Язык программирования

Язык и технология

расскажет Михаил В. Шохирев

Клуб программистов
Шадринск
2025

Большое ПО

Большинство современного ПО выполняется на серверах. И почти всегда оно большое, очень большое. Причём в разных измерениях:

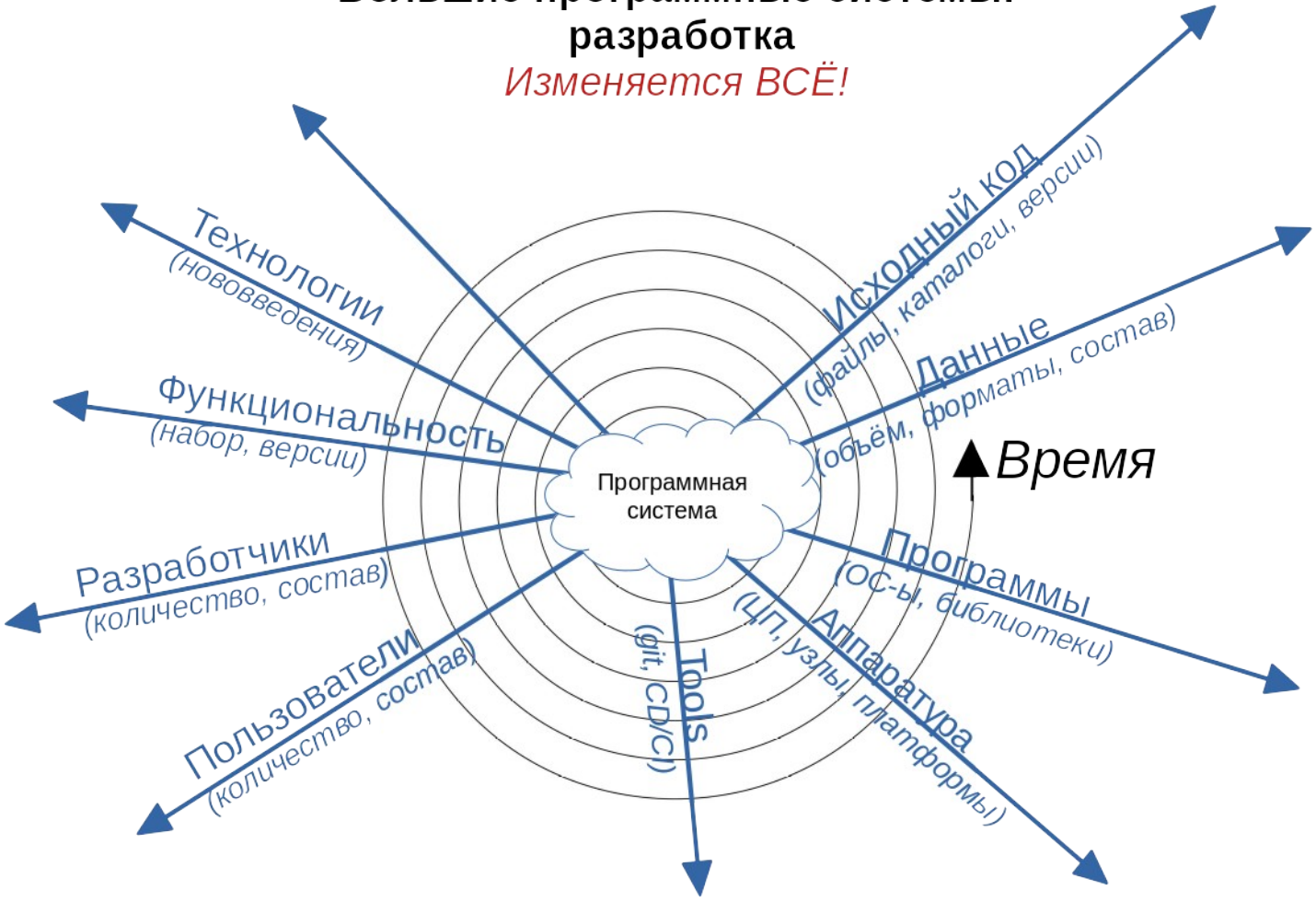
— состоит из большого количества исходных текстов;
  — которые расположены во множестве файлов и каталогов;
  — и должны изменяться параллельно;
— использует много стороннего ПО;
  — которое часто обновляется;
— создаётся большой командой разработчиков,
  — состав которой время от времени меняется,
— обладает широкой функциональностью;
  — которая должно постоянно эволюционировать;
  — поскольку меняются требования;
  — и новые возможности должны встраиваться в существующую систему;
— представлено в нескольких версиях и вариантах (prod, test, dev);

— используется длительное время;
— выполняется на многих ЦП, сетевых узлах;
— к нему обращается возрастающее количество клиентов;
  — часто с разных устройств (с разной аппаратной архитектурой);
  — из под разных ОС;

Разработка больших программ

Большие программные системы: разработка

Изменяется ВСЁ!



Проблемы и решения

Люди совершают ошибки.	— все переменные имеют «нулевое» начальное значение, нет неинициализированных переменных; — структуры языка простые и понятные, взаимно независимы в применении; — язык со статической и строгой типизацией; — компилятор очень строгий; — программа не скомпилируется, если есть неиспользуемые переменные или импортированные пакеты;

Проблемы и решения

Разные программисты пишут в собственных разных стилях («диалекты языка»).	— программа **`go fmt`** форматирует исходники одинаковым для всех способом; — строгий минималистичный синтаксис принуждает записывать алгоритмы единообразными конструкциями *(~~TIMTOWTDI~~)*; — исходники в единственной кодировке UTF-8;

Проблемы и решения

Программы должны изменяться разными программистами в течение длительного времени.	— ясный простой синтаксис способствует читабельности и хорошему пониманию программ; — правила видимости имён простые и понятные; — все имена полностью определяются идентификаторами пакетов; — синтаксис языка стабилен; — спецификация языка совместима с предыдущими и последующими версиями *(Go Compatibility Promise)*;

Программная система

— программный код (версии)
    — — исходные файлы в каталогах
    — — внешние библиотеки
— разработчики
    — — разной квалификации
    — — приходят и уходят
— средства разработки + процесс
    — — инструментарий
    — — тех. процесс (этапы)
    — — документация
— сервер
    — — железо: компьютеры (ЦП, память, диски, …) и коммуникационные средства
    — — системное ПО
— работающая система
    — — исполняемые файлы и библиотеки
    — — конфигурации, шаблоны (~неизменяемые)
    — — данные (изменяемые), в т. ч. мониторинга
— пользователи
    — — клиентское ПО
    — — люди и программы

Проблемы и решения

Программы должны постоянно развиваться (по мере изменения требований).	— нет необходимости зависеть от жёсткой иерархии классов; — композиция вместо наследования позволяет программным компонентам эволюционировать независимо друг от друга; — интерфейсы позволяют прикреплять новые действия к уже существующим компонентам, а также легко применять действия от старых компонентов в новых; — функции — это полноценные типы данных, применение которых даёт гибкость при взаимодействии компонентов *(делаем с действиями, что пожелаем)*; — богатая стандартная библиотека упрощает добавление новой функциональности; — сообщество и компании разрабатывают много дополнительных специализированных модулей;

Проблемы и решения

Программы должны эволюционировать в течение долгого времени.	— синтаксис совместим с предыдущими и последующими версиями языка; — система модулей позволяет обновлять их до новых версий и использовать разные версии параллельно; — есть инструменты для обнаружения ошибок и уязвимостей в старых версиях модулей для их обновления; — обновления внешних системных и сторонних модулей делаются легко;

Проблемы и решения

Программы должны модифицироваться многими разработчиками.	— исходные тексты свободно располагаются в разных файлах и каталогах проекта и легко объединяются через **`go.mod`**; — один пакет состоит из множества файлов в каталоге, которые могут независимо изменяться разными людьми; — унифицированное представление исходников (с помощью **`go fmt`**) упрощает выявление изменений в репозитарии;

Проблемы и решения

Серверные программы имеют большой размер.	— система импортирования пакетов минимизирует включения при компоновке программ; — эффективный компилятор быстро обрабатывает большую кодовую базу; — программный код одного пакета можно располагать в одном или в разных файлах в каталоге *(не изменять, а добавлять)*;

Проблемы и решения

Серверные программы должны эффективно использовать аппаратные ресурсы.	— одновременность (concurrency) реализована как встроенный в язык механизм (задействующий все ядра ЦП), поэтому её легко использовать понятным образом; — сборщик мусора (GC) эффективно управляет распределением и освобождением оперативной памяти; — исходники компилируются в быстро исполняемые двоичные программы;

Проблемы и решения

Программы должны иметь возможность выполняться на разных аппаратных платформах под разными ОС.	— очень легко выполнить компиляцию исходной программы на новую архитектуру и ОС *(для настройки задаются 2 переменные окружения)*; — поддерживаются все основные ОС и платформы «железа»; — есть программное средство для разработки программ, которые будут выполняться на «голом железе» *(bare metal)*; — есть TinyGo для разработки встроенных систем; — можно компилировать в JavaScript или WASM для выполнения программ в браузерах;

Проблемы и решения

Программы должны тестироваться в процессе разработки.	— в системе программирования Go поставляются стандартные средства тестирования (включая измерение покрытия тестами); — тестовые программы располагаются рядом с исходниками, но не включаются в исполняемую программу; — средства профилирования также стандартные; — ещё больше средств тестирования разработано сообществом разработчиков Go; — все средства тестирования, отладки, телеметрии можно удобно объединять в конвейеры для автоматизации разработки;

Проблемы и решения

Разработка программ должна быть автоматизирована.	— в распоряжении разработчика — богатый набор стандартных инструментов *(go build, doc, fmt, get, mod, test, vet, ...)*; — есть доступный способ улучшать и добавлять инструменты разработчика; — сообщество Go расширяет набор средств для автоматизации разработки; — распределённая система модулей с идентификацией по URL упрощает автоматизацию и масштабирование; — синтаксис языка предусматривает простоту создания инструментария; — у системы программирования Go открытые исходники;