DevOps – Track 4

How can I *shift-left* to achieve better business results?

IBM Service Virtualization and Automated Integration Testing - Continuous Testing

Rational Development and Test Environment for System z (RD&T)

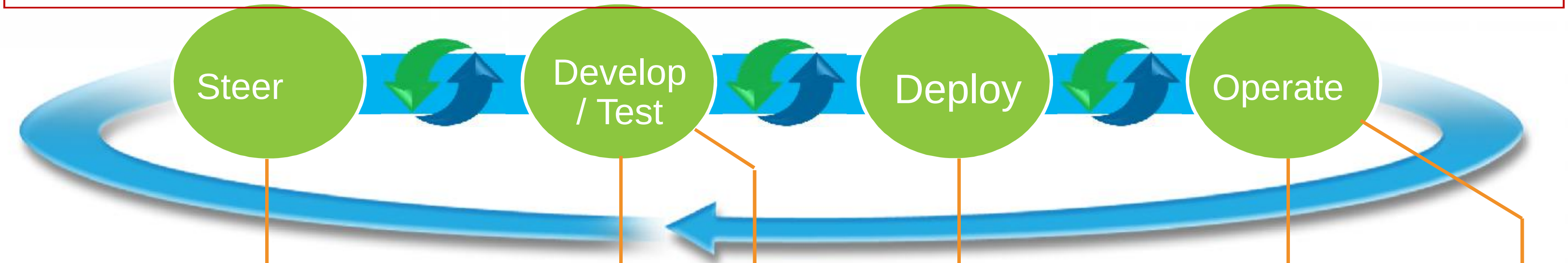
Rational Integration Tester (RIT)

Rational Test Control Panel (RTCP)

How can I *shift-left* to achieve better business results?

IBM DevOps – Broad set of DevOps capabilities

Address bottlenecks across the application delivery lifecycle



Continuous Business Planning

Rational Team Concert

Collaborative Development

Rational Collaborative Lifecycle Management
Rational Developer for System z
Rational Compilers
Worklight Studio

Continuous Release and Deployment

UrbanCode Release
UrbanCode Deploy with Patterns
SmartCloud Orchestrator

Continuous Monitoring

SmartCloud Application Performance Management
SmartCloud Control Desk

Continuous Testing

Rational Quality Manager
Rational Test Virtualization Server
Rational Test Workbench
Rational Test Workbench – Mobile Test Edition
Rational Development and Test Environment for System z

Worklight Quality Assurance
InfoSphere Optim Test Data Management
Security AppScan

Continuous Customer Feedback & Optimization

Tealeaf CX
Digital Analytics
Worklight Quality Assurance

Bluemix DevOps services

Agile planning and tracking
Application auto-scaling
AppScan mobile analyzer
Continuous delivery pipeline
Git hosting

Mobile application security
Mobile data
Mobile quality assurance
Monitor & analytics
Push

RapidApps (beta)
Server-side code
Web IDE

Only 1/10th of an iceberg is above water

Traditional UI Testing:

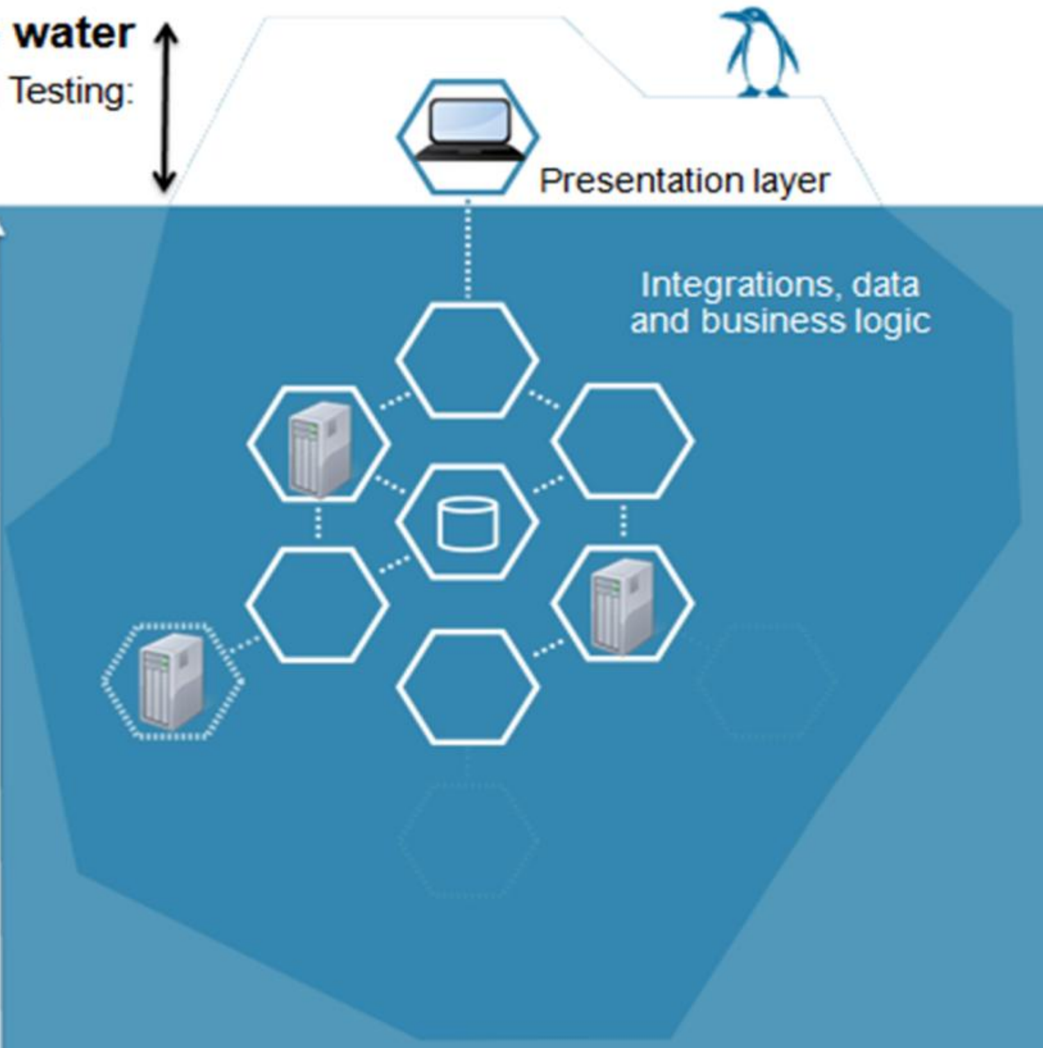
Automated Integration Testing and Service Virtualization:

Tests can bypass the UI and execute beneath the iceberg. Test while your system is being built and at an accelerated rate. Much greater coverage = much greater confidence

Introduces the concept of testing side effects. Validate the contents of message queues, log files and databases as well as simple inputs/outputs.

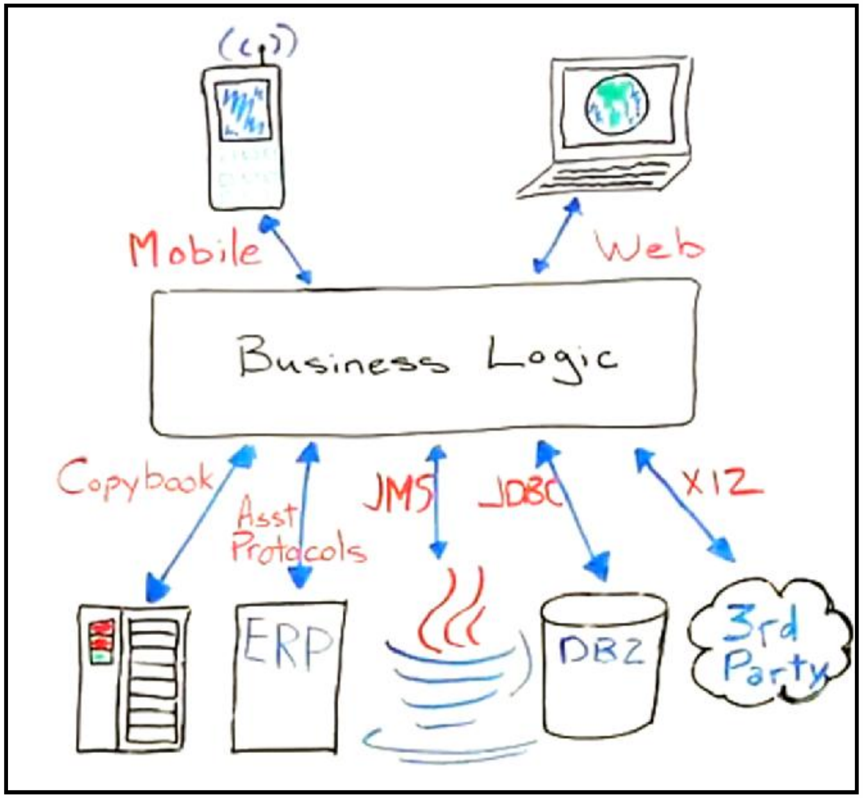
Service virtualization can replace components and databases in complex test scenarios. You can test when environments are unavailable or even before they have been built.

Service virtualization can work with any type of test – manual, automated or performance. It doesn't matter



The majority of risk in today's systems is seldom tested as it is unseen.

Applications are becoming more complex to Test



- Increasing expense of standing up complex environments
- Delays caused by lack of system availability
- Third-party services can be especially costly
- Testers spend more time setting up tests than testing

$5-30 million investment

Typically required to build a single test lab for a Fortune 500 company. Most have dozens…

30-50% average time

Testing teams spend on setting up test environments, instead of testing

Primary *issues* with “current” Virtualized Test/Service solutions…

Test Lab $$$/People

- HW-based virtualization or cloud based resources have provided only a partial savings (20-30%)
- Installation and configuration of software is still very labor intensive
- Certain systems cannot leverage HW virtualization, e.g. costly third party services, mainframe applications, proprietary systems

Cycle Time

- Testing new functions still require to have an environment available to develop test scripts
- The time wasted waiting for a test environment is severely reducing the ability to do proper acceptance testing

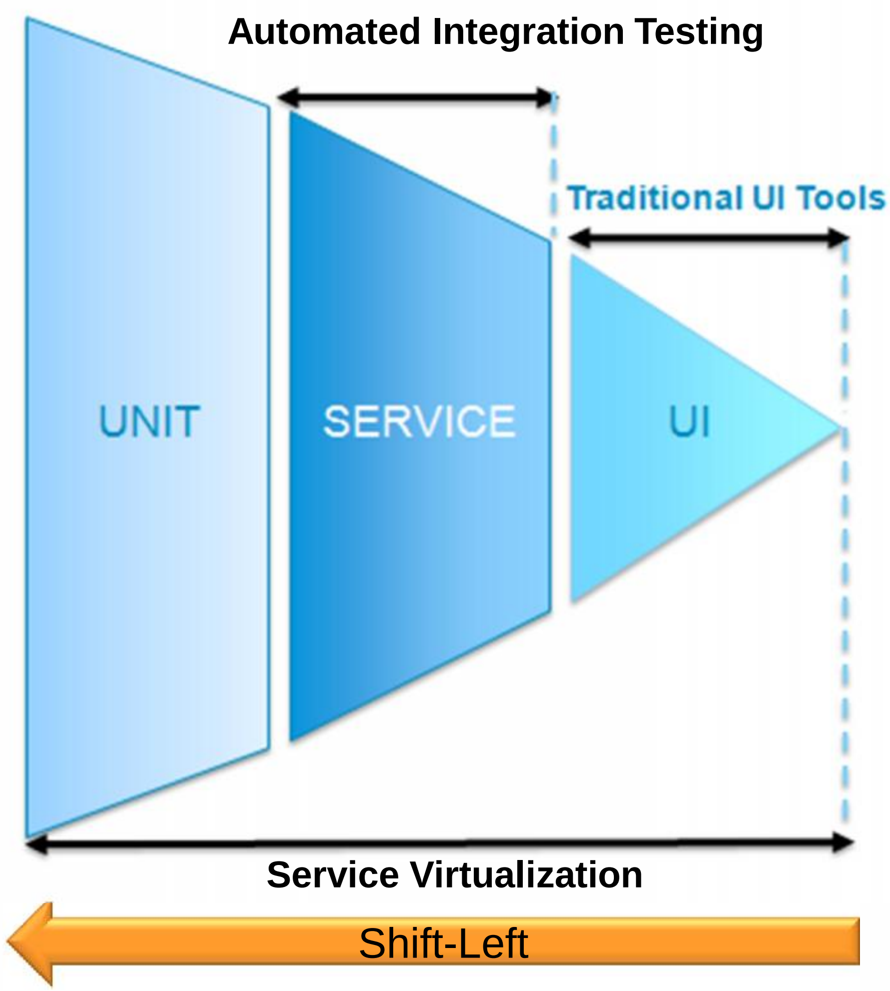
Risk

- Too many “trivial” defects are still found late in the process by Quality Assurance teams

Service Virtualization – *What it Is and Isn't !*

- **Service Virtualization**
  - A solution that includes some level of automation, for creation and management of "stubs" and reusing them across IT for the purposes of doing integration and other types of testing
- **Different names/terms…same application intent:**
  - Virtualization…"interchangeable"…Simulation
    - **Test / Application / Service / Stubbing / Mocking**
- **Service Virtualization is NOT hardware or OS virtualization**
  - This is not "VMware", although we can help customers reduce hardware/software licensing needs
- **Service Virtualization allows you to…**
  - Eliminate critical development and testing constraints by virtualizing IT resources
    - ***"Shifting Left" the Development and Testing***
  - Deliver realistic simulated development and test environments at a fraction of the cost
  - Isolate defects and drill into their root cause
  - Create automated regression test suites to reduce test cycle time and improve test coverage and quality
  - Drive benefits across the entire application development and test lifecycle

Shift-Left
Are the Testing efforts directed at the highest risks?



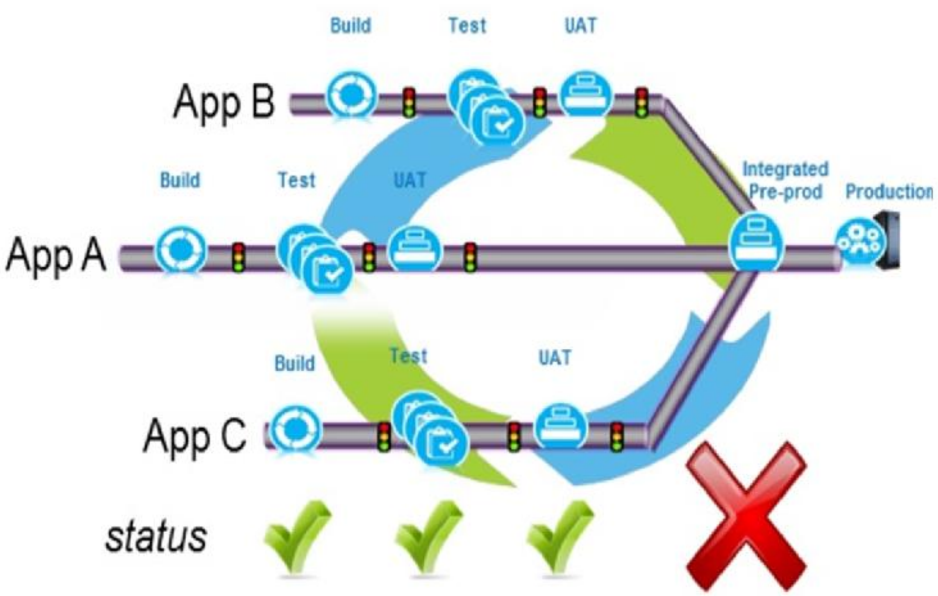
- Testing only at the UI level provides little confidence for deployment of complex integrated mainframe and distributed systems
- Testing must be shifted left to focus on the highest areas of risk
- Automated integration massively increases testing coverage and confidence
- Service virtualization can be applied at all phases – even allowing integration testing to begin in parallel with Unit Testing.

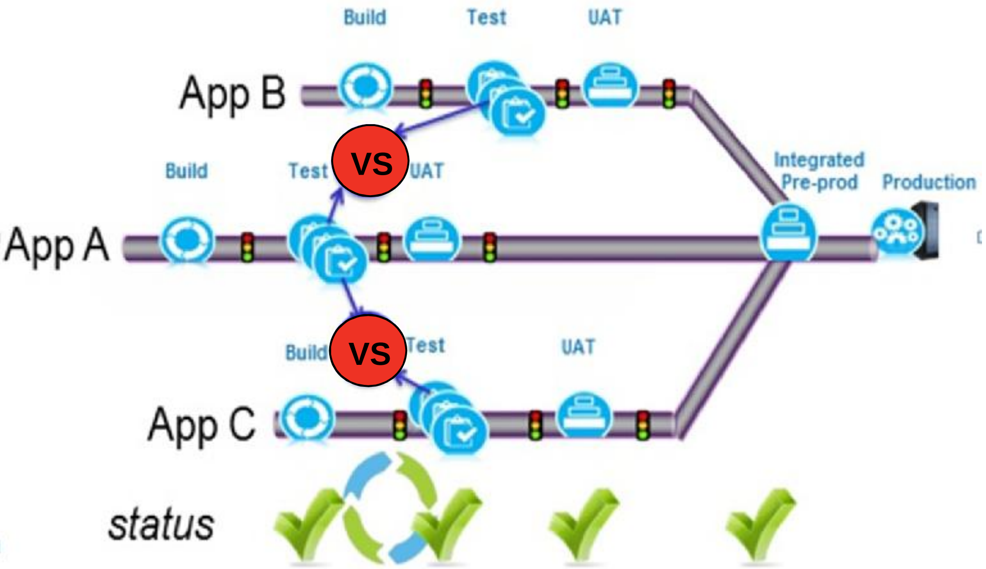
Service/Test Virtualization Enables *Early* Automated Integration Testing → *Shifts testing to the left!* *Shifts focus to Integration*

Problem

Dependencies between applications traditionally means that integration testing is done:

- Late, putting release schedule at risk
- Bugs detected later, costing more to fix
- Overall longer release cycle



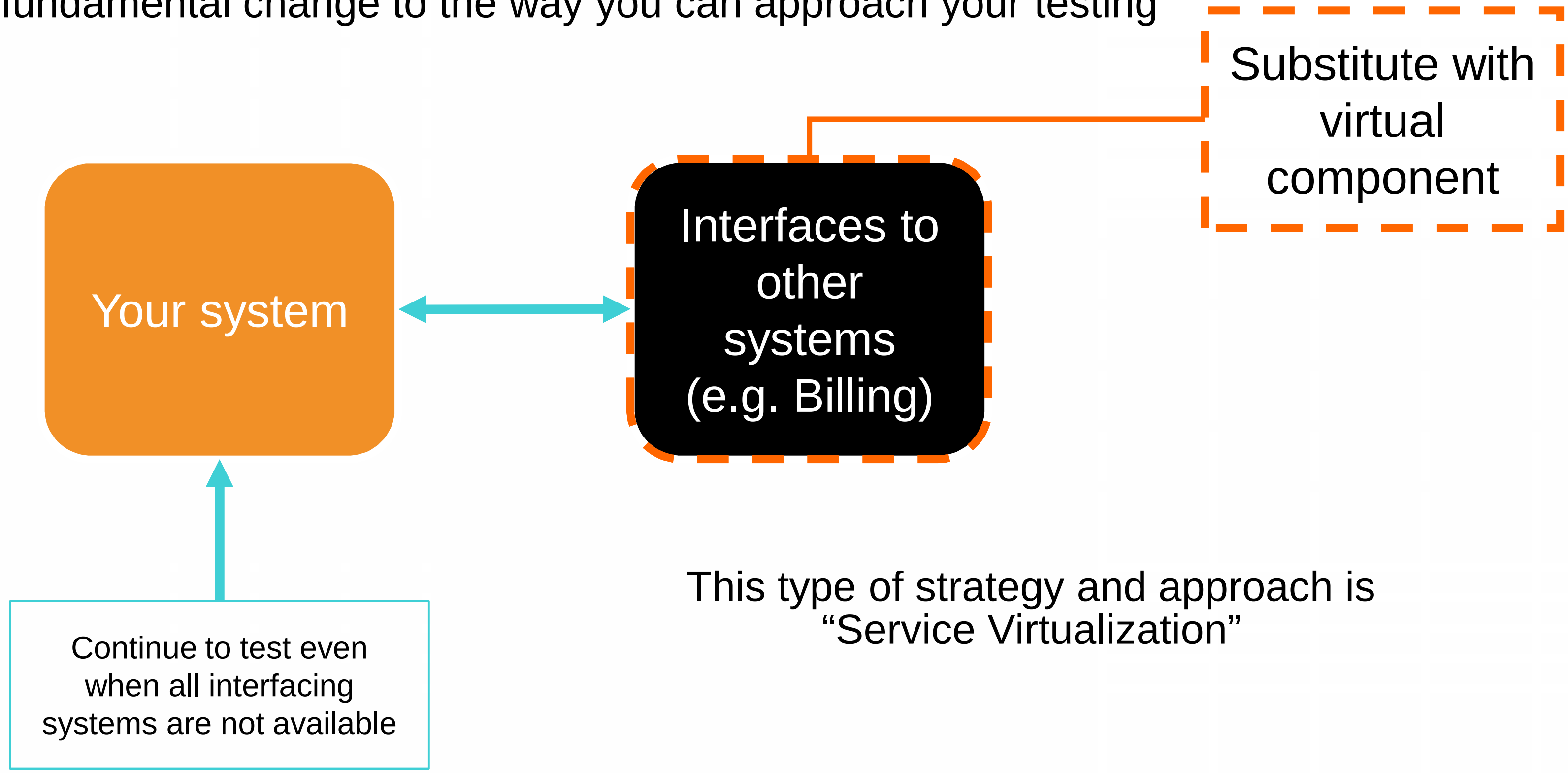


The Shift-Left Approach

- On-simulation/virtualization solutions which can emulate the production environment
- Demand available system integration test environments
- Testing earlier into the development life cycle

Shift-Left with *a "shift" in your TEST Environment Strategy*

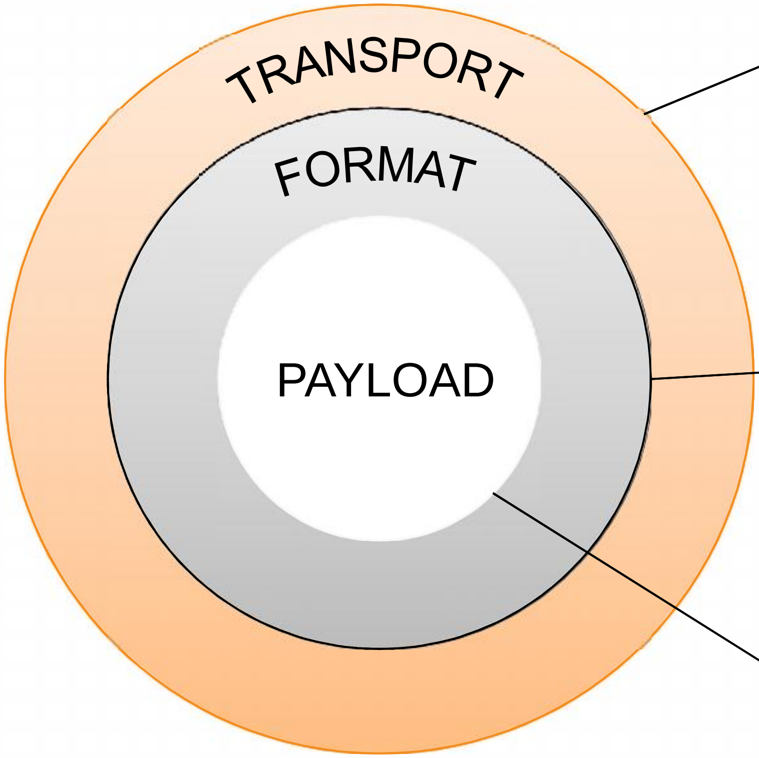
- IBM technology helps to eliminate project and environment contention. Create virtual components that faithfully simulate behavior of missing components
- Now you can test when environments are unavailable or even before they have been built
- A fundamental change to the way you can approach your testing


Substitute with virtual component
Your system
Interfaces to other systems (e.g. Billing)
Continue to test even when all interfacing systems are not available

This type of strategy and approach is "Service Virtualization"

Transports and Formats

The building blocks of an Integration Test or Virtualized Service



The transport is the technology that carries your message. e.g. HTTP

Several transports have advanced support e.g. sift and pass through

The format is the type of message that is carried on the transport e.g. XML

Formats are presented in an easily readable and configurable tree structure

The payload is the contents of the message fields

Fields can be parameterized using data sources

Supporting technologies businesses Depends On!

Protocols, message formats, and technologies (not a complete list)

Transport/Protocol

- ActiveMQ
- Email (SMTP, IMAP)
- Files
- FTP/S
- HTTP/S
- WebSphere MQ
- Java RMI
- JMS
- SAP IDoc, BAPI, RFC
- Software AG’s IB & IS
- Sonic MQ
- TCP/IP
- TIBCO Rendezvous, Smart Sockets & EMS
- CICS TG
- CICS IPIC
- IMS Connect

Message Formats

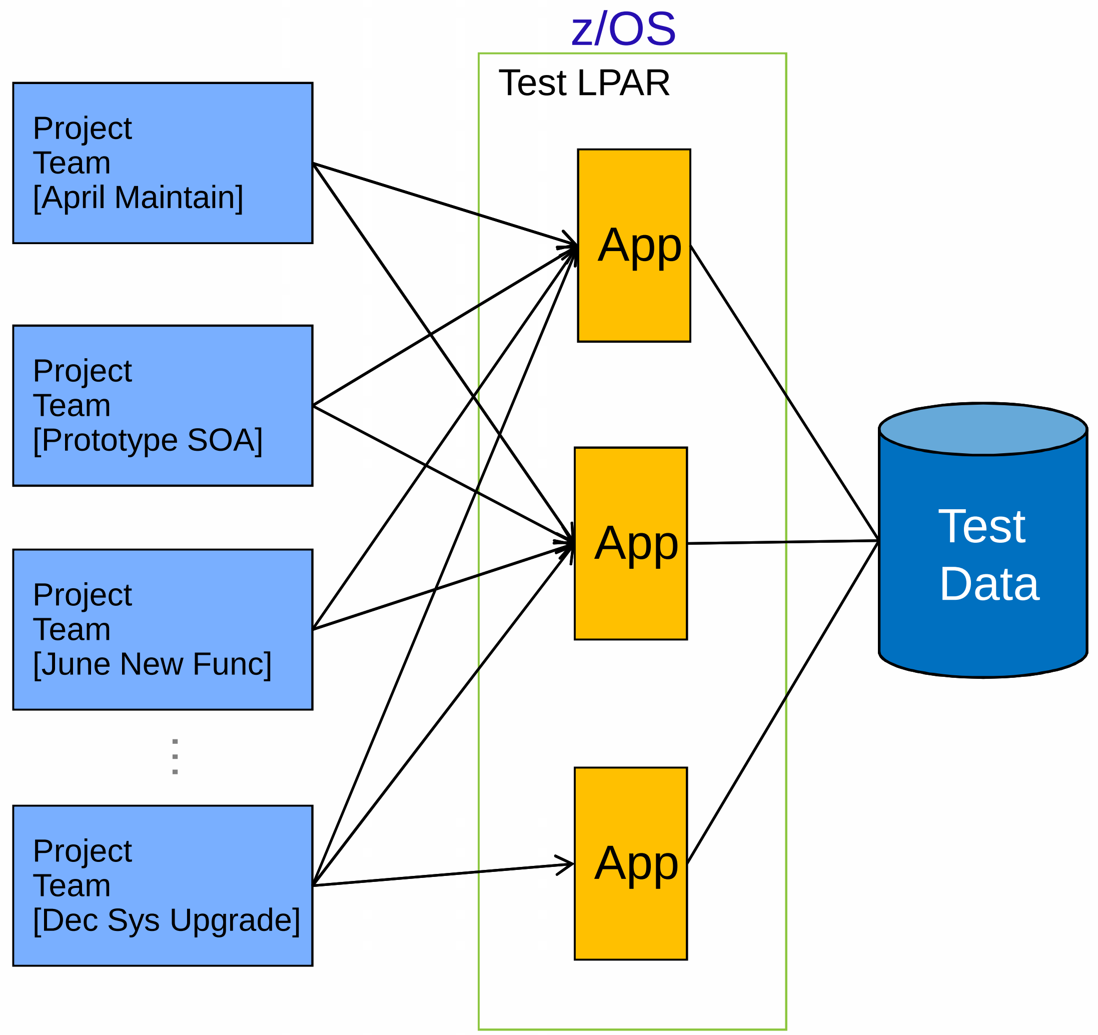
- .Net Objects
- Bytes
- COBOL Copybook
- ebXML
- ASC X12
- UN/EDIFACT
- DFDL
- Fixed Width
- Java Objects
- MIME
- SOAP
- Software AG Broker Docs
- SWIFT
- CHIPS
- FedWire
- TIBCO ActiveEnterprise
- HL7 v3
- XML (DTD, XSD, WSDL)
- ISO8583
- Custom

Technologies

- CentraSite
- Oracle Fusion
- SCA Domain
- Software AG IS, Broker, BPMS
- WebSphere Application Server
- WebSphere Message Broker
- WebSphere MQ
- Websphere SI Bus
- Sonic ESB
- System z (CICS and IMS)
- TIBCO ActiveMatrix
- UDDI
- Web Services
- WebSphere RR
- WSDL
- BPM
- Databases
- Log Files

Typical z/OS Testing Architecture

Organized by project team, vertically scaled, sharing resources, limited automation

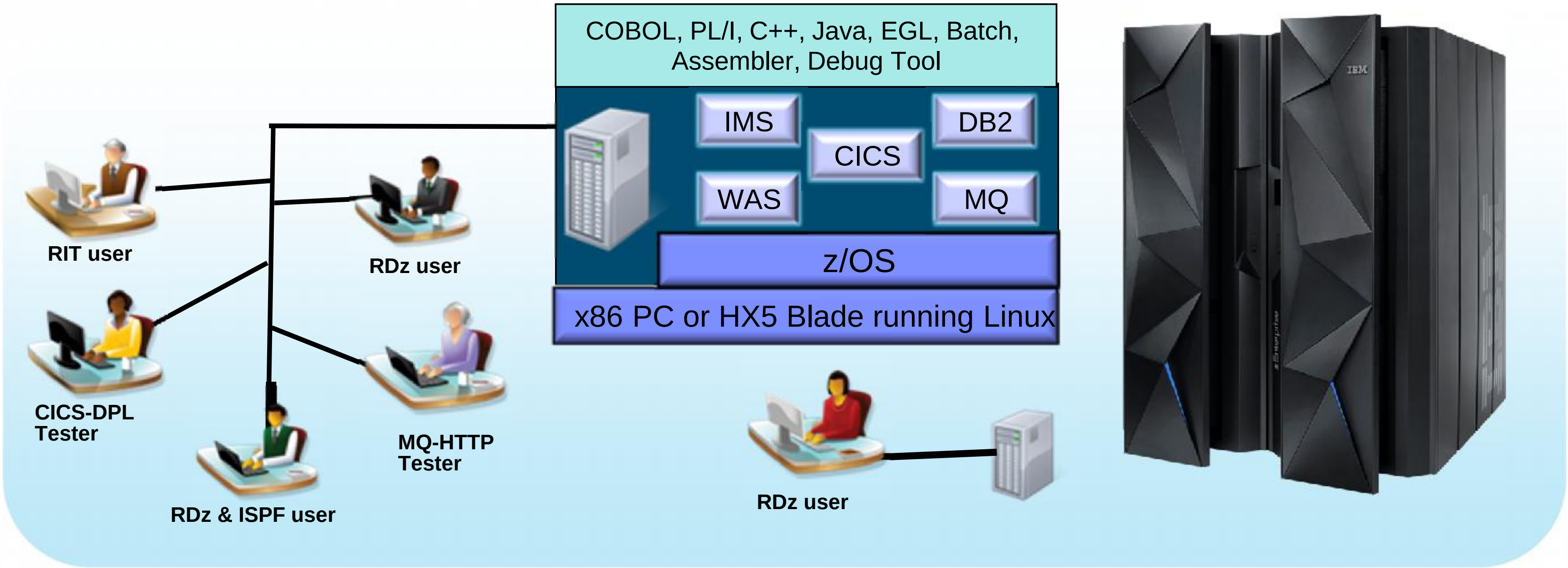

z/OS
Test LPAR
Project Team [April Maintain]
Project Team [Prototype SOA]
Project Team [June New Func]
⋮
Project Team [Dec Sys Upgrade]
App
App
App
Test Data

Problems Encountered

- Shared resources combined with overlapping schedules can elicit conflicts, impede innovation and slow code delivery
- Coordination of environmental changes and releases cause bottlenecks, delays and additional overhead
- Shared test data is difficult to manage and can lead to over testing or incorrect test results

Rational Development & Test Environment for System z (RD&T)

The ultimate in modern application development for System z



COBOL, PL/I, C++, Java, EGL, Batch, Assembler, Debug Tool

IMS

DB2

CICS

WAS

MQ

z/OS

x86 PC or HX5 Blade running Linux

RIT user

RDz user

CICS-DPL Tester

MQ-HTTP Tester

RDz & ISPF user

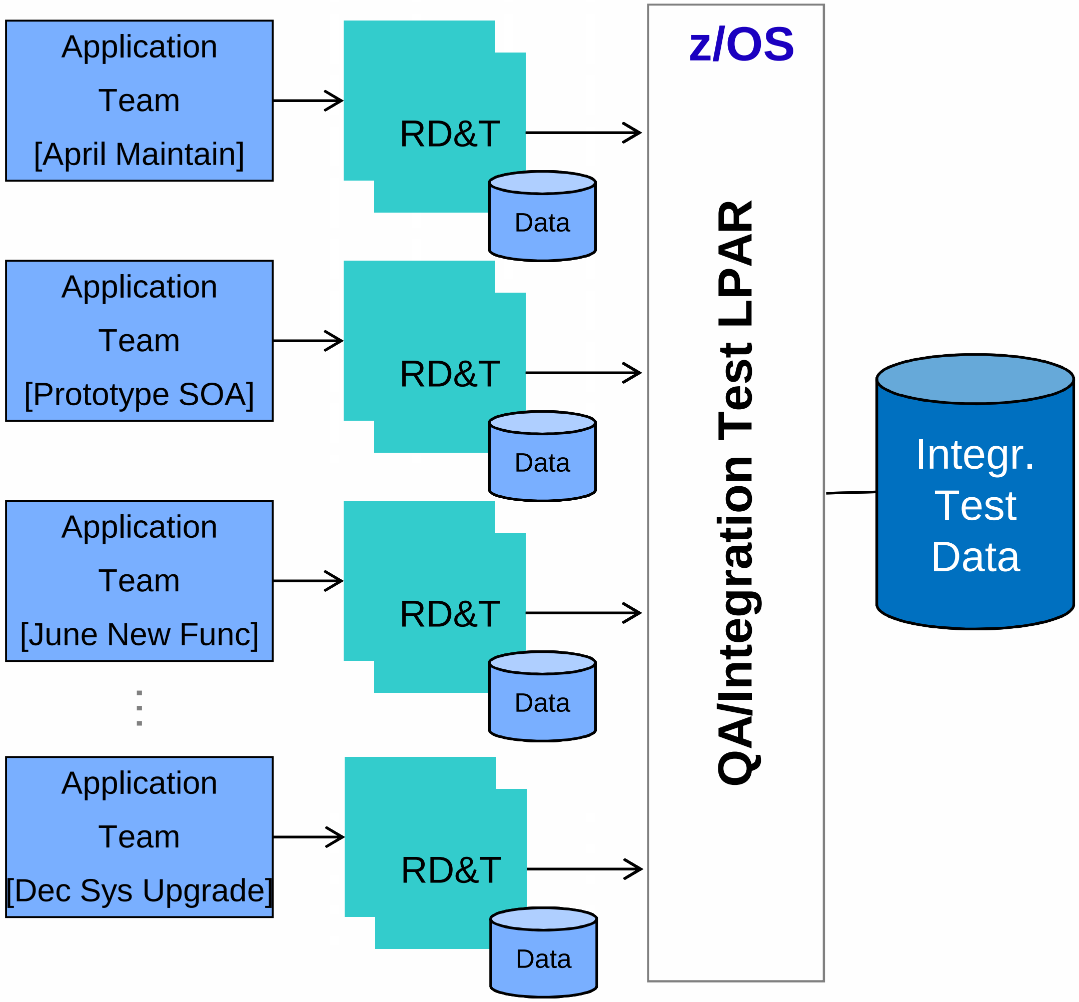
RDz user

- Liberate developers to rapidly prototype new applications (Sandbox environment)
- Develop and test System z applications anywhere, anytime! (Developer freedom)
- Free up mainframe development MIPS for production capacity (Hard cost savings)
- Eliminate costly delays by reducing dependencies on operations staff (Time-to-Market)

Note: This Program is licensed only for development and test of applications that run on IBM z/OS. The Program may not be used to run production workloads of any kind, nor more robust development workloads including without limitation production module builds, pre-production testing, stress testing, or performance testing.

Using RD&T as a Development & Testing Platform

Organized by application team, horizontally sliced, dedicated resources, highly automated


Application Team [April Maintain]
RD&T
Data
Application Team [Prototype SOA]
RD&T
Data
Application Team [June New Func]
RD&T
Data
Application Team [Dec Sys Upgrade]
RD&T
Data
z/OS
QA/Integration Test LPAR
Integr. Test Data

The Dev and Test Environment consists of:

- **Dev and Test Environment (based on zPDT)**
  - Dev and Test Environment can provide a System z development platform on a PC
  - Capable of running z/OS
  - Provides great flexibility to run a customized environment
- **Software stack** provides a choice of IBM middleware test environments
  - Actual middleware software (including z/OS)
  - Actual enterprise compilers
  - No API simulation

How & What value does Rational Service/Test Virtualization provide?

Cost, Cycle Time, Risk

Capabilities provided:

- Capture and model virtual services
- Virtualize heterogeneous hardware, software and services to provide 24x7 testing capabilities

Value:

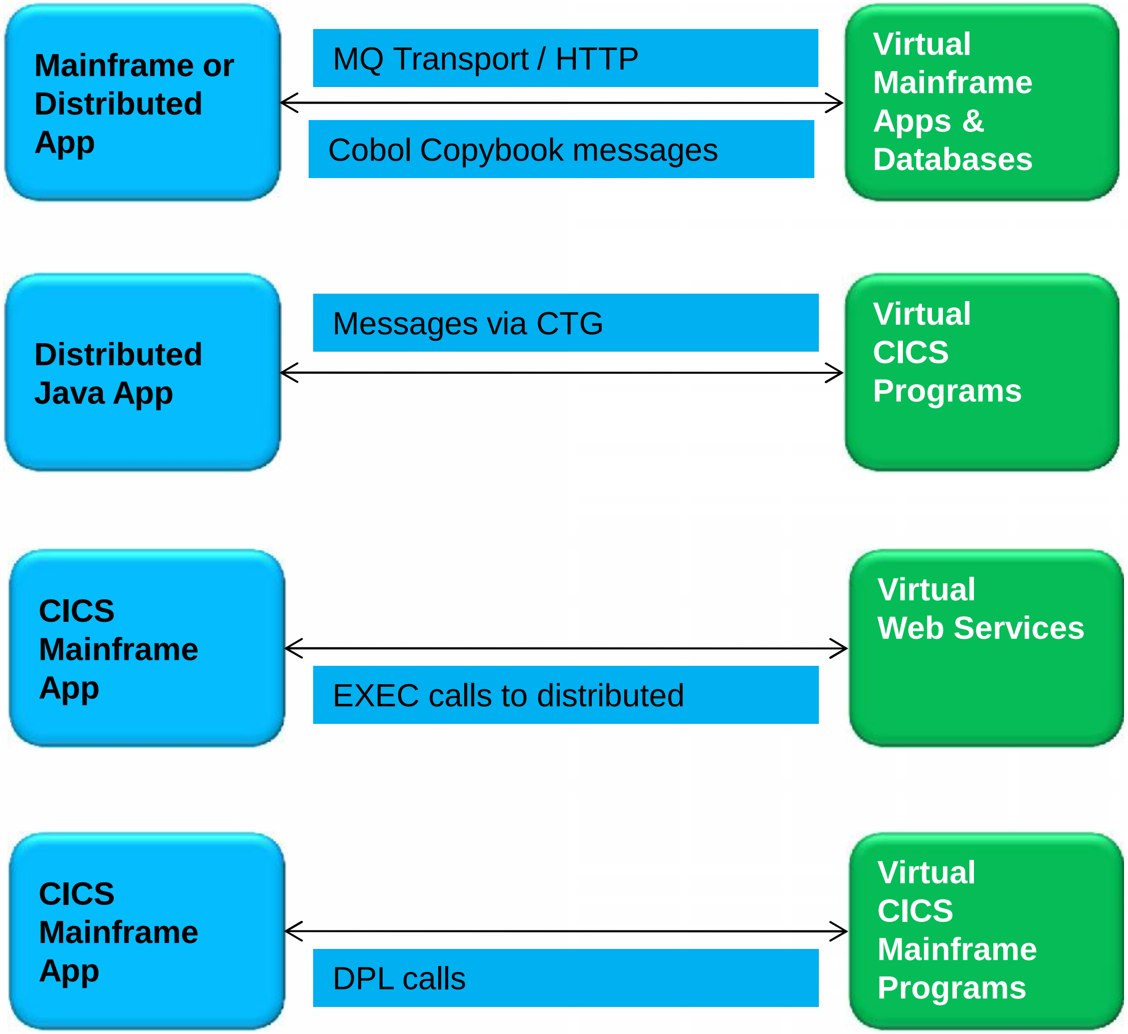
- Reduce infrastructure costs of traditional testing environments by reducing the number of integration platforms. Reduced technical debt for projects.
- Deliver earlier, end-to-end **continuous integration testing** across hardware, software by relieving dependencies and Lowering TCO to consume service
- Reduce delivery risk due to unmanaged dependencies
- Faster time to provision versus conventional environment stand-up times
- Share test environments across their team enabling parallel development
- Employ traceability and collaborate in-context across their development domains
- Test services and applications before their user interfaces becomes available and do integration testing

Drives Test Virtualization → Continuous Integration Testing

- Test Virtualization is an enabler for Continuous Integration Testing
- Services, applications, systems are introduced into the continuous integration cycle in a prioritized, controlled fashion *(when you need them!)*

Benefits of Virtualizing System z Applications

- Set up multiple **Test environments in minutes vs. weeks**
- **Test multi-platform applications** that produce and consume **Cobol Copybooks** / data files and interact with MQ, DB2 and other systems by virtualizing complex dependencies
- Improve coverage for integration and **negative test scenarios**
- Improve coverage for latency tests by simulating application behavior taking too long to respond
- Early performance testing to **ensure the robustness** of system under test
- Execute **Parallel development** on different parts of an application or system by virtualizing dependencies



The ROI of Service Virtualization

The results are real!

A major **European bank** implemented service virtualization and automated integration testing as part of an Agile transformation to enable continuous testing. As a result of IBM capabilities, their system integration test (SIT) process was reduced from 3 weeks down to 4 hours!

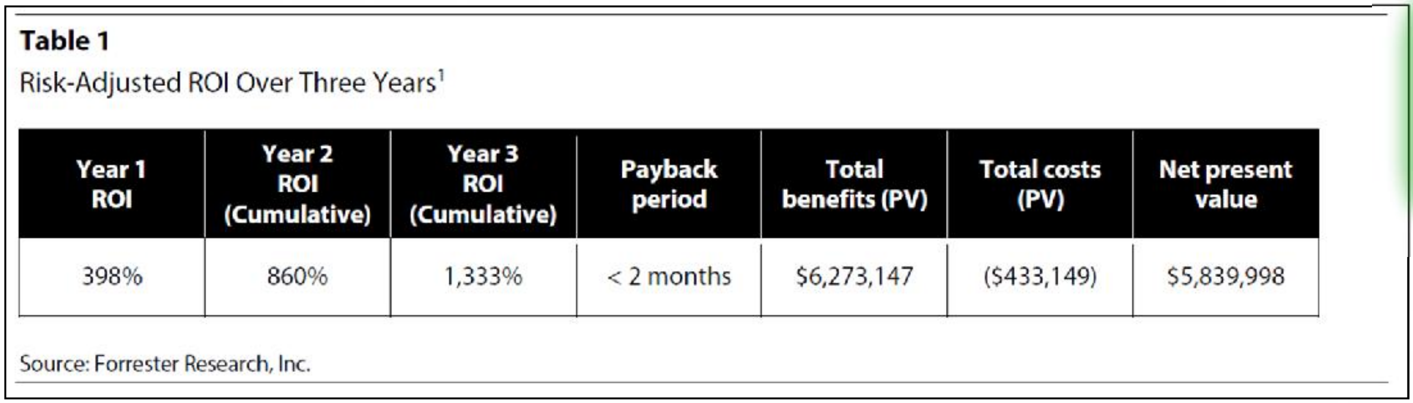


Table 1

Risk-Adjusted ROI Over Three Years[1]

Year 1 ROI	Year 2 ROI (Cumulative)	Year 3 ROI (Cumulative)	Payback period	Total benefits (PV)	Total costs (PV)	Net present value
398%	860%	1,333%	< 2 months	$6,273,147	($433,149)	$5,839,998

Source: Forrester Research, Inc.

13x ROI in 3 years!

The Total Economic Impact™ of IBM Rational Service Virtualization and Test Automation Solutions, a July 2014 commissioned study conducted by Forrester Consulting on behalf of IBM

3 Year Benefits:

- **Increased project delivery capacity 100%** (40 projects to 80 projects)
- **Avoided hiring three additional FTE's**, saving almost $600k.
- **Reduced incidents found in production from 2.5 to .3 per project, saving $3 million**
- **Identified defects earlier** and freed the team up to focus on new capabilities, saving over $560k
- **Reduced consulting and third party testing fees** with better test automation, saving $408k
- **Eliminated the need for additional hardware and labor** to set up test environments, saving $700k
- **Generated new revenue for the business** – estimated at tens of millions!

Bank Of Montreal accelerates project delivery 51-75% with IBM Continuous Testing solution

Research by TechValidate

51-75% Accelerated
Project delivery

10-25% Decreased
Defects in production

10-25% Improved
Team productivity

Business Challenge: The testing organization suffered from inefficient testing processes that were a drain on team productivity.

The Solution: The Bank switched from Parasoft to IBM's test automation because IBM had superior product functionality and was easier to use. IBM service virtualization and test automation was used to:

- Deliver applications built on a service oriented architecture
- Increase the speed at which they could delivery TIBCO applications

Software:
Rational Test Virtualization Server, Rational Test Workbench

"Automated testing is a requirement when building Web Service solutions. IBM's Green Hat solution was the best tool on the market at the time of our investigation and the support was second to none."
— Patrick Brophy, Director / VP , Bank of Montreal

The Coca-Cola Company decreased Production defects by 26-50% with IBM

26-50% Decreased
Defects in production

26-50% Accelerated
Project delivery

26-50% Reduced
Cost of testing

26-50% Improved
Team productivity

Business Challenges:

- Quality issues/defects found in production
- Insufficient or unavailable test environments

The Solution: The Coca-Cola Company chose IBM's service virtualization and automated integration testing solution over other offerings for the following reasons:
1) broadest support for the technologies in their environment; 2) cost of the solution; 3) realizing increased developer and tester productivity.

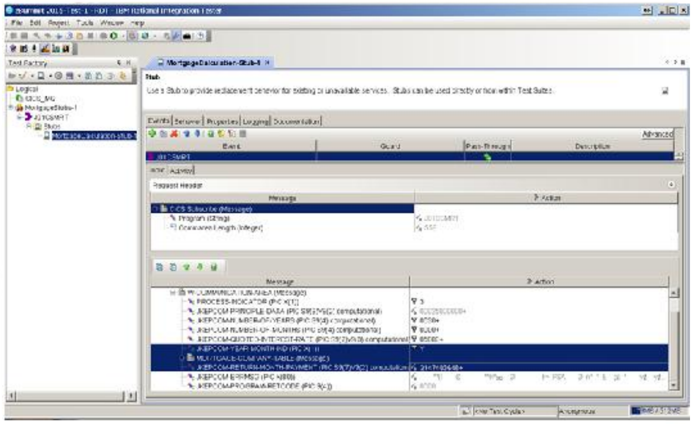
Software:
IBM Rational Test Virtualization Server, IBM Rational Test Workbench

Test & Virtualization Scenario

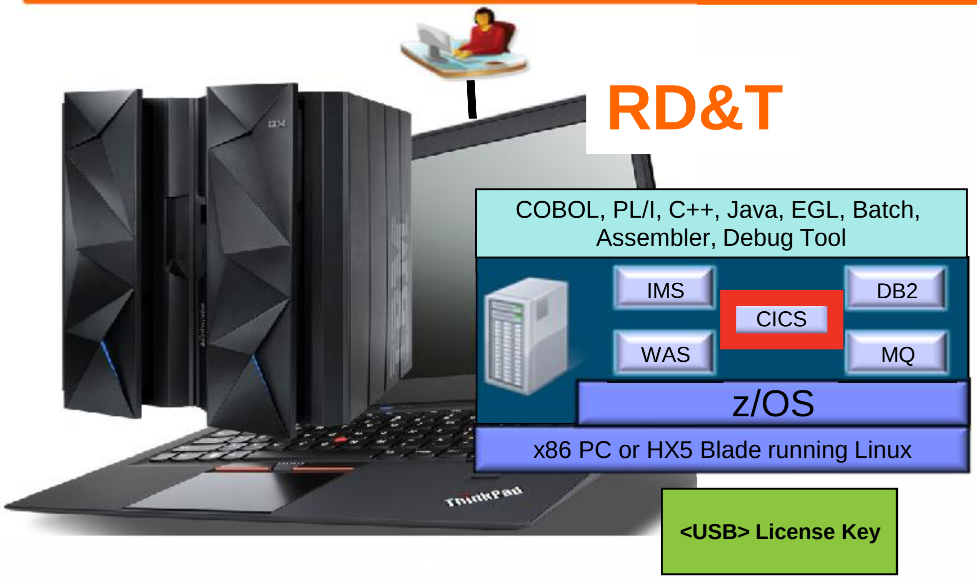
Mortgage Payment Calculation Application

- **Problem**:
  - CICS application is unavailable for testing
- **Resolution**:
  - Create a stub to virtualize CICS-DPL transaction
  - Test/Run Stub
  - Apply change to Stub
    - Demonstrate Sift & Pass-Through
    - Force Error condition
  - Publish/Deploy Stub to Development Team

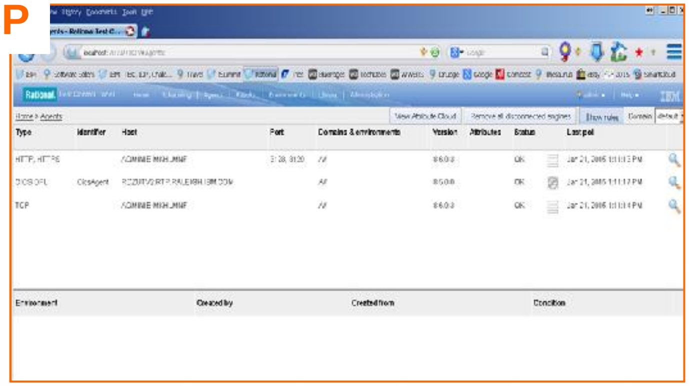
RIT



- Create a Logical and Physical model of our system under test
- Record events from the system
- Create, design and test stubs before publishing to Dev team


RD&T
COBOL, PL/I, C++, Java, EGL, Batch, Assembler, Debug Tool
IMS
CICS
DB2
WAS
MQ
z/OS
x86 PC or HX5 Blade running Linux
<USB> License Key

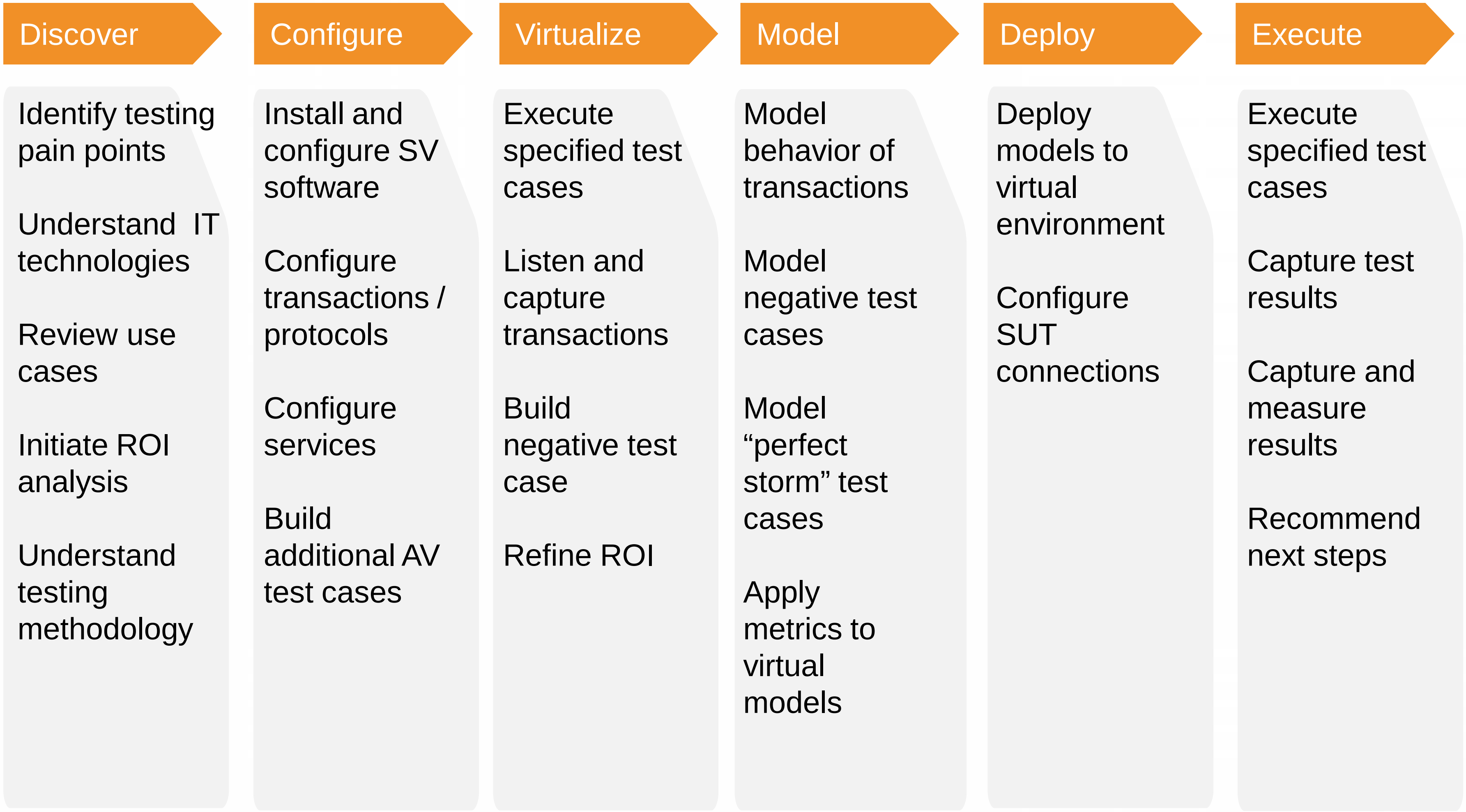
RTCP



- Web-based UI to administer RIT agent, publish stubs, tests and test suites

Demo

Steps to Getting You Started… From Discovery to Execution



Discover

- Identify testing pain points
- Understand IT technologies
- Review use cases
- Initiate ROI analysis
- Understand testing methodology

Configure

- Install and configure SV software
- Configure transactions / protocols
- Configure services
- Build additional AV test cases

Virtualize

- Execute specified test cases
- Listen and capture transactions
- Build negative test case
- Refine ROI

Model

- Model behavior of transactions
- Model negative test cases
- Model “perfect storm” test cases
- Apply metrics to virtual models

Deploy

- Deploy models to virtual environment
- Configure SUT connections

Execute

- Execute specified test cases
- Capture test results
- Capture and measure results
- Recommend next steps

ibm/devops

Acknowledgements and Disclaimers

Availability: References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates.

The workshops, sessions and materials have been prepared by IBM or the session speakers and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided AS-IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. Nothing contained in this presentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.

IBM, the IBM logo, ibm.com, Rational, the Rational logo, Telelogic, the Telelogic logo, Green Hat, the Green Hat logo, and other IBM products and services are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml

END

- Optional Reference & Backup Slides

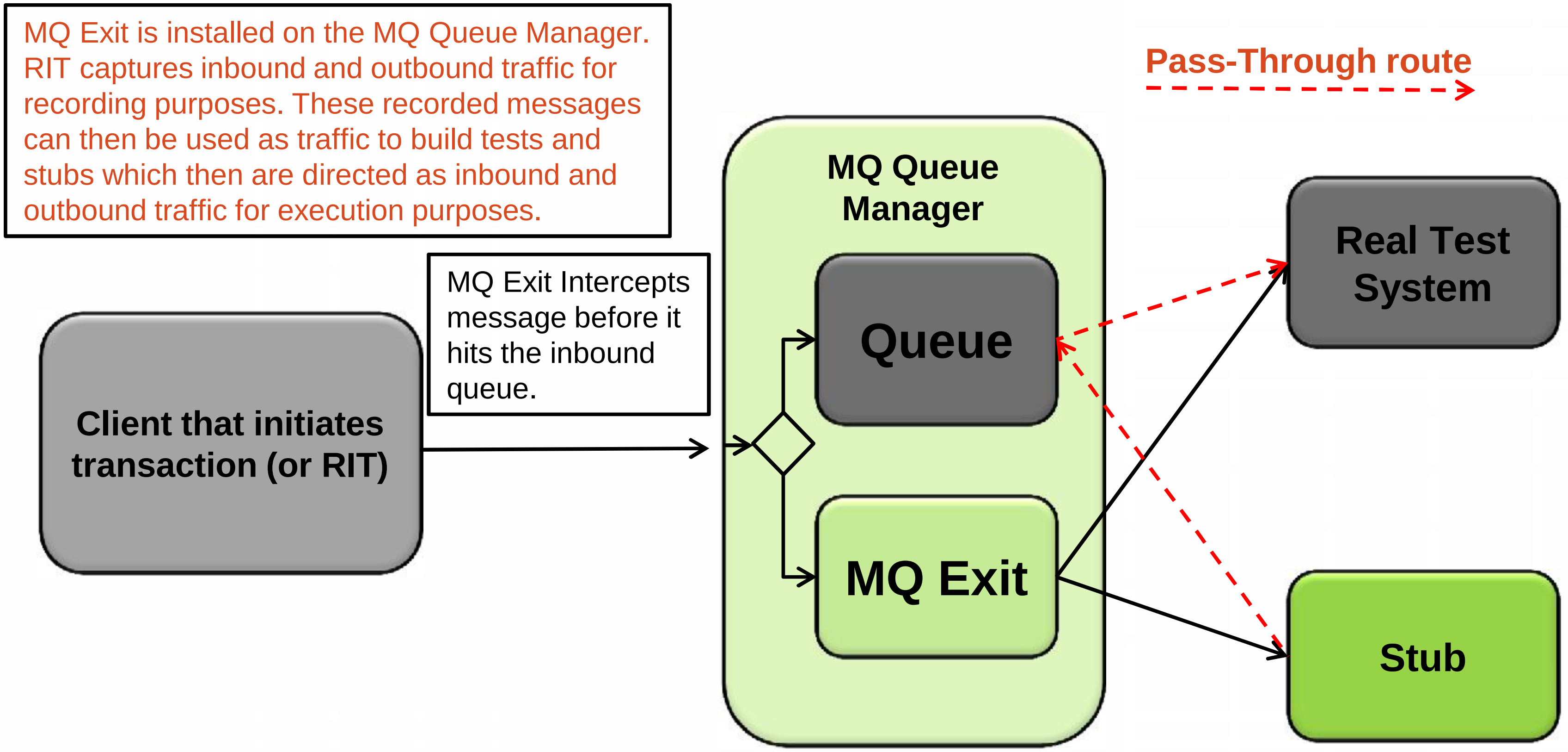
IBM’s Rational Test Virtualization Solution

- **IBM’s Rational Test Virtualization Solution** focuses on a key set of problems most organizations face: that being time, and resources required to set up and manage test environments. Traditionally, people have been running around installing hardware, setting up application servers, database servers, installing application software, configuring all of that. Not only this is very capital intensive problem, but as environments have become more and more complex, this is also a very error prone process that typically involves a lot of scrap and rework. What the IBM solutions does is to enable organizations to address that problem by Virtualizing complete stacks of software, hardware, and services, enabling developers and testers to stand up test environments in a matter of minutes vs. weeks, which enables them to start testing much earlier than previously possible.

An example using a Virtualized MQ Service/Stub

Sift and Pass-Through with MQ

MQ Exit is installed on the MQ Queue Manager. RIT captures inbound and outbound traffic for recording purposes. These recorded messages can then be used as traffic to build tests and stubs which then are directed as inbound and outbound traffic for execution purposes.



MQ Exit Intercepts message before it hits the inbound queue.

If stub is down, then MQ Exit directs message to Live system.

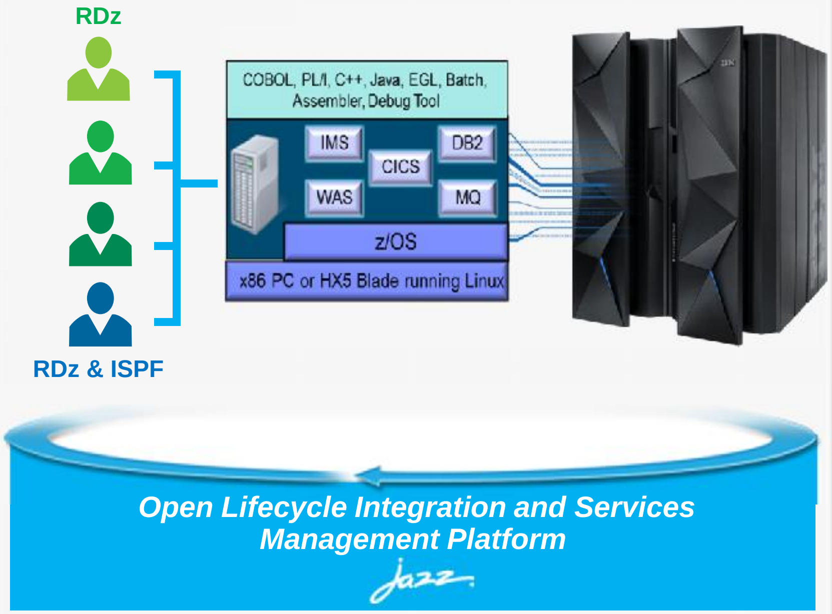
Rational Development and Test Environment for IBM System z v9

Reduce System z development costs with high fidelity off host development and test

- Exploit added zEnterprise capability for increased mainframe compatibility
  - **New** Parallel Sysplex support via the latest System z coupling facility
  - **New** zEnterprise EC12 platform support
  - Better development and test coverage early in the cycle for higher quality applications delivered more quickly
- Improve hardware utilization and simplify management
  - **New** support for virtualization technologies, including VMWare vSphere and IBM zEnterprise BladeCenter Extension
- **New**, flexible licensing to support automated testing and variable usage patterns

ENHANCED

Rational Development and Test Environment for System z


RDz
COBOL, PL/I, C++, Java, EGL, Batch, Assembler, Debug Tool
IMS
DB2
CICS
WAS
MQ
z/OS
x86 PC or HX5 Blade running Linux
RDz & ISPF
Open Lifecycle Integration and Services Management Platform
Jazz

"RD&T provides us with the ability to try development approaches that may introduce instability to our normal environment with none of the associated risks. It provides an environment where we can quickly start a new project without waiting for setup delays inherent in our day-to-day processes."

Development Team Lead

DEVELOPER VALUE

RD&T Device maps – Defining devices to z/OS

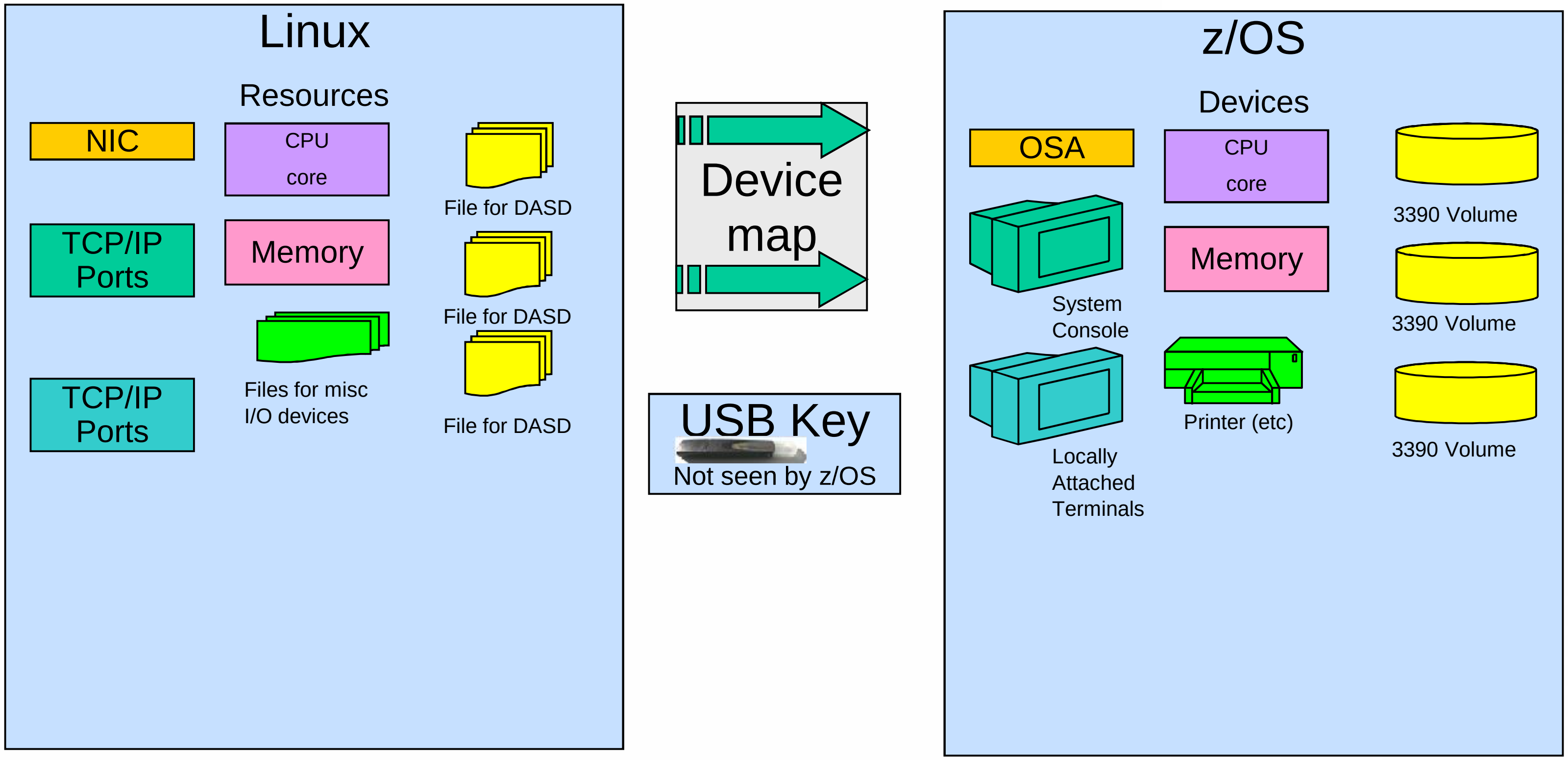
"**Devmap**" – will map the entities known to Linux, to the devices known to z/OS

z/OS volumes are contained in Linux files

z/OS Communication device is a Linux ethernet card or a logical tunnel device

z/OS printer or card readers can be Linux files

… Other devices are also possible such as SCSI attached tape drives.


Linux
Resources
NIC
CPU core
File for DASD
TCP/IP Ports
Memory
File for DASD
Files for misc I/O devices
TCP/IP Ports
File for DASD
Device map
USB Key
Not seen by z/OS
z/OS
Devices
OSA
CPU core
3390 Volume
System Console
Memory
3390 Volume
Locally Attached Terminals
Printer (etc)
3390 Volume