Добротность программных систем

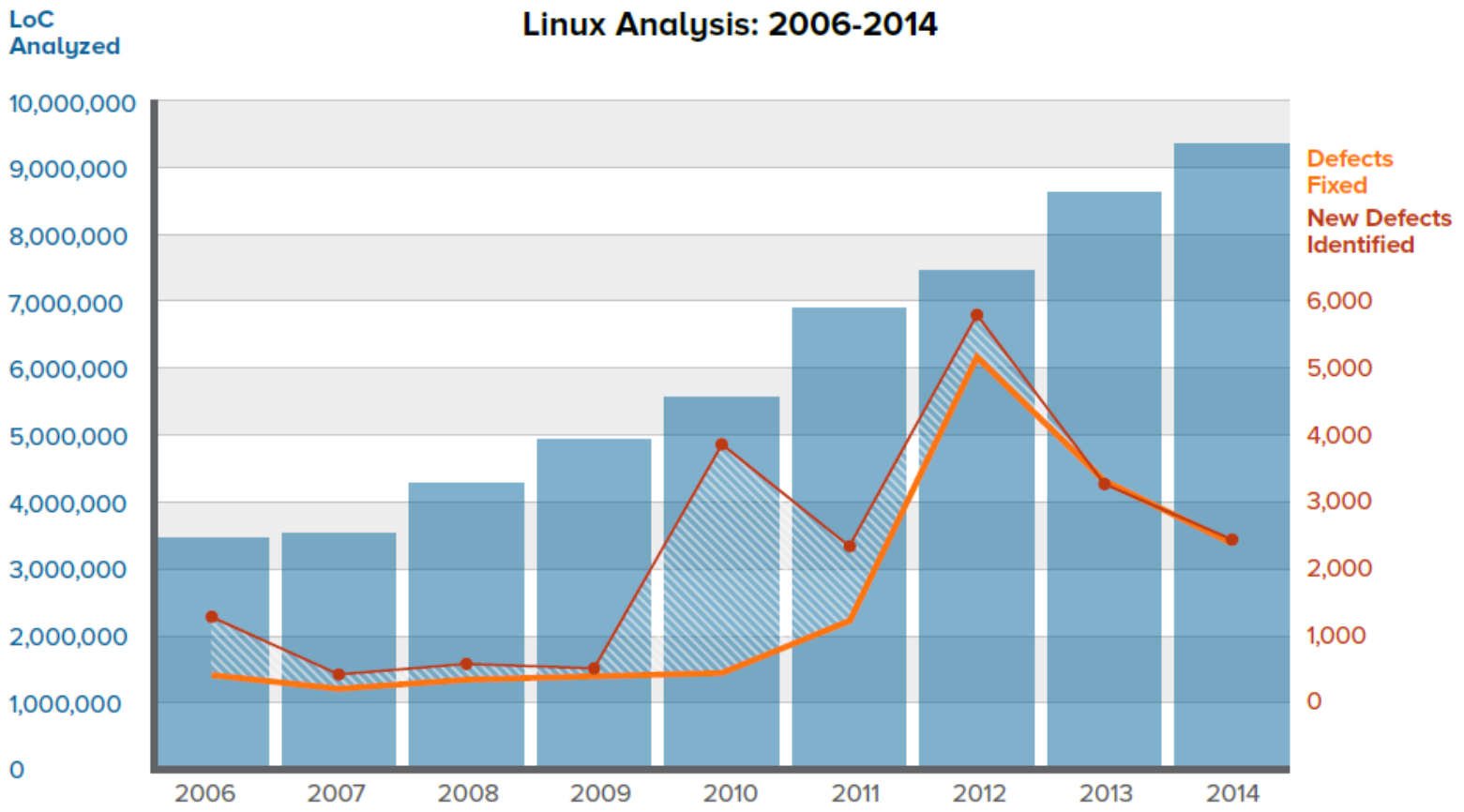
Петренко Александр Константинович,
Институт системного программирования РАН (ИСП РАН)
www.ispras.ru
http://www.ispras.ru/groups/se/
http://panda.ispras.ru/~petrenko/index_ru.html

Ершовские лекции по информатике, 19 апреля 2016 года

Добротность ПО – вечная тема (?)

- 40-50е - Период ламповых ЭВМ
- Конец 60х - Первый кризис программирования
- 70-90е - Технологический прорыв 70-90-х
- 1985 - Стандарты требований к ответственным системам
- 2000е - Проблемы информационной безопасности
- 2010+ - Человек в среде интернета вещей
- 2020+ - The Internet of Things: **Anywhere, Anytime, Anything**

Оценка числа дефектов в открытых проектах



Coverity® Scan. Open Source Report 2014

Плотность дефектов ПО. Тенденции

- Средняя плотность около 1 дефекта на 1 Kloc(*)
- Плотность дефектов в открытых и коммерческих проектах примерно одинаковая
- Скорость исправления уязвимостей в коммерческих проектах выше
- Средняя плотность известных видов дефектов постепенно снижается
- Добротность vs. Time to Market – кто кого?

(*)Способ выявления и подсчета числа дефектов зависит от методики! В данном случае представлена только оценка снизу.

Добротность vs. Time to Market

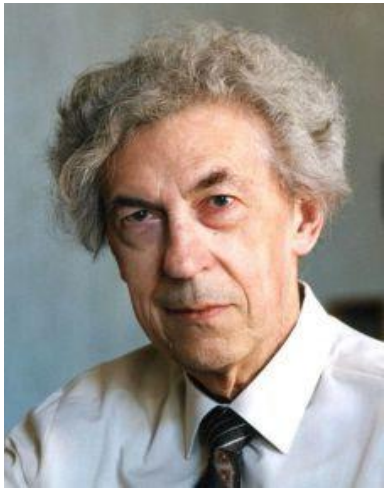
- Факторы, влияющие на добротность
- Экономические, социальные, технические и другие источники проблем
- **Классификация систем/функций по требованиям к уровню добротности**
- Оценка границ возможного
- Баланс усилий и результата
- Интеграция подходов

Направления работ. Ретроспектива

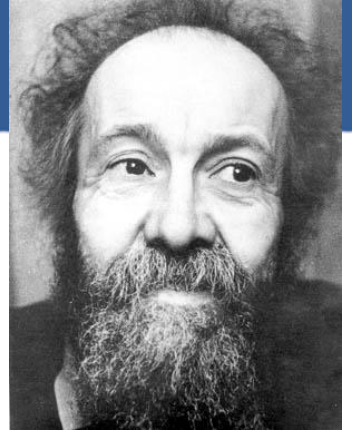
- Автоматизация (механизация)
- Синтез/генерация/re-use
- Разработка «сверху-вниз». Строгий или формальный подход
- Верификация
- MDA/MDD/MDSE
- Дисциплина программирования. Управление, инженерия, наука.

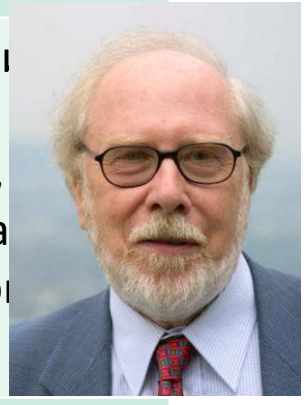
Направление работ	Вехи
Автоматизировать (механизировать)	М. Уилкс, М.Р.Шура-Бура, Е.А.Жоголев, А.П.Ершов, С.С.Лавров, Э.З.Любимский
Синтезировать по спецификации свойств /Генерация по шаблонам / Re-use	Э.Тыугу, С.С.Лавров, американские инициативы, ЦФАП, Сборочное программирование (Г.С.Цейлин, Е.М.Лаврищева), Product Line, Face (Future Airborne Capability Environn
Разработка «сверху-вниз» Специфицировать интерфейсы	А.А.Ляпунов, N.Wirth, B.Liskov (CLU), D. Bjorner и др.(VDM), B.Meyer (Design-by-contract)
Верифицировать	R. Floyd (Дедуктивная верификация), E.Clarke (Model checking), BLAST, CPAchecker (software model checkers), Model/Requirements based testing
Моделировать и прототипировать	MDA/MDD/MDSE
Процессы разработки	Стандарты на процессы (DO-178B/C и др.)

Направление работ	Исторические вехи
Автоматизировать (механизировать)	М. Уилкс, М.Р.Шура-Бура, Е.А.Жоголев, А.П.Ершов, С.С.Лавров, Э.З.Любимский
Синтезировать по спецификации свойств /Генерация по шаблонам / Re-use	С.С.Лавров(СПОРА), Э.Тыугу (ПРИЗ), американские инициативы, ЦФАП, Сборочное программирование (Г.С.Цейлин, Е.М.Лаврищева), Product Line, Face (Future Airborne Capability Environment)
Разработка «сверху-вниз» Специфицировать интерфейсы	А.А.Ляпунов, N.Wirth, B.Liskov (CLU), D. Bjorner и др.(VDM), B.Meyer (Design-by-contract)
Верифицировать	R. Floyd (Дедуктивная верификация), E.Clarke (Model checking), BLAST, CPAchecker (software model checkers), Model/Requirements based testing
Моделировать и прототипировать	MDA/MDD/MDSE
Процессы разработки	Стандарты на процессы (DO-178B/C и др.)

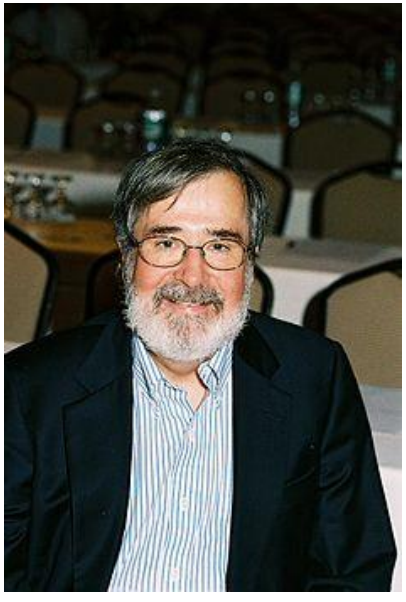


Направление работ	Исторические вехи
Автоматизировать (механизировать)	М. Уилкс, М.Р.Шура-Бура, Е.А.Жоголев, А.П.Ершов, С.С.Лавров, Э.З.Любимский
Синтезировать по спецификации свойств /Генерация по шаблонам / Re-use	Э.Тыугу, С.С.Лавров, американски инициативы, ЦФАП, Сборочное программирование (Г.С.Цейлин, Е.М.Лаврищева), Product Line, Fa (Future Airborne Capability Enviro
Разработка «сверху-вниз» Специфици-ровать интерфейсы	А.А.Ляпунов, N.Wirth, B.Liskov (CLU), D. Bjorner и др.(VDM), B.Meyer (Design-by-contract)
Верифицировать	R. Floyd (Дедуктивная верификация), E.Clarke (Model checking), BLAST, CPAchecker (software model checkers), Model/Requirements based testing
Моделировать и прототипировать	MDA/MDD/MDSE
Процессы разработки	Стандарты на процессы (DO-178B/C и др.)





Направление работ	Исторические вехи
Автоматизировать (механизировать)	М. Уилкс, М.Р.Шура-Бура, Е.А.Жоголев, А.П.Ершов, С.С.Лавров, Э.З.Любимский
Синтезировать по спецификации свойств /Генерация по шаблонам / Re-use	Э.Тыугу, С.С.Лавров, американские инициативы, ЦФАП, Сборочное программирование (Г.С.Цейлин, Е.М.Лаврищева), Product Line, Face (Future Airborne Capability Environment)
Разработка «сверху-вниз» Специфицировать интерфейсы	А.А.Ляпунов, N.Wirth, B.Liskov (CLU), D. Bjorner и др.(VDM), B.Meyer (Design-by-contract)
Верифицировать	R. Floyd (Дедуктивная верификация), E.Clarke (Model checking), BLAST, CPAchecker (software model checkers), Model/Requirements based testing
Моделировать и прототипировать	MDA/MDD/MDSE
Процессы разработки	Стандарты на процессы (DO-178B/C и др.)



Направление работ	Предполагаемый источник проблемы
Автоматизировать (механизировать)	Человек делает свою работу медленно и ошибается
Синтезировать по спецификации свойств /Генерация по шаблонам / Re-use	Человек часто находит лишь квазиоптимальные решения
Разработка «сверху-вниз» Специфицировать интерфейсы	Разработчики разных модулей имеют разное понимание их функциональности
Верифицировать	Все исходные данные и все инструменты могут быть неконсистентными
Моделировать и прототипировать	Нечеткое понимание целевой задачи и сложности реализации
Процессы разработки	Слабость управления

Направление работ	Предполагаемый источник проблемы	Что ограничивает результативность
Автоматизировать (механизировать)	Человек делает свою работу медленно и ошибается	Возможна лишь «прямолинейная» механизация
Синтезировать по спецификации свойств /Генерация по шаблонам / Re-use	Человек часто находит лишь квазиоптимальные решения	Сложность алгоритмов синтеза понижает уверенность в системе
Разработка «сверху-вниз» Специфицировать интерфейсы	Разработчики разных модулей имеют разное понимание их функциональности	Точные спецификации требуют завершенного дизайна
Верифицировать	Все исходные данные и все инструменты могут быть неконсистентны	Задача верификации, как минимум, не проще реализации и требует дефицитных ресурсов
Моделировать и прототипировать	Нечеткое понимание целевой задачи и сложности реализации	Затраты на моделирование не окупаются
Процессы разработки	Слабость управления	Управление само по себе не дает технического результата

О трансформации понимания задач «автоматизации программирования»

А.Перлис - язык программирования – это средство, которое поможет решать интересные нам задачи.

Д.Калинин (по поводу «литературного программирования» Д.Кнута) – это язык, который не позволит написать неправильную программу.

Тони Хоар – верифицирующий компилятор – это скорее ограничивающий, а не помогающий инструмент.

О проблемах MDA/MDD/MDSE

MDA – Model Driven Architecture

MDD – Model Driven Development

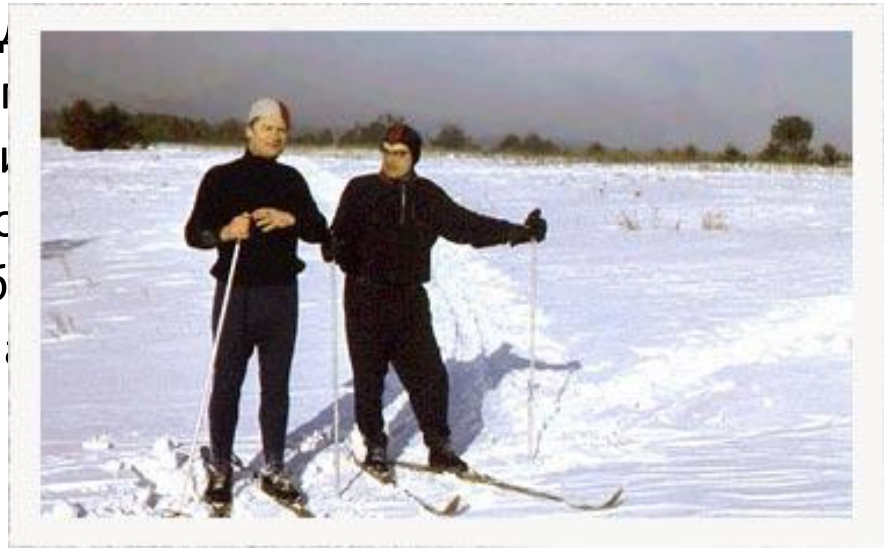
MDSE – Model Driven Software/System Engineering

- Дополнительные издержки на разработку моделей и инфраструктуры для каждого уровня моделирования – без полной обвязки нельзя отлаживаться
- Дополнительная работа по поддержке связей между уровнями – опыт MS Interoperability Initiative
- Поддержка в согласованном виде всех уровней моделей.
- Эффект многоязыковой системы – повышенные требования к персоналу.

О «дисциплине программирования»

И.В.Поттосин «Открытые системы», № 06, 1998
«Добротность программ и информационных потоков»:

Действительно, всякая «хорошая» технология программирования предполагает хорошо продуманную дисциплину разработки, надежно переходящую от спецификации прод
та
со
то
пр
сп
по



О «дисциплине программирования»

И.В.Поттосин - Действительно, всякая «хорошая» технология

…

лишь бы спецификация была полной и адекватной нуждам пользователя.

Алан Перлис – Машина может сделать больше, чем мы можем специфицировать – This is not surprising since computers can compute so much more than we yet know how to specify (Лекции лауреатов премии Тьюринга // Перевод под редакцией Ю.М.Баяковского, Москва, "Мир«, 1993, стр. 16-29).

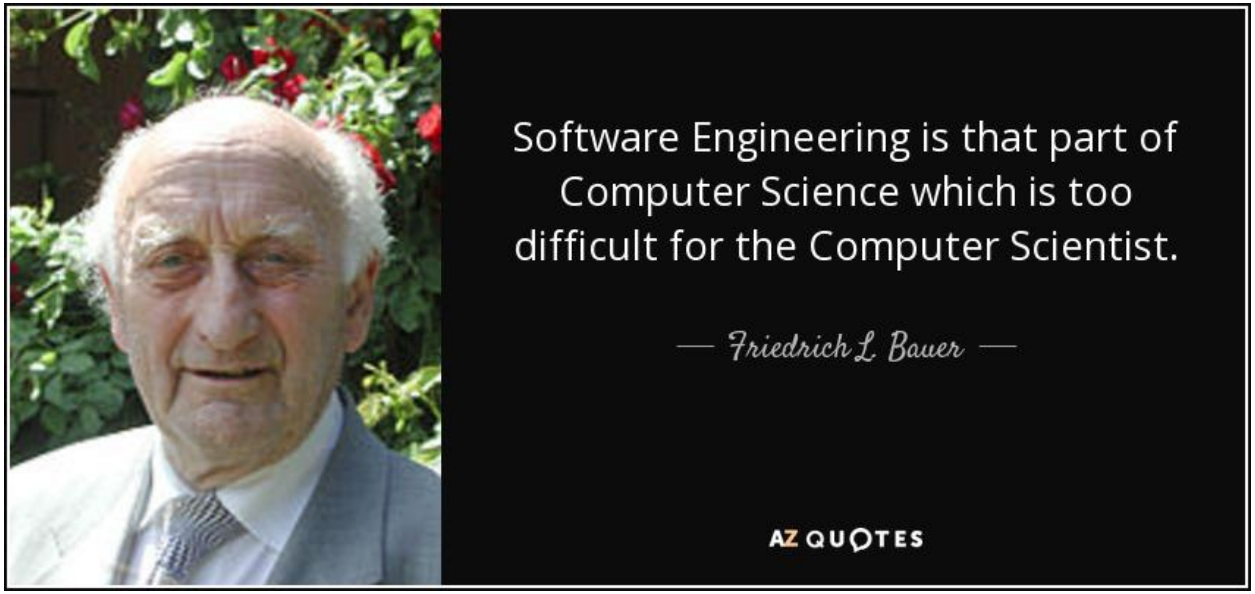
О добротности – промежуточные итоги

- Добротные системы в общем случае сделать невозможно.
- Точнее – не известно как делать абсолютно добротные системы в общем случае.
- Надежда остается – есть методы по повышению добротности систем
  - Задача «измерения добротности» и точного сравнения добротности различных систем сомнительна
  - Остается возможность сравнения характеристик добротности.

Виден ли свет в конце туннеля?

Фридрих Бауэр:

- Программная инженерия это часть Компьютерных наук, которая слишком трудна для ученых из Компьютерных наук.



Истории практического внедрения (1)

Run-time верификация и compile-time **инструментация** кода. Поиск ошибок в работе с памятью, гонки

– Safety critical системы управления (авиация, атомная энергетика, железные дороги, медицинское оборудование и др.)

- **К.Серебряный (Google)** –
  - – MemorySanitizer: fast detector of uninitialized memory use in C++
  - – AddressSanitizer: A Fast Address Sanity Checker
  - – Dynamic Race Detection with LLVM Compiler
  - – ThreadSanitizer – data race detection

Истории практического внедрения (2)

Поиск уязвимостей при помощи нацеленного фази-тестинга. Microsoft SAGE (Patrice Gofroid)

Инфраструктура:

- Valgrind – симулятор
- Z3 – прувер
- DART-Concolic подход

– Статистика.

Пример верификации одного MS Office приложения (файл 47 Kb):

- Число пройденных инструкций – 1,455,506,956
- Число «выполнимых» путей – 2,980
- Число «невыполнимых» путей – 22,978

– Статистика по верификации объемом в 500 машино-лет

- 129,648,907 логических уравнений

Истории практического внедрения (3)

Верификация драйверов и других моделей ядра ОС

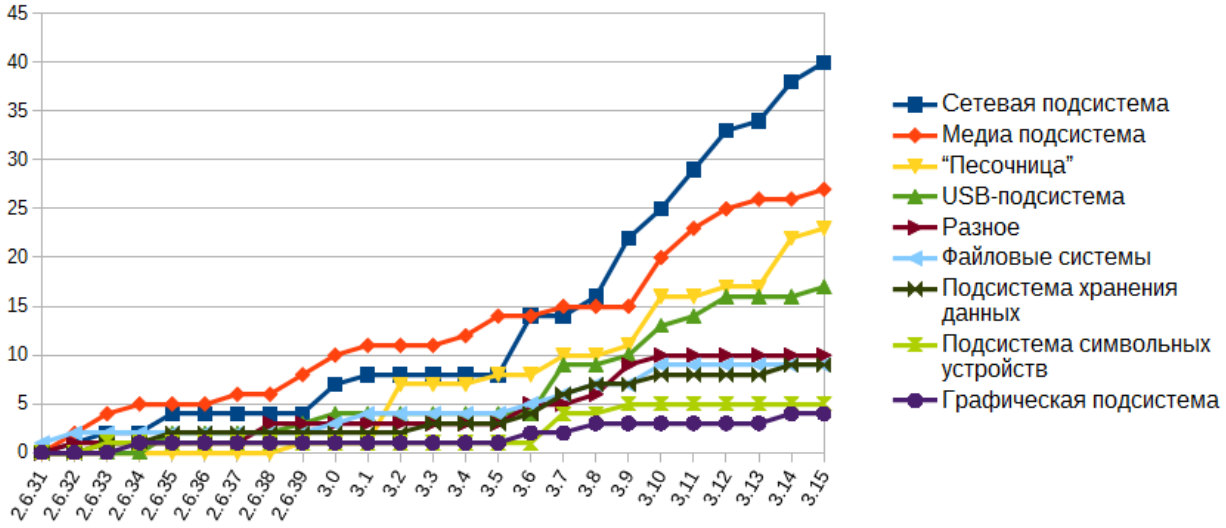
Microsoft SLAM/SDV Windows и ИСП РАН Linux Driver Verification (LDV)

- Подход - Software model checking,
  CEGAR - Counter-Example Guided Abstraction Refinement
- Назначение - Поиск нарушений правил безопасного взаимодействия модулей в ядре ОС
- **Статистика LDV (ИСП РАН)**

  Инфраструктура: CPAchecker (Passau University), BLAST 2.8 (Berkeley&ИСПРАН)

Результаты на январь 2014

Число др
Объем ко
Число пр
Максима
Среднее правила



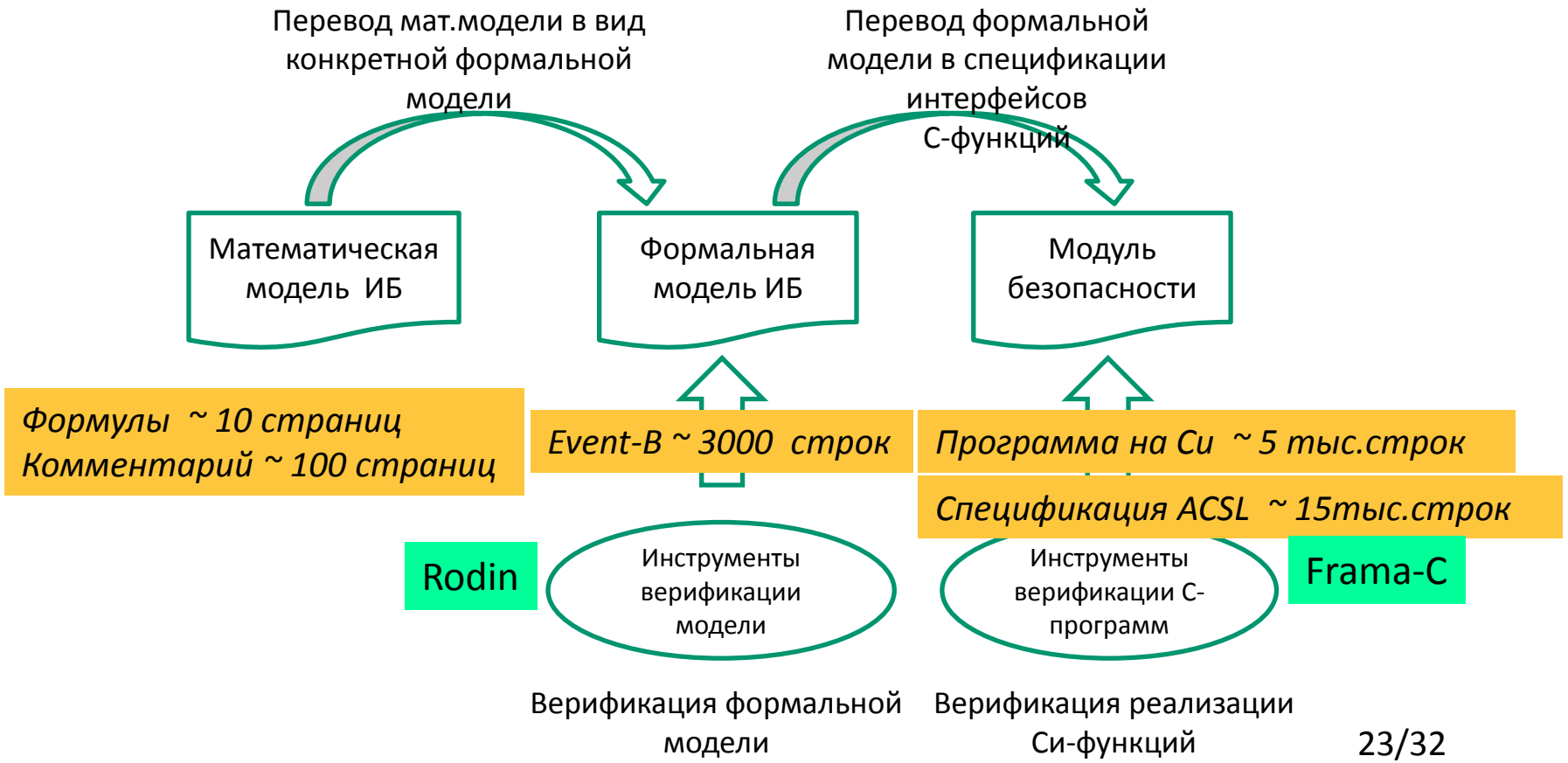
Истории практического внедрения (4)

Дедуктивная верификация операционных систем

Операционная система	Команда верификаторов
seL4 (~10Kloc)	NICTA (Австралия)
PikeOS (~10Kloc)	SYSGO (Германия)
Hyper-V (~300Kloc)	Verisoft/Microsoft Research

Истории практического внедрения (5)

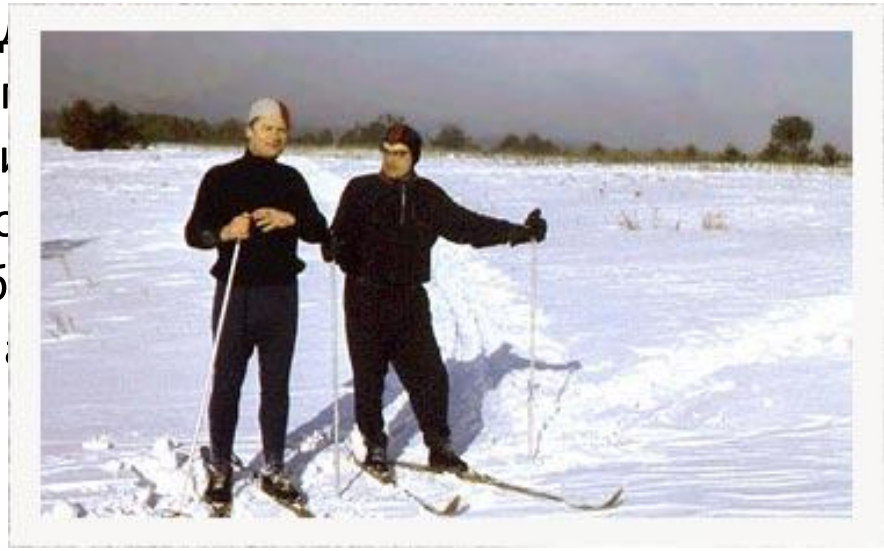
Дедуктивная верификация модуля безопасности ядра ОС (Linux Security Module - LSM)



О «дисциплине программирования»

И.В.Поттосин «Открытые системы», № 06, 1998
«Добротность программ и информационных потоков»:

Действительно, всякая «хорошая» технология программирования предполагает хорошо продуманную дисциплину разработки, надежно переходящую от спецификации прод
та
со
то
пр
сп
по



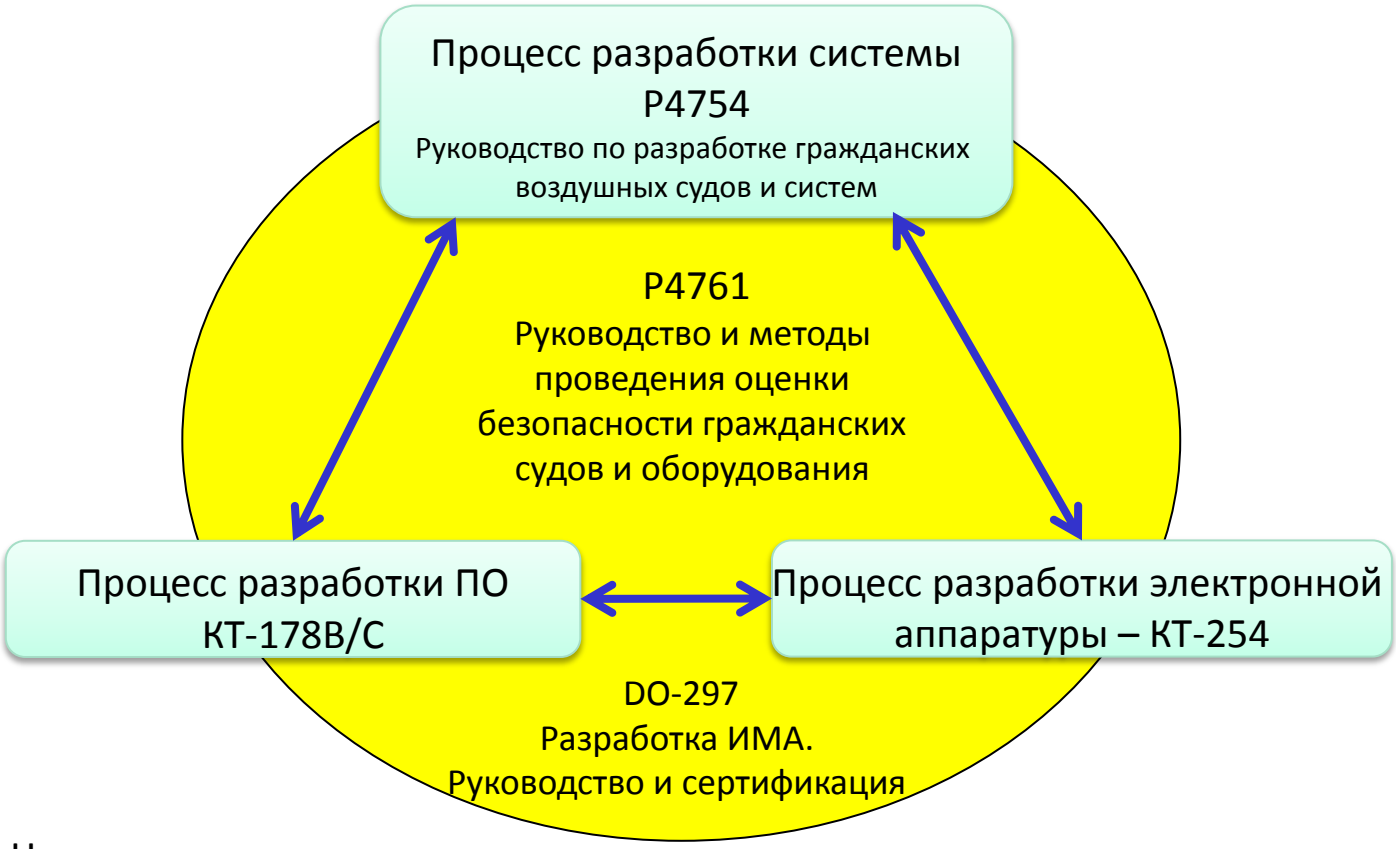
FACE - Future Airborne Capability Environment

FACE Vision

… технологически приемлемая **открытая архитектура**, **стандарты** и **бизнес-модель**, которые позволят обеспечить:

- Более низкие затраты
- **Надежную** архитектуру и **качество** программного обеспечения
- **Интероперабельность** компонентов от различных поставщиков
- **Переносимость** приложений

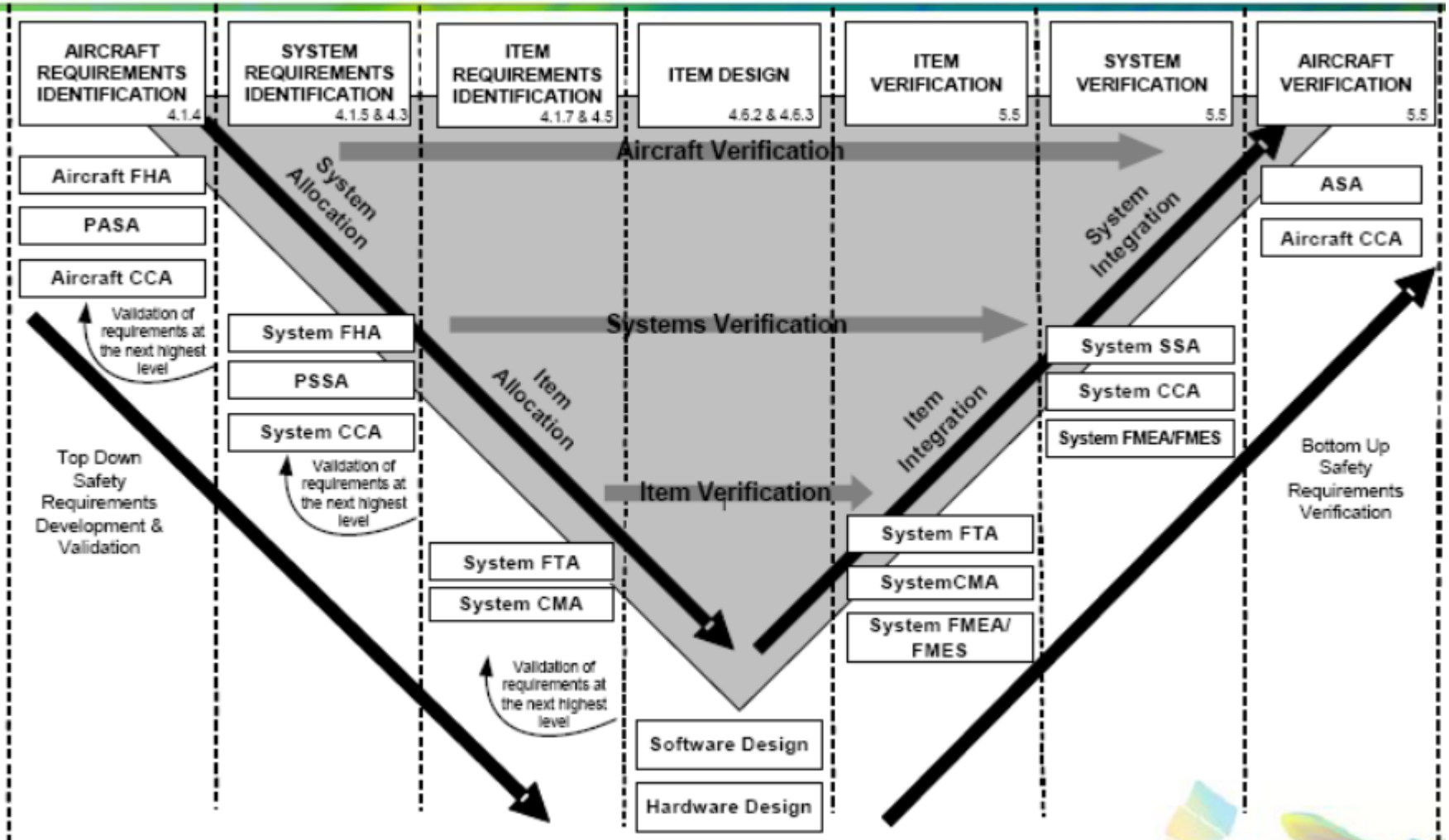
Интегрированная модульная авионика - ИМА



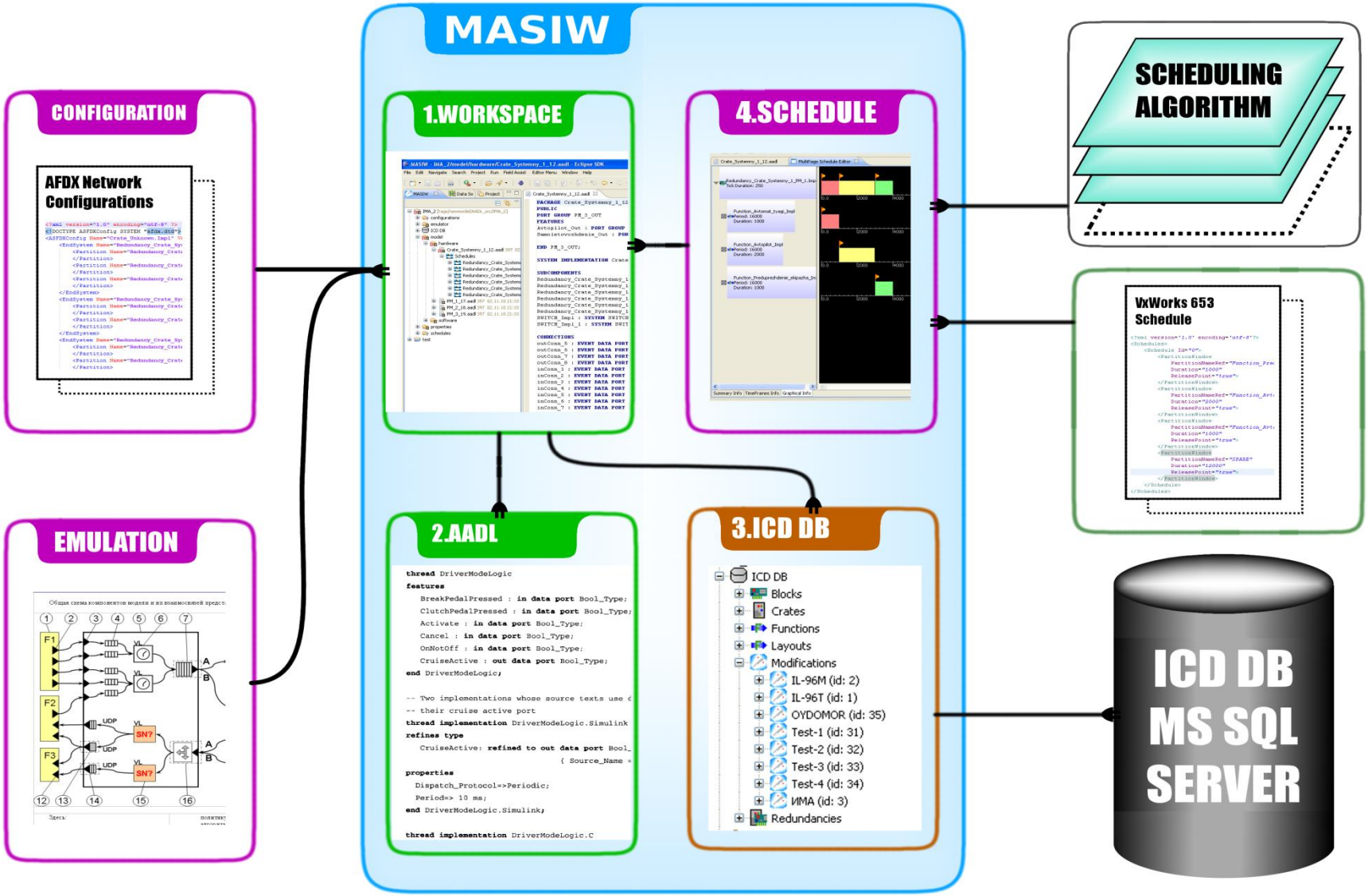
Академик Федосов Е.А.

Цитируется по
Koverninskiy Igor V., Kan Anna V. (GosNIIAS). Organization-technical methods for development of on-board equipment based on IMA // IMA Conference, Moscow, 2012.

ARP 4754A: Interactions of Requirements, Safety, and Development



Проектирование ИМА и системная интеграция



Добротность программных систем. Заключение (1)

- На всех направлениях есть проблемы и ожидается эволюционное развитие
- Ощущается необходимость в интеграции разных методов проектирования и анализа систем
- Сфера применения строгих стандартов на процессы разработки (клоны DO-178) постепенно расширяется
- Овладение этими стандартами дает конкурентное преимущество
- Хорошая новость для специалистов по формальным методам: Формальные методы нашли свое место в DO-178C/DO-333 "Formal Methods Supplement to DO-178C and DO-278A”
  (NB: addressing formal methods to complement (but not replace) testing).

Заключение (2)

- Big Data. Тенденция привлечения огромных вычислительных ресурсов для анализа/тестирования/верификации – это позволяет снизить плотность дефектов известных типов, но пока не дает интегрального качественного скачка добротности.
- По-прежнему – самыми дефицитными исходными данными являются спецификации требований всех видов
- Наиболее сложными представляются работы по
  - стыковке спецификации и верификации на стыках «программа-аппаратура» и «поведение системы – поведение компонентов»
  - методике композиции техник верификации в сложном программно-аппаратном комплексе

Заключение (3)

Тема добротных программных систем еще долго будет актуальной.

Спасибо!

Что еще посмотреть:

  - http://www.ispras.ru/groups/se/
- Публикации
  - http://www.ispras.ru/groups/se/publications.php
- Открытые проекты: UniTESK, OLVER, LDV, BLAST, CPAchecker, MASIW, Requality, Frama-C/Why3/Jessie
  - http://unitesk.ru
  - http://forge.ispras.ru
  - http://hardware.ispras.ru
  - http://linuxtesting.org
  - http://www.linuxbase.org/navigator/commons/welcome.php
  - http://www.ispras.ru/technologies/
  - http://sdat.ispras.ru/
  - http://syrcose.ispras.ru/
  - http://openisprasconf.ru

Добротные программные системы