Intelligent Virtualization

How to create better virtual services

James Walker
Principal Software Engineer
Continuous Delivery Business Unit

28th March 2017

Agenda

1. CA SERVICE VIRTUALIZATION
2. WEATHER APPLICATION
3. CREATE A MODEL OF DATA TRAFFIC
4. GENERATE TRAFFIC
5. DEPLOY A VIRTUAL SERVICE
6. LOOK AT JAVELIN FLOWS

CA Service Virtualization

Remove Development and Test Constraints to Ignite Digital Transformation

Simulate To Accelerate

Remove constraints by simulating dependent systems, API, as well as virtual services

~~System Constraints~~ ~~Data Constraints~~ ~~Cost Constraints~~ ~~Development Constraints~~

- **5** of the **top 5 Telcos**
- **4** of the **top 5 Banks**
- **3** of the **top 5 Insurers**

Source: 2014 Fortune 500

REAL WORLD RESULTS

- Up to **50% reduction** in in application dev/test cycles
- Up to **90% more defects detected** at least 1-step earlier in SDLC with "Shift-Left" Testing
- **Millions saved** in infrastructure costs and 3rd party fees

Lloyds Bank

- Before … Took 5 people days to do 20 tests
- AFTER SV … Takes 1 person 1-minute to do 20 tests

http://ow.ly/HxErN

[1]Bender RBT

Integration Testing

Create test environments via SW that exactly replicate production infrastructure.

- ✓ Catch defects in unit testing where less expensive than full system
- ✓ Test components out of order when needed
- ✓ Capture behavior/performance data of composite applications
- ✓ Avoid manual repetitive development of stubs or mockups

Increase availability of testing platforms up to 90%

KPN Telecommunications

Enable Agile Testing Sprints

Save $2.5 million in infrastructure

Forrester TEI Study November 2015

Cut overall development cycle by 66%

Nordstrom

AutoTrader

- **Avoided $300,000 in test hardware and software costs**
- **Cut integration time from three days to three hours**
- **Decreased software defects by 25 percent hours**

Autotrader

STABILITY

"CA Service Virtualization gives us a more stable and available environment for Testing"

CA Service Virtualization

Who Benefits and How?

DEVELOPER

Eliminate Wait Time : Eliminate waiting on dependent systems to continue development , reproduce defects or fix a defect.

Enable Parallel Development : Eliminate waiting for other systems undergoing changes to provide “finished” services.

Eliminate Mocking/Stubbing : No more creation/maintaining of brittle stubs and mocks.

Improve Agility : Easily update virtual service behavior as requirements rapidly evolve.

Shift Left : The ability to create more test scenarios allow you to catch defects early in the development stages.

FUNCTIONAL TESTER

Eliminate Wait Time : No more waiting for downstream systems to start test cycles and verify defects.

Negative Test Coverage : Provide the ability to test application behavior for negative/ exception scenarios.

Provide Stable Test Environment : Minimize testing delays due to the lack of a stable middleware or backend environments impacting you ability to develop and execute tests.

INTEGRATION & PERFORMANCE TESTER

Reduce Infrastructure Costs : Eliminate the need to procure equivalent hardware/software for dependent systems to support production-like load on the system under test.

Eliminate Environment Sharing : The ability to isolate the system under test via virtual back ends eliminate environment sharing.

Scalability : Allow environments to scale to conduct performance, load and endurance test

Reduce Transaction Costs : Reduce the cost incurred due to the load supported by third party systems.

Traditional Service Virtualization

What's non-intelligent virtualization?

Service Request

```
{
    "header": {
        "headerAttributes": {}
    },
    "payload": {
            "postalCode": "94008",
            "currencyCode": "USD",
            "storeFrontId":"5534",
            "customerId": "8d10a7a3-dce8-1114-abe6-171353c8313a"
    }
}
```

Service Response

```
Content-Type: application/json
Server:  LISA VSE
Status: 200 OK

{
    "status":"OK",
    "errors":[],
    "header":null,
    "payload":
        {
            "cartId":"9e69d0e6-fe03-4d4f-b338-072410603bb7",
            "customerId":"8d10a7a3-dce8-1114-abe6-171353c8313a"
            "cartItemCount":0
        }
}
```

Traditional Service Virtualization

What's non-intelligent virtualization?

Traditional Service Virtualization:

- Write request & responses by hand
- Lookup data in the back-end data base
- Query TDM team to generate data we need
- No notion of coverage
- Challenging to maintain when the service changes

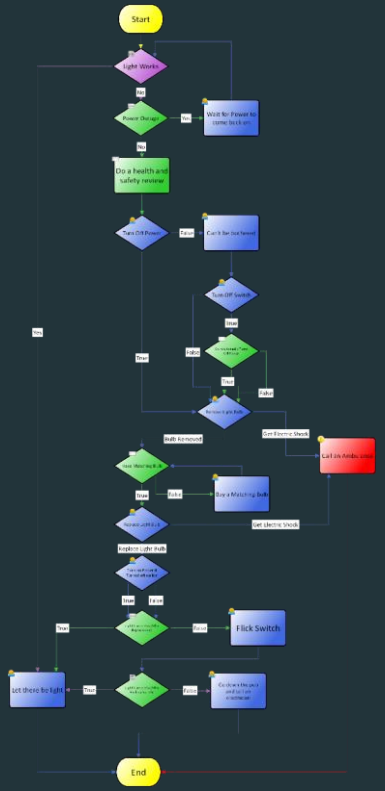
Intelligent Service Virtualization:

- Model the behaviour of the virtual service -> create the RR pairs for free
- Find / make the necessary backend data we require
- Perform data requests on-the-fly as needed
- Obtain data coverage
- When changes happen -> change the model and update our RR pairs

Agile Requirements Designer

Core of intelligent virtualization

- A formal model that is accessible to the business who already use VISIO, BPM, etc.
- Which is also a mathematically precise model of a system, so that it eliminates ambiguity and incompleteness
- It can be used by testers and developers – it brings the end-user, business and IT into close alignment

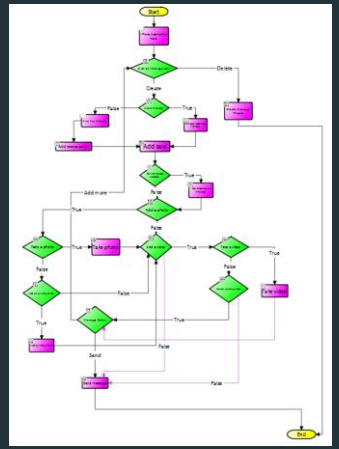


ca technologies

Agile Requirements Designer

Core of intelligent virtualization

- Exhaustively test a model – extract all possible routes from start -> end.
- Each route / path becomes a test case



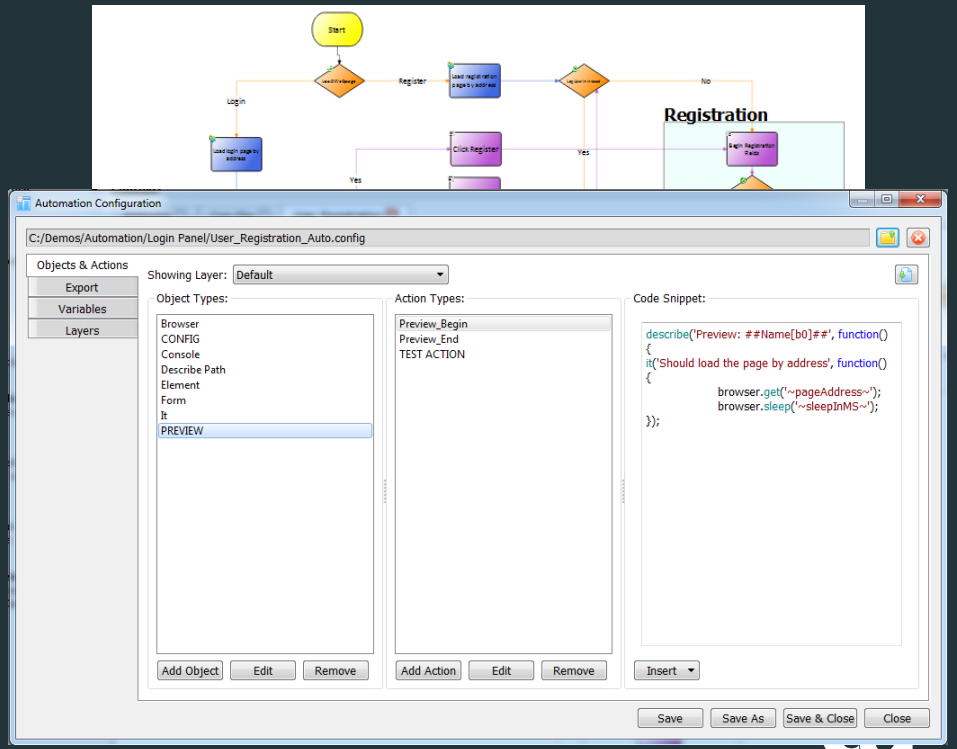
2145 Paths

- Optimisation techniques to create tests based on model coverage (nodes, edges, pairs)

Agile Requirements Designer

Core of intelligent virtualization

- Test automation engineers overlay code snippets onto processes which correspond to code (RR pairs).
- Automation scripts can therefore be automatically derived from it



Demo

Summary

- Traditional service virtualisation
  - RR pair generation is a manual process
  - Non-trivial task to align the right data
  - When tests change -> typically throw away RR pairs and start over
- Intelligent virtual services with CA Service Virtualization and CA Agile Requirements Designer:
  - Model the behaviour of the virtual service -> create the virtual service on-the-fly
  - Find / make the necessary backend data we require
  - Perform data requests on-the-fly as needed
  - Notion of data coverage
  - When changes happen -> change the model and update our RR pairs

Q & A

Huw Price
VP, App Delivery, Global QA Strategist
Huw.Price@ca.com

James Walker
Software Engineer
James.Walker@ca.com