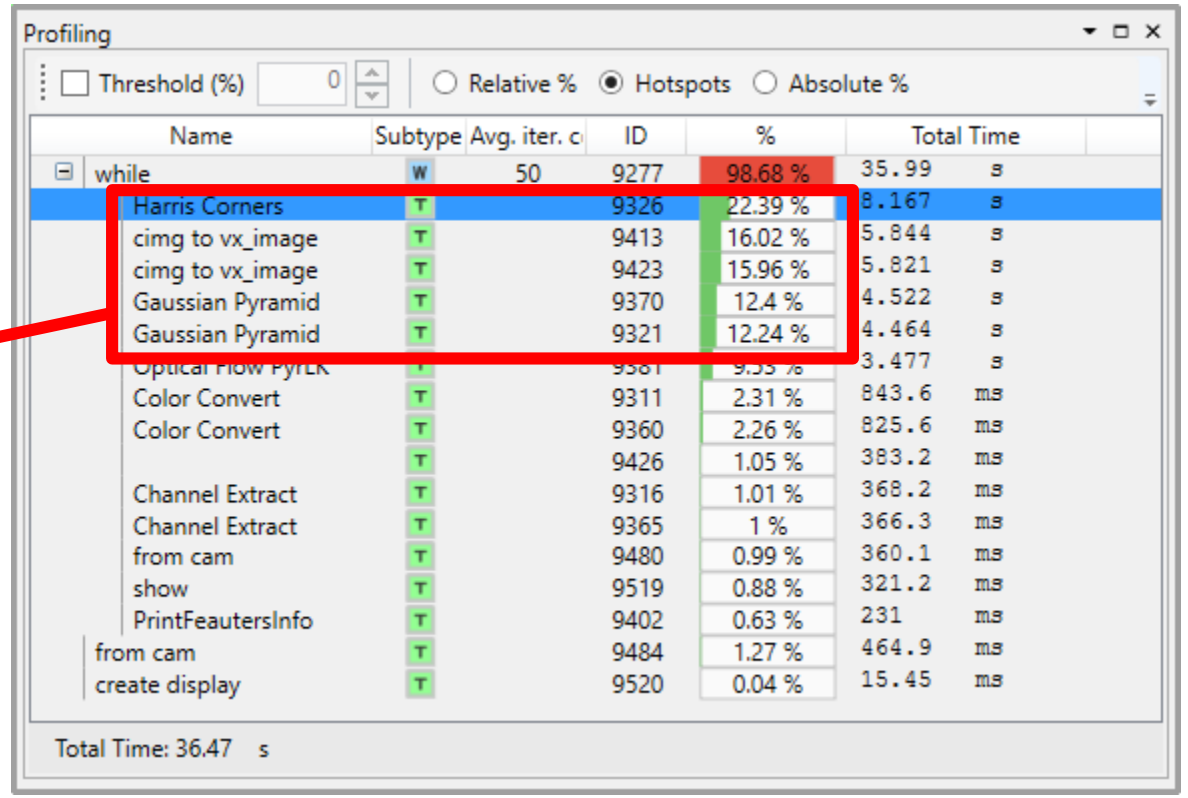


Оценка производительности отслеживания особенностей сцены с **последовательной** обработкой кадров

Оценка производительности отслеживания особенностей сцены с **параллельной** обработкой кадров

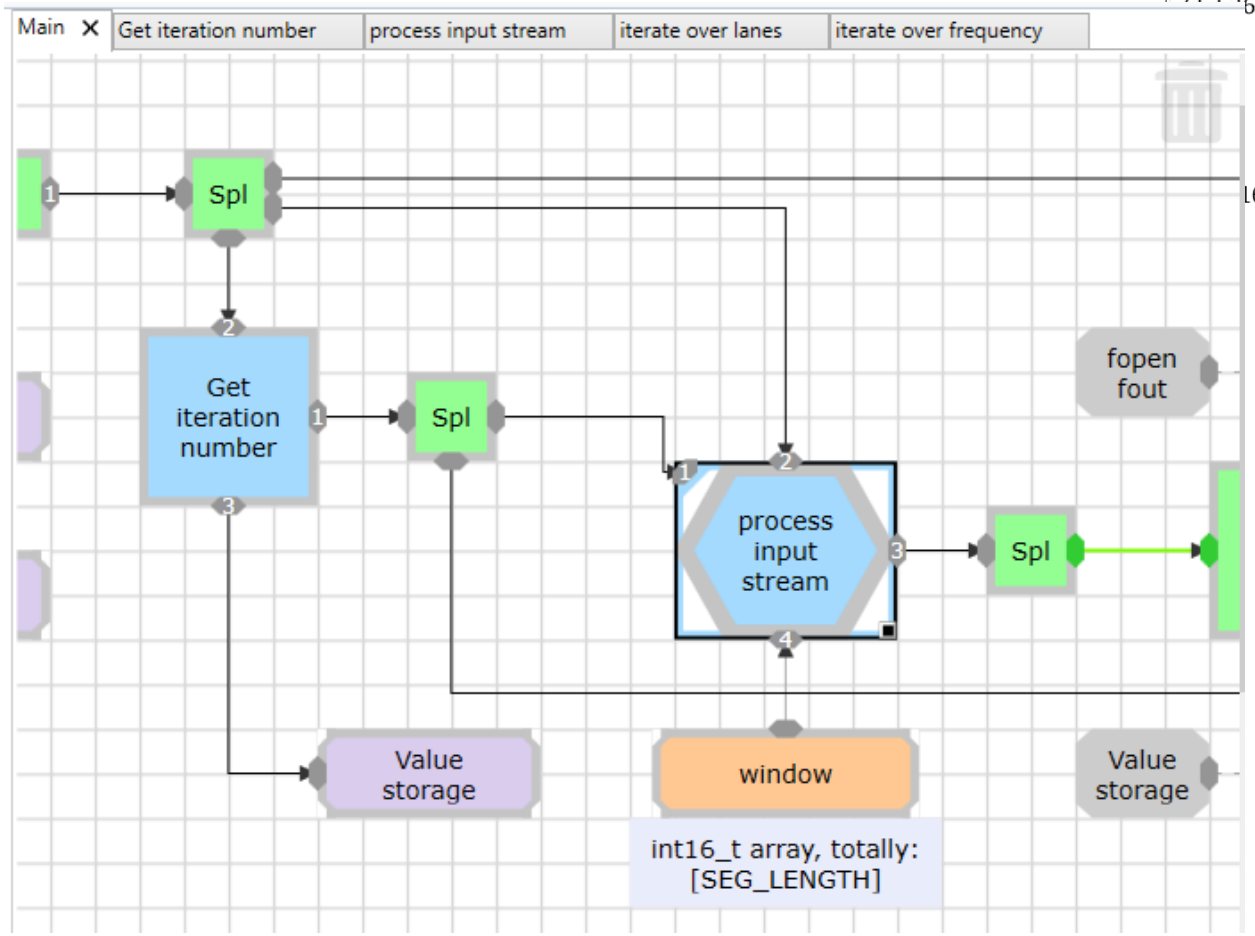
Слежение за особенностями сцены
Визуальный профилировщик

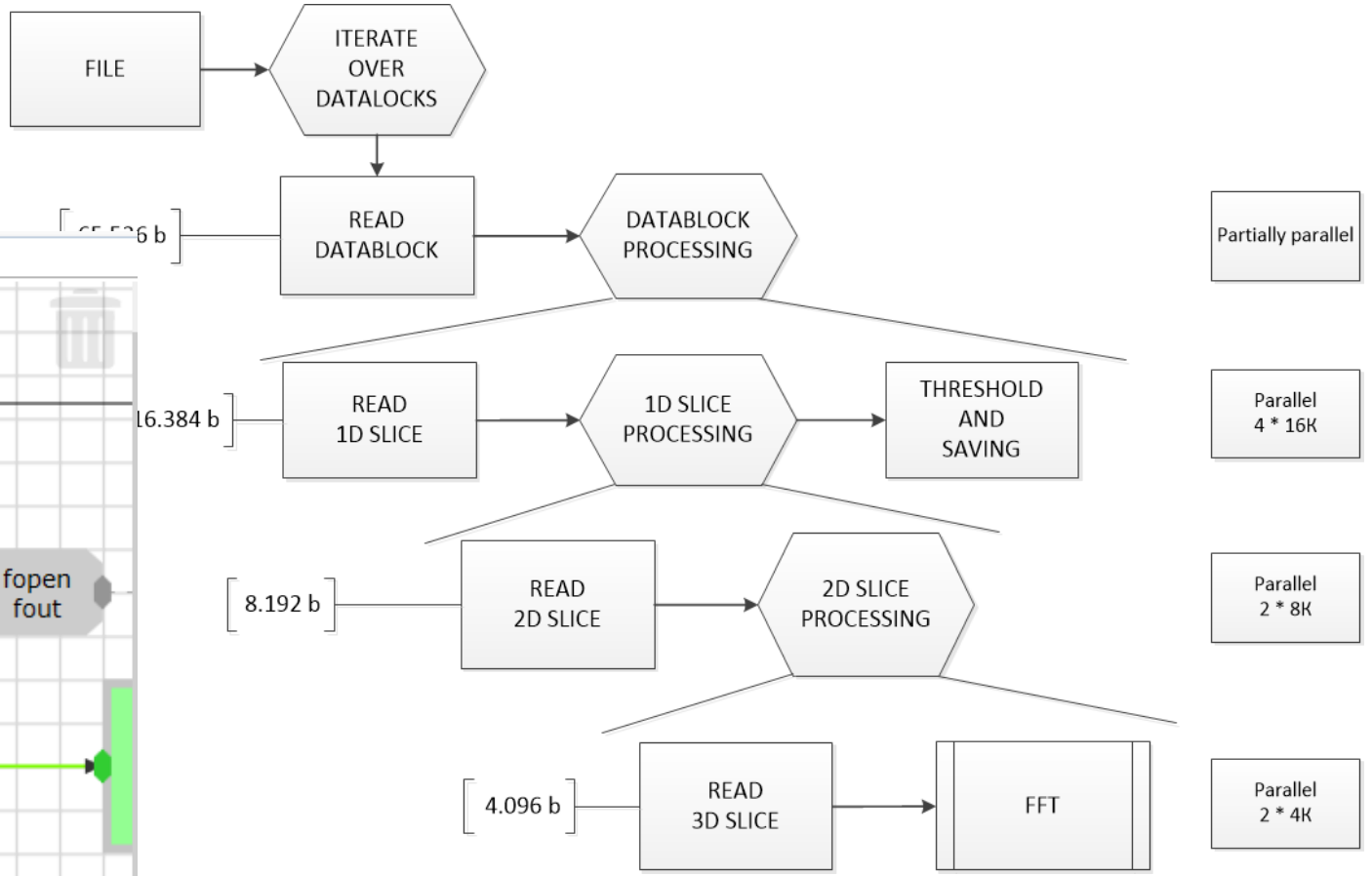
Большое количество времени проходит внутри функций конвертирования изображений (из формата OpenVX и обратно)



Профилирование программы отслеживания особенностей сцены

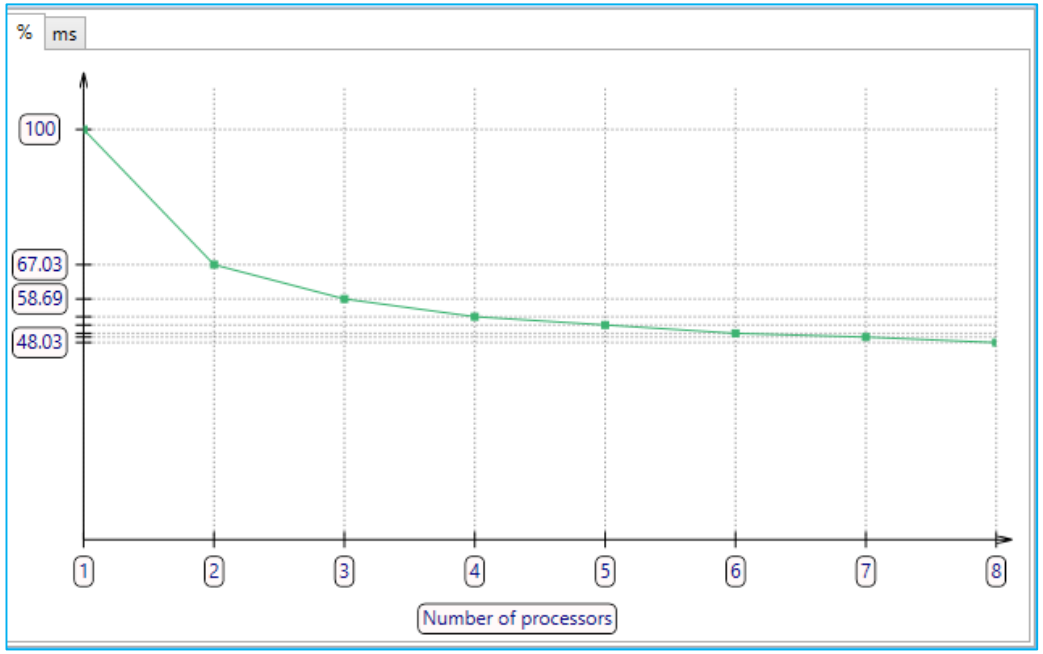
Отслеживание движения объектов радаром
Проектирование



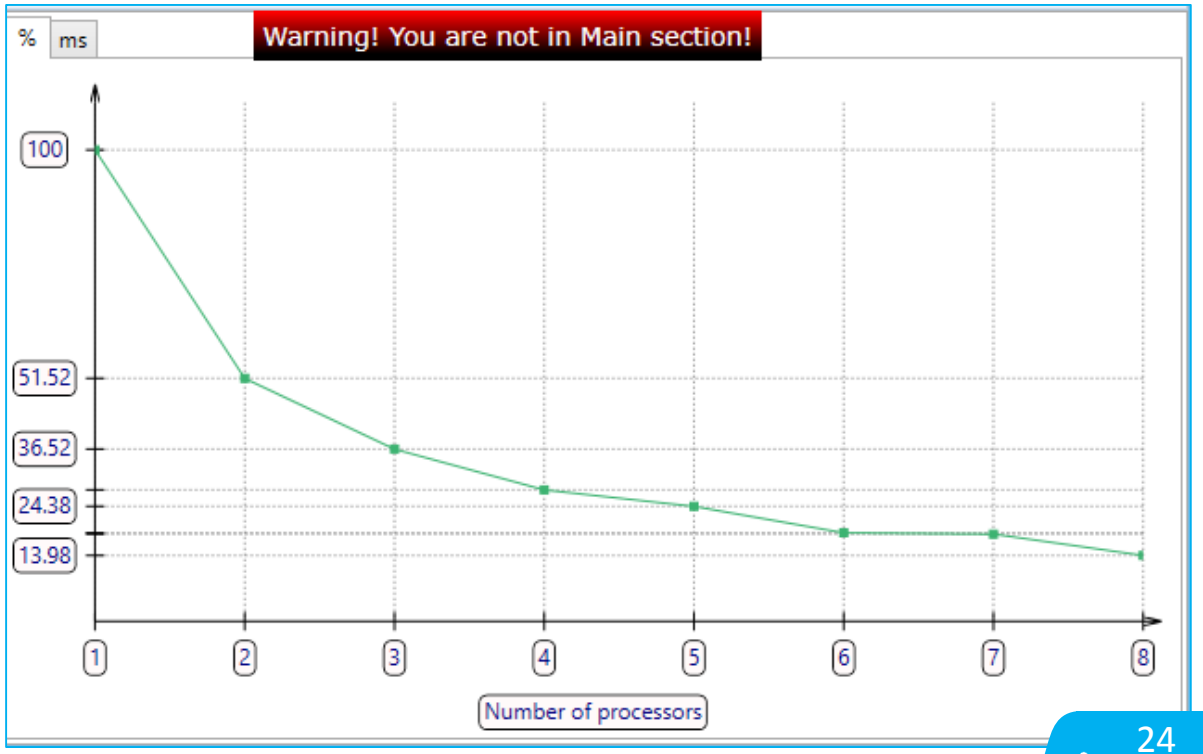


Программа отслеживания движения объектов радаром разрабатывался в проекте PaPP Европейской технической платформы ARTEMIS/ECSEL.

Отслеживание движения объектов радаром
Статический анализ



Статический анализ показывает приемлемое сокращение времени на 2-3 ядрах



Однако, результаты оказались хуже ожидаемых. Статический анализ подпрограммы “Обработка блоков данных” показывает близкое к линейному сокращение времени на 8 ядрах

Отслеживание движения объектов радаром
Визуальный профилировщик

Profiling

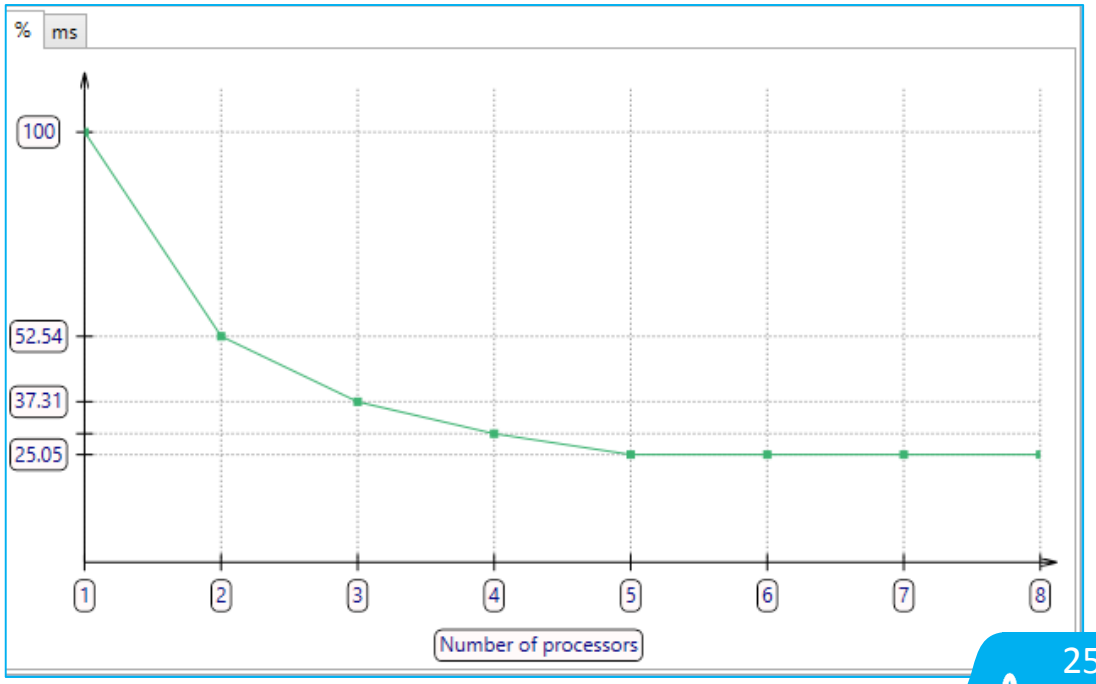
Threshold (%) 0 Relative % Hotspots Absolute %

Name	Subtype	Avg. iter. c	ID	%	Total Time
process input stream	F	97	10009	58.84 %	344.7 ms
Read File	T		10315	42.94 %	251.6 ms
flag==1	IF		10302	15.9 %	93.15 ms
iterate over lanes	F	4	10019	15.9 %	93.15 ms
iterate over fre	F	2	10040	15.22 %	89.19 ms
iterate for l	F	2	10095	15.22 %	89.19 ms
kiss fft	T		10149	15.22 %	89.19 ms
Thresh holding	C		10031	0.34 %	1.986 ms
idx_max>0	IF		10203	0.34 %	1.986 ms
fprintf	T		10229	0.26 %	1.528 ms
get spe	T		10252	0.07 %	421 us
Spl	T		10240	0 %	24 us
range	T		10244	0 %	10 us
angle	T		10248	0 %	2 us
Spl	T		10265	0 %	1 us
Spl	T		10477	0 %	0 us
meaningful pe	T		10054	0.34 %	1.978 ms

Total Time: 585.9 ms

Функция чтения файла находится в последовательной части, следовательно параллелизм ограничен законом Амдала.
Реальное получение данных должно быть оптимизировано, чтобы занимать меньше времени.
Оценка с уменьшенным временем работы функции чтения показывает удовлетворительные результаты.

Визуальный профилировщик выявляет, что большое количество времени тратится на функцию, которая считывает образцы данных из файла (нынешний прототип использует образцы, хранящиеся в файле).



Отслеживание движения объектов радаром
Сравнение результатов анализа, моделирования и исполнения

Ядра (симулятор) или потоки (OpenMP)	Статический анализ сек. / %	Моделирование VPL сек. / %	Исполнение сек. / %
Без OpenMP			1.60
1	1.26 / 100	1.29 / 100	1.65 / 100
2	0.64 / 50.8	0.88 / 68.2	1.00 / 60.6
3	0.60 / 47.6	0.72 / 55.8	0.81 / 49.1
4	0.34 / 30.0	0.55 / 42.6	*1.25 / 76.7*

Платформа
- Core i7 8 ядер -> VirtualBox VM 4 ядра
- Ubuntu 14.04, GCC 4.9.2.

Входные данные
- 12 MB данных сэмплирования сигнала