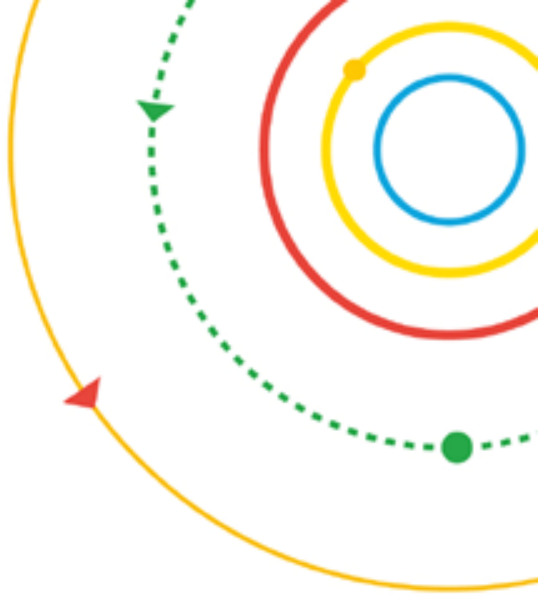
Технический долг

пока не слишком поздно

Единая
Фронтальная
Система

Цифры и факты



- **$10^3 – 10^5$** – сценариев обслуживания клиентов
- **$10^4 – 10^6$** – пользовательских форм
- **$10 – 30 \times 10^6$** – строк кода

Единая Фронтальная Система

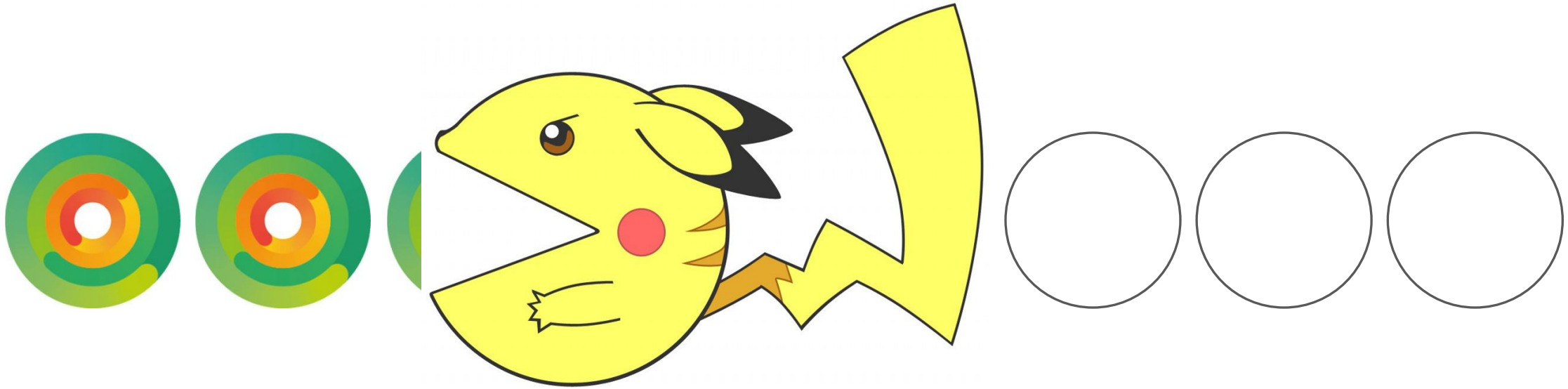
- **$10^3 – 10^5$** – сценариев обслуживания клиентов
- **$10^4 – 10^6$** – пользовательских форм
- **$10 – 30 \times 10^6$** – строк кода

Единая Фронтальная Система

Сбербанка

Технический долг – «друг» любой крупной корпоративной IT-системы

- **$10^3 – 10^5$** – сценариев обслуживания клиентов
- **$10^4 – 10^6$** – пользовательских форм
- **$10 – 30 \times 10^6$** – строк кода



Создание технического долга

Заказчик

Энтропия

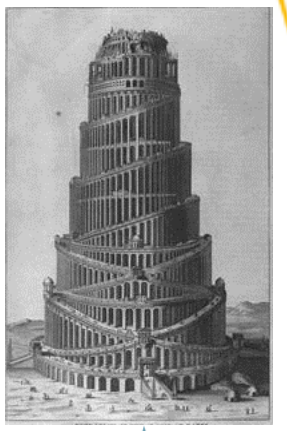
Продукт

Бизнес-аналитики:
Концентрация на потребностях заказчика

UX-проектировщики:
Концентрация на пользователе

Системные аналитики:
Концентрация на системном функционале

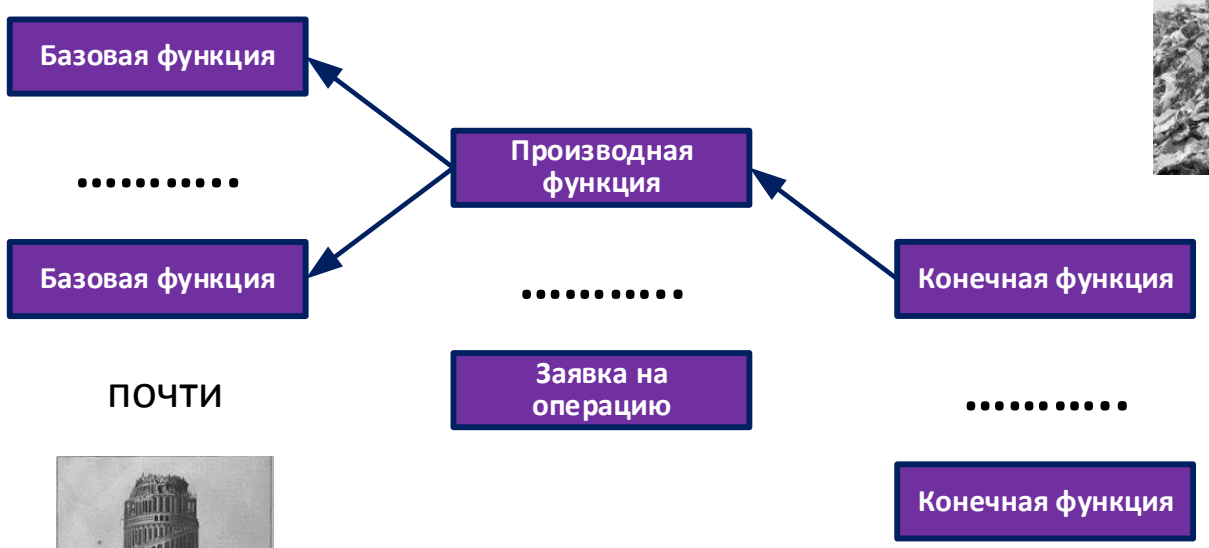
Прикладные разработчики:
Концентрация на ограничениях реализации



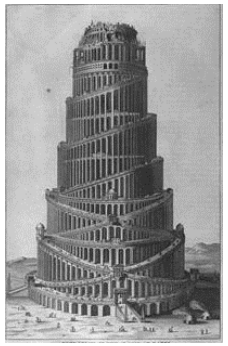
Прямой импорт технического долга

Чистота реализации нового функционала ограничена техническим долгом используемого

полная

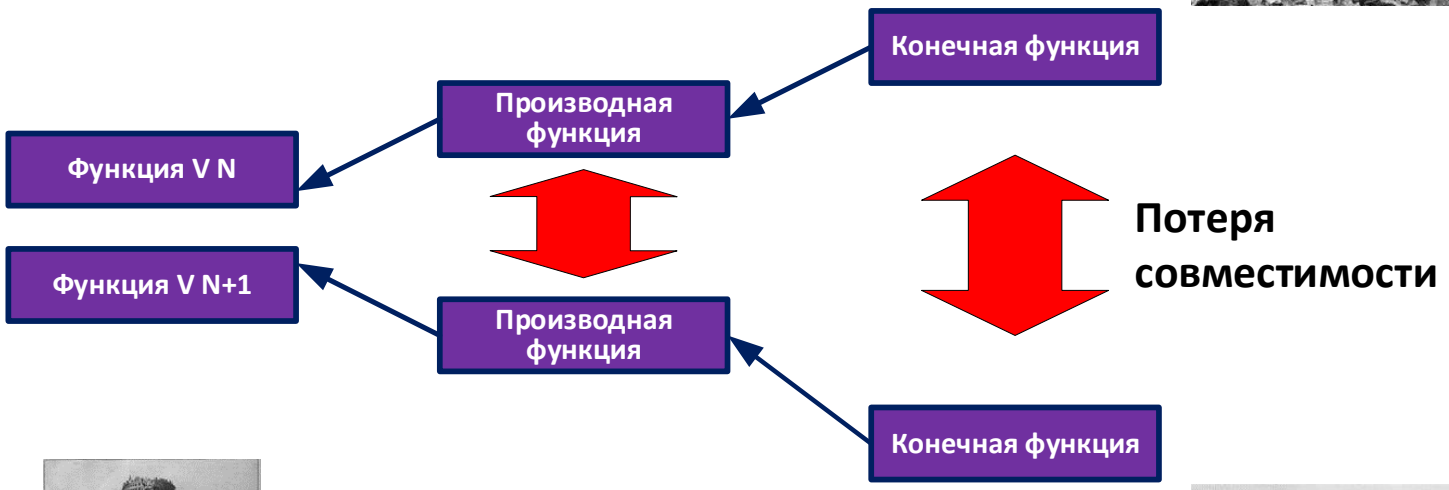


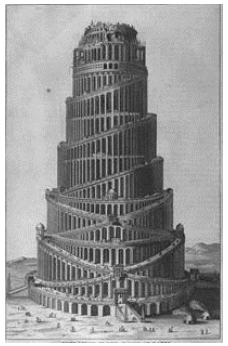
почти



Наследуемая потеря совместимости

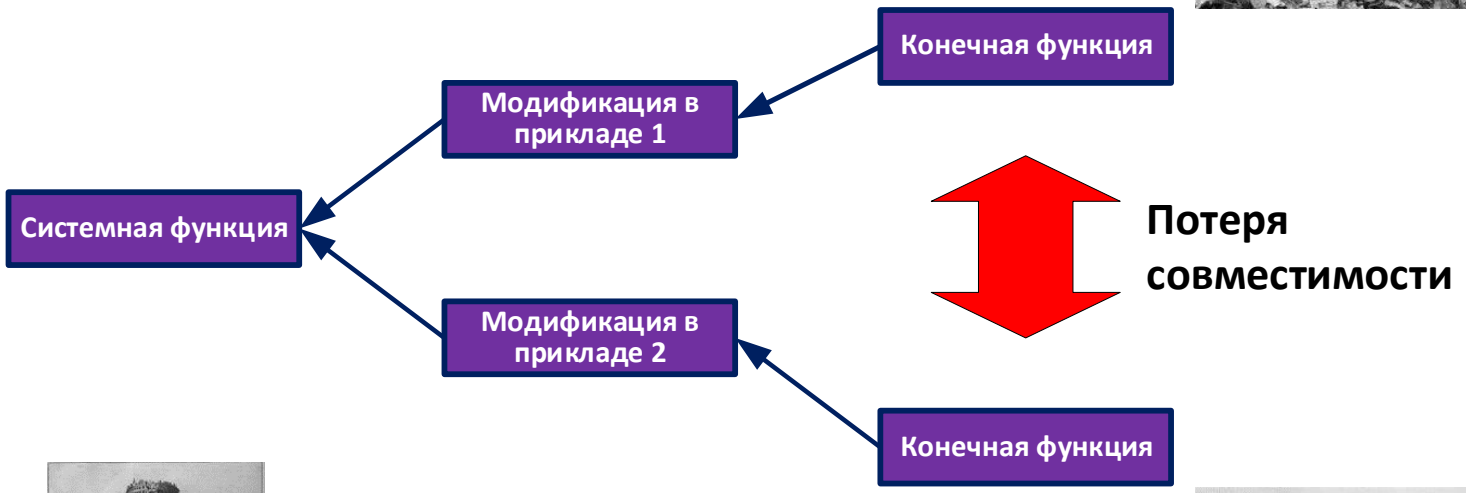
(Jar hell) Потеря совместимости передается по зависимостям

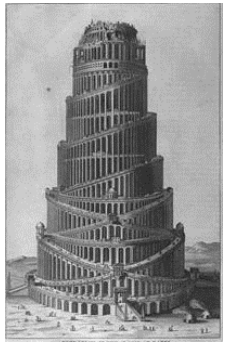




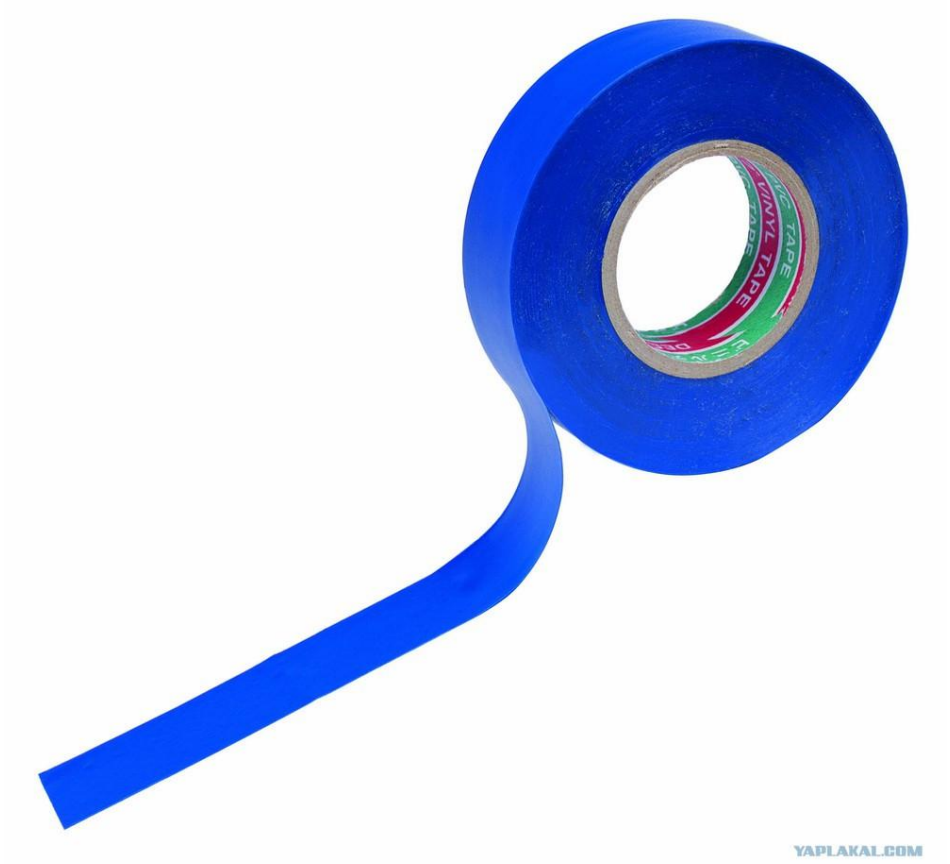
Миграция системного функционала в прикладной

Попытка доработки системного функционала в месте потребления является причиной потери совместимости



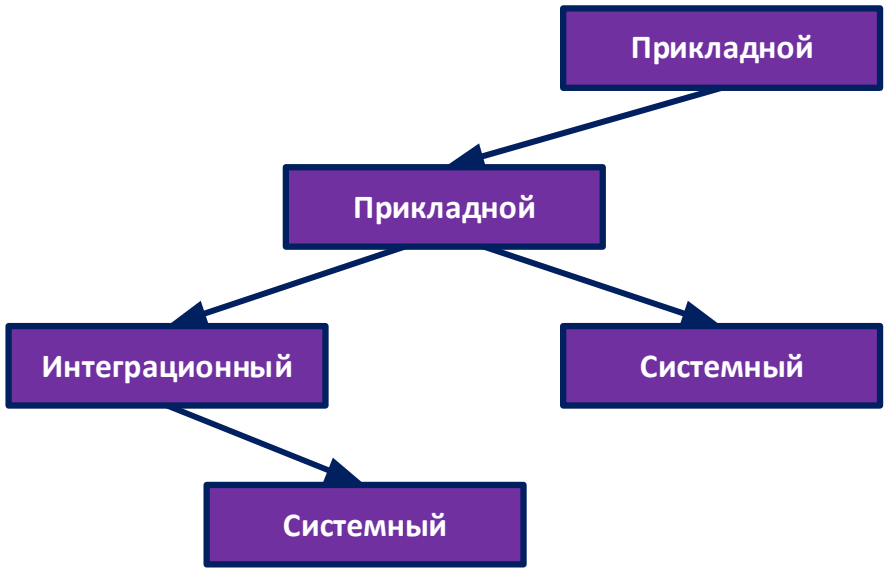


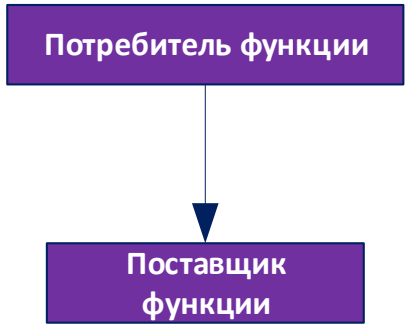
Главный элемент архитектуры ЕФС



- **Модуль** – единица сильно связанного функционала
- Гранулярность функционала: 20-50 типов (95% модулей)

Изоляция по типу функционала

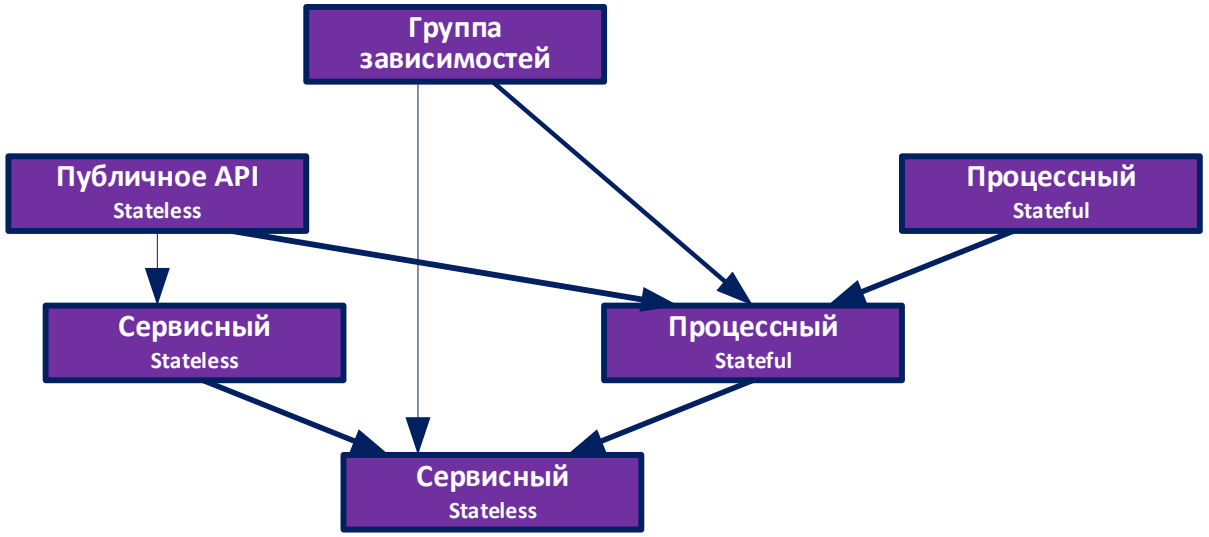


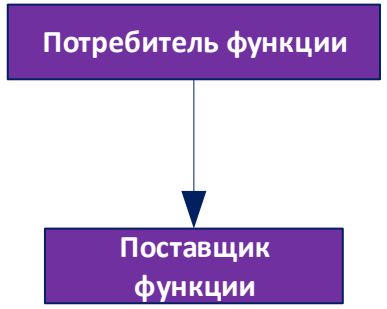


Разделение в ЕФС:

- Прикладной **(85%)**
- Системный (10%)
- Интеграционный (5%)

Изоляция по режиму использования

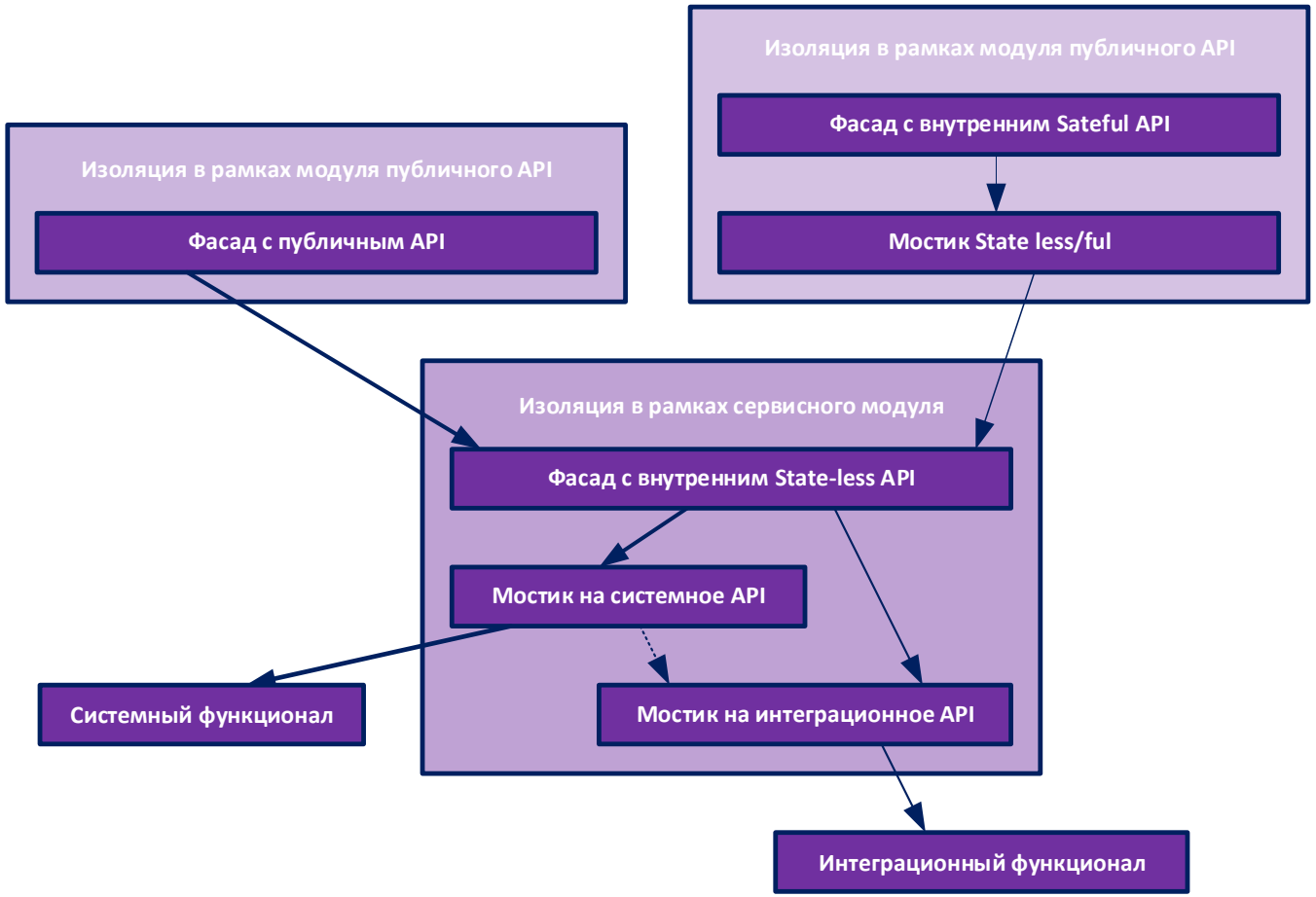




Разделение в ЕФС:

- Stateless (25%)
- Stateful (65%)
- Open API (10%)

Еще больше изоленты!!!

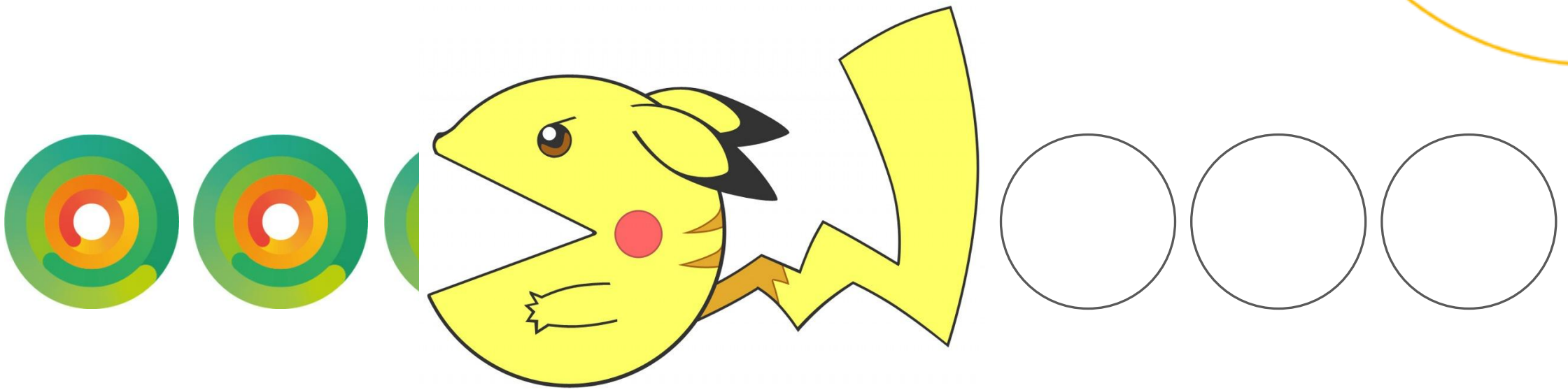


Рекомендуется всем:

- Фасады
- Мостики
- Адаптеры

Так что же такое технический долг?

Технический долг – нарушение требований архитектуры к реализации прикладного и системного функционала

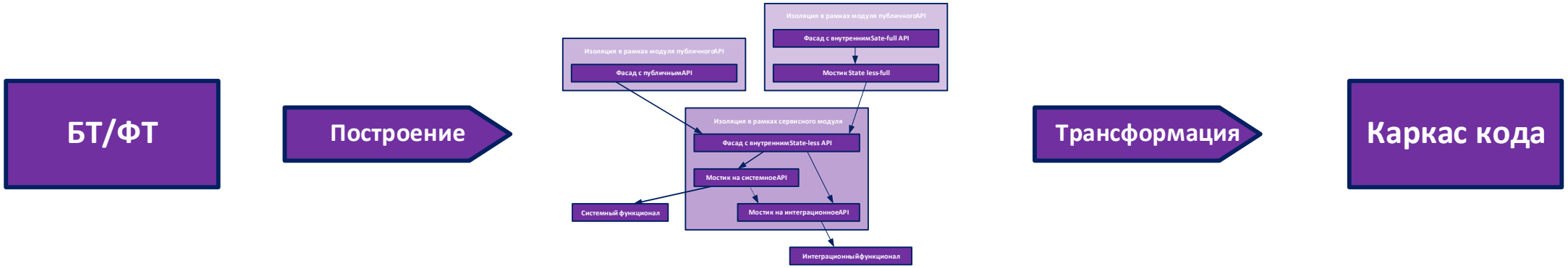


Все остальное это:

- Функциональный долг
- Нарушение правил кодирования

MDD: как доставить архитектурное требование в код?

Разработка, управляемая моделями, (англ. model-driven development) — это стиль разработки программного обеспечения, когда модели становятся основными артефактами разработки, из которых генерируется код и другие артефакты

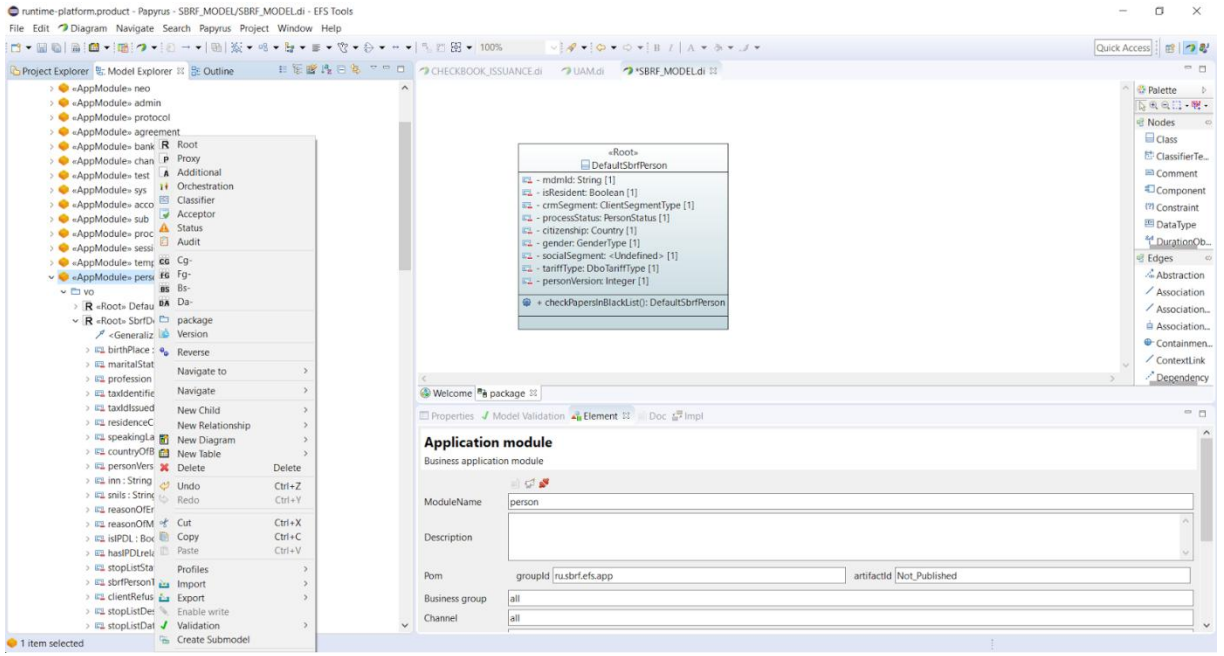


Модель

Характер использования MDD

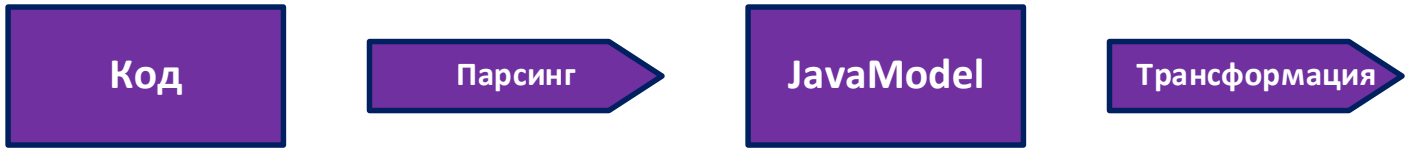
Генерируемые элементы программного кода

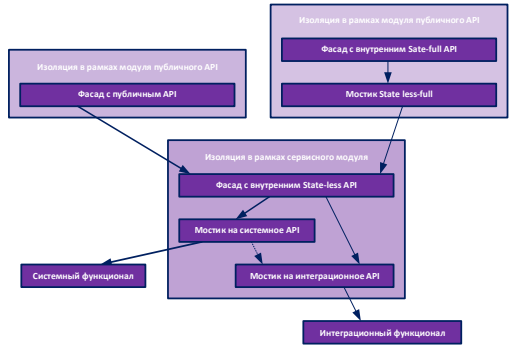
- Классы прикладных объектов, включая JAXB, JPA-аннотации
- Классы сервисов и контракты методов, включая JAX-WS и JAX-RS аннотации
- Spring-конфигурация
- Конфигурация JPA
- POM-файл модуля



Reverse: как проконтролировать технический долг?

Обра́тная разрабо́тка (обратный инжиниринг, реверс-инжиниринг; англ. reverse engineering) — исследование некоторого готовой программы с целью понять принцип его работы; например, чтобы обнаружить недокументированные возможности.



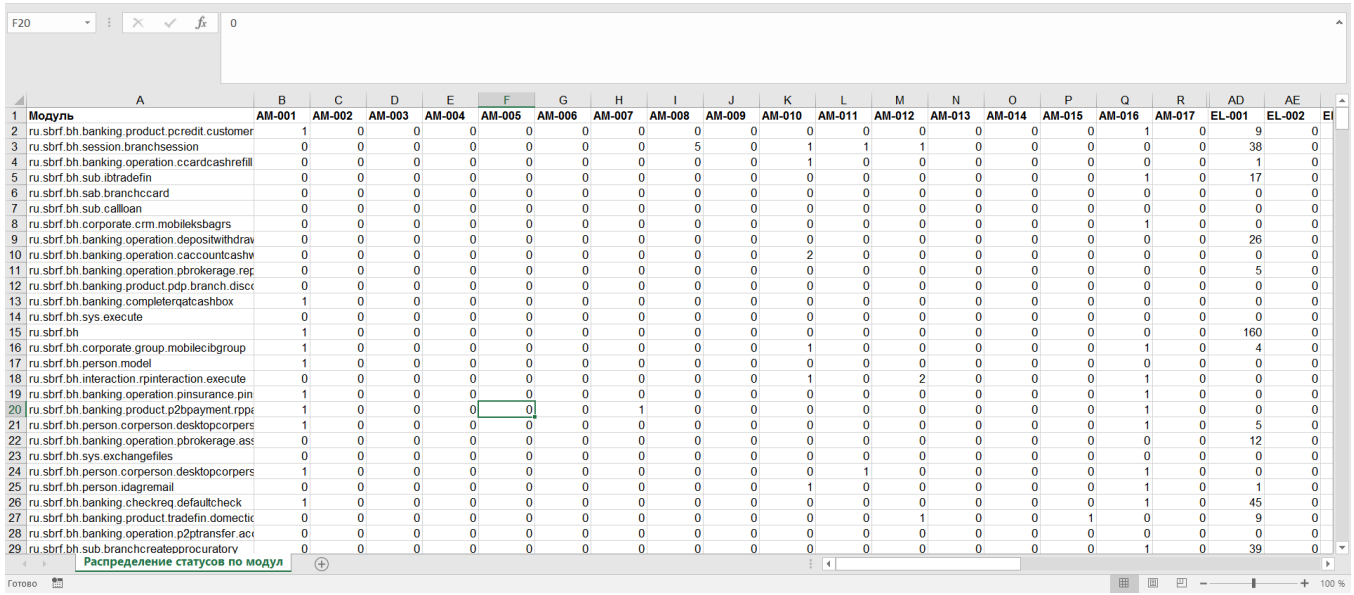


Обогащенная модель

Имплементация Reverse Engineering

Группы правил валидации

- На допустимые зависимости модулей
- На версионность по зависимостям
- На зависимости между типами по ролям
- На объем имплементации системного функционала
- На использование паттернов проектирования



Модуль	AM-001	AM-002	AM-003	AM-004	AM-005	AM-006	AM-007	AM-008	AM-009	AM-010	AM-011	AM-012	AM-013	AM-014	AM-015	AM-016	AM-017	EL-001	EL-002
ru.sbrf.bh.banking.product.pcredit.customer	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	9	0
ru.sbrf.bh.session.branchsession	0	0	0	0	0	0	0	5	0	1	1	1	0	0	0	0	0	38	0
ru.sbrf.bh.banking.operation.ccardcashrefill	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
ru.sbrf.bh.sub.ibtradefin	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	17	0
ru.sbrf.bh.sab.branchccard	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ru.sbrf.bh.sub.calloan	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ru.sbrf.bh.corporate.crm.mobileksbagrs	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
ru.sbrf.bh.banking.operation.depositwithdrav	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	26	0
ru.sbrf.bh.banking.operation.caccountcashv	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
ru.sbrf.bh.banking.operation.pbrokerage.rep	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
ru.sbrf.bh.banking.product.pdp.branch.disc	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ru.sbrf.bh.banking.completerqatcashbox	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ru.sbrf.bh.sys.execute	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ru.sbrf.bh	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	160	0
ru.sbrf.bh.corporate.group.mobilecibgroup	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	4	0
ru.sbrf.bh.person.model	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ru.sbrf.bh.interaction.rpinteraction.execute	0	0	0	0	0	0	0	0	0	1	0	2	0	0	0	0	1	0	0
ru.sbrf.bh.banking.operation.pinsurance.pin	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
ru.sbrf.bh.banking.product.p2bpayment.rpp	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0
ru.sbrf.bh.person.corperson.desktopcorpers	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	5	0
ru.sbrf.bh.banking.operation.pbrokerage.ass	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0
ru.sbrf.bh.sys.exchangefiles	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ru.sbrf.bh.person.corperson.desktopcorpers	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
ru.sbrf.bh.person.idagremail	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0
ru.sbrf.bh.banking.checkreq.defaultcheck	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	45	0
ru.sbrf.bh.banking.product.tradefin.domestic	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	9	0
ru.sbrf.bh.banking.operation.p2ptransfer.ac	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ru.sbrf.bh.sub.branchcreatepprocuratory	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	39	0

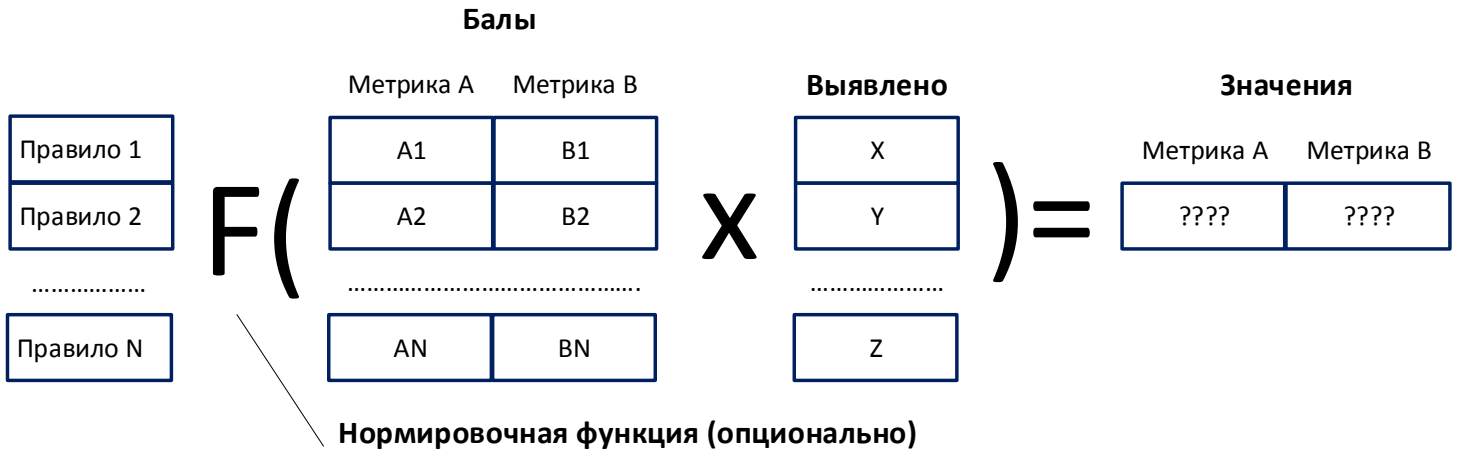
Расчет метрик тех. долга

Простые численные метрики

- Прямой подсчет «чего-то» в коде
- Прямые расчеты на основании итогов подсчета

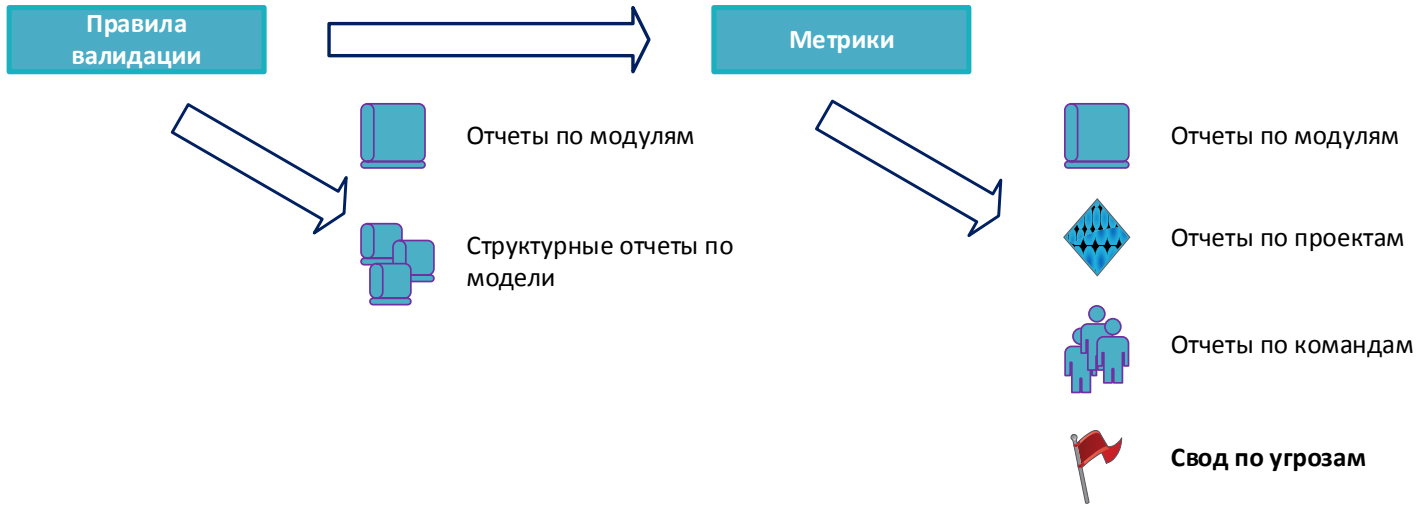
Экспертные скоринговые карты для комплексных метрик

- Каждому значимому для метрики правилу валидации присваивается определенный бал
- По объекту наблюдения выявляются правила валидации
- Метрика рассчитывается как сумма произведений количества выявленных правил на величину бала каждого правила

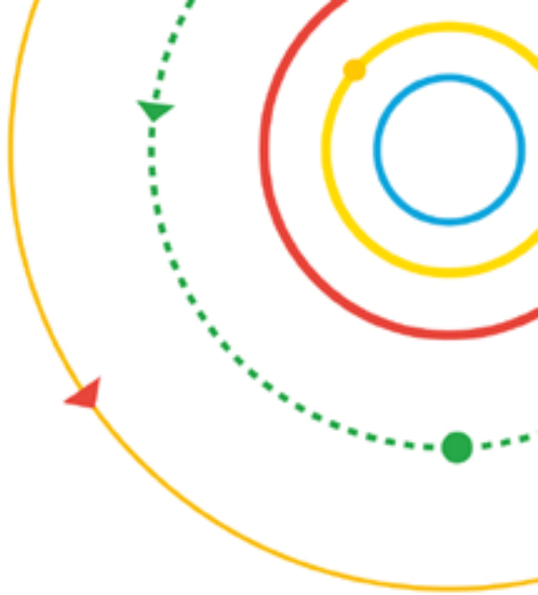


Формат агрегации метрик

- **Детализация в разрезе отдельных модулей**
  - Состав правил валидации, сработавших по каждому из модулей
- **Агрегированные метрики в разрезах**
  - Прикладных модулей (переиспользуемых блоков кода)
  - Команд (владеющих и изменяющих код)
  - Проектов (в рамках целей которых код создается и меняется)
- **Агрегирующие показатели в целом по коду**
  - Детализация итогов валидации для выявления тенденций



Ключевые комплексные метрики



- **Прикладной технический долг** (0 - ∞)

Величина бала, характеризующая условный объем прикладного долга

- **Системный технический долг**(0 - ∞)

Величина бала, характеризующая условный объем прикладного долга

Каждые **100** балов долга – это **0.5 – 2** дня на исправление.

Последствия наличия прикладного долга

При прикладном долге более **0.3-1К** балов, скорость инкремента начнет падать

При прикладном долге более **1-2К** балов, любой новый инкремент функционала будет порождать новый прикладной долг

При прикладном долге более **2-3К** балов, инкремент функционала будет почти невозможен

При прикладном долге **0.5-1К** балов упадет скорость фиксации багов и возможность реюза

Последствия наличия системного долга

При системном долге более **300** балов есть вероятность проблем при взаимодействии с другим прикладным функционалом

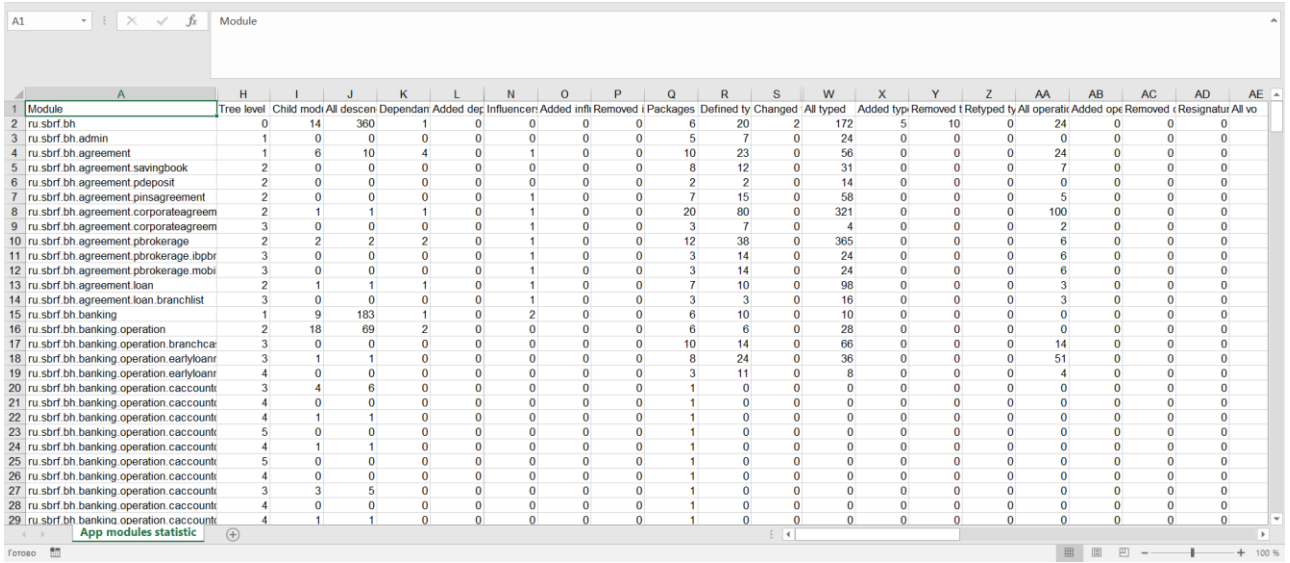
При системном долге более **500** балов прохождение ИФТ будет невозможно

Статический анализ

Стати́ческий ана́лиз ко́да (англ. static code analysis) — анализ программного обеспечения, производимый без реального выполнения исследуемых программ.

Анализируемые группы показатели по модулям

- Количество и структура типов по ролям
- Зависимости по прикладным модулям
- Зависимости по интеграционным и системным модулям
- Характер и динамика вносимых изменений



Module	Tree level	Child modi	All descen	Dependan	Added dep	Influencer	Added infl	Removed i	Packages	Defined ty	Changed	All typed	Added typ	Removed t	Retyped ty	All operati	Added ope	Removed c	Resignatur	All vo
ru.sbrf.bh	0	14	360	1	0	0	0	0	6	20	2	172	5	10	0	24	0	0	0	
ru.sbrf.bh.admin	1	0	0	0	0	0	0	0	5	7	0	24	0	0	0	0	0	0	0	
ru.sbrf.bh.agreement	1	6	10	4	0	1	0	0	10	23	0	56	0	0	0	24	0	0	0	
ru.sbrf.bh.agreement.savingbook	2	0	0	0	0	0	0	0	8	12	0	31	0	0	0	7	0	0	0	
ru.sbrf.bh.agreement.pdeposit	2	0	0	0	0	0	0	0	2	2	0	14	0	0	0	0	0	0	0	
ru.sbrf.bh.agreement.pinsagreement	2	0	0	0	0	1	0	0	7	15	0	58	0	0	0	5	0	0	0	
ru.sbrf.bh.agreement.corporateagreem	2	1	1	1	0	1	0	0	20	80	0	321	0	0	0	100	0	0	0	
ru.sbrf.bh.agreement.corporateagreem	3	0	0	0	0	1	0	0	3	7	0	4	0	0	0	2	0	0	0	
ru.sbrf.bh.agreement.pbrokerage	2	2	2	2	0	1	0	0	12	38	0	365	0	0	0	6	0	0	0	
ru.sbrf.bh.agreement.pbrokerage.ibpbr	3	0	0	0	0	1	0	0	3	14	0	24	0	0	0	6	0	0	0	
ru.sbrf.bh.agreement.pbrokerage.mobi	3	0	0	0	0	1	0	0	3	14	0	24	0	0	0	6	0	0	0	
ru.sbrf.bh.agreement.loan	2	1	1	1	0	1	0	0	7	10	0	98	0	0	0	3	0	0	0	
ru.sbrf.bh.agreement.loan.branchlist	3	0	0	0	0	1	0	0	3	3	0	16	0	0	0	3	0	0	0	
ru.sbrf.bh.banking	1	9	183	1	0	2	0	0	6	10	0	10	0	0	0	0	0	0	0	
ru.sbrf.bh.banking.operation	2	18	69	2	0	0	0	0	6	6	0	28	0	0	0	0	0	0	0	
ru.sbrf.bh.banking.operation.branchca	3	0	0	0	0	0	0	0	10	14	0	66	0	0	0	14	0	0	0	
ru.sbrf.bh.banking.operation.earlyloanr	3	1	1	0	0	0	0	0	8	24	0	36	0	0	0	51	0	0	0	
ru.sbrf.bh.banking.operation.earlyloanr	4	0	0	0	0	0	0	0	3	11	0	8	0	0	0	4	0	0	0	
ru.sbrf.bh.banking.operation.caccount	3	4	6	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
ru.sbrf.bh.banking.operation.caccount	4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
ru.sbrf.bh.banking.operation.caccount	4	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
ru.sbrf.bh.banking.operation.caccount	5	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
ru.sbrf.bh.banking.operation.caccount	4	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
ru.sbrf.bh.banking.operation.caccount	5	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
ru.sbrf.bh.banking.operation.caccount	4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
ru.sbrf.bh.banking.operation.caccount	3	3	5	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
ru.sbrf.bh.banking.operation.caccount	4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
ru.sbrf.bh.banking.operation.caccount	4	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	

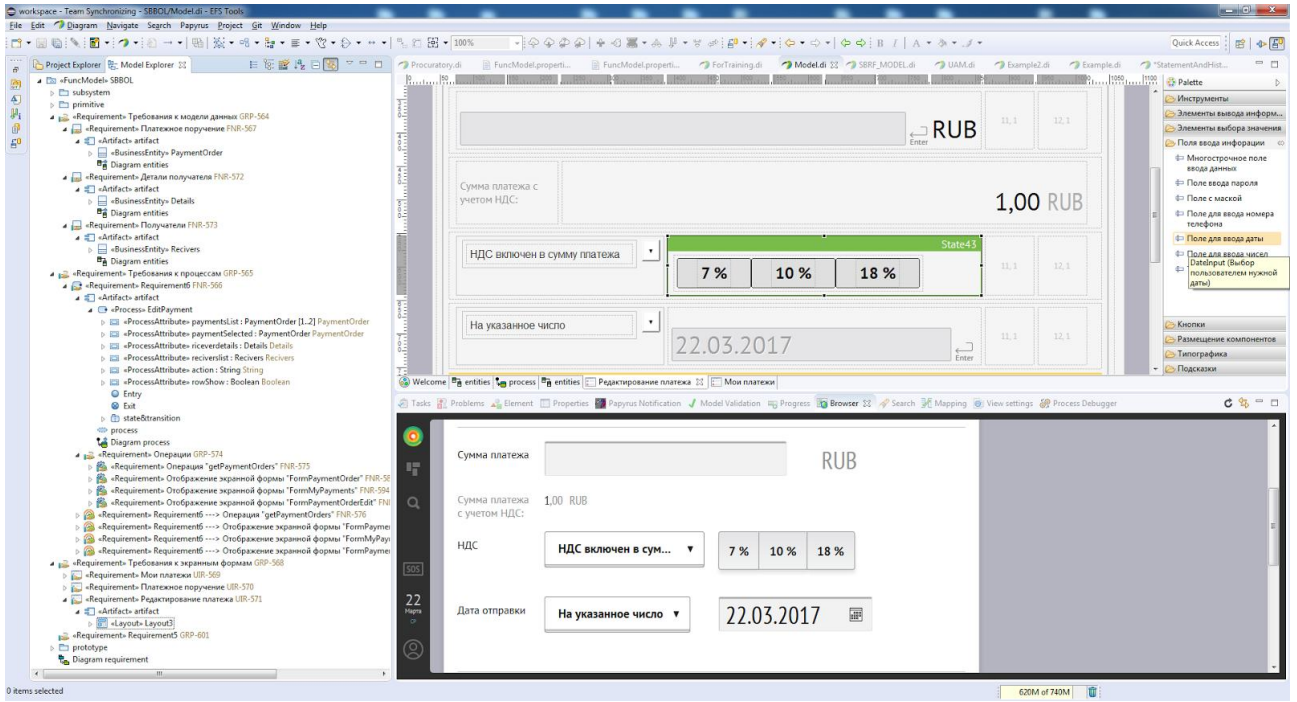
App modules statistic

RAD: как разрабатывать без технического долга?

RAD (от англ. rapid application development — быстрая разработка приложений) — концепция создания средств разработки программных продуктов, уделяющая особое внимание быстроте и удобству программирования, созданию технологического процесса, позволяющего программисту максимально быстро создавать компьютерные программы.

Для применения требуется:

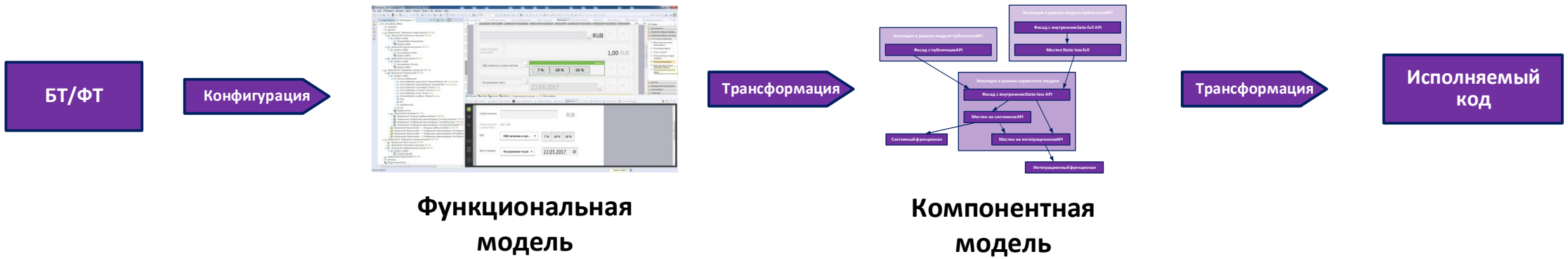
- Низкая вычислительная сложность
- Тиражируемая архитектура
- Зрелость системного функционала



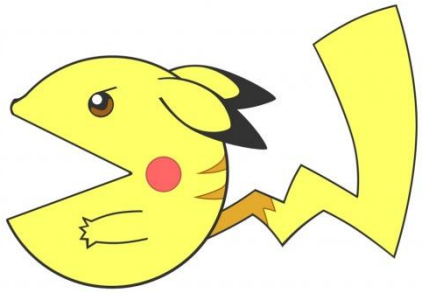
Имплементация RAD

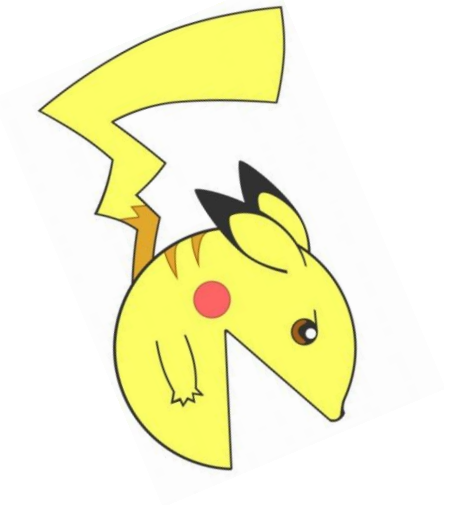
Применяется для конфигурации:

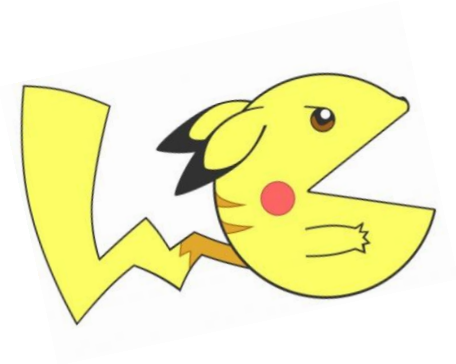
- Объектов данных и транспортных объектов
- Системных процессов
- Визуальных форм
- Точек интеграции
- Режима использования система функционала



Спасибо! Вопросы?







Слекеничс Андрей
AYSlekenichs.SBT@sberbank.ru
Сбербанк-Технологии