



Диалектика Гегеля и Закон Седова как способ верификации IT трендов с примерами из Автоматизации тестирования

Антон Семенченко



Founder of communities www.COMAQA.BY, www.CoreHard.by, www.InterIT.by and www.ITUp.by, co-founder of company www.DPI.Solutions, «tricky» manager at EPAM Systems. Almost 15 years of experience in IT, main specialization: Automation, C++ and lower development, management, sales.

План беседы

1. Формулировка проблемы: сложность прогнозирования
2. Вариант решения: поиск IT - «лакмусовой бумажки»
3. «Научная» база решения
 - Диалектика Гегеля
 - 5 признаков сложных систем по Гради Бучу
 - Закон «Иерархических компенсаций» Седова
4. «Исторические» примеры
5. Некоторые тренды Автоматизации тестирования
6. Примеры реальных стратегических задач в Автоматизации тестирования
7. Вопросы

Проблема

Прогнозирование ... в Автоматизации тестирования и IT в целом ...
как на личном уровне, так и на уровне целой компании



Взгляд в будущее

Туман, туман, седая пелена

А всего в двух шагах за туманами ...



Решение ...

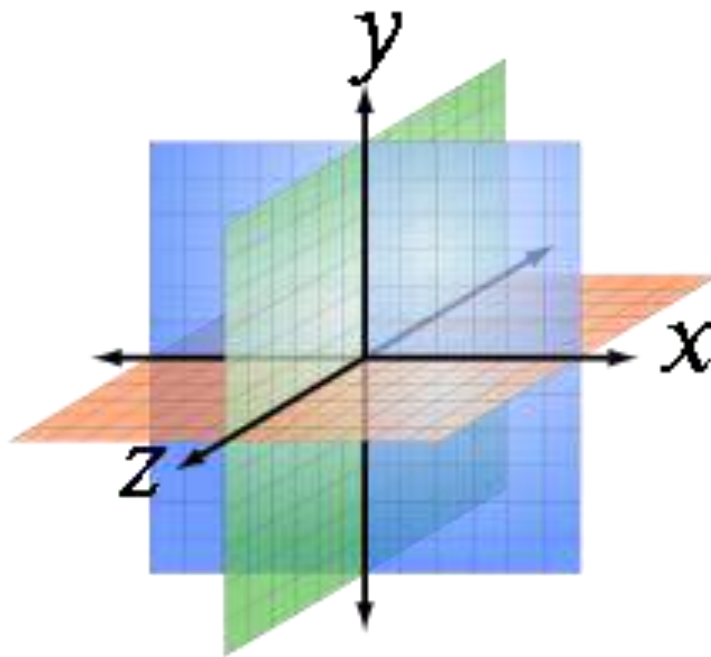


Не сформулировать текущие тренды!



Инструмент прогнозирования на базе:

- Диалектика Гегеля
- 5 признаков сложных систем по Гради Бучу
- Закон «Иерархических компенсаций» Седова

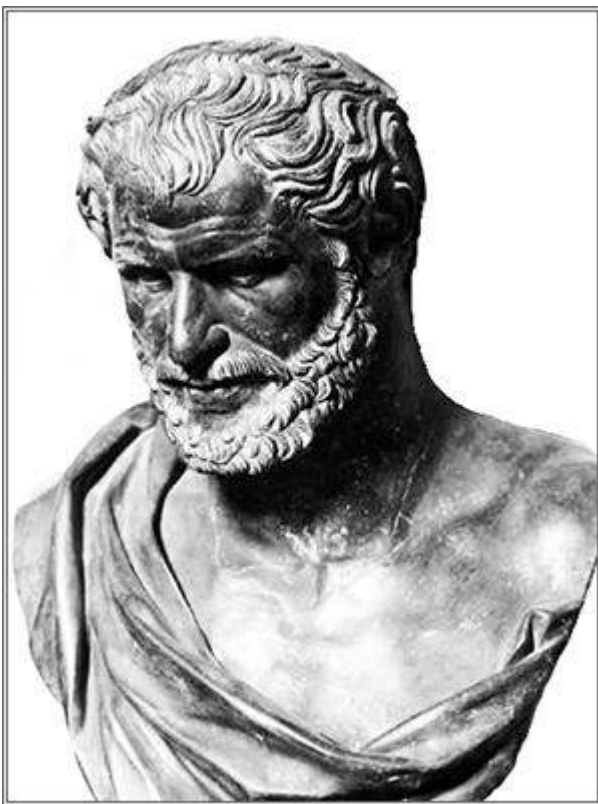


Диалектика Гегеля



Диалектика (в диалектическом материализме) – общая теория развития материального мира и вместе с тем теория и логика познания.

Гераклит - первооткрыватель



- «Все течет, все изменяется»
За «создание» Диалектики Гераклит получил прозвище «темный», то есть непонятный. Даже Сократ не всегда понимал Гераклита, но очень его уважал.

Мнение Кьеркегора



- «Я думаю, что те места у Гегеля, которые я не понимаю, он сам тоже не понимал»
- «“ Неспособность понять великого человека” (Гегеля) есть мой “позор” и “несчастье”»
- «Я вступаю в полемику с Гегелем, а в его лице – со всей западной философией, берущей начало в античности»

«Бытовая» диалектика

Каждый человек, на своем простом житейском уровне, порой напрягает мозги, стараясь уразуметь, как же это так **странно, нелогично и противоречиво** устроена жизнь: должно быть вот эдак, а на самом деле почему-то наоборот. И приходит к простым выводам, с формулировками вроде:

- **«Слишком хорошо - тоже не очень хорошо»**
- **«Делаешь-делаешь хорошо - а в результате выходит плохо»**
- **«Противоположности сходятся»**

Это даже не объяснение, а констатация часто встречающегося положения, результат опыта, наблюдений за жизнью!

Фицджеральд о диалектике



Скотт Фицджеральд, нормальный малообразованный 😊 Американец, сказал:

- «Признаком первоклассных мозгов является способность держать в голове две взаимоисключающие мысли одновременно, не теряя при этом способности соображать»

Вот это, в переводе на общепринятый язык, и есть диалектика.

Диалектику невозможно выучить!

- Выучить диалектику невозможно.
- Запоминание ничего не даст.
- Требуется неторопливое, последовательное думанье - единственный способ и средство понимания всего на свете.



Прозрение

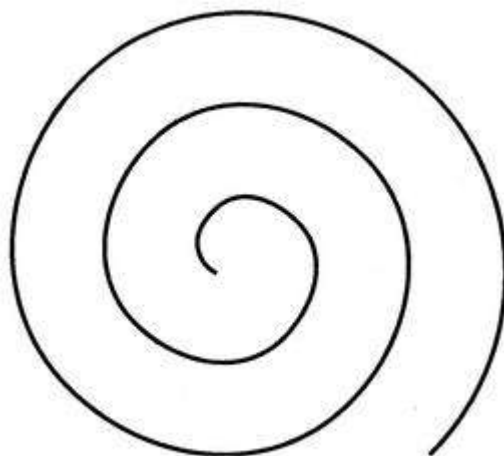


Диалектика Гегеля

Диалектическая триада: «тезис – антитезис – синтез»

«Тезис» - некоторая идея, теория или движение

«Антитезис» - «оппозиция» к тезису, негативное утверждение



Диалектика Гегеля

Диалектическая триада: «тезис – антитезис – синтез»

«Синтез» - противоположность «тезиса» и «антитезиса»
продолжается до тех пор, пока не находится такое решение,
которое в каких-то отношениях выходит за рамки и «тезиса», и
«антитезиса», не исключая их относительную ценность и пытаясь
сохранить их достоинства и избежать недостатков

**Однажды достигнутый «синтез», в свою очередь, может стать
первой ступенью новой диалектической триады**

Диалектика Гегеля

Закон «Единства и борьбы противоположностей»

Закон «Перехода количественных изменений в качественные»

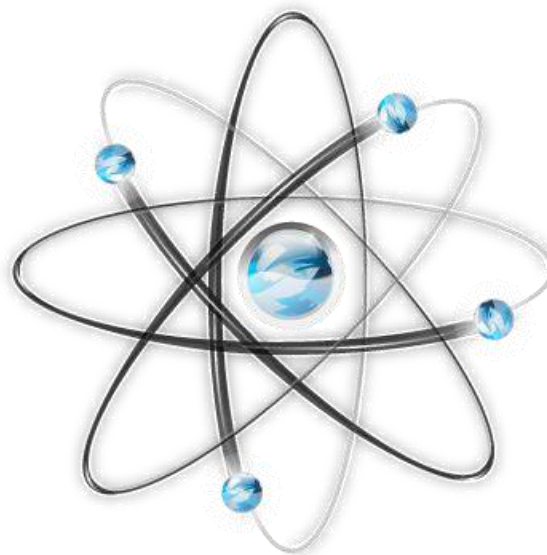
Закон «Отрицания отрицания»



Единства и борьбы

Закон «Единства и борьбы противоположностей»

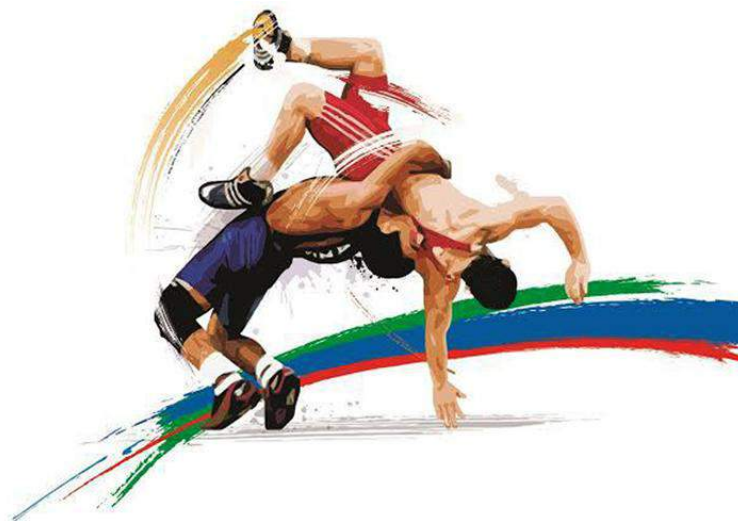
- Принцип «Корпускулярно-волнового дуализма»
- Биологическая эволюция - борьба наследственности и изменчивости



Единства и борьбы

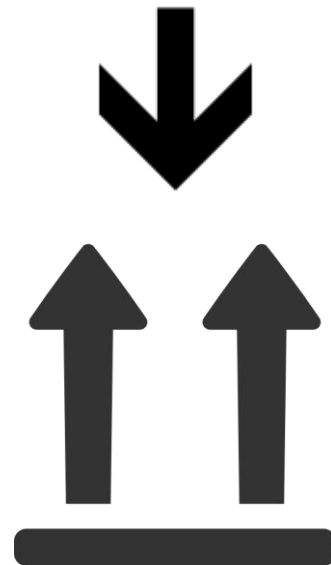
Борьба - условный термин. Борьбы в обычном смысле слова нет, просто одно противостоит другому, активно взаимодействует с ним.

Как сказал киногерой: «**Есть хочется ... худеть хочется ... всего хочется!...**» Вот это и есть единство и борьба противоположностей.



Верх и низ

Все, что имеет верх, имеет и низ. Одно без другого никак невозможно. Мы и определяем одно через другое. Берем «чего-то» два, даем им названия - и противопоставляем друг другу. А на самом деле это просто **две разные стороны одного и того же**. Нет верха - нет и низа. Как, скажем, у шара в космическом пространстве.



Единства и борьбы

Лево и право

Аналогично верх \ низ. Ну возможно ли, чтоб лево было, а право - нет? В том и суть, что это две противоположные стороны, и одна определяется относительно другой.



Единства и борьбы



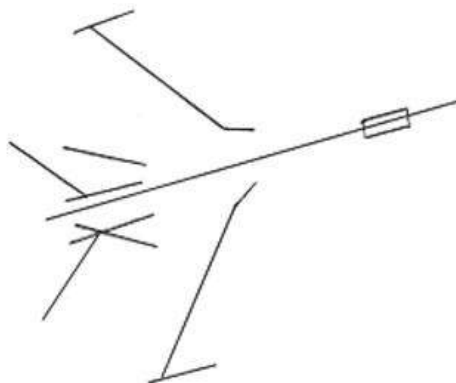
Рабочий и работодатель

Рабочий хочет меньше работать и больше получать. Владелец завода хочет меньше ему платить, а чтоб он побольше выработывал.

Так и живут в классовом противоречии и компромиссе. Друг без друга им никак.

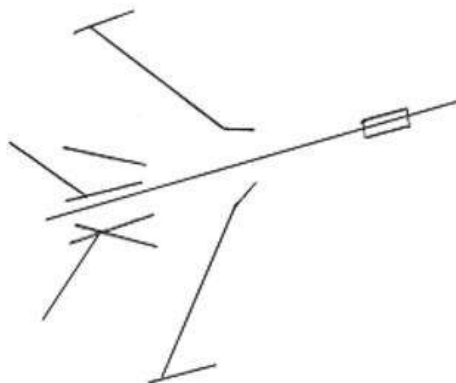
Самолет: полет \ падение

Вот летит самолет. Он тяжелый, и поэтому хочет упасть. Но двигатели прут его вперед, и на скорости воздух под его крылом давит крыло кверху и хочет поднять выше, выше, выше. Вот в единстве этих противоположных стремлений - крыло хочет вверх, а фюзеляж хочет вниз - самолет и держится на одной высоте, части его скреплены прочно.



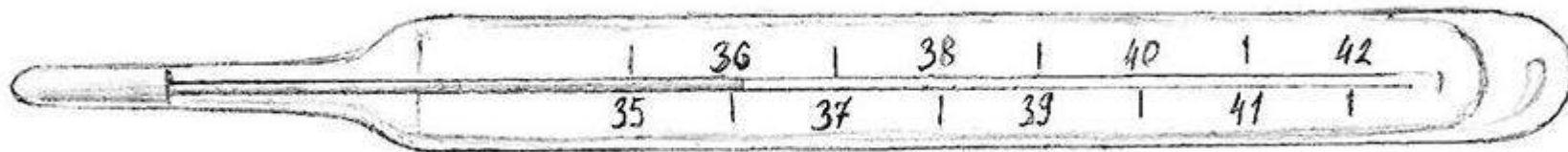
Стремление к счастью

Человек стремится к счастью, а попутно добывает себе хлопоты и переживания. Он бы предпочел обойтись без них, да так не бывает. «Без труда не вытянешь рыбку из пруда». Это всегда вместе.



Тепло - холодно

Если бы всегда жили при одной и той же температуре и даже не знали, что возможна другая - не было бы у нас этих понятий. Ну, вот такова среда нашего обитания, чего тут скажешь. Вроде как воздух до эпохи ныряния и полетов ввысь - он плотный или разреженный? Идиотский вопрос - воздух он и есть воздух, вы что имеете в виду? А поскольку разница температур каждому известна, понятия тепла и холода противопоставлены друг другу.



Единства и борьбы

Хорошо - плохо

Это опять же одно относительно другого. На что и с какой стороны взглянуть. Если боль - плохо, то хорошо есть ее отсутствие. Если богатство хорошо, то плохо есть его отсутствие. Одно понятие есть противопоставление другому



Диалектические пары

Все вышеперечисленное называется диалектические пары.

Почему веревка натянута? Потому что ее тянут за оба конца в разные стороны.

Вот все на свете внутри себя устроено как эта веревка. Это вот в каком смысле:



Тенденции

- Одна тенденция: **больше танков**: задавим врага! Другая тенденция: **к черту танки!** топливо сожрут, дороги загромоздят! Результат - **равнодействующая: некое разумное количество войска.**
- **Дереву** - расти выше! вылезти из чащи к солнцу, брать листвой как можно больше его энергии. Нет - ниже: устойчивее быть, крепче, чтоб ветер не свалил. **Результат: оптимальная высота.**



Тенденции

- Хорошо бы у всех все забрать себе. Еще хорошо быть добрым, всем все раздать, любить будут. Ладно, хапну втихаря немного, а чуть-чуть дам друзьям. Нет? 😊
- К чему мы неизбежно придем? К смерти. А что мы делаем? Да живем как можем. Это и называется единство и борьба противоположностей



Тенденции

- Надо быть абсолютно свободным и независимым от всех. Но если не ограничить всех законами государства, то самый сильный и агрессивный начнет всех убивать и грабить. Ограничить законами! Но не слишком... и здесь противоположные тенденции.



Единства и борьбы

Противоположные тенденции всегда сдерживают друг друга!

- Не то самолет или в космос улетит, или гряпнется. Если бы живые существа не умирали - жил бы папоротник, не превратившись постепенно в человека. И места бы человеку не было.
- Сила гравитации хочет собрать все вещество Земли в маленький сверхплотный центр. А центробежная сила хочет разметать все ее вещество в стороны, в космос. Вот и живем мы на круглой планете.



Диалектика Гегеля

Закон «Перехода количественных изменений в качественные»

- Превращение «Лёд – Вода – Пар»



Количество в качество

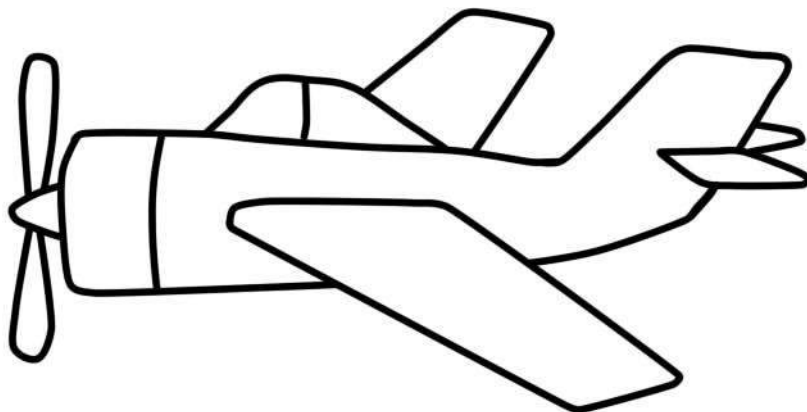
У врага есть танки. Много. Пять тысяч. Чтобы победить, нам тоже нужны танки. И побольше. Чтоб - наверняка. Десять тысяч. Два наших на один «ихний» 😊 Они его победят. А вдруг нет?.. Ладно! Сделаем пятьдесят тысяч танков - и враз его разнесем, да он и не посмеет полезть. Готово! И что? Эта армада сожрала все горючее, загромоздила все дороги, обученных экипажей не хватает - и гигантская бронированная пробка загромоздила все пространство, без толку мешая друг другу, теснясь мертвым грузом. И сжег их враг меньшими силами.

Нарастив сил сверх меры, оказались на деле бессильными.



Так два легиона Лукулла обратили в прах двухсоттысячное войско Тиграна - те в давке больше сами себя подавили.

Вот так СССР создал столь мощную, эшелонированную и структурированную систему ПВО, что авиетка Руста беспрепятственно села на Красной площади. Сверхгигант не в силах сдвинуть собственную тяжесть.



Количество в качество

Время поездки равно расстоянию, поделенному на скорость. Сделали автомобиль со скоростью 300 км/час. Сели, газанули, поехали. Сцепление с дорогой мало, поворот, кювет, дерево, больница, кладбище. Сократили время пути?.. Подумали, написали эпитафию - поговорку «Тише едешь - дальше будешь». Звучит, вроде, противоречиво, неправильно, - но смысл всем ясен и житейски верен. «Поспешись - людей насмешишь».

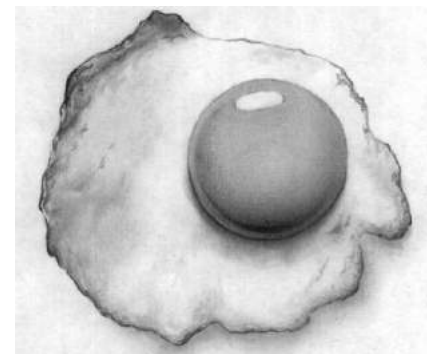
Нарастив чрезмерную скорость, вообще не доехали до места.



Количество в качество

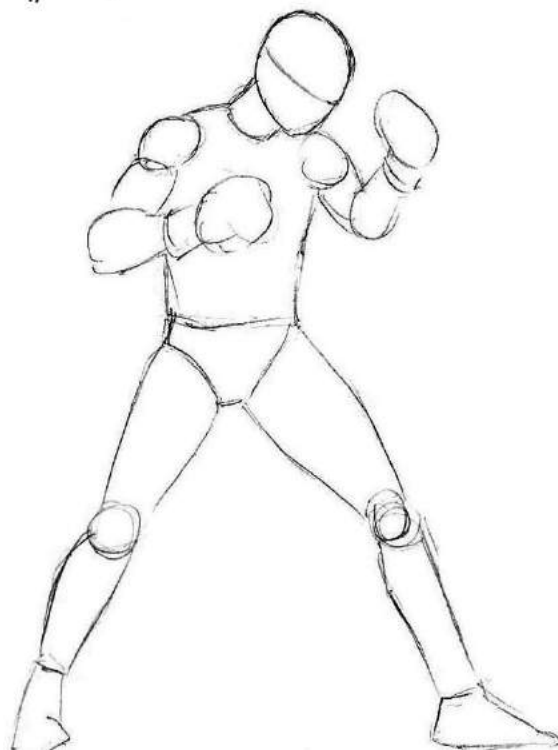
Хилому ребенку с плохим аппетитом объясняют: будешь много кушать - станешь сильным и здоровым. Кормят пичкают, убеждают, - ребенок начинает есть, как землеройная машина, - и в конце концов становится полным, тучным, малоподвижным, сердце не справляется, почки не справляются, готов инвалид и кандидат в покойники. Докормили.

Съедая сверх меры необходимых для жизни и здоровья продуктов - угробили здоровье и жизнь.



Количество в качество

Для высоких результатов в спорте необходимы усиленные и частые тренировки. Стал тренироваться с утра до ночи, утомился, ослаб, сорвал сердце, нарушился обмен веществ, стал инвалидом.



Количество в качество

- Больше оружия: вместо победы - поражение.
- Больше скорости: вместо езды - авария.
- Больше еды: вместо здоровья - болезнь.
- Больше тренировок: вместо рекордов - инвалид.



Количество в качество

Ты делаешь правильные усилия, совершаешь правильные действия для достижения нужного результата. Но если вовремя не остановиться, то те же усилия и действия начнут уводить тебя от этого результата как бы уже в другую сторону: **ты переходишь** **нужную тебе грань** и начинаешь от нее удаляться, пока не **придешь к обратному тому, чего хотел.**

Поэтому и говорят: «**Все хорошо в меру**».

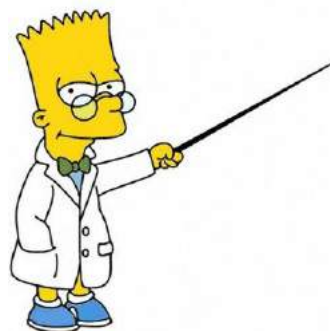
Мера - это соответствие количества твоих действий тому результату, которого ты ими хотел достичь.

Вот и во всей природе точно то же самое.



Количество в качество

- Хотели вскипятить воды чайку попить, а она вся и выкипела. От огня количество тепла в воде все увеличивалось, пока вода не изменила все свои качества и не перестала вообще быть водой: жидкость превратилась в пар, газ.
- Откованную сталь решили для закала, прочности, охладить жидким азотом: она охладилась до минус ста и стала хрупкой, как стекло. А охладили бы только до плюс двадцати - был бы булатный клинок.



Количество в качество

Любой процесс, если продолжается бесконечно, в конце концов приобретает какие-то новые, иные черты, свойства, качества. Те самые действия, что его вызывали, начинают в конце концов иметь результатом не то, что имели результатом сначала, раньше, до определенной границы.



Закон «Отрицания отрицания»

- Математический пример: «возьмём положительное число a , подвергнем его отрицанию и получим $-a$ (минус a). Если же мы подвергнем отрицанию это отрицание, помножив $-a$ на $-a$, то получим $+a^2$ (a в квадрате), то есть первоначальную положительную величину, но на более высокой ступени»



Отрицание отрицания

Преобразование «Лёд — Вода — Пар»

- Вода при кипении превратилась в пар и тем отрицает воду!
- Пар «обречен» на конденсацию и превращение в воду!
- Произошло отрицание отрицания.
- Переход Лед-Вода и обратно - так же является отрицанием отрицания



Отрицание отрицания

Ненаучное «название» закона - **Переход в новое состояние.**

Этот закон вытекает из предыдущего, он его родственник и сосед.

Частично «обыватель» сформулировал его так: «Ничто не вечно под луной».

Хозяйка захотела есть, купила продукты. Нет больше у нее денег - есть продукты. Спекла пирог - нет больше яиц, муки, сметаны, сахара, превратились в пирог. Съела пирог - нет больше пирога; переваривается в желудке питательная масса. Продолжать? 😊



Отрицание отрицания

Каким бы ни был процесс - в основе своей он состоит из каких-то действий. Появляются новые клетки в организме - это действия. Растрескивается камень веками, превращаясь в песок, - появление трещины тоже действие, механическое, природное. Яйцо разбивается над кастрюлей - действие.

А любое действие - это какое-то изменение. Что-то стало в мире хоть чуть-чуть не так, как было раньше.

Изменение - это тот механизм, который всегда лежит в основе любого процесса.



Отрицание отрицания

Даже гранитная скала - нагревается-охлаждается, нагревается-охлаждается, и так каждый день и каждую ночь. Через миллион лет нет больше скалы - есть песок на ее месте.

За что бы мы в мире ни схватились - когда-то на его месте было что-то другое. И когда-нибудь будет что-нибудь другое. Такие дела. И без этого никак.



Отрицание отрицания

Дерево все свои лучшие соки отдало маленькому каштану. Раскрылась кожура, и упал он на землю - красивый, круглый, крепкий, глянцевый. Полил дождик, лопнул каштан, пустил корешок, зацепился он за землю, и стало расти новое дерево. А где каштан? Нет его больше, умер. Зато вырос лес. Росли-росли деревья, состарились, упали, гнили-гнили - превратились в нефть: выкачали ее, выделили бензин, залили в машины - и превратилось зеленое дерево в тот газ, который мы вдыхаем в городах. Раньше деревья поглощали углекислый газ и выделяли кислород - а теперь что? А теперь получившийся из них бензин сжигает кислород атмосферы.



Отрицание отрицания

Бросили камень вверх. Упал он вниз и разбился. Та самая сила, что бросила его вверх - послужила причиной его падения, а то бы он спокойно лежал. Чтоб полететь вниз - надо сначала полететь вверх.

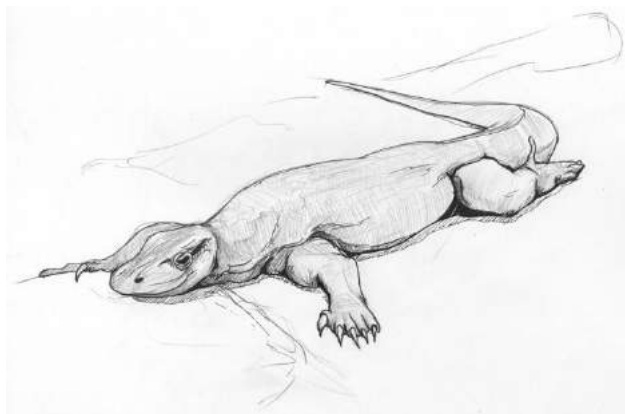
Вот так в каждом явлении, вещи, действии заключен механизм, который послужил его причиной, есть основа его существования - и он же приводит его к концу. И не просто к концу - а превращает его в нечто вовсе иное, чем было раньше, и даже в обратное противоположное.



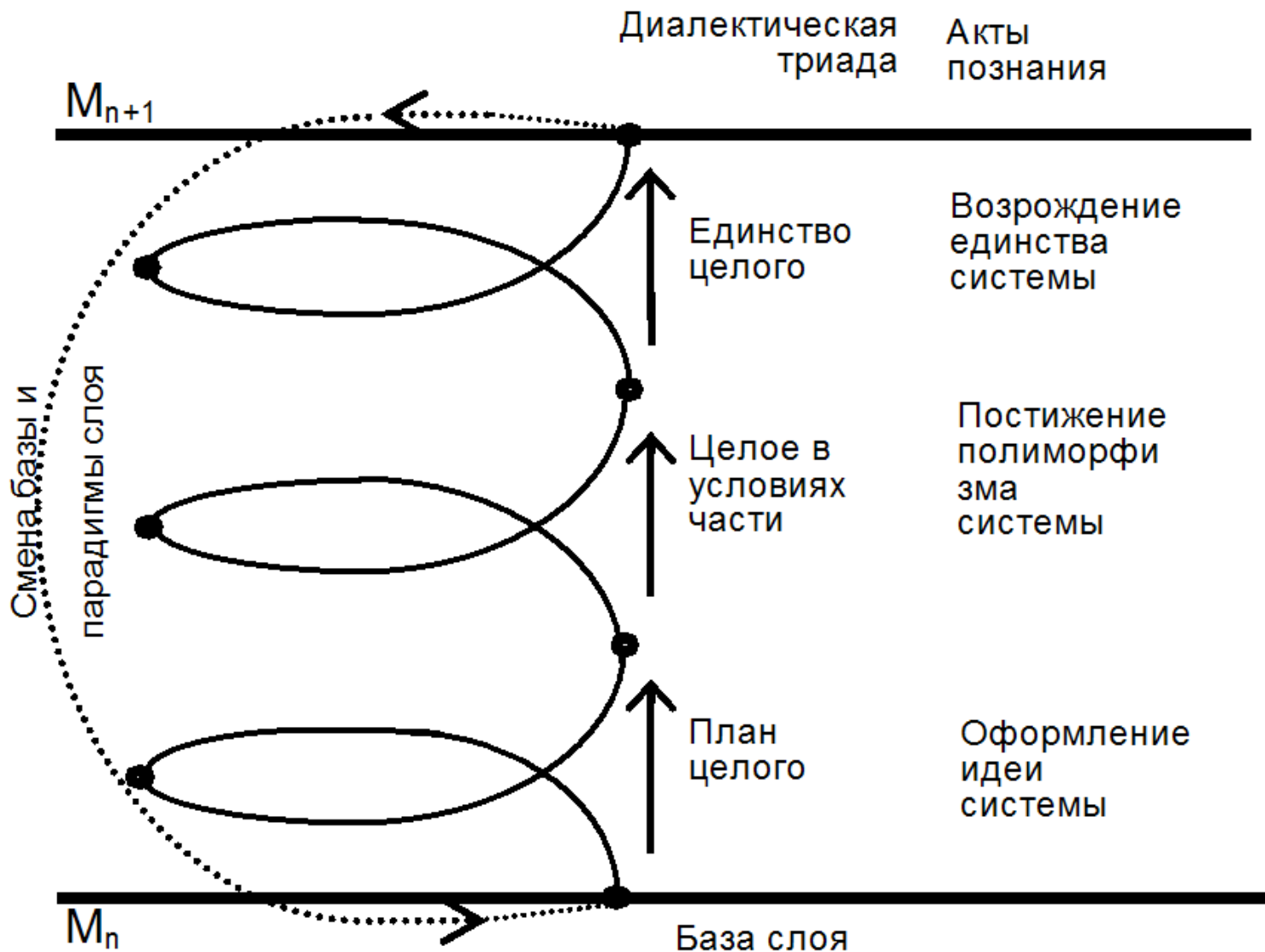
Отрицание отрицания

Если что-то есть - оно получилось из чего-то. До этого на его месте было что-то другое. А из этого когда-нибудь получится что-то новое - потому что всегда происходят какие-нибудь изменения.

На месте курицы было яйцо, на месте города была степь, на месте пустыни был город, на месте человека был другой человек, его прадед, а на его месте была обезьяна, а на ее месте была ящерица.

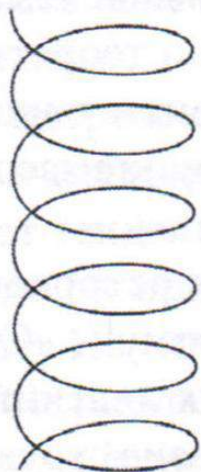


Диалектика Гегеля



Диалектика Гегеля

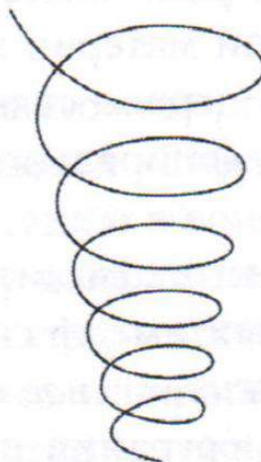
видимость обновления



догматический круговорот

а) Тоталитарный застой
(абсолютизация устойчивости)

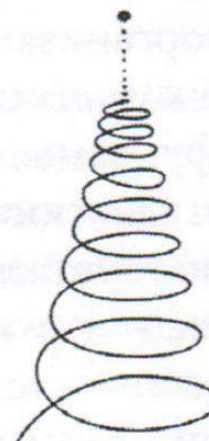
обновление без границ



духовный бунт

б) Анархический взрыв
(абсолютизация изменчивости)

сбалансированное обновление



появление путеводной звезды
и «обретение смысла жизни»

в) Творческий порыв
(гармонизация устойчивости и
изменчивости, золотая спираль)

«Изменяясь,
воскресаю
неизменным»
(Я. Бернулли)

Качественное описание закона самоорганизации. Основные типы спиралей (а, б, в), представляющие философский интерес

Мы

диалектику

учили не по Гегелю.

Бряцанием боев

она врывалась в стих,

когда

под пулями

от нас буржуи бегали,

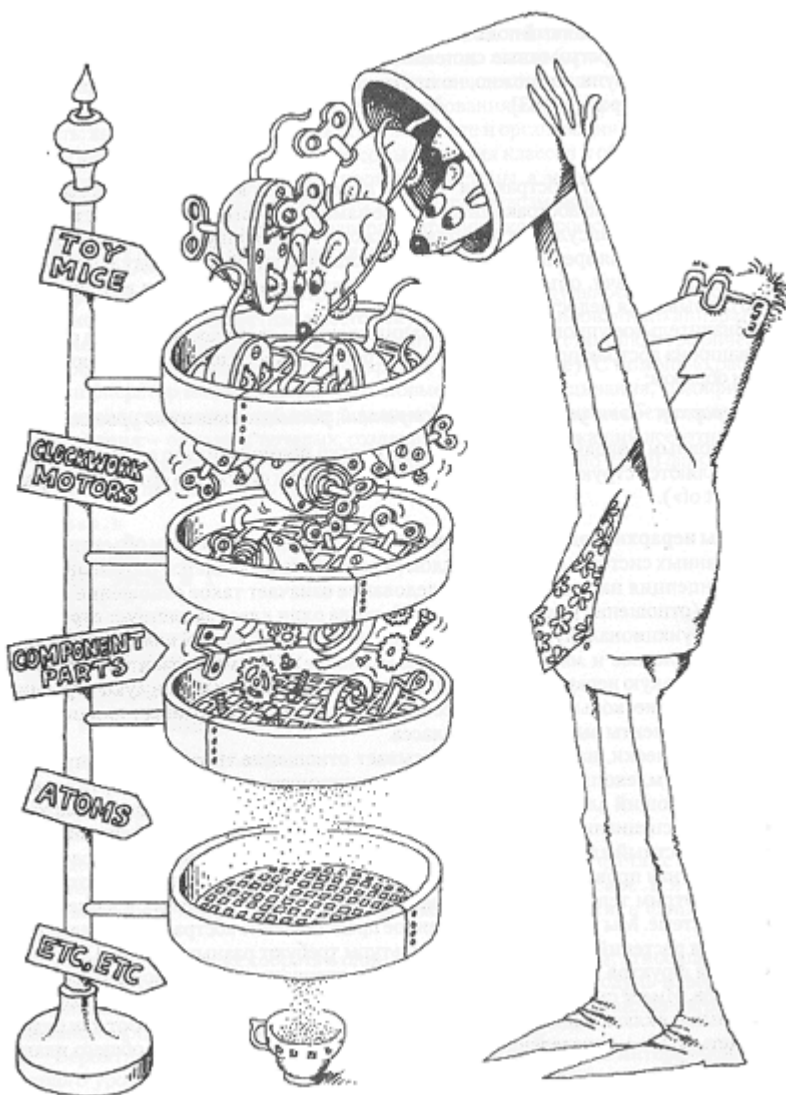
как мы

когда-то

бегали от них.

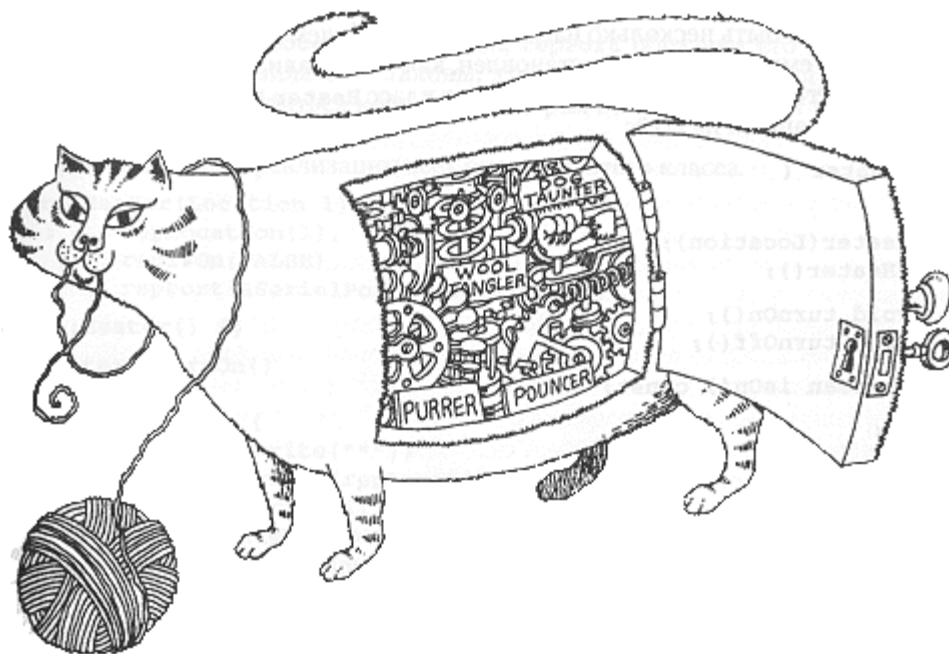


Гради Буч



5 признаков сложных систем

1. "Сложные системы часто являются иерархическими и состоят из взаимозависимых подсистем, которые в свою очередь также могут быть разделены на подсистемы, и т.д., вплоть до самого низкого уровня."



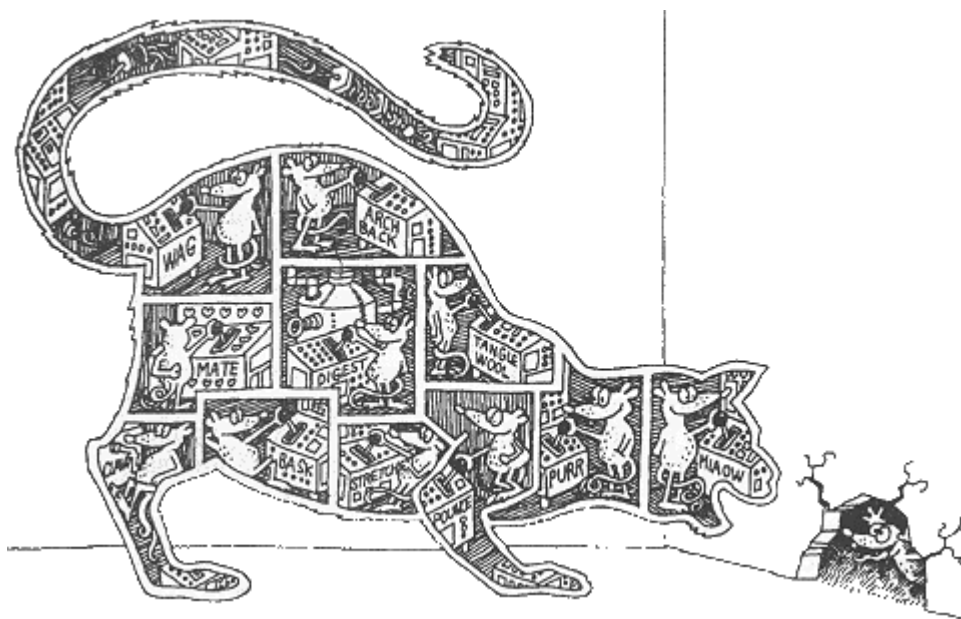
5 признаков сложных систем

2. "Выбор, какие компоненты в данной системе считаются элементарными, относительно произволен и в большой степени оставляется на усмотрение исследователя."



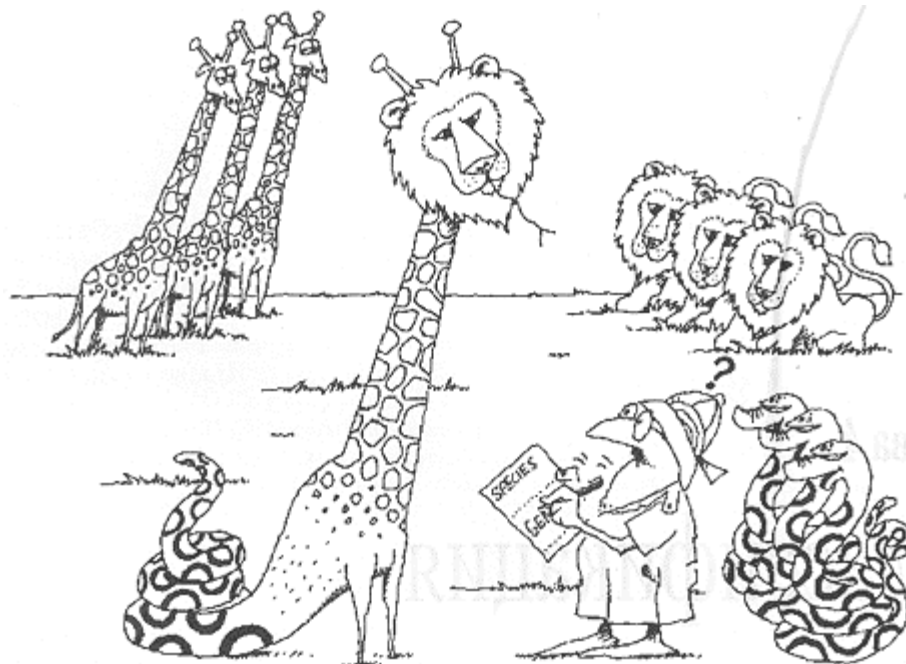
5 признаков сложных систем

3. "Внутрикомпонентная связь обычно сильнее, чем связь между компонентами. Это обстоятельство позволяет отделять "высокочастотные" взаимодействия внутри компонентов от "низкочастотной" динамики взаимодействия между компонентами".



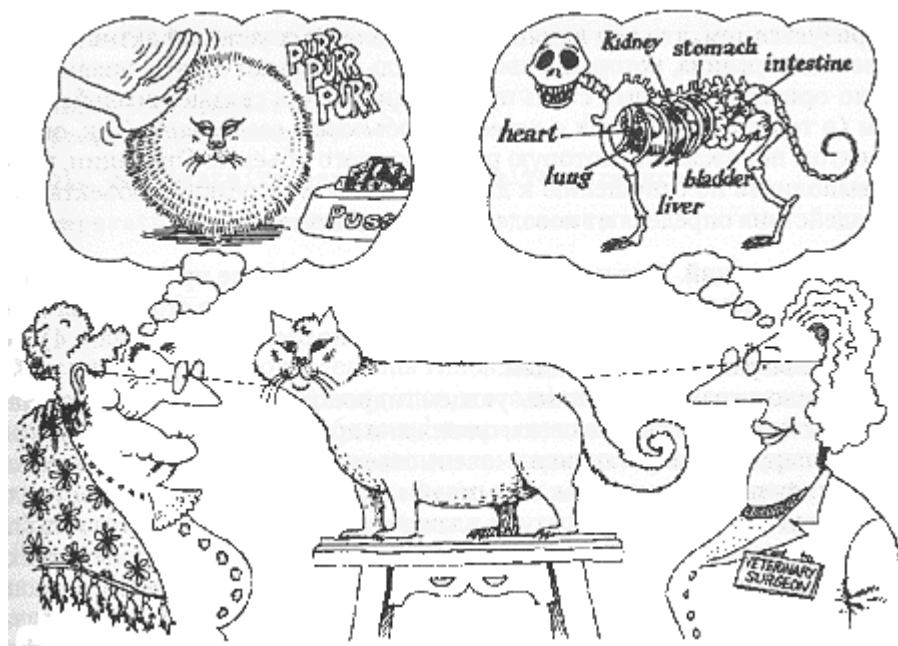
5 признаков сложных систем

4. "Иерархические системы обычно состоят из немногих типов подсистем, по-разному скомбинированных и организованных".



5 признаков сложных систем

5. "Любая работающая сложная система является результатом развития работавшей более простой системы... Сложная система, спроектированная "с нуля", никогда не заработает. Следует начинать с работающей простой системы".



Евгений Александрович Седов



Советский ученый, инженер-практик, изобретатель, педагог, популяризатор науки

Разработка и внедрение систем в промышленности и «военке»

Руководил отделом из **11 лабораторий** в течение 10 лет

Седов: междисциплинарные исследования

Более 200 публикаций: научных и научно-художественных!

кибернетика, теория информации, самоорганизация, стандартизация, искусственный интеллект

Ключевая тема: проблема разнообразия



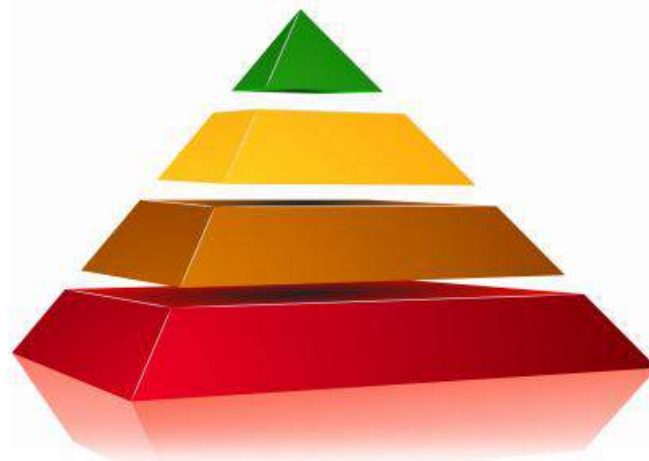
Закон Седова, 1988

1. Действительный рост разнообразия на высшем уровне обеспечивается его эффективным ограничением на предыдущих уровнях. Закон иерархической компенсации (закон Седова), охватывающий живую и неживую природу, язык, культуру, все сферы социального управления, существенно дополняет классический закон необходимого разнообразия Эшби.



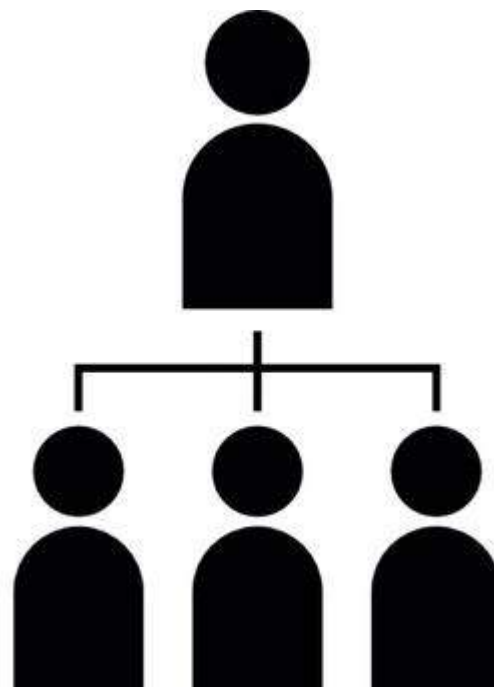
Закон Седова

2. Только при условии **ОГРАНИЧЕНИЯ РАЗНООБРАЗИЯ** нижележащего уровня можно формировать разнообразные функции и структуры находящихся на более высоких уровнях систем, таким образом, возникает проблема поисков оптимального соотношения детерминации и непредсказуемости составных частей и системы в целом.

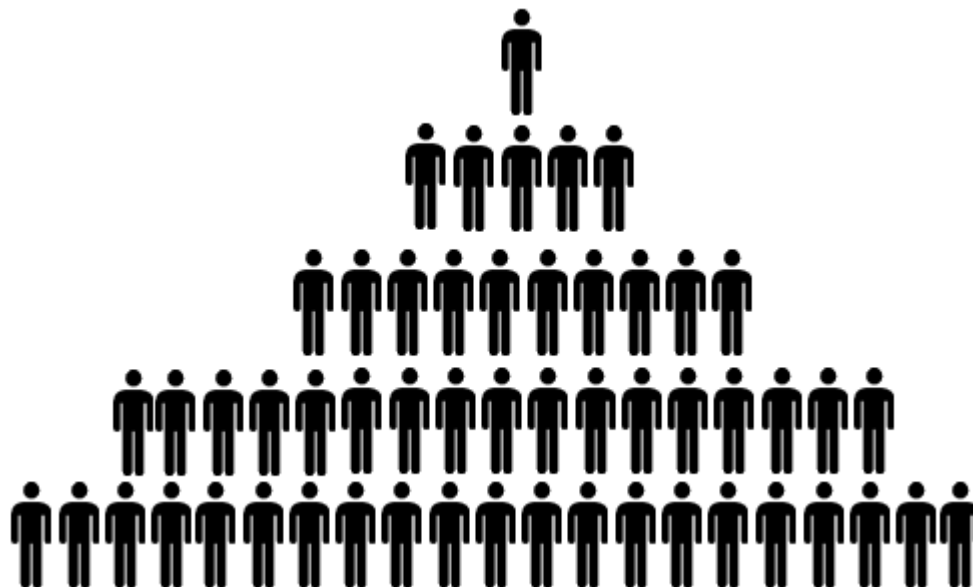


Закон Седова

3. В структурном смысле закон означает, что «отсутствие ограничений приводит к деструктуризации системы как целого, что приводит к общей диверсификации системы в контексте объемлющей её среды.



4. Существует взаимосвязь второго начала термодинамики с негэнтропийным принципом информации, установленным Бриллюэном: накопление информации (отрицательной энтропии) внутри какой-либо системы всегда оплачивается возрастанием энтропии внешней среды.



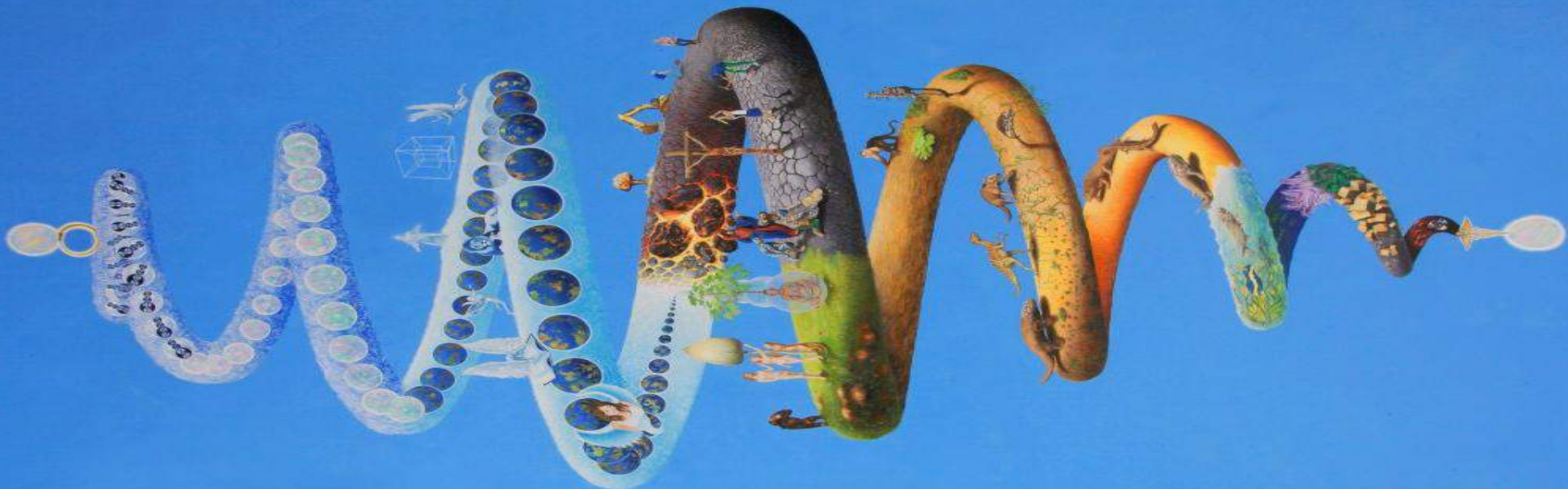
Формулировка Назаретяна

1. В сложной иерархически организованной системе рост разнообразия на верхнем уровне обеспечивается ограничением разнообразия на предыдущих уровнях, и наоборот, рост разнообразия на нижнем уровне разрушает верхний уровень организации, то есть, система как таковая гибнет.

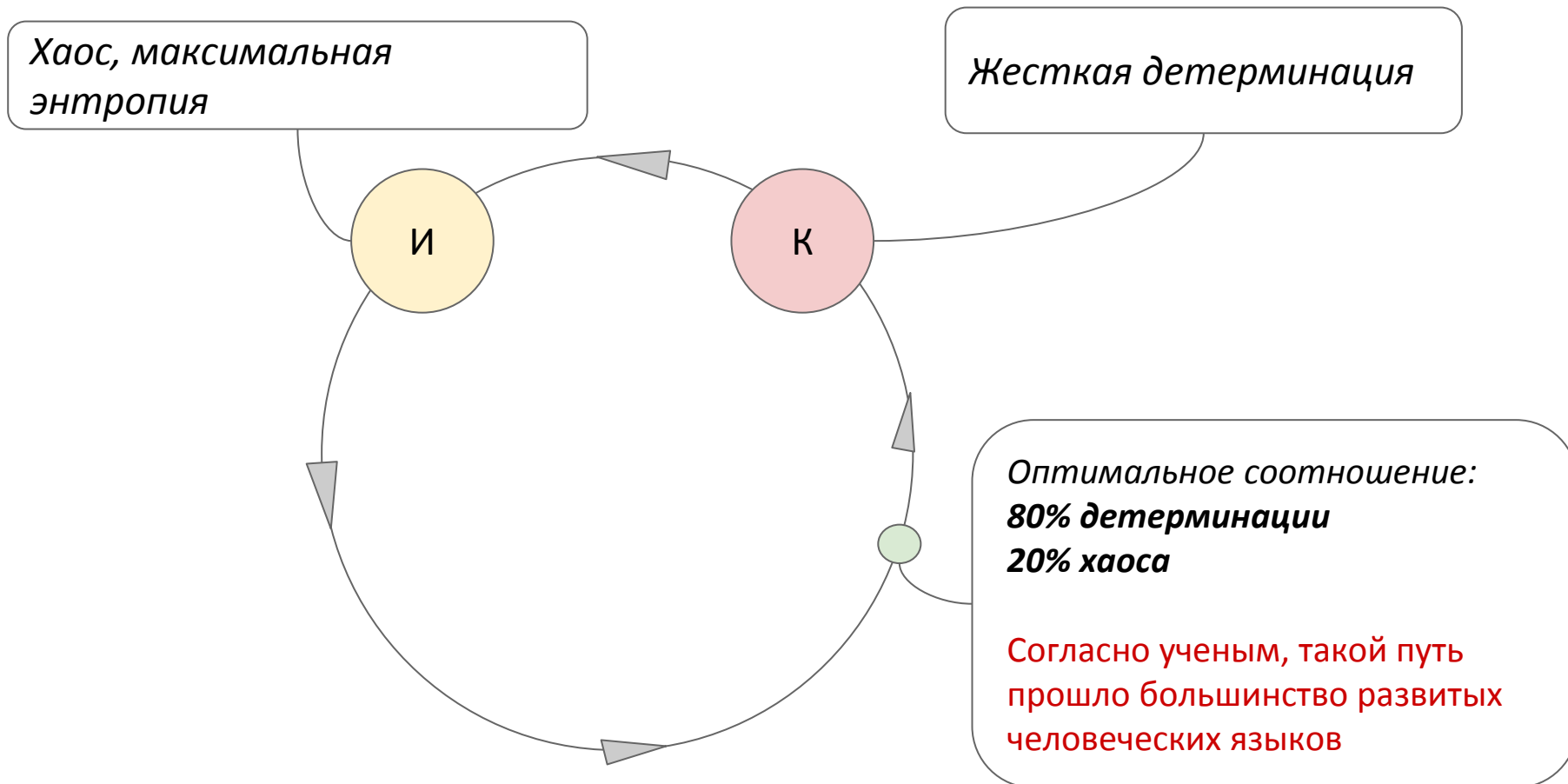


Сокращается ли внутреннее разнообразие систем в процессе эволюции?

Живая и неживая природа, язык, культура, технологии



От хаоса к детерминированности



Путь от И к К - накопление структурной информации

Магическое соотношение 80/20

80% детерминированности: языковая структура
20% хаоса: вариабельность, “мутации”, “новости”, ради которых и пишется текст

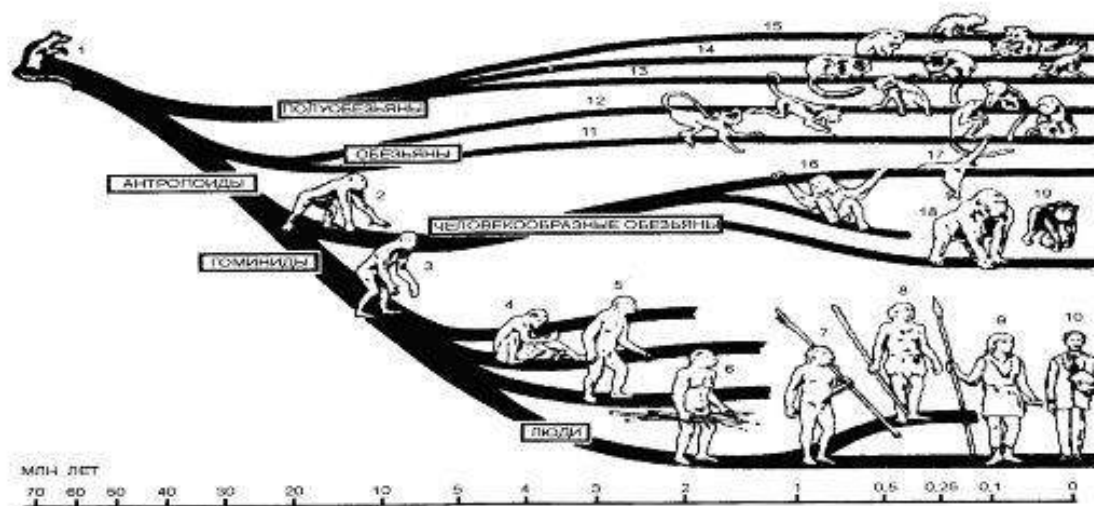
При увеличении детерминированности теряется адаптивность, и система разрушится при изменении внешних условий

Единственный выход: разрушение, скачок от К к И и создание новой системы



Новые уровни иерархии драматически увеличивают число новых связей между элементами системы

Связи = энергия, и единственный способ сохранить систему - это ограничить число элементов



Разнообразие на верхних уровнях иерархии
может быть обеспечено только за счет
ограничения разнообразия на нижних
уровнях

Сложные системы можно строить только
из ограниченного числа простых

или

Стандартизация неизбежна!

Многообразие на верхнем уровне иерархии возможно только при строжайшем ограничении сущностей и стандартизации на всех ниже лежащих уровнях иерархии и наоборот, как только на $(n-1)$ уровне иерархии наблюдается многообразие сущностей, n -й уровень иерархии разваливается, перестает существовать, $(n-1)$ уровень становится верхним уровнем иерархической системы

«Исторические примеры»



60-е и виртуализация

```

VIRTUAL MACHINE/SYSTEM PRODUCT

      NN      NN      AA      XXX  XXX
      NNNN  NNNN      AAAA      XXX  XXX
      NN NNNN NN      AA  AA      XXXXXX
      NN  NN  NN      AA  AA      XXXX
      NN      NN      AAAAAAAAAA      XXXXXX
      NN      NN      AA          AA      XXX  XXX
      NN      NN      AA          AA      XXX  XXX

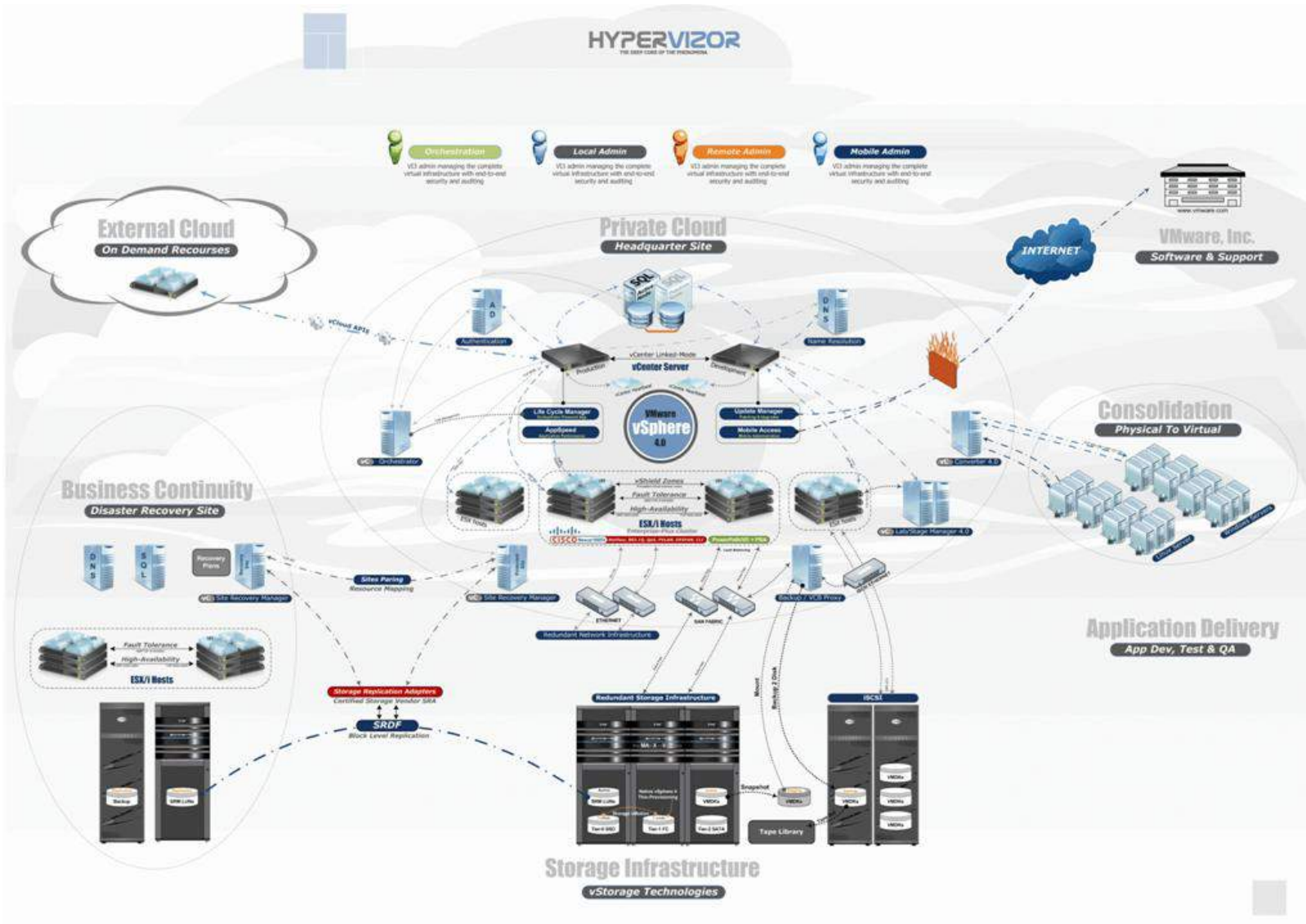
To access MURLIN enter D VSESP in the command
field below.

Fill in your USERID and PASSWORD and press ENTER
(Your password will not appear when you type it)

USERID  [XXXXXXXXXX]
PASSWORD [XXXXXXXXXX]

COMMAND [XXXXXXXXXX]
  
```

Виртуализация сегодня

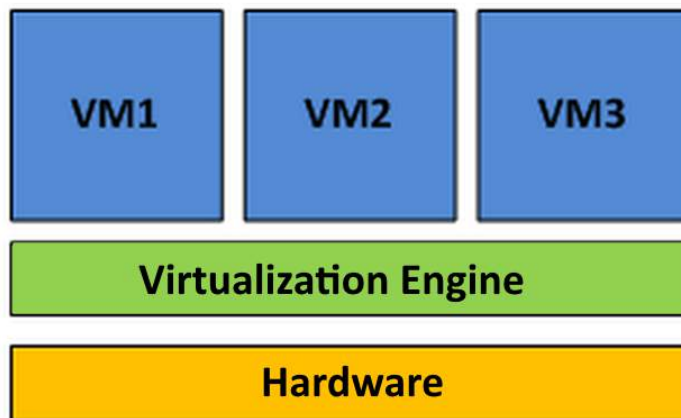


Зачем разбираться в виртуализации

- горячий тренд индустрии
- элегантное решение многих проблем, особенно в автоматизации
- междисциплинарная область для разработки, тестирования, DevOps
- работодатели любят ее на собеседованиях :)



Виртуализация – это слой абстракции между операционкой и “железом”



Виртуализация ≠ Облака

Зачем виртуализировать?

- Эффективнее утилизация ресурсов
- Меньше энергозатраты
- Легче управлять виртуалками чем железными серверами
- Проще управление миграцией и непредвиденными ситуациями



Конец 1960-х – первый факт применения виртуализации (в операционке CP/CMS)



1972 – начал использоваться термин *гипервизор*, подчеркивая отличие от термина *супервизор*



Конец 1990х – первые широко применяемые виртуализационные продукты на архитектуре x86 (Virtual PC, VMWare Workstation)



Начало 2000х – появление аппаратной виртуализации на x64 (VMWare ESX, Xen, Virtual Server)



Конец 2000х – появление первых облачных платформ (Amazon, Azure, Google)



Типы виртуализации

Bare Metal

Аппаратная

Type 1

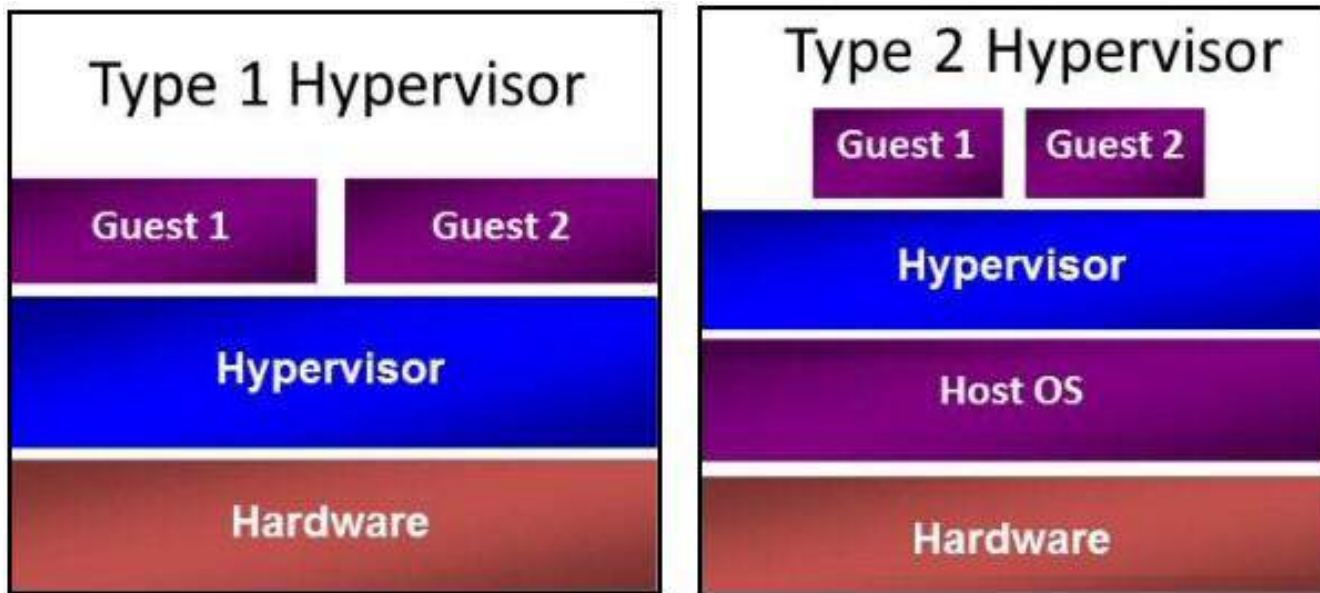
Hosted

Эмуляция

Type 2



Типы виртуализации



... а также гибридные варианты

Type 1/Bare Metal

- **Промышленная виртуализация**
- **Аппаратный уровень** (поддержка от производителей процессоров - Intel VT и AMD-V)
- **Минимальный overhead**
- **Основа для облаков**



Type 2/Hosted

- Эмуляция
- Для “ручного” использования
- Большой overhead



Physical to Virtual (P2V)

Создание виртуальной машины из физической

- Hyper-V: **Disk2VHD**
- VMWare: **vCenter Converter**

Существует также *Continuous P2V*

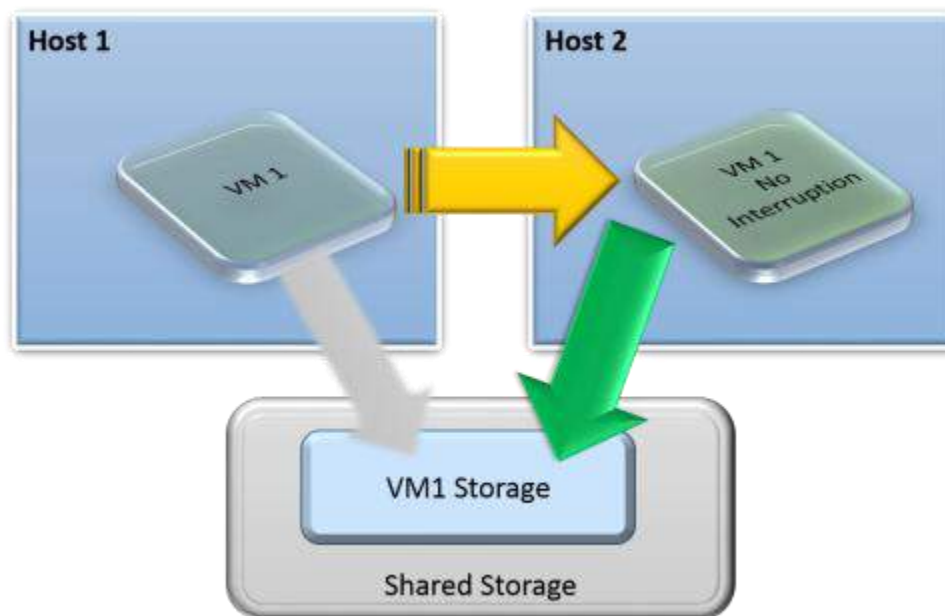
Создание “точек отката” для важных изменений



Идеально для автоматизации

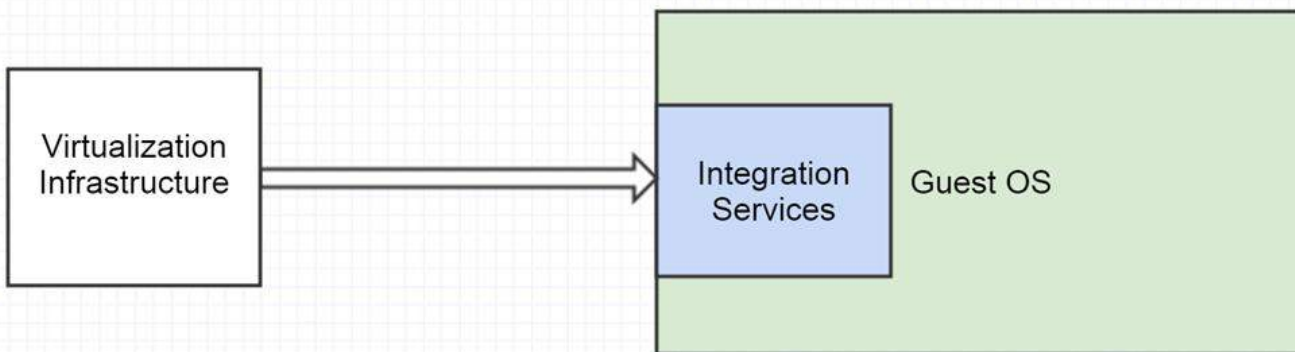
Live Migration (VMotion)

Миграция “живой” виртуалки с одного сервера на другой



Взаимодействие с гостевой операционкой извне

- обмен данными
- запуск команд втури
- ... и прочие вспомогательные вещи



Основа для автоматизации

- Выросла из Connectix (поглощена Microsoft в 2003)
- Работает на Windows Server и Windows 8 Pro
- Гибридная виртуализация (Type 1+)

** Как определить, что ПО запущено на Hyper-V? :)*

```
A problem has been detected and windows has been shut down to prevent damage
to your computer.

DRIVER_IRQL_NOT_LESS_OR_EQUAL

If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:

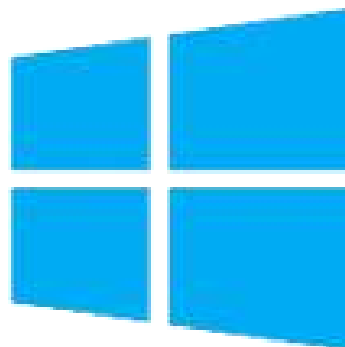
Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup options, and then
select Safe Mode.

Technical information:
*** STOP: 0x00000001 (0x0000000C,0x00000002,0x00000000,0xF86B5A89)

***      gv3.sys - Address F86B5A89 base at F86B5000, DateStamp 3dd991eb

Beginning dump of physical memory
Physical memory dump complete.
Contact your system administrator or technical support group for further
assistance.
```



Microsoft
Hyper-V

Player/Workstation – Type 2 виртуализация
ESX/ESXi – гипервизор (Type 1)
vSphere – платформа (“инфраструктура”)
vCloud – приватное облако построенное на vSphere
vCenter – инструмент управления vSphere

```
VMware ESX Server [Release build-52542]
Exception type is 10, error: 0x1 fsk:killv_PC_R 0x6c40d3
state=0xc frame=0x3497890 eip=0x6c3043 cr2=0x4059f000 cr3=0x41697000 cr4=0x160
eax=0x0ffff0 ebx=0x0c20 ebx=0x3ff4411 ebx=0x2e9 es=0x402b ds=0x349420
fs=0x3490000 ss=0x0 ebp=0x3497860 ipl=0x64e1058 edi=0x4099f000 err=2 efl=0x1020
cpu 0 1024 console: CPU 1 1061 mkt:Klik : cpu 2 1080 vmm:Virt: cpu 3 1065 vmm
0x3497860: [0x6c40d3]Pkt_CopyOutMappedInoCum+0x4f(0x40970048, 0x644b280, 0xffff
ff)
0x34978c8: [0x6c3ee9]Pkt_CopyOutAndCumDerIE+0x271(0x4097000, 0x6401210, 0x0)
0x3497910: [0x6c3e1d]E1000Interrupt+0x238(0x4461540, 0x349799c, 0x3497a00)
0x3497940: [0x6c3eb3]E1000PollRxRing+0x10d(0x4461540, 0x3497a00, 0x3497a98)
0x34979e0: [0x6c17f2]E1000RecvRx+0x5c(0x4461540, 0x0, 0x3497a00)
0x3497a30: [0x6d133a]IICbdrv_Rename+0x40(0x4461540, 0x4461540, 0x0)
0x3497a50: [0x6b8761]PortOutput+0x39(0x4461540, 0x0, 0x3497b10)
0x3497b54: [0x6b9c43]E1000SwitchPortDispatch+0x80(0x4461540, 0x3497bb0, 0x4d5b
1)
0x3497b70: [0x675a8b]Port_Input+0x27(0x4461540, 0x3497bb0, 0x0)
0x3497ba0: [0x62ced]NetIF+0x50(0x0, 0x34, 0x1)
0x3497c10: [0x60d41]BHCallHandler+0x78(0x40400010, 0x705184, 0x2000000)
0x3497c40: [0x60c10]BH_Exec+0x0(0x1, 0x0c02c, 0x1000000)
0x3497c58: [0x61c185]IIT_HandleInterrupt+0x77(0x3497c58, 0x0, 0x143c944)
0x3497c6c: [0x61c26]IIT_Handler+0x24(0x3497c6c, 0x0c4020, 0x3494020)
0x3497d0: [0x6a76]ConnonInt+0xc(0x1, 0x10f0060, 0x131c3)
0x3497d40: [0x70458]CpuChen_ID1LowIn+0x4(0x0, 0x0, 0x624)
0x3497d10: [0x70458]CpuChenID1+0x54(0x0, 0xc, 0x0)
0x3497e10: [0x6ea7c]DeuschelDispatch+0x20(0x0, 0x0, 0x339542d)
0x3497e40: [0x7022b]CpuChenID1+0x1a(0x1594b0, 0x339542d, 0x4f)
0x3497e70: [0x7022b]CpuChenID1+0x39(0x339542d, 0x1f, 0x0)
VM uptime: 123:50:48.159 UTC: 4939122430900
Starting core dump to disk using slot 1 of 1... 9876666543210 Disk dump success
MI
Debugger is listening on serial port ...
```



API для управления виртуалками

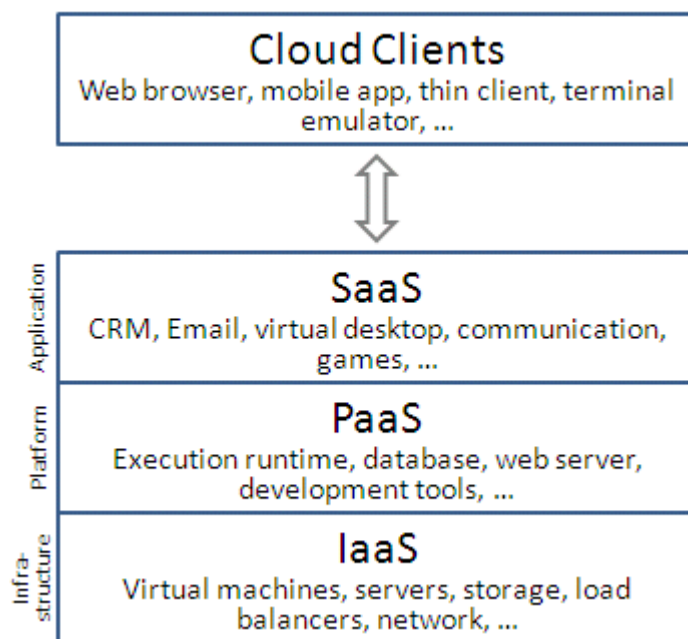


- WMI
- PowerShell



- Command-line
- PowerShell (*PowerCLI*)
- SOAP

Основаны на *bare-metal* виртуализации



Azure в числах 2015

> 57%

Fortune 500 companies
deployed on Azure

1,200,000

SQL databases in Azure
> 10% growth MoM

> 30 Trillion

Storage objects
in Azure

350 Million

Azure Active Directory users

> 18 Billion

Azure Active Directory
authentications/week

> 1.65 Million

Developers registered with
Visual Studio Online

10% growth MoM

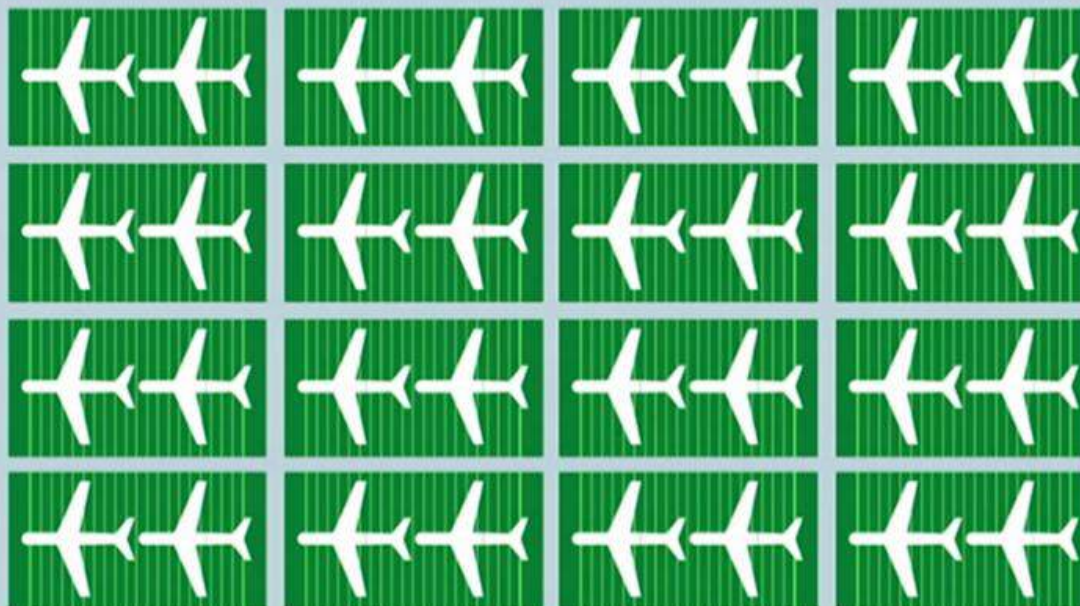
> 280%

Compute Growth YoY

Датацентры Azure 2015



Масштабы датацентра Azure



That's up to 600,000 servers in
each Azure region

Ложка дегтя в облаках

Апрель 2011 года, из-за аварии системы, несколько серверов вышли из строя. После многодневного простоя, 0,07 % данных пользователей были навсегда утеряны (Amazon EC2)

Azure Virtual Machines за **2014 год** – 42 часа даунтайма

Август 2015 года – потеря данных в датацентре Google Compute Engine из-за удара молнии



- Вся промышленная виртуализация всерьез платная, с очень витиеватым лицензированием
- Познакомиться с виртуализацией проще всего на бесплатных инструментах (например, VMWare Player)
- У Hyper-V и VMWare существуют образовательные программы

Виртуализация: выводы

- Диалектика Гегеля
- 5 признаков сложных систем по Гради Бучу
- Закон «Иерархических компенсаций» Седова

70-е и «бум» языков прог.

- Верхний уровень иерархии - непосредственно язык программирования, а не способы его использования, или ПО.
- Языков программирования в сотни раз больше >>
- ПО в тысяч раз меньше <<<
- Язык программирования == единицы ПО



80-е и сетевые протоколы

- В 80-е верхним слоем, демонстрирующим «техническое» многообразие, был слой сетевых протоколов, именно поэтому, у нас было так мало решений, успешно внедренных проектов, построенных на базе протоколов, но очеееень много вариантов самих протоколов.



80-е и сетевые протоколы

- Стандартизация уровня сетевых протоколов позволила запустить стандартные локальные сети, а затем интернет, со всем его многообразием решений.
- Миллионы единиц инет ПО возможны только благодаря крошечному набору сетевых протоколов, которые легко пересчитать по пальцам.



90-е годы и STL C++

- В начале 90-х язык C++ был тем самым верхним уровнем; многообразие реализовывалось не множеством отличных успешно внедренных проектов, не множеством концептуальных способов использования языка, таких как, мета программирование шаблонов или элементы функционального программирования, а в миллионе способов работы с языком непосредственно, например, тысячи, если не десятки тысяч реализаций одной и той же стандартной библиотеки STL.



2000-е годы и мобильные ОС

- 15 лет назад было многообразие ОС для смартфонов, и крайне ограниченное количество программ, десятки, сотни, для конкретной ОС ... Сегодня мы видим единицы ОС и миллионы новых программ.



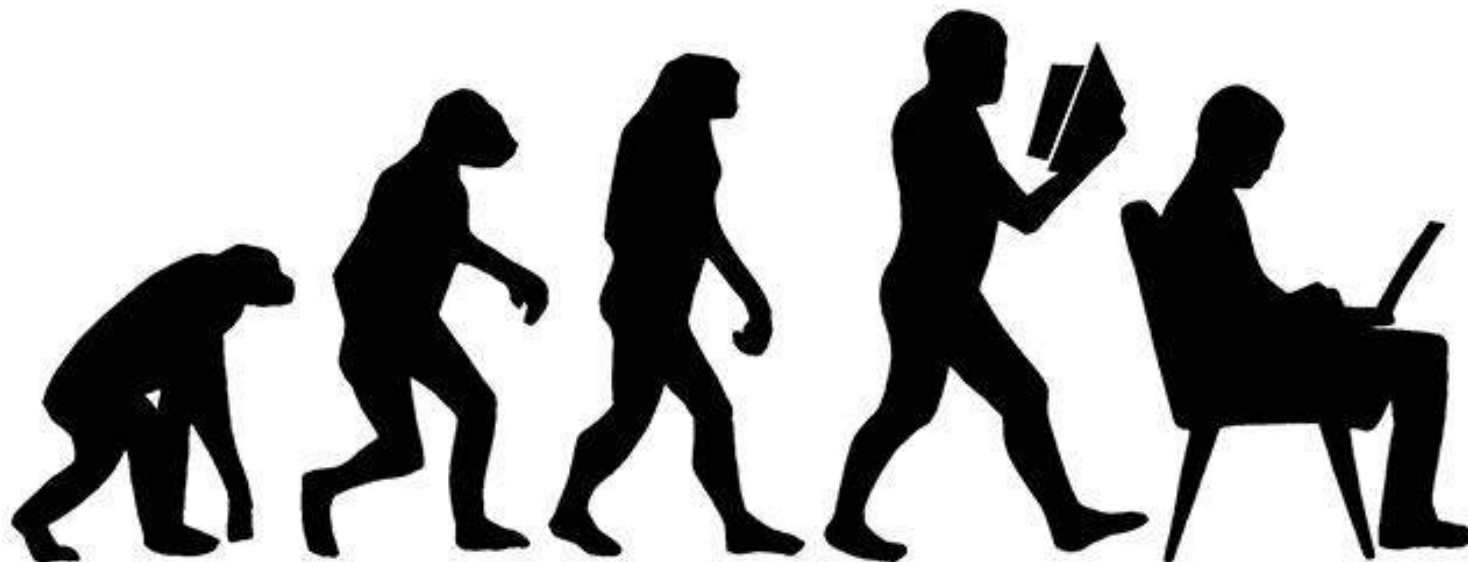
- Закон Седова «усиленный» диалектикой Гегеля и 5 признаками Сложных систем по Гради Бучу - лакмусовая бумажка, для определения вектора, фокуса усилий и наиболее перспективных решений.



Тренды Автоматизации



Эволюция Web Автоматизации



Бесплатные инструменты 2006 года

1. В 2006, «почти одновременно» появилось, причем в рамках одной организации OpenQA, 2 конкурирующих бесплатных инструмента Web Автоматизации:
 - **Selenium**
 - **WatiR**
2. А так же
 - **Sahi**
 - **Windmill**

Инструменты 2006 года



Selenium



Watir

WatiN

Watij



Selenium

- Неудобный API
- Низкоуровневый API
- Продукт разрабатывается энтузиастами ... без бизнес процессов, инфраструктуры
- Изначально, хорошая поддержка только одного браузера Firefox
- Record Play plug-in только для Firefox

Выводы: Selenium на первый поверхностный взгляд не выглядит «лидером»

Wati*

- Удобный API
- Умеренно высокоуровневый API
- Изначально поддерживал только IE
- Проект оказался настолько успешен на старте, что породил клоны для .Net и Java
- WatiR - изначальный вариант инструмента для Ruby
- WatiN - .Net clone, используется до сих пор
- WatiJ - Java clone, deprecated

Выводы: WatiR на первый поверхностный взгляд гораздо ближе к понятию «лидер» индустрии, чем Selenium

Sahi

- Удобный API
- Поддерживает основные популярные браузеры
- Существует коммерческая поддержка
- «Профессиональная», не Open Source разработка

Windmill

- Удобный API

Выводы: Sahi и даже Windmill на первый поверхностный взгляд гораздо ближе к понятию «лидер» индустрии, чем Selenium

Почему Selenium стал лидером?

Все, о чем мы будем поговорить дальше, похоже на детали реализации ... Но именно эти «детали» и позволили Selenium-у стать лидером индустрии, оказались принципиальным конкурентным преимуществом!



Почему Selenium стал лидером?

Удаленный интерфейс

- Клиент-серверная Архитектура
- Работа с браузером по сети

Как следствие

- Распределенная архитектура - браузеры на др. машине, эффективное построение гетерогенного окружения
- Мультиязычность = клиентский протокол + клиентские библиотеки на разных языках программирования



Почему Selenium стал лидером?

Стандарт W3C

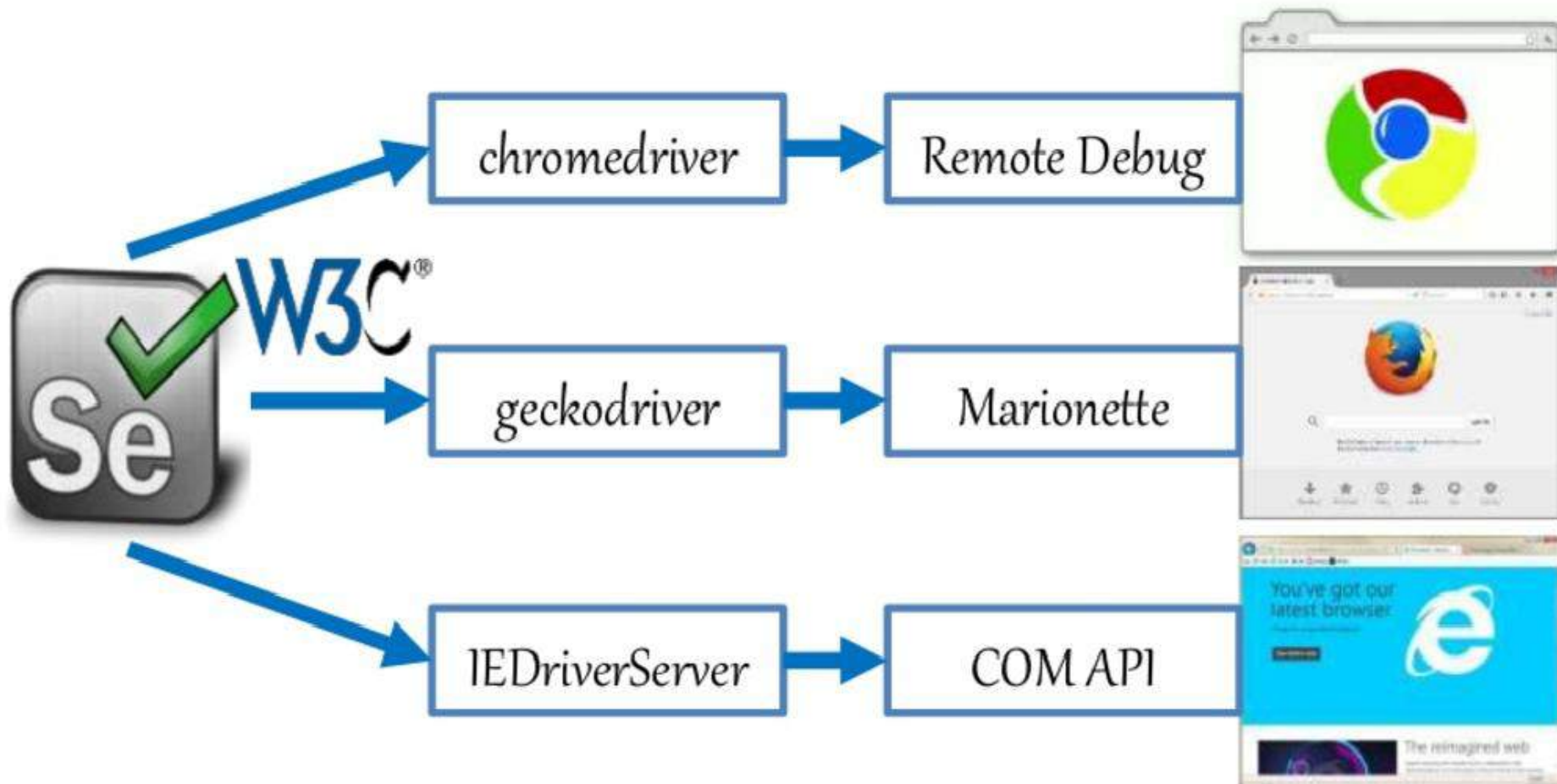
Note: Разработка стандарта ведется с 2011 и формально еще не завершена

Разработка driver-ов легла на плечи разработчиков браузеров, стала внешней, по отношению к проекту Selenium

- Как ни кто, знают детали API браузера
- Могут использовать не документированные интерфейсы браузера
- Могут обновить \ исправить браузер



Эволюция Web Автоматизации



Почему Selenium стал лидером?

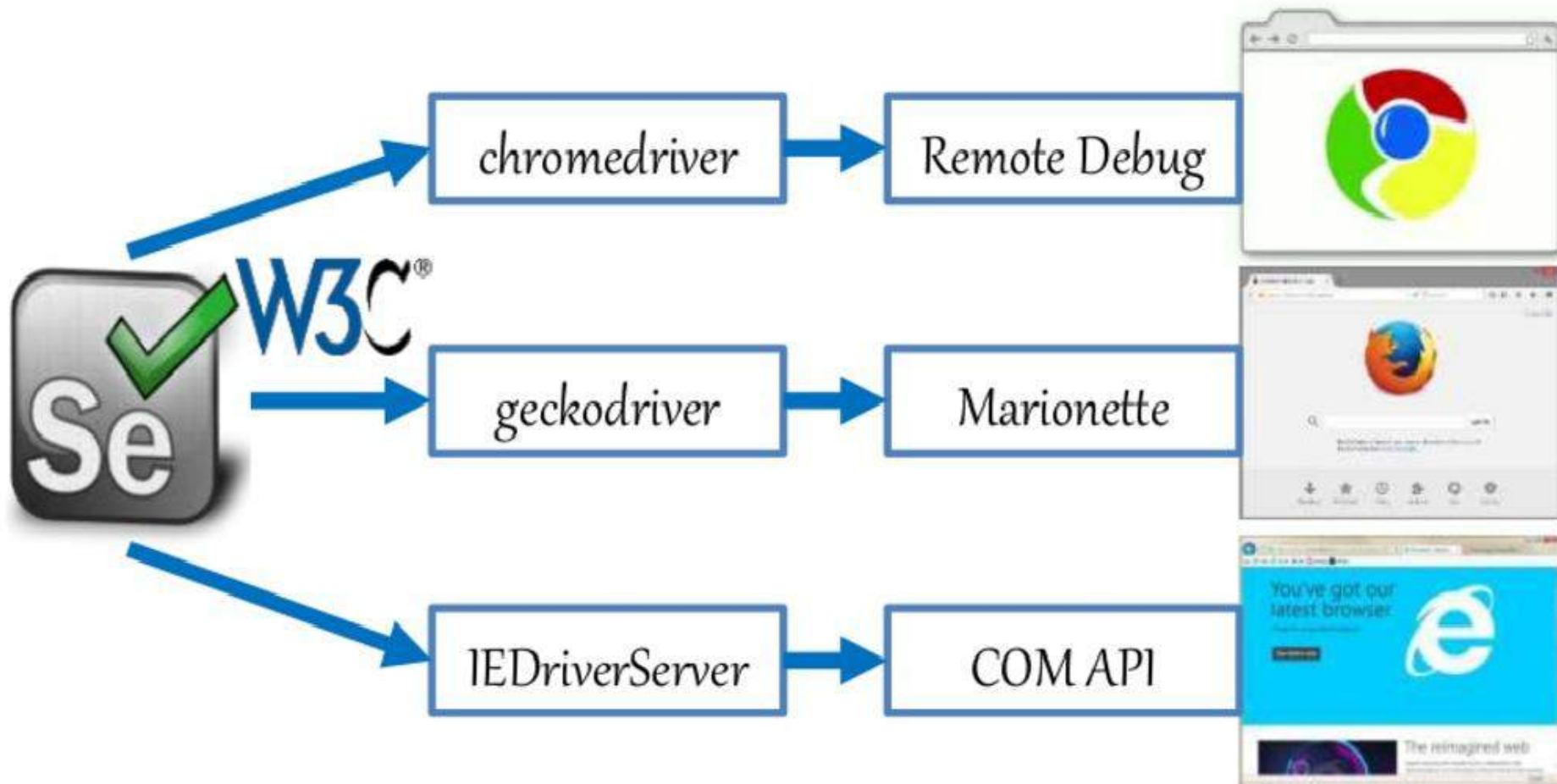
Selenium проект непосредственно состоит из стандартного очень простого легко расширяемого низкоуровневого протокола и набора binding-ов, под каждый из официально поддерживаемых языков

Плюсы:

- Только клиентские библиотеки
- Универсальные
- Простые



Эволюция Web Автоматизации



Почему Selenium стал лидером?

Унификация «интерфейсов интеграции»

Notes:

- 1. Изначально, все «составные части» Selenium-а разрабатывались внутри проекта*
- 2. После стандартизации - driver-ы разрабатываются вне*
- 3. Таким образом, Selenium из инструмента эволюционировал в платформу*



Почему Selenium стал лидером?

Унификация «интерфейсов интеграции»

Note:

1. *Binding-и - клиентские библиотеки*
2. *Driver-ы браузеров*
3. *Языковая независимость*
4. *Очень простой API W3S который можно реализовать на любом языке программирования: функциональном, объектно-ориентированном, процедурном*
5. *Заложена возможность расширения, т.е. Адаптивность*

Почему Selenium стал лидером?

Bindings

Notes:

1. *Официальные (часть проекта Selenium, релиз вместе с новой версией Selenium-а, «гарантии» качества, благодаря UnitTest-ам и прочей инфраструктуре): Java, C#, Python, Ruby, JS - нет конкуренции*
2. *Энтузиасты: JS, PHP, Perl - есть конкуренция*
3. *Экзотика: Go, Haskell, Objective-C, R, VBS*



Почему Selenium стал лидером?

Drivers

Notes:

- 1. На вход со стороны клиента \ теста - стандартный W3C протокол*
- 2. На выход со стороны браузера - специфический для браузера протокол*



Почему Selenium стал лидером?

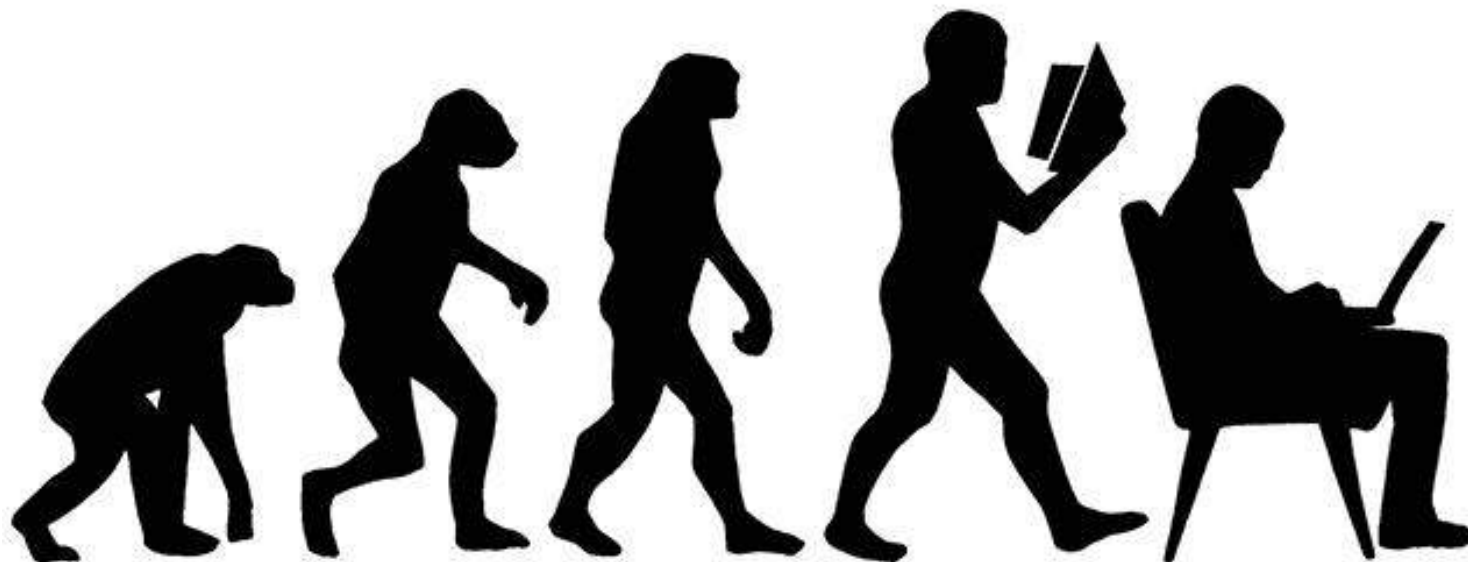
Browsers

Notes:

1. *Настоящие браузеры - нет конкуренции*
2. *Псевдобраузеры: GhostDriver, QTWebDriver, jBrowserDriver - нет конкуренции*
3. *Mobile - Appium, Selendroid, iOSDriver - есть конкуренция*
4. *Desktop - WiniumDesktop, AutoltDriverServer*



Эволюция Mobile Автоматизации



Сравнительный анализ:

- 3 условно эталонных проекта: native, web, hybrid
- 3 условных размера \ сложности проекта
- 20 инструментов Автоматизации

Summary: нет явного лидера, но **Appium** 😊

Note: часть проделанной работы со временем превратилась в доклады, мастер классы, некоторые материалы можно найти на сайте сообщества COMAQA.by



Mobile Automation 2013

В 2013 году не было явного лидера, не было, по-настоящему, стабильного и универсального инструмента, не было даже четкого однозначного понимания как правильно строить решения по Автоматизации тестирования мобильного ПО.

Но уже тогда, мы обратили пристальное внимание на «юный» инструмент Appium, который сегодня стал стандартом де-факто.



Почему Appium?

1. Требуется минимум времени для переобучения, при условии опыта в использовании Selenium-а
2. Можно использовать с любыми стандартными инструментами совместимыми с Selenium, т.е. Appium автоматически интегрируется в mature экосистему
3. Можно использовать с любыми совместимыми с Selenium custom-ными решениями, наработками заказчика и компании субподрядчика
4. Безшовно интегрируется с решениями по одновременной Автоматизации Web и Mobile
5. «Подталкивает» к использованию стандартных решений, а не изобретений собственных «велосипедов»



Почему Appium стал лидером?

Совместим с Selenium WebDriver, т.е. инструментом, соответствующим закону Седова, инструментом - стандартом в одной из областей Автоматизации тестирования.

Notes:

- 1. Многообразие на уровне инструментов мобильной Автоматизации не дает сформироваться следующему уровню иерархии - многообразию вариантов применения того или иного инструмента, многообразию ПО, которое можно тестировать Автоматизированно.*



Почему Арриум стал лидером?

Notes:

- 2. Значит со-временем, обязательно появится лидер или очень небольшой набор лидеров в силу принципа «избыточного многообразия», который станет стандартом, и будет базироваться на «ниже» лежащих стандартах, максимально адаптивный с архитектурной точки зрения, способный к использованию в измененном или ином контексте, универсальный.*
- 3. Концептуально и Архитектурно лишь Арриум соответствовал эволюционному вектору согласно Седову, именно поэтому, мы выбрали этот инструмент в качестве основного.*



Почему Arrium стал лидером?

Notes:

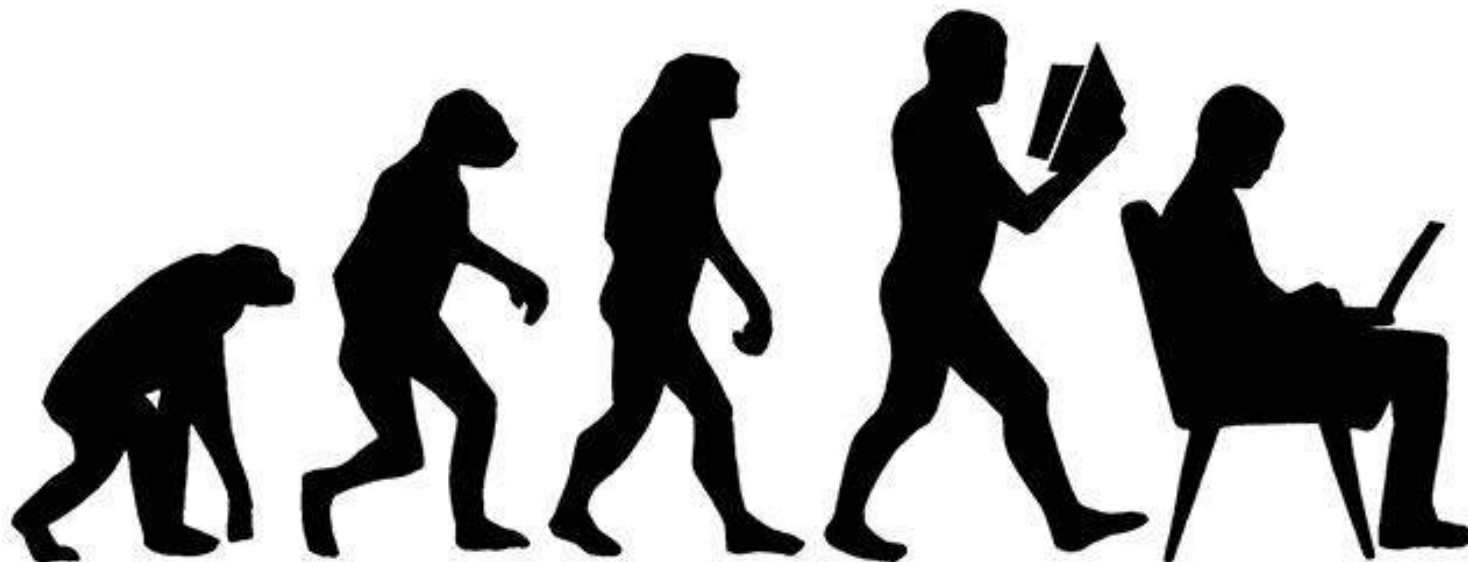
- 4. В 2016 году, я планировал подать на EuroStar доклад «Сравнительный анализ инструментов Мобильной Автоматизации», но орг. комитет, как и EPAM Testing Competency Center посчитал тему устаревшей и не перспективной, считая Arrium грядущим стандартом де-юре, т.е. сознательно или несознательно, явно или косвенно, но был использован закон Иерархических компенсаций Седова для принятия решения по докладу.*

LOVE SOFTWARE
TESTING?
YOU BELONG AT
EUROSTAR 2016!



© EuroSTAR International
Business Testing
CONFERENCE 2016

Эволюция Desktop Автоматизации



Note:

1. *В 2016 году, я планировал подать на EuroStar доклад «Сравнительный анализ инструментов Мобильной Автоматизации», причины отказа по Седову вы знаете ...*
2. *А вот доклад «Сравнительный анализ инструментов Desktop-ной Автоматизации» был одобрен орг комитетом плюс EPAM TSS и благосклонно принят публикой.*
3. *Еще год назад тема была более чем актуальна, в этом году, она носит исключительно прикладной, тактический, а не стратегический характер, в следующем году, ее можно будет смело отложить в чулан, с пометкой от TSS «устаревшая и не перспективная».*

Почему Appium стал лидером?

Appium так же поддерживает Автоматизацию тестирования Desktop-ных приложений. Решение строится на базе так называемого Windows Driver-а. **Microsoft около 2-х лет работает над созданием универсального драйвера для Selenium like Автоматизации любых Desktop-ных приложений.** Учитывая факт сближения компании Microsoft с Linux и Open Source, заявлено, что со временем, Windows Driver так же будет поддерживать не только Windows, но и Linux \ Unix OS.



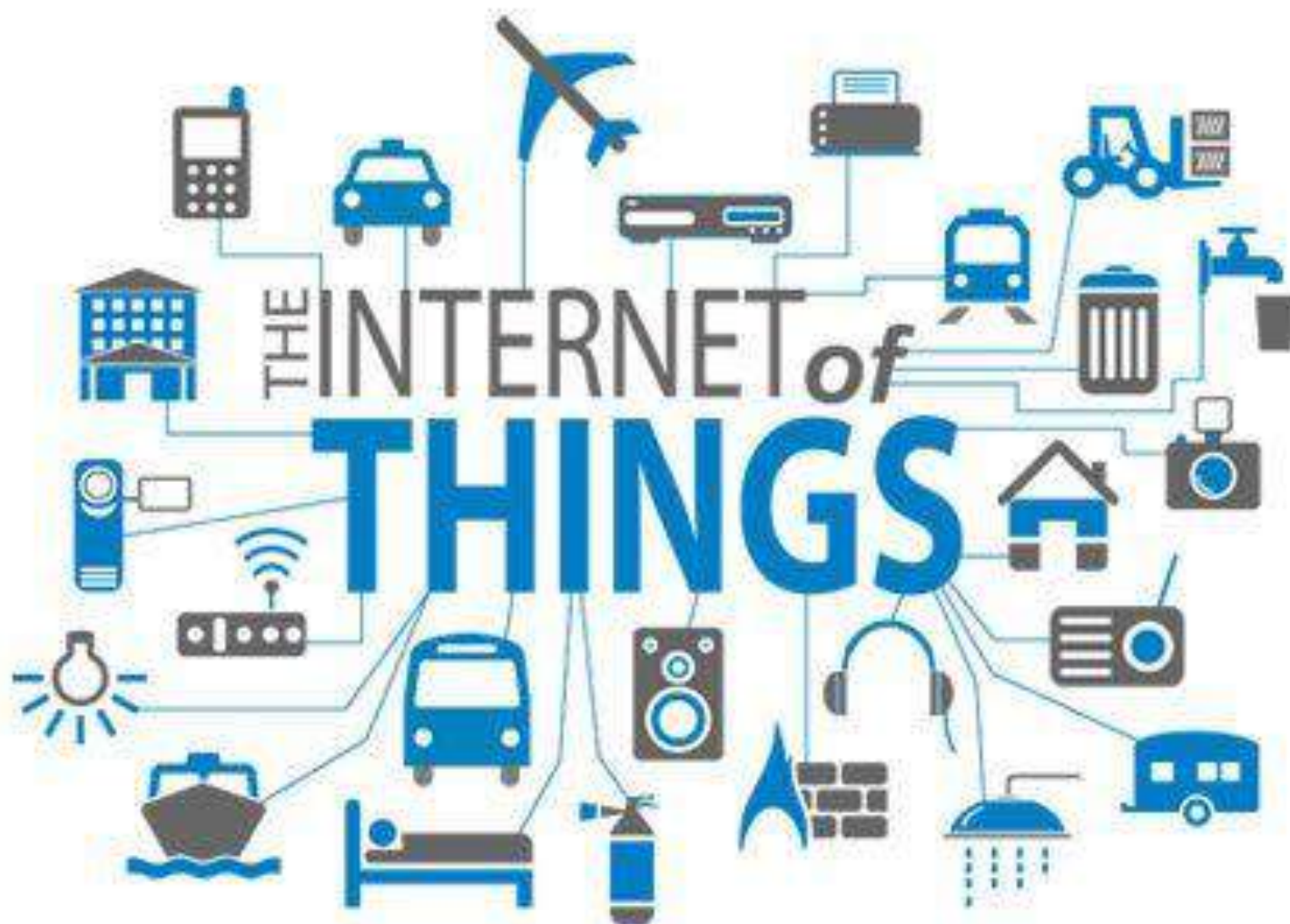
Почему Appium стал лидером?

Я подробно проговаривал планы развития Appium-а с его автором Dan Cuellar, особенно направление Desktop-ной Автоматизации, на конференции CodeFest в Новосибирске и Heisenbug в Москве.

Appium и правда строит планы о «мировом господстве», в соответствии с законом Седова.



Богатство IoT платформ



Богатство IoT платформ

В 2015 - 2016 годах на конференциях только и говорили, что про IoT, но как много 100% IoT решений мы используем сегодня ..? В рамках работы над докладами для ряда конференций в 2016, я сверх поверхностно изучил более 50! платформ для создания IoT решений, уже сегодня более половины из них «мертва». На данном этапе верхний уровень - это уровень платформ для IoT, а не IoT решений, поэтому мы видим сотни докладов, десятки специализированных конференций, еженедельно зарождающиеся и умирающие платформы, но почти не видим ПО.



Богатство IoT платформ

Через несколько лет ситуация координально изменится, останутся считанные платформы, соответствующие закону Седова, и десятки, возможно сотни тысяч повседневных IoT решений. Все те специалисты, кто инвестировал свое время в глубокое изучение IoT платформ в 2015 и 2016 годах потратили свое время зря, или почти зря. Все те компании, ну почти все, которые инвестировали свои деньги в серьезную разработку IoT платформ построили замечательные «замки из песка», но не получили прибыли, по крайней мере здесь и сейчас.



Богатство IoT платформ

Оптимальную стратегия инвестиций в IoT инновации в соответствии с законом Седова.

В «начале»: прототипирование, ознакомление, но не серьезные инвестиции на этапе, когда IoT платформа является верхним уровнем технической иерархии.

В «конце»: выделить несколько потенциальных лидеров, среди IoT платформ, и планомерно строить многочисленное ПО, сделав «ставку» на несколько решений, одно из которых, в будущем станет стандартом.



Богатство IoT платформ

Что касается «середины»: мастерство специалиста в том, что бы увидеть первое решение, соответствующее закону Седова и начать использовать его, таким образом, можно занять перспективную экологическую нишу; мастерство компании в том, что бы увидеть, когда многочисленные решения начинают сходиться в одну точку, выделить общее, и начать строить свою IoT платформу, которая, как минимум, гарантированно не противоречит закону Седова.

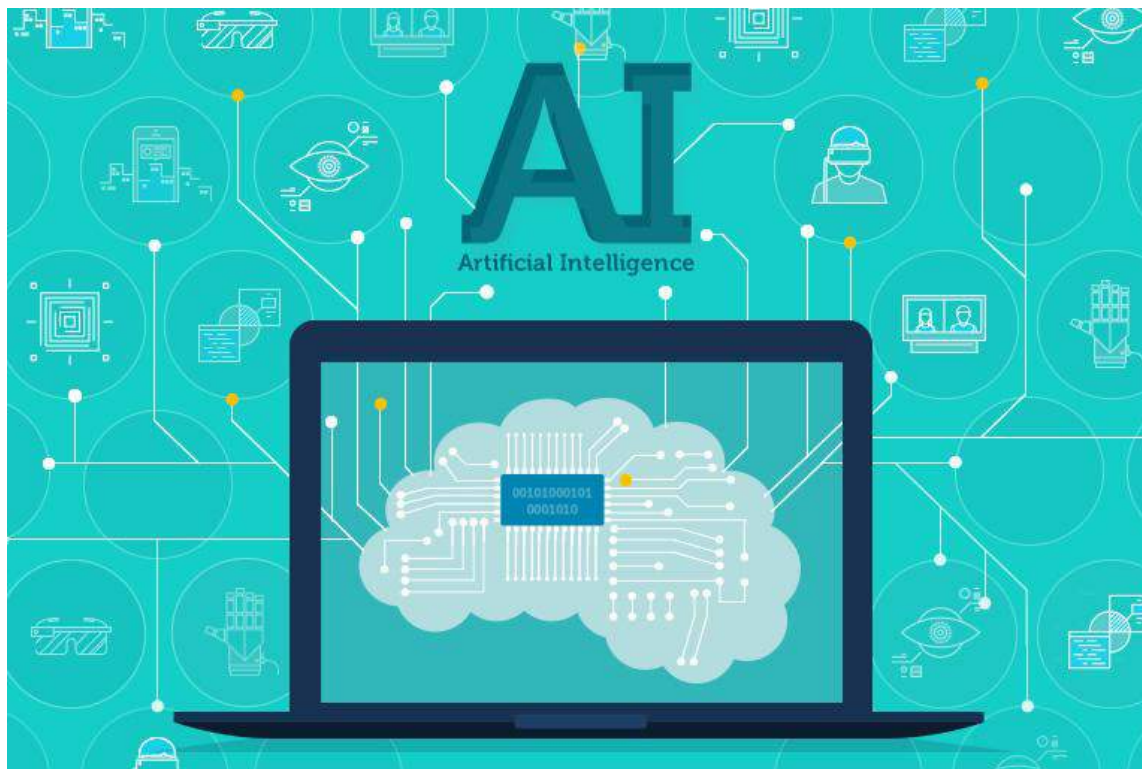


Богатство AI платформ



Богатство AI платформ

Все наши рассуждения про IoT платформы и решения на их базе, а так же оптимальное инвестирование, можно распространить на AI engines. Ситуация ну совершенно зеркальная.



1. Эволюция подчинена не только **закону Седова**, но и **диалектике Гегеля**. **Kotlin – перезапуск языка, новый диалектический виток по Гегелю**. В чем глубинная причина популярности Kotlin? На мой взгляд – **концептуальная целостность, минимализм, универсальность, расширяемость, обратная совместимость с Java, т.е. опора на проверенный годами стандарт**. Звучит как перезапуск верхнего уровня иерархии в плане языка программирования, в соответствии с законом Седова и диалектикой Гегеля. Таким образом, Kotlin – одна из перспектив Автоматизации, язык, на который стоит обратить внимание, так как он **убирает излишнюю enterprise сложность Java, помогает нам принимать правильные для Автоматизации, простые решения**.



Kotlin

Kotlin – о чем пойдет речь

1. Почему мы обратили внимание на Kotlin
2. Где мы его попробовали
3. Что из этого получилось
4. Какие выводы были сделаны



Kotlin

Сложность проектов растёт

```
List<String> userNames = new  
ArrayList<>();  
final String FORMAT = "User name: %s";  
userNames.add("Tom");  
userNames.add("Tim");  
userNames.add("Sam");  
for (String user: userNames){  
    String message =  
    String.format(FORMAT, user)  
    System.out.println(message);  
}
```

```
try {  
    ...  
} catch (Exception e){  
    ...  
} finally {  
    ...  
}
```

```
public class PhoneData {  
    private String name;  
    private String surname;  
    private byte age;  
  
    public String getName() {  
        return name;  
    }  
    ...  
    @Override  
    public int hashCode() {  
        ...  
    }  
}
```

Решения уровня архитектуры



1946 stars at GitHub

**458 questions at
GitHub**

**7 publications at
HabraHabr**



481 stars at GitHub

**47 questions at
StackOverflow**

**16 publications at
HabraHabr**

Решения уровня языка



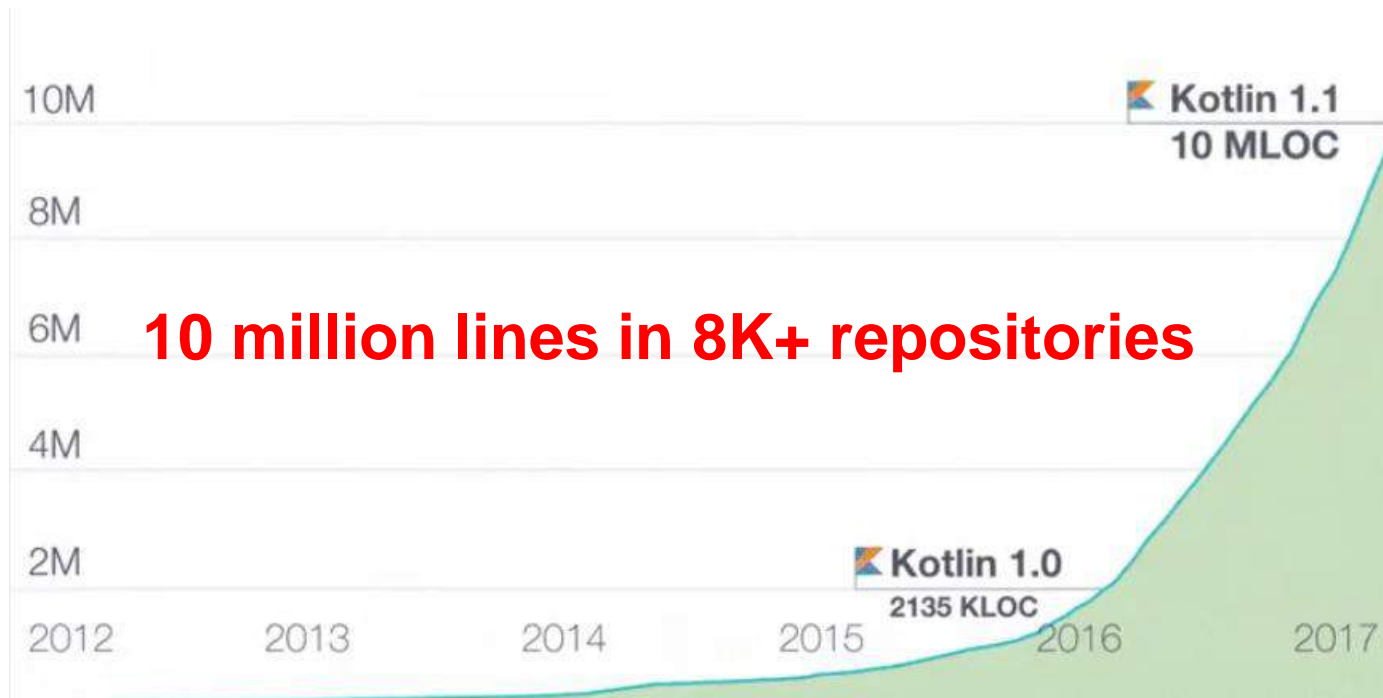
С чем едят Kotlin



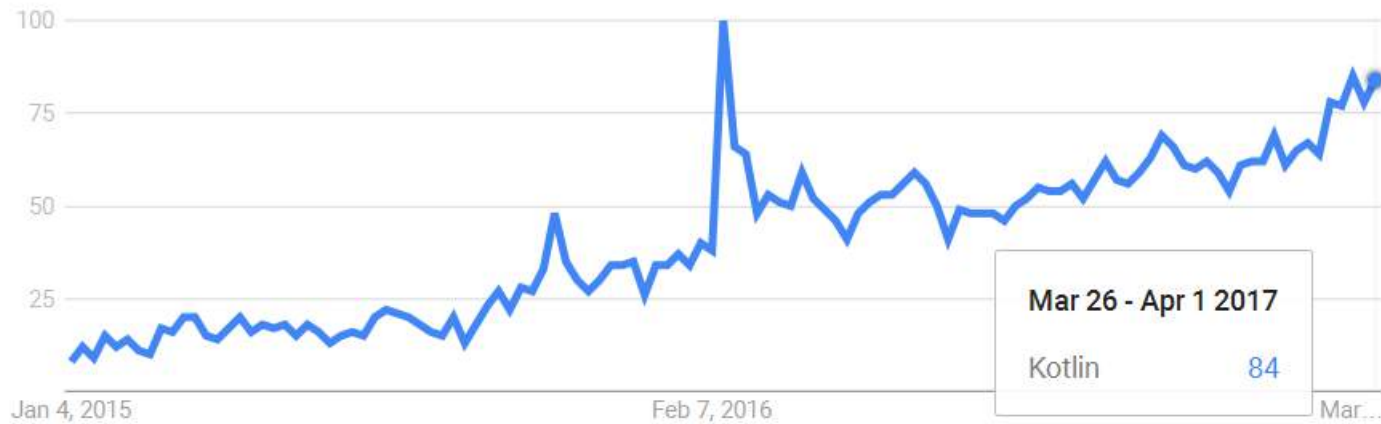
Statically typed programming language
for the **JVM**, Android and the browser



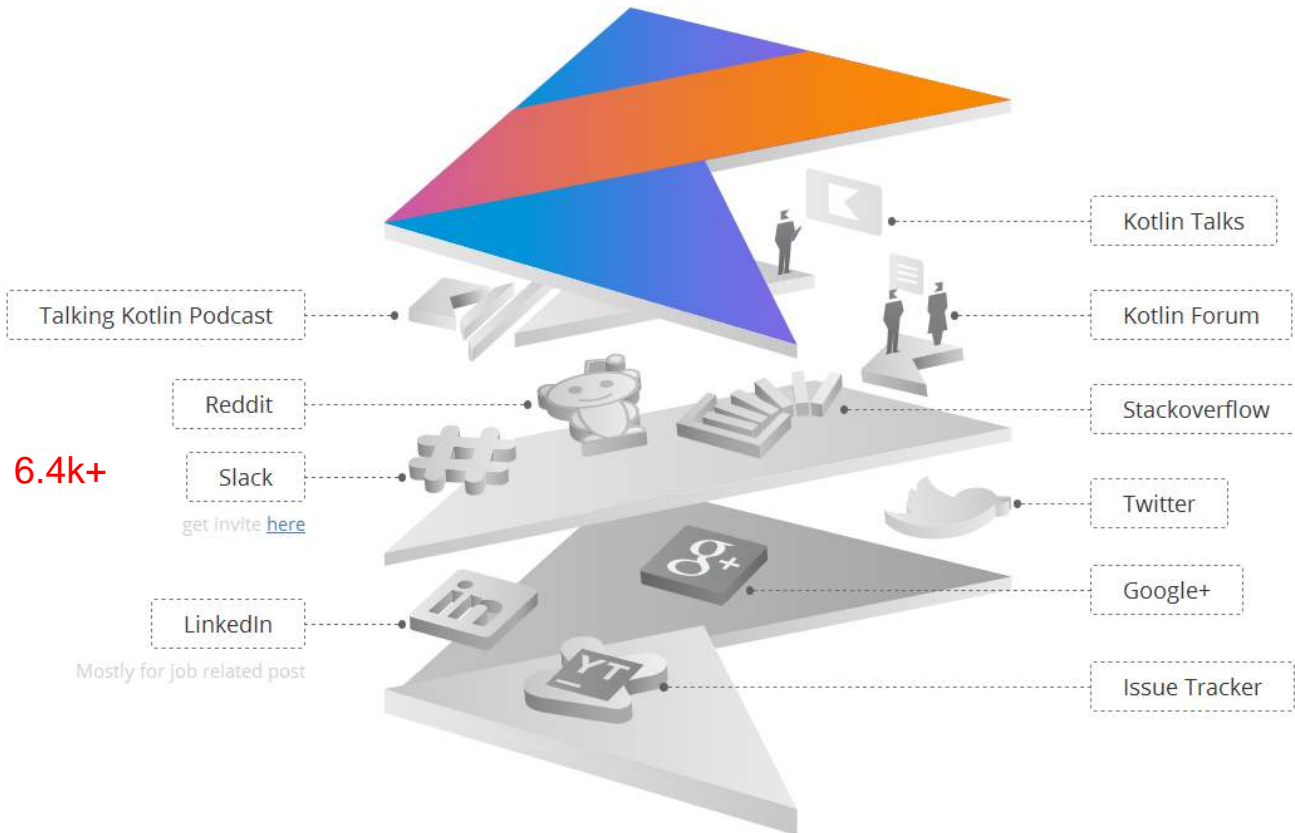
Интерес мира к Kotlin: строки кода на GitHub



Интерес мира к Kotlin: запросы в Google



Мировое сообщество



Сообщество в ЕРАМ, Минск

1. 4 production проекта: Mobile, Backend
2. 5 tech talk-ов в 2016 году
3. Ежемесячные встречи в 2017
4. 150+ упоминаний в профайлах сотрудников

А что есть в Kotlin чего нет в Java?

1. Data classes
2. Smart casts
3. Null safety
4. Extension functions
5. String templates (easy)
6. ~~Lambda's~~
7. no checked exceptions
8. Singletons

Как мы пробовали Kotlin?

1. Web UI tests – senior QE
2. Web Services tests – senior QE
3. Web UI tests – junior QE
4. Web Services tests – junior QE
5. Kotlin libs – senior QE

Библиотеки на Kotlin для тестирования

<https://github.com/KotlinBy/awesome-kotlin#libraries-frameworks-tests>



Tests [Back ↑](#)

- [JetBrains/spek](#) - A specification framework for Kotlin.
- [npryce/hamcrest](#) - A reimplementaion of Hamcrest to take advantage of Kotlin language features.
- [nhaarman/mockito-kotlin](#) - Using Mockito with Kotlin.
- [MarkusAmshove/Kluent](#) - Fluent Assertion-Library for Kotlin.
- [winterbe/expect](#) - BDD assertion library for Kotlin.
- [kotlintest/kotlintest](#) - KotlinTest is a flexible and comprehensive testing tool for the Kotlin ecosystem based on and heavily inspired by the superb Scalatest.
- [dmcg/konsent](#) - An acceptance test library for Kotlin.
- [raniejade/kspec](#) - Kotlin Specification Framework.
- [EPadronU/balin](#) - Balin is a browser automation library for Kotlin. It's basically a Selenium-WebDriver wrapper library inspired by Geb.
- [dmcg/k-sera](#) - A JMock wrapper for Kotlin.
- [dam5s/aspn](#) - Aspen is an RSpec and Spek inspired test runner for Kotlin.

Data classes



```
public class Person {  
    final String firstName;  
    final String lastName;  
    public JavaPerson(...) {  
        ...  
    }  
    // Getters  
    ...  
    // Hashcode / equals  
    ...  
    // toString  
    ...  
    // Egh...  
}
```

```
data class Person(  
    val firstName: String,  
    val lastName: String  
)
```

Optional method parameters



```
class Account {  
  
    void update (String email,  
                String fBiD, Boolean active){  
    ...  
    }  
  
    void update (String email,  
                boolean active){  
        update (email, "NA", active)  
    }  
  
    ...  
    }
```

```
class Account {  
  
    fun update(  
        email: String,  
        fBiD : String = "NA",  
        active : Boolean = true){  
        ...  
    }  
}  
  
account.update ("a@b.com")  
account.update ("a@b.com", fbId =  
"john")  
account.update ("a@b.com", fbId =  
"john", active = false)
```

Smart casts



```
if (expr instanceof Number) {  
    System.out.println((Number)expr).getValue();  
}
```



```
if (expr is Number) {  
    println(expr.getValue());  
}
```

Types inference



```
final AccessDeniedException myExceptionInst = new  
AccessDeniedException("Msg");
```



```
val myExceptionInst = AccessDeniedException("Msg");
```

Null safety



```
var a: String = "abc"
```

```
// compilation error
```

```
a = null
```

```
// will never fail
```

```
val l = a.length
```

```
-----
```

```
var b: String? = "abc"
```

```
b = null // ok
```

```
// will fail
```

```
val l = b.length
```

```
// safe call
```

```
result = b?.length
```

```
// safe call
```

```
if (null!=bd) result = bd.length
```

```
// quick check for null
```

```
val l =
```

```
if (b != null) b.length else -1
```

```
// using Elvis operator
```

```
val l = b?.length ?: -1
```

```
// safe casts
```

```
...
```

Extensions



```
fun <T : BasePage> WebDriver.open(page: KClass<T>): T {  
    waitForPageOpened(page)  
    return HtmlElementLoader.createPageObject(page.java, this)  
}
```

```
driver.open(LoginPage::class)
```

String patterns



```
val myWord: String = "Test"  
println("Length of word '$myWord' is ${myWord.length}")
```


No checked exceptions



```
@Throws (IOException::class)  
fun foo() {  
    throw IOException()  
}
```

workaround

```
val a: Int? = try {  
    parseInt(input)  
} catch (e: NumberFormatException) {  
    null  
}
```

'try' returns value!

Our DriverManager

Java

```
1 public final class DriverManager {
2
3     private static final Logger LOGGER = LoggerFactory.getLogger(DriverManager.class);
4
5     private static final String DEFAULT_NAME = "Default name";
6
7     private ThreadLocal<Map<String, Driver>> driversCache = new ThreadLocal<Map<String, Driver>>() {
8
9         @Override
10        protected Map<String, Driver> initialValue() {
11            return new HashMap<String, Driver>();
12        }
13    };
14
15    private Map<DriverType, WebDriverFactory> factories = new HashMap<DriverType, WebDriverFactory>() {
16        /**
17         *
18         */
19        private static final long serialVersionUID = 1L;
20
21        {
22            put(DriverType.FIREFOX, new FirefoxWebDriverFactory());
23        }
24    };
25
26    private static final DriverManager INSTANCE = new DriverManager();
27
28    private DriverManager() {
29
30    }
31
32    public static DriverManager getInstance() {
33        return INSTANCE;
34    }
35
36    public synchronized void setFactory(DriverType type, WebDriverFactory factory) {
37        factories.put(type, factory);
38    }
39 }
```

Kotlin

```
object DriverManager {
    private const val DEFAULT_NAME = "Default name"
    private val LOGGER = LoggerFactory.getLogger(DriverManager::class.java)
    private val driversCache = object : ThreadLocal<MutableMap<String, WebDriver>>() {
        override fun initialValue(): MutableMap<String, WebDriver> = HashMap()
    }
    private val factories = mutableMapOf<DriverType, WebDriverFactory>(DriverType.FIREFOX to FirefoxWebDriverFactory())
    @Synchronized fun setFactory(type: DriverType, factory: WebDriverFactory) = factories.put(type, factory)
}
```

Выводы: где косяки?

1. Пересечение Java-Kotlin (null safety issue)
2. Сложно тем кто в первый раз видит код
3. «Сырой» плагин для Eclipse
4. Слабее Reflection (w/around: any Kotlin класс может быть сконвертирован в Java класс через `.java()`)
5. Конфликты ключевых слов языка с ключевыми словами сторонних библиотек (w/around: используй `косые кавычки`)

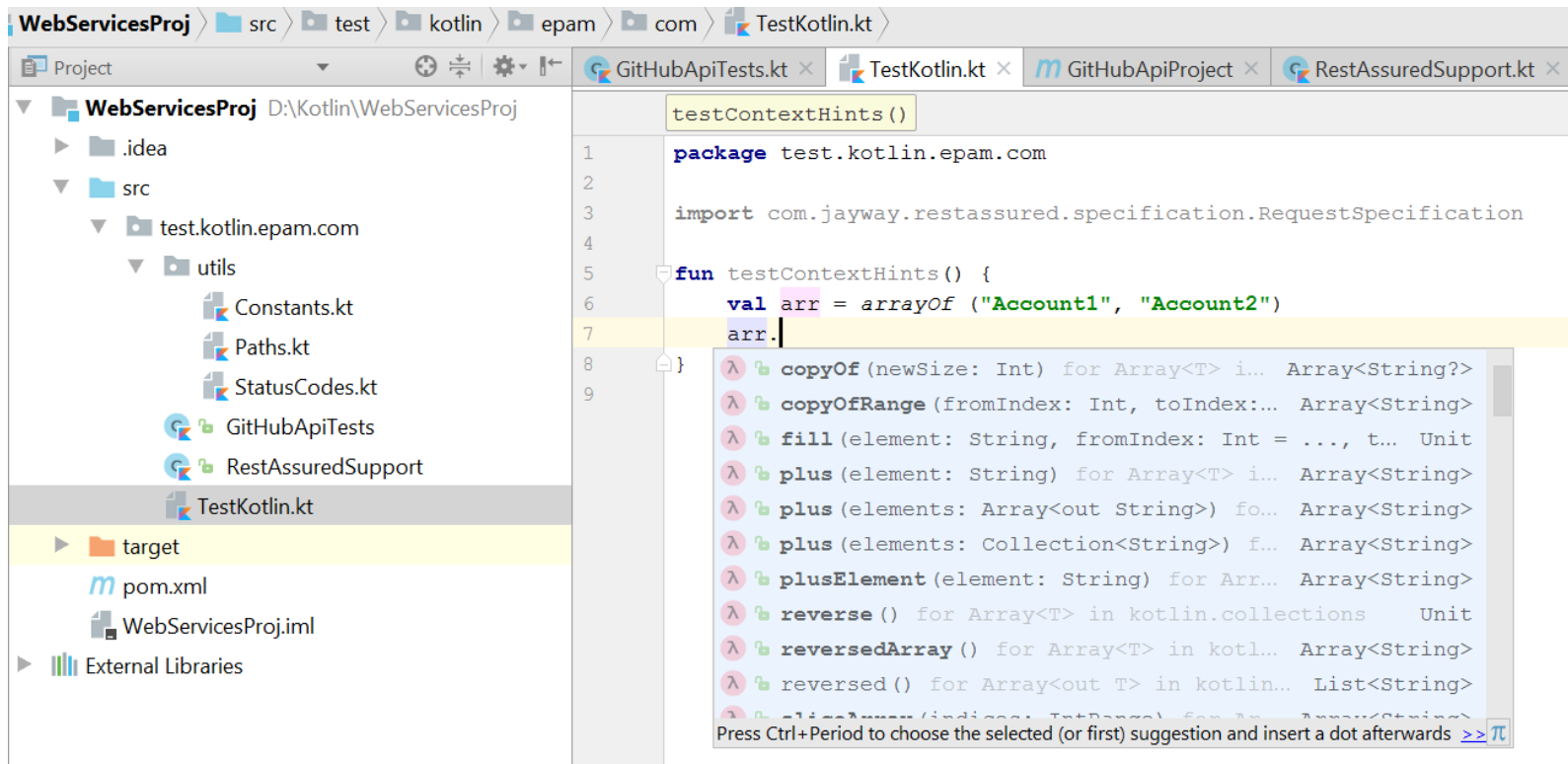
Выводы: что хорошо?

1. 100% совместимость с Java
2. Отличная работа в IntelliJ IDEA
3. без плагинов
4. Быстрое изучение самостоятельно (learning curve 1-4 weeks)
5. KDос (поддержка Markdown)

Совместимость с Java

1. Можно использовать любые Java-библиотеки
2. Можно создать проект с Java и Kotlin кодом
3. Даже есть конвертер Java-Kotlin

Написание кода в IntelliJ IDEA



The screenshot shows the IntelliJ IDEA IDE interface. The breadcrumb navigation at the top indicates the file path: `WebServicesProj > src > test > kotlin > epam > com > TestKotlin.kt`. The left sidebar shows the project structure for `WebServicesProj`, with the `TestKotlin.kt` file selected. The main editor displays the following Kotlin code:

```
1 package test.kotlin.epam.com
2
3 import com.jayway.restassured.specification.RequestSpecification
4
5 fun testContextHints() {
6     val arr = arrayOf("Account1", "Account2")
7     arr.
8 }
9
```

A completion popup is visible at the end of line 7, listing various Kotlin array methods such as `copyOf`, `copyOfRange`, `fill`, `plus`, `reverse`, and `reversedArray`. The popup also includes a hint: "Press Ctrl+Period to choose the selected (or first) suggestion and insert a dot afterwards >>π".

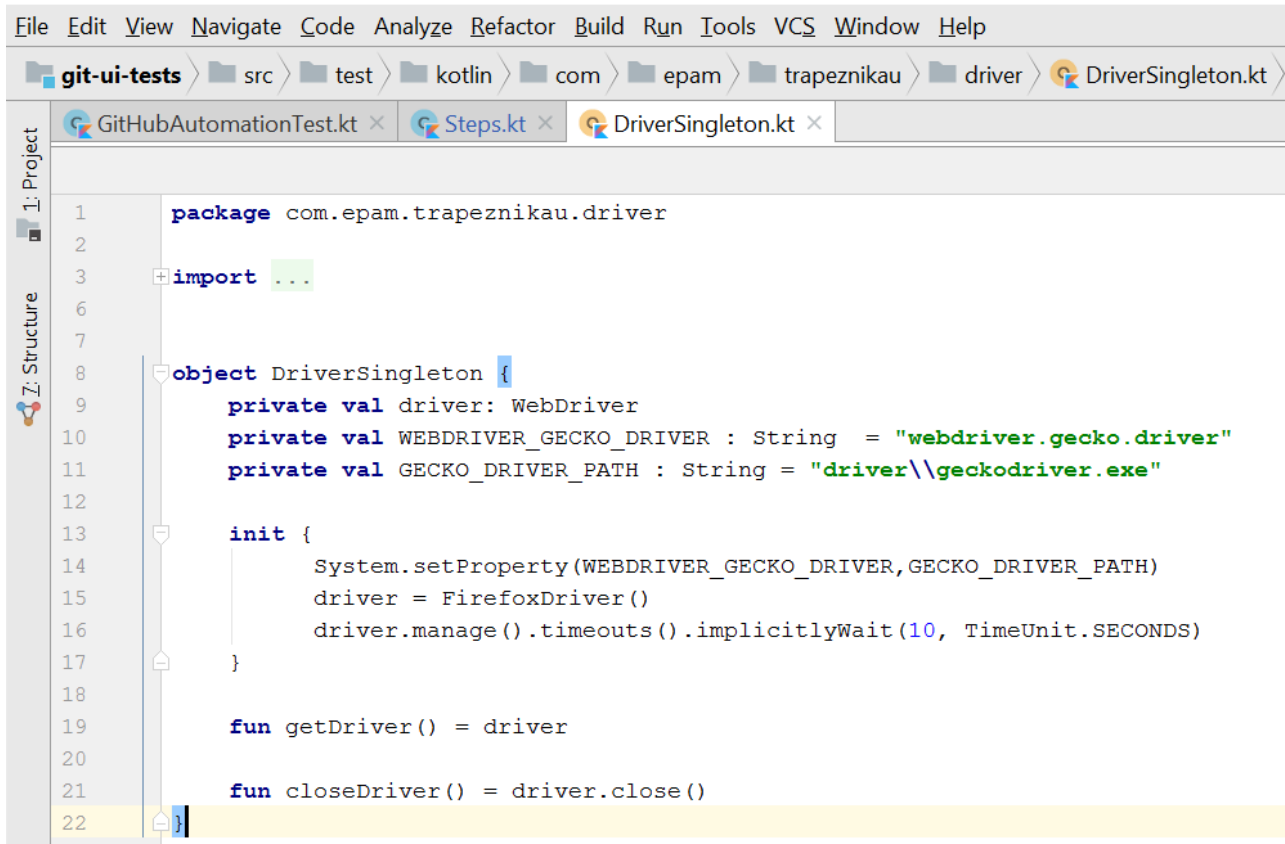
Изучение: результаты эксперимента

githubautomationtest - [D:\Temp\kotlin-ta-lab-test-automation\git-ui-tests] - [githubautomationtest] - ...src\test\kotlin\com\epam\trapeznikau\GitHubAutomationTest.kt - IntelliJ IDEA 2017.1

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
git-ui-tests > src > test > kotlin > com > epam > trapeznikau > GitHubAutomationTest.kt
GitHubAutomationTest.kt x Steps.kt x DriverSingleton.kt x
GitHubAutomationTest
11
12 class GitHubAutomationTest {
13
14     lateinit var step : Steps
15
16     @BeforeTest
17     fun initSteps() {
18         step = Steps()
19         step.initBrowser()
20     }
21
22     @AfterTest
23     fun closeDriver() {
24         // step.closeDriver()
25     }
26
27     @Test
28     @Parameters("login", "password")
29     fun tstGitLoginIn(login : String, password : String)
30         = assertEquals("https://github.com/GitHubTestingProject", step.signIn(login, password))
31
32     @Test(dependsOnMethods = arrayOf("tstGitLoginIn"))
33     @Parameters("nameRepo")
34     fun tstCreateRepository(nameRepo : String)
35         = assertEquals("NewRepo", step.createRepo(nameRepo))
36
37     @Test(dependsOnMethods = arrayOf("tstCreateRepository"))
38     @Parameters("newRepoName")
39     fun tstModifyRepository(nameRepo : String)
40         = assertEquals("ModifyRepo", step.modifyRepo(nameRepo))
41
```

Изучение: результаты эксперимента

githubautomationtest - [D:\Temp\kotlin-ta-lab-test-automation\git-ui-tests] - [githubautomationtest] - ...src\test\kotlin\cc



```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
git-ui-tests > src > test > kotlin > com > epam > trapeznikau > driver > DriverSingleton.kt
GitHubAutomationTest.kt x Steps.kt x DriverSingleton.kt x
1 package com.epam.trapeznikau.driver
2
3 import ...
4
5
6
7
8 object DriverSingleton {
9     private val driver: WebDriver
10    private val WEBDRIVER_GECKO_DRIVER : String = "webdriver.gecko.driver"
11    private val GECKO_DRIVER_PATH : String = "driver\geckodriver.exe"
12
13    init {
14        System.setProperty(WEBDRIVER_GECKO_DRIVER,GECKO_DRIVER_PATH)
15        driver = FirefoxDriver()
16        driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS)
17    }
18
19    fun getDriver() = driver
20
21    fun closeDriver() = driver.close()
22 }
```


Изучение: результаты эксперимента

1. Демо Web и UI тесты с нуля
2. Junior-ы : за 20 часов (70% времени на изучение Kotlin)
3. Senior-ы: 12 часов
4. Самообучение для того чтобы сделать Code Review: 10 часов
5. **Сокращение кода** по сравнению с Java: 0-70%



Выводы: что с этим делать?

1. Может сократить время на поддержку кода фреймворков с большим Business и Test Layers
2. Риски минимальны: можно писать на Kotlin только ту часть кода, где это имеет смысл
3. Есть демо проекты и отличный online учебник, благодаря которым можно сделать быстрый старт своего фреймворка



Стратегическое планирование 2017

Давайте проговорим **результаты стратегического планирования ЕРАМ Test Automation Department-а на 2017 год** и покажем, как сильно, векторы эволюции Автоматизации тестирования согласно закону Седова совпали с нашим виденьем развития Автоматизации.

Стратегическое планирование было условно разбито на **3 большие группы задач: «Talent management in automated testing», «Solution Accelerators» и «Customer Management in automated testing».**

Давайте разберем секцию **«Solution Accelerators»**, так как техническая составляющая в ней преобладает.



Solution Accelerators

1. Working solutions: Report Portal, Mobile Farm & IoT Farm
2. From idea to product
3. Machine learning in automated testing
4. Kotlin in automated testing
5. Contributing to Selenium, EPAM and open-source



Вселенная подчиняется фундаментальным законам; знание и применение простых прикладных физических законов в нашей ежедневной работе, позволяет строить качественное прогнозирование, как на индивидуальном уровне, так и на уровне компании в целом.



Не пренебрегайте элегантным мощным инструментом, **законом Седова**, который так удобно использовать в качестве «лакмусовой бумажки» эволюции: соответствует, не соответствует; будет использоваться сегодня, завтра и послезавтра или же это инструмент-однодневка, который, можно использовать, но инвестировать время и силы в его глубокое изучение и тем более развитие не следует; закон Седова позволяет определить вектор, фокус усилий, набор наиболее перспективных решений.



EPAM Test Automation Division: solution accelerators 2017

- 1 Working solutions: Report Portal, Mobile Farm, IoT farm
- 2 From Idea to Product
- 3 Machine Learning in automated testing
- 4 Kotlin in automated testing
- 5 Contribution to Selenium, EPAM and open source

Contribute to Selenium: why

1. To be a creator, inventor, implement something new
2. Help colleagues; a chance to make our life, our world, a little bit better
3. An opportunity to communicate with the best world professionals via source code
4. Understand how the favorite tool works
5. Ability to learn from the best professionals



Contribute to Selenium: why

6. Add public recognition
7. Increase professional value
8. Self-positioning instrument
9. To be a part of something important
10. First steps to indie programming, your own start-up
11. Open source – as a way of life



Contribute to Selenium: together with EPAM, why

1. Grow together with company
2. Use companies support Open Source contribution
3. A way to learn something new not only inside your current project
4. An opportunity to boost your career
5. An opportunity to play some special role



Contribute to Selenium: how

1. Build process to support engineers

Technical support

Contribution process support

- Help to choose proper point of improvements
- Help with code review and other technical stuff
- Help in preparing packages
- Help with org questions related to Selenium committee
- Build a Kanban process to simplify org team efforts



Contribute to Selenium: how

1. Ideally:

You want to code

All other “not technical” questions – our responsibility

And we are working on such a supportive process



Contribute to Selenium: EPAM and COMAQA Community

1. Internal activists from EPAM
2. External activists from COMAQA
3. Mixed 😊



Contribute to Selenium: what

1. Bugs verification \ systematization
2. Bug fixing (based on project needs)
3. Selenium documentation improvement
4. Decorators documentation improvement
5. Decorators implementation
6. New features development
7. Bug fixing (based on our subjective vision)



Антон Семенченко «Закон иерархических компенсаций
Седова и C++ Core Guidelines»

<https://corehard.by/2016/02/15/conf2016-c-core-guidelines/>

Антон Семенченко «Законы создания IT команд и
следствия законов для IT проектов»

<https://corehard.by/2016/12/06/secr-2016-antonsemenchenko/>



1. [Универсальная история](#)
2. [Акоп Назаретян. Цивилизационные кризисы в контексте Универсальной истории](#)
3. [Евгений Седов. Информационно-энтропийные свойства социальных систем](#)
4. [Вертикаль Панова-Снукса](#)
5. [Питер Друкер. Менеджмент. Вызовы XXI века](#)
6. [Виген Геодакян. Эволюционная теория пола](#)
7. Франс де Вааль. Политика у шимпанзе.
Власть и секс у приматов



8. Михаил Веллер. Испытатели счастья
9. [Герберт Спенсер. Основные начала](#)
10. Гради Буч. Объектно-ориентированный анализ и проектирование, первая часть
11. [Конференция Global Future 2045](#)
12. [GF2045. Акоп Назаретян. Проблема середины XXI века](#)
13. [GF2045. Михаил Веллер. Человек в системе энергоэволюционизма](#)
14. [GF2045. Александр Панов. Сингулярность эволюции и будущее фундаментальной науки](#)
15. И многие, многие, многие другие



Источники информации

- http://rulibs.com/ru_zar/prose_contemporary/veller/1/j18.html



Feel free to ask questions

semenchenko@dpi.solutions

Skype: dpi.semenchenko

+375 33 33 46 120

+375 44 74 00 385

<https://www.linkedin.com/in/anton-semenchenko-612a926b>

<https://www.facebook.com/semenchenko.anton.v>

<https://twitter.com/comaqa>

www.comaqa.by

www.corehard.by



