Пример архитектуры — Bash

Юрий Литвинов
yurii.litvinov@gmail.com

25.10.2019г

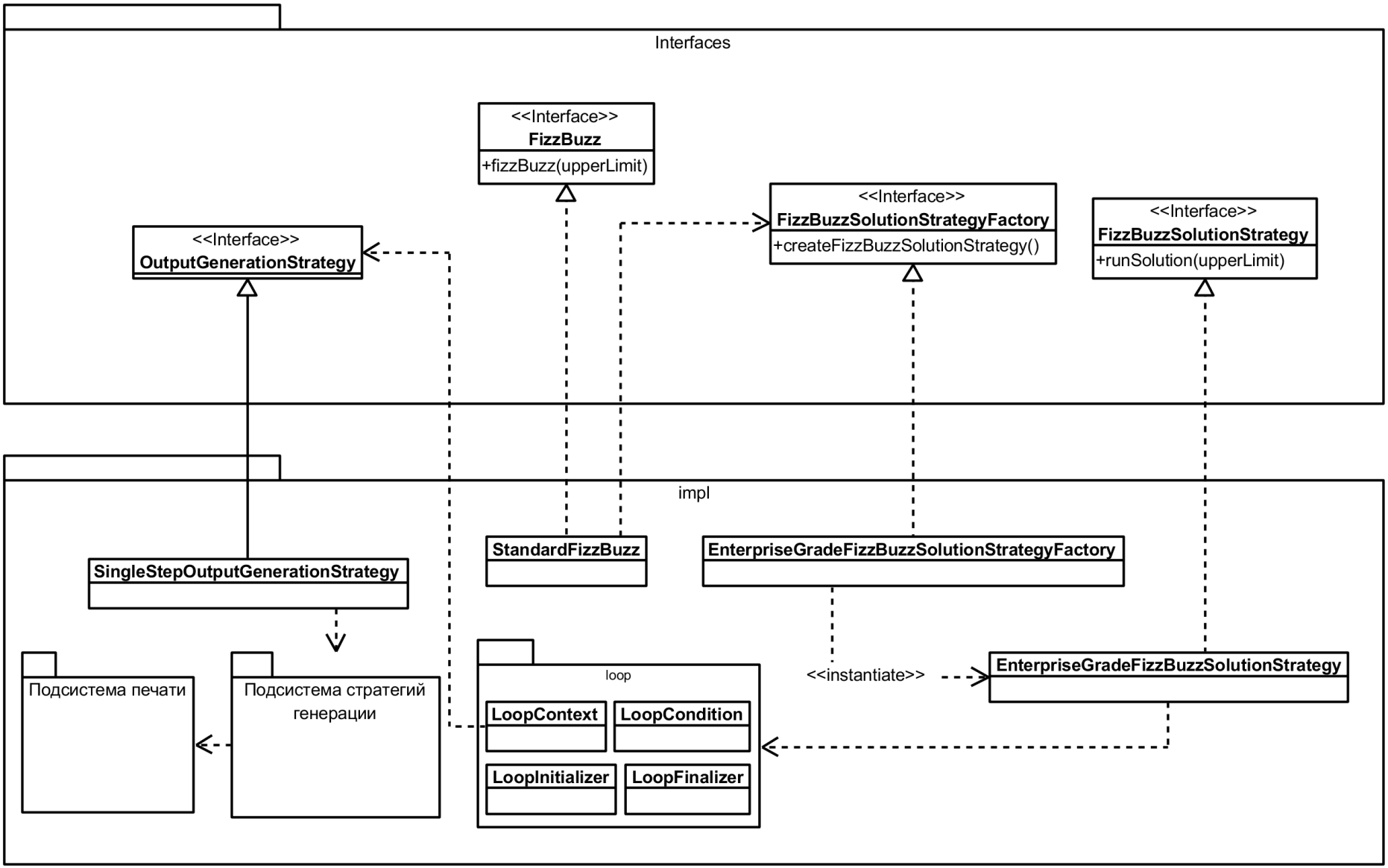
Enterprise Fizz-Buzz

Задача:
Для чисел от 1 до 100:

- если число делится на 3, вывести “Fizz”
- если число делится на 5, вывести “Buzz”
- если число делится и на 3, и на 5, вывести “FizzBuzz”
- во всех остальных случаях вывести само число

Решение:
https://github.com/EnterpriseQualityCoding/FizzBuzzEnterpriseEdition

Структура системы



Хорошие идеи

- Separation of Concerns
- Dependency Inversion
- Dependency Injection
  - Spring Framework
- Паттерны “Фабрика”, “Стратегия”, “Посетитель”, “Адаптер”, что-то вроде паттернов “Спецификация” и “Цепочка ответственности”

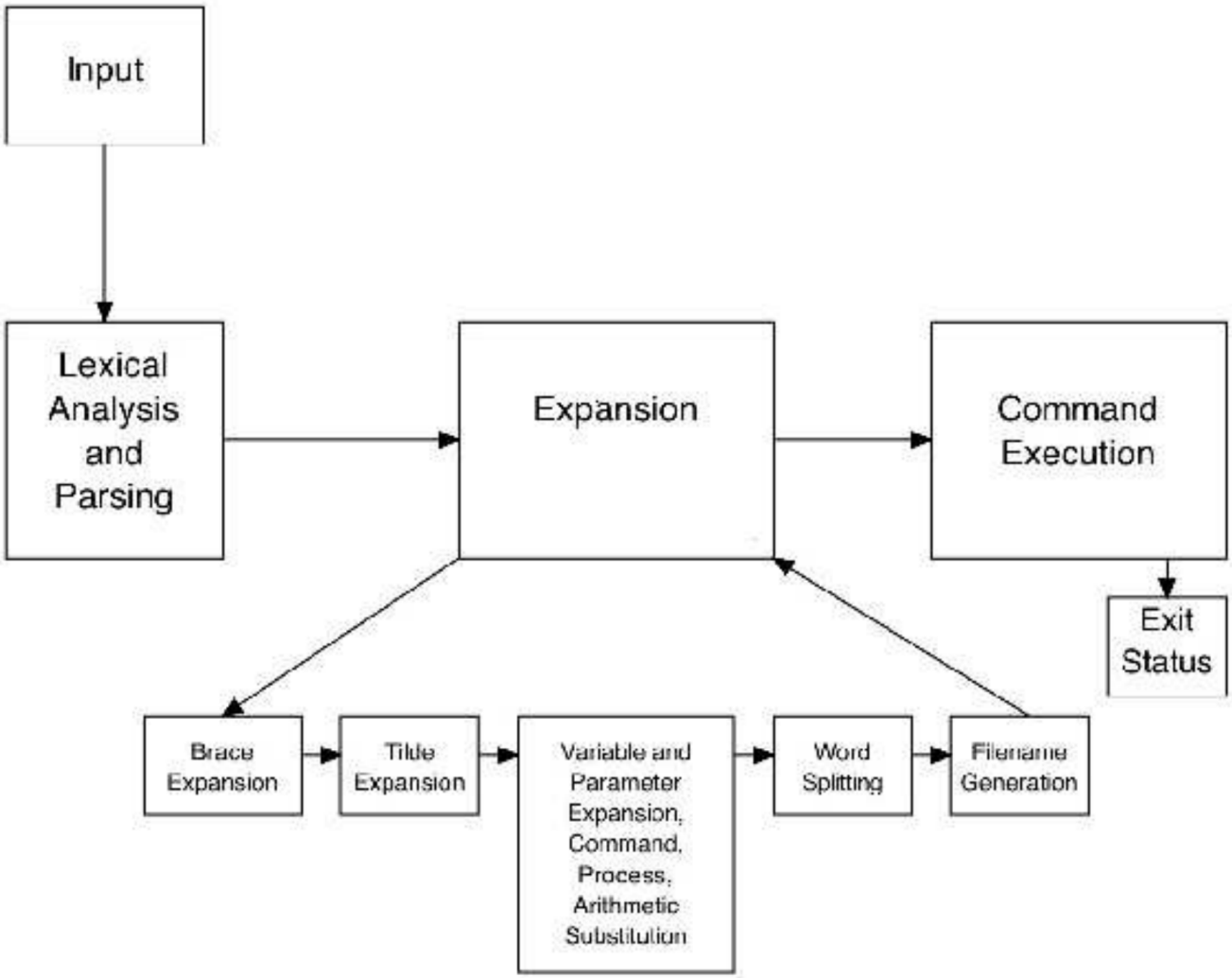
Плохие идеи

- Не выполняется принцип Keep It Simple Stupid
  - Неправильно говорить “строк кода написано”, правильно — “строк кода израсходовано”
- “Синтаксическое” разделение на пакеты, а не “семантическое”
  - Отсуствие модульности, антипаттерн “Big Ball of Mud”
- Хардкод основных параметров вычисления
- Нет юнит-тестов, только интеграционные; нет логирования
- 1663 строки кода и всего 40 строк комментариев
  - Отсутствие архитектурного описания

Bash

- Примерно 70K строк кода
- Исходный автор — Brian Fox, maintainer — Chet Ramey
- Первый релиз — 1989
- Написан на C
- Архитектурное описание — глава в *The Architecture of Open Source Applications*, написанная Chet Ramey

Архитектура Bash



Основные структуры данных

```
typedef struct word_desc {
    char *word; /* Zero terminated string. */
    int flags; /* Flags associated with this word. */
} WORD_DESC;

typedef struct word_list {
    struct word_list *next;
    WORD_DESC *word;
} WORD_LIST;
```

Ввод с консоли

- Библиотека Readline
  - независимая библиотека, но пишется в основном для Bash
- Цикл read/dispatch/execute/redisplay
- Dispatch table (или Keymap)
- Буфер редактирования, хитрый механизм расчёта действий для отображения
- Хранит все данные как 8-битные символы, но знает про Unicode

Синтаксический разбор

- Зависимый от контекста лексический анализ
  for for in **for**; **do for**=**for**; **done**; echo $for
- Использует lex + bison
- Подстановка alias-ов выполняется лексером
- Сохранение и восстановление состояния парсера

Подстановки

${parameter:-word}

раскрывается в *parameter*, если он установлен, и в *word*, если нет

pre{one,two,three}post

раскрывается в

preonepost pretwopost prethreepost

Ещё бывает подстановка тильды и арифметическая подстановка, сопоставление шаблона

Исполнение команд

- Встроенные и внешние команды, обрабатываются единообразно
- Перенаправление ввода-вывода, отмена перенаправления
- Принимают набор слов
  - Иногда обрабатывают по-особому, например, присваивание в *export*
- Присваивание — тоже команда, но особая
- Перед запуском внешней команды — поиск в PATH, кеширование результатов
- Job control, foreground и background

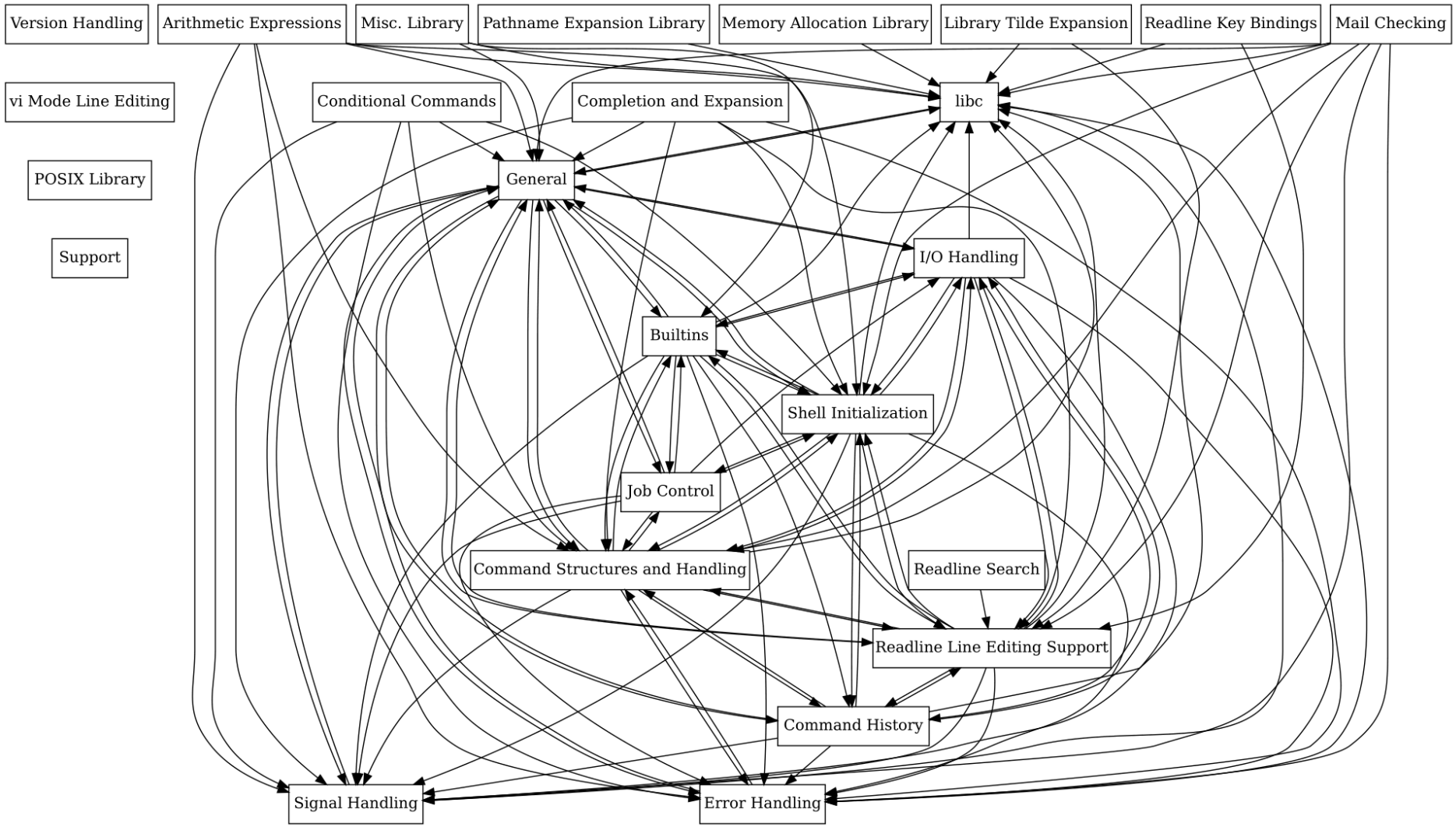
Lessons Learned

- Комментарии к коммитам со ссылками на багрепорты с шагами воспроизведения
- Хороший набор тестов, в Bash их тысячи
- Стандарты, как внешние на функциональность шелла, так и на код
- Пользовательская документация
- Переиспользование

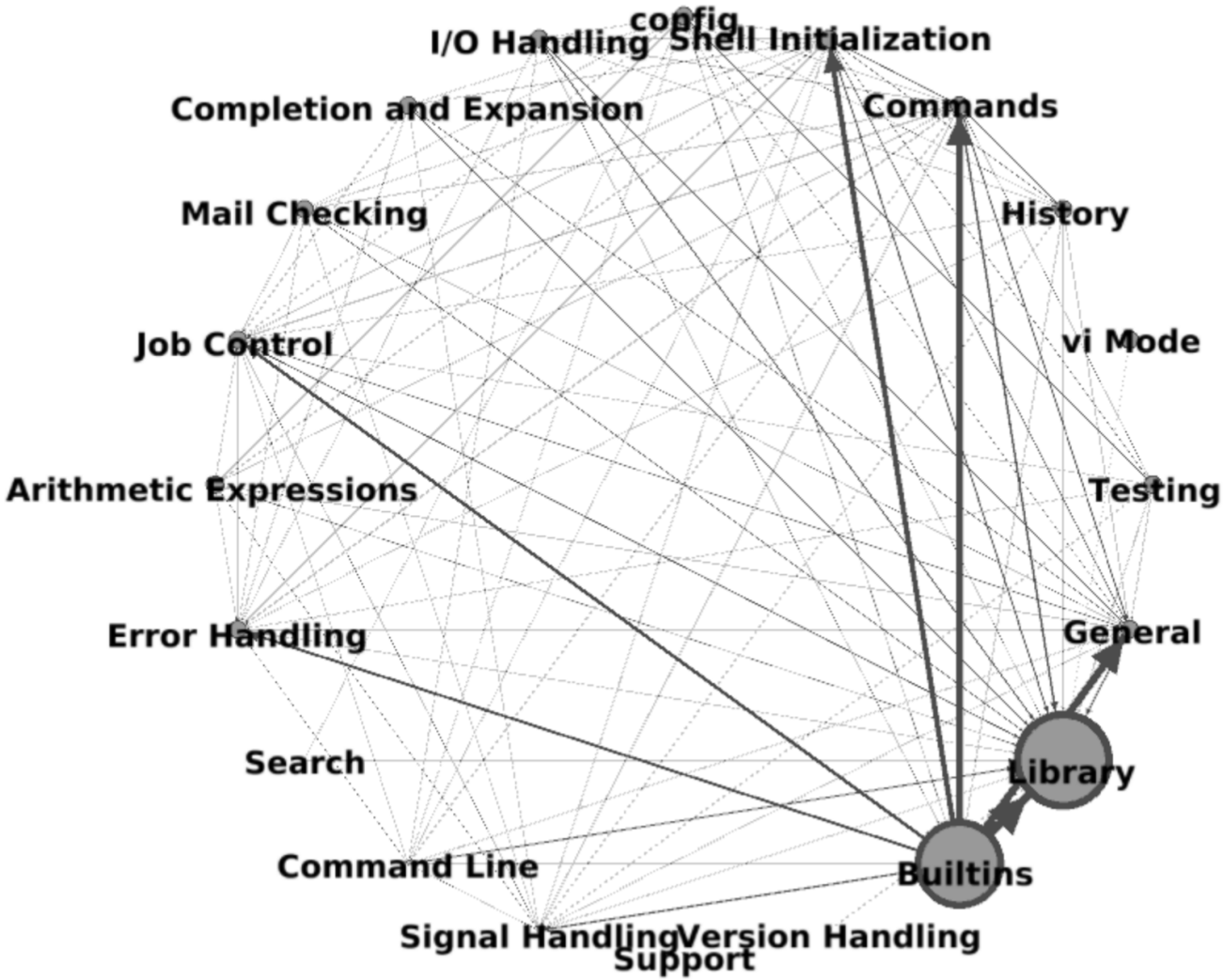
Архитектура Bash, на самом деле

- J. Garcia et al., *Obtaining Ground-Truth Software Architectures*
- 1 аспирант, 80 часов работы
- Верификация от Chet Ramey
- 70K строк кода, 200 файлов, 25 компонент
  - 16 — ядро, 9 — утилиты
- Структура папок почти не соответствует выделенным компонентам

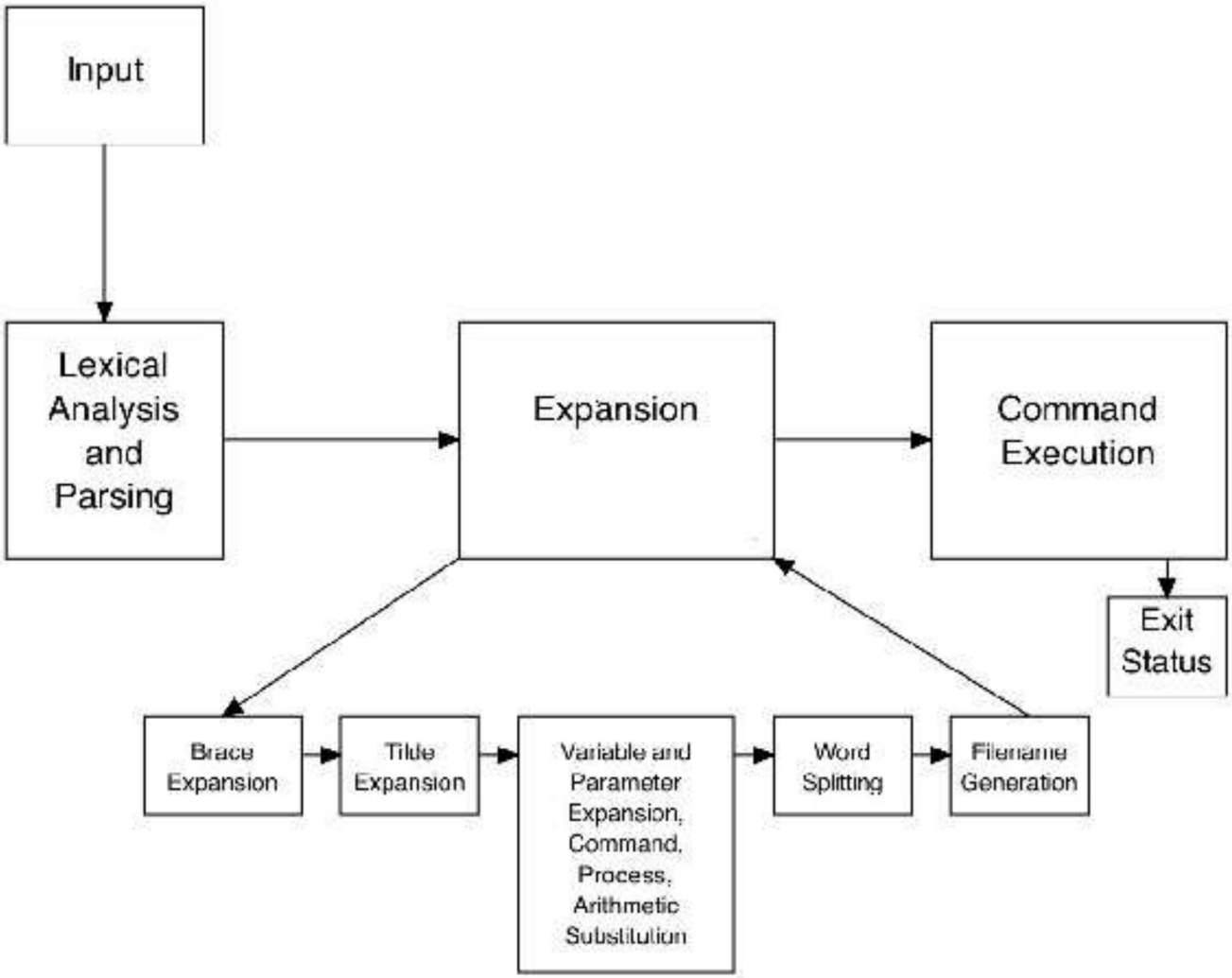
Архитектура Bash, на самом деле



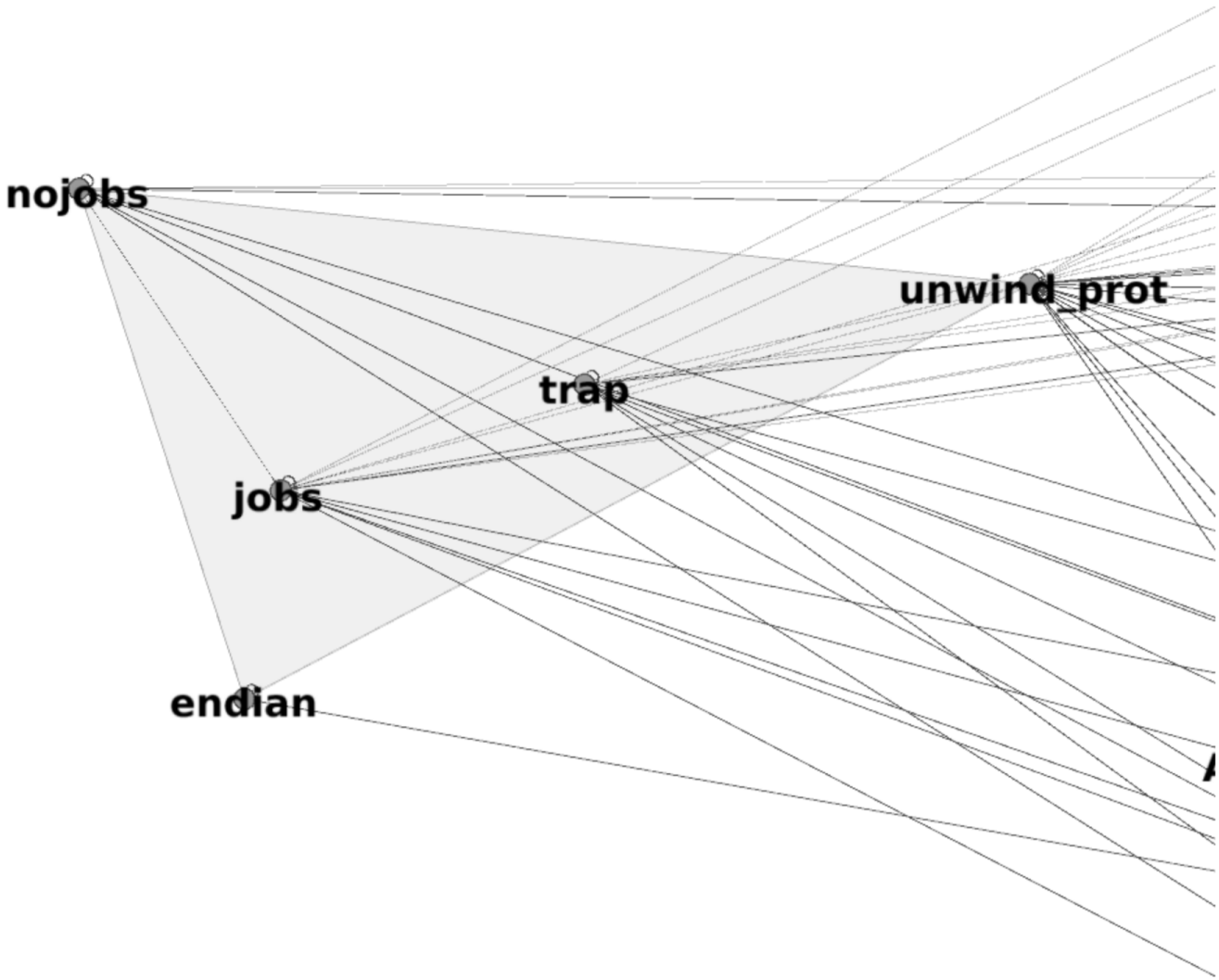
Результаты анализа кода



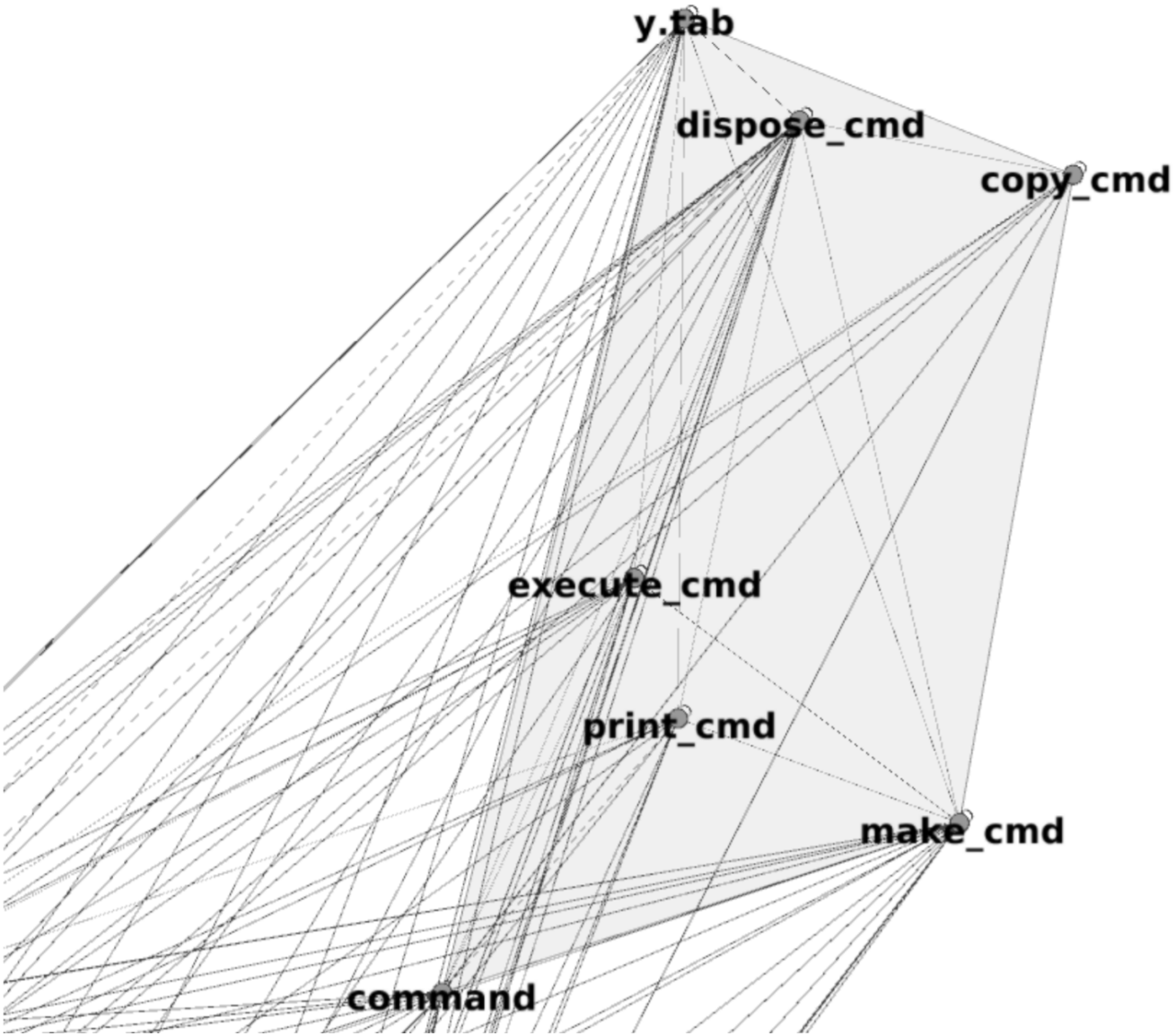
Сравним с исходной



Job Control



Commands


y.tab
dispose_cmd
copy_cmd
execute_cmd
print_cmd
make_cmd
command