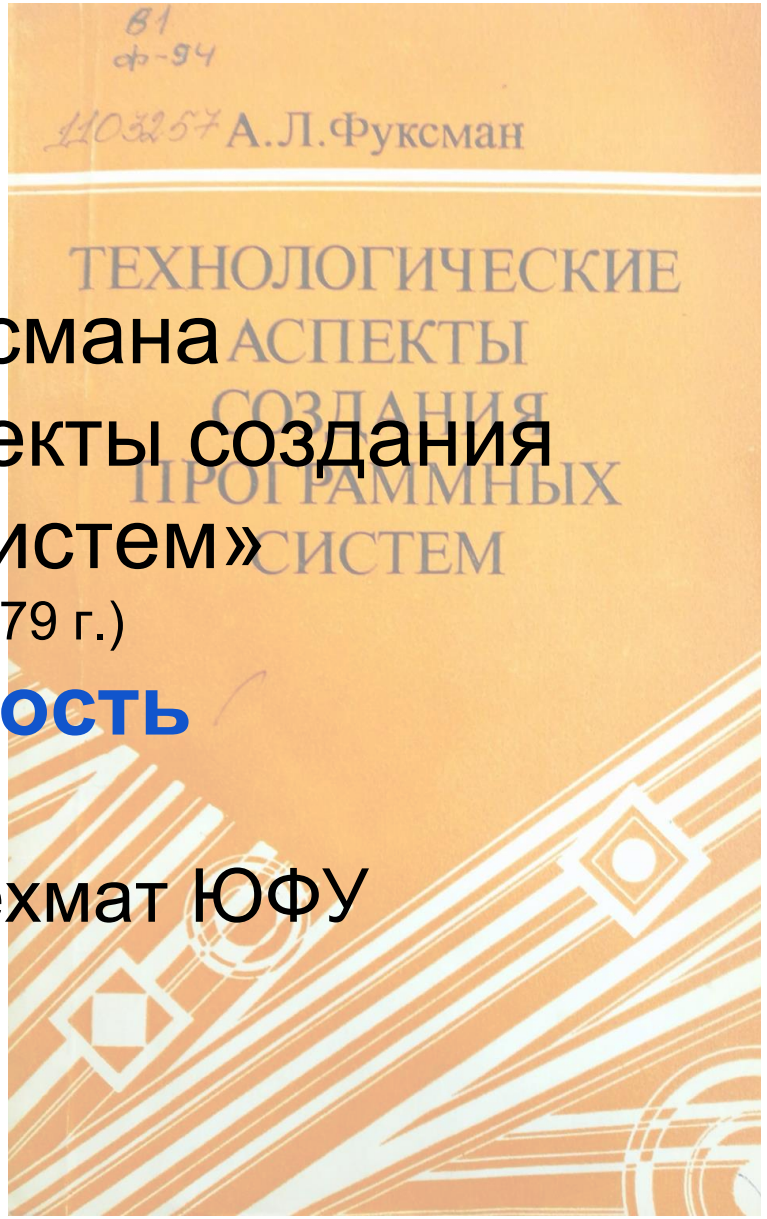


Книга А.Л.Фуксмана «Технологические аспекты создания программных систем»

(М., Статистика, 1979 г.)

и современность

Михалкович С.С., мехмат ЮФУ

Выборочный список работ А.Л.Фуксмана (собрано Ю.С.Налбандян, 2017)

1. О некоторых грамматиках для описания контекстно-свободных языков / Труды 1-й Всесоюзной конференции по программированию. Секция А. Вопросы теории программирования. Киев, **1968**, с. 135-143.
2. Алгоритмы синтаксического анализа для некоторых классических языков / Применение методов вычислительной математики и вычислительной техники для решения научно-исследовательских и народнохозяйственных задач, в.4 Воронеж, 1969. С.61-64.
3. О некоторых свойствах формальных грамматик /Труды Всесоюзной конференции по программированию. Заседание К. **Новосибирск, 1970**, 3-6 февраля. С.21-31.
4. Система проектирования трансляторов СПТ-РГУ /Труды симпозиума «Теория языков и методы построения системы программирования». Киев-Алушта, 1972.
5. Разработка одной многоязыковой модульной системы пакетной обработки задач /Системное программирование, Ч. III. **Новосибирск, 1973** (совместно с С.М.Абрамовичем и др.)
6. **Основы разработки трансляторов** (совместно с С.П.Крицким, А. А.Дагалдьяном, Х.Д.Дженибалаевым). – Ростов-на-Дону: ИРУ, 1974. 281 с.

Выборочный список работ А.Л.Фуксмана (собрано Ю.С.Налбандян, 2017)

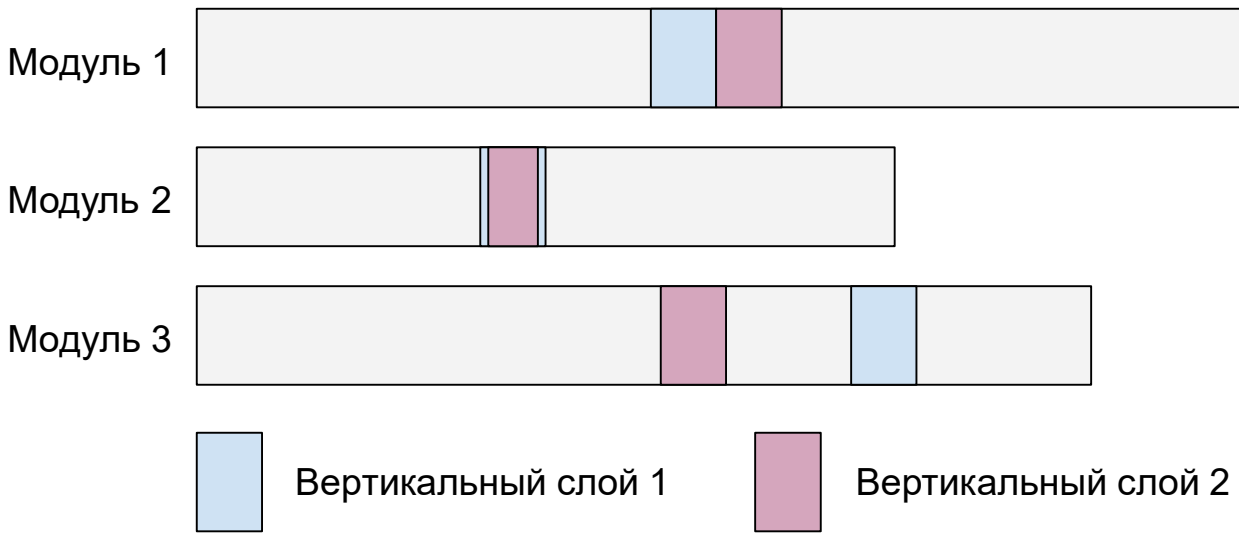
7. **Расслоенное программирование** / Системное и теоретическое программирование, Новосибирск, 1974.
8. **Макрогенерация** с управляющим языком высокого уровня /Труды всесоюзного семинара по вопросам макрогенерации. Тбилиси, 1975.
9. Некоторые принципы построения трансляторов /Труды всесоюзного симпозиума по методам реализации новых алгоритмических языков. **Новосибирск, 1975**
10. **Слаборазделенные грамматики** //Журнал вычислительной математики и математической физики, 1976, т.16, № 5, С.1293-1304.
11. Диагностические и другие упрощенные формы магазинных автоматов // **Программирование 1976**, № 5 (журнал «Программирование» основан в **1975 г.**)
12. **Технологические аспекты создания программных систем - М. : Статистика, 1979. – 183 с.**

Глава 3.
Основы вертикального слоения программ

Реализующие и расширяющие функции

Реализующие функции образуют основу программной системы.

Расширяющие функции: их изъятие не лишает систему работоспособности. Код, реализующий расширяющую функцию, назовем **вертикальным слоем**. Вертикальный слой состоит из фрагментов кода, входящих в модули разных горизонтальных уровней. Представление программы в виде основы и последовательности расширяющих функций - **процесс вертикального слоения** программы



Сосредоточенное описание рассредоточенных действий

Вертикальный слой является **рассредоточенным действием**. Его текст – это набор вставок в основу, возможно, с добавлением некоторых других слоев.

Сосредоточенное описание вертикального слоя:

- ссылка на основу
- какие слои должны быть подключены перед добавлением этого слоя
- список программных фрагментов с указанием координат мест вставки
- неформальное описание расширяющей функции, реализуемой слоем
- понятия и термины, связанные с расширяющей функцией

Для получения выполнимой программы необходимо осуществить процесс **интеграции**. Текст интегрированной программы должен быть **послойно размеченным**: пометки позволяют отнести любую часть текста к определенному слою.

Современность - АОП (Википедия)

Аспе́ктно-ориенти́рованное программи́рование (АОП) — парадигма программирования, основанная на идее разделения функциональности для улучшения разбиения программы на модули. Методология АОП была предложена группой инженеров исследовательского центра Xerox PARC под руководством Грегора Кичалеса (Gregor Kiczales). Ими же было разработано аспектно-ориентированное расширение для языка Java, получившее название AspectJ (**2001 год**). Первая статья по АОП: Gregor Kiczales: Aspect-Oriented Programming. ACM Comput. Surv. 28(4es): 154 (**1996**)

Основные понятия АОП:

Аспект (англ. *aspect*) — модуль или класс, реализующий сквозную функциональность. Аспект изменяет поведение остального кода, применяя *совет* в *точках соединения*, определённых *срезом*.

Совет (англ. *advice*) — средство оформления кода, которое должно быть вызвано из точки соединения. Совет может быть выполнен до, после или вместо точки соединения.

Точка соединения (англ. *join point*) — точка в выполняемой программе, где следует применить совет (обычно это вызовы методов и обращения к полям объекта).

Срез (англ. *pointcut*) — набор точек соединения. Срез определяет, подходит ли данная точка соединения к совету. Для определения срезов часто используется синтаксис основного языка.

Внедрение (англ. *introduction*, введение) — изменение структуры класса и/или изменение иерархии наследования для добавления функциональности аспекта в инородный код.

Взаимодействие слоев

Определение. Два вертикальных слоя **независимы** если:

включение одного слоя не требует включения другого

их фрагменты не пересекаются

два находящихся рядом фрагмента разных слоев могут быть расположены в произвольном порядке и выполняться параллельно

Виды зависимостей слоев

1. Зависимость включения слоя B от слоя A
2. Зависимость расположения слоев A и B
3. Зависимость по составу слоев A и B
    a. Пересечение слоев A и B
    b. Совмещение слоев A и B
    c. Зависимость модификации слоя A от слоя B:

Зависимости слоев

1. **Зависимость включения слоя B от слоя A**: без слоя A слой B лишен смысла. Пример: в системе учета сотрудников без слоя “атрибуты о сотруднике” невозможен слой “печать сотрудника”
2. **Зависимость расположения слоев A и B**: некоторые фрагменты A и B являются соседними в интегрированной программе и могут быть расположены только в определенном порядке.
3. **Зависимость по составу слоев A и B**
   a. **Пересечение слоев** A и B. Фрагмент кода c входит в текст слоя A и в текст слоя B и отсутствует, если в программу не входит хотя бы один из слоев A или B
   b. **Совмещение слоев** A и B. Фрагмент кода c одновременно входит в текст слоя A и в текст слоя B и присутствует в программе, если в ней есть хотя бы один слой A или B. Пример: сервисная функция, необходимая для реализации слоев “учет текстовых атрибутов сотрудников” и “учет числовых атрибутов сотрудников”
   c. **Зависимость модификации** слоя A от слоя B. Внесение слоя B приводит к удалению фрагмента c из слоя A. Фактически, может происходить **замена**: слой B заменяет в слое A фрагмент c1 на c2.

Технология вертикального слоения - алгоритм

Главная процедура: создать программу вертикальными слоями;
/1 продумать архитектуру системы;
/2 выделить основу системы;
/3 *СОВМЕСТНО;*
=1 создать и отладить основу;
= 2/1 создать перечень вертикальных слоев;
/2 составить сетевую зависимость С создания слоев с учетом зависимости соответствующих функций;
/3 выделить первую очередь системы;
/3 *КСОВМ;*
/4 *СОВМЕСТНО:*
=1 вести проверочную эксплуатацию текущей версии системы;
=2 *СЕТЬ* в соответствии с последовательностью С для слоев первой очереди системы:
/1 создать слой, добавить его к системе и отладить в ней;
КСЕТЬ;
/4 *КСОВМ;*
/5 *СОВМЕСТНО:*
=1 вести рабочую эксплуатацию системы;
=2 вести модернизацию системы наращиванием слоев второй очереди;
Процедура 1. Создать слой, добавить его к системе и отладить в ней;
/1 записать цели и понятия слоя;
/2 записать идеи и алгоритмы реализации слоя;
Аномалии: 1. Для реализации слоя необходимо переделать фоновую программу;
/3 записать документацию по слою в виде вставок в фоновую документацию по системе (эксплуатационную и техническую);
/4 получить распечатку нужных частей фоновой программы;
/5 записать программу слоя в виде вставок в описание информации и в операторы текущей фоновой программы;
Аномалии: 1. Среди технических оказались процедуры, близкие процедурам другого слоя, создаваемого параллельно (избыточность);
/6 разработать методику проверки слоя;
/7 вставить в программу слоя подтверждающую печать;
/8 записать текст слоя в виде входной информации системы;
/9 ввести в систему документацию, методику проверки, программу и тест слоя;
/10 выполнить синтаксическую отладку слоя;
/11 проверить интегрированную программу с новым слоем на тестах предыдущих слоев;
/12 отладить интегрированную программу на тесте нового слоя;
/13 фиксировать отлаженную интегрированную программу в качестве фоновой;
/14 включить слой в эксплуатируемую систему;
/15 закончить процедуру.

87

Алгоритм содержит исчерпывающие указания, как проектировать систему, содержащую основу и вертикальные слои

Достоинства технологии вертикального слоения

1. **Создание** на ранней стадии развития программы некоторого полностью работоспособного варианта. Увеличение полноты этого варианта с каждым добавлением нового вертикального слоя в систему.
2. Т.к. создание слоя включает создание документации, система в каждый момент полностью **документирована**.
3. **Освоение системы** возможно в той последовательности, в которой она создавалась (спорный тезис). Процесс познания системы, написанной методом вертикального слоения, отличается от такового при изучении системы *сверху вниз* (до окончания изучения не ясны подробности функционирования) и *снизу вверх* (до окончания изучения не ясны стыковки частей в целое).
4. Облегчение **модернизации** системы. Любая модернизация - процесс добавления слоев к основе.
5. **Отладка системы** по слоям позволяет использовать реальные данные в полностью работающей системе.ьи

Глава 4. Техника вертикального слоения и инструментальный комплекс

Оформление программы слоя

Состав вертикального слоя

1. Документация по слою
2. Программа слоя
3. Тест слоя
4. Описание методики проверки слоя

Программа слоя - совокупность изменений (обычно вставок) в фоновую программу (основа + предыдущие слои)

Запись изменения - вид изменения (ВСТАВКА, УДАЛЕНИЕ, ЗАМЕНА), координаты и текст изменения. **Координата** - это имя процедуры [, номер строки [, номер столбца]].

Примеры:

```
ВСТАВИТЬ В проц 1 ЗА ВСТ 5.3 ЗА ВСТ 4.7 ПЕРЕД ВСТ 6.2
```

или

```
ИСПОЛЬЗ проц 1.5 ДО 7 ВКЛЮЧ
ЕСЛИ (1.5 И НЕ 2.3) или 7 ТО
        ВСТАВИТЬ ЗА (ВСТАВИТЬ ПЕРЕД) ...
```

Коррекция слоя - исправление ошибок

Исправления могут относиться к нескольким слоям сразу. **Инструментальный комплекс** осуществляет прогон тестов для всех исправляемых слоев

Исправление может состоять в восстановлении ошибочно удаленного фрагмента. Вставки, удаления и замены в старых фрагментах проводятся так же, как и при добавлении слоя. Возможно корректирование разметки слоя:

НЕ ИСПОЛЬЗ (фрагмент)

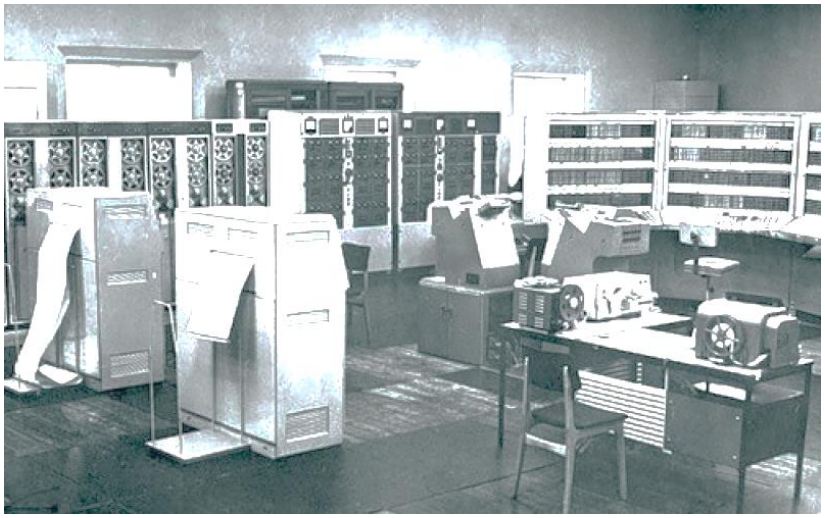
БЕЗУСЛ НОВ (фрагмент) - отмена условия вставки фрагмента

ЕСЛИ (усл) ТО НОВ (фрагмент) - добавление условия вставки фрагмента

Если изменений много, целесообразно изъять слой и затем добавить его в исправленном виде.

При внесении исправлений Инструментальный комплекс проверяет, является ли автор исправляемого участка его разработчиком, и шлет извещение разработчику если это не так.

Каждое исправление сопровождается указанием класса ошибки. Инструментальный комплекс **накапливает статистические данные об ошибках**.



Глава 5.
Развитие языка программирования

Критерии ценности языка программирования

Первичные критерии:

1. **Надежность**, соответствие программы назначению, отсутствие ошибок
2. Экономный расход объема **памяти**
3. Экономный расход **времени** процессора
4. Низкие затраты на создание программы
5. Низкая трудоемкость эксплуатации программы
6. Низкие затраты времени на создание программы

Вторичные критерии:

1. Познаваемость программы
2. Производительность программирования
3. Адаптируемость программы
4. Защищенность от ошибок
5. Связь с реализацией
6. Изучаемость языка

Вторичные критерии - подробно

Познаваемость программы

1. **Расшифрованность текста** программы
2. **Мнемоничность обозначений** языка
3. **Близость** языка **к естественному**
4. **Произносимость конструкций** языка
5. **Лаконичность** конструкций языка

Производительность программирования

1. **Одновременное создание** частей программы разными исполнителями
2. **Укрупненный характер** информационных **объектов** и операций

Адаптируемость программы

1. Адаптируемость программы к изменению задачи
2. Адаптируемость программы к изменению оборудования, на котором она выполняется
3. Машинная независимость языка

Вторичные критерии - подробно (2)

Защищенность от ошибок

1. Возможность обнаружить описки при компиляции
2. Возможность **статического контроля** несоответствия между планируемыми и фактическими свойствами информационных объектов

Связь с реализацией

1. Возможность **простой машинной реализации** языковых средств
2. Ясное **ощущение** программистом **стоимости** (по времени и по памяти) **языковых конструкций**
3. Низкая трудоемкость эффективной реализации языка

Изучаемость языка

1. **Независимость** (ортогональность) **конструкций языка**. Возможность менять язык в отношении одной конструкции, не затрагивая прочих
2. Логическая однородность языка. Аналогичное изображение идентичных функций.

Книга А.Л.Фуксмана «Технологические аспекты создания программных систем»

(М., Статистика, 1979 г.)

и современность

Михалкович С.С., мехмат ЮФУ