csg

DEVOPS & MODERNIZATION

AN ENGINEERING EXCELLENCE STORY 2.0

Presented by @ScottPrugh

SCOTT PRUGH CHIEF ARCHITECT & SVP, SOFTWARE ENGINEERING

we are CSG

OPTIMIZE **MONETIZE** **REVOLUTIONIZE**

2.0

65 MILLION SUBSCRIBERS

8 BILLION TRANS/M

100+ GLOBAL DEVOPS TEAMS

24 COUNTRIES

50+ APPS

20+ TECHSTACKS

REVENUE MANAGEMENT

DIGITAL MONETIZATION

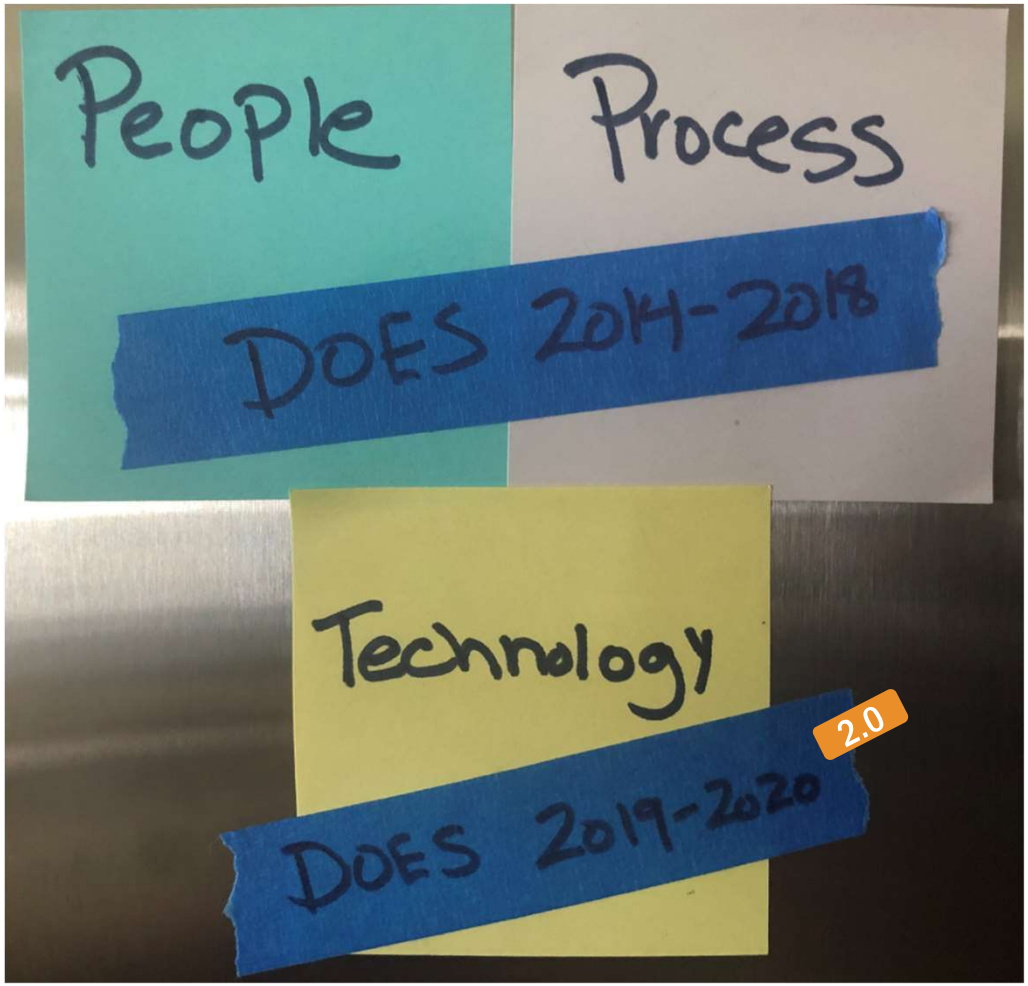
CUSTOMER COMMUNICATION MANAGEMENT

PAYMENT GATEWAY SERVICES

THANKS

#gladtobehere

Shape a Better World!



RECAP: DEVOPS JOURNEY IN METRICS

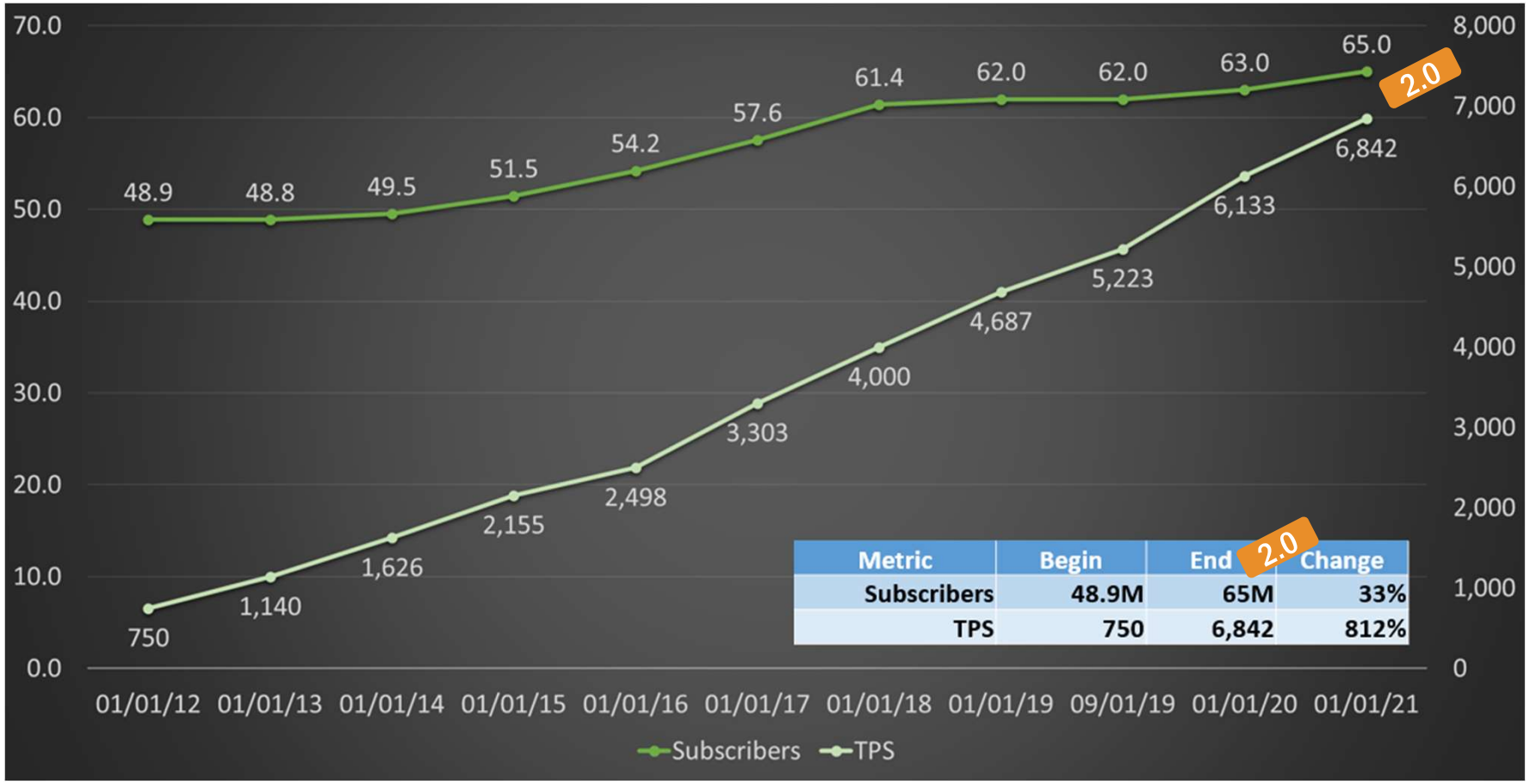
	Begin	2018		
Release Impact	507	85	-83%	Agile: 2012-2016
Incidents/mo.	1640	427	-74%	DevOps: 2016+
Subscribers	48.9M	62M	27%	Agile + DevOps: 2012-2018
TPS	750	4,000	433%	
Impact Minutes	22,932	9,481	-58%	2017-* Lean Portfolio Leadership PM Meets DevOps
Release On Demand	<5%	28%	460%	
eNPS	4	20	400%	

Stability and Growth Enabled via Agile/Lean/DevOps

PROBLEM: ENABLING SPEED, GROWTH & COST MANAGEMENT

2010

PROBLEM: ENABLING SPEED, GROWTH & COST MANAGEMENT



Metric	Begin	End 2.0	Change
Subscribers	48.9M	65M	33%
TPS	750	6,842	812%

THE PROBLEM WITH "LEGACY" SYSTEMS

HOW DO YOU MAINTAIN, PATCH AND SECURE?

HOW DO COMBAT EXTERNAL THREATS & MARKET FORCES?

HOW DO YOU INCREASE STABILITY AND SAFETY?

HOW DO YOU GO FASTER AND DELIVER FEATURES?

HOW DO YOU SUPPORT GROWTH WITHOUT A MASSIVE INCREASE IN COST?

HOW DO YOU MINIMIZE EXPOSURE FROM HOSTILE VENDORS?

— DON'T BE LEGACY. BE HERITAGE. MODERNIZE.

GREAT ENGINEERING COMPANIES ALLOCATE TIME FOR MODERNIZATION

MODERNIZATION FUELS DEVOPS:

CREATE SAFETY & REDUCE TECHNICAL DEBT

IMPROVE: PRODUCTIVITY, QUALITY, LEAD TIME, RECOVERY

REDUCE RISK: LEGACY TECH, HOSTILE VENDORS, WORKFORCE

THE JOB OF MODERNIZATION IS NEVER DONE

APPLICATION MODERNIZATION REALITIES

2.0

65M SUBSCRIBERS

1B TRANS/DAY

75M BILLS/MONTH

$87B/YEAR CUSTOMER REVENUE PROCESSED

APPLICATION MODERNIZATION: KEY CAPABILITIES

How do you approach a massive application modernization?

Feature Switching	Add switches to swap between new and old code.
Code Porting	Port or retarget the code. Assist porting with automation intelligence.
Incremental Rollout	Use feature switches to canary test the rollout. Rollback fast. Small batches.
Strangulation	Strangle off the legacy technology and then cut it out.

Automated Testing	Implement prolific modern test coverage. Do you know how it works today?
Continuous Integration	Version Control. Automate build and deploy. Create fast feedback.
Telemetry	Instrument the code. Make it visible. Understand how production behaves.
Infrastructure	Remove proprietary infrastructure. Self service. Infra as code. Cloud.

Very carefully with great perseverance and engineering excellence....

FOUNDATIONAL PITFALLS 2.0

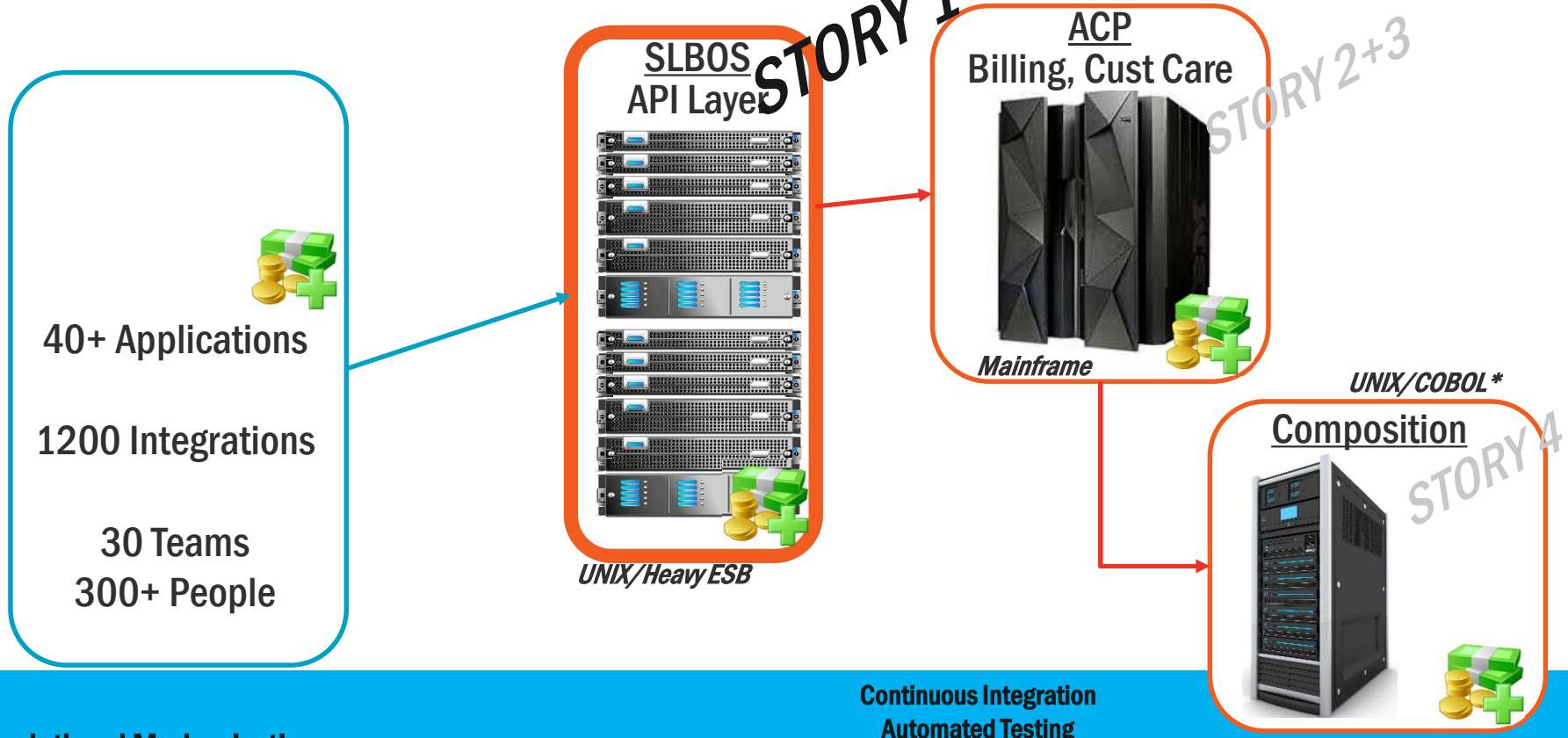
THE BI-MODAL TRAP

THE REWRITE TRAP
(AKA: MICROSERVICES WON'T SAVE YOU)

TECH DEBT & GIVING AWAY YOUR PIVOT
(IT'S A LONG ROAD WITH LOTS OF OUTSIDE FORCES)

Automated Testing	Implement prolific modern test coverage. Do you know how it works today?
Continuous Integration	Version Control. Automate build and deploy. Create fast feedback.
Telemetry	Instrument the code. Make it visible. Understand how production behaves.
Infrastructure	Remove proprietary infrastructure. Self service. Infra as code. Cloud.

APPLICATION MODERNIZATION: 4 STORIES



STORY 1: GOLF COURSE SOFTWARE

"Low-Code"

"No Developers!"

"Just map your data!"

"Easy to operate"

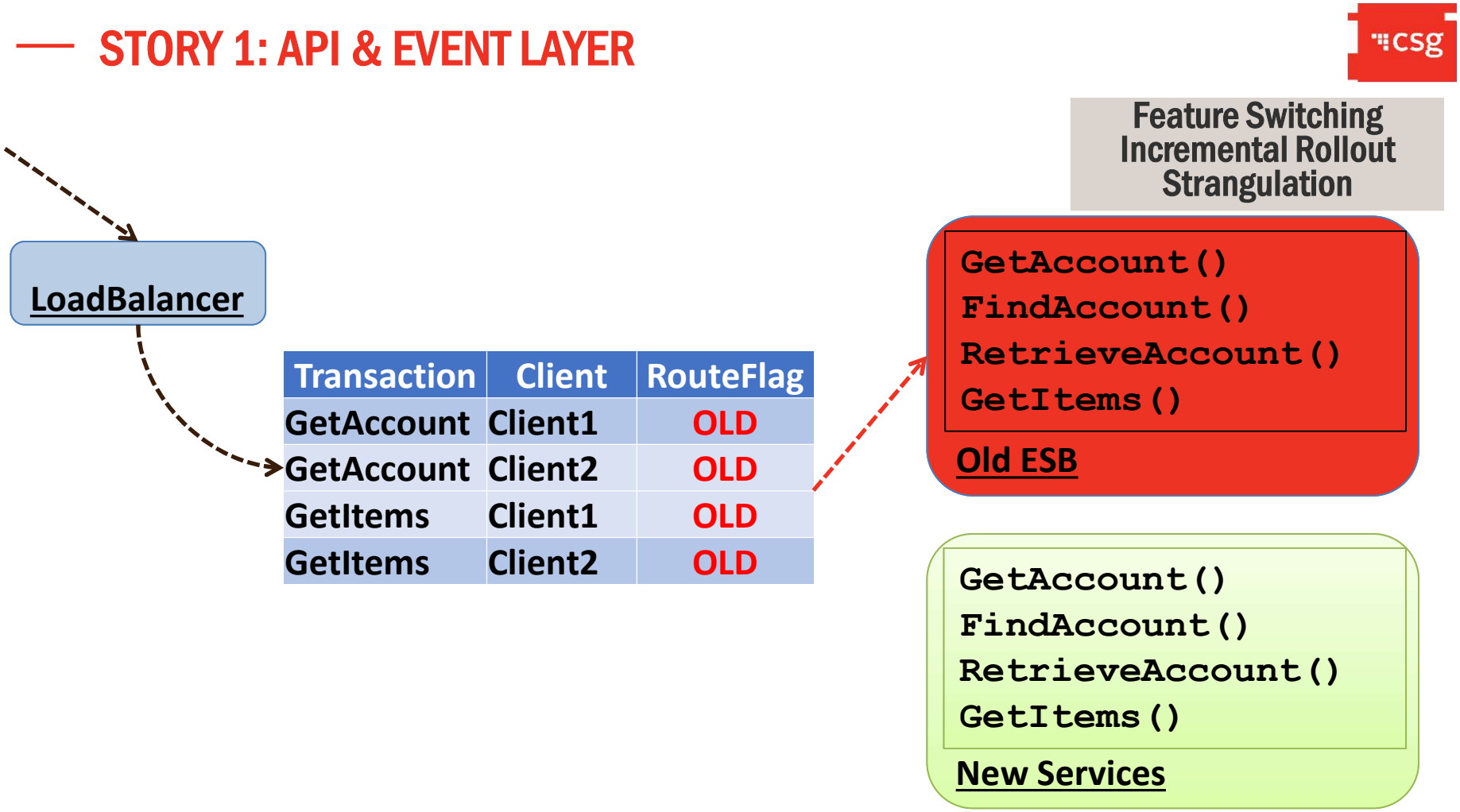
"Already integrated"

Don't do this to your people! Let them pick their own tools!

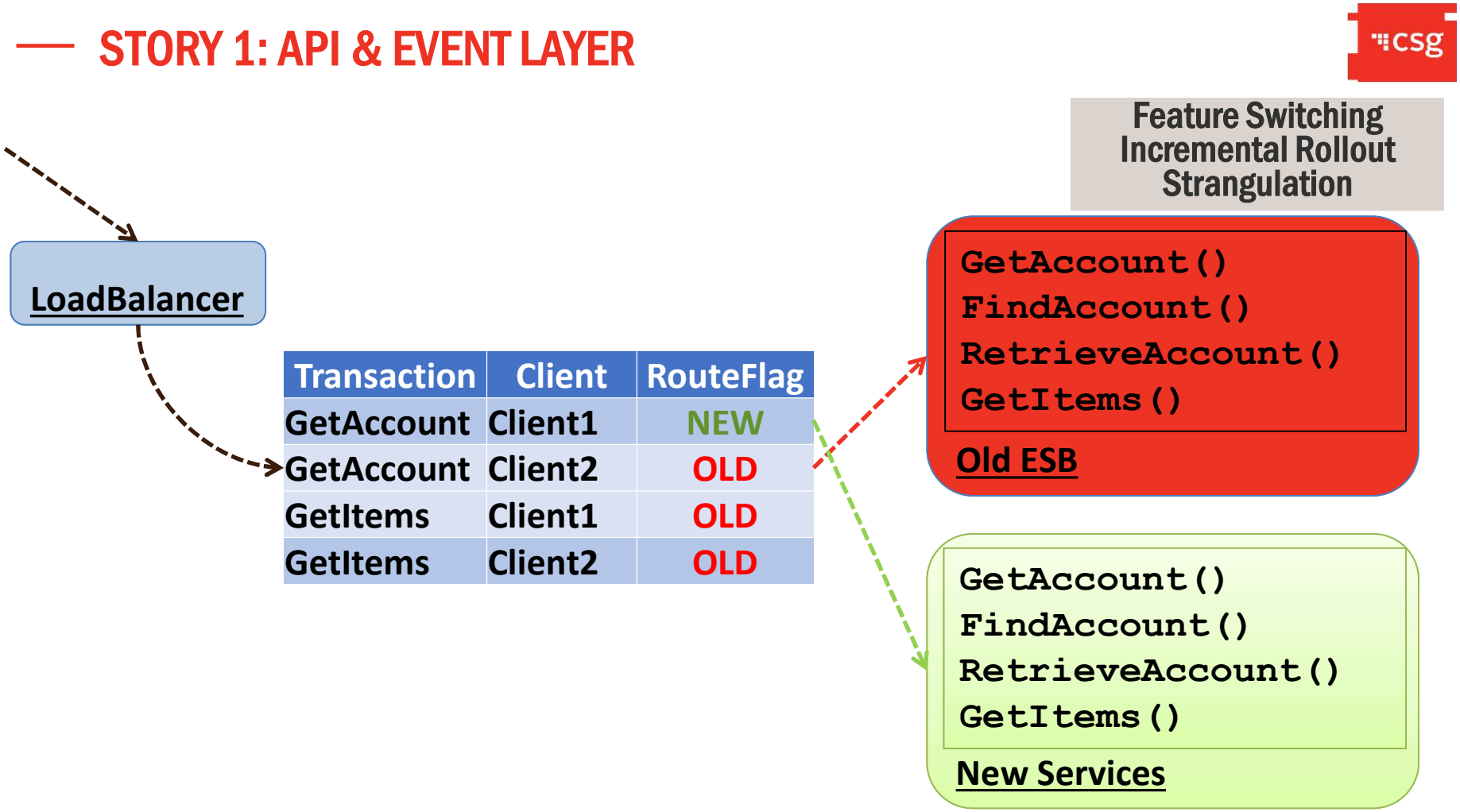
STORY 1: API & EVENT LAYER

Problem	Poor developer aesthetics: Low-code, “point and click..” High build and test effort: 14-hour builds, 4 weeks of test Poor operational aesthetics: massive deploys, 45min recycles, low observability Low TPS density Unsustainable cost to support business growth
Approach	Move platform to commodity stack Port 300 transactions leveraged by 1200 integrations to native code Strangle old platform off: feature flags and canary Apply Foundational Modernization: Testing, CI, Telemetry, Infrastructure

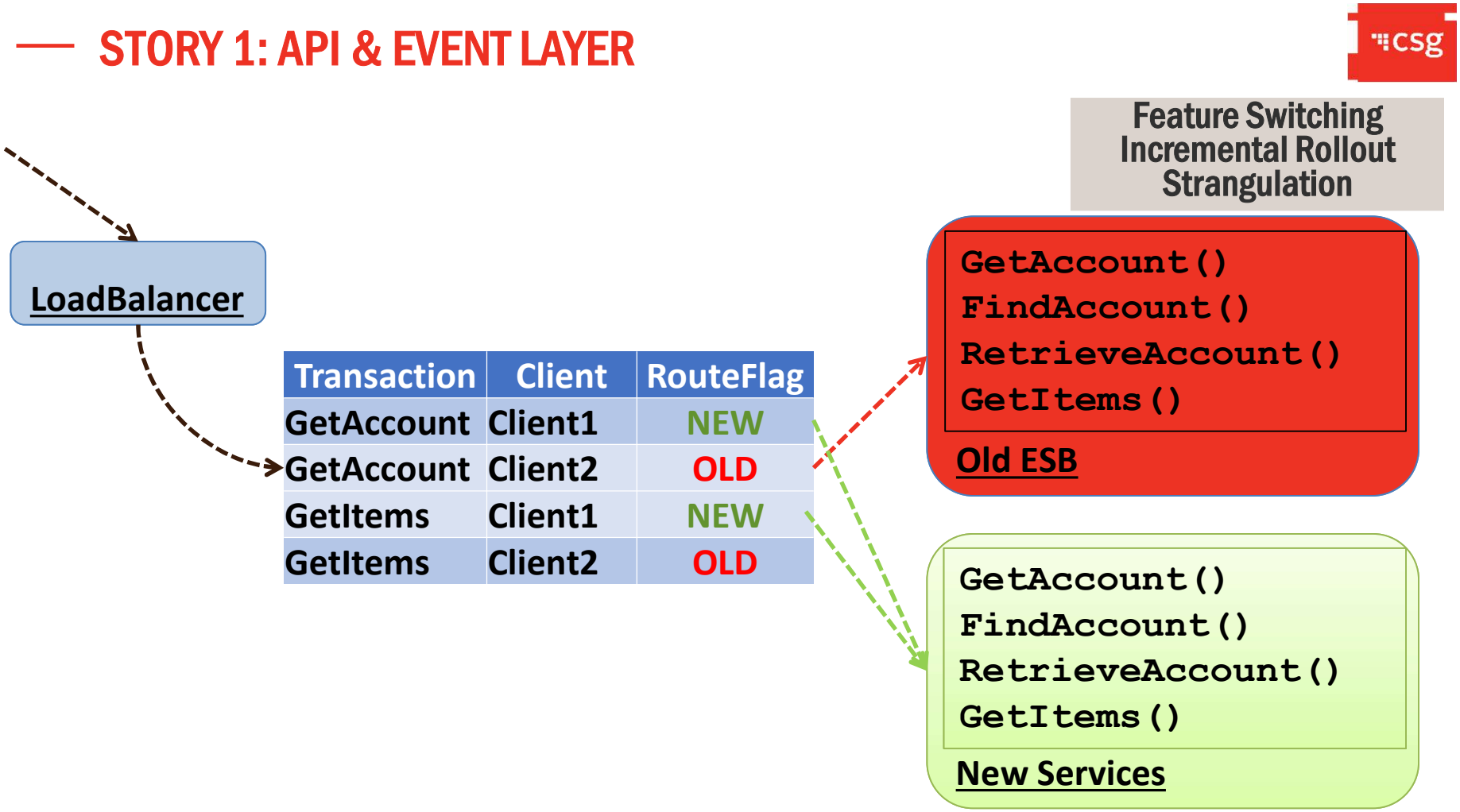
STORY 1: API & EVENT LAYER



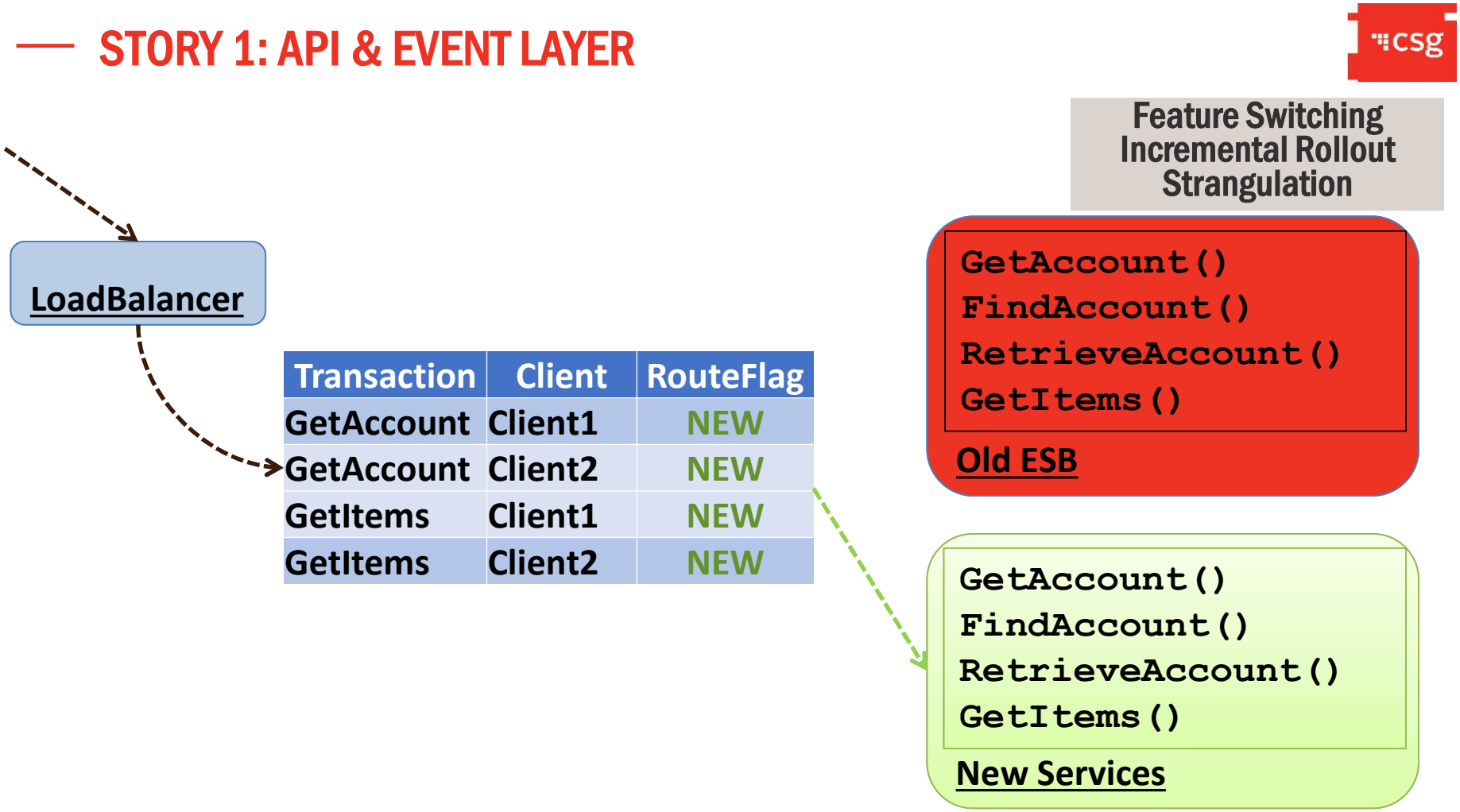
STORY 1: API & EVENT LAYER



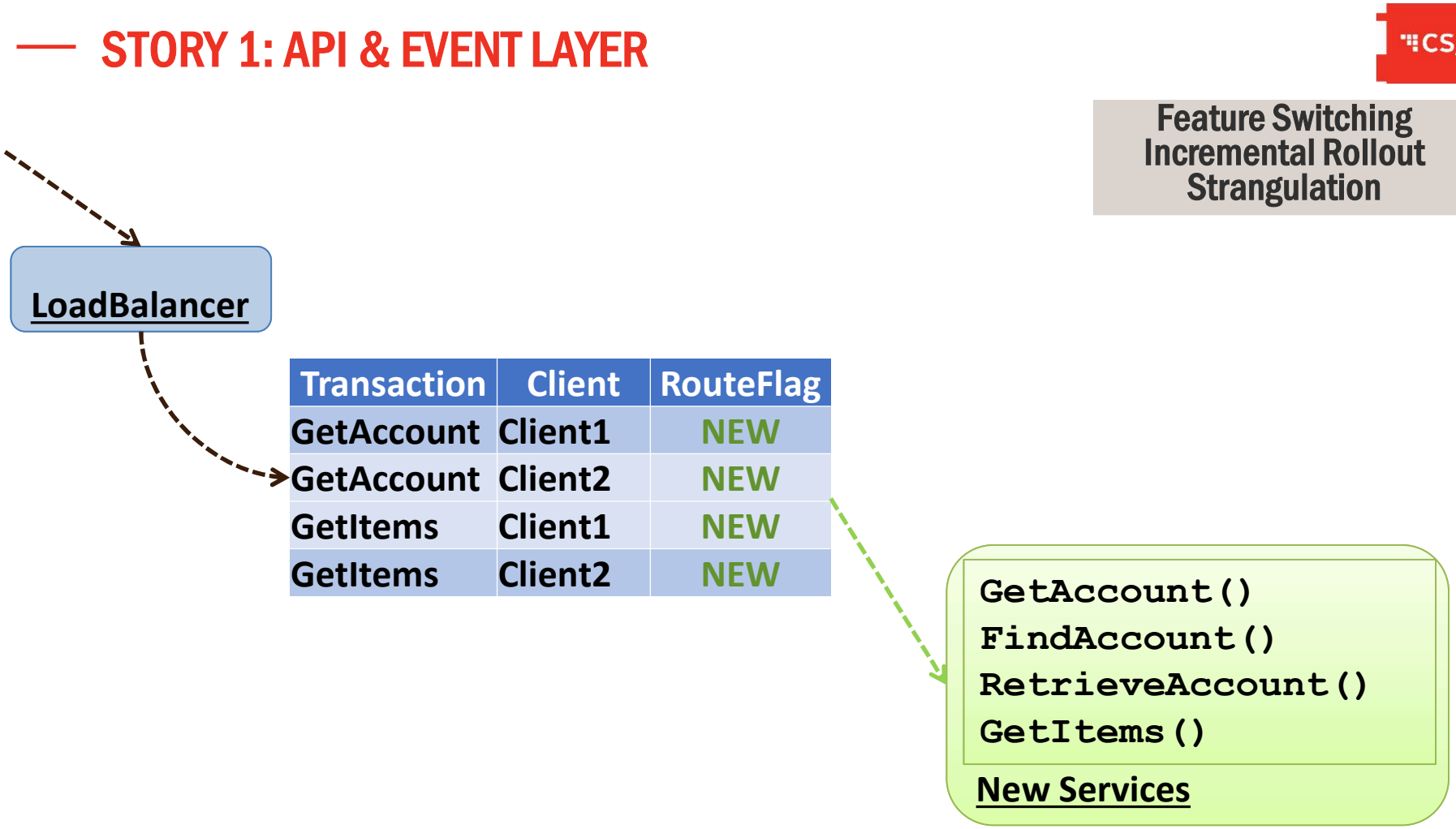
STORY 1: API & EVENT LAYER



STORY 1: API & EVENT LAYER



STORY 1: API & EVENT LAYER



FOUNDATIONAL MODERNIZATION: AUTOMATED TESTING

Automated Testing

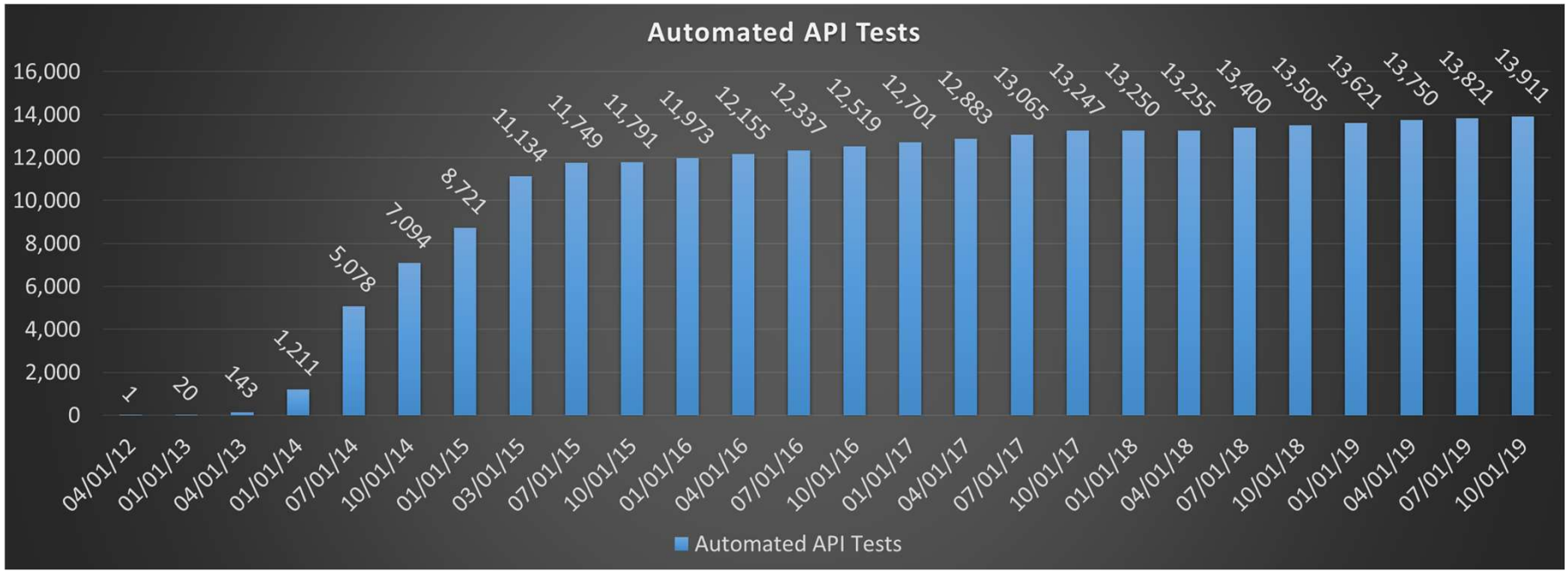
- FindAccount_DB2_2_26.feature
- GetAccountNegative_DB2.feature
- GetAccountDetail.feature
- FindAccount_WCF.feature
- FindAccountVoice_WCF.feature
- FindAccountNegative_WCF.feature
- Google15_2.feature
- UpdateAccount.feature
- RetrieveAccountVoice.feature
- RetrieveAccountMediacomNegative.fea
- RetrieveAccountMediacom.feature
- RetrieveAccountFinancialsNegative.feat
- RetrieveAccountFinancialsKeys.feature
- RetrieveAccountFinancials.feature
- RetrieveAccountNegative.feature
- GetAccountNegative.feature
- RetrieveAccountKeys.feature
- AccountMissingRequired.feature
- RetrieveAccount.feature
- GetAccount.feature

```
@SlbosRegression @Account @FindAccount @AccountManagement
Feature: FindAccount Voice WCF
    In order to accurately find accounts and port to WCF
    As consumer of Slbos RetrieveAccount Transactions
    I want to validate WCF FindAccount Transactions are accurate

Scenario: User Defined Id 2.2_2.24
    Given PDB Query 'UDAN_Current' results index '1' tags 'key'
    When I send Xml from file
    | _Endpoint | _File       | RoundTrip | _ReturnCode | RoutingArea                                | KeyType | Key | Version |
    | WCF       | FindAccount | FAWCF22   | 0           | $FeatureDefaultValue[DefaultRoutingArea] | UD      | key | 2.2     |
    | WCF       | FindAccount | FAWCF224  | 0           | $FeatureDefaultValue[DefaultRoutingArea] | UD      | key | 2.24    |
    Then The diff between 'FAWCF22' and 'FAWCF224' should be
    | changetype | xpath              |
    | Change     | //head/Version     |
```

Before	After
Proprietary test tools only used by testers	Tests commoditized in Gherkin
High cost & increasing	Tests in version control with code
High manual test effort	Testers and developers collaborate on test suites

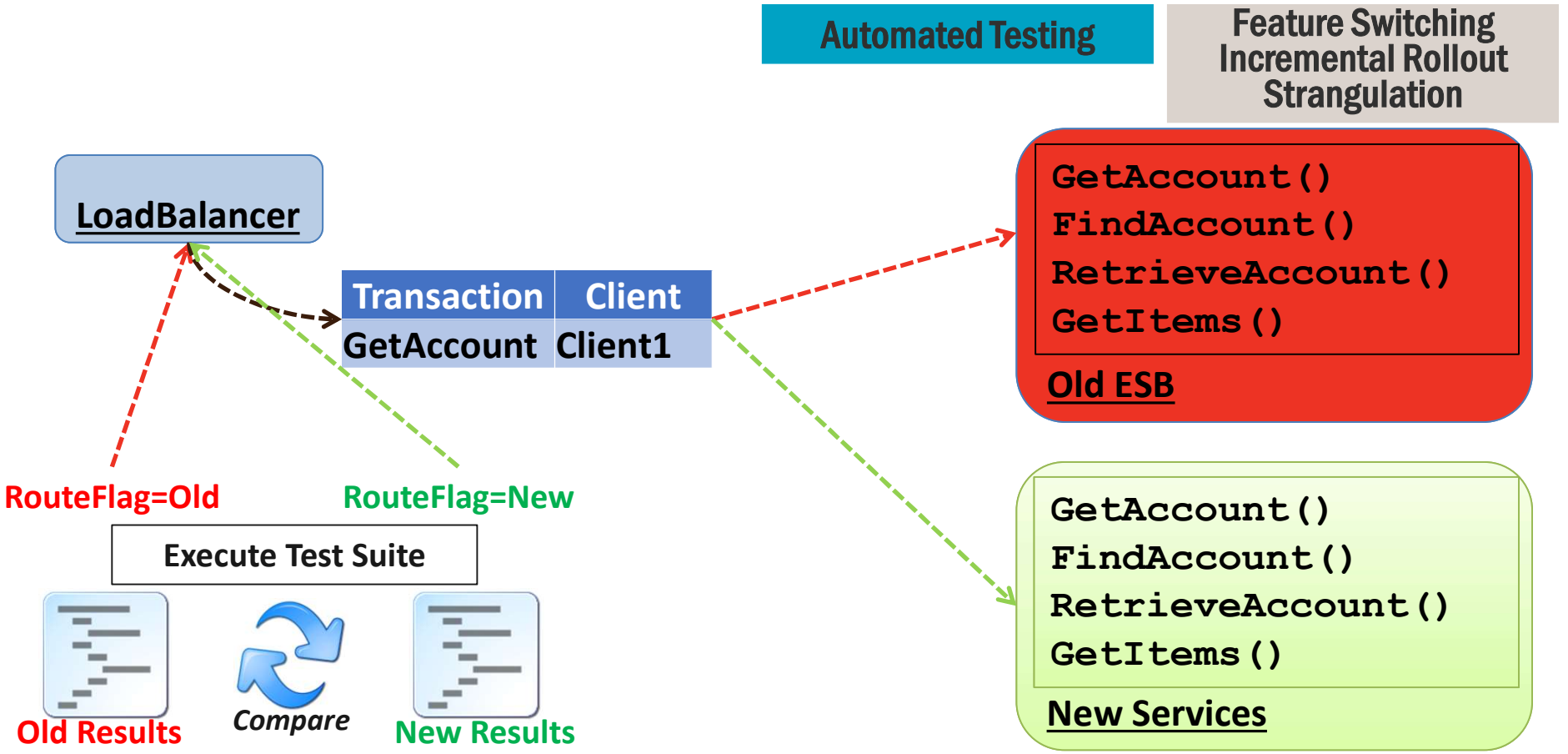
STORY 1: API PLATFORM TEST COVERAGE

Automated Testing



https://itrevolution.com/devops-resource-legacy-code/

STORY 1: API & EVENT LAYER



STORY 1: API & EVENT LAYER RESULTS

Metric	Before	After
Required TPS	6,000	6,000
Max TPS/Node	215	1,333
Nodes Required	28	4.5
$/TPS	$7,306	$31
Feature Development	15%	55%
Build + Deploy	14hr	5min
Server Recycle	45m	2m
Automated Tests	0	13,911

https://itrevolution.com/devops-resource-legacy-code/

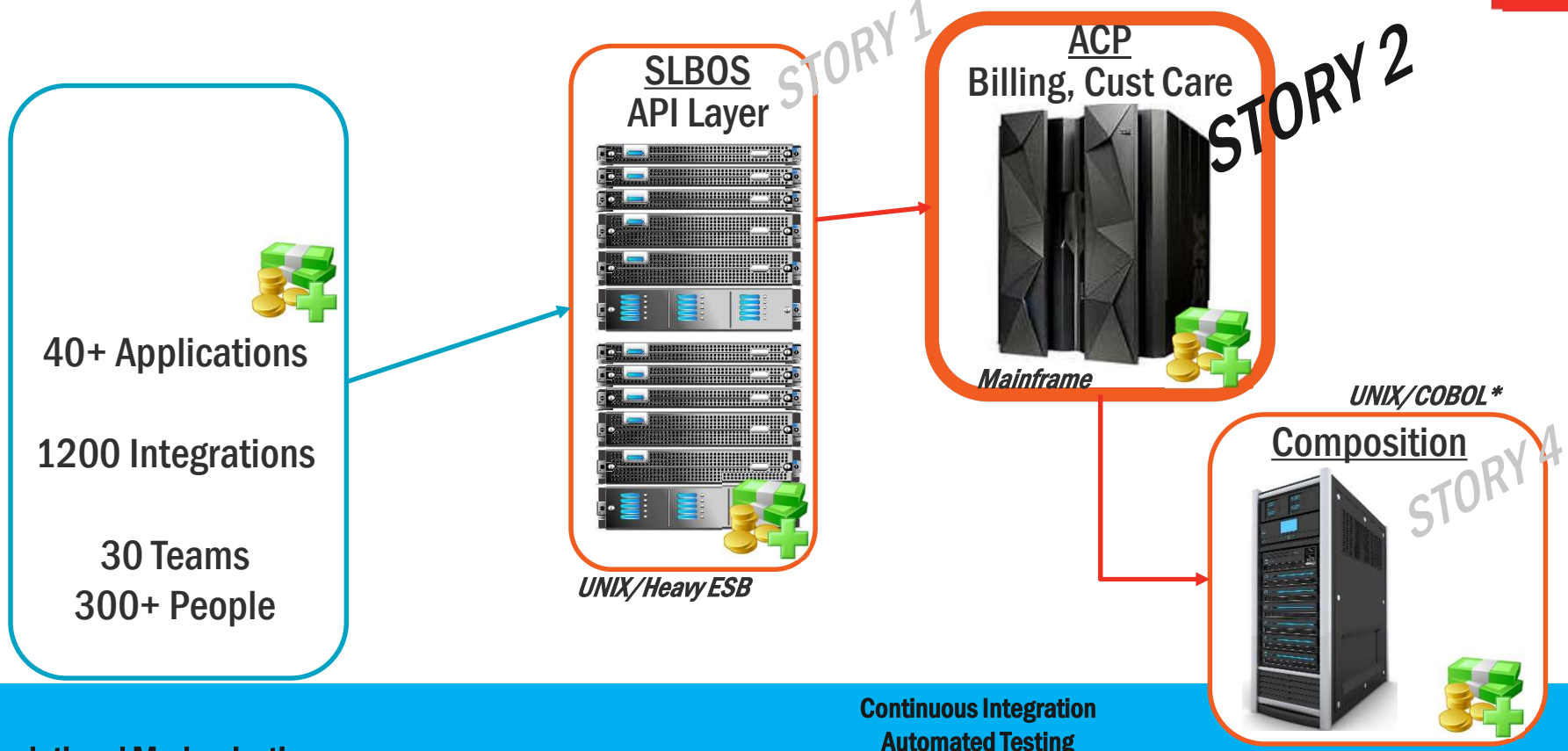
STORY 1: API & EVENT LAYER THANKS

STORY 1: API & EVENT LAYER CELEBRATION

"Luck favors the prepared." – Edna Mode

"Goodbye, you awful pile of 1990's 8U garbage!!"
-Scott, CSG Parking Lot
-Wes, The Unicorn Project

STORY 2: MAINFRAME DB2



STORY 2: MAINFRAME DB2

Problem	Lack of commodity data access(Access via CICS only) Maintainability at risk: Productivity, Agility, Workforce Sustainability Unsustainable cost increases jeopardizing viability
Approach	Convert VSAM master files into 500+ DB2 tables Incremental rollout via: Feature switching and VSAM transparency Strangle off VSAM datastore and VSAM subsystems Offload **READ** transactions from CICS to direct DB2 queries

STORY 2: MAINFRAME DB2

Feature Switching
Datastore Edition

PORTING DATASTORES IS HARDER THAN CODE...

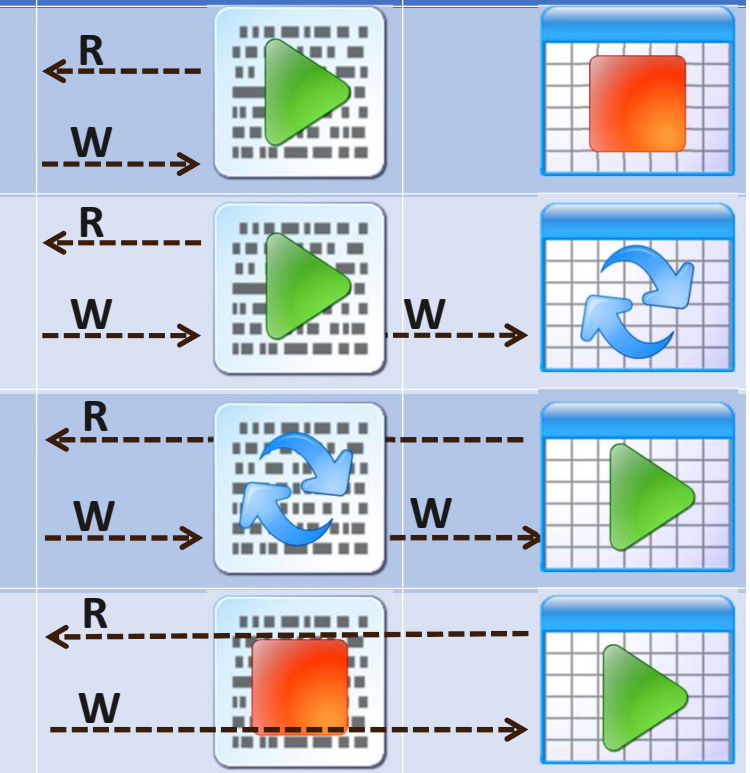
DATASTORE MIGRATION PATTERN

1. Old datastore is primary.
2. Old datastore is primary, new datastore is replica. Backfill. Compare. 2.0
3. New datastore is primary, old datastore is replica. Compare.
4. New datastore is primary.

THIS PATTERN CAN BE USED FOR MIGRATING ANY TYPE OF DATASTORE

STORY 2: MAINFRAME DB2

Data Switch	Behavior	VSAM	DB2
SW1	VSAM only	R, W	
SW2	VSAM: RW DB2: W Rollback to SW1	R, W	W
SW3	DB2: RW VSAM: W Rollback to SW1, SW2	R, W	W
SW4	DB2 Only	R, W	

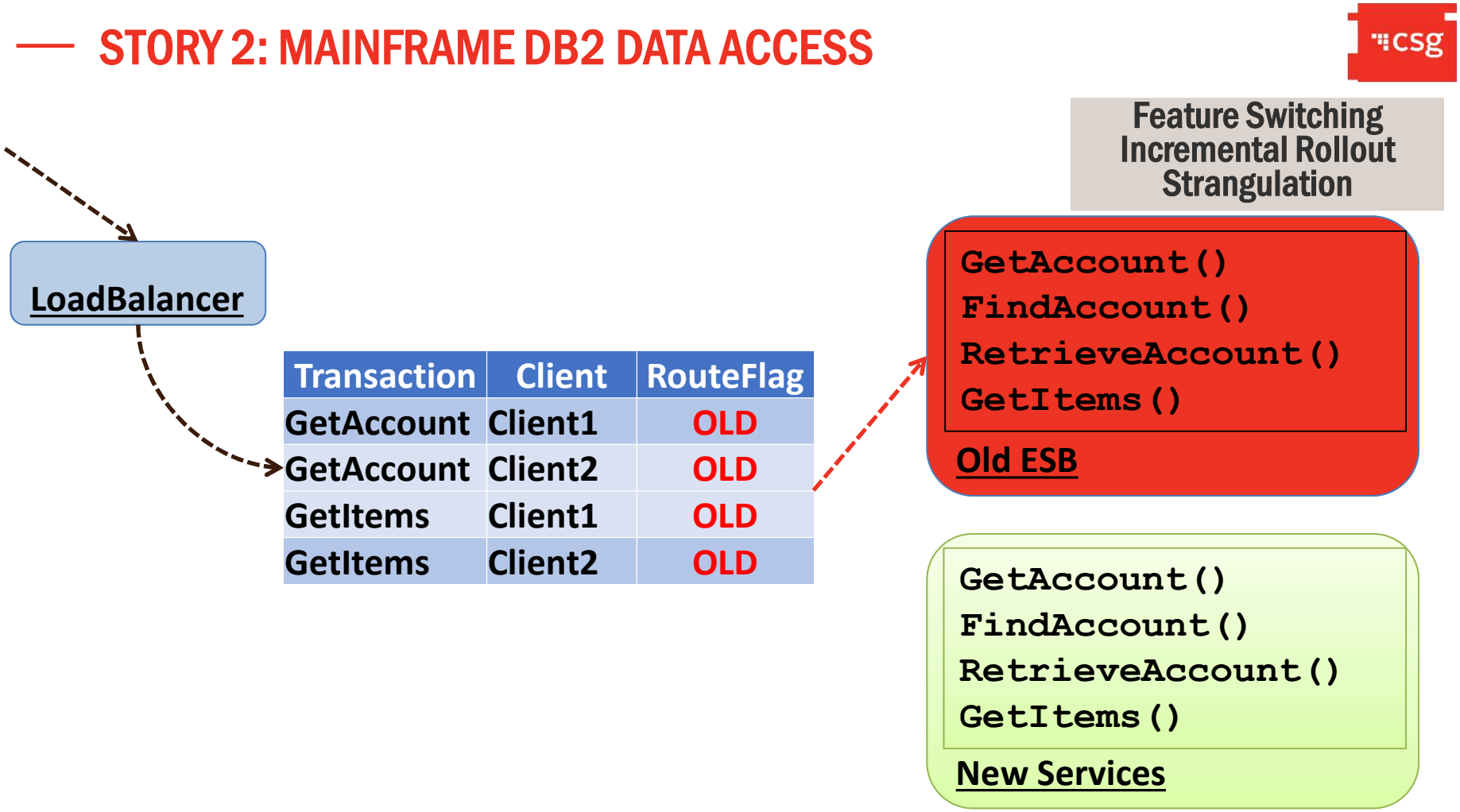


Feature Switching Datastore Edition

Nightly batch compares

Find bad and dirty data!

STORY 2: MAINFRAME DB2 DATA ACCESS

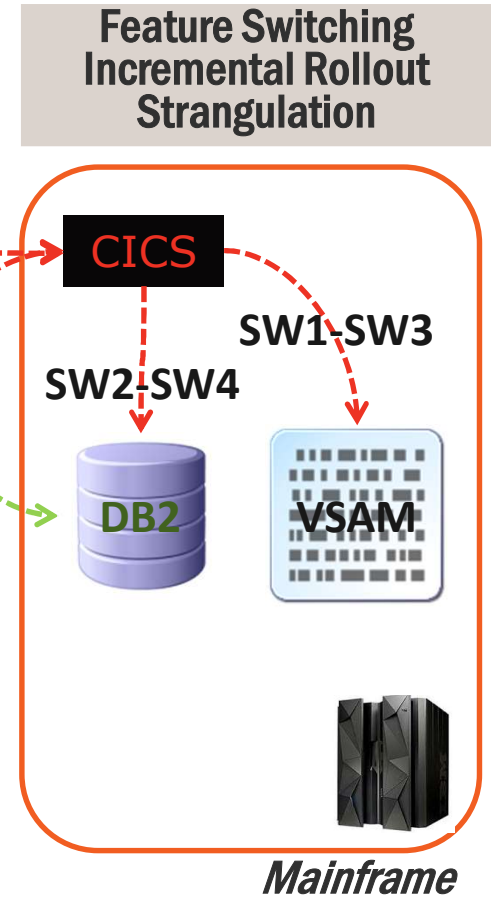


STORY 2: MAINFRAME DB2 DATA ACCESS

```
GetAccount()
FindAccount()
RetrieveAccount()
GetItems()
```

New Services

Transaction	Client	ReadFlag
GetAccount	Client1	CICS
GetAccount	Client2	DB2
GetItems	Client1	DB2
GetItems	Client2	CICS

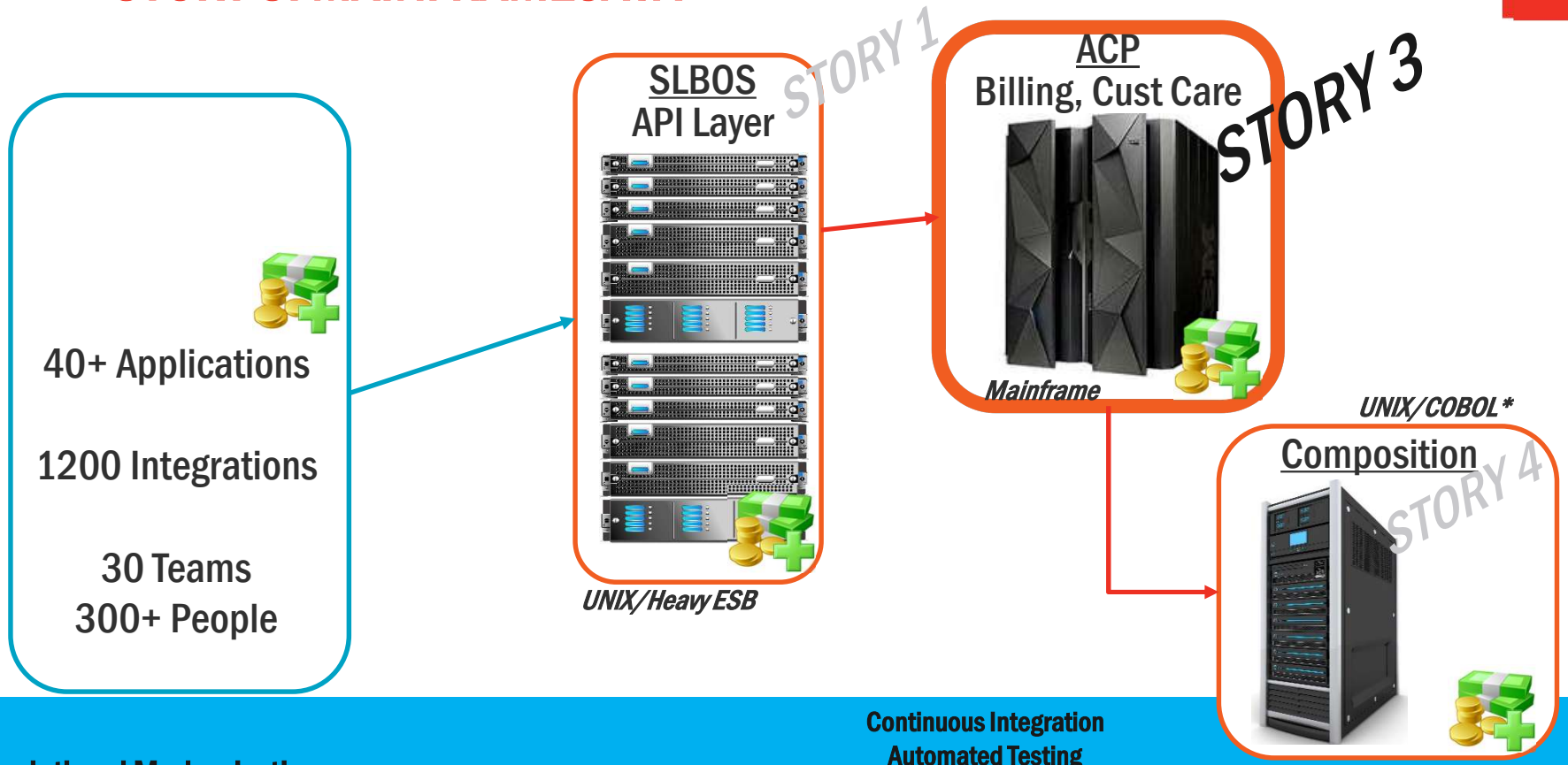


STORY 2: MAINFRAME DB2 RESULTS

Metric	Before	After
Data Access	VSAM via CICS	DB2 direct and CICS
Read Transactions	100% CICS	62% direct DB2
Data Accessibility	Low	High
Average Response	40ms	25ms(38% better)

Near zero customer impact

STORY 3: MAINFRAME JAVA

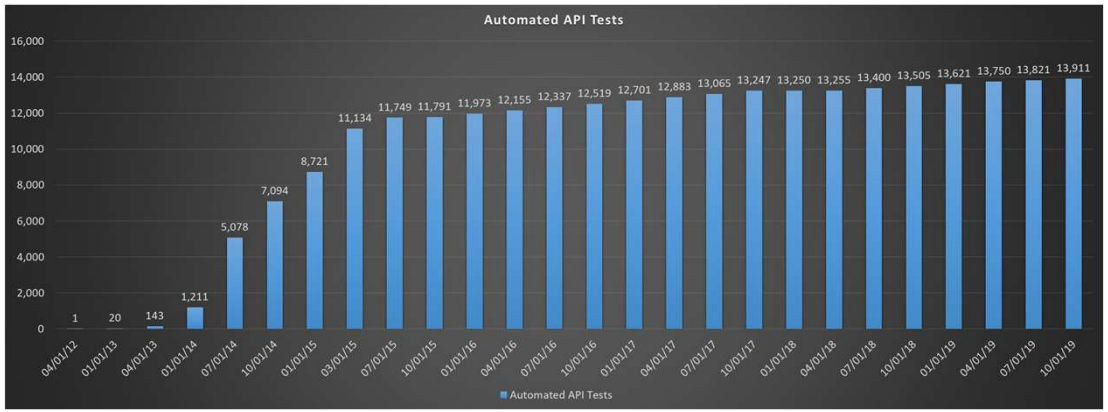


STORY 3: MAINFRAME JAVA

Problem	3.7M lines of HLASM Maintainability at risk: Productivity, Agility, Workforce Sustainability Unsustainable cost increases jeopardizing viability
Approach	Cross compiler tooling: HLASM->Java Target more complex **UPDATE** logic All converted code in CI with full test coverage All Java code can run ***OFFBOARD*** or on mainframe Incremental rollout via: Feature switching. Deploys during the day! Strangle off HLASM subsystems

STORY 3: MAINFRAME JAVA: FUNCTIONAL TEST COVERAGE

Cover entire legacy code base(HLASM) with both breadth and depth

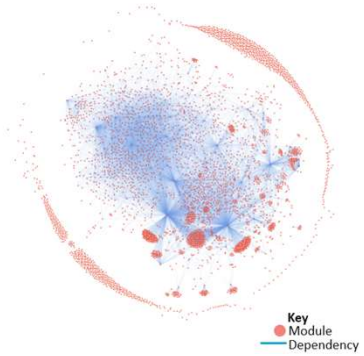


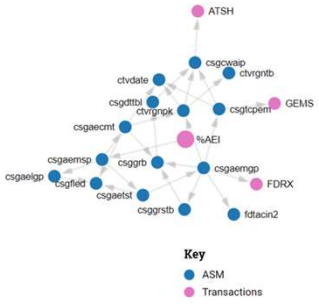
Automated Tests=13,911

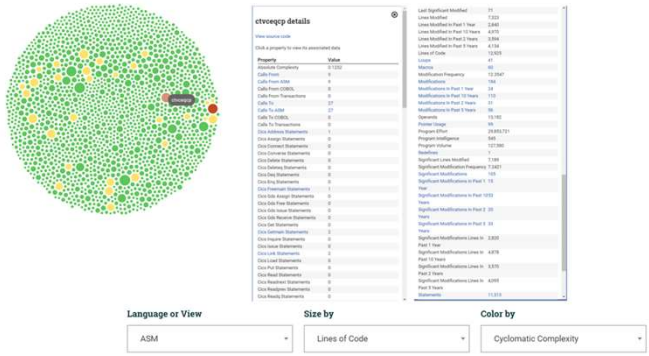
STORY 3: MAINFRAME JAVA: CODE ANALYSIS

Functional Test Coverage → **Code Analysis** → Cross Compilation → Version Control & CI → Unit Testing → Refactor Cross Compilation → Functional Test Coverage

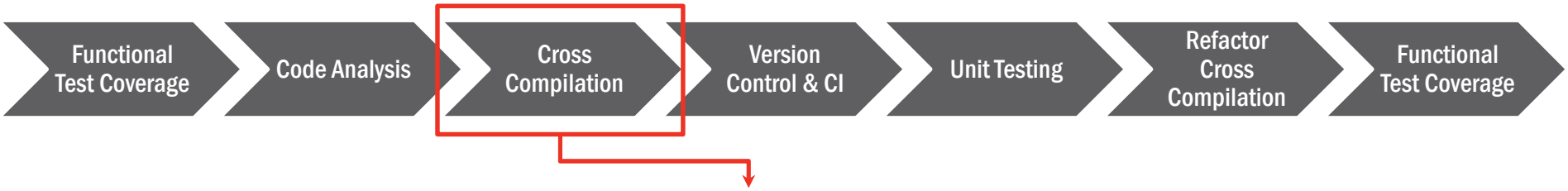
Leverage Code Analysis to Better Understand Dependencies



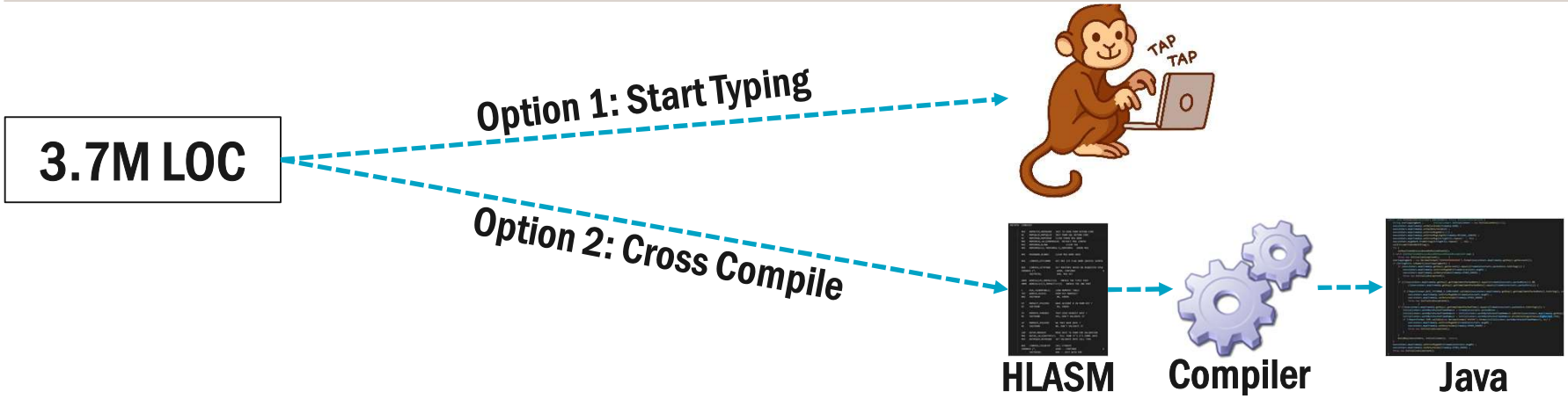




STORY 3: MAINFRAME JAVA: CROSS COMPILATION



Leverage Cross Compilation to Mass Migrate, Codify Patterns & Refactor Learnings

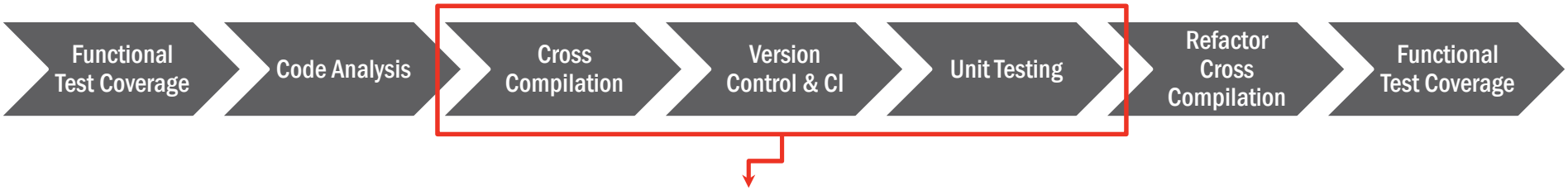


STORY 3: MAINFRAME JAVA: CROSS COMPILATION

```
INITRTN  CSMBEGRT

         MVI   MOPRETCD,MOPRGOOD   INIT TO GOOD PARM RETURN CODE
         XC    MOPSQLCD,MOPSQLCD   INIT PARM SQL RETURN CODE
         XC    MOPERRAD,MOPERRAD   CLEAR ERROR MSG ADDR
         MVC   MOPERRLN,=AL2(MOPMSGLN)  DEFAULT MSG LENGTH
         MVI   MOPERMSG,BLANK              CLEAR THE
         MVC   MOPERMSG+1(L'MOPERMSG-1),MOPERMSG   ERROR MSG

         MVC   MSG98WRK,BLANKS     CLEAR MSG WORK AREA

         BAS   LINKREG,GTFLG000    GET DB2 I/O FLAG ADDR (BATCH) 14207A

         BAS   LINKREG,SETRTNAD    SET ROUTINES BASED ON REQUESTED VIEW
         CBRANCH (*,               GOOD, CONTINUE                    X
               INITR970)           BAD, MSG SET

         UNPK  WORK16(14),MOPACCT(8)   UNPACK THE FIRST PART
         UNPK  WORK16+13(3),MOPACCT+7(2)   UNPACK THE 2ND PART

         L     R14,=A(NUMTABLE)    LOAD NUMERIC TABLE
         TRT   WORK16,0(R14)       PARM KEY NUMERIC?
         BNZ   INITR850            NO, ERROR

         CP    MOPACCT,PCKZERO     HAVE ACCOUNT # IN PARM KEY ?
         BZ    INITR900            NO, ERROR

         CP    MOPDATE,P4NINES     THEY USED HIGHEST DATE ?
         BE    INITR400            YES, DON'T VALIDATE IT

         CP    MOPDATE,PCKZERO     DO THEY HAVE DATE ?
         BE    INITR400            NO, DON'T VALIDATE IT

         ZAP   DAT9P,MOPDATE       MOVE DATE TO PARM FOR VALIDATION
         MVC   DATIN,=AL2(DATFMT17)   TELL TEHM IT'S 9'S COMPL DATE
         MVI   DATREQCD,DATREQVD   SET VALIDATE DATE CALL TYPE

         BAS   LINKREG,CALDATEP    CALL CTVDATE
         CBRANCH (*,               GOOD -- CONTINUE                  X
               INITR950)           BAD -- EXIT WITH ERR
```

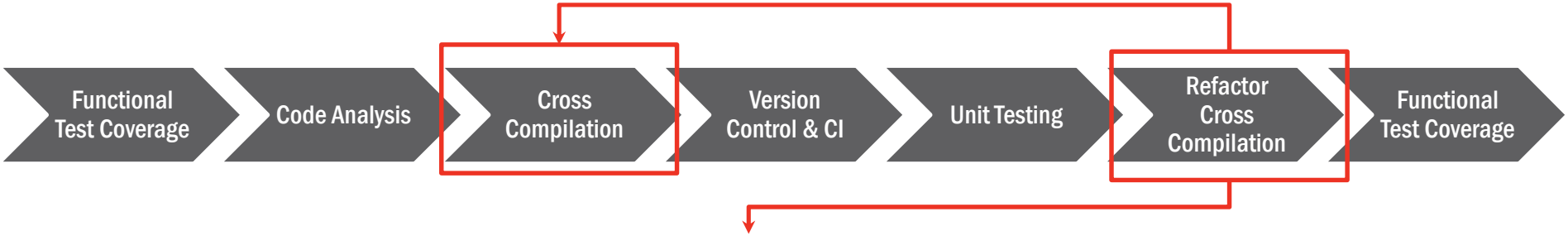
```
public void initialize(ExecuteVars executeVars) throws InitializeException4 {
    String overlappingWork ;        InitializeVars initializeVars = new InitializeVars(null);
    executeVars.mopCtvmemlp.setReturnCode(Ctvmemlp.GOOD) ;
    executeVars.mopCtvmemlp.setSqlReturnCode(0) ;
    executeVars.mopCtvmemlp.setErrorMsgAddr(null) ;
    executeVars.mopCtvmemlp.setErrorMsgLength(Ctvmemlp.MESSAGE_LENGTH) ;
    executeVars.mopCtvmemlp.setErrorMsg(StringUtils.repeat(" ", 39)) ;
    executeVars.msg98wrk.fromString(StringUtils.repeat(' ', 28)) ;
    callCtvcomitToGetDb2IOFlag();
    try {
        setRoutineAddressesBasedOnPassedViewId();
    } catch (SetRoutineAddressesBasedOnPassedViewIdException4 exp) {
        throw new InitializeException4();       }
    overlappingWork = new DecimalFormat("000000000000000").format(executeVars.mopCtvmemlp.getKey().getAccount());
    if (StringUtils.isNumeric(overlappingWork)) {
        if (executeVars.mopCtvmemlp.getKey().getAccount().equals(CtvmemluConstants.packedZero.toString())) {
            executeVars.mopCtvmemlp.setErrorMsgAddr(CtvmemluConstants.msg02) ;
            executeVars.mopCtvmemlp.setReturnCode(Ctvmemlp.OTHER_ERROR) ;
            throw new InitializeException4();
        }
        if ((!(executeVars.mopCtvmemlp.getKey().getComplimentPackedDate().equals(CtvmemluConstants.packedNines))) &&
                (!(executeVars.mopCtvmemlp.getKey().getComplimentPackedDate().equals(CtvmemluConstants.packedZero)))) {

            if (!ReportFormat.DATE_YYYYMMDD_9_COMPLEMENT.validate(executeVars.mopCtvmemlp.getKey().getComplimentPackedDate().toString(), exe
                executeVars.mopCtvmemlp.setErrorMsgAddr(CtvmemluConstants.msg07) ;
                executeVars.mopCtvmemlp.setReturnCode(Ctvmemlp.OTHER_ERROR) ;
                throw new InitializeException4();
            }          }
        if (!(executeVars.mopCtvmemlp.getKey().getComplimentPackedTime().equals(CtvmemluConstants.packedZero.toString()))) {
            initializeVars.work4BytePackedTimeHhmmsst = CtvmemluConstants.packedNines ;
            initializeVars.work4BytePackedTimeHhmmsst = initializeVars.work4BytePackedTimeHhmmsst.subtract(executeVars.mopCtvmemlp.getKey()
            initializeVars.work4BytePackedTimeHhmmsst = initializeVars.work4BytePackedTimeHhmmsst.divideToIntegralValue(BigDecimal.TEN);
            if (!ReportFormat.TIME.validate(new DecimalFormat("000000").format(initializeVars.work4BytePackedTimeHhmmsst), 6)) {
                executeVars.mopCtvmemlp.setErrorMsgAddr(CtvmemluConstants.msg08) ;
                executeVars.mopCtvmemlp.setReturnCode(Ctvmemlp.OTHER_ERROR) ;
                throw new InitializeException4();
            }
        }
        buildKey(executeVars, initializeVars);  return;
    }
    executeVars.mopCtvmemlp.setErrorMsgAddr(CtvmemluConstants.msg06) ;
    executeVars.mopCtvmemlp.setReturnCode(Ctvmemlp.OTHER_ERROR) ;
    throw new InitializeException4();
}
```

STORY 3: MAINFRAME JAVA: CROSS COMPILATION



All Cross Compiled Code Gets Foundational Version Control, CI, Unit Tests

STORY 3: MAINFRAME JAVA: CROSS COMPILATION



Continually Refactor Cross Compilation:

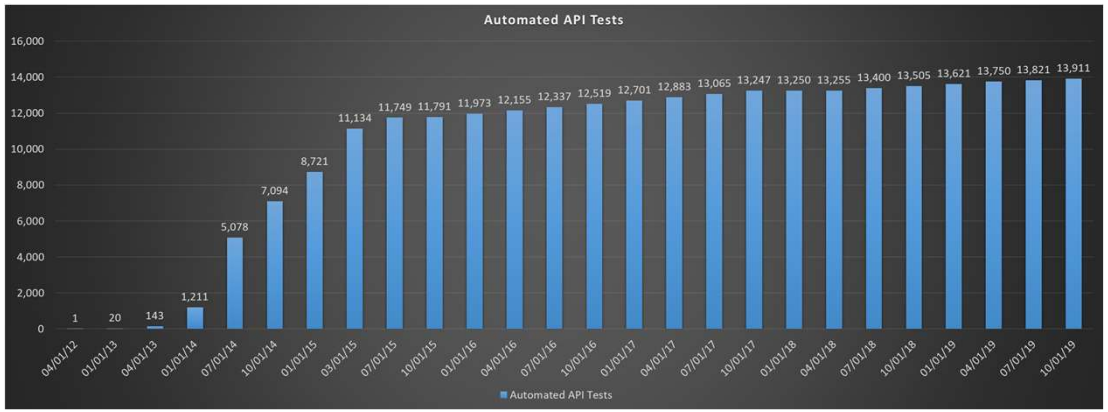
- Recognize domain specific patterns
- Increase target code base maintainability

File	Original Name	Generated Name	File Ty	Generated Ty	View Name
ci#table	I#CALTC2	BASE_CUST_NUMBER_VALIDATION	Copy	Constant	2018-03-22 memo
ci#table	I#CALTC3	RETURN_CURRENT_OL_REG_ID	Copy	Constant	2018-03-22 memo
ci#table	I#CALTC5	ENHANCED_CAMPAIGN_CALL_TYPE	Copy	Constant	2018-03-22 memo
ci#table	I#CALTC9	RETURN_CYCLE_INFORMATION	Copy	Constant	2018-03-22 memo
ci#table	I#CALTCN	CUSTOMER_NUMBER_VALIDATION	Copy	Constant	2018-03-22 memo
ci#table	I#CALTD5	RETURN_NEXT_DAY_OL_REG_ID	Copy	Constant	2018-03-22 memo
ci#table	I#CALTE8	CONVERGENT_XREF	Copy	Constant	2018-03-22 memo
ci#table	I#CALTE9	GENERATE_CHECK_DIGIT	Copy	Constant	2018-03-22 memo
ci#table	I#CALTF0	VAL_VISA_6_DIG_BINS_ICAS	Copy	Constant	2018-03-22 memo
ci#table	I#CALTF1	FILL_IN_TABLE_FOR_NUMBER	Copy	Constant	2018-03-22 memo
ci#table	I#CALTF2	FILL_IN_TABLEGEN_DUMMY	Copy	Constant	2018-03-22 memo
ci#table	I#CALTF3	GENERATE_DISPLAYS_ONLY	Copy	Constant	2018-03-22 memo
ci#table	I#CALTF4	GENERATE_CHECK_DIGET	Copy	Constant	2018-03-22 memo
ci#table	I#CALTF5	FILL_IN_TABLE_FOR_ICA_NUMBE	Copy	Constant	2018-03-22 memo

Naming Overrides

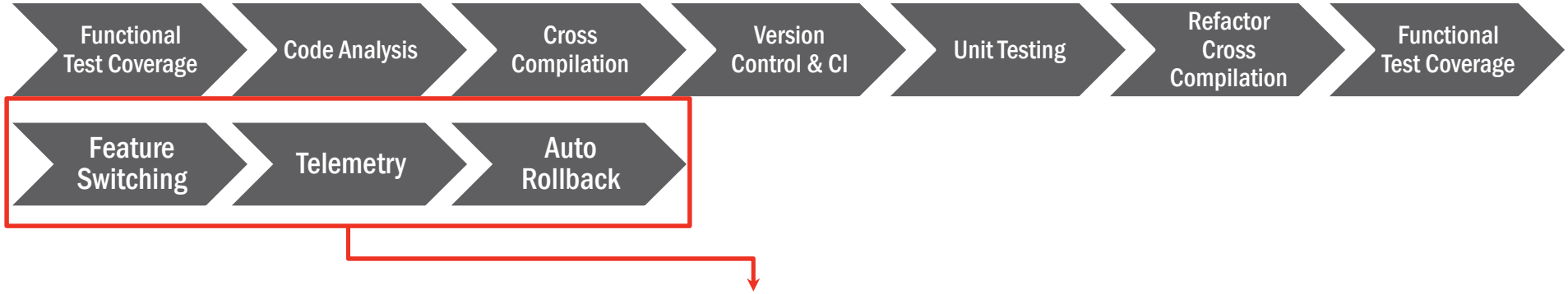
STORY 3: MAINFRAME JAVA: FUNCTIONAL TEST COVERAGE

Leverage Functional Coverage to Verify Congruent Behavior



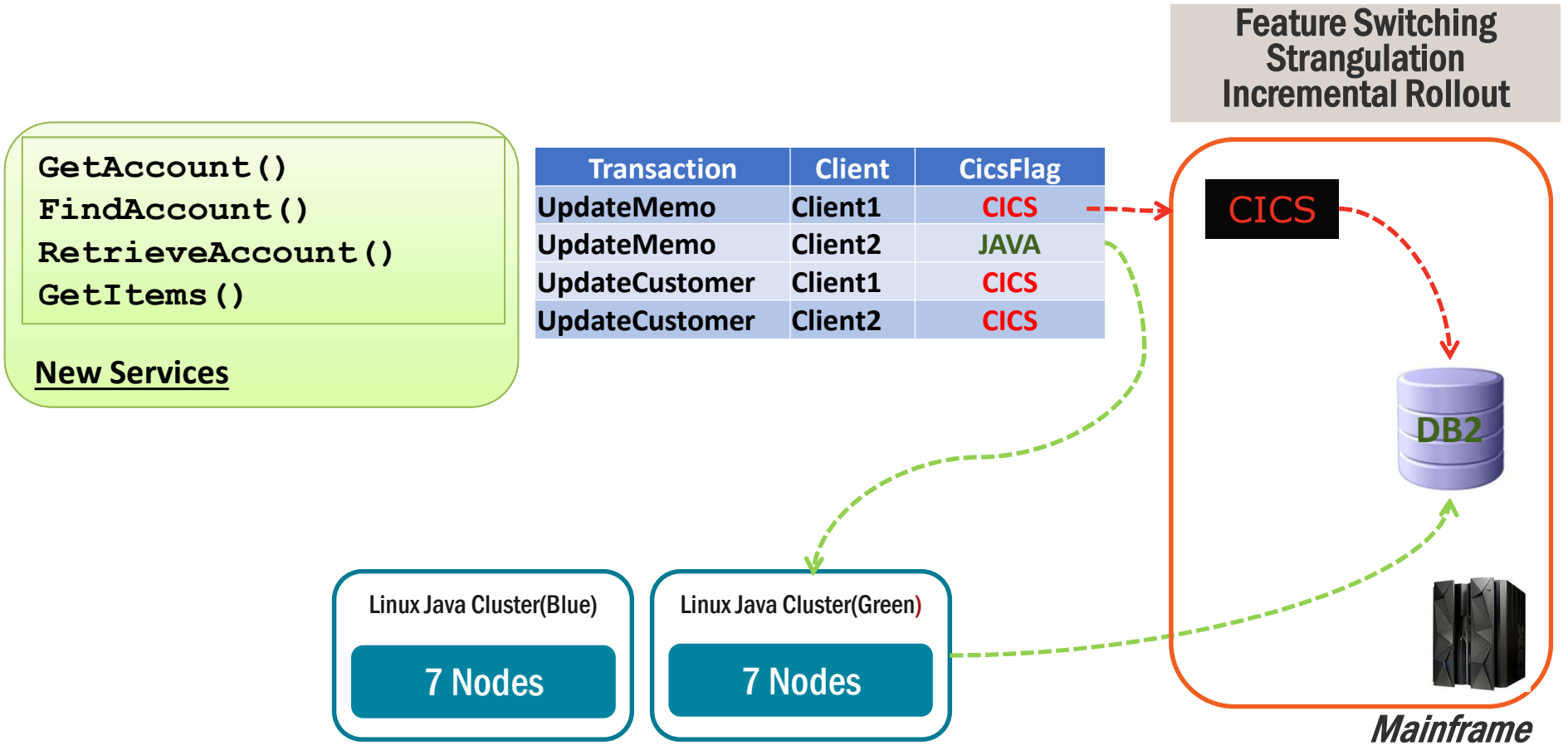
Automated Tests=13,911

STORY 3: MAINFRAME JAVA: PRODUCTION ROLLOUT

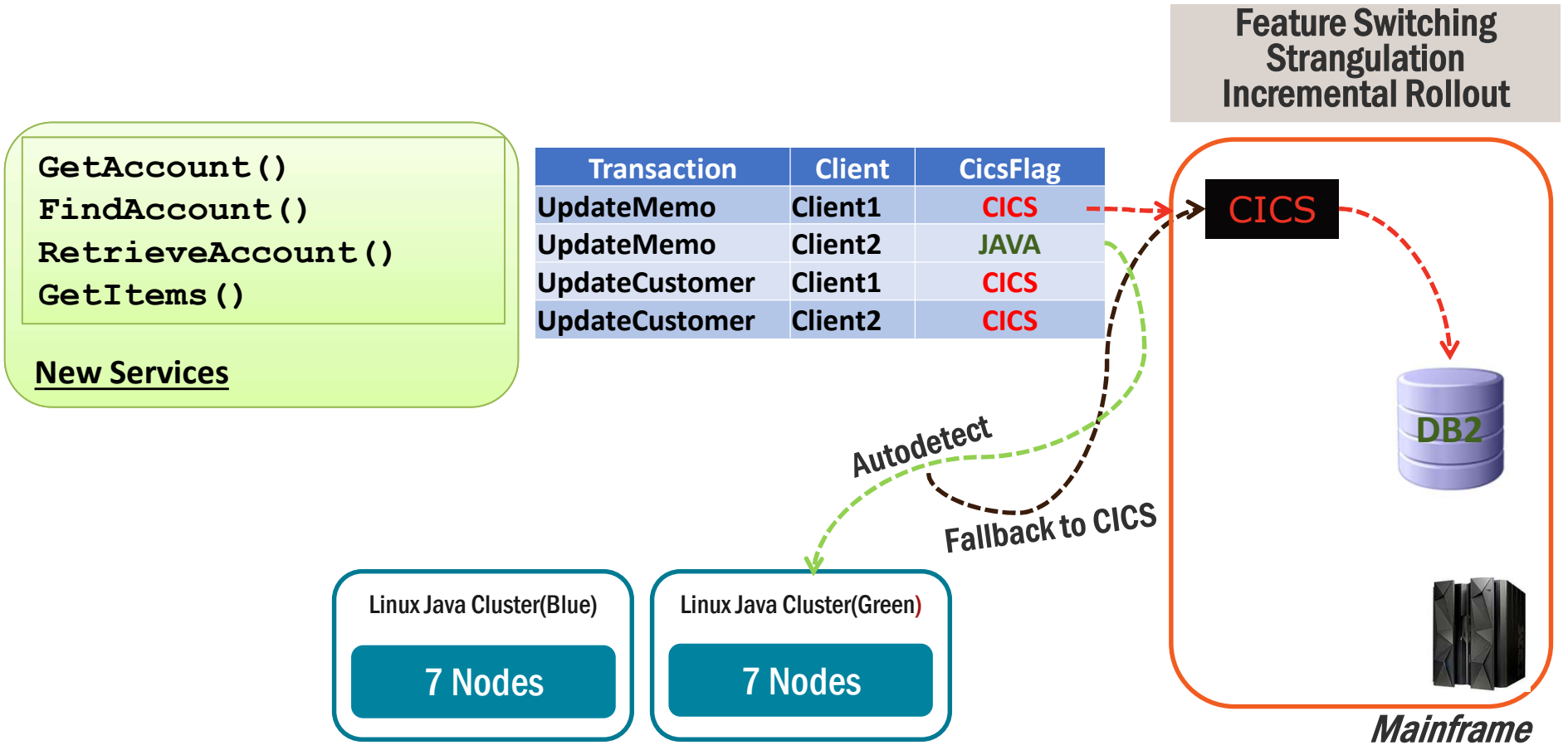


Leverage Feature Switching to Rollout and Rollback
Integrate Heavily to Production Telemetry
Detect Errors and Auto Rollback Upon Failure

STORY 3: MAINFRAME JAVA: PRODUCTION TOPOLOGY



STORY 3: MAINFRAME JAVA: PRODUCTION TOPOLOGY



STORY 3: MAINFRAME JAVA RESULTS

Metric	Before	After*
Online Code Base(3.7M)	100% HLASM	85% Java
Update Transactions	100% CICS	40% direct DB2
Maintainability	Hard	Supportable
Productivity	Low	Moderate
Telemetry	Low/Closed	Open/Common
Shared Practices/Tools	Low	High

Near zero customer impact

STORY 3: MAINFRAME DB2 & JAVA THANKS

STORY 4: COMPOSITION PLATFORM

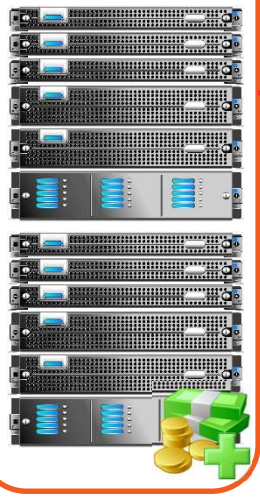
40+ Applications

1200 Integrations

30 Teams
300+ People

SLBOS
API Layer

STORY 1



UNIX/Heavy ESB

ACP
Billing, Cust Care

STORY 2+3

Mainframe

*UNIX/COBOL**

Composition

STORY 4

Foundational Modernization

Continuous Integration
Automated Testing
Telemetry
Infrastructure

STORY 4: PRINT PLATFORM

Problem	4M lines of Proprietary Cobol(25 years) without Version Control or CI Proprietary Unix & Proprietary Vendors No Unit or E2E Functional Tests No telemetry Lack of horizontal scale. Unaffordable vertical scale. Multiple impacting tickets/day
Approach	Add foundational CI, Cobol Unit Testing, E2E Functional Testing Add feature flags to support trunk-based development & incremental rollout Convert Proprietary UNIX & COBOL to Linux/GnuCobol Remove difficult/hostile vendors

STORY 4: PRINT PLATFORM CELEBRATION

STORY 4: PRINT PLATFORM RESULTS

2.0

95% converted to date:

4M lines proprietary COBOL to GNUCOBOL

Almost zero impacting tickets

Telemetry: doc retrieval, run time stats, job run time, errors

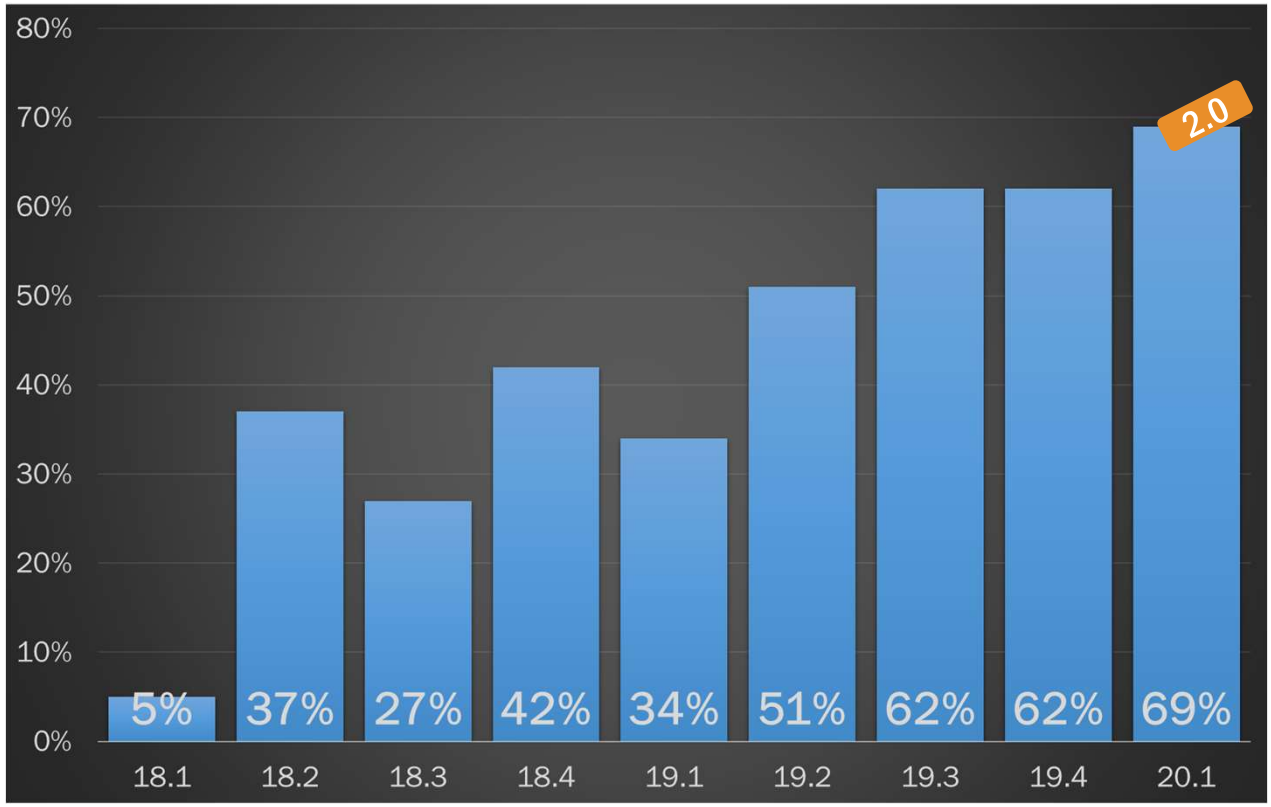
Commodity solution: Linux and GnuCobol

Hostile Vendor Leapfrog 2.0

	Before	After*
Incidents	Multiple/day	1 in 24 months
Release On Demand	<5%	100%
Lead Time	Months	Days

Near zero customer impact

CSG PROCESS UPDATE: RELEASE ON DEMAND(ROD)

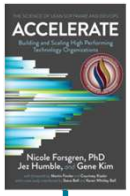


Modernization Improves
Lead Time
Deploy Frequency
Change Failure%
MTTR

ROD means releasing value when it is ready and decoupling it from large releases.

ROD is a forcing function(behavior shaping constraint) for high-performance behaviors and foundational modernization capabilities.

DEVOPS JOURNEY IN METRICS



CF%=Change Failure%
MTTR
LT=Lead Time
DF=Deploy Frequency

2.0

		Begin	2018		2019-2020 2.0	
CF%	Release Impact	507	85	-83%	13	-97% ↘
CF%	Incidents/mo.	1640	427	-74%	324	-80% ↘
	Subscribers	48.9M	62M	27%	65M	33% ↗
	TPS	750	4,000	433%	6,842	812% ↗
MTTR CF%	Impact Minutes	22,932	9,481	-58%	5,970	-74% ↘
LT DF	Release On Demand	<5%	28%	460%	69%	1280% ↗
	eNPS	4	20	400%	24	500% ↗
LT DF	Feature Cycle Time	249			56	-77% ↘

MODERNIZATION: WHAT WE LEARNED

YOU CAN MODERNIZE YOUR LEGACY APPLICATIONS. BE HERITAGE.

MODERNIZATION IS VITAL AND REQUIRES ENGINEERING EXCELLENCE

LEVERAGE CI, AUTOMATED TESTING, TELEMETRY, INFRASTRUCTURE 2.0

BE AWARE OF PITFALLS: BI-MODAL, RE-WRITE TRAP, PIVOT GIVEAWAY

OPTIMIZE FOR DEVELOPER AND OPERATIONS AESTHETICS

FEATURE SWITCHES, CODE PORTING, INCREMENTAL ROLLOUT, STRANGULATION

FUEL DEVOPS:

CREATE SAFETY & REDUCE TECHNICAL DEBT

IMPROVE: PRODUCTIVITY, QUALITY, LEAD TIME, RECOVERY

REDUCE RISK: LEGACY TECH, HOSTILE VENDORS, WORKFORCE

HELP I'M LOOKING FOR

"CAPACITY" FORECASTING AND "ESTIMATION" & WISHFUL THINKING

PORTFOLIO LEVEL WIP CONSTRAINTS

(STOPPING FEATURE TETRIS)

IMPROVING INTAKE LEADTIME WITH TRADATIONAL IT MINDSETS

CREATING CAPACITY FOR "BACKLOG SWARMING"