АНО «Институт логики, когнитологии и развития личности»
ООО «Базальт СПО»
Институт Программных Систем РАН

Тринадцатая конференция «Свободное программное обеспечение в высшей школе»

Переславль, 26–28 января 2018 года

Сборник материалов конференции

Москва,
Basealt,
2018

Тринадцатая конференция «Свободное программное обеспечение в высшей школе»: Материалы конференции / Переславль, 26–28 января 2018 года. М.: Basealt, 2018. — 120 с. : ил.

В книге собраны материалы конференции, одобренные Программным комитетом тринадцатой конференции «Свободное программное обеспечение в высшей школе».

ISBN 978-5-9908979-6-0

Программа конференции

26 января, пятница

11.30-14.00 Заселение, обед, регистрация участников

Дневное заседание
14.00–18.00

14.00 А. Е. Новодворский. Открытие. Информация оргкомитета

14.10 И. В. Захаров. О сотрудничестве Базальт СПО с образовательными организациями высшего образования

14.20–14.50 Н. Н. Непейвода

Почему Шалыто прав насчёт открытых спецификаций? ... 8

14.50–15.20 Е. Р. Алексеев, Д. А. Лутошкин и др.

Использование компиляторов языка Фортран при решении вычислительных задач 9

15.20–15.50 Е. Р. Алексеев, Д. А. Лутошкин и др.

Разработка кроссплатформенных библиотек на языке Fortran и C++ построения графиков функций 15

15.50–16.10 Кофе-пауза

16.10–16.40 А. В. Бондарев и др.

Embox — студенческий проект в области системного программирования 20

16.40–17.10 В. В. Олоничев

Приборы фирмы Овен и свободное ПО 22

17.10–17.40 А. Г. Михеев

17.40–18.00 Г. Злобин и др. (зачитывает А. Новодворский)

18.00 Фуршет

27 января, суббота

09.30 Автобус из гостиницы

Утреннее заседание
10.00–13.00

10.00–10.30 С. М. Абрамов

10.30–11.00 Д. А. Костюк, А. А. Маркина

11.00–11.30 Д. А. Костюк и др.

11.30–12.00 Г. В. Курячий

12.00–12.20 Кофе-пауза

12.20–12.40 С. Голубев

Вечернее заседание
15.00–18.30

28 января, воскресенье

09.30 Автобус из гостиницы

Утреннее заседание
10.00–13.10

13.00 Автобус в Москву

Вне программы

Н. Н. Непейвода
Переславль-Залесский, Институт программных систем РАН, Вятский Государственный Университет

Проект: Проект РАН АААА-А16-116021760040-6

Почему Шалыто прав насчёт открытых спецификаций?

В 2017 году группа В. Л. Малых (ИПС РАН) по ходу работ над медицинской информационной системой поддержки принятия решений врачом получила интересные результаты. На базе данных о более чем миллионе клинических случаев было построено несколько вариантов распознавания ситуаций и генерации рекомендаций по лечению. Выяснилось любопытное явление. Меньше всего затрат программистского труда (практически на порядок) потребовала нейросеть, что компенсировалось затратами машинного времени и ресурсов для обучения и переобучения системы. Она в данном случае дала прекрасный коэффициент правильных решений (98%), но другие подходы в этом отношении оказались не намного хуже. Ещё одним преимуществом нейросети оказалось полное отсутствие необходимости кодировать информацию о медицинских инструкциях и «стандартах лечения», по сути дела она восстановила эти примитивные программы по прецедентам работы врачей, которые на 98% механически следуют им[1].

Эти эксперименты интересны также с точки зрения перспектив программирования. Намечается переход для многих задач к специализированным нейропроцессорам, имеется положительный опыт их применения и в суперкомпьютерах высшего ранга, и в компьютерах средней категории. В этом случае главным является не программа. а её спецификации, программирование полностью становится не фоннеймановским. При малейшем изменении аппаратной структуры нейрокомпьютера конкретные пользухи конкретных программ могут оказаться ненужными и скорее даже вредными. Так что передаваемыми становятся лишь спецификации.

[1] Поразительно, что в стандартах, диктуемых бюрократией и медико-фармацевтической «мафией», прописан именно такой процент допустимых нестандартных действий врача.

Поэтому я считаю, что замечания А. А. Шалыто по поводу разумности во многих случаях открытых спецификаций, а не открытого текста программ будут становиться всё более актуальными и что концепция свободного программного обеспечения скоро нащупает свои естественные границы применимости.

Попытаюсь обрисовать некоторые области, где именно открытые спецификации полезны.

Сложные нейросети, в особенности с аппаратной поддержкой, тем более после неизбежного появления нейросетей высших порядков вместо «глубоких».

Другие классы программ с нетривиальным самообучением.

Фотонные и квантовые процессоры.

Программы, эксплуатирующие большие кластеры специализированных процессоров.

Евгений Алексеев, Денис Лутошкин, Александр Огородов, Вячеслав Стародумов
Киров, Вятский Государственный Университет

Использование компиляторов языка Фортран при решении вычислительных задач

Аннотация

Представлен обзор основных возможностей современных компиляторов (gfortran, ifort) языка Фортрана при решении вычислительных задач

Фортран — первый язык программирования высокого уровня, ориентированный на решение вычислительных задач. Современный Фортран в первую очередь ориентирован на высокоэффективные вычисления.

Среди основных нововведений Фортрана можно выделить:

- поддержка длинных чисел (шестнадцатибайтные целые и вещественные тридцатидвухбайтные комплексные числа);
- динамические массивы переменной длины;

- все основные операции, процедуры и функции языка выполняются не только над скалярными величинами, но и над массивами, матрицами, а также их фрагментами — секциями[1];
- в современных компиляторах реализована конвейерная обработка операций и функций над массивами и секциями, благодаря чему исполняемый код программ обработки массивов и матриц, генерируемый компиляторами Фортрана, является самым быстрым;
- поддержка параллельных вычислений на уровне стандарта языка: параллельные циклы, комассивы;
- поддержка технологий параллельного программирования MPI, OpenMP;
- внедрение технологии автораспараллеливания в компиляторе ifort.

Многие экономические, инженерные и вычислительные задачи сводятся к задачам умножения матриц и решению систем линейных алгебраических уравнений большой размерности. Рассмотрим какие возможности предоставляют современные компиляторы для решения этих задач.

При анализе быстродействия авторами использовались следующие компиляторы языка Фортран:

- свободный компилятор gfortran (ver 7.2), поддерживающий большинство возможностей современных стандартов языка; компилятор gfortran портирован в Windows;
- проприетарный компилятор компании Intel — ifort, входящий в состав комплекса проприетарных программ Intel Parallel Studio; компилятор доступен для бесплатного использования в некоммерческих целях преподавателями, студентами и научными работниками;
- компилятор pgfortran, входящий в состав PGI Edition, которые поддерживает технологию CUDA; существует версия компиляторов семейства PGI — PGI Community Edition.

В очень многих вычислительных задачах необходимо выполнять матричные операции. Самой затратной матричной операцией явля-

[1] Секция массива — часть массива, которая определяется именем массива и границами изменения индексов (например, `x(:,4), y(1:15:3)`).

		Классический алгоритм, порядок выполнения циклов						Блочный алгоритм (размер блока 16)	matmul
	Время, t(c)	ijk	jik	kij	ikj	jki	kji		
1536	gfortran	20.5	23.65	52.9	49.9	2.22	3.77	2	0.5
	ifort	0.67	0.7	0.7	0.67	1.99	0.7	2.68	0.69
	pgfortran	1.86	2.05	2.05	1.9	1.9	2.08	5.64	0.91
2048	gfortran	91.12	89.84	219.5	222.48	5.26	8.84	4.75	1.16
	ifort	1.52	1.66	1.65	1.6	4.68	1.64	6.98	1.63
	pgfortran	4.46	4.86	4.91	4.54	4.54	4.51	61.78	2.19

Таблица 1: Быстродействие программ (время, с), реализующих классический и блочный алгоритмы на Фортране с использованием свободных и проприетарных компиляторов

ется операция умножение матриц[2]. Авторами была проанализирована эффективность умножения матриц на современном Фортране. В Фортране присутствует встроенная подпрограмма умножения матриц matmul, в последней версии gfortran (gfortran-7) функция значительно оптимизирована. В таблице 1 приведены результаты времени выполнения программ, реализующих классический и блочный алгоритмы умножения матриц и подпрограммы matmul.

При использовании проприетарных компиляторов, не важно в каком порядке записывать циклы в коде программ умножения матриц. Компиляторы ifort, pgfortran, достаточно хорошо оптимизируют исходный код. При использовании gfortran программист должен аккуратно писать код, записывая циклы в оптимальном для данного языка порядке. Подпрограмма умножения матриц, сформированная с помощью процедуры matmul и скомпилированная с помощью gfortran оказалась быстрее любого другого кода (даже быстрее блочного алгоритма и алгоритма Штрассена). Авторы сравнивали быстродействие программ умножения, написанных с использованием классических циклов на C и Фортране. Программы на Фортране работают на 10-12% быстрее. Использование matmul и свободного компилятора позволяет ускорить работу программы ещё в несколько раз. Поэтому именно matmul и новый компилятор gfortran-7 следует использовать для решения задач, сводящихся к умножению матриц.

[2]Частным случаем матричного умножения являются — умножение матрицы на вектор и скалярное произведение векторов.

Размерность системы	Компилятор Фортрана	Время счёта, с	Компилятор C(C++)	Время счёта, с
1000	gfortran	3.78	g++	0.89
	ifort	3.9	icpc	0.88
	pgfortran	1.44	pgc++	0.85
2000	gfortran	34.37	g++	10.10
	ifort	33.02	icpc	10.11
	pgfortran	16.13	pgc++	9.95

Таблица 2: Время счёта программ на Фортране и C(C++) решения СЛАУ методом Гаусса.

Размерность системы	Компилятор	Метод Зейделя			Метод простой итерации		
		k	1	2	k	1	2
10000	gfortran	12	3.58	3.58	37	10.4	2.32
	pgfortran		3.48	3.48		2.28	2.27
	ifort		3.5	3.5		7.5	2.19
12000	gfortran	13	6.31	6.31	37	18.2	3.34
	pgfortran		5.6	5.6		3.28	3.25
	ifort		5.64	5.64		12.8	3.17

Таблица 3: Время счёта программ решения СЛАУ итерационными методами.

Одной из тестовых задач анализа быстродействия является метод Гаусса решения СЛАУ. Было проведено сравнение быстродействие программ на C и Фортране (см табл. 2)

Таблица 2. Время счёта программ на Фортране и C(C++) решения СЛАУ методом Гаусса.

Как видно из результатов, программы на C++ работают значительно быстрее, что является следствием большого количества циклов в алгоритме.

При написании последовательных приложений решения задач вычислительной математики использование конвейерных операций позволяет значительно повысить быстродействие приложений. Авторами было проведено тестирование метода Зейделя и метода простой итерации (см. табл. 3, k — количество итераций, 1 — программа с использованием циклов, 2 — программа с использованием конвейерных операций).

Значительное ускорение наблюдается, если удаётся написать код с использованием только матричных подпрограмм и операторов (без циклов). При совместном использовании обычных циклов и матрич-

Размерность системы	Компилятор	Метод Зейделя		Метод простой итерации
		обычный	усовершенствованный	
10000	gfortran	3.57	2.61	2.32
	pgfortran	3.48	2.54	2.27
	ifort	3.5	2.48	2.19
12000	gfortran	6.26	4.01	3.34
	pgfortran	5.6	3.78	3.25
	ifort	5.62	3.76	3.17

Таблица 4: Быстродействие метода Зейделя, усовершенствованного метода Зейделя и метода простой итерации.

ных операции ускорение работы программы оказывается незначительным.

Коммерческий проприетарный компилятор pgfortran генерирует быстродействующее параллельное приложение при написании программы без использования современных матричных операторов и подпрограмм.

Если внимательно проанализировать результаты, представленные в табл. 3, можно заметить, что, не смотря на то, что метод Зейделя требует значительно меньшее количество итераций, время счёта в методе простой итерации значительно меньше, чем в методе Зейделя. Это связано с тем, что в методе Зейделя невозможно избавиться от последовательных циклов и перейти полностью к матричным (конвейерным) операциям и подпрограммам. Алгоритм сложения является рекуррентным, что не позволяет избавиться от последовательных циклов.

Однако можно ускорить алгоритм Зейделя при написании кода на Фортране, используя операцию матричного умножения matmul. На каждой итерации можно предварительно умножать фрагмент матрицы на вектор x. Это позволит сократить время счёта. Платой за это будет память для хранения второй матрицы размерности $n \times n$. Ниже приведены результаты сравнения быстродействия обычного и усовершенствованного метода Зейделя и метода простой итерации.

Аналогичным образом можно ускорять и другие алгоритмы вычислительной математики, в которых используются матричные операции.

Одним из новейших методов построения параллельных программ является технология автораспараллеливания, полноценная поддерж-

размерность	matmul	Классический алгоритм	Блочный алгоритм
1000	1.4	1.4	2
2000	2.2	2.2	4
3000	2.3	2.3	5.2
4000	2.4	2.4	7.1
5000	2.5	2.5	6.8

Таблица 5: Ускорение в программах умножения матриц с опцией автораспараллеливания (4 ядра).

	Количество процессорных узлов			
	1	2	4	8
Метод простой итерации	1	1.93	3.68	5.6
Метод Зейделя	1	1.3	1.64	1.75

Таблица 6: Ускорение при решения СЛАУ размерности 7000 и с использованием технологии комассивов на компьютерном кластере ВятГУ.

ка которой реализована в только проприетарных компиляторах Intel посредством опций командной строки -parallel.

В таблице 5 приведены результаты применения опции автораспараллеливания к последовательным программам умножения матриц (matmul, классический и блочный алгоритмы).

Кроме поддерживаемых в других языках программирования технологий параллельного программирования OpenMP и MPI, в новый стандарт языка Fortran-2008 включены встроенные средства распараллеливания Co-Arrays (комассивы), которые могут быть реализованы в системах как с распределенной, так и с общей памятью.

С помощью комассивов были разработаны параллельные реализации методов Зейделя и простой итерации. Оценка времени работы программ приведена в таблице 6.

Современные компиляторы Фортрана позволяют разрабатывать высокоэффективный код решения вычислительных задач с использованием современных технологий. Самые новые технологии параллельного программирования полноценно поддерживаются на сегодняшний день только проприетарными компиляторами. При разработке последовательных и параллельных приложений имеет смысл использовать последние версии свободного компилятора gfortran, который генерирует самый быстрый код при работе с матрицами.

Евгений Алексеев, Денис Лутошкин, Вячеслав Стародумов
Киров, ВятГУ

Проект: Plotter `https://github.com/LibPlotter/`

Разработка кроссплатформенных библиотек на языке Fortran и C++ построения графиков функций

Аннотация

Представлены разработанные авторами кроссплатформенные библиотеки построения графиков функций на языках C++ и Fortran.

Введение

Наиболее часто используемыми языками разработки быстроработающих приложений, решающих вычислительные задачи, являются C++ и Фортран. В вычислительных приложениях могут использоваться блоки, предназначенные для рисования двух- и трёхмерных графиков. Данные блоки повышают сложность кода, что снижает его производительность и повышает сложность отладки. Поэтому, при решении инженерных задач и задач вычислительной математики, где нельзя обойтись без построения графики, используются сторонние графические библиотеки.

Целью данной работы являлась разработка доступных, кроссплатформенных, свободных и простых для пользователя программных комплексов на языках C/C++ и Fortran построения графиков функций.

Обзор существующих графических библиотек

На сегодняшний день существует несколько графических библиотек и программ, из них можно выделить: PGPlot[1], PLPlot[2], Dislin[3], GNUPlot[4].

Среди особенностей существующих библиотек можно выделить:

- отсутствие поддержки 3D графики в PGPLot;

[1] `http://www.astro.caltech.edu/tjp/pgplot/`
[2] `http://plplot.sourceforge.net/`
[3] `http://www.mps.mpg.de/dislin`
[4] `http://www.gnuplot.info/`

- плохая портируемость PLPlot в операционные системы отличные от семейства Linux;
- коммерческая (проприетарная) библиотека Dislin генерирует только статические изображения (фигуру нельзя повернуть или изменить масштаб);
- Dislin бесплатна только для некоммерческого использования;
- при использовании свободной графической библиотеки GNUPlot необходимо иметь её целиком на компьютере.

Исходя из вышеизложенного возникает необходимость разработки свободной, переносимой кроссплатформенной графической библиотеки.

Разработка библиотеки Plotter

Была поставлена задача разработки универсальных программных комплексов построения графиков на языках программирования Fortran и C++. Разрабатываемая библиотека должна удовлетворять следующим условиям:

- должна быть достаточно простой в применении, для её использования следует просто подключить к программе и передать данные в команду вызова;
- пользователь должен иметь возможность взаимодействовать с изображением;
- должна иметь возможность изображать все основные объекты: графики и поверхности.

При разработке программных комплексов была использована графическая библиотека OpenGL и свободная кроссплатформенная реализация библиотеки GLUT — FreeGLUT. Были разработаны свободные и кроссплатформенные библиотеки Plotter построения двух- и трёхмерных графиков. Библиотеки представляет собой набор функций. Для использования библиотек достаточно использовать главную функцию `Draw()` (C++) и `DrawGraph()` (Fortran).

Пользователь с помощью библиотеки Plotter может строить графики по точкам, графики функций в двумерном и трёхмерном пространстве, строить плоскости в трёхмерном пространстве. При отрисовке графиков проводится автоматическая проверка на допустимость значения. При недопустимом значении координаты точки, ко-

ордината обнуляется, а сама точка помечается исключённой. Пользователь при передаче данных может дополнительно передать информацию о том, какие точки являются исключёнными или при нахождении в каких интервалах считать точку исключённой.

Пользователь может управлять положением камеры: смещать камеру по осям, вращать её, масштабировать изображение. Изображение учитывает соотношение сторон окна. Пользователь также может изменять режим отображения объектов, что осуществляется через обработчик нажатий на клавиши клавиатуры. Пользователь может изменить некоторые параметры перед стартом отображения. Это осуществляется посредством задания параметров отображения.

Для подключения библиотеки Plotter на языке Fortran необходимо добавить в программу строку use plotter, на языке C++ следует в файле исходного кода подключить непосредственно библиотеку: #include "Plotter.h". Для её использования пользователю достаточно знать одну функцию Draw() (C++) и DrawGraph()(Fortran) и вызывать с различными параметрами. В зависимости от параметров меняется изображение.

На сегодняшний день тестирование библиотек проведено в ОС Ubuntu Linux, Linux Mint с использованием следующих компиляторов: Intel Fortran Compiler, Intel C/C++ Compiler, GNU GCC и ОС Windows (Intel Fortran Compiler, Intel C/C++ Compiler, GNU GCC, Microsoft (R) C/C++ Optimizing Compiler). Планируется адаптация библиотек для ОС Debian, Альт Образование, Simply Linux.

Возможности работы подпрограммы DrawGraph и Draw

1. рисование графика одной функции;
2. рисование графиков нескольких функций. Пример графиков представлен на рисунке 1;

1. рисование графика функции с интервалами исключения;
2. рисование графиков нескольких функций с интервалами исключения;
3. рисование графика по точкам;
4. построение графика по точкам с исключёнными точками;
5. построение графика по точкам с интервалами исключения;
6. построение графиков по нескольким наборам точек;

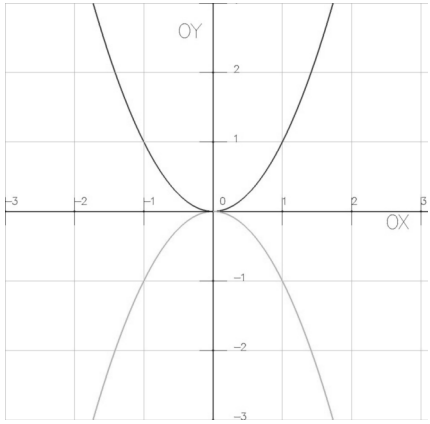


Рис. 1: Пример графиков

7. построение графиков по нескольким наборам точек с исключёнными точками. Пример графиков представлен на рисунке 2;
8. построение графиков по нескольким наборам точек с интервалами исключения ;
9. построение поверхности. Пример поверхности представлен на рисунке 3;
10. построение нескольких поверхностей.

Библиотеки можно загрузить по адресу: `https://github.com/LibPlotter/`.

Библиотека Plotter используется в учебной деятельности ВятГУ при обучении бакалавров и магистрантов направления «Прикладная математика и информатика».

Данный программный комплекс можно использовать в научной деятельности: при решении инженерных задач, задач вычислительной математики, при разработке вычислительных приложений на языках C++ и Fortran.

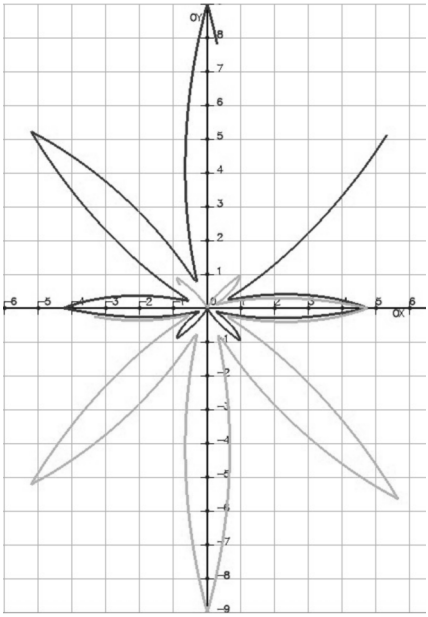


Рис. 2: Пример графиков с исключенными точками

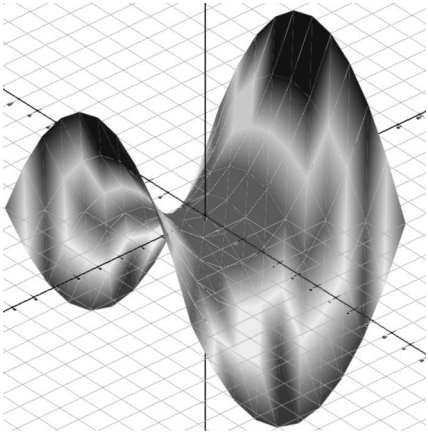


Рис. 3: Пример поверхности

Бондарев Антон Владимирович, Калмук Александр Игоревич, Дерюгин Денис Евгеньевич
Санкт-Петербург, ООО «Ембокс», СПбГУ
https://github.com/embox/embox

Embox — студенческий проект в области системного программирования

Аннотация

Embox — открытый проект по созданию конфигурируемой операционной системы для встроенных систем. Проект начинался «с нуля» как маленькая оболочка для отладки аппаратуры и драйверов. Вскоре наработки были выложены в open-source, параллельно был запущен студенческий проект на кафедре системного программирования СПбГУ. Проект является хорошей базой для отработки практических навыков в системном программировании и платформой для вовлечения студентов в мир open-source.

Одним из основных недостатков современного IT-образования является отрыв теоретического обучения от промышленных задач. Это отметил Бьерн Страуструп в своей статье «What should we teach new software developers? Why?» [1]. В частности, он пишет о том, что учебные задачи совсем не похожи на промышленные, а глубокое знание теории вовсе не гарантирует способность применить полученные знания на практике.

Для практических занятий по курсу построения операционных систем и системного ПО в ведущих IT-вузах, как правило, используют существующие ОС, так или иначе адаптированные для обучения или изначально разработанные для применения в академических целях. Самая известная ОС для обучения — это, конечно, MINIX Эндрю Таненбаума [2]. Но существуют и другие, например, операционная система Nachos [3], используемая в университете Беркли и предназначенная именно для обучения студентов. В том же университете Беркли для исследований в области ОС и сетевых технологий используют различные версии BSD. В MIT используется упрощенная версия UNIX Xv6 [4].

Все приведённые примеры, конечно же, являются проектами с открытым исходным кодом. В то же время эти проекты можно разделить на две категории: первые используются исключительно в обра-

зовательных целях, вторые имеют попытки применения в промышленности. Существенную разницу между этими двумя категориями проектов можно увидеть на истории развития проекта MINIX. Автор этой ОС, Эндрю Таненбаум, отказывался принимать патчи от сторонних разработчиков для первых двух версий системы. Мотивировалось это тем, что проект является исключительно образовательным. Третья версия была доработана для использования в промышленных системах и сейчас является полноценным open-source проектом с сообществом, которое его развивает и поддерживает. Таким образом, студенты стали более вовлеченными в проект, они уже не только читают код, но и могут предлагать его развитие, тем самым участвуя в команде распределённого промышленного проекта. Кроме того, на базе проекта могут проводиться и научно-исследовательские работы.

Таким образом, хотя проекты, ориентированные только для обучения, необходимы для образования, но в тоже время без участия студентов в промышленных проектах очень трудно добиться качественно подготовленного специалиста.

Embox — открытый проект по созданию конфигурируемой операционной системы для встроенных систем. Изначально проект был закрытым и представлял из себя внутреннюю разработку в виде простой оболочки для отладки аппаратуры и драйверов. С ростом функциональности проект обрёл черты не большой, но полноценной ОС для встроенных устройств. Поскольку в проекте удалось сохранить простую и понятную структуру, а также небольшой объем исходного кода, то было принято решение использовать его для улучшения практических навыков студентов Мат-Меха СПбГУ. Причём основной акцент был сделан именно на получении законченного, пусть и небольшого, результата. То есть на базе общего проекта Embox организовываются студенческие проекты с различными целями, например, одним из первых подпроектов была адаптация Embox для робоконструкторов lego mindstorm NXT 2. При этом необходимо было не только портировать Embox на данную платформу, но и организовать работу по протоколу bluetooth и реализовать различные алгоритмы управления (например, стабилизацию обратного маятника).

Участие в «живом», развивающемся проекте позволяло студентам более полно почувствовать все стадии разработки программного обеспечения. При этом Embox за счёт своей изначальной простоты сохранял свойства обучающего проекта с относительно низким порогом вхождения.

Литература

[1] https://dl.acm.org/citation.cfm?doid=1629175.1629192

[2] http://www.minix3.org/

[3] http://www.cs.washington.edu/homes/tom/nachos/

[4] http://pdos.csail.mit.edu/6.828/2011/xv6.html

Василий Олоничев
Кострома, Костромской государственный университет

Проект: Библиотека с реализацией протокола Овен для ОС GNU/Linux
https://github.com/basv0/owenlib

Приборы фирмы Овен и свободное ПО

Аннотация

В статье кратко описывается использование свободного ПО в Костромском государственном университете на кафедре автоматики и микропроцессорной техники. Особое внимание уделено проблемам сопряжения приборов фирмы Овен с компьютерами по сети RS-485 с использованием протокола Овен, а также проблемам реализации ПО для старших моделей ПЛК с ОС Linux. Приводятся причины, по которым ПО для ОС Linux должно выкладываться в свободный доступ и дается ссылка на GitHub, где размещена библиотека owenlib с реализацией протокола Овен для ОС Linux.

На кафедре автоматики и микропроцессорной техники Костромского государственного университета при обучении студентов по специальности «Управление в технических системах» широко используется операционная система GNU/Linux и входящее в ее состав свободное ПО, а именно: компилятор gcc, СУБД PostgeSQL, Octave, научная библиотека gsl и т. д. Объясняется это двумя причинами. Во-первых, специалистам в области автоматизации технологических процессов и производств в своей профессиональной деятельности обязательно придётся иметь дело с Unix-подобными операционными системами, в частности, с Linux, который установлен на старших моделях программируемых логических контроллеров [1] и [2] и на многочисленных одноплатных мини-ЭВМ [3] , которые сегодня так популярны в «малой автоматике», и, наконец, с промышленными операционными система

жесткого реального времени, такими как QNX. Во-вторых, работать с ОС GNU/Linux в целом удобней, чем с ОС Windows, а использование свободного ПО значительно облегчает администрирование учебных классов.

Одновременно в учебном процессе, включая курсовое и дипломное проектирование, широко используются приборы для промышленной автоматизации отечественной фирмы Овен, в первую очередь терморегуляторы и программируемые логические контроллеры. Данные приборы имеют возможность обмениваться данными как между собой, так и обычными компьютерами и одноплатными мини-ЭВМ по линии связи RS-485 с использованием своего собственного протокола, который так и называется, протокол Овен. В настоящее время наряду с данным протоколом возможно использование протокола Modbus. Но Modbus поддерживает не все функции и, кроме того, в эксплуатации имеется достаточно много старых приборов без Modbus. Поэтому поддержка протокола Овен по-прежнему актуальна. Фирма Овен выложила в открытый доступ, как описание своего протокола, так и библиотеку с его реализацией только для ОС Windows.

Поэтому мне пришлось заняться реализацией данного протокола на языке C для Linux. Специалисты фирмы Овен помогли тем, что выслали исходные тексты для кодирования данных и команд на языке Java. Кстати, соответствующий модуль предлагаемой ниже библиотеки так и остался подстрочником с Java. А вскоре после этого на форуме Овен появился исходный текст этих функций на языке C [4]. Для обмена данными только этих функций недостаточно, требуется набор функций, образующих пользовательский интерфейс и функции для обмена данными через последовательный порт. Все это было выполнено как часть проекта, реализующего службу для Linux, которая с заданным интервалом времени опрашивает датчики температуры, размещенные во всех лабораториях кафедры и записывает их в базу данных. Эта программа была использована в нескольких дипломных проектах, посвященных созданию клиентской части как для Windows, так и Linux с возможность фильтрации и усреднения данных, а также построения графиков.

Затем подошла очередь программируемого логического контроллера Овен ПЛК-308, который появился в 2011 году и имел предустановленную ОС Linux. Но к сожалению фирма Овен отказала в предоставлении какой-либо документации по прямой работе с Linux в данном приборе: [5] «Linux Вы не увидите. Linux в нем для пользователя

не доступен. А писать программу можете только в CoDeSys». Ситуация не изменилась и сегодня [6]. Но свободное ПО, на то и свободное ПО, потому что для него подобные препятствия, в отличие от проприетарного, всегда преодолимы. Легко выяснилось, что в ПЛК-308 и подобных ему, запущена служба sshd, а логин и пароль имелись некоторое время на официальном сайте Овен, правда вскоре они были удалены. После достаточно долгого перебора вариантов был найден toolchain ronetix-arm-linux-uclibc-4.1.2 [7], позволяющий создавать исполняемые бинарные файлы для данной серии приборов. В результате был создан набор программ, взаимодействующих между собой через разделяемую память с синхронизацией на семафорах System V. К одной из программ была статически прилинкована откомпилированная при помощи этого же toolchain научная библиотека gsl, а одна из программ включала в себя механически скопированные из предыдущего проекта модули для обмена данными по протоколу Овен. Данный программный комплекс предназначен для адаптивной системы управления нестационарным объектами в пространстве состояний [8]. Этот опыт показывает, что после нахождения подходящего toolchain, написание ПО для подобного рода приборов ничем не отличается от написания ПО для настольных компьютеров и одноплатных мини-ЭВМ.

Однако полная реализация протокола Овен для Linux по прежнему оставалась только частью перечисленных выше проектов и не была оформлена в виде отдельной библиотеки. Ситуация сдвинулась с мёртвой точки, когда в учебный план была введена дисциплина «Иерархические микропроцессорные системы управления», для лабораторного практикума которой и потребовалась данная библиотека: ведь студентов надо сразу приучать к грамотному использованию ПО, которое исключает копирование больших и непонятных участков кода, которые хранятся на кафедральном файлообменнике. Так и появилась первая версия библиотеки, которая после некоторых доработок выложена на GitHub [9] под лицензией GPL.

Это, несомненно, значительное упущение, что данная библиотека не была создана и выложена для всеобщего доступа немедленно после самой первой реализации в виде набора модулей. Основной причиной было нежелание выставлять на обозрение сырой и неоптимизированный код. Однако выкладывать код следует всегда по следующим причинам:

1. На производстве имеется множество грамотных инженеров, которые при наличии подобной библиотеки, пусть даже и неполной, смогли бы перейти на ОС GNU/Linux. Их квалификации вполне достаточно для написания программ с использованием QtCreator, но недостаточно, чтобы по обрывочным и неполным данным написать библиотеку — это не их профессия.
2. К открытому проекту могут присоединиться более грамотные разработчики, которые выполнят рефакторинг кода и расширят набор интерфейсных функций.

В настоящее время библиотека была скомпилирована и установлена на настольные компьютеры с ОС CentOS и одноплатные мини-ЭВМ CubieBoard с ОС Debian GNU/Linux и используется для сбора данных и оперативного изменения настроек на терморегуляторах ТРМ-151, ТРМ-201 и модулях аналогового ввода МВА-8.

На GitHub, помимо самой библиотеки представлены примеры ее использования, как в консольном приложении, так и в графическом — проект QtCreator.

Как уже было отмечено, библиотека еще неполная и нуждается в оптимизации и возможно в ревизии пользовательского интерфейса.

Литература

[1] Mintchel, Gary, Linux Lives on a PLC, https://www.automationworld.com/article/technologies/plcs-pacs/linux-lives-plc

[2] Linux Programmable Fieldbus Controller, http://www.wago.us/products/components-for-automation/modular-io-system-ip-20-750753-series/plc/linux/index-2.jsp

[3] Brown, Eric, http://linuxgizmos.com/ringing-in-2017-with-90-hacker-friendly-single-board-computers/

[4] Реализация протокола Овен, http://www.owen.ru/forum/showthread.php?t=7311&p=35715#post35715

[5] ПЛК308, http://www.owen.ru/forum/showthread.php?t=10394&highlight=Linux

[6] ПЛК304 и ПЛК323, http://www.owen.ru/forum/showthread.php?t=23607&highlight=Linux

[7] Index of /toolchains/arm, http://download.ronetix.info/toolchains/arm/

[8] М. А. Смирнов, В. В. Олоничев, Б. А. Староверов, Особенности разработки программного обеспечения для Linux-контроллеров, Сборник материалов Международной научно-практической конференции ТМПА-2013 «Инструменты и методы анализа программ (Tools & Methods of Program Analysis)». КГТУ, 2013

[9] Олоничев В. В., owenlib, `https://github.com/basv0/owenlib`

Михеев Андрей Геннадьевич
Москва, НИТУ МИСиС, ООО «Процессные технологии»
Проект: RunaWFE `http://runawfe.org/`

Работа с внешними данными в курсе процессной автоматизации предприятий на основе свободной системы RunaWFE

Аннотация

В докладе предложен подход к реализации взаимодействия экземпляров бизнес-процессов с данными, позволяющий студентам, не обладающим знаниями теории баз данных, разрабатывать бизнес-процессы, взаимодействующие с таблицами, расположенными на листах документов формата MS Excel.

Обучение процессному управлению

Внедрение систем управления бизнес-процессами (СУБП) на предприятиях привело к тому, что у бизнеса появилась потребность в аналитиках, обладающих процессным мышлением. Таких специалистов в настоящее время вузы готовят в рамках специальности «бизнес-информатика». Для этой специальности в НИТУ МИСиС был разработан курс обучения процессному управлению на свободном программном обеспечении.

На занятиях студенты изучают теорию исполнимых бизнес-процессов, графические нотации описания бизнес-процессов, основные компоненты типичных СУБП и получают практический опыт разработки и исполнения простейших бизнес-процессов. Во время обучения изучаются и закрепляются на практике вопросы работы с переменными бизнес-процессов, правилами выбора маршрута движения точек управления, возможности задания сроков выполнения заданий, а

также взаимодействие с внешними данными. Разработанные бизнес-процессы студенты исполняют под разными ролями в программной среде.

Студенты, обучающиеся по специальности «бизнес-информатика», как правило, не обладают знаниями теории баз данных, необходимыми для практической работы. Поэтому потребовалось создать для них упрощенной механизм работы с данными, позволяющий проводить отдельные операции (INSERT, SELECT, UPDATE, DELETE) с реляционными таблицами, расположенными на листах документов формата MS Excel, которые можно смотреть при помощи приложения LibreOffice Calc.

Такой механизм [1] был реализован в свободной системе управления бизнес-процессами предприятия RunaWFE [2]. Однако, оказалось, что для реализации учебных работ также требуется механизм, поддерживающий выполнение нескольких действий бизнес-процесса в рамках одной транзакции. В настоящем докладе предлагается упрощенный механизм транзакций, основанный на особенностях процессного управления, легкий для освоения студентами, обучающимися по специальности «бизнес-информатика»

Этот механизм запланирован к выпуску в следующей версии системы RunaWFE.

Предлагаемый механизм упрощенной реализации транзакций в бизнес-процессе

Для ботов (автоматических исполнителей заданий бизнес-процесса) предлагается добавить признак «Транзакционный», параметром которого будет количество минут таймаута.

Для внутреннего подпроцесса предлагается добавить признак «Транзакция». Если значение признака — False, то элемент является обычным внутренним подпроцессом-композицией. Если значение признака — True, то поведение элемента изменяется следующим образом:

1. Граница подпроцесса-композиции рисуется двойной линией (см. Рис. 1).
2. Если в подпроцессе-транзакции используется бот, в настройках которого признак «Транзакционный» принимает значение True, то после начала выполнения этим ботом какого-либо задания

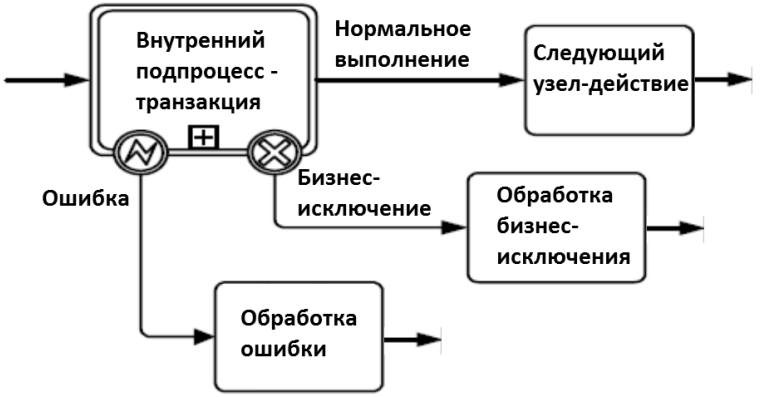


Рис. 1: Изображение внутреннего подпроцесса-транзакции

подпроцесса-транзакции данного экземпляра бизнес-процесса и до выхода всех точек управления из подпроцесса-транзакции или срабатывания таймаута, установленного для данного бота, бот выполняет задания только из этого подпроцесса, остальные задания он игнорирует.

3. Если сработал таймаут, то генерируется событие-ошибка, которое можно «поймать», присоединив к подпроцессу соответствующее событие. Блокирование остальных заданий бота при этом снимается.

4. Если все точки управления ушли из подпроцесса-композиции, то блокирование заданий бота снимается.

То есть, в режиме «Транзакция» бот может обрабатывать группы своих заданий только последовательно, друг за другом. Таким образом исключается одновременный доступ нескольких подпроцессов-транзакций к одним и тем же данным.

Данный механизм позволят избежать конфликта одновременного доступа к данным и не потерять в производительности потому, что часть заданий бизнес-процесса выполняется людьми, которые делают это во много раз медленнее, чем компьютерные приложения. Вследствие этого синхронизации исполнения определенных заданий

ботов не приведет к существенным задержкам выполнения бизнес-процессов.

Функциональность предполагается применять, в частности, для бота работы с внешним хранилищем [3], если, например, требуется прочитать записи из таблицы приказов, найти максимальный номер записи приказа определенного вида, прибавить к нему единицу, и на основе этого номера добавить в таблицу новую запись.

Система RunaWFE

Система RunaWFE является свободной СУБП с открытым кодом. В 2016 году Система RunaWFE внесена в Единый реестр российских программ для электронных вычислительных машин и баз данных под номером 951 по классу ПО «системы управления процессами организации». Команда разработчиков находится в Москве, к разработчикам легко обратиться с вопросами, предложениями и пожеланиями.

Литература

[1] Михеев А. Г. Обучение процессному управлению: Работа со слоем данных // Одиннадцатая конференция Свободное программное обеспечение в высшей школе. Тезисы докладов, М.: Альт Линукс, 2016

[2] Михеев А. Г., Орлов М. В. Система управления бизнес-процессами и административными регламентами. // Программные продукты и системы, № 3, 2011

[3] Михеев А. Г. Методика обучения работе с данными в курсе процессного управления // Открытое образование, 2016 — № 6 С. 4 — 8

[4] Ссылка на сайт проекта RunaWFE: `http://wf.runa.ru/`, `http://runawfe.org/`

Григорий Злобин, Петр Рыковский, Александр Чмыхало
Львов, Украина, Львовский национальный университет имени Ивана Франко
zlobingg@gmail.com

Использование свободного программного обеспечения в учебном курсе «Основы программирования»

Аннотация

В докладе рассмотрено использование СПО в лекционном курсе «Основы программирования». Проведено сравнение оболочек для структурного программирования на языке Си.

Предистория

В ноябре 2012г. ректор ЛНУ имени Ивана Франко получил от Microsoft Ukraine вежливое письмо с предложением составить план приобретения лицензий на продукты Microsoft в 2012-2013 гг. Поскольку в бюджете факультета электроники не было средств на приобретение лицензий проприетарного ПО, то после обсуждения ситуации было принято решение о немедленном переводе всех учебных лабораторий на СПО. Однако, после заявления лектора, который читал курсы «Алгоритмизация и программирование», «Объектно-ориентированное программирование» с использованием иключительно проприетарного ПО в ОС Microsoft Windows, о «гибели дела его жизни» Ученый Совет факультета выделил 1000 у.е. на приобретение подписки Dream Spark на факультет. Но из-за ошибок в апликационных формах факультет так и не получил этой подписки. Впоследствии удалось оформить бесплатную подписку Dream Spark на кафедру радиофизики и КТ. Благодаря этому в трех учебных лабораториях кафедры радиофизики и КТ появились лицензионные версии Microsoft Windows 7, а во всех других лабораториях факультета была установлена ОС Linux. Это привело к тому, что студенты не могли использовать в лабораториях с ОС Linux тех средств разработки, которые они использовали при изучении курсов «Алгоритмизация и программирование» и «Объектно-ориентированное программирование».

В 2017/18 учебном году на факультете открыли новую специальность «Информационные технологии». Для студентов этой специальности в первом семестре читается лекционный курс «Основы про-

граммирования». Для изучения начинающими программирования на языке Си надо выбрать простую оболочку, работу с которой очень легко освоить. Рассмотрим следующую таблицу:

Оболочка	Операционные системы	Состояние разработки
Turbo C	DOS	устарело
Borland C/C++	DOS, Windows	устарело
Kuzya	OS X, Windows, Linux	развивается
Geany	OS X, Windows, Linux	развивается
Anjuta	Linux	развивается
Atom	OS X, Windows, Linux	развивается
Code::Blocks	OS X, Windows, Linux	развивается
Eclipse	OS X, Windows, Linux	развивается
Visual Studio Code	OS X, Windows, Linux	развивается
QtCreator	OS X, Windows, Linux	развивается

В первых двух строках таблицы упомянуты Turbo C и Borland C/C++, которые хорошо описаны в учебной литературе, но они рассчитаны на работу только в MS DOS и Windows. Остановим наш выбор на оболочках Kuzya, Geany и Code :: Blocks.

Kuzya — это максимально простая оболочка для обучения. Именно на простоту делался основной акцент. Например, в данной оболочке отсутствует менеджер проектов и вообще понятие «проект». Робота проходит с одним файлом, который содержит текст программы. Сразу после запуска IDE Kuzya, студент, не задумываясь над порядком работы с оболочкой, может начать вводить код программы. После сохранения он может скомпилировать и запустить его. Для удобного и быстрого набора кода в главном меню находятся шаблоны синтаксических конструкций языков C/C++ и Pascal. Нажимая левой клавишей мыши на выбранном шаблоне можно вставить соответствующий текст в место, где расположен курсор. Изменение языка перевода кода автоматически приводит к переводу шаблонов. Также поддерживаются подсветка текста программы, которая позволяет легко ориентироваться в нём. Минимум возможностей данной оболочки (оставлены только полезные или необходимые из них, с точки зрения обучения) позволяют максимально сконцентрироваться на изучении языков программирования. Также был создан графический движок,

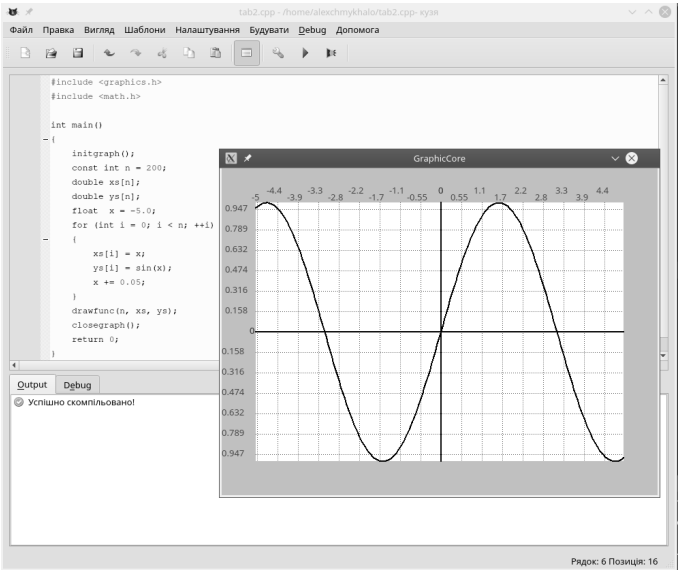


Рис. 1: Оболочка Kuzya

который позволяет отображать графические примитивы, используя библиотеку в стиле Borland C++. Язык диалога — английский, украинский, белорусский, русский.

Geany — среда разработки программного обеспечения, написанная с использованием библиотеки GTK+. Доступна для следующих операционных систем: BSD, Linux, Mac OS X, Solaris и Windows. Geany распространяется согласно GNU General Public License. Geany не включает в свой состав компилятор. Для создания исполняемого кода используется GNU Compiler Collection или, при необходимости, любой другой компилятор.

Code::Blocks — свободная кроссплатформенная среда разработки. Code::Blocks написана на C++ и использует библиотеку wxWidgets. Имея открытую архитектуру, может масштабироваться за счёт подключаемых модулей. Поддерживает языки программирования C, C++, D (с ограничениями), Fortran. Code::Blocks разрабатывается для Windows, Linux и Mac OS X.

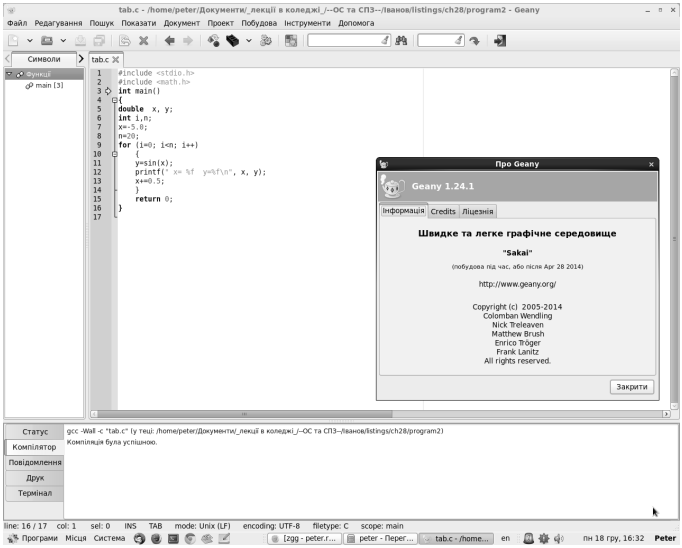


Рис. 2: Оболочка Geany

Выводы:

1. Все три оболочки являются кроссплатформенными, что позволяет студентам работать в той ОС, которая установлена на его рабочем месте (стационарной или переносной ЭВМ). Благодаря этому студенты без дополнительных затрат времени могут познакомиться с кроссплатформенным программированием на уровне компиляции.
2. Оболочки Kuzya и Geany допускают выбор пользователем языка интерфейса (русский, украинский, английский), что позволяет студенту максимально сосредоточиться на работе с разрабатываемой программой.
3. Оболочка Kuzya имеет графический движок, который позволяет студентам строить графики исследуемых функций. Для построения графиков функций, протабулированных в Geany или Code::Blocks, придется использовать внешние средства, например gnuplot.

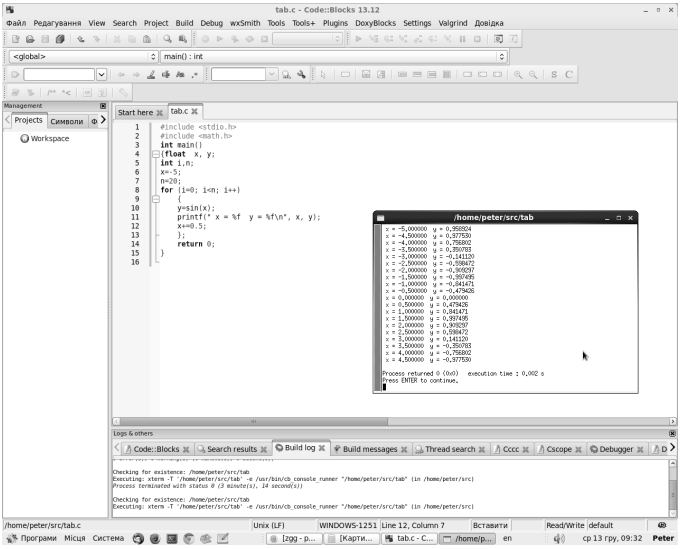


Рис. 3: Оболочка Code::Blocks

Серегей Абрамов
Переславль, Институт Программных Систем РАН

Ошибки в государственном надзоре за высшим образованием — главная проблема высшего образования в России

Очевидная цель надзора — обеспечение высокого качества образовательного процесса во всех вузах страны; обеспечение подготовки вузами востребованных страной кадров высшей квалификации. В конце концов, цель — кадровое обеспечение решений поставленных задач (цитаты): «повышение производительности труда» и «создание 25 миллионов высокотехнологических модернизированных рабочих мест».

И в то же время, сегодня выстроенный надзор за вузами и система требований к вузам (показателей, стандартов, уложений, мониторинга):

- **Во многом не соответствуют этим целям.** Иногда «с точностью до наоборот» — иногда от вузов требуются вещи прямо противоположные поставленным целям! Примеры приводил.
- **Переусложнены.** Это накладывает на вузы (и на надзорные органы) никому не нужную нагрузку. Сначала по всей стране огромная армия людей (в вузах) вычисляют бесполезные для образовательного процесса показатели и оформляют бесполезные для образовательного процесса бумажки, потом другая армия (чиновников и экспертов от надзора) все это дело проверяют.

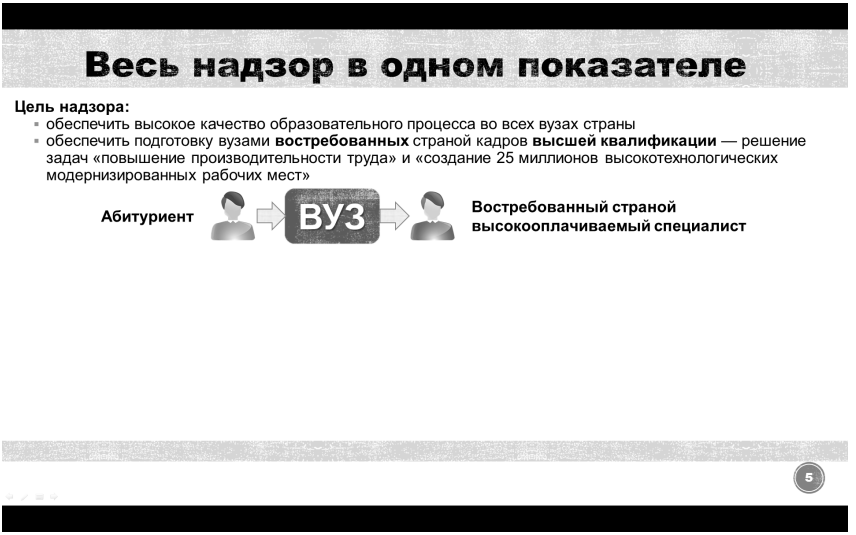
Этот мартышкин труд отвлекает ресурсы от образовательного процесса, приводят к неоправданным финансовым затратам. В конце концов, все это пожирает существенную долю бюджета страны, выделенного на образование. И эти расходы на бессмысленный надзор финасово убивает малые и/или негосударственные вузы.

Системная ошибка: государство зарегулировало процесс (квадратные метры, учебники, РУПы и всякое прочее) высшего образования и контролирует процесс, но никак не контролирует результат работы вуза: что за выпускников он выпустил и что с ними стало. В области образования регулировать процесс — дело вредное для страны: подходы в разных случаях разные, да и не в процессе дело. Вузы потихоньку начинают все громче стонать от этого, тем более что чиновник вряд ли разбирается (да и не должен) в процессе, зато вполне может сформулировать государственный заказ на результат, но он этого сегодня не делает. Как итог, мы видим первые «бунты на корабле»[1].

Мною подготовлено множество конкретных примеров ошибочных критериев, показателей и требований из области надзора (`https://goo.gl/6dmfH4`). Они передавались в стенах Госдумы России из рук в руки представителям Минобрнауки и Рособранадзора России, публиковались в журнале «Ректор вуза» со словами «Редакция ждет реакции из Минобрнауки России». Тишина.

А, возможно, это и правильно? Зачем править отдельные недочеты системы надзора? Нас, технарей, особенно IT-шников, практика учит: иногда систему не надо исправлять. Ее надо выкинуть. И создать заново. С белого листа.

[1] Творческие вузы рассказали В.В.Путину о неприменимости стандартизации образовательного процесса к ним: `https://goo.gl/mVSc8o` — «Одна из главных проблем — вузы, дающие образование в сфере искусства, подчиняются Министерству образования, а не культуры, как было раньше, а значит **должны работать по общим стандартам**».



Вот давайте так и сделаем. Перейдем от критики к конструктиву. Начнем от печки. Еще раз: нам нужно обеспечение высокого качества образовательного процесса во всех вузах страны; обеспечение подготовки вузами востребованных страной кадров высшей квалификации. В конце концов, кадровое обеспечение решений поставленных задач (цитаты): «повышение производительности труда» и «создание 25 миллионов высокотехнологических модернизированных рабочих мест». Вот так (см. рис.).

С этой точки зрения, стране (власти) все равно, как устроен вуз. Черный ящик. Очный он, заочный, дистанционный или еще какой. Какие у него учебники. Как там учат детей... Возможно не так, как считают чиновники, возможно при помощи гипноза или таблеток... Но если на выходе востребованные страной высокооплачиваемые специалисты, то это правильный вуз. И он нужен стране. Все. Мы готовы сформулировать единственный показатель эффективности работы вуза:

Вуз оценивается по единственному показателю: сумма белых зарплат, полученных в России выпускниками вуза за первые три года их работы

Здесь каждая часть очевидна: стране интересна работа выпускников только в России, а не за границей; в реальной, а не теневой экономике; высокие зарплаты являются индикаторами и востребованности, и высокой квалификации; три года сильно́ влияние работы вуза, потом человек может сам изменить свою квалификацию.

А теперь рассмотрим последствия такой перестройки с нуля всей системы надзора, что будет, если мы отринем все сегодняшние требования, показатели и критерии и будем использовать только этот показатель:

- **Простота и прозрачность надзора.** От вуза не требуется сдавать (а власти не требуется собирать и перепроверять) никаких сведений: списки выпускников есть у государства, сведения об их НДФЛ-2 есть у государства.
- **Простота реализации и полная объективность.** От власти не требуется никаких сложных кадровых и финансовых затрат для этого надзора: речь идет о достаточно простой IT-системе — просуммировать НДФЛ-2 в базах данных. Копеечный вопрос.
- **Освобождение ресурсов.** Масса времени у сотрудников вуза освобождается для образовательного процесса. Армия чиновников и экспертов от надзора освобождается для созидательного труда. Это все образованные и трудолюбивые люди! Они для страны смогут принести гораздо больше пользы, чем сегодня приносят вреда для образования в России.
- **Поддержка принятия решений, управление вузами.** Все просто: бюджет на высшее образование делить по данному показателю между всеми вузами — как государственными, так и частными. Опять: простота реализации и полная объективность. Прямое управление. Бюджет тратится на расширенное воспроизводство человеческого капитала и уже им — расширенное воспроизводство бюджета — воспроизводство НДФЛ-2. Прямая бюджетная эффективность: бюджет тратится на расширенное воспроизводство бюджета. Что проще?
- **Принятие решения о неэффективности вуза, российские рейтинги вузов:** вычисляем приведенный показатель — на одного студента,— и сортируем вузы. Руки чешутся кого-то закрыть? Отчеркивайте группу снизу рейтинга. Хотя, если они не расходуют российский бюджет... или мало его расходуют... Оставьте их в покое!

В завершении материала, укажу на разумные уточнения к предложенному показателю.

- **Региональная поправка.** Конечно, в разных регионах зарплаты разные. Поэтому разумно при вычислении показателя вуза зарплаты (и НДФЛ) выпускников считать не в рублях, а в «средних зарплатах по региону», как это сделано в майских указах. Опять-таки, это не сильно усложняет систему — мелкая задачка для программистов.
- **Учет госприоритетов по отраслям знаний и по специальностям.** Если по каким-то направлениям государство желает обозначить свои особые интересы, то это легко реализовать: надо зарплаты выпускников с теми или иными специальностями умножать на коэффициенты, отражающие заинтересованность государства в этих специальностях. Но, здесь главное не перемудрить. Простая система будет лучше работать.

Заключение. И это предложение 2017.03.28 в стенах Госдумы России мною из рук в руки передано представителям профильного комитета Госдумы, Минобрнауки и Рособранадзора России... Почему бы России не упростить свой надзор в области высшего образования? Не поставить надзор и цели высшего образования в прямую зависимость? Кто против этого? Кто заинтересован в сохранении чудовищных уложений и огромной армии людей, тратящих ресурсы страны (человеческое время и народные деньги) на бессмысленный мониторинг бессмысленных требований, никак не связанных с качеством высшего образования? Какой стране это выгодно? Кто на это ответит?

Д. А. Костюк, А. А. Маркина
Брест, Брестский государственный технический университет

Подход к комплексному межгрупповому usability-тестированию для платформы GNU/Linux

Аннотация

В работе представлен комплексный подход к оценке эффективности взаимодействия пользователя с программным обеспечением на платформе GNU/Linux, включающий комбинированное использование методов психологического анализа и биометрического подхода. Используемые методы включают в себя прохождение тестов, заполнение опросников, протоколирование действий пользователя и коммуникативное взаимодействие с ним, а также измерение биометрических показателей. Приводятся разработанные на принципах свободного контента материалы, а также результаты апробации подхода на задаче сравнения свободных и коммерческих табличных процессоров.

Введение

Для измерения эффективности работы пользователя выделяют два принципиально различных подхода. В первом подходе активно используются экспертные оценки и самосообщаемые параметры; он не поддаётся значительной автоматизации и затратен по времени из-за опросов, хронометража и видеопротоколирования, больше подвержен влиянию человеческого фактора при обработке результатов. Также ряд параметров, таких как физическая нагрузка, может быть оценен лишь по косвенным признакам и самосообщаемым параметрам. Альтернативный подход предполагает экспресс-оценку состояния пользователя с помощью приборов, позволяющих регистрировать параметры, связанные с физической и когнитивной нагрузкой. Он также имеет ряд ограничений. Биометрические измерения предоставляют большой объем данных, поддающихся автоматической обработке, однако они позволяют оценивать эмоциональное состояние и когнитивные процессы лишь по косвенным признакам, что даёт в ряде случаев лишь приблизительную картину.

В настоящей работе предпринята попытка объединения достоинств обоих подходов в рамках комплексного usability-тестирования, нацеленного на получение картины процессов, имеющих место в ходе

человеко-машинного взаимодействия, максимально полной и максимально абстрагированной от влияния человеческого фактора в процессе его оценки. По мнению авторов, подобная методика особенно актуальна для GNU/Linux, не относящейся к числу платформ, «избалованных» полномасштабными usability-исследованиями. Материалы для тестирования пользователей (в особенности универсальные) и программное обеспечение, разработанное и используемое авторами в ходе работы, распространяются под свободной лицензией и доступны в репозитории проекта UXDump (`https://bitbucket.org/AsyaAliset/uxdump`).

Метрики

Список метрик, которые имеет смысл учитывать при проведении исследования, включает показатели результативности (время выполнения задания, ошибки, процент выполненных заданий), нагрузку (пульс, концентрацию внимания, фиксации взгляда) и впечатления респондентов (уровень ожиданий и уровень удовлетворения пользователя). Время и успешность выполнения задания — одни из основных показателей, используются только в сравнении. Успешность выполнения кодируется в бинарном коде (выполнил/не выполнил).

Проблемы, с которыми столкнулись пользователи, регистрируются как менеджером эксперимента, так и с помощью видео-захвата экрана. После тестирования проводится ретроспектива: с пользователем обсуждаются задания, вызвавшие проблемы, проигрывается запись, анализируется реакция и поведение в ходе теста. Это позволяет классифицировать проблемы, выявить более значимые для пользователя, рассчитать частотность проблем (сколько пользователей с ними столкнулись).

Уровень ожидания отражает отношение пользователя к продукту и представляемую им комфортность работы, а уровень удовлетворения — оценку удобства использования системы после прохождения теста. Эти метрики можно получить с помощью стандартных международных опросников System Usability Scale (SUS) и Post-Study System Usability Questionnaire (PSSUQ) [1, 2]. Кроме этого, пользователям предлагается выбирать из списка эпитеты, которые могут описать их впечатления от программного продукта.

Нагрузка может оцениваться в соответствии с опробованной ранее экспресс-методикой [3], по которой регистрируются быстрота вы-

полнения заданий, физическая и умственная нагрузка, направление взгляда пользователя [4]. Для оценки концентрации внимания нами использовалась метрика «Attention» бытового энцефалографа Neurosky Mindwave, физическая нагрузка оценивалась по частоте сердечных сокращений (ЧСС), измеряемой фитнес-трекером, а для регистрации направления взгляда применялись айтрекеры фирмы Tobii. Сбор и первичная обработка данных выполнялись в параллельном режиме программной системой, разработанной в рамках проекта UXDump.

Методика проведения

Проведение тестирования было разделено на следующие части:

1. *Составление сценария использования*, включающего типовые задания для использования программного продукта с нарастающей сложностью, что позволяет отследить тенденцию обучаемости. Задания составляются исходя из опыта респондентов.

2. *Набор участников* одной возрастной группы, имеющих сходный опыт работы с продуктами-аналогами. Оговаривается время проведения каждой стадии эксперимента.

3. *Инструктаж* (приветствие, описание мероприятия, целей исследования, метрик и тестов, подписание соглашения на предоставление и обработку персональных данных).

4. *Вводное интервью* (заполнение анкеты участника, проверка уровня владения продуктами-аналогами, проведение психологических тестов — в нашем случае теста Айзека на определение темперамента и теста Равена на уровень интеллекта).

5. *Демонстрационный показ работы с продуктом.*

6. *Ожидания от работы с продуктом* (заполняются опросники SUS, обсуждаются основные вопросы использования, особенности функционала).

7. *Настройка системы* (в зависимости от имеющегося оборудования, подключение и калибровка биометрических датчиков, айтрекера, а также настройка видео-протоколирования и захвата видео с экрана).

8. *Работа с продуктом* (выполнение тестовых заданий с ведением протокола менеджером, где фиксируются реакции пользователя).

9. *Сбор итоговых впечатлений* (заполняются опросники PSSUQ, проводится ретроспектива).

Важно, что в ходе эксперимента менеджеры, регистрируя время и реакцию пользователей, не вмешиваются в ход выполнения заданий. В нашем случае менеджеры подбирались из круга респондентов, были с ними знакомы, что могло снизить стресс. Ответы на вопросы «Как это сделать?» должны быть расплывчаты («А как вы сами думаете?», «А что бы вы сделали в реальной жизни?»), т. к. это мотивирует респондента разбираться с системой и не смещает фокус. Также респонденты знают о видео-наблюдении, но внимание в ходе теста на этом не акцентируется.

Апробация на примере табличных процессоров

Апробация методики выполнялась на задаче сравнения эргономики табличных процессоров, входящих в состав современных офисных пакетов. В качестве подопытных выступали студенты в возрасте 18–19 лет, получающие техническое образование, которые имеют представление о табличном процессоре, но не используют его ежедневно. Для тестирования были выбраны Microsoft Excel 2016 и LibreOffice Calc 5.4.4. В тестировании участвовали 14 респондентов.

Выбор был обоснован следующими соображениями:

- современный табличный процессор является мощным приложением с развитым функционалом и сложными инструментальными средствами управления;

- ограниченное владение подопытных приложениями данного типа позволяет оценить, как особенности интерфейса приложения влияют на освоение его функционала.

Заметим, что ленточные интерфейсы имеют опредённые преимущества для сложных приложений [3]. Но т. к. сравнение классических и ленточных интерфейсов чаще проводится на примере текстовых процессоров, представляет дополнительный интерес их сравнение на сложных задачах другого типа. Кроме того, интересно оценить и результативность переработки, которую претерпел Libreoffice Calc 5.

В ходе вводного интервью три респондента были отсеяны по тесту Айзенка из-за низкой эмоциональной устойчивости и/или непреодоления порога достоверности теста, а один – по тесту Равена, показавшему существенно отличающийся от остальных респондентов уровень интеллекта, не вписывающийся в средние показатели группы.

Результаты снятия метрик приведены в таблице. Как можно заметить, работа в Calc проходила успешнее и быстрее, при меньшем уровне стресса. Анализ показал отсутствие изменений употребления негативных прилагательных при описании интерфейса Excel до и после эксперимента (8 случаев), в то время как для Calc таких случаев было 8 до эксперимента и 7 после. Для обоих пакетов заметна тенденция, когда респонденты чаще считали его более полезным и эффективным после тестирования, чем до. Мнение о простоте использования Excel в процессе работы снизилось на 67%, а мнение о простоте использования Calc — на 25%. Аналогично можно сказать об оценках экономии времени при использовании пакета (Excel — снижение на 60%, Calc — снижение на 50%). При этом есть категория пользователей, полностью справившихся с тестом только в Calc.

Основными проблемами, выявленными при при выполнении заданий в Excel оказались частое отсутствие корректных подсказок, сложный и запутанный поиск нужного функционала, неинтуитивность пользовательских настроек, низкая заметность элементов. При работе с Calc наблюдались проблемы заметности некоторых элементов управления и поиск имён функций по общему справочнику. Заметим также, что некоторые пользователи после работы с Excel нашли его «личным», «покровительствующим», «властным», «неконтролируемым» и «вызывающим смущение», а Calc — «высококачественным», «нетрадиционным», «насыщенным», «захватывающим», «вызывающим смущение», «ценным», и немного «разочаровывающим».

	Excel	Calc
Результативность		
Минимальное время выполнения всего теста	23 мин	22 мин
Максимальное время выполнения всего теста	1 ч 6 мин	55 мин
Среднее время выполнения всего теста	31 мин	30 мин
Минимальное время выполнения одного задания	20 сек	15 сек
Максимальное время выполнения одного задания	24 мин	14,5 мин
Среднее время выполнения одного задания	2 мин	1,5 мин
Сколько респондентов выполнили все задания	14%	43%
Наименьший процент выполненных заданий в тесте	40%	50%
Процент выполненных заданий теста	80%	89%
Физическая нагрузка		
Максимальная ЧСС	162 уд/мин	132 уд/мин
Минимальная ЧСС	59 уд/мин	74 уд/мин
Впечатления респондентов		
Процент ожидания по SUS	51%	52%
Процент удовлетворения по PSSUQ	64%	64%
Рост тенденции (от ожидания к удовлетворению)	16%	16%

Литература

[1] *Sauro J.* Measuring usability with the system usability scale (SUS). `https://measuringu.com/sus/` Posted on February 2, 2011.

[2] Post-Study System Usability Questionnaire (PSSUQ). // UX Glossary `http://www.conetrees.com/2010/12/ux-glossary/post-study-system-usability-questionnaire-pssuq` Posted on December 10, 2010.

[3] *Костюк Д.А., Латий О.О., Маркина А.А.* Подход к биометрической оценке эргономики графического интерфейса пользователя // Вестник Брестского государственного технического университета. Физика, математика, информатика. — 2016. — № 5(101). — С. 46–49.

[4] *Дубицкий А., Костюк Д., Маркина А., Фомин С.* Применение айтрекеров для юзабилити-исследований в GNU/Linux // Четырнадцатая конференция разработчиков свободных программ: Тезисы докладов. — Калуга, 22–24 сентября 2017 г. М.: Базальт СПО, 2017. — С. 36–41.

В. В. Буслюк, Д. А. Костюк, Д. И. Кульбеда, Н. А. Терешкевич, В. А. Юхно
Брест, Брестский государственный технический университет

О возможностях применения в вузах аутентификации на основе персональных устройств

Аннотация

Рассматриваются альтернативы в сфере аппаратных средств идентификации и аутентификации пользователей в компьютерных классах и информационных терминалах под управлением GNU/Linux. Анализируется возможность использования штатных идентификационных карт, а также персональных гаджетов студентов и/или профессорско-преподавательского состава в качестве аппаратных ключей. Рассмотрен спектр PAM-модулей, применимых для решения данной задачи. Обсуждаются вопросы безопасности, а также возможности интеграции средств альтернативной аутентификации и LDAP.

Под средствами аппаратной идентификации и/или аутентификации пользователей обычно понимают устройства, относящиеся к двум категориям:

- аппаратные ключи защиты, взаимодействующие с компьютером по шине USB, через устройство считывания контактной памяти, либо посредством беспроводной передачи данных малого радиуса действия;
- средства биометрии, чаще сканеры отпечатков, реже – более сложные системы, выполняющие сканирование в оптическом и, опционально, инфракрасном диапазоне (распознавание лица, рисунка радужной оболочки глаза и др.).

В обоих случаях приходится комплектовать рабочие места стационарными считывателями и/или предоставлять (и периодически восстанавливать) персональные ключи пользователей — смарт-карты, удостоверения либо брелоки, содержащие RFID-метку, и др. Аппаратные ключи позволяют избавить конечных пользователей от забот о запоминании и вводе криптостойких персональных паролей; однако из-за возникающих дополнительных проблем они обычно используются в организациях, нуждающихся в дополнительных мерах защиты, и практически не применяются для упрощения и повышения удобства

аутентификации (исключением является сравнительно недавняя тенденция биометрической разблокировки мобильных устройств). Для вузов использование аппаратных средств аутентификации не характерно в силу обширного компьютерного парка с чрезвычайно большим числом пользователей, а также не слишком высокими требованиями секретности, делающими излишней массовую двухфакторную аутентификацию и другие дополнительные защитные меры.

Наоборот, в вузах существует подмножество задач, где аутентификация пользователя фактически служит только его идентификации. Поэтому в ряде случаев оправдана облегчённая авторизация с отсутствием доступа к критичному функционалу или с активацией дополнительных мер защиты при запросе такого доступа. Пример таких типичных задач, не связанных с чувствительными к утечке данными – доступ студентов и преподавателей к личному расписанию, оповещение студентов о заданиях и результатах проверки работ. К этой же категории (при определённых условиях) можно отнести доступ студентов к компьютерам учебных классов [1].

Рабочие станции под управлением GNU/Linux позволяют использовать биометрическую аутентификацию с помощью ряда PAM-модулей, включая следующие:

- `fprint` — проект, поддерживающий ряд стандартных сканеров отпечатков (как правило, встроенных в ноутбуки), а также несколько его менее универсальных аналогов: `PAM Fingerprint` для аутентификации с помощью датчика отпечатков в Arduino/Raspberry Pi, аналогичный модуль `PAM_BFP` для USB-сканера Futronic [2, 3]. К этой же категории относится `pam-bioapi` — фреймворк, являющийся ещё одной попыткой унифицировать считыватели отпечатков в Linux, сам по себе поддерживающий подгружаемые драйвера;

- `PAM face-recognition` — классический и, пожалуй, наиболее известный модуль, пытающийся распознать лицо пользователя с помощью веб-камеры и библиотеки OpenCV [4].

При разной степени работоспособности перечисленных модулей, всех их объединяет ориентация скорее на индивидуальное применение, а не на общедоступные устройства. Однако в последнее время множество пользователей располагает персональными гаджетами, которые могут с не меньшим успехом использоваться для идентификации (и, в случае не критичных к безопасности задач, авторизации),

но при этом применимы для мест публичного доступа. Список PAM-модулей, поддерживающих аппаратную аутентификацию, включает:

- `PAM-PKCS#11` — классический модуль для аутентификации (опционально, также идентификации) пользователей на основе смарт-карт, имеет экспериментальную поддержку LDAP [5];
- `pam_usb` позволяет использовать USB-накопители в роли аппаратных ключей [6];
- `pam_nfc` — модуль для авторизации на основе ID-тегов карт RFID, изначально рекомендованный для идентификации и систем многофакторной аутентификации [7, 8];
- `pam-blue` — модуль для аутентификации с помощью гаджетов по протоколу bluetooth [9] (используется MAC-адрес Bluetooth-устройства), а также `pamble` — сходный модуль для Bluetooth Low Energy, написанный одним из участников проекта и ориентированный на авторизацию с помощью популярных фитнес-трекеров Xiaomi [10].

В случае Беларуси список персональных устройств, пригодных для задач идентификации и аутентификации, дополняется студенческими билетами на основе смарт-карт, активно применяемыми белорусскими вузами. По факту, это разработка БГУ, основанная на использовании бесконтактных карт формата MIFARE Standard 4k [11] (протокол существует долгое время и поддерживается многими серийными NFC-считывателями).

Заметим, что выше рассматривались модули аутентификации на рабочих станциях под управлением Linux, в то время как существенная часть задач, связанных с информированием пользователей, реализуется отдельными приложениями. Модули расширения системы PAM наглядно иллюстрируют ситуацию с аппаратной аутентификацией, а случае других применений (например, информационных киосков) ситуация является аналогичной, поскольку данные PAM-модули и конечные приложения с функцией аутентификации в большинстве случаев строятся на одних и тех же свободных библиотеках.

Отличием является то, что в случае информационных киосков и других специализированных решений, забота об авторизации перекладывается на разработчика приложения и может решаться по разному в зависимости от конкретной задачи. При авторизации же на рабочих станциях возникает дополнительная проблема, связанная со

стандартной схемой хранения учетных записей пользователей на сервере LDAP (что, учитывая численность студентов, заметно упрощает администрирование компьютерного парка вуза).

PAM по определению является модульной системой, и стандартная схема предполагает использование libnss для получения информации о пользователях и группах, а затем последовательное применение модулей для попытки аутентификации, включая, например, запасной вариант использования `pam_ldap`. Учитывая, что поддержка LDAP заявлена только `PAM-PKCS#11`, сетевое хранение сертификатов (или их аналогов) немэйнстримных PAM-модулей для учетных записей становится проблемой. Очевидное решение предполагает монтирование домашних каталогов по сети с помощью autofs. При этом домашние каталоги и хранящиеся в них служебные данные, необходимые для аутентификации пользователей, оказываются в несколько большей сетевой доступности для потенциальных злоумышленников. Однако с учетом отправного тезиса об использовании идентификации и аутентифкации на основе гаджетов за пределами областей, критических с точки зрения информационной безопасности, данная проблема не является существенной для рассматриваемых применений.

Литература

[1] *Пойта П.С., Костюк Д.А., Дереченник С.С., Луцюк П.Н.* Повышение сетевой безопасности в компьютерном парке вуза за счет буферизации и изоляции ресурсов // Электроника инфо. – 2013. – No.6 (96). – С. 111–113.

[2] pam fprint. `https://www.freedesktop.org/wiki/Software/fprint/pam_fprint/`. Last edited on May 18, 2013.

[3] PAM Fingerprint `https://github.com/philippmeisberger/pam-fingerprint`. Commited on Dec 21, 2017.

[4] *Wills N.* A look at PAM face-recognition authentication `https://lwn.net/Articles/523199/`. Published on November 7, 2012.

[5] PAM-PKCS#11 Login Tools. `https://github.com/OpenSC/pam_pkcs11/wiki`. Commited on Sep 28, 2016.

[6] Hardware authentication for Linux using ordinary USB Flash Drives. `https://github.com/aluzzardi/pam_usb`. Commited on 27 Apr 2016.

[7] NFC-based PAM authentification module. `https://github.com/nfc-tools/pam_nfc`. Committed on 13 May 2013.

[8] Linux NFC compatibility. http://nfc-tools.org/index.php?title=Devices_compatibility_matrix. Published on on 25 December 2016.

[9] Аутентификация при помощи Bluetooth телефона или USB Flash в Debian/Ubuntu Linux https://www.opennet.ru/tips/1973_pam_auth_usb_bluetooth.shtml. Опубликовано 04.03.2009.

[10] Pluggable authentication module for low level authentication in unix systems by bluetooth low energy device. https://github.com/jaffeetv/pamble. Committed on 23 Apr 2017.

[11] *Абламейко С. В. и др.* Опыт и перспективы внедрения интеллектуальных документов в учебные заведения // Материалы Международной научно-практической конференции «Современные технологии автоматической идентификации и электронного бизнеса. Состояние и перспективы развития ID Comptence 2011», Минск, апрель 2011. – С. 49–57.

Георгий Курячий
Москва, ВМК МГУ им. М. В. Ломоносова

Как я делал проверку копипасты для спецкурса по Python3 и что из этого вышло

Часть первая: why?

В 2017 году (осень) мы решили перезапустить[1] спецкурс 2014 года по Python[2]. Причин было много — наработанная практика за предыдущие два учебных года (базовый курс в Севастопольском филиале МГУ), перевод на Python3, некоторая реструктуризация изложения. Впрочем, подход остался тем же: мы читаем авторское «Знакомство с Python3[3]», дополняем его объяснениями и примерами и решаем множество практических домашних заданий (36 задач на написание программы или функции). Лекции записывались на видео[4] — в основном, скринкаст с небольшой говорящей головой.

[1] https://uneex.ru/LecturesCMC/PythonIntro2017
[2] https://uneex.ru/LecturesCMC/PythonIntro2014
[3] https://docs.python.org/3/tutorial/
[4] https://www.youtube.com/playlist?list=PL6kSdcHYB3x7VJXiCA8OjYAiRBHi7mZTJ

Условия домашних заданий (в среднем по 3 к лекции) и некоторые подсказки по решениям выкладывались на сайте[5], а вот проверку мы доверили факультетской системе проведения олимпиад EJudge[6]. Мы использовали только одну функцию EJudge: программе участника скармливается на стандартный ввод некий текст из набора входных тестов, а результат сравнивается с эталонным.

К сожалению, в курс не входили практические занятия под чьим-либо руководством, так что требования «красоты» или «лаконичности» программ-решений пришлось опустить — не стоит оценивать то, чему не учил.

Видео к курсу регулярно выкладывались в сети, регистрация на «соревнование» в EJudge была свободной, так что на время подведения итогов у нас было 131 зарегистрированный участник соревнования (включая меня), из которых примерно половина (включая меня) «дошла до финиша», т. е. решила более $\frac{2}{3}$ задач. Всего было более 6000 попыток сдать задачу, из которых примерно половина была, с точки зрения EJudge, успешной.

Понятно, что в таких условиях полномасштабный экзамен, с учётом требований к проведению экзамена[7] — дело очень ресурсоёмкое. С другой стороны, человек, который успешно решил порядка 30, временами не самых простых задач, вряд ли нуждается в строгой экзаменовке. Беглое чтение написанного им кода вкупе с данными о решённых задачах даёт достаточное основание для оценки.

Правда, тогда резко повышается значимость плагиата при написании программ-решений. Для начала мы решили строго ограничить время решения каждой задачи, но потом ввели градацию: за решение в первые две недели участник получает 4 балла, в третью неделю — 2, а после — 1. Таким образом, люди, которые не сумели решить её вовремя, имеют возможность посмотреть беглый разбор решения (как раз через две недели) и оперативно заработать половину своих бонусов, и даже те, кто махнул было на домашние задания рукой, могут немножко прибавить себе баллов. К экзамену допускались лишь набравшие более $\frac{2}{3}$ из возможных баллов, причём оценки-автоматы «дошедших до финиша» делились строго: до $\frac{7}{9}$ — «удовл», до $\frac{8}{9}$ — «хор», остальное — «отл». Заметим, что человек, желающий сдать

[5] `https://uneex.ru/LecturesCMC/PythonIntro2017/HomeworkRules`

[6] `https://ejudge.ru/`

[7] `https://uneex.ru/LecturesCMC`

все решения незадолго перед экзаменом (вряд ли самостоятельные), сделать это не мог, т. к. не наберёт «проходного балла».

Тем не менее даже поверхностный взгляд на содержимое EJudge показал, что идея плагиата (он же «копипаста») продолжает будоражить неокрепшие умы участников, как если бы они не хотели научиться ЯП Python3, а хотели... чего-то другого, ума не приложу, чего.

В решениях присутствовала как прямая и беззастенчивая копипаста (идентичные программы от разных участников), так и разнообразные приёмы, как сейчас говорят, «рерайтинга» — тривиальной модификации исходного кода (расстановка незначащих пробелов, комментариев и пустых строк, переименование идентификаторов, перестановка независимых блоков кода и т. п.). Причём зачастую даже это не спасало от внимательного взгляда проверяющего, потому что при копипасте сохраняются индивидуальные особенности стиля программирования; что важнее — огрехи стиля, не влияющие на правильность ответа.

Но откуда взяться внимательному взгляду проверяющего, когда таких программ чуть менее, чем 3000??

Часть вторая: how?

Признаюсь честно, трудно сказать, что больше двигало мной: стремление навести какую-то справедливость или возможность попрактиковаться в программировании (разумеется, на Python3). Сама задача оценки «похожести» оказалась привлекательной. Плюс эдакий вызов: сам учил питону, вот теперь сам проверялку пиши.

1. PEP8-фикация[8] (autopep8[9]) и сравнение (difflib[10]) исходный текстов

   Первая мысль была простой: избавиться от незначимых изменений *форматирования* текстов, для чего причесать их «улучшателем» — например, оформляющим программу в соответствие с pep-0008. Построчное сравнение двух программ после «pep8-фикации» делает похожие программы *действительно* похожими, а с учётом того, что `duifflib`

[8] `https://www.python.org/dev/peps/pep-0008`
[9] `https://pypi.python.org/pypi/autopep8`
[10] `https://docs.python.org/3/library/difflib`

умеет размечать совпадения и различия в соответствующих друг другу строках, выделенные имена идентификаторов только прибавляют такому сравнению пафоса. Так или иначе выявленные случаи копипасты/рерайтинга *демонстрировать* удобно именно в отформатированном виде.

2. Расстояние Левенштейна[11] (editdistance[12])

   Для каждой задачи было прислано в среднем порядка 70 верных решений, так что о ручном сравнении нельзя было и думать. И, опять-таки, первая мысль — померить т. н. «редакторское расстояние» (расстояние Левенштейна) в группе решений одной задачи. Было очевидно с самого начала, что одним только расстоянием обойтись не удастся, т. к. переименования и комментарии могут приводить к довольно большому разбросу в оценках. Вот если бы у нас были *препараты* исходного кода, по возможности лишённые синтаксически незначащих различий, этот инструмент пригодился бы.

3. Абстрактное синтаксическое дерево[13] разбора Python3 кода (ast.html[14])

   Преодолев искушение вручную преобразовывать что-то в препарируемом исходном коде, я вовремя вспомнил о том, что имею дело не просто с программой, а с *синтаксически верной* (мало того, работающей и выдающей правильный ответ) программой. Так что если заставить сам Python построить дерево синтаксического разбора этого кода, а сравнивать уже текстовое представление этих деревьев, в них не будет ни пустых строк, ни пробелов, ни комментариев, ни лексически различных, но синтаксически одинаковых элементов, вроде строковых констант, задаваемых кавычками или апострофами.

4. Удаление имён

[11] `https://ru.wikipedia.org/wiki/`Расстояние_Левенштейна
[12] `https://pypi.python.org/pypi/editdistance`
[13] `https://greentreesnakes.readthedocs.io/en/latest/`
[14] `https://docs.python.org/3/library/ast.html`

Осталось только заменить все идентификаторы на один и от же, и переименование так же не будет учитываться при сравнении. Можно было бы унифицировать и строки, но специфика EJudge — строгая проверка соответствия вывода эталону — исключает возможность выводить один текст вместо другого. Получившийся препарат представляет собой нечто вроде «топологии» программы — здесь завели и поименовали несколько объектов, здесь объявили функцию, здесь вызвали функцию и метод объекта, а потом связали именем и т. п.

5. Компрессия

   Вычисление расстояния Левенштейна — алгоритм высокой вычислительной сложности, если считать строкой весь исходный код программы (всё дерево разбора), но выхода нет (иначе придётся разбираться с перестановкой строк). К счастью, решения домашних заданий — небольшие программы, а вдобавок из текстового представления дерева разбора я поудалял всевозможные пустые/повторяющиеся атрибуты и заменил названия синтаксических конструкций на однобуквенные. После этого, конечно, в получившемся препарате ничего уже разобрать нельзя, но нам и не надо, зато при измерении расстояния меньше работы и — что важнее — различия «топологии» имеют большую значимость.

6. Кластеризация

   Теперь возникает неприятная задача: выяснить «кто у кого списал?». Для каждого решения выделяются «близкие» (расстояние до которых не превышает 1% общего объёма препарата), после чего все доступные по близости задания объединяются в единый кластер. Лидер кластера (человек, первым сдавший задание) считается автором решения, остальные — списавшими либо у него, либо друг у друга.

7. Оценка

   - Как уже было сказано, задания, сданные вовремя оценивались в 4 балла, с опозданием в неделю — в 2, с опозданием более 2 недель — в 1. Если решение оказывалось в кластере копипасты, лидер кластера получал полную оценку (это,

естественно, всегда было 4 балла), остальные члены кластера — 1 балл, что приравнивало их к списавшим с доски во время разбора с недельной задержкой.

Получившийся инструмент[15] (написан в течение недели, поэтому за программный продукт не считается, условия распространения — public domain) делает следующее:

1. Открывает и разбирает полученный из EJudge архив решений (время сдачи, автор и идентификатор задачи в архиве присутствуют в имени файла в программой)
2. Строит вспомогательные таблицы с препаратами программ и отформатированными версиями
3. Для каждого решения составляет список близких к нему, после чего разбивает решения на кластеры
4. Выставляет оценку каждому решению исходя из времени сдачи и участию в кластере копипаст
5. Запускает интерпретатор командной строки, позволяющий
   - Посмотреть оценки для всех пользователей и индивидуально
   - Посмотреть список решённых пользователем задач
   - Посмотреть списки пользователей и задач
   - Посмотреть список задач и кластеров копипасты по ним
   - Посмотреть участие пользователя в кластерах копипаст
   - Посмотреть исходный код решения
   - Сравнить два отформатированных решения

Последний пункт — уже чистое развлечение: дело в том, что командная строка в Python (с историей, редактированием и даже достраиванием) организуется слишком просто, грешно было её не сделать.

Как и следовало ожидать, pep8-фикация и вычисление расстояний оказались довольно медленными операциями, поэтому дополнительно между стадиями обработки сохранятся сериализованные промежуточные данные (и восстанавливаются вместо того, чтобы заново считать их при повторном запуске).

[15] `https://uneex.ru/FrBrGeorge/PythonCopypasteProof?action=AttachFile&do=view&target=contest_86.n.py`

Часть третья: so what?

Таблица результатов[16]

Отличники. Довольно много «отл» — полных наборов решений, не вошедших ни в один кластер, или случайно залипших в пару-тройку кластеров (см. ниже). Мы даже некоторое время думали, не сделать ли «отл» более строгим.

False positives. Кластеров предполагаемой копипасты оказалось подозрительно много. Причины:

1. Задача подразумевала решение настолько короткое, что его сложно было записать 60 различными способами
2. Задача предполагала или содержала очевидный алгоритм (например, в пояснениях), реализации которого вполне могли сами совпасть
3. Решение было переписано с доски после разбора 2 недели спустя.
4. Люди и в самом деле решали задачу сообща, после чего сдавали одно и то же или слегка переписанное решение

Первые три категории пришлось учитывать: не рассматривались слишком большие (больше 5 человек) кластеры и кластеры, в которых был я (я решал задачи вместе со всеми и именно свои решения объяснял на доске), а также любые кластеры в задачах первого типа. Это не исключало ложных срабатываний, когда у двух-трёх участников реализация очевидного алгоритма случайно слегка отличалась от остального «пелетона», но зато отслеживало регулярно сотрудничающие пары и тройки.

Выводы

1. Числовые оценки, равно как и оценки сложности, при описанном подходе не могут быть предсказаны заранее, метод требует постоянной адаптации со стороны эксперта.
2. Относительно возможных ложно выявленных копипаст пришлось проводить разъяснительную работу
3. Некоторые коллективные авторы явно получили свои тройки или даже четвёрки
4. Защита от рерайта помогает только от неизобретательного рерайта

[16] `https://uneex.ru/LecturesCMC/PythonIntro2017/HomeworkGradePaste`

Как было сказано вначале, копипаста наиболее очевидна, когда из решения в решение без изменений копируются ошибочные, не имеющие смысла или стилистически странные куски. Ошибок как таковых в правильных решениях нет, странности форматирования в препаратах отсутствуют начисто, бессмысленные же части, вроде повторных инициализаций, неиспользуемых объектов, недостижимых частей программы и прочего, простым синтаксическим разбором не выявить.

При этом именно изменение форматирования, перестановка независимых блоков кода, замена конструкция на аналогичные и прочие ухищрения применяются при более-менее изобретательном рерайте.

И главное. Получившийся инструмент настолько «хорош», что выявляет даже не плагиат, а факт *реализации одного и того же алгоритма*. Я почти уверен, что разрешённый приём — «clean room reimplementation», при котором решение показывают, объясняют принципы его работы, после чего участник пишет своё решение с нуля, — выдал бы значимое количество ложных срабатываний.

Так что определение плагиата (за исключением прямой копипасты) остаётся процессом отнюдь не автоматическим, хотя, с применением инструментов, подобным представленному, несколько автоматизируемым.

Часть четвёртая, заключительная: till when?

На самом деле я бы не стал просто рассказывать об одном частном решении частной же проблемы, если бы не хотел выйти на более общий — и более актуальный! — круг вопросов.

Дело в том, что в нынешних условиях информационной связности всегда есть возможность оперативно аутсорсить решение практически любой задачи, и соревнование методов такого аутсорсинга с методами его подавления превращаются в утомительную игру «полицейские и воры», которая отнимает время, силы, и вообще делает учебный процесс довольно унылым для всех его участников.

Главная неприятность — в том, что «успешное» прохождение тестов становится всё более доступным для людей *без познавательной мотивации*. Новый контингент — это не просто глуповатые или ленивые студенты, вовремя не справившиеся с задачей. Наоборот, они по-своему неглупые и могут проявлять чудеса усидчивости, переставляя строки чужой программы и вручную изменяя имена переменных.

В каком-то смысле они выбирают *самый эффективный* метод выполнения задания «вовремя сдать работающую программу». Просто им *неинтересно решать* саму задачу.

Наверное, это будущие эффективные менеджеры.

Вопрос в том, зачем им учиться на факультете вычислительной математики и кибернетики, и где учиться тем ребятам, которые не так эффективно, *но самостоятельно*, справляются с решениями задач?

Короче говоря, зло — не сам плагиат, а его «успешность» в тех областях, где он по определению неэффективен.

За прошедшие два десятилетия возник и доказал свою эффективность огромный пласт технологии, заключающийся в оперативном поиске и применении *результатов* интеллектуальной деятельности. В отличие от отраслевой науки предыдущих двух-трёх столетий, такое занятие на начальном этапе практически не требует каких-то определённых знаний, принося удовлетворительные результаты. Не требует настолько, что долгое время было объектом насмешек и осуждения. «Скрипт кидди», «индус триальный программист», «stackoverflow программирование» — в одной только в нашей области таких явлений полно.

А между тем, огромная (никем не измеренная) часть *работающих* решений повседневных задач создаётся в наши дни именно таким способом! Победное шествие миллионов PHP-непрограммистов, Java-непрограммистов, Python-непрограммистов, последняя мода — JavaScript-непрограммистов и их блистательных поделий только ширится.

Особенно если вспомнить, что для них изготовлены (когда — такими же непрограммистами, а когда и вполне профессионалами) сотни тысяч удобных инструментов. Это часть совсем другого разговора, но напомню, чему мы учим современных wannabe-разработчиков: не пишите с нуля! не изобретайте *все* велосипеды! на свете полно уже решённых подзадач, оценивайте качество решений и комбинируйте их. И собственное решение оформляйте как такой модуль, решающий вашу подзадачу — этим вы поможете всем вашим коллегам.

А возвращаясь к разговору нашему — все эти люди не знают, где и чему учиться, вот и идут в академические (точнее — исследовательские) вузы, где их заставляют разбираться в чём-то довольно неинтересном, типа линейной алгебры для того, чтобы впоследствии заниматься чем-то массово невостребованным, типа составления алгоритма к доказательству математической проблемы или постановкой

и решением неочевидной научно-технической задачи, не говоря уже о собственно науке — области, в которой нет вообще никаких гарантий успеха.

Если мыслить масштабно, действующая модель образовательной площадки, до сих пор воспроизводящая средневековые каноны — аудитория, доска, что-то бубнящий профессор, студиозусы, записывающие с доски и со слуха, редкие, дорогие и бестолковые учебники — требует изрядной реорганизации. На самом деле совершенно непонятно — с какой стороны, потому что практика замены живых лекторов говорящими головами, бумажных учебников — PDF-ами, а экзамена — электронным тестированием достаточно явно показывает, что так делать не надо. Но это, опять-таки, тема большого и сложного разговора, требующего нескольких точек зрения, опыта в современных образовательных технологиях и т. д.

Итак, как бороться с копипастой?

1. Узаконить.

    - Единственное, что не вызывает сомнений: усилия по копипасте/рерайтингу не могут быть полностью подпольными. Должна существовать какая-то дисциплина или какой-то вид контроля, в котором применялись бы поиск и адаптация готового решения — явно или как альтернатива полностью самостоятельной разработки. К сожалению, далеко не все существующие учебные задания допускают такое применение.

2. Пресечь.

    - Если за плагиатом строго приглядывать, риск станет невыгодным. Строго приглядывать — значит, тратить *больше* времени/внимания на каждом уровне контроля: увеличивать долю заданий, которые выполняются и проверяются в присутствии преподавателя, постоянно наблюдать за выполнением, в оффлайн-заданиях добиваться обратной связи (опять-таки очно), жёстко отслеживать наличие плагиата, и т. д., и т. п. Причём на всякую строгость имеются технические средства её преодоления — от мобильного интернета до микронаушников.

3. Параметризовать задания.

- Если выданная одному студенту задача будет в достаточной мере отличатся от задачи, выданной другому, сам процесс превращения решения одной в решение другой, буде на то возникнет желание, окажется не мене обучающим, чем самостоятельное решение. Некоторые наши шаги в эту сторону мы уже предприняли: так, в разработках 2015/2016 и 2016/2017 учебных годов для первого курса условия *контрольных* генерировались программами (разумеется, написанными на Python3), которые надо было скачать, запустить, получить точную формулировку задачи и в качестве ответа продемонстрировать
  - «номер варианта» (начальное значение датчика случайных чисел, на основании которого генерировалось условие)
  - Формулировку условия
  - Программу-решение

  К сожалению, проверка таких заданий практически ручная: всё, что можно сделать за проверяющего — это определить, что номер варианта соответствует формулировке (т. е., что она действительно сгенерирована случайно, а не списана у товарища или придумана для удобства решения). Необходим особый фреймворк, позволяющий одновременно с условием генерировать эталонные тесты/ответы, а ещё лучше — эталонную программу-решение. Это не выглядит делом невозможным (в конце концов, при изучении не определённых алгоритмов, а самого языка программирования мы вольны в выборе *класса* задач), но, кажется, на практике этим никто особо не занимался.

4. Сменить мотивацию.

   - Голубая мечта, конечно, сделать как-нибудь так, чтобы заниматься плагиатом было невыгодно, а лучше — вообще *неинтересно*. При некоторых входных условиях это неплохо получается: включение учащихся в процесс постановки/проверки/оценки заданий, совместная разработка, введение элемента соревновательности и вообще геймификация и т. п. Однако все такие подходы либо ориентированы на изначально мотивированных участников, либо подразу-

мевают какие-то особые, чудесные подачу и контроль материала, которые сделают весь учебный процесс интересным большинству участников, независимо от предметного содержания.

Если коротко, то мы пока держимся — комбинируя все эти методы понемногу. Но всё чётче ощущение, что воюем мы не с копиастой, а с главным достижением нашей эпохи — информационной связностью. И если продолжать в том же духе, не меняя вектор, копипаста победит.

Сергей Голубев
Раменское, Московская область

Проблема участия непрограммистов в свободных проектах

Аннотация

В докладе анализируются трудности, препятствующие массовому участию непрограммистов в свободных проектах. Считая их непродолимыми, автор приходит к выводу о целесообразности интеграции элементов гуманитарного образования в технические вузы. Причём, интеграция должна основываться на принципах СПО.

Про участие непрограммистов в открытых проектах известно немного. Причём, вероятнее всего, это «немного» довольно точно отражает реальное положение дел.

Привлечение непрограммистов в открытые проекты — основная тема OpenSource.com (проект Red Hat). Красная нить множества статей — это делать надо, но пока непонятно как. Прежде всего непонятна мотивация непрограммистов.

Синтия Харви на сайте `Datamation.com` называет девять наиболее распространённых мотивов, которыми руководствуются компании или независимые программисты при принятии решения участвовать в том или ином открытом проекте.

- Улучшение кода;
- Получение конкурентного преимущества;
- Сокращение затрат на разработку;

- Соответствие базовым ценностям организации;
- Содействие отраслевой стандартизации;
- Развитие программного обеспечения;
- Вклад в собственную карьеру;
- Желание «вернуть долги»;
- Получение удовольствия;
- Обмен знаниями и повышение квалификации.

Получается, что у программиста действительно много стимулов. Причём, минимум половина не имеет ничего общего с работой за идею, а носит сугубо прагматичный характер.

В случае непрограммистов почти ничего из вышеизложенного не работает, за исключением общих вещей типа морального долга и собственного интереса. Никаких прагматичных причин у них нет.

Кто же в этом виноват? Никто.

Red Hat поднимает актуальную тему, но в рамках существующего СПО-сообщества задача не имеет решения. Сообщество сумело построить некую инфраструктуру, но она предназначена исключительно для программистов. Расширить сообщество невозможно, поскольку оно изначально основано на профессиональном признаке. Непрограммисты всегда будут в некотором смысле «сбоку», а часто — просто «пятым колесом».

Что же делать? Этого не знает никто. Даже Red Hat. Пока они ограничиваются призывами к непрограммистам принять участие в свободных проектах, поскольку это нужно проектам. Зачем это нужно непрограммистам, Red Hat предпочитает молчать.

И это при том, что в гуманитарной среде некоторые принципы СПО прекрасно работают. Например сообщество активно привлекалось к написанию книги «Люди, принесшие холод» Вадима Нестерова. То есть, «свои» сообщества есть не только у программистов.

А нужно ли вообще что-то делать? Если решение проблемы неизвестно, то можно считать, что её вовсе нет. Тем более, что у сообщества программистов её действительно нет — независимым разработчикам безразличны логотипы, инструкции, популяризация и т. п. Найдутся желающие — хорошо. Не найдутся — «бабу с возу. . . ».

Если такие проблемы есть у конкретного проекта, то его лидерам следует понимать, что участники-непрограммисты всегда будут играть некую подчинённую роль и иметь недостаточную мотивацию,

которую можно усилить исключительно при помощи универсального мотиватора. Причём, это не обязательно деньги самого проекта.

Например, студентов нетехнических специальностей можно заинтересовать зачётом участия в свободном проекте, как курсовой или даже дипломной работы. Какие-то PR-компании могут оплачивать работу своих сотрудников в свободном проекте и использовать результат для собственной рекламы.

При чём же тут университеты и образование? При том, что «женщины уже в волейбол играют, а мы все время на месте топчемся». Причём, это продолжается давно и поэтому российское СПО-сообщество заметно отстало от мирового. Если полагаться на некое естественное развитие, то потребность в расширении сообщества у нас появится, когда «там» система уже будет вовсю работать и мы снова окажемся в положении догоняющих.

Задачи завтрашнего дня решает не демократия, а авторитаризм. Университет — учреждение, управляемого старым добрым командно-административным способом. Если компании и университеты начнут сотрудничество в этом направлении по принципу «бензин ваш — идеи наши», то не исключено, что через несколько лет совместными усилиями проблема отсутствия нетехнической составляющей проектов будет решена.

Причём, начать следует со студентов технических специальностей, которые хотят получить какие-то нетехнические навыки. Например, правильно снимать видеоролики или вести блог. Их заинтересованность очевидна — умение написать внятную инструкцию по продукту наверняка будет востребованно любым работодателем.

К сожалению, значительная часть выпускников естественно-научных и технических кафедр не умеет писать. Гуманитарное образование так и не интегрировалось в общую систему образования. Это создаёт проблемы уже сегодня.

Решение — внедрение в естественно-научное и техническое образование гуманитарных элементов. В частности, практикума по написанию инструкций, пособий или статей. Причём, это можно и нужно максимально приблизить к реалиям СПО — дистанционные коммуникации, совместная работа и т. п.

Игорь Воронин[1], Вероника Воронина[2]
Шатура, Московская обл., г.Павлово, Нижегородская обл., [1]ИПЛИТ РАН, [2]МБОУ СОШ №7 г.Павлово Нижегородской области

Проект: Образовательный проект УМКИ http://umki.vinforika.ru/, http://robotobum.ru

Роботы УМКИ для обучения программированию через SNAP

Аннотация

Анализируются различные варианты задействования школьников через систему дополнительного образования в сферу интересов технического творчества. Предлагается решение в виде комплекта образовательной робототехники УМКИ, как пакет в школьном дистрибутиве UMKI с использованием среды визуального программирования SNAP

Образовательные роботы приносят прибыль тем, что ребята и девчонки могут получить знания и навыки, которые им пригодятся, когда они поступят после школы в вуз. Поэтому затраты родителей на закупку необходимых комплектов или на обучение своих детей окупятся в том случае, если их ребенок поступит на бюджет технического вуза.

Чтобы получить добавочный бал к ЕГЭ и встать на более выгодную позицию при зачислении в потоке, необходимо принимать участие в различных мероприятиях. Результаты такого участия должны быть подкреплены дипломами и грамотами. Этот выбор можно ограничить следующим списком:

- олимпиады от вузов;
- научно-практические конференции или выставки роботов с защитой своих проектов, проводимые муниципалитетами;
- соревнования по спортивной робототехнике;
- недавно появившиеся и набирающие популярность хакатоны.

По текущей версии закона об образовании, преимущество при поступлении в вуз дается победителям только тех олимпиад, которые утверждены приказом федерального министерства образования, а их в федеральном перечне почти 100. Но реально связанных с робототехникой там не более 5 олимпиад. Если ученик победил в такой олимпиаде, то может либо получить 100 баллов по предмету (информати-

ке, физике, математике), либо поступить вообще без экзаменов в тот вуз, который проводит эту олимпиаду.

Выставки роботов организуют и проводят, как правило, муниципалитеты для того, чтобы отчитаться перед своим руководством о показателях по охвату детей научно-техническим творчеством. На них часто приглашаются дети из начальной школы, либо вообще детсадовского возраста, которые о высшем образовании еще пока не думают.

В конференциях же участвуют средняя и старшая школа, но чтобы грамоту за участие в ней зачел вуз, надо смотреть положение о мероприятии. Его должен организовывать муниципалитет, и в нем должно быть указано, что проводится с целью поиска одаренных детей.

Конечно, самым зрелищным, ярким шоу, вызывающим бурю эмоций, являются соревнования по робототехнике. Но победы в соревнованиях не идут в зачет при поступлении в вуз, даже в международных — таких как WRO. Поскольку подготовка к ним ведется не в рамках системы образования, а скорее как досуг. Причем, пожалуй, ни у одной сети кружков робототехники нет образовательной лицензии.

Почему кружки робототехники часто работают без лицензии — не совсем понятно. Ведь наличие образовательной лицензии может позволить родителям предъявлять оплату кружка к налоговому вычету.

Как вариант можно предположить тезис, что если ты хоть немного становишься заметным, то жди визита контролирующих органов, которые вместо помощи от нашего государства, скорее приведут к потери бизнеса.

Если предложенный тезис окажется верным, то закупать необходимые комплектующие для сборки роботов предстоит не школам из бюджетных средств, а родителям, которые заботятся о будущем своих детей, из личных сбережений.

Так же тогда становится понятным механизм засилия конструкторами роботов международного бренда, который готов предоставлять существенные скидки и товарные кредиты на свой продукт сетевым кружкам. Это доминирование приводит к унификации регламентов различных соревнований, которые оказываются не отличимы один от другого. Все та же езда по линии, толкание баночек разного цвета, толкание роботами друг друга, гонки и футбол с пультов.

При этом остается совершенно не понятно, куда же уходят все те огромные средства бюджетного финансирования, которые выделяют-

ся государственными чиновниками на развитие научно-технического творчества у школьников. Если информация о выделяемых грантах размером в миллиарды рублей еще как-то попадает на страницы прессы, то кроме общих рассуждений об организации Кванториумов и коротеньких заметок в фейсбуке, из которых совершено не понятно, чем же там занимаются, и на что тратятся эти бюджетные средства, какой от них положительный эффект нам, налогоплательщикам, к сожалению, нигде прочитать не удается. Организуются ли где-то для педагогов курсы для обмена передовыми методиками по робототехнике и 3d прототипированию? Или закупаются ли тысячами наборы российских производителей образовательной робототехники? Увы, ответов на эти вопросы от тех коллег, которые работает в этом направлении, мы не встречаем.

Время неумолимо предлагает новые формы организации занятий с ребятами в увлекательной форме так, чтобы у них не пропадал интерес к техническому творчеству. Наравне с олимпиадами и соревнованиями по робототехнике, сейчас становится популярным организация и проведение так называемых хакатонов. Они отличаются от олимпиад, где результат уже заранее придуман организаторами, тем, что участникам предстоит решать задачи, предлагая наиболее эффективный способ достижения поставленной цели.

Как правило, в хакатонах ставятся задачи из реального бизнеса организатора, например, Сбербанка или ГАЗа. В хакатонах школьники должны не просто показать, как они научились программировать роботов для езды по линии, а разрабатывают полезный и осязаемый продукт. Причем победителям вполне возможно получить на свою команду денежный приз. Такие призы, как правило, серьезно мотивируют ребят на подготовку к состязаниям. И они уже самостоятельно изучают те дисциплины, без которых невозможно определить координаты робота в пространстве, для того чтобы решать реальную задачу, востребованную реальными заказчиками с реальными деньгами.

Для развития системы дополнительного образования через сферу интересов технического творчества — предлагается решение в виде учебно методического комплекта образовательной робототехники УМКИ, как пакет в школьном дистрибутиве UMKI с использованием среды визуального программирования SNAP или КУМИР. Это снижает уровень психологического барьера для входа в робототехнику большому числу школьных учителей.

Никита Бесшапошников[1], Михаил Кузьменко[1 2], Анатолий Кушниренко[1], Александр Леонов[1 2 3]
Москва, [1]ФГУ ФНЦ НИИСИ РАН, [2]Московский государственный университет, [3]Московский педагогический государственный университет.
www.mirera.ru

Элементы цифровизации образовательного процесса на примере системы Мирера

Аннотация

Компоненты цифровой образовательной среды должны соответствовать целям и задачам современного образовательного процесса. Обсуждается методика построения цифровых курсов, упрощающих работу современного педагога. В качестве примера излагаются принципы цифровизации педагогического процесса в системе Мирера.

Перед современным образованием стоят непростые задачи и вызовы. С одной стороны, вхождение общества в цифровую эпоху открывает педагогам новые возможности, предоставляя доступ к разнообразным новым инструментам для подготовки и проведения образовательного процесса. С другой стороны, к обучающимся, уже погруженным в цифровой мир, слабо применимы отточенные годами методики преподавания. Внутренняя мотивацию к получению знаний в процессе образования у современной молодежи уменьшилась радикально. Львиная доля дня современного студента или школьника тратится на простое общение в чатах и социальных сетях. Фронтальная методика обучения уже не дает ожидаемого эффекта.

Доступность информации сыграла злую шутку с педагогическим сообществом. Сейчас педагог уже не может на лекции играть роль «воспроизводящего устройства», излагая материал, копия которого в цифровом виде доступна в Интернете. Слушателям это просто не интересно и никакими педагогическими приемами это исправить нельзя. Однако, можно попытаться исключить ненужное дублирование материала, если не излагать его на аудиторных занятиях, а распространять исключительно в цифровом виде для самостоятельного освоения, оставив аудиторное время для консультирования (вплоть до индивидуального) и контролирования уровня достигнутых компетенций. При этом, практические работы также удобно проводить в автоматизированных системах, позволяющих учитывать текущую успеваемость обучаемого при прохождении курса. По мнению авторов, для

STEM-дисциплин, в особенности для программирования, такой подход к преподаванию сегодня дает наилучший результат.

В целях конструирования нового подход к организации образовательного процесса авторами спроектирована и апробирована на механико-математическом факультете МГУ и нескольких факультетах МПГУ система Мирера цифровизации образовательной среды [1].

Основной образовательной единицей в системе является *курс*, который в свою очередь состоит из *контестов* (набора заданий и материалов, объединенных общей темой). Авторы придерживаются тезиса «без расписания нет образования» и считают необходимым привязать каждый контекст к временному периоду его выполнения (изменение этого периода возможно, но только путем выполнения определенных бюрократических процедур).

Задания, в рамках одного контеста, привязаны к конкретному проверяющему модулю (например, компилятор C++, ЭВМ-практикум, и т.д.) и каждое задание, как правило, оцениваются отдельно и бинарно (сдано/не сдано) [2]. Проверяющей модуль использует заранее подготовленный набор тестов или анализирующую программу, включая возможную проверку на «антиплагиат» путем анализа не только предъявляемого студентом результата, но и истории его получения с временными метками. Задания могут быть снабжены дополнительной информацией для студентов, облегчающей (по мнению преподавателя) выполнение одного задания или последовательности заданий в одном контесте. Контрольные, включенные в курс, представляют собой контесты, период выполнения которых ограничен временем начала и конца того аудиторного занятия, на котором проводится контрольная.

В начале учебного года преподаватель создает курс, и, опираясь на текущее расписание занятий, привязывает расписание курса к календарю. Например, если курс рассчитан на 17 недель, с аудиторными занятиями по определённым дням в определённое время, то контрольные работы будут, например, на 5, 10 и 15 неделях на аудиторных занятиях.

Далее, преподаватель объединяет слушателей курса в группу социальной сети ВКонтакте, где и будут появляться (по расписанию) материалы его курса и другая дополнительная информация. По определенным датам, в зависимости от расписания курса, студенты автоматически получают рассылку материалов в группу, сообщения о предстоящих контрольных, персональных достижениях при сдаче контестов и т.п.

Курсы необходимо менять со временем, может возникнуть необходимость передать готовый базовый курс другому преподавателю для использования в отличающихся условиях, типична ситуация построения преподавателем нового курса на базе уже существующего собственного с добавлением/удалением/изменением контестов/задач/материалов. Система Мирера предоставляет средства автоматизации при решении как подобных задач, так и задач компоновки нескольких курсов в один или, наоборот, разделения одного курса на несколько независимых.

Литература

[1] Леонов А. Г., Райко М. В., Бесшапошников Н. О., Ерёмин Д. Б. «Мирера — система поддержки непрерываемого образования»// Сборник материалов Двенадцатой конференции «Свободное программное обеспечение в высшей школе», Basealt, 2017, М., тезисы, с. 47–50

[2] Бесшапошников Н. О., Леонов А. Г., Мащенко К. А., Прилипко А. А. «МетаМир — система для проведения индивидуальных и командных олимпиад по алгоритмике и программированию для дошкольников и младших школьников» // Наука нового времени: сохраняя прошлое — создаем будущее, Санкт–Петербургский Центр системного анализа, 2017, СПб, тезисы, с. 49–55

Никита Бесшапошников [1], Анатолий Кушниренко [1], Александр Леонов [1 2 3]
Москва, [1] ФГУ ФНЦ НИИСИ РАН, [2] Московский государственный университет, [3] Московский педагогический государственный университет.

Новые возможности «Пиктомира» — параллельно-кооперативное программирование и командные соревнования

Аннотация

Бестекстовая учебная система ПиктоМир позволяет ребенку составить программу управления виртуальным роботом, действующим в некоторой псевдореальной трехмерной обстановке. В целях поощрения коллективной работы детей в ПиктоМир были добавлены возможности параллельного выполнения двух или более программ, параллельно

управляющих двумя или более роботами. Эти параллельно выполняющиеся программы могут составляться командой, каждый член которой, работая на своем планшете, составляет программу управления своим роботом, согласуя свою работу с другими детьми так, чтобы роботы выполнили некую общую задачу.

По причинам, изложенным в статье [1], при обучении алгоритмике и программированию необходимо часть усилий направить на организацию кооопературной, командной работы детей.

Для организации командной работы, в частности, для организации командных соревнований, ПиктоМир предоставляет возможность параллельного выполнения нескольких программ, каждая из которых составляется одним ребенком — членом команды — и монопольно управляет одним подвижным виртуальным роботом. Все эти виртуальные роботы действуют в единой, глобальной обстановке. Кроме того, в эту глобальную обстановку могут быть включены неподвижные роботы, Флаги и Кувшины (счетчики), которые можно использовать для синхронизации.

ПиктоМир предназначен, в первую очередь, для дошкольников и младших школьников и потому модель параллельного выполнения должна быть максимально проста и должна осваиваться путем рассмотрения нескольких простых примеров, не требуя каких-либо вербальных объяснений. Мы не ставим своей целью обучение параллельному программированию. Для синхронизации действий разных роботов используются наглядные приемы

— выдача роботам «пустой» команды, которую мы называем «подмигнуть» и визуализируем в соответствии с этим названием.

— циклы ожидания (buzy loop), отслеживаюшие перемещения подвижных объектов на поле, например

нц пока <впереди нет ящика> подмигнуть **кц**
нц пока <ящик можно сдвинуть> вперед **кц**

— циклы ожидания (byzy loop), отслеживаюшие изменение глобального объекта, например

нц пока <кувшин пуст> подмигнуть **кц**
нц пока <впереди свободно> вперед **кц**
опустошить кувшин

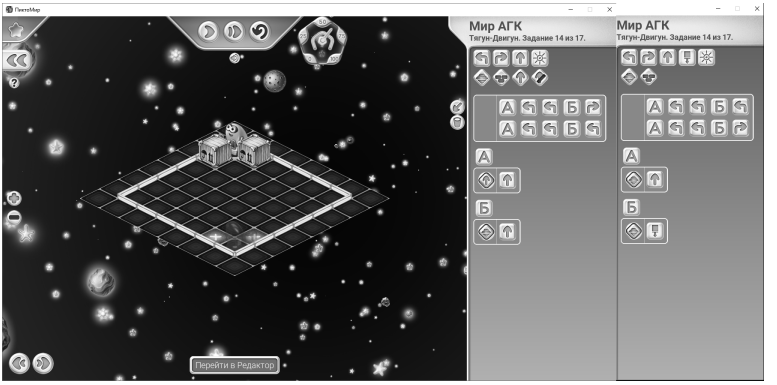


Рис. 1:

Однако начинаться знакомство с параллельным выполнением может и должно с простых примеров, вообще не требующих синхронизации.

Рассмотрим простейшую обстановку, в которой действуют два робота — Тягун и Двигун, которые должны переместить на нужные места — помеченные крестиками — 2 ящика. В этой задаче решение очевидно и никакая синхронизация не требуется. Двигун двигает до упора первый ящик и возвращается в исходную позицию, затем двигает до упора второй ящик и снова возвращается в исходную позицию. Тягун же вначале идет в направлении финишной позиции первого ящика, утыкается в него, разворачивается и тянет ящик до упора. После чего проделывает то же со вторым ящиком.

При командной работе два члена команды, два ребенка, составляют две программы для Двигуна и Тягуна независимо, каждый на своем планшете. Программы управления роботами в этом задании оказываются очень похожими. Программу другого члена команды ребенок может посмотреть, но не может изменить. Члены команды сидят рядом и планируют, как вместе выполнить работу. Когда каждый член команды составил свою программу (или ее фрагмент) эти две программы можно запустить параллельно.

Для тренировки полезен режим, когда один ребенок попеременно выполняет роли двух членов команды, то есть составляет обе программы, которые выполняются параллельно.

Для создания привлекательных графических эффектов ПиктоМир позволяет клонировать роботов, то есть размещать на поле несколько роботов одного типа, выполняющих одну и ту же программу.

Настоящая работа выполнена по теме 0065-2018-0017 госзадания 2018 года для ФГУ ФНЦ НИИСИ РАН.

1. Бесшапошников Н.О., Реализация параллельно-кооперативного выполнения заданий в учебной системе программирования для дошкольников и младших школьников // Вестник кибернетики — Сургут, 2017, № 4 (28), с. xx–xx

Ивченко Олег Николаевич, Драль Алексей Александрович, Ройтберг Михаил Абрамович

Москва, Московский физико-технический институт (государственный университет)

Проверка домашних заданий в экосистеме Hadoop с использованием Continuous Integration

Аннотация

Тестирование приложений в области больших данных имеет ряд сложностей по сравнению с тестированием традиционных программ. Одна из них — это собственно большие данные и, как следствие, большой объём результата программы (что создаёт сложности при его проверке). Кроме того, при проверке Hadoop-задания важно не только проверить его результат, но и оптимальность реализации. В этом случае полезен мониторинг ресурсов кластера в период выполнения программы. Но при большом количестве проверяемых задач такой мониторинг занимает дополнительное время. На конференции OSEDUCONF-2017 году была представлена система HJudge, которая решала первую проблему, т.е. достаточно эффективно производила проверку результата работы программы. Но HJudge практически не проверяла оптимальность программ. Она также имела ещё один недостаток: проверяющий самостоятельно инициировал процесс проверки каждой задачи, для чего требовался прямой доступ к клиентской машине Hadoop-кластера. Если мы имеем дело со сравнительно небольшим количеством задач, этот недостаток не кажется существенным. Как показала практика 2017 года, если в день поступает несколько десятков задач, студент в среднем ожидает оценки в течение 1–2 дней. Это создаёт предпосылки для

серьёзной доработки системы HJudge. В данной работе будет рассказано про улучшенный вариант HJudge, позволяющий студенту сдавать задания по обработке больших данных и получать оценку без участия преподавателя. В этом случае студент ожидает не более нескольких секунд после того, как его программа отработала на Hadoop-кластере.

Популярность Больших данных в образовании

В наше время разработчики и аналитики BigData востребованы на рынке и это порождает спрос на BigData-ориентированные курсы в высшей школе. В частности, остановимся подробнее на курсах, в которых участвуют авторы данных тезисов.

1. Хранение и обработка больших объёмов данных. Курс читается с 2015 г. на 1-м курсе магистратуры ФИВТ МФТИ. В рамках курса студенты знакомятся с фреймворками Hadoop, Hive, Spark, Hbase [1].

2. Многопроцессорные вычислительные системы. Курс читается на 3-м курсе бакалавриата ФИВТ МФТИ. В 2016 г. в курс был включён модуль «Распределённые системы», в рамках которого студенты знакомились с основами Hadoop, внутренним устройством распределённой файловой системы HDFS, а также фреймворком Hive, позволяющим работать с SQL поверх MapReduce.

3. Специализация «Big Data for Data Engineers», запущенная осенью 2017 г. на образовательной платформе Coursera. В ней сотрудники ведущих компаний, специализирующихся на обработке данных (Yandex, Amazon, Mail.ru и др.) знакомят слушателей с последними разработками в области Больших данных.

Материал всех курсов подкреплён набором программистских задач, которые нужно проверять. Решения имеют вид скриптов либо Jupyter-ноутбуков. Все задачи работают с текстовыми данными.

Процесс проверки заданий. Необходимость автоматизации

Процесс проверки задания, написанного с использованием фреймворков экосистемы Hadoop, можно разбить на следующие этапы.

1. Подготовка к запуску задачи на кластере. На этом этапе происходит сборка кода либо конвертация Jupyter-ноутбука в скрип-

ты на языке Python. Также генерируются временные входные данные.

2. Запуск и мониторинг ресурсов, используемых программой. Преподаватель запускает программу, следя при этом за потребляемыми ей ресурсами. Мониторинг ресурсов происходит с помощью инструментов экосистемы Hadoop.
3. Проверка результата работы программы. Просматривается результат работы программы и при необходимости сверяется с эталонным.
4. Code review. Этот этап на данный момент не автоматизирован, поэтому останавливаться на нём не будем.

В таблице ниже показано, сколько времени в среднем тратит преподаватель на проверку одной задачи.

Этап	Затраты времени, с
1	20
2	120 с (зависит от времени работы программы)
3	100

Поскольку в домашнем задании может быть больше 7 задач, время только лишь тестирования (без учёта code review и написания ответа студенту) может превысить нормативное время проверки задания (30 мин). Это создаёт предпосылки для автоматизации процесса.

Автоматизация проверки заданий

Как было показано в [2], единой системы, которая бы охватывала все описанные выше требования, не найдено. Поэтому была разработана система *HJudge* [3] (Роспатент № 216660616). Внедрение HJudge в образовательный процесс улучшило ситуацию с этапом 3, но вместе с тем выдвинуло новые требования к автоматизации проверки. Главными недостатками системы были:

1. Участие преподавателя в процессе проверки. Он должен был запускать тестирование задачи и вести мониторинг использования ресурсов.
2. Постоянные исходные данные. Задача тестировалась на одних и тех же данных от запуска к запуску. Это приводило к тому,

что студент мог подобрать правильный ответ ко многим тестам (кроме тестов на использование ресурсов).

Эти проблемы решены в новой версии HJudge.

Обновление HJudge и его влияние на проверку заданий

Для исключения участия преподавателя в тестировании заданий, система была интегрирована с GitLab CI. Это дало несколько преимуществ.

1. Тестирование задачи запускается автоматически после коммита в репозиторий. По его окончании пользователь видит результат и оценку, но при этом не имеет доступа к тестам.
2. В GitLab есть интерфейс администратора, позволяющий следить за процессом сдачи и собирать статистику.
3. Проекты и учётные записи студентов можно автоматически создавать и удалять, а также блокировать, исключая таким образом сдачу заданий позже срока. Для этого в системе используется GitLab Python API [4].

Система была интегрирована с облачной платформой Everest [5], используемой в качестве прослойки между GitLab и Hadoop-кластером. Это улучшило безопасность тестирования. Everest управляет ресурсами, на которых выполняется приложение. Можно подключить несколько Hadoop-кластеров. При этом Everest будет осуществлять балансировку задач между ними.

На рисунке 1 показана общая архитектура системы HJudge. В качестве HJudge core было взято ядро HJudge 2016 г. Оно описанно в тезисах [2] и [3]. Здесь же стоит только упомянуть о ещё одной доработке. Для автоматизации мониторинга использования ресурсов система использует REST API двух сервисов экосистемы Hadoop: YARN JobHistory [6] и Spark History [7]. Эти службы хранят данные о Hadoop-контейнерах, использованных для задачи.

Коды HJudge на данный момент не выложены в открытый доступ, а находятся в репозитории GitLab. Производится доработка кода с целью публикации на GitHub.

Средние затраты времени при проверке 1 задачи с помощью Hjudge приведены ниже. Для сравнения приведены аналогичные затраты при ручной проверке.

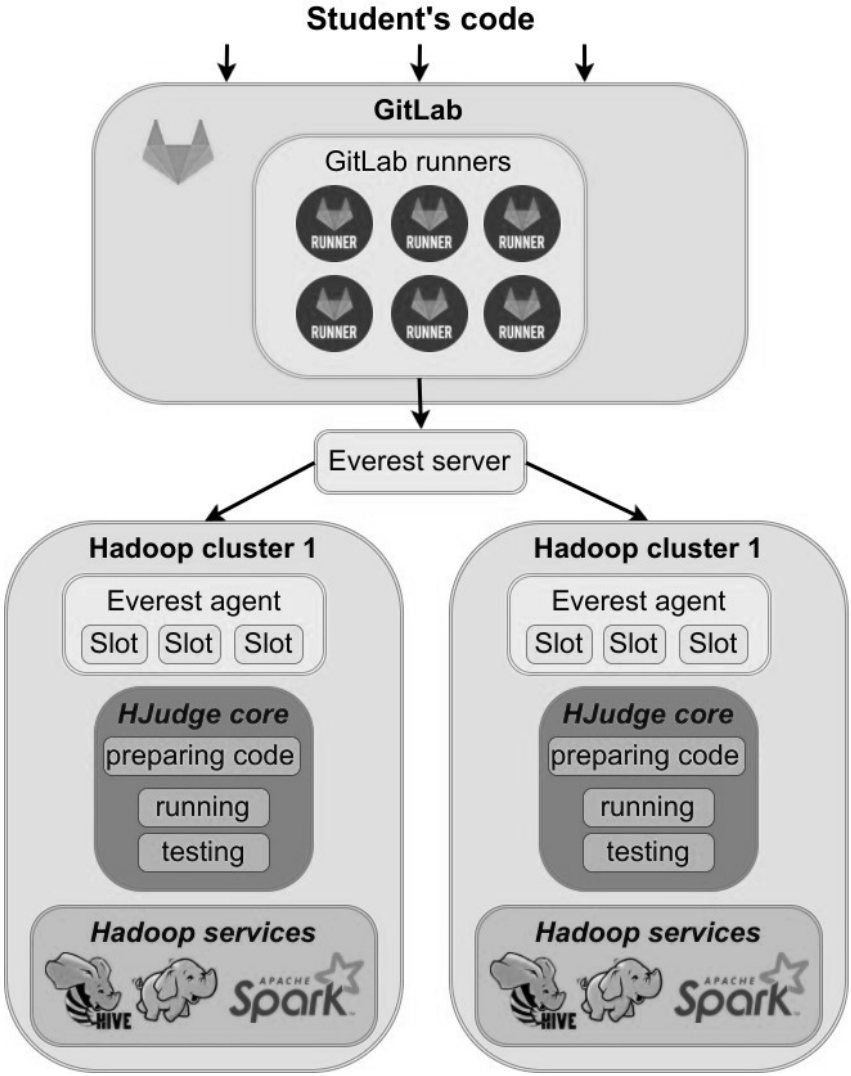


Рис. 1: Архитектура HJudge

Этап	Ручная проверка	HJudge
1	20	1
2	120 с (зависит от времени работы программы)	
3	100	5

Основное преимущество системы не столько в выигрыше во времени тестирования, сколько в освобождении преподавателя от процесса проверки из-за чего студент получает обратную связь сразу по окончании тестирования. Это критически важно для обучения.

Планы по дальнейшему развитию

1. Проверка программ обработки данных в реальном времени.

Большая часть тестирования происходит по окончанию работы программы. Но многие приложения обрабатывают данные, поступающие пакетно в реальном времени.

1. Анализ исходного кода (в частности на плагиат).

Планируется автоматизировать последний этап проверки — code review. В частности, анализ кода позволит обнаружить плагиат. Предполагается использование нейронных сетей [8].

Вывод

Доработка системы HJudge позволила исключить участие преподавателя в проверке заданий, а также ускорить сам процесс проверки.

Литература

[1] Tom White. Hadoop: The Definitive Guide // O'Reilly, — 657.

[2] О. Н. Ивченко, А. А. Драль. Система HJudge или как автоматизировать проверку заданий при изучении работы с большими данными. XII Конференция «Свободное программное обеспечение в высшей школе», М.: Basealt, 2017.

[3] Ивченко О. Н., Драль А. А. Hjudge: система тестирования приложений в экосистеме Hadoop // Сборник научных трудов МФТИ «Модели и методы обработки информации», — 112.

[4] Документация по GitLab Python API [Электронный ресурс]: `http://python-gitlab.readthedocs.io/en/stable/api-usage.html`, время обращения: 18.01.2018

[5] O. V. Sukhoroslov, A. O. Rubtsov, S. Yu. Volkov. Development of distributed computing applications and services with Everest cloud platform // Computer Research and Modeling, — 593–599.

[6] Документация по YARN History Server API [Электронный ресурс]: https://hadoop.apache.org/docs/r2.4.1/hadoop-yarn/hadoop-yarn-site/HistoryServerRest.html, время обращения: 18.01.2018

[7] Документация по Spark History API [Электронный ресурс]: https://spark.apache.org/docs/latest/monitoring.html, время обращения: 18.01.2018

[8] J. Yasaswi, S. Purini, C. V. Jawahar. Plagiarism Detection in Programming Assignments Using Deep Features // Proceedings of the 38th SIGCSE technical symposium on Computer science education, 2007, — 34–38.

Чернышов Л. Н., Лукин В. Н.
Москва, МАИ
http://mai.ru

Контроль и самоконтроль знаний по базам данных

Аннотация

Предлагается подход, основанный на генерации тестовых заданий по дисциплине «Базы данных». Задания формируются по некоторому шаблону и предъявляются студенту.

Контроль знаний реализуется во всех существующих обучающих системах, которые обладают средствами создания учебных курсов и баз тестовых заданий (БТЗ). Однако вопросы формирования заданий, их механизмы и реализация, как правило, прорабатываются слабо. Автоматизация разработки заданий, генерация большого числа различных вариантов — вопросы, важность которых возрастает в современной системе образования.

Компьютерное тестирование имеет ряд недостатков: негативные реакции на представление тестов, влияние предшествующего опыта, воздействие интерфейса тестирующей системы. Суждение об уровне знаний только по ответам проигрывает очной форме собеседования.

Среди типов тестовых заданий более эффективны вопросы с открытым ответом, но автоматическая оценка таких ответов связана с

существенными сложностями, особенно если ответ представлен обычным текстом. Задача упрощается, если ответ формализован. Так, если ответ представлен программой на некотором языке, его правильность проверяется исполнением. Возможность формализации упрощает и задачу генерации заданий. Рассмотрим две близкие системы для подготовки тестовых заданий и проведения тестирования для дисциплины «Базы данных» (БД) по теме «Язык запросов SQL».

Используется следующая технология создания базы тестовых заданий.

Подготовка тестового материала:

- определить предметную область для базы данных;
- создать БД в 3 нормальной форме в среде заданной СУБД;
- определить множество запросов и реализовать их на SQL;
- подготовить тестовые варианты для запросов и загрузить их в БД;
- протестировать реализованные запросы;
- зарегистрировать БД в системе контроля.

Проведение контрольного мероприятия:

- студент регистрируется в системе;
- выбирает предметную область;
- получает вариант задания;
- решает задачу на SQL, вносит полученный запрос в систему и получает результат;
- из системы получает правильный ответ;
- если выборки семантически эквивалентны, задача зачтена, работа заканчивается. В противном случае увеличивается счётчик подходов, и задача решается повторно.

Первая система реализована как web-приложение, что позволяет формировать БТЗ одновременно несколькими преподавателями. Описание БД для каждой предметной области создаётся в виде SQL-скрипта, который может динамически выполнятся как на этапе подготовки БТЗ, так и на этапе тестирования. Для каждой БД создаётся несколько заданий в виде формулировки запроса к БД на естественном языке и на языке SQL.

Для создания и проверки корректности описания БД и запросов автоматически формируется экран с кнопками для создания и выполнения запросов. При выполнении запроса на экран выдаётся его формулировка, соответствующий SQL-запрос, результат запроса в виде таблицы, а также содержимое участвующих в запросе таблиц.

По созданной таким образом БТЗ проводятся контрольные мероприятия или самоподготовка.

Рассматривается два основных типа заданий.

1. Студент составляет SQL-запрос по заданной формулировке на естественном языке и по заданной БД. Для проверки ответа система генерирует БД, выполняет запрос студента и запрос из БТЗ, и сравнивает результаты.

2. Студент заполняет таблицу результата запроса по заданному SQL-запросу. На экране отображаются значения исходных таблиц и запрос. Для проверки система генерирует БД, выполняет запрос и сравнивает результат с тем, что ввел студент.

В первом случае проверяется умение решить задачу, во втором — умение проанализировать решение.

Наиболее трудоемкий этап — это составление вопросов для тестовых заданий и формирование индивидуальных вариантов. Его можно упростить переходом от фиксированных формулировок к параметризированным шаблонам, по которым генератор формирует уникальные задания. Для увеличения числа тестовых заданий в системе предусматривается генерация вариантов путём изменения таблиц. Это может быть как число записей в таблицах, так и значения атрибутов из их доменов. При этом корректируются и формулировки запросов, если в них участвуют изменяемые значения атрибутов.

Генерация содержимого БД (тестовых заданий) может производиться как однократно при создании БТЗ, так и непосредственно при тестировании. Первый вариант используется при групповом контроле, второй при индивидуальном или при самоконтроле.

Во второй системе преподаватель определяет предметную область: её описание и структуру БД. Затем для выбранной предметной области он формулирует запросы и реализует их на SQL. Формирование тестовых данных производится, исходя из характера запросов.

Предлагаемая технология существенно снижает трудоёмкость подготовки материала к контрольным мероприятиям, позволяет проводить контроль в условиях, похожих на производственные. В режиме

самоподготовки студенту может дополнительно предъявляться корректное решение, но нужно, чтобы он понимал, что оно обычно не единственное.

Литература

[1] Рыжов А.А., Чернышов Л.Н., Булыгин А.С. Система автоматической проверки ответов. — Двенадцатая конференция «Свободное программное обеспечение в высшей школе»: Материалы конференции / Переяславль, 27–28 января 2017 года. М.: Basealt, 2017. — 107 с. : ил., с. 22–24

[2] Лукин В.Н. Введение в проектирование баз данных: учебное пособие — М. : Вузовская книга, 2015. — 144 с.

[3] Махлай В.С., Чернышов Л.Н. Web-приложение для проведения контрольных и практических работ по программированию с автоматической генерацией заданий. — Десятая конференция «Свободное программное обеспечение в высшей школе»: Тезисы докладов / Переяславль, 24–25 января 2015 года. М.: Альт Линукс, 2015. — 100 с. : ил., с.86–88.

Сергей Мартишин, Владимир Симонов, Марина Храпченко
Москва, Институт системного программирования РАН, отдел «Теоретической информатики», Российский государственный социальный университет, кафедра информационных систем, сетей факультета информационных технологий

Использование свободного программного обеспечения для решения задач классификации

Аннотация

Рассмотрены задачи дискриминантного анализа и классификации по ряду признаков. Представлена система, реализованная на фреймворке Apache Spark и языке Scala, использующая дискриминантные функции и предназначенная для выявления различных групп рисков. Система может применяться для профессиональной ориентации абитуриентов вузов

На практике достаточно часто возникают задачи, в которых отсутствие методики предсказания возникновения рисков может привести к негативным последствиям. Одним из решений таких задач является

классификация — разбиение совокупности объектов на несколько однородных групп (классов, совокупностей) по одному или нескольким признакам. Дискриминантный анализ здесь используется для принятия решения о том, какие переменные дискриминируют две или более возникающие совокупности. Указанное разделение позволяет в дальнейшем моделировать зависимости внутри каждой группы, при этом для оценки степени риска необходимо проводить исследования на имеющихся данных, чтобы в дальнейшем уже в режиме реального времени выявлять соответствующую группу риска.

Разрабатываемая система дискриминантного анализа основана на использовании дискриминантных функций и предназначена для выявления различных групп рисков.

В статистике множество известных исходов принято называть значениями зависимой переменной. Значение зависимой переменной может быть предсказано на основе факторов (предикторов). Для отнесения риска к соответствующей группе необходимо использование дискриминантных функций, рассчитанных на основе имеющихся данных, что позволит в дальнейшем осуществлять прогнозирование на основе предикторов.

Одной из наиболее популярных областей применения дискриминантного анализа является банковская сфера, например, оценка финансового состояния клиентов при выдаче кредита. Банк классифицирует клиентов на надежных и ненадежных по ряду признаков. Также дискриминантный анализ успешно применяется для прогнозирования оттока клиентов, в медицине, строительстве, анализе экологической ситуации и пр. В системе высшего и среднего образования такого рода анализ может применяться для профессиональной ориентации абитуриентов вузов.

Однако в каждой конкретной области (банковское дело, медицина, строительство, образование и пр.), существуют строгие стандарты и регламенты для определения каждой группы риска. Основной задачей разрабатываемой системы анализа данных является разделение имеющихся статистических данных на группы, о поведении которых накоплено достаточное количество статистической информации. Тогда в дальнейшем при поступлении новых данных, они могут быть отнесены к определенной группе и оценены с точки зрения возникновения рисков.

Первоначально система анализа данных должна пройти обучение на подготовленных экспертами статистических данных с известным

исходом. Указанное обучение аналогично нейросетевому «обучению с учителем».

В частности, при применении данной системы для профориентации абитуриентов и оценки их мотивации, в качестве статистических данных для обучения эксперты могут использовать такие показатели потенциальных абитуриентов как успеваемость по профильным дисциплинам, сведения о рейтинге школы, успехи во внеучебной деятельности (кружки, олимпиады), профессии родителей и т.п.

Необходимо отметить, что вычисление дискриминантных функций на этапе обучения системы требует значительных вычислительных мощностей, однако в дальнейшем работа с поступающими данными происходит в режиме реального времени и полностью в автоматическом режиме.

По мере накопления новых данных обучение системы необходимо повторять, при этом с увеличением объема базы данных точность дискриминантных функций возрастает. Особенностью предлагаемой системы является полное отсутствие ручного труда экспертов при построении дискриминантных функций.

Развитие свободного и свободно распространяемого программного обеспечения, такого как фреймворк Apache Spark и язык Scala, направленного на распределенную разработку больших данных, позволяет обеспечить практическую реализацию рассматриваемой системы.

Фреймворк Apache Spark — это фреймворк с открытым кодом, в котором имеется механизм распределения программ по кластеру машин. То есть — Spark это быстрая кластерная вычислительная система для обработки больших данных, обладающая линейной масштабируемостью и обеспечивающая отказоустойчивость. Фреймворк предоставляет высокоуровневые API-интерфейсы со Scala, Java, Python, R для анализа данных. Кроме того, Spark обладает новыми возможностями работы с оперативной памятью за счет введения понятия RDD (Resilient Distributed Dataset — устойчивого распределенного набора данных). Такой набор данных позволяет выполнять обработку данных параллельно. RDD распределяется по кластеру в виде совокупности секций, содержащих подмножество данных. Spark обрабатывает секции параллельно, а данные внутри секций — последовательно [1, 2].

Scala — язык программирования, предназначенный для компонентного программирования, то есть создания независимых модулей исходного кода программы, предназначенных для повторного исполь-

зования и развёртывания. Scala сочетает возможности функционального и объектно-ориентированного программирования. В Scala все является объектом (включая числа и функции), а любая операция — вызовом метода [3].

Естественная связка фреймворка Apache Spark и языка Scala позволяет ускорить обработку данных за счет механизма кластерных вычислений, на поддержку которых ориентирован язык Scala. Использование распределенных вычислений позволяет получать эффективные дискриминантные функции для классификации по группам риска.

Литература

[1] Лезерсон У. Spark для профессионалов. Современные паттерны обработки больших данных/ У. Лезерсон, С. Риза. — Изд-во Питер, 2017. — 272 с.

[2] Apache Spark [Электронный ресурс] / Режим доступа: http://spark.apache.org, свободный. (Дата обращения: 11.01.2018.).

[3] Scala [Электронный ресурс] / Режим доступа: http://www.scala-lang.org, свободный. (Дата обращения: 11.01.2018.).

Татьяна Сундукова
Тула, Тульский государственный педагогический университет им. Л.Н. Толстого

Особенности LMS с открытым исходным кодом для высшей школы

Аннотация

В докладе рассматриваются преимущества и недостатки LMS с открытым исходным кодом, анализируются слабые и сильные стороны двух самых распространённых LMS с открытым исходным кодом, такие как Canvas и Moodle.

В настоящее время высшие учебные заведения сталкиваются со многими трудностями и сложными проблемами, включая увеличение числа учащихся в своих образовательных программах и расширение инфраструктуры — такой как система управления обучением

(Learning Management System – LMS) — с целью привлечения учащихся, диверсификации занятий, поддержки обучения студентов и преподавательского состава. Для реализации данных целей и создания благоприятной учебной среды в высшие учебные заведения внедряют различные типы LMS, такие как патентованные LMS, LMS с открытым исходным кодом и облачные LMS.

Преимущества LMS с открытым исходным кодом:

- Экономическая выгода: в целом, такие LMS, если не бесплатны, то гораздо дешевле патентованных LMS.
- Инновации пользователей: LMS с открытым исходным кодом предоставляют возможность совместного кодирования системы со своими пользователям, поэтому у них больше свободы и гибкости в инновациях.
- Увлеченное и совместное сообщество: LMS поддерживаются сообществами, которые открыты для новых идей.
- Безопасные и надежные: LMS с открытым исходным кодом рассматриваются как защищенные LMS.

Недостатки LMS с открытым исходным кодом:

- Отсутствие поддержки и обслуживания: сервис является ключевым вопросом использования данных LMS. Хотя они имеют лояльную и привлекательную сеть онлайн-сообществ, производители LMS могут не часто получать адекватную обратную связь по своим вопросам.
- Большая зависимость от сети онлайн-сообществ: все еще спорно, будут ли изменения в исходном коде приемлемы или ограничены для улучшении таких LMS. Кроме того, нет гарантий, что пользователи готовы поделиться своим нововведением с другими пользователями.
- Меньше ответственности: никто не несет ответственность, если что-то не так с рассматриваемыми LMS.

Рассмотрим достоинства и недостатки двух самых распространённых LMS с открытым исходным кодом.

Достоинства	Недостатки
Canvas	
Доступность: Золотой сертификат от Национальной федерации слепых Современный пользовательский интерфейс: современный интерфейс пользователя во всем приложении Простота администрирования: простота в освоении и использовании возможностей и функций Замечательное удобство: преподаватели и студенты могут сразу использовать приложение Эффективный рабочий процесс: простота настройки задач инструктора в сочетании с деятельностью учащихся Plug-and-play совместимости учебных средств: простые обучающие инструменты взаимодействия могут быть добавлены в систему Сильная поддержка сотрудничества: поддерживает групповую работу, предоставляя совместное рабочее место Быстрые инновации: новые функции и нововведение выпускаются каждые 2 недели Сосредоточенность студентов: учащиеся могут самостоятельно контролировать свое обучение	Разрешение: разрешения устанавливаются на уровне организации и не настраиваются на уровне сайта Управление разделами: преподаватели не могут публиковать объявления или работать в определенных разделах Внутренняя система электронной почты: она не поддерживает обмен текстовыми сообщениями, поиск или сортировку Упрощенный инструмент рубрик: создание рубрик громоздко и дескриптор ячейки имеет всего несколько слов Упрощенный редактор расширенного текста: расширенные параметры редактирования скрыты в системе Нет предпочтений в отношении конфиденциальности: студенты не могут скрывать свои имена и профиль
Moodle	
Экономически эффективна: экономически выгодна, даже если могут возникнуть дополнительные затраты Простота использования: несмотря на множество наборов инструментов, таких как управление курсом и коммуникациями, инструменты регистрации и зачисления, варианты управления пользователями, все функции просты и эффективны Настройка: позволяет настраивать и контролировать опыт преподавателей и студентов Быстрое развертывание: система предоставляет возможность быстрого развертывания, предоставляя онлайн программу обучения на основе проекта	Доступность: нет сертификата от Национальной федерации слепых Большая зависимость от сторонних дополнений: высокая зависимость от стороннего ПО увеличивает временную задержку и нагрузку для обновления LMS Недостаточное капиталовложений в техническое обслуживание: не хватает масштаба для инвестиций в техническое обслуживание

По мере того, как LMS становятся все более важными для повышения качества преподавания и обучения в высшем образовании, существует значительная потребность в выборе подходящая LMS в высших учебных заведениях для повышения эффективности преподавания и обучения студентов. Кроме того, как преподаватели, так и студенты, как правило, довольны основными функциями и функциями LMS, но они все еще нуждаются в более сильной поддержке

для работы с LMS. Обучение должно побуждать преподавателей и студентов активно использовать инновации и совершенные функции LMS, чтобы они могли чувствовать себя более уверенно в использовании LMS. Наряду с текущими тенденциями в области LMS многие пользователи и заинтересованные стороны представляют следующее поколение LMS в области высшего образования. Рассматривая общую картину, функции LMS следующего поколения, вероятно, будут включать: взаимодействие и интеграцию с другими системами, автоматическую и расширенную аналитику обучения, персонализацию создания собственных траекторий обучения и сотрудничество на нескольких уровнях.

А. Н. Пустыгин, А. А. Ковалевский
Челябинск, Челябинский государственный университет
http://www.csu.ru/

Алгоритмы и программный инструмент построения метрики эквивалентных представлений для исходных текстов программ

Аннотация

Рассматривается прототип программного инструмента статического анализа программных систем, основанный на специальном наборе данных, полученном из исходного текста программ с помощью компилятора с открытым исходным кодом.

Задача статического анализа программных систем является одной из традиционных способов улучшения программ [1].

Ранее [2] было предложено универсальное промежуточное представление (УПП) открытого исходного текста, предназначенное для последующего преобразования, анализа и извлечения информации из исходного кода программ. Это представление по смыслу является сериализованным синтаксическим деревом разбора исходного текста. На основе этого набора данных был построен прототип генератора системы эквивалентных представлений, предназначенных для решения некоторой совокупности задач над исходным текстом. Эквивалентное представление есть результат преобразования и фильтрации УПП, в простейшем случае срез дерева по какому-либо условию.

Рассмотренный прототип позволяет получать линейные и нелинейные эквивалентные представления исходного текста из его УПП, выполнять анализы над этими представлениями и получать срезы этих представлений [3]. Помимо простых преобразований, возможно получение объединенных представлений с заданными общими параметрами среза. Правила построения представлений для реализованного прототипа задаются в конфигурационном файле.

При анализе эквивалентный представлений одним из шагов часто является получение метрических оценок исходного текста по выбранной метрике, а одной из известных способов определения метрик является температурная диаграмма [4].

Температурной диаграммой исходного текста будем считать зависимость условного количественного параметра (температуры) от номера строки исходного текста.

Температурная диаграмма как эквивалентное представление исходного текста может быть построена для любого другого эквивалентного представления этого же текста (метрика второго уровня) [7].

Приведем пример построения ТД, используя в качестве базы универсальное промежуточное представление в виде AST[5]. Таким образом, метрика строится на основе структуры графа дерева разбора исходного текста.

Правила назначения температур строкам

Пользователь задает вес для выбранных элементов УПП в соответствующем фильтре конфигурационного файла.

В соответствии с весом фильтруемого элемента УПП для каждой строки вычисляется ее суммарная температура, которая складывается из:

- Температура основания — вес структуры AST, в которую вложен первый взвешенный элемент строки;
- Собственная температура строки — суммарный вес всех входящих в нее взвешенных элементов.

Формат выходного XML документа представляет собой последовательность тегов <Line> с обозначенной в их атрибутах температурой, объединенных в блоки <Method>.

Так как имеется возможность произвольного назначения весов каждому фильтруемому элементу УПП, а значит и каждой синтаксической конструкции исходного текста, можно получить большое число разнообразных метрик в виде температурных диаграмм.

«Температуризация» может быть использована и для других эквивалентных представлений, например, для построения температурной карты диаграммы классов.

«Температурный» анализ потока управления алгоритмически перегруженных строк и сложных блоков исходного текста может быть использован для полуавтоматического поиска участков кода, которые требуют рефакторинга, например, потенциально «медленные» участки кода. Методика ранжирования весов тех или иных синтаксических конструкций лежит на исследователе. Использование УПП, которое в свою очередь основано на AST, в данном случае играет особую роль, так как «на глаз» в исходном тексте перегруженные операторы выявить сложно, поскольку метрика такой оценки интуитивна, а в УПП и, следовательно, в температурной диаграмме они отражены. Перегруженный конструктивно оператор по смыслу и весу равнозначен методу, так что оценка сложности исходного кода в данном случае будет более точно соответствовать оценке сложности кода исполняемому.

Рассмотрим пример вычисления температуры для проекта с открытым исходным кодом pigixml — парсер форматированных текстов по стандарту xml.

Строка 514 исходного текста в файле «pugixml.hpp» [6] имеет «температуру» 16 (см. рис. 1), которая складывается из следующих составляющих:

- температура основания 5, складывающаяся из вложенности в цикл while (вес 3, 506 строка), в условный оператор if (вес 1, 510 строка) и еще один if (вес 1, 511 строка);
- собственная температура строки 11, складывающаяся из весов конструкций:
  - цикл while (вес 3)
  - вызов метода next_sibling() (вес 2)
  - неявный вызов оператора operator!() (вес 2)
  - вызов метода parent() (вес 2)
  - неявный вызов оператора operator=() (вес 2)

Наличие операторов «operator!()» и «operator=()» обусловлено тем, что они перегружены для типа xml_node, которому принадлежит объект cur (см. 504 строка). Перечисленные элементы проиллюстрированы деревом синтаксического разбора этого участка исходного текста рис 2.

```
500(16) template <typename Predicate> xml_node find_node(Predicate pred)
502(4) if (!_root) return xml_node();
504(2) xml_node cur = first_child();
506(3) while (cur._root && cur._root != _root)
508(5)  if (pred(cur)) return cur;
510(10)  if (cur.first_child()) cur = cur.first_child();
511(11)  else if (cur.next_sibling()) cur = cur.next_sibling();
514(16)   while(!cur.next_sibling() && cur._root!=_root) cur=cur.parent()
516(10)   if (cur._root != _root) cur = cur.next_sibling();
520(3) return xml_node();
```

Рис. 1: Температурная диаграмма фрагмента исходного текста

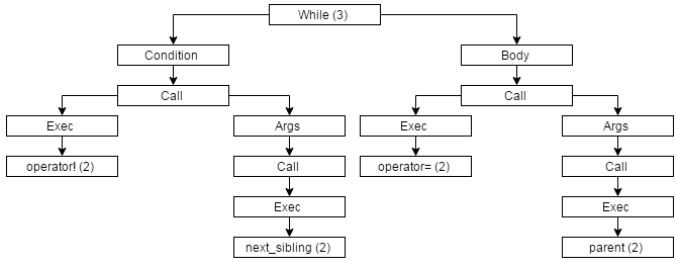


Рис. 2: Синтаксическое дерево разбора строки исходного текста 514 в контексте потока управления

Выводы

Рассмотренный прототип инструмента позволяет получать линейные и нелинейные эквивалентные представления исходного текста из его УПП с целью последующего анализа над этими представлениями и получать срезы этих представлений, а также получать объединения выбранных представлений с заданными общими параметрами среза. Одним из оснований анализа является введение метрики для количественных оценок в процессе анализа текста.

Литература

[1] Инструменты статического анализа кода // Электронный ресурс (дата обращения: 10.01.2018) `http://www.viva64.com/ru/t/0074/`

[2] Зубов М. В., Построение универсального представления графа потока управления для статического анализа исходного кода /Зубов М.В., Пу-

стыгин А.Н., Старцев Е.В. // В книге: Девятая конференция «Свободное программное обеспечение в высшей школе» Тезисы докладов. 2014. с. 46–51.

[3] Ковалевский А.А. Построение эквивалентного представления исходных текстов программ в форме, пригодной для выполнения анализов потока данных в потоке управления / Ковалевский А.А., Пустыгин А.Н., Ошнуров Н.А.// ИЗВЕСТИЯ Юго-Западного государственного университета. Серия Управление, вычислительная техника, информатика. медицинское приборостроение. 2015. № 1 (14) с. 28–34

[4] Ковалевский А.А. Алгоритмы и программный инструмент построения эквивалентных представлений исходных текстов программ / Ковалевский А.А., Пустыгин А.Н.// Инфокоммуникационные технологии. Поволжский государственный университет телекоммуникаций и информатики. 2015 Том:13 № 4, С.398-404

[5] Пустыгин А.Н. Прототипы построителей промежуточных представлений исходных текстов программ, основанные на компиляторах с открытым исходным кодом / Пустыгин А.Н., Тарелкин Б.А., Ковалевский А.А., Огуречникова Е.А., Десинов А.В., Ошнуров Н.А., Старцев Е.В., Зубов М.В. // В книге: Свободное программное обеспечение в высшей школе тезисы докладов. НОУ «ИПС-Университет г. Переславля им. А. К. Айламазяна», Институт Программных Систем РАН, Институт Логики, ALT Linux. 2012. С. 82-86.

[6] pugixml v1.2[Электронный ресурс]. URL: `https://pugixml.org/` (дата обращения: 10.01.2018).

[7] Пустыгин А. Н. Построение эквивалентного представления исходных текстов программ в форме, пригодной для выполнения анализов потока данных в потоке управления / Ковалевский А.А., Пустыгин А.Н., Ошнуров Н.А. // Известия Юго-Западного государственного университета. Серия: Управление, вычислительная техника, информатика. Медицинское приборостроение. 2015. № 1 (14). с. 28–34.

Губина Татьяна
г. Елец, ООО «Базальт СПО»

Требования федеральных государственных образовательных стандартов и дистрибутив Альт Образование: текущее состояние

Аннотация

В работе представлены результаты по проверке соответствия программных продуктов, включённых в дистрибутив Альт Образование, требованиям федеральных государственных образовательных стандартов (ФГОС) основного (общего) образования (ОО).

В последние годы нас захватили в свои сети информационные технологии и с каждым годом мы всё глубже погружаемся в них. Практически во всех учебных кабинетах в школах установлены мультимедийные комплексы, позволяющие разнообразить уроки, повысить уровень наглядности. Весь документооборот перешел в электронный формат. Вся информация о деятельности учителей и в целом школы освещается в сети Интернет.

Без компьютеров мы никуда. От удобства и функциональности операционной системы, её ориентированности на потребности участников образовательного процесса зависит комфортность работы учителя, администрации, скорость и качество выполняемой работы.

За многие годы большинство из нас привыкли работать на компьютерах под управлением операционной системы Windows. И мысль о переходе на другую операционную систему, например, Linux, обычно всегда вызывает негативную реакцию педагогов, поскольку все считают, что это сложно. Кроме того, возникают сомнения по поводу возможности реализации федеральных образовательных стандартов на базе дистрибутивов семейства Linux.

Linux относится к категории свободного программного обеспечения, то есть обеспечения, которое дает право неограниченное количество раз ее устанавливать, запускать. Предоставляет нам такие возможности как свободно использовать ее с любой целью, свободно распространять копии, что несомненно дает преимущество перед операционными системами семейства Windows.

Бытует такое сложившееся мнение, что Windows — самая удобная система, а Linux — это сложно и неудобно. Какой смысл тратить время и разбираться в Linux? Зачем осваивать педагогу Linux?

Наше государство поставило цель: обрести технологическую независимость от зарубежных производителей программного обеспечения и компьютерного оборудования. С рамках реализации этой цели была разработана программа импортозамещения, подкрепленная законодательными актами, которые, например, предписывают всем органам государственной власти уже к концу 2018 года перейти на отечественное ПО. Следующие на очереди — госкорпорации, государственные организации и ведомства, в том числе и образовательные организации.

В ноябре 2015 года российское правительство своим постановлением запретило российским госорганам и госучреждениям закупать иностранное программное обеспечение при наличии аналогов, указанных в реестре отечественного ПО [1]. В настоящее время в реестр входит около 3 тыс. наименований российского софта. Это означает, что повсеместно будут внедряться операционные системы на базе ядра Linux.

Поэтому сейчас актуально и остро опять стоит вопрос о подборе программных продуктов, с одной стороны, удовлетворяющих требованиям ФГОС, с другой стороны, включенных в реестр отечественного ПО.

Что важно на наш взгляд при выборе операционной системы и образовательных программ?

- Включен ли дистрибутив в Единый реестр российских программ и баз данных?
- Есть ли версия для работы с персональными данными, сертифицированная ФСТЭК России?
- Ориентирован ли он на образование?
- Соответствует ли дистрибутив требованиям Распоряжения Правительства РФ от 18.10.2007, 1447-р [2]?
- Условия распространения (лицензия)?
- Есть ли возможность построения образовательного процесса в школе в соответствии с требованиями федеральных государственных образовательных стандартов (ФГОС)?

Ответ на последний вопрос потребовал его детальной проработки. Была проделана следующая работа:

- выполнено сопоставление перечня и описания основных элементов, формируемых ИКТ-компетенций учащихся [3], и возможностей программных пакетов, включённых в дистрибутив Альт Образование;
- проверено соответствие «Образовательная цель (по предметам) → программное обеспечение из пакета Альт Образование» (уровень основного общего образования (5-9 классы));
- проверено соответствие перечня программных инструментов для обеспечения образовательного процесса в школе (5-9 классы) с включённым в пакет Альт Образование программным обеспечением.

Как оказалось, в дистрибутив входит все необходимое программное обеспечение для выполнения требований ФГОС ОО, что, в значительной мере, способствует востребованности данного продукта среди образовательных организаций.

Результаты проделанной работы будут представлены на конференции.

Список источников:

1. Единый реестр российских программ для электронных вычислительных машин и баз данных [Электронный ресурс]. URL: `https://reestr.minsvyaz.ru/reestr/`
2. Распоряжение Правительства РФ от 18 октября 2007 г. № 1447-р ГАРАНТ.РУ [Электронный ресурс]. URL: `http://www.garant.ru/products/ipo/prime/doc/92021/\#ixzz54ZecHkCq`
3. Примерная основная образовательная программа основного общего образования // Одобрена решением федерального учебно-методического объединения по общему образованию (протокол от 8 апреля 2015 г. № 1/15).
   URL: `http://минобрнауки.рф/документы/938/файл/4587/POOP\_OOO\_reestr\_2015\_01.doc` (п.2.1.7 Программы развития универсальных учебных действий).

Проскурнёв Артём
Москва, ГБОУ Школа № 830
Проект: ОС Комета https://www.altlinux.org/Комета

Внедрение и использование СПО в московской школе

Аннотация

Данный доклад призван познакомить слушателей с опытом внедрения СПО, начиная с операционной системы на базе Linux в одной из московских школ. Освещаются проблемы использования, принятия и совместимости, а так же варианты решения этих проблем. Так же затрагиваются не решённые проблемы и предлагаются пути для совместного решения.

Оснащённость школы Москвы, в целом, несколько отличаются по оснащённости от школ многих других городов. Чаще всего все новейшие разработки для школ, вроде интерактивных досок и тому подобного появляются в первую очередь именно в московских школах и проходят тестирование. Импортозамещение в виде использования отечественных операционных систем хоть и можно считать инновацией, но конкретно этой инновации сопротивляется вся система образования Москвы, начиная с ДОгМ.

Ни для кого не секрет, что самая первая и основная причина по которой импортозамещение ОС не принимается пользователями — это банальная привычка. Все привыкли к самой популярной системе, как они сами говорят, и просто никак не могут привыкнуть к чему-то новому. В большинстве случаев, данные слова пользователей относятся не к операционной системе. Почти все пользователи говорят, что у них нет Windows, когда у них стоит, например, Windows 7 + LibreOffice. Если бы в Linux этим пользователям включили Microsoft Office, то они бы и не заметили что работают в какой-то другой системе.

Итак, основными пунктами спокойного перевода пользователей на другую ОС являются:

1. Наличие офисного пакета, желательного выглядящего как MS Office;
2. Внешний вид ОС похожий на привычную систему;
3. Наличие браузера, которым использовался до перехода;

4. Воспроизведение аудио и видео файлов в чём-нибудь знакомом вроде VLC плеера;
5. Простая программа для сканера;
6. Использование принтера без каких-то проблем.

Оболочка KDE почти полностью решает 2 пункт. Пользователя видят, что кнопка «Пуск» там же, часы там же, приложения там же и рабочий стол можно использовать так же.

Почти все браузеры кроссплатформенные, поэтому установка привычного браузера так же не является проблемой. Это же касается и VLC плеера.

Среди многих программ для сканирования, после двух лет тестов и проб выбрана программа Simple Scan, которая является максимально наглядной и простой из всех опробованных вариантов. Пользователи с радостью переходят на использование именно Simple Scan, т. к. даже под Windows программы сканирования чаще всего несколько более сложные.

Однако, проблемы остались довольно серьёзные.

Принтеры, за очень редким исключением, спокойно работают в ОС Linux. Однако до сих пор существует проблема, которая сильно усложняет поддержку и работу с большим количеством рабочих мест. Это проблема деактивации принтера при малейшей проблеме, вроде застрявшей бумаги. Поставить галочку «активен» обычный пользователь не может, т. к. требуется пароль root. Делать для каждого пользователя специальный ярлык для конкретного принтера при большом количестве рабочих мест очень и очень сложно.

Второй серьёзной проблемой был и остаётся LibreOffice. На самом деле, всё что нужно пользователю в нём, конечно, есть. Однако, вид панели инструментов, как в MS Office для LibreOffice до сих пор только экспериментальный и не предлагается ни в одном из дистрибутивов. Надо заметить что внешний вид тут даже не настолько критичен, по сравнению с проблемой презентаций. Презентации, за редким исключением, либо не будут работать так как были сделаны в MS Office и перенесены, либоб вообще не будут работать. На текущий момент имеются как минимум две большие проблемы: зависание при попытке использовать в презентации видеофайл (при использовании LibreOffice в Windows такой проблемы нет) и невозможность сохранить слайд со звуков в презентации (сохранить — закрыть — открыть и звука нет).

По принципу «критикуешь-предлагай», имеются, конечно, и предложения по двум основным проблемам (проблем, конечно, больше, но с ними можно жить и о них можно говорить совсем отдельно). Было бы очень хорошо сделать так, чтобы устанавливать, активировать и настраивать принтеры пользователи могли самостоятельно. Собрать для этого пакет или сделать какую-то настройку. Тогда при большом количестве сильно разнесённых в пространстве рабочих мест намного проще было бы осуществлять поддержку пользователей, например, по телефону.

По LibreOffice можно сказать только, что идеальным решением было бы в Альтлинукс или Базальт создать поддержку конкретно LibreOffice. Это является основой импортозамещения, поэтому востребованность просто огромна. Основные функции — держать постоянно руку на пульсе:

1. Принимать информацию от любых пользователей, даже самых неопытных;
2. Писать о найденных проблемах в багтрекер грамотно так, как никакой пользователь этого действительно грамотно сделать не сможет;
3. Теребить разработчиков в багтрекере, если проблема действительно критичная для импортозамещения.

Надеюсь, совместными усилиями, удастся сделать так, чтобы импортозамещение действительно удалось!

Алексей Драгунов
Псков, ГБУ ПО «Региональный центр информационных технологий»
Проект: Информационная образовательная среда Псковской области
www.pskovedu.ru

Информационная образовательная среда региона, основанная на свободном программном обеспечении: технологии и содержание

Аннотация
В статье рассмотрен 15-летний опыт, особенности и некоторые результаты создания на базе свободного программного обеспечния в

Псковской области комплексного решения, объединяющего все образовательные организации и органы управления образованием

Начало развития информационных системы в сфере образования Псковской области было обусловлено участием региона в эксперименте по введению ЕГЭ в 2002 году. У субъектов Российской Федерации есть возможность выбора технологии ведения базы данных для проведения ЕГЭ, включавшей на начальном этапе минимальные сведения, необходимые для планирования экзамена, его проведения и последующей обработки результатов. В 2002 году эта база называлась «региональная база данных ЕГЭ» (РБД ЕГЭ). С 2003 года наши специалисты подключились к работе по формированию единых требований к составу и формату РБД ЕГЭ в рамках участия в проектах, направленных на развитие технологии ЕГЭ, проводимых Рособрнадзором.

В рамках исследований *по мониторингу содержания, условий и результатов обучения* (мониторингу СУРО), проводившихся с нашим участием был доработан формат данных для обеспечения формирования сведений о связи обучающихся и учителей-предметников, используемых программах и учебниках, школьных результатах успеваемости (итоговых оценках, полученных без учета ЕГЭ). В базе данных появились учебные группы и привязанные к ним сведения о содержании и условиях обучения. Этот расширенный вариант РБД ЕГЭ был положен в основу единой системы, которая в дальнейшем получила название «Открытая школа».

С самого начала нами было принято решение об использовании в сфере образования региона свободного программного обеспечения, в том числе — при разработке собственных региональных решений. Первый вариант региональной системы представлял собой WEB-приложение на PHP, работающее под управлением Apache. В качестве базы данных была выбрана СУБД Firebird. Web — приложение обеспечивало просмотр данных, отчетов и обмен информацией между клиентским приложением и центральной базой данных в пакетном режиме (путем импорта и экспорта csv — файлов с использованием web — формы).

Подсистема, пришедшая на смену первого решения, разработанного нами в 2003 году, называется сегодня «Мониторинг образования». Постепенно стало понятно, что структуры первичных данных и создаваемая на их основе отчетность постоянно расширяются, появляются новые процедуры и процессы, которые должны охватываться

Системой. В результате было принято решение о разработке гибкой платформы, позволяющей «налету» вносить изменения в действующую распределенную информационную систему, учитывая изменения внешних требований. При разработке новой версии системы, мы отказались от контроля гарантированной целостности данных, обычно реализуемой средствами реляционной СУБД, была разработана структура метаданных, которые позволяли описать: структуру базы данных, правила их представления в пользовательском интерфейсе, группы пользователей и права доступа представителей этих групп к информационным объектам системы, правила предоставления информации внешним системам, отчеты. Была разработана модель, описывающая работу системы распределенных приложений, взаимодействие между которыми реализуется как в пакетном режиме, который с успехом использовался раньше, так и с использованием сервисов. Метаданные и данные системы представляются в виде xml-файлов одного формата, поскольку модели представляются в виде схемы связанных таблиц, данные в которых полностью определяют поведение распределенной информационной системы. Таким образом, приложение представляет собой три xml-файла, первый из которых содержит метаописание информационной системы, второй — нормативно-справочную информацию (НСИ), третий — данные. Разделение на НСИ и остальные данные носит условный характер и сделано для удобства.

Выбранный подход позволил создавать гибкие системы, рабочий прототип которых возможно предъявлять заказчику на следующий день после первого обсуждения, но, как следствие — обладающие типовым интерфейсом, ориентированным на ведение учета. Платформа получила название Integrics x1. Ее ядро для организации учета получило название *x1.db* и включает в себя следующие подсистемы: контроллер базы данных PostgreSQL или Firebird (*node*), оффлайн-клиент и конструктор метаданных (*gatherer*), онлайн-клиент (*web — point*).

Для решения задач, выходящих за рамки стандартной учетной парадигмы, были разработаны:

- подсистема управления проектами и задачами, выполняющая функции единого коммуникационного центра, в том числе — bug tracking, help desk, project management (*x1.communicator*);

- подсистема электронного документооборота: автоматизация стандартных задач документооборота в организации (*x1.docflow*);

- портал: ведение «кольца сайтов», например — системы образования региона, с размещением в единой системе сайта и информационной карты каждой организации или структурного подразделения, организации работы с функциями x1.db в кастомизированном интерфейсе (*x1.portal*).

Подсистема электронного документооборота работает с базой данных PostgreSQL, портал может использовать для хранения структуры Firebird или PostgreSQL, а для хранения контента — MongoDB.

Для обеспечения разработки кастомизированных нестандартных решений платформа включает в себя фреймворк x1.db.php, реализующий все основные функции по работе с моделями x1.db. Он используется как для разработки плагинов к порталу, так и отдельных приложений.

Все подсистемы, за исключением x1.docflow, разработаны на PHP 5.6. Завершается их перенос на PHP 7.1. Для формирования отчетов используется сервер и дизайнер отчётов BIRT (`http://www.eclipse.org/birt/`), написанный на java, но также возможно формировать HTML-отчёты и отчёты в формате Calc с использованием PhpSpreadsheet (`https://github.com/PHPOffice/PhpSpreadsheet`). Отчёты, сгенерированные с использованием BIRT встраиваются в систему с помощью PHP-Java-Bridge (`http://php-java-bridge.sourceforge.net`).

AuthServ — подсистема платформы, отвечающая за идентификацию и аутентификацию пользователей. Разработана на NodeJS, обеспечивает использование технологии SSO и авторизацию для всех подсистем платформы, а также для LMS Moodle и ряда внешних систем, используемых в информационной среде региона для обеспечения электронного обучения. AuthServ поддерживает интеграцию с ЕСИА Ростелекома.

Региональный сегмент межведомственной системы «Контингент» реализован на java. Он, с одной стороны, взаимодействует со всеми основными функциональными подсистемами, перечисленными далее, а с другой — с федеральным сегментом системы «Контингент». В качестве собственной базы данных «Контингент» использует PostgreSQL.

В таблице 1 приведен перечень и краткое описание функциональных подсистем, созданных с использованием свободного программно-

го обеспечения и обеспечивающих функционирование информационной образовательной среды Псковской области.

Таблица 1. Подсистемы РИС «Открытая школа»

№	Наименование модуля	Назначение подсистемы и некоторые функции
1	Мониторинг образования	Ведение сведений о контингенте общеобразовательных организаций Планирование проведения ЕГЭ, ОГЭ, ГВЭ; Формирование базы данных содержания, условий и результатов обучения (СУРО); Получение информации каждой ОО и МОУО о результатах оценочных процедур; Мониторинг поступления выпускников общеобразовательных организаций (продолжения обучения и работы); Мониторинг ИКТ, планирование и мониторинг поставки оборудования; Мониторинг качества доступа в Интернет; Мониторинг отдельных статистических показателей; Подготовка и проведение регионального этапа Президентских состязаний школьников; Формирование отчетов, в том числе актов сверки данных по планированию ГИА и РКМ Контроль формирования данных.
2	Реестр образовательных организаций и органов управления образованием	Внесение, корректировка, импорт и экспорт данных об образовательных организациях и органах управления образованием, в том числе - имплементация запросов на изменение ключевой информации, поступающих от ОО и МОУО, автоматическая фиксация изменений в специальном журнале.

3	Открытые данные	Ведение базы данных сведений и документов, обязательных для публикации образовательными организациями в интернет, в том числе — на сайтах образовательных организаций; Предоставление внешним системам доступа к опубликованным данным, в том числе с использованием web-сервисов в формате json для их встраивания в официальные сайты ОО; Обеспечение web-доступа к сведениям, сформированным образовательными организациями по адресу org<код ОО>.pskovedu.ru Поиск через единую форму поиска информации на портале; Отображение образовательных организаций на карте области с возможностью перехода по щелчку к карточке соответствующей ОО.
4	Приемка образовательных организаций к новому учебному году	Подсистема предназначена для автоматизации процесса планирования и проведения приемки всех типов ОО к новому учебному году в соответствии с методическими рекомендациями Минобрнауки по организации проведения приемки организаций, осуществляющих образовательную деятельность, к началу учебного года для обеспечения: — прозрачности и открытости работы приемочных комиссий; — доступности результатов приемки в интернет, в том числе выявляемых замечаний; — контроля за устранением выявленных в ходе приемки проблем в установленные сроки.
5	Государственные и муниципальные услуги	Подсистема предназначена для ведения электронной очереди в дошкольные образовательные организации, ведения электронных журналов и дневников, предоставления других услуг в сфере образования в электронном виде.
6	Региональная информационная система «Оценка качества образования» (РИС ОКО)	Подсистема предназначена для организации единого информационного пространства оценки учебных достижений обучающихся на уровне школы, муниципалитета и региона, информационной поддержки построения индивидуального обучения и повышения квалификации педагогов.

7	Региональная система электронного обучения	Подсистема предназначена для организации и сопровождения электронного обучения школами, а также областными Центрами в рамках их компетенции (в т.ч. — для повышения квалификации работников образования, работы с одаренными детьми и т.д.).
8	Инвентаризация состояния образовательных организаций и формирование отчетности по форме ОО2	Подсистема предназначена для первичного учета материально — технического оснащения и финансового состояния общеобразовательных организаций.
9	Региональный сегмент межведомственной информационной системы «Контингент»	Назначением подсистемы является агрегация и обмен данными об ОО, контингенте обучающихся, педагогов, а также их достижениях и образовательных траекториях.

С использованием платформы Integrics x1 реализовано более 50 крупных проектов, в основном — в сфере образования.

Сергей Тулинов, Константин Кряженков, Игорь Дешко, Дмитрий Двоеглазов, Анатолий Тихонов
Москва, МИРЭА

Проект: Интернет — лаборатория ТЕРМИЛАБ `http://termilab.ru/`

Пример реализации сервиса Laboratory-as-a-Service в вузе с использованием СПО

Аннотация

Рассмотрены возможности свободного программного обеспечения для реализации массового онлайн обучения и возможности интеграции нового функционала при помощи API для обеспечения нового функционала: интеграция с реальным инфокоммуникационным оборудованием. Приведены особенности платформы, результат ее апробаций и предложения по совместному использованию научным коллективам и учебным организациям.

Ранее, в материалах [1] рассматривалась аппаратная составляющая платформы ТЕРМИЛАБ, как прототип реализации облачной лаборатории, как услуги (Laboratory-as-a-Service, LaaS), для изучения широкого круга дисциплин по информационно-коммуникационным технологиям (ИКТ).

По программной составляющей можно отметить, что интернет — лаборатория ТЕРМИЛАБ основана на свободных компонентах, таких как: Open edX [2], BigBlueButton [3] и Opencast Matterhorn [4].

Платформа Open edX позволяет организовать одновременный доступ большого числа участников к образовательному контенту и дополнительным сервисам. В базовой инсталляции имеются следующие возможности для реализации массового открытого онлайн обучения: управление датами начала и завершения онлайн классов, гибкие настройки по зачислению слушателей, публикация текста и видео, тестирование, дневник класса, email рассылки, форум и другие.

Сервис проведения вебинаров BigBlueButton позволяет организовать конференцию между инструкторами/тьюторами и слушателями курса. Данный сервис был интегрирован в платформу Open edX и обеспечена возможность инструкторам платформы Open edX автоматического участия в сервисе вебинаров BigBlueButton в качестве модератора и возможности записи вебинаров для их дальнейшего просмотра.

Решение Opencast Matterhorn используется для стриминга записанных видео-лекций, в т.ч. при помощи встроенного многопоточного интерактивного видеоплеера с возможностью переключения между различными вариантами отображения видео. Подобное решение затрудняет нелегальное распространение авторского контента и предоставляет возможности для лучшего восприятия содержимого видеолекций. Сами видеопотоки каждого отображаемого экрана оптимизированы для качественного просмотра видео в мобильных сетях связи.

Дополнительно в интернет-лабораторию ТЕРМИЛАБ была интегрирована авторская разработка по полнофункциональному удаленному доступу к учебным стендам с реальным ИКТ оборудованием (коммутаторы, маршрутизаторы, системы хранения данных и т.д.) и виртуальным средам с открытым или проприетарным программным обеспечением. Взаимодействие осуществляется на всех стадиях жизненного цикла (заказ стенда, активация устройств, работа, деактивация, возвращение в исходное состояние) посредством веб-интерфейса.

При подключении к телекоммуникационному оборудованию обеспечивается широкий функционал по удаленному взаимодействию с любым устройством в лабораторных стендах: консольное подключение с требуемой скоростью, отправка специальных комбинаций клавиш на устройство, включение и отключение электропитания устройств и т.д.

Возможности работы с виртуальными машинами (ВМ) в лабораторных стендах на базе VMware и oVirt представлены: отправкой специальных комбинаций клавиш в ВМ, включением/выключением и перезагрузкой ВМ, восстановлением оригинальной копии ВМ.

Платформа ТЕРМИЛАБ апробирована в ряде крупных проектов онлайн обучения. Среди них проект «Сетевая академии Cisco для людей с ограниченными возможностями здоровья» [5], комплекс поддержки формирования профессиональных компетенций учащихся в рамках интеграции общего и дополнительного образования на базе инновационно-образовательного кластера в сфере информационных технологий МГТУ МИРЭА и ГБПОУ «Воробьевы горы» [6], проект обучения преподавателей по международной образовательной программе Сетевой академии Cisco для повышения качества образования и эффективного участия в чемпионатах WorldSkills и JuniorSkills [7].

Последний из перечисленных проектов имел целью подготовку сотрудников образовательных учреждений РФ к получению статуса инструктора Сетевой академии Cisco по программе курса CCNA-1 «Введение в сетевые технологии», приобретению необходимых компетенций для подготовки учащихся к соревнованиям WorldSkills/Junior Skills по направлению №39 «Сетевое и системное администрирование» и прохождению демонстрационного экзамена. Всего в этом проекте приняло участие более 3000 чел. практически из всех субъектов РФ. Вопросы результативности обучения в других проектах рассмотрены в материалах [8] и [9].

Возможности платформы ТЕРМИЛАБ позволяют рассматривать её как потенциальное средство для проведения обучения пользователей отечественного программного обеспечения и его популяризации, в т. ч. в интересах реализации задач госконтракта от «17» ноября 2017г. № 03.596.11.0025 «Выстраивание и консультационное сопровождение процесса организации работ по установке на рабочие станции отечественного офисного программного обеспечения из удостоверенного репозитария в зависимости от профиля пользователя, хранимого на переносном носителе».

Дополнительно коллектив авторов предлагает всем научным коллективам и учебным организациям совместное использование платформы для реализации учебных проектов и научных исследований, поскольку возможности интернет-лаборатории позволяют в формате единого окна задействовать удаленные ресурсы заинтересованных вузов и организаций.

Литература

[1] Дешко И. П., Кряженков К. Г. Лабораторный практикум как сервис в курсах по информационно-коммуникационным технологиям [Электронный ресурс] // Международный электронный научный журнал «Перспективы науки и образования», Воронеж: 2015. — №1(13). — С.70–74. — Режим доступа: https://pnojournal.files.wordpress.com/2014/12/pdf_150111.pdf

[2] Open Source MOOC Platform: [сайт]. URL: https://open.edx.org/

[3] BigBlueButton: [сайт]. URL: https://bigbluebutton.org/

[4] Opencast Community: [сайт]. URL: http://www.opencast.org/

[5] Сетевая академия Cisco для людей с ограниченными возможностями здоровья: [сайт]. URL: http://www.4disabled.ru/

[6] Комплекс поддержки формирования профессиональных компетенций учащихся в рамках интеграции общего и дополнительного образования на базе инновационно-образовательного кластера в сфере информационных технологий: [сайт]. URL: http://mooc.moscow/

[7] Проект бесплатного обучения преподавателей по международной образовательной программе Сетевой академии Cisco для повышения качества образования и эффективного участия в чемпионатах WorldSkills и JuniorSkills: [сайт]. URL:http://www.1000ccna.ru/

[8] Двоеглазов Д. В., Дешко И. П., Кряженков К. Г. Массовый открытый онлайн курс по требованию: опыт реализации и результаты // Интернет-журнал «Мир науки» 2016, Том 4, номер 2 http://mir-nauki.com/PDF/52PDMN216.pdf (доступ свободный). Загл. с экрана. Яз. рус., англ.

[9] Дешко И. П., Кряженков К. Г., Тулинов С. В. Оценка результативности обучения в открытом онлайн курсе по сетевым технологиям // Муниципальное образование: инновации и эксперимент. 2016. №3. с.10–14

Евгений Синельников
Саратов, ООО «Базальт СПО»

Проект: Практика для дисциплин Операционные системы и Технологии программирования http://github.com/dmit-ssu

Опыт использования открытых технологий в образовательном процессе на факультете компьютерных наук

Аннотация

Активное включение популярных передовых информационных технологий в образовательный процесс, зачастую, приводит к монопольному доминированию наиболее дорогостоящих проприетарных программных продуктов в сфере образования. Преодолевать, устоявшийся, при этом, набор методик и практик на основе соответствующих программных решений – задача, как правило, сложная и требующая усилий. Причём не столько даже волевых, от руководства образовательных учреждений, сколько личных, от преподавателей и студентов. В данном докладе представлен десятилетний опыт перехода к применению свободного программного обеспечения в курсах факультета компьютерных наук и информационных технологий в саратовском государственном университете им. Н.Г. Чернышевского.

Существует спорное, но логичное утверждение о том, что «Computer Science – это не наука, а набор инженерных практик». Для дисциплин, которые непосредственно связаны с информационными технологиями и архитектурными особенностями электронно-вычислительных устройств, такое утверждение выглядит вполне правомерным. Авторский опыт преподавания практически ориентированных дисциплин Операционные системы [2] и Технологии программирования [3], во многом, оправдал это утверждение. Способствовали этому и так называемый компетентностный подход в образовании, и введение болонской системы с переходом к четырёхлетнему бакалавриату.

Тем не менее, задача — отделить опыт профессионального технического образования от фундаментального вузовского знания, особенно практически ориентированных дисциплинах, с каждым годом приобретает всё большую актуальность. Как и за счёт чего будет преодолеваться разрыв между приобретением навыков квалифицированного пользователя и абстрактным опытом разработчика? С кого момента этот абстрактный опыт может быть квалифицирован как научное

знание? Повседневный опыт преподавания в современном вузе скорее отдаляет от ответов на эти вопросы, чем приближает к ним [1].

Ещё один важный момент — это мотивация преподавателя. Понятно, что молодой технический специалист, заинтересованный своей профессиональной сферой, может проявить готовность поделиться своими навыками. Но какими? Как правило, это навыки привязанные к инструментам, к программным и техническим средствам, а не к актуальным задачам и фундаментальным проблемам.

Таким образом, работа со студентами вузов над задачами в сфере информационных технологий требует, как минимум, актуализации этих задач. При этом использование свободного программного обеспечения и открытых информационных технологий в качестве основного инструментария, действительно, даёт огромное преимущество. Этому многое способствует:

- во-первых, непосредственная возможность, и даже необходимость, взаимодействия с действующими разработчиками свободных программ;
- во-вторых, возможность знакомства с рабочим исходным кодом ключевых программных механизмов и, тем самым, с реальной, а не модельной архитектурой программных систем;
- в-третьих, необходимость практического использования, даже для учебных задач, актуальных средств разработки.

Первая возможность наиболее ярко представлена в проекте Google Summer of Code, в рамках которого ежегодно сотни разработчиков СПО привлекают студентов к своим задачам. Участие в собственных, вузовских разработках эта возможность также выводит студентов на реальных разработчиков. Знакомство с рабочим исходным кодом даёт огромный набор возможностей, с точки зрения демонстрации архитектурных решений. При этом не выхолощенные учебные примеры, в которых применяются промышленные средства сборки и разработки, позволяют шаг за шагом переходить от знакомства с примерами к рабочему исходному коду.

По прошествии десяти лет постепенного обобщения данного подхода, можно с уверенностью сказать, что задачи профессионального технического образования такой подход вполне оправдывает. Что вполне укладывается содержание программ бакалавриата. При этом возможность дополнительного, самостоятельно участия в учебном процессе у студентов также остаётся. Прикоснувшись к актуальным

задачам, возможность заняться фундаментальными проблемами, при таком подходе, сохраняется.

Конечно, в этом подходе имеются и свои издержки. Даже не на уровне сложности, а на уровне целостности учебных подходов в уже сложившихся, связанных дисциплинах (Языки высокого уровня и Технологии программирования, например) и методической проработанности примеров и задач. Но и это отсутствие целостности тоже является проблемой. Как организационной, так и методической. Практика же внедрения СПО и отрытых технологий в образовательный процесс даёт повод заняться и этой проблемой.

Литература

[1] *Сергей Голубев*, СПО в высшей школе: проблемы и перспективы, `https://www.itweek.ru/foss/article/detail.php?ID=192050`

[2] Дисциплина «Операционные системы», 2016, `https://www.sgu.ru/structure/computersciences/courses/bachelor-vychislitelnye-mashiny-kompleksy-sistemy-i-seti/operacionnye-sistemy`

[3] Дисциплина «Технологии программирования», 2016, `https://www.sgu.ru/structure/computersciences/courses/bachelor-vychislitelnye-mashiny-kompleksy-sistemy-i-seti/tehnologii-programmirovaniya`

Евгений Кондратьев
Москва, Московский технологический университет

Информационная система на основе LibreOffice Calc для численного решения обыкновенных дифференциальных уравнений

Аннотация

На основе LibreOffice Calc реализована информационная система для численного решения обыкновенных дифференциальных уравнений.

В электронной таблице Calc, входящей в состав свободного программного обеспечения (далее СПО) LibreOffice The Document Foundation [1], обычному пользователю персонального компьютера стало

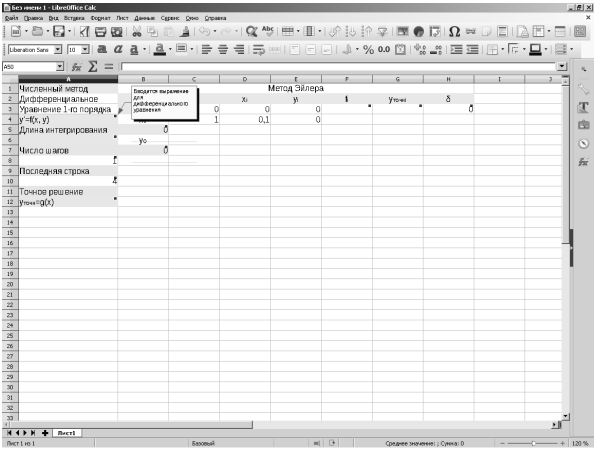


Рис. 1: Шаблон для численного решения методом Эйлера обыкновенных дифференциальных уравнений первого порядка

доступным решение различных инженерных задач с использованием численных методов [2]. Переход на СПО в федеральных бюджетных учреждениях был определен планом, утвержденным распоряжением Правительства РФ от 17 декабря 2010 г. № 2299-р [3].

На основе LibreOffice Calc реализована информационная система для решения обыкновенных дифференциальных уравнений первого и второго порядков разными численными методами. Система использует разработанные шаблоны, содержащие комментарии для заполнения ячеек на начальном этапе расчета.

На рис. 1 приведен шаблон для численного решения методом Эйлера обыкновенных дифференциальных уравнений первого порядка, а на рис. 2 показано построенное с помощью этого шаблона решение следующей задачи Коши:

$$y' = 2xy, \tag{26.1}$$

$$y(0) = 1 \tag{26.2}$$

на отрезке $[0, 1]$ с шагом $h = 0.1$.

На рис. 3 приведен шаблон для численного решения четырехшаговым методом Адамса–Башфорта обыкновенных дифференциальных

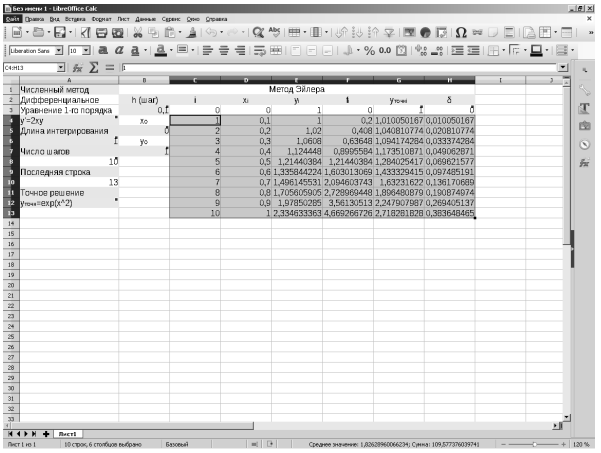


Рис. 2: Вычисления по методу Эйлера

уравнений второго порядка, а на рис. 4 показано построенное с помощью этого шаблона решение следующей задачи Коши:

$$y'' = x + y, \tag{26.3}$$

$$y(0) = 2, y'(0) = -1 \tag{26.4}$$

на отрезке $[0, 2]$ с шагом $h = 0.1$.

Литература

[1] `http://ru.libreoffice.org/`

[2] Кондратьев, Евгений. Элементы численных методов в LibreOffice Calc // Одиннадцатая конференция «Свободное программное обеспечение в высшей школе»: Материалы конференции/ Переславль, 30–31 января 2016 г. — М.: Альт Линукс, 2016., С. 83–85.

[3] Об утверждении плана перехода федеральных органов исполнительной власти и федеральных бюджетных учреждений на использование свободного программного обеспечения на 2011–2015 г.г.: распоряжение Правительства РФ от 17 декабря 2010 г. № 2299-р., `http://base.garant.ru/6746035/`

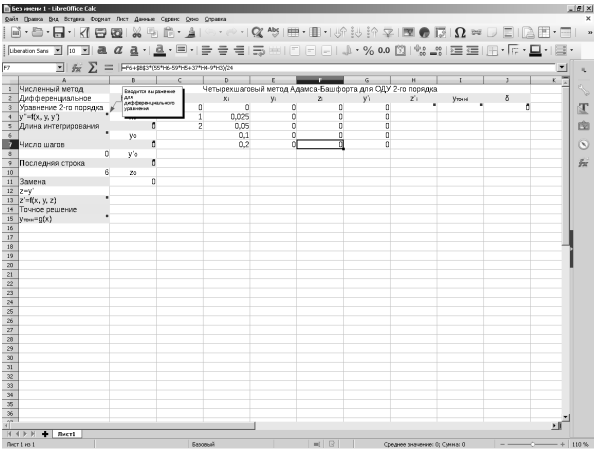


Рис. 3: Шаблон для численного решения четырехшаговым методом Адамса–Башфорта обыкновенных дифференциальных уравнений второго порядка

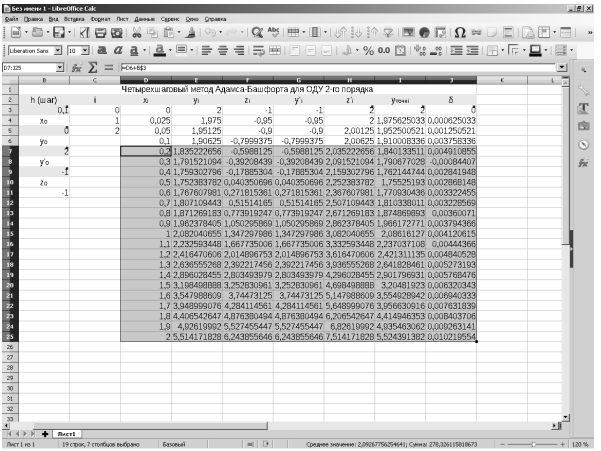


Рис. 4: Вычисления по четырехшаговому методу Адамса–Башфорта

Константин Кряженков, Игорь Дешко, Дмитрий Двоеглазов, Анатолий Тихонов, Сергей Тулинов
Москва, МИРЭА

Разработка репозитория отечественного офисного программного обеспечения

Аннотация

Рассмотрены вопросы основных рабочих процессов и архитектуры разрабатываемого репозитория отечественного офисного программного обеспечения для подведомственных Минобрнауки России организаций.

В соответствии с Программой «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2014-2020 годы» в МИРЭА разрабатывается репозиторий отечественного офисного программного обеспечения (ПО) для подведомственных Минобрнауки России организаций науки и образования.

Репозиторий должен содержать отечественные свободно распространяемые или распространяемые по коммерческим лицензиям удостоверенные пакеты офисного ПО и ОС, и обеспечивать для пользователей следующие основные процессы:

1. Предоставление пакета ПО удостоверенного качества на пользовательские рабочие станции (ПРС);
2. Развертывание пакета ПО удостоверенного качества на ПРС;
3. Обеспечение поддержки.

Перед размещением пакета ПО сервисные компоненты Репозитория проводят его валидацию. Проверка осуществляется по спектру базовых и настраиваемых критериев, позволяющих оценить достоверность пакета ПО, его целостность, безопасность и соответствие иерархии файловой системы. В случае успешного прохождения тестов пакет ПО удостоверяется цифровой подписью Репозитория и располагается в выбранном разделе. Каждый размещаемый пакет в разделе Репозитория классифицируется внутренним классификатором, разработанным на основе ГОСТ Р ИСО/МЭК ТО 12182-2002 и Приказа

Минкомсвязи России от 01.04.2016 №134. Используя данные классификаторы, пользователи Репозитория определяют необходимый им набор пакетов ПО.

Развертывание пакета ПО проводится посредством специализированного кроссплатформенного клиента на ПРС. Автоматизация этого процесса реализована с применением технологии виртуализации на уровне ОС. Это позволяет создавать локальные и удаленные изолированные контейнеры, содержащие экземпляры ПО, явно неустанавливаемые в ОС ПРС. Контейнеры также применяются для проведения входного контроля, в частности для поиска вредоносного содержимого в пакетах ПО, размещаемых в Репозитории.

Для удобства пользователей взаимодействие с Репозиторием будет осуществляться через официальный сайт Минобрнауки России с использованием безопасных протоколов. Графический интерфейс соответствует концепции DWIM, предоставляет доступ к экранным объектам и возможности манипулирования ими с ПРС и мобильных устройств.

В целом, разрабатываемый Репозиторий отвечает следующим основным функциональным требованиям, реализованным в сценариях выполнения рабочих процессов:

1. Проведение входного контроля ПО перед размещением в Репозиторий.
2. Хранение ПО в Репозитории;
3. Управление правами доступа к ПО Репозитория;
4. Принятие пользователями лицензионных соглашений с правообладателями ПО, включенного в Репозиторий;
5. Подключение с ПРС к Репозиторию;
6. Разворачивание виртуальной среды для автоматизированной установки на ПРС выбранного ПО из Репозитория;
7. Административное управление серверной частью Репозитория.
8. Масштабирование и высокая доступность Репозитория.
9. Мониторинг состояния серверной части Репозитория.
10. Отслеживание ошибок в процессе функционирования Репозитория и в пакетах ПО, размещенных в нем.

Реализация этих требований нашла свое отражение в архитектуре Репозитория. Она сформирована по модульному принципу, поддерживает наращивание функционала за счет применения унифицированной интеграционной шины и обеспечивает децентрализацию и горизонтальное масштабирование для адаптации к условиям роста нагрузки. При этом Репозиторий может быть развернут как на базе отдельно стоящих серверов, так и в программно-определяемом ЦОДе.

Все программные решения для построения Репозитория используют только свободно распространяемое открытое ПО.

Сергей Мартишин, Марина Храпченко
Москва, Институт системного программирования РАН

Принципы применения модели акторов для распределенных вычислений на кластерах с использованием кэширования.

Аннотация

Проанализированы принципы распределенной обработки больших данных на основе модели акторов[1] . Рассмотрены возможности фреймворка Akka для работы на кластерах и построения системы кэширования для увеличения скорости обработки данных.

Обучение студентов направления подготовки «Информационные системы и технологии» требует постоянного изучения новых технологий, что актуально из-за быстрого развития данного направления. В соответствии с требованиями ФГОС ВПО, выпускник по направлению подготовки «Информационные системы и технологии» должен, в частности, обладать такими профессиональными и специальными компетенциями, как: способностью к проектированию базовых и прикладных информационных технологий (ПК-11), способностью разрабатывать средства реализации информационных технологий (ПК-12), готовностью адаптировать приложения к изменяющимся условиям функционирования (ПК-32) [1].

Наиболее актуальным направлением исследований по-прежнему остается обработка больших данных (Big data), необходимая во многих сферах (страхование и банковское дело, медицина, демография и

[1]Работа выполнена при финансовой поддержке РФФИ, проект № 17-07-01006 А

пр.), которая требует значительных вычислительных ресурсов и специальных алгоритмов.

Часто сама логика обработки такого рода данных подразумевает распределенный характер вычислений, осуществляемый на различных кластерах. В последнее время появилось свободно распространяемое программное обеспечение (СПО), предоставляющее возможности параллельной обработки информации за счет использования асинхронных вычислений, отсутствия блокировки памяти, обмена кэшированными данными. При его использовании многие алгоритмы работы с большими данными, такие как операция MapReduce, анализ текста, обработка статистических данных и др. могут быть реализованы значительно эффективнее, чем в случае последовательной обработки информации или параллельной обработке, основанной на потоках.

Одной из хорошо известных идей, нашедшей свое воплощение в СПО Akka, является модель акторов, возникшая первоначально как модель параллельных вычислений. Ее теоретические основы были изложены в статье Carl Hewitt; Peter Bishop & Richard Steiger «A Universal Modular Actor Formalism for Artificial Intelligence» в 1973 г..

В рассматриваемой модели актор — это универсальный примитив для выполнения параллельных вычислений. Его можно рассматривать как рабочий процесс или поток, который в ответ на получаемые сообщения может принимать локальные решения, создавать новые акторы, посылать свои сообщения (некоторые данные, используемые для связи с процессами).

Акторы осуществляют коммуникацию с внешним миром посредством передачи сообщений, которая осуществляется асинхронно. Передача сообщений может, в том числе, изменять состояние актора. Актор обрабатывает сообщение или отвечает на сообщение по мере своей готовности. Важной особенностью сообщений является то, что сообщения не имеют возвращаемого значения. Отправляя сообщение, один актор делегирует работу другому актору, при этом работа первого актора не блокируется, поскольку он не ожидает ответа. Впоследствии принимающий актор передает результаты в ответном сообщении.

Для буферизации сообщений, адресованных каждому экземпляру актора, используется так называемый почтовый ящик — место в памяти, где сообщения хранятся до тех пор, пока актор не сможет их обработать. Также имеется почтовый адрес (ссылка на актор) —

информация о том, куда необходимо направлять сообщения для конкретного актора.

Фремворк Akka — это СПО с открытым исходным кодом (написан на Scala) для разработки масштабируемых, устойчивых систем, позволяющий писать программы, способные выполняться на JVM (Java Virtual Machine) в единственном процессе, в нескольких процессах на одной машине или на нескольких машинах [2]. Языковые привязки существуют как для Scala, так и для Java.

Akka позволяет развертывать системы акторов в распределенных приложениях. Для этого производится настройка работы системы через некоторый сетевой порт (из числа свободных на данной машине) и установления связей между системами акторов на различных машинах при помощи обмена сообщениями. Таким образом применение Akka ориентировано на кластеры. А с точки зрения пользователя обеспечивается прозрачность местоположения актора, поскольку пользователь может выполнять вычисления, не имея никакого представления о конфигурации сети [3].

Также использование акторов Akka для кластеров позволяет организовать эффективную работу с памятью. Например, при работе в back-end системах (основное ПО) данные постоянно обновляются. Для работы front-end системы (интерфейсной части) должен происходить обмен данными с back-end системой. Для быстроты взаимодействия данные необходимо сохранять в кэше, который обновляется с некоторой периодичностью. При помощи акторов легко создать систему кэширования, поскольку в этом случае можно обеспечить равномерную нагрузку и спланировать задачи для периодического обновления кэша по некоторому временному параметру.

Однако не смотря на все свои достоинства, модель акторов имеет ряд недостатков, которые могут явится критическими при работе крупных систем. Например, узким местом является то, что акторы имеют возможность принимать сообщения по очереди и их нельзя настроить на ожидание определенной последовательности сообщений. Логичным продолжением явилось создание и развитие фреймворка Reactors [4], который также предназначен для создания систем распределенных вычислений, основан на модели акторов и может быть использован для языков Scala и Java [3].

Изучение студентами способов организации распределенной обработки данных на основе модели акторов является актуальным и позволит выпускникам применить свои знания на практике.

Литература

[1] Федеральный государственный образовательный стандарт высшего профессионального образования по направлению подготовки 230400 Информационные системы и технологии (квалификация (степень) «бакалавр») / Утвержден приказом Минобрнауки России от 12.03.2015 N 219 (Зарегистрировано в Минюсте России 30.03.2015 N 36623). .

[2] Akka [Электронный ресурс]. Режим доступа: URL:– в https://akka.io Яз. англ. Дата обращения: 22.12.2017. 1.

[3] Прокопец А., Конкурентное программирование на Scala, Изд-во ДМК, М., 2018г, с.342 /

[4] Reactors.io. Foundational framework for distributed computing [Электронный ресурс]. Режим доступа: URL:– в http://reactors.io Яз. англ. Дата обращения: 22.12.2017.

Крюков Александр
г. Рязань, Рязанское высшее воздушно-десантное командное училище им. В.Ф.Маргелова

Свободное и проприентарное ПО в учебном процессе кафедры

Аннотация

Разнообразие учебных задач и «зоопарк» компьютеров накладывают ограничения на выбор ПО

На кафедре РРТСиПС РВВДКУ обучаются будущие специалисты с высшим образованием по ФГОС ВПО 11.04.04 [1], техники по ФГОС СПО 11.02.09 [2], 11.02.10 [3], 11.02.11 [4].

Большое разнообразие учебных задач, для выполнения которых предполагается использовать компьютерное оборудование, накладывает ограничения на выбор используемого программного обеспечения.

Для тестирования обучающихся по 5 предметам используется унаследованная программа Assistent [5]. При проведении лабораторных работ по РРВиАФУ применяется MMANA [6]. Для моделирования переходных процессов на лабораторных работах по ОТЦ, спектральных характеристик сигналов на ТЭС используется MicroCap [7]. Эти программы установлены на wine. В рамках военно-научной работы

курсантов и дипломного проектирования СПО создаются в формате *.ppt как в LibreOffice Drav, так и в Microsoft Office PowerPoint, установленный на wine, анимированные плоские изображения процессов, происходящих внутри элементов инфокоммуникационной техники.

Для работы с документами в форматах MS Office 2003–2007 используется LibreOffice 4.2, при этом для сохранения форматирования и шрифтов в операционную систему установлен пакет ttf-mscorefonts-installer. Рисование, редактирование изображений производится в kolourpaint4, русификация которого выполнена установкой в GNOME нескольких пакетов от KDE. GIMP на рабочих станциях также установлен, но большинство обучающихся его возможности не используют.

Несмотря на различия элементов управления MS Office и LibreOffice, Paint и KolourPaint, курсанты на занятиях «методом тыка», с помощью более подготовленных товарищей (файлы учебников есть на каждой машине, успехом не пользуются, а преподаватель, дав задание, организует «зачёт по последнему» и устраняет только сбои; в полной мере используются принципы андрагогики [8]) обучаются пользоваться ими и в дальнейшем не испытывают трудностей.

Хотя различия в органах управления Microsoft Office и китайского WPS Office [9] и меньше, чем между Microsoft Office и LibreOffice, а возможностей у WPS Office больше, чем у LibreOffice, WPS Office «не прижился» из-за больших требований к «железу», большого времени открытия файлов, а также сбоев при открытии повреждённых или не полных файлов (что частенько наблюдается в поражаемой вирусами среде курсантских носителей).

Ранее на компьютерах была установлена Debian 7.7. При обновлении операционной системы обновилась wine, в результате ряд программ перестал запускаться. Был выполнен переход на Ubuntu 14.4 LTX, а на более старых компьютерах - Runtu XFCE 14.4. С надеждой на переход на отечественные процессоры «Эльбрус» автором проводятся эксперименты по их замене на АЛЬТ Рабочая станция 8.2. Эксперименты осложняются тем, что в репозитории Sisyphus не содержится ряд инструментов, типовых для Ubuntu, и меньшим числом форумов, посвящённых АЛЬТ.

Литература

[1] Федеральный государственный образовательный стандарт высшего образования по специальности 11.05.04 Инфокоммуникационные технологии и системы специальной связи (уровень специалитета). Утверждён приказом Министерства образования и науки Российской Федерации от 11 августа 2016 г № 1035.

[2] Федеральный государственный образовательный стандарт среднего профессионального образования по специальности 11.02.09 Многоканальные телекоммуникационные системы. Утверждён приказом Министерства образования и науки Российской Федерации от 28 июля 2014 г. № 811.

[3] Федеральный государственный образовательный стандарт среднего профессионального образования по специальности 11.02.10 Радиосвязь, радиовещание и телевидение. Утверждён приказом Министерства образования и науки Российской Федерации от 28 июля 2014 г. № 812.

[4] Федеральный государственный образовательный стандарт среднего профессионального образования по специальности 11.02.11 Сети связи и системы коммутации. Утверждён приказом Министерства образования и науки Российской Федерации от 28 июля 2014 г. № 813.

[5] Система автоматизированного контроля знаний Assistent. Свидетельство о государственной регистрации № 2008610441. Электронный ресурс `http://asksystem.narod.ru`.

[6] Программа моделирования антенн MMANA. Электронный ресурс `http://gal-ana.de/basicmm/ru/` `http://www.radio.ru/mmana/`

[7] Программа моделирования Micro-Cap Электронный ресурс `http://www.spectrum-soft.com/demoform.shtm`

[8] Андрагогика: теория и практика образования взрослых: Учеб. пособие для системы доп. проф. образования; учеб. пособие для студентов вузов /М.Т.Громкова. — М.: ЮНИТИ-ДАНА, 2012. — 495 с. (Серия «Высшее профессиональное образование: Педагогика») ISBN 5-238-00823-6

[9] Воспользуйтесь лучшими в мире возможностями офиса для Linux. Электронный ресурс `https://www.wps.com/linux`