Reduce Wait Time with Simulation + Test Data Management

How to approach test data in an agile world

Data is the key to business

How do you test a lock

- So many data combinations (So many Keys)
  - 9^5 combinations = 59,049
  - for a standard house key
  - Disnyland receives 44,000\day

- Data is complicated (I need all the keys)
  - August, Baldwin, Kwikset, Masterlock, Medco, Schlage, Yale
  - 413,343 combinations (Omaha = 411,630

- Data is Dangerous
  - (GDPR, PII, …)

Dev & Testing activities that need test data

Experimentation	• Playing with a new idea or capability
Unit	• GenerateData(complex) to test this.object
Functional Testing	• Large data sets for per mentation and non-nominal testing that still "Make Sense"
Integration Testing	• Introduce corrupt or unexpected data
Regression Testing	• Does todays data break the system
Non- Functional Testing - Performance Testing	• Data burning • Shift left performance testing

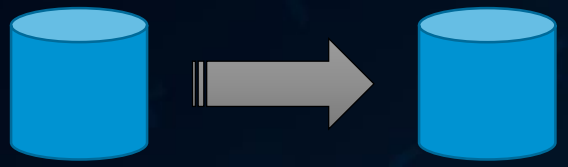
The increasing complexity of your data requirements

The Cost of Data Complexity

- Up to **60%** of application development and testing time is devoted to **data-related tasks**
- Many project overruns, 46% (cost) and 71% (schedule), due to **inefficiencies in test data provisioning**
- **20%** of average SDLC lost **waiting for data**
- System functionalities are not adequately tested, during continuous enhancements, due to required **test data not being available or created**
- Leads to **defects in production**

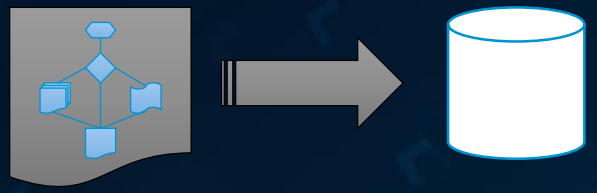
3x Traditional Approaches to TDM

1. Clone/Copy the production database

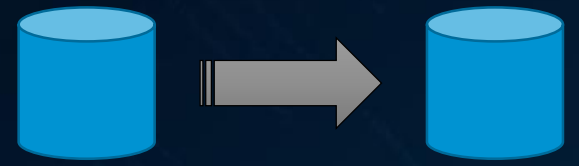


2. Subset/Sample the production database

3. Generate/Synthesize data



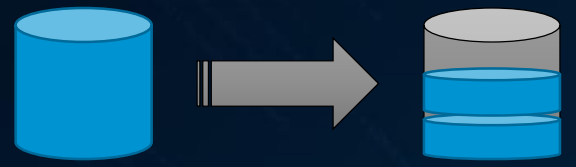
1) Clone/Copy the production database



- Pros:
  - Relatively simple to implement

- Cons:
  - Expensive in terms of hardware, license and support costs
  - Time-consuming: Increases the time required to run test cases due to large data volumes
  - Not agile: Developers, testers and QA staff can't refresh the test data
  - Inefficient: Developers and testers can't create targeted test data sets for specific test cases or validate data after test runs
  - Not scalable across multiple data sources or applications
  - Risky: data might be compromised or misused
  - **DO NOT FORGET TO MASK!!!**

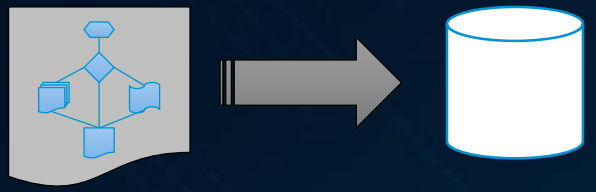
2) Subset/Sample the production database



- Pros:
  - Quick-win
  - Less expensive compared to cloning or generating synthetic test data

- Con:
  - Difficult to build a subset which maintains referential integrity
  - Skill-intensive: Without an automated solution, requires highly skilled resources to ensure referential integrity and protect sensitive data
  - Typically only 20-30% of functional coverage in production data
  - Dev/test spend 50-70% of time looking for useful data (20% of the SDLC cost)
  - Requires underlying database infrastructure
  - **DO NOT FORGET TO MASK!!!**

3) Generate/Synthesize data



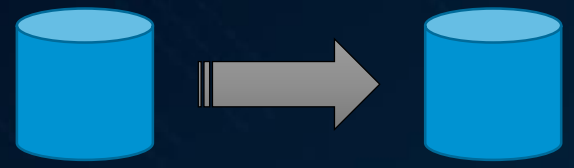
- Pros:
  - 100% functional coverage without the need to mask data
  - Does not contain sensitive/real data
  - Model data relationships + test requirements = complete set of data

- Cons:
  - Needs knowledge to 'design'/model the data
  - Requires underlying database infrastructure
  - Resource-intensive: Requires DBA and Domain experts to understand the data relationships
  - Tedious: Must intentionally include errors and set boundary conditions
  - Challenging: Doesn't always reflect the integrity of the original data set or retain the proper context

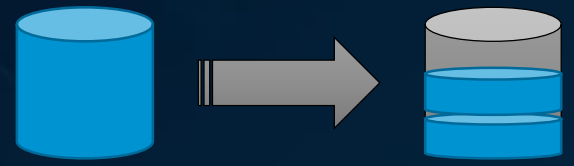
Test Data Modeling

3x Traditional Approaches to TDM

- Clone/Copy the production database
  - Expensive and time consuming



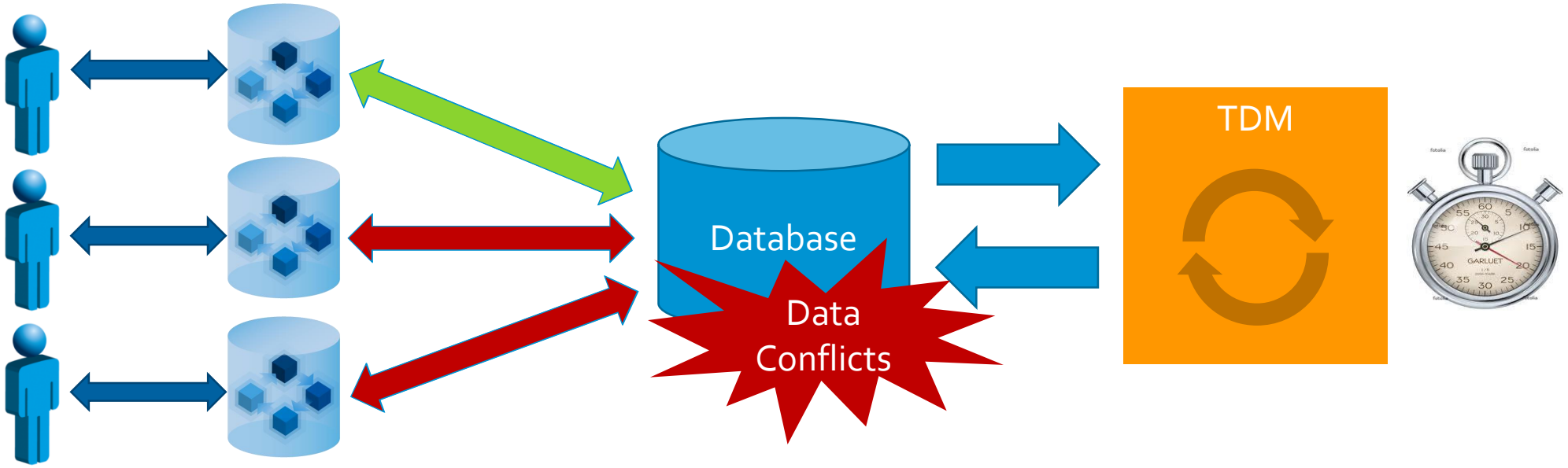
- Subset/Sample the production database
  - Difficult to build a subset which maintains referential integrity



- Generate/Synthesize data
  - Requires DBA and domain experts to understand the data relationships

… but there is a problem with the traditional approach

Reliance on a shared database



1. Multiple teams using the same test database
2. TDM solution takes time and resources
3. Teams not respecting data integrity or other team's test data records
4. Regression tests consistently failing.

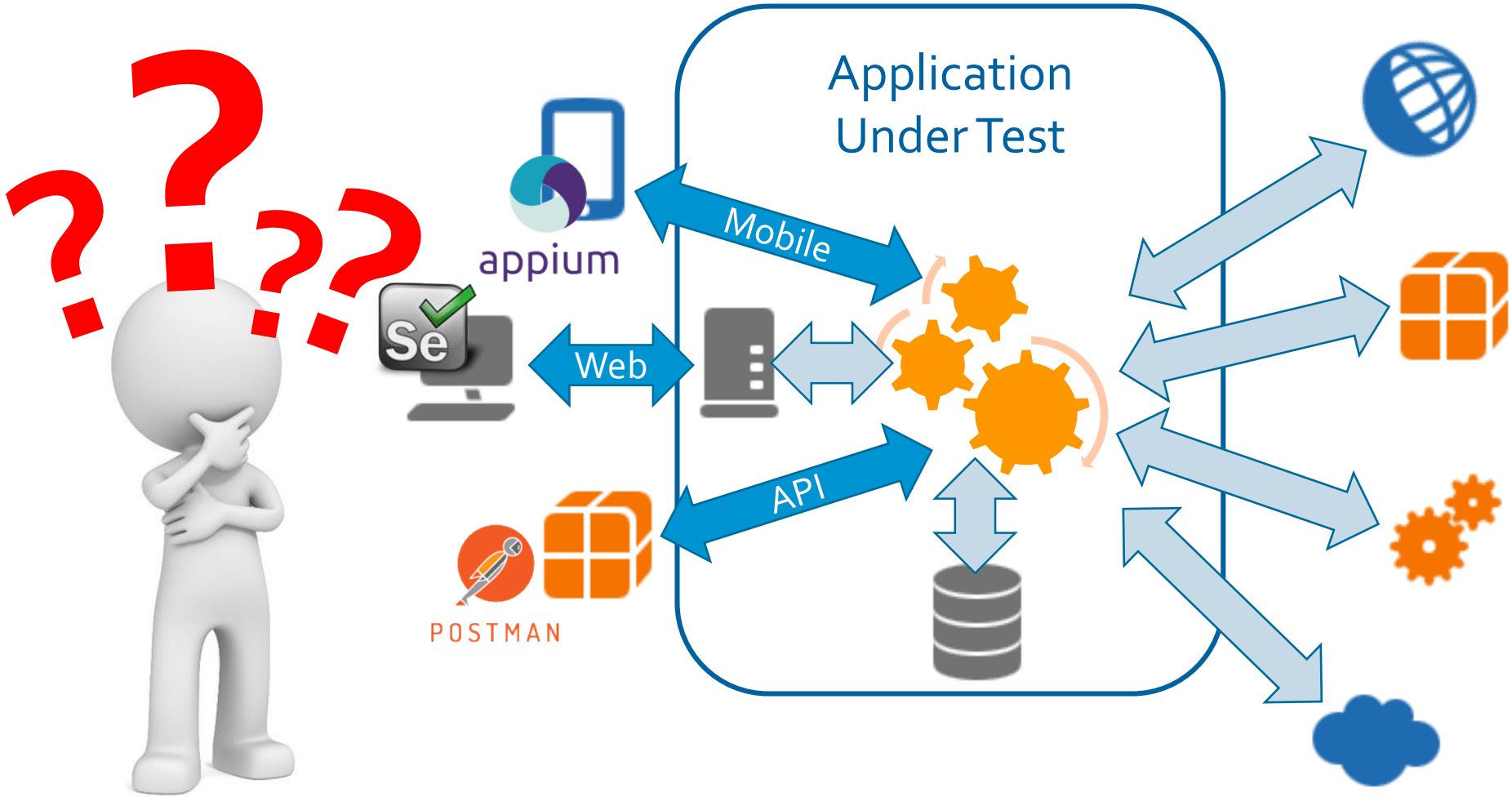
"Takes hours to determine that it was due to data changes".

"Real problems are getting lost in the noise"

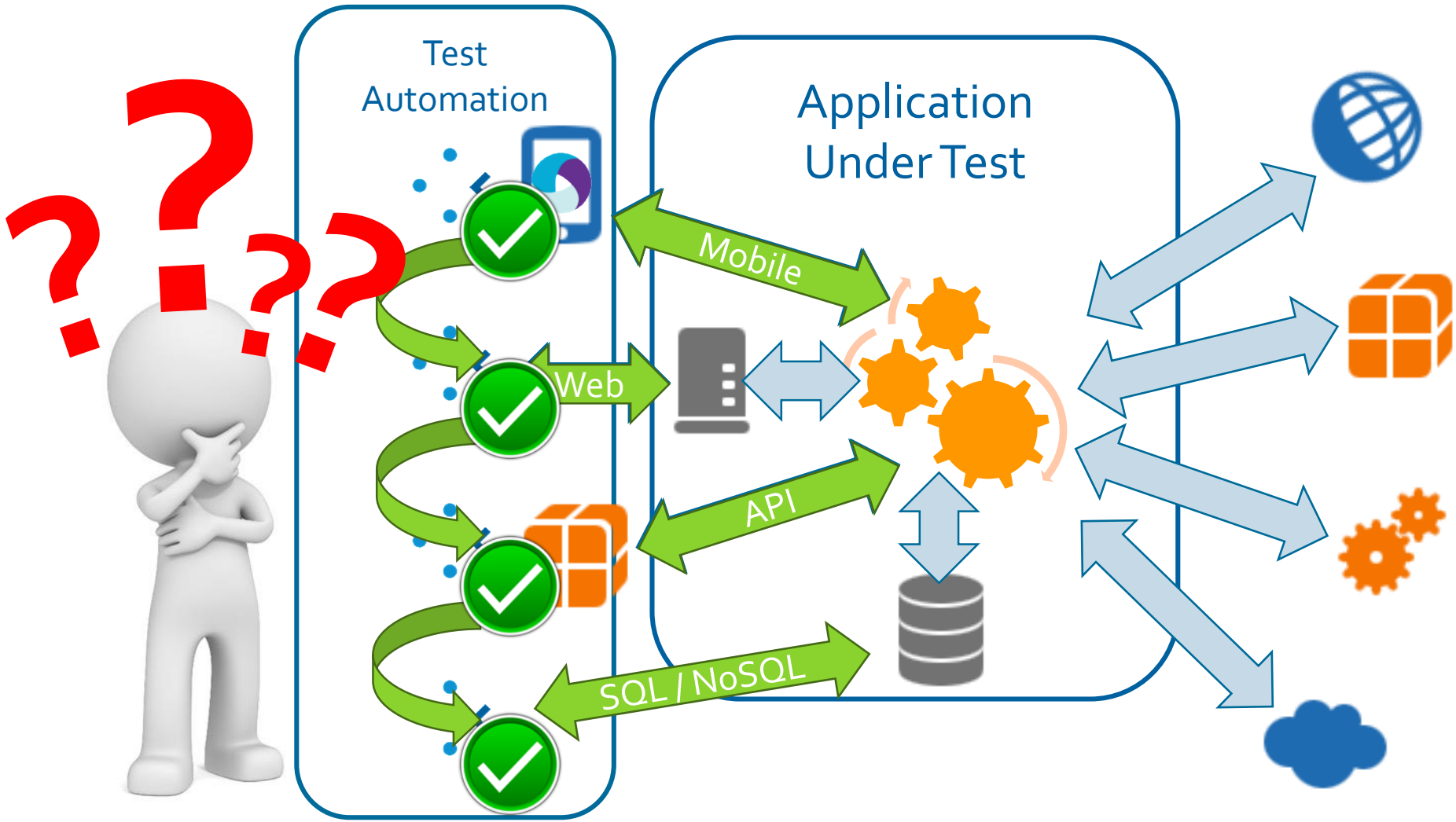
Option #4 … Service Virtualization

delivers a
simulated dev / test environment
allowing an organization to test
anytime or anywhere

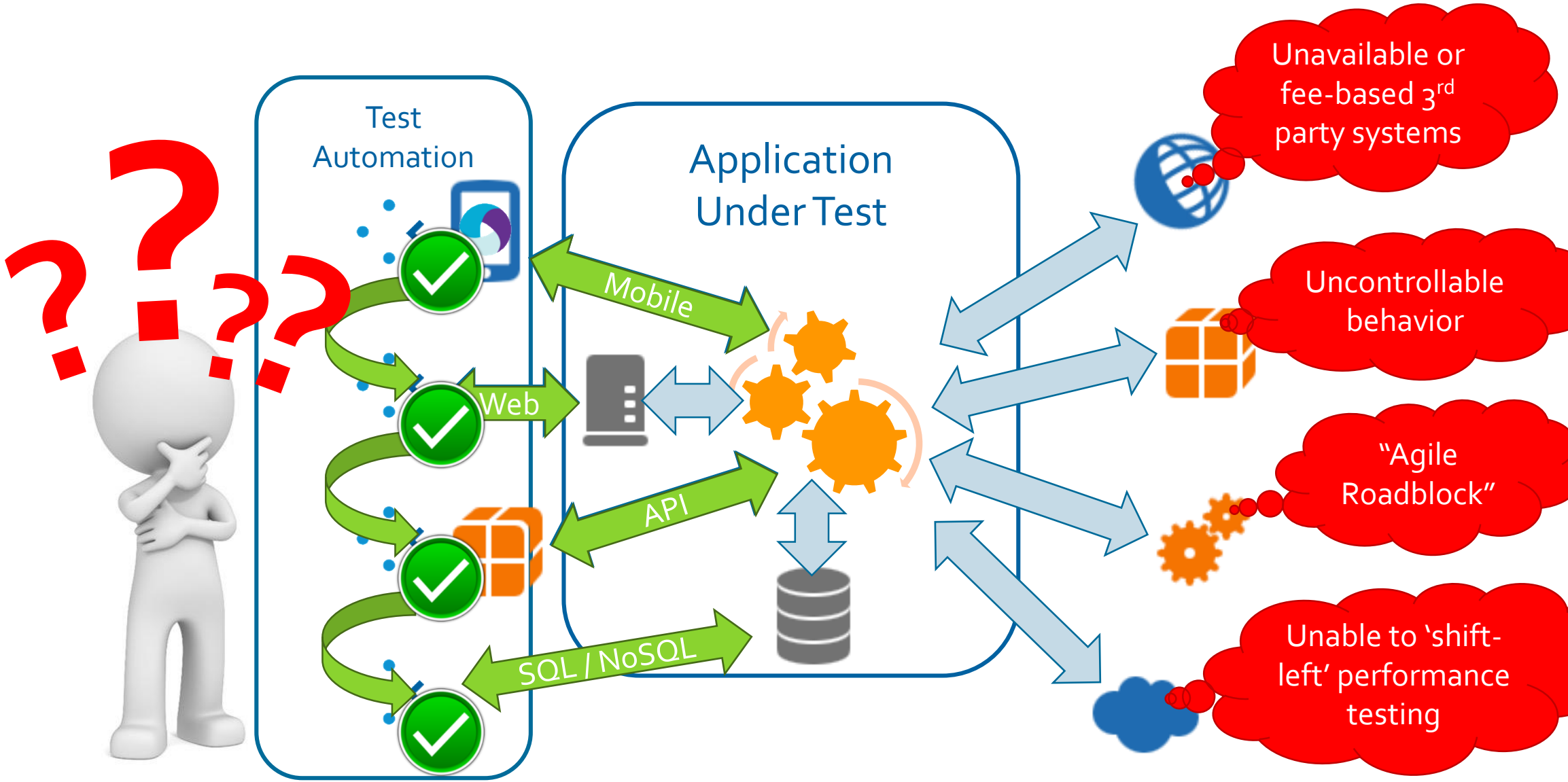
Increasing complexity of testing requirements



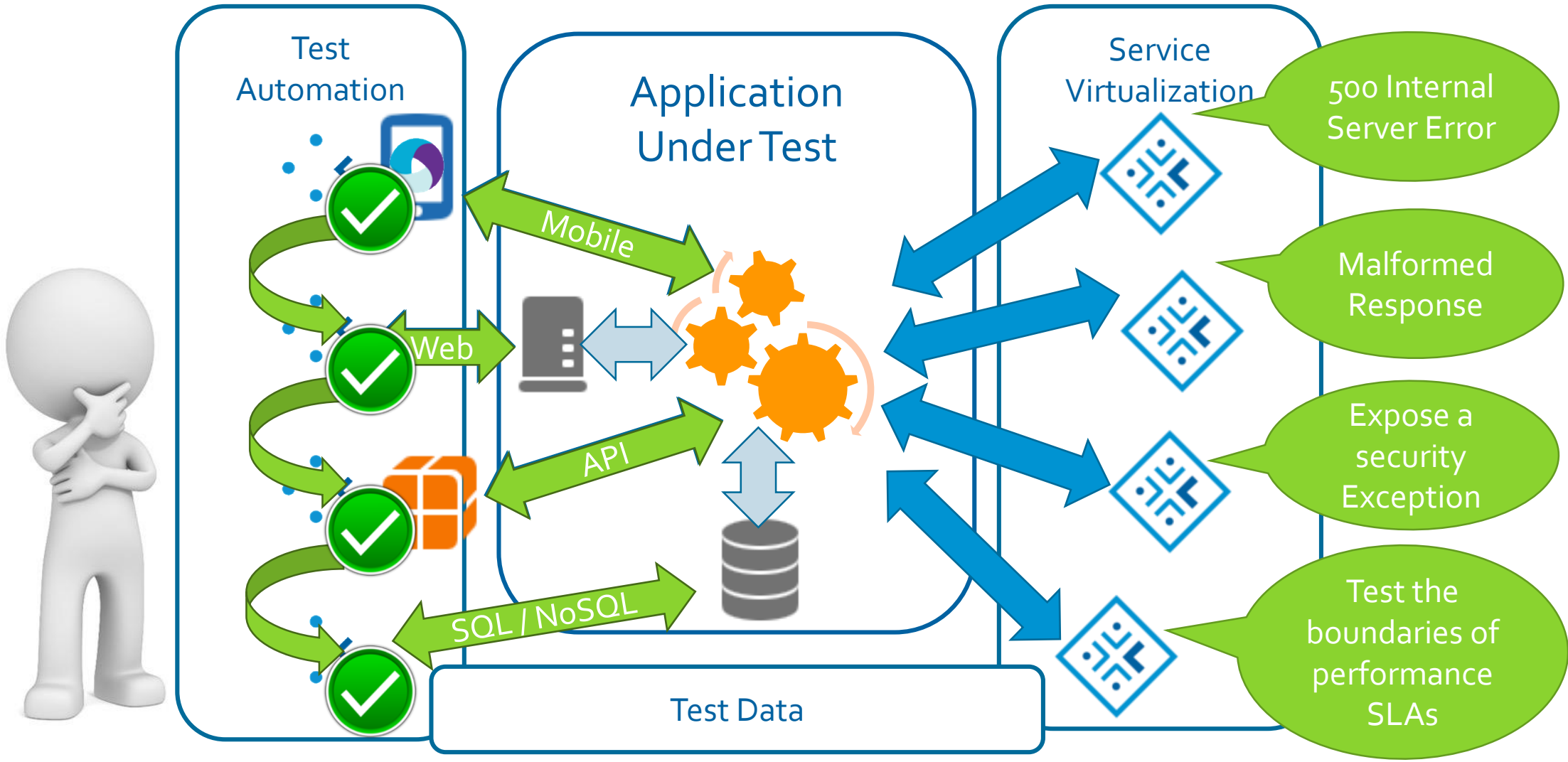
Omni/Multi-Channel Test Automation



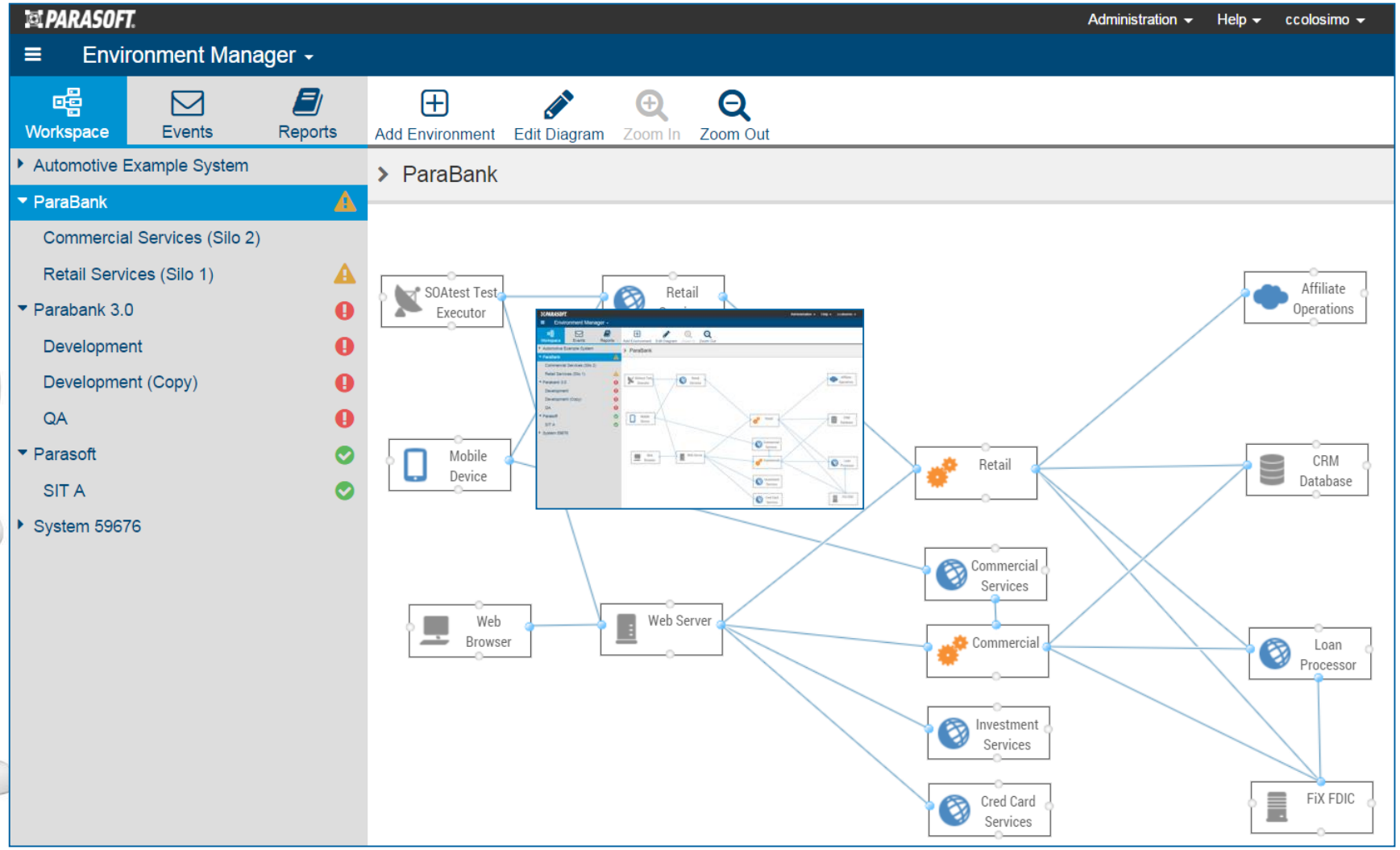
Omni/Multi-Channel Test Automation



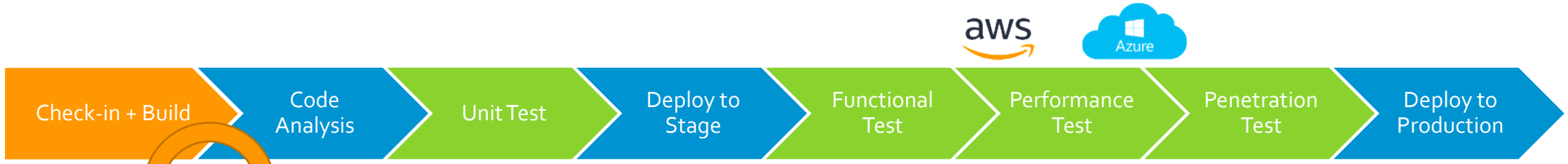
Total control of the Test Environment



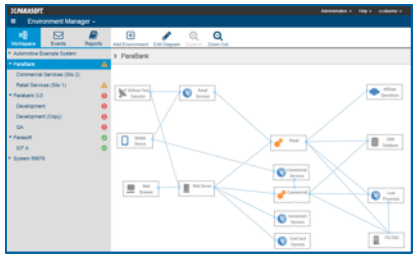
Environment based approach to testing

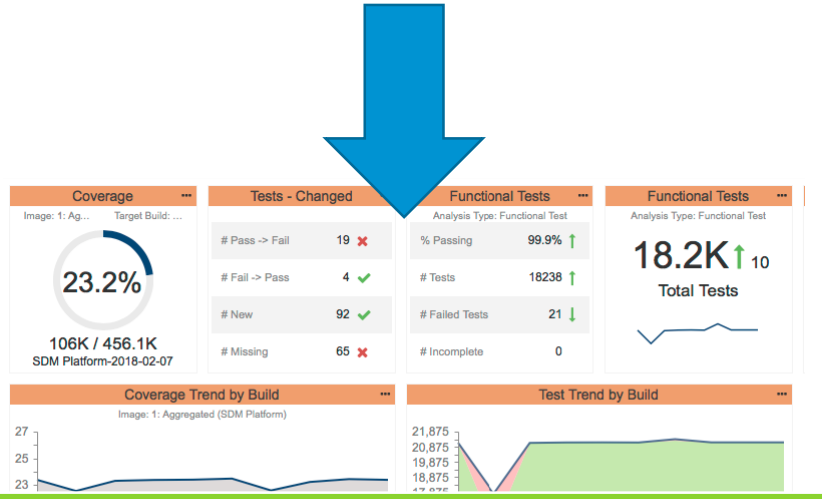


Enabling Continuous Quality in the CI/CD Pipeline



docker

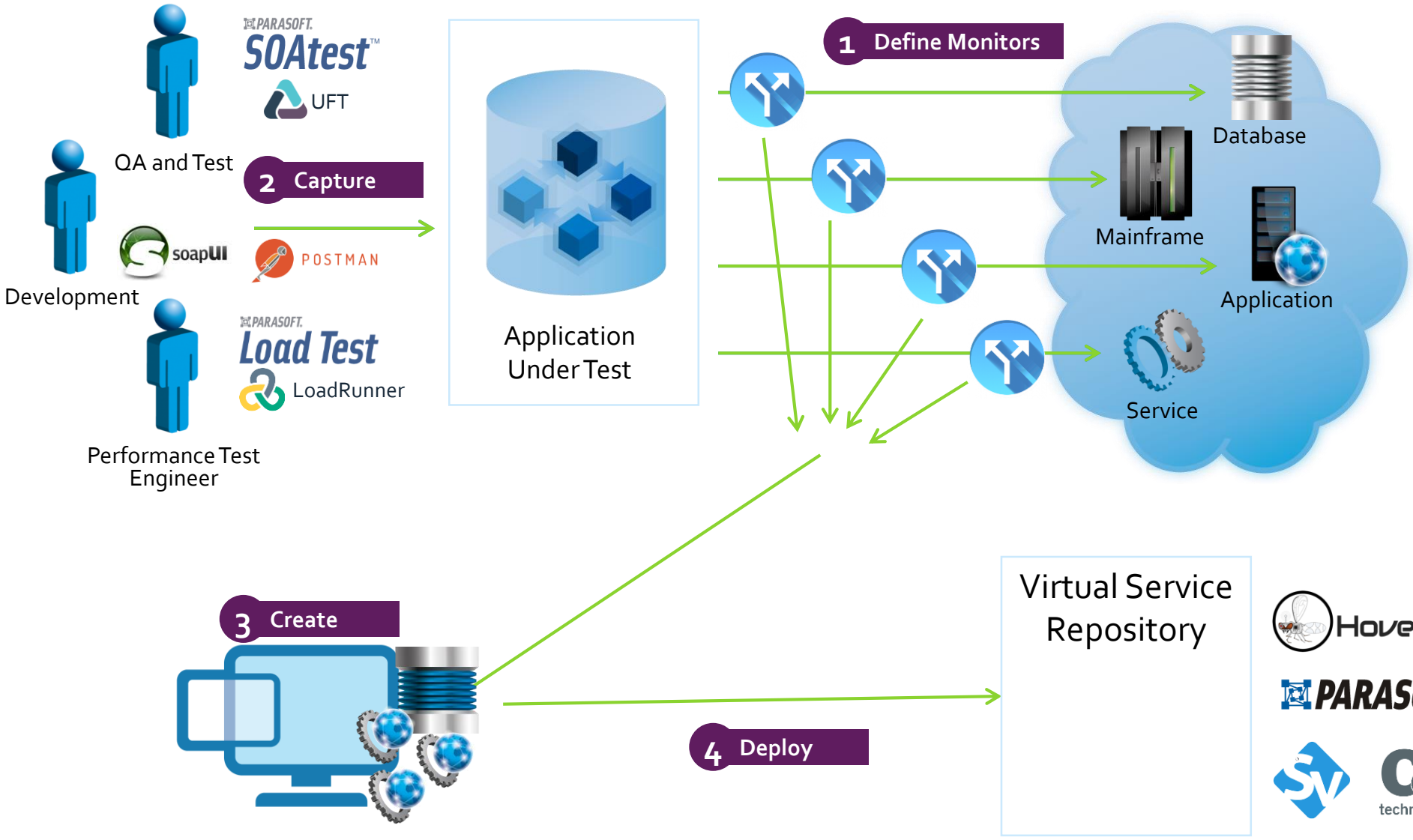




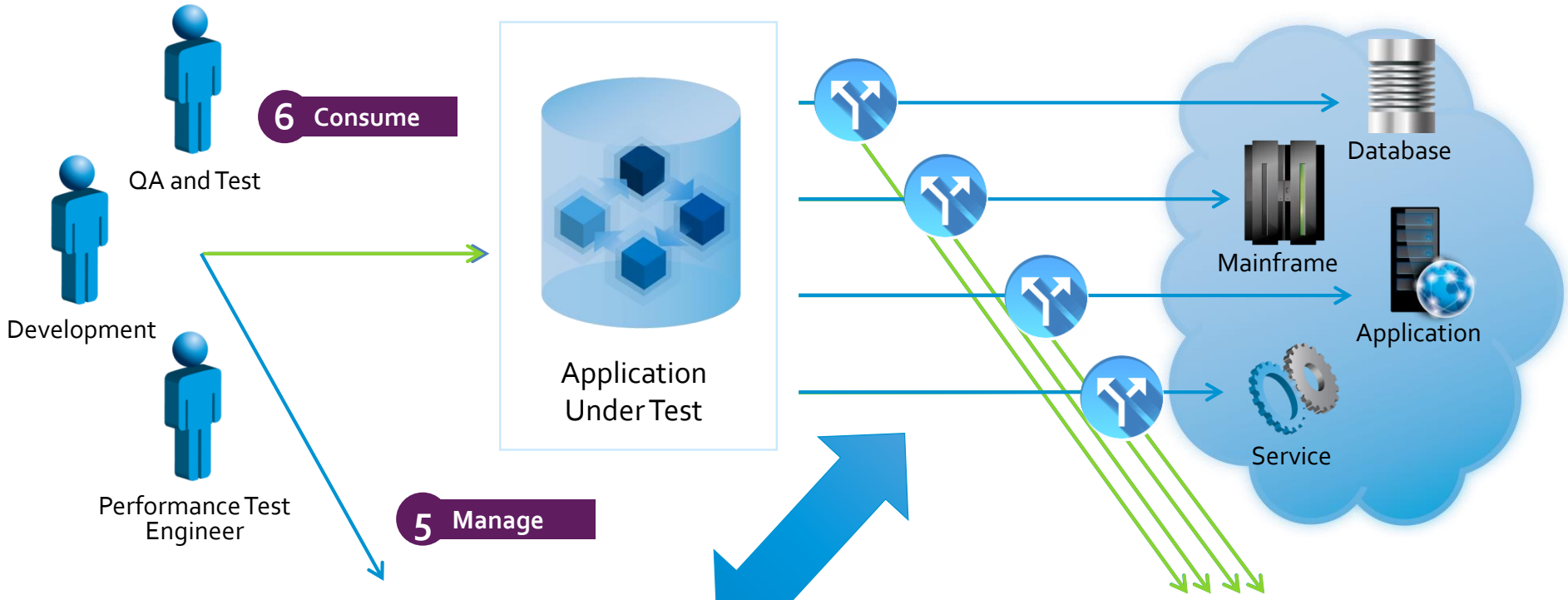
Bamboo

Combining tests, virtualize assets, and data into disposable test environments to enable complete test coverage

Service Virtualization: Capturing current behavior



Service Virtualization: Capturing current behavior



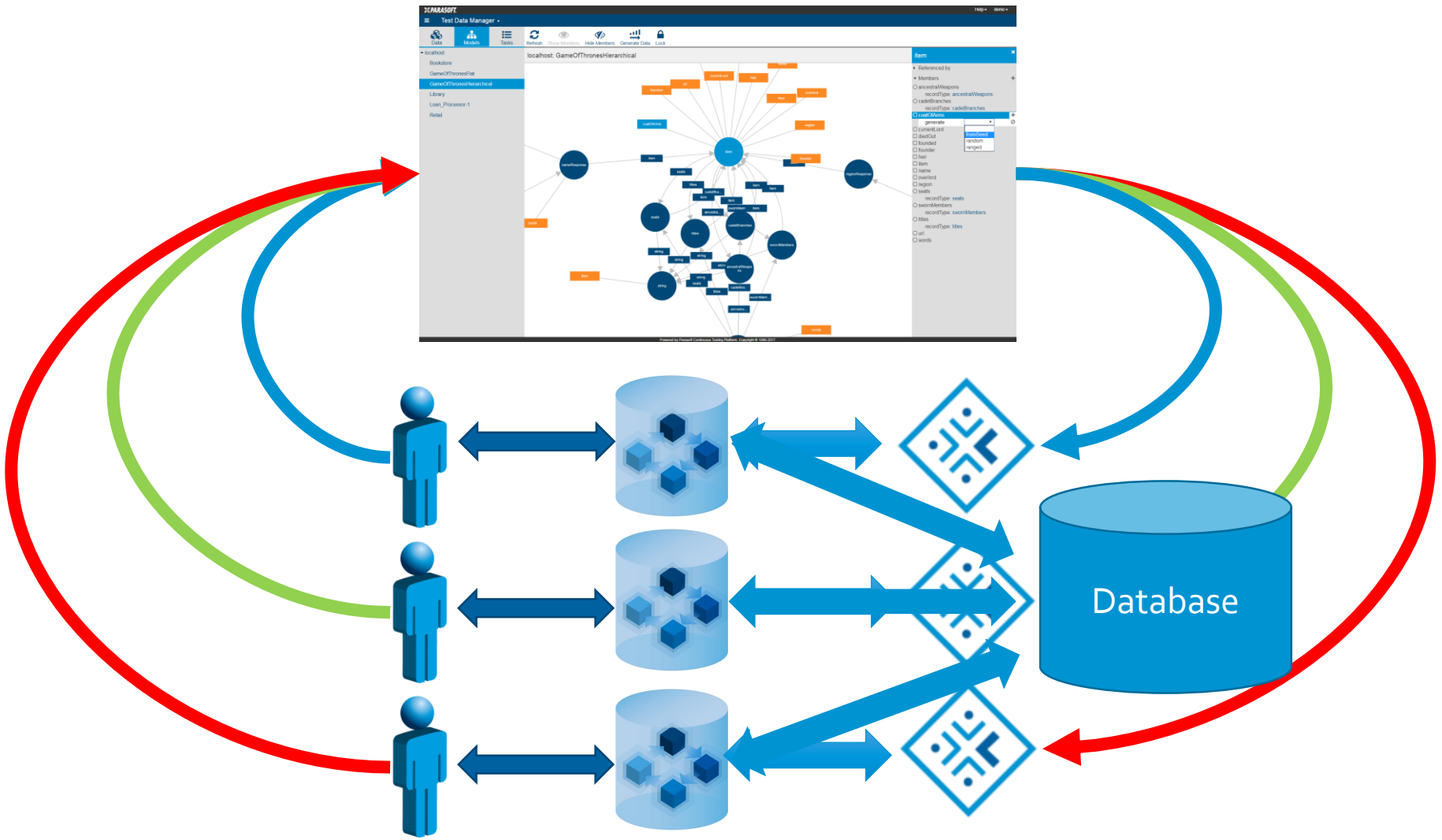
Jenkins
Bamboo
TeamCity
IBM
Rational
MICRO FOCUS
QC/ALM
Visual Studio Team Services

DevOps Platform

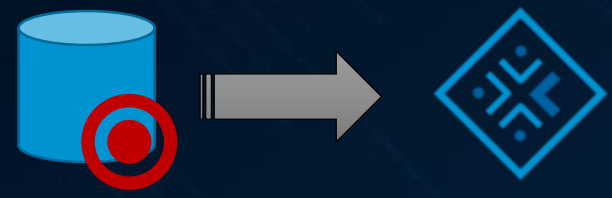
Virtual Service Repository

Hoverfly
PARASOFT
SV
ca technologies

Service Virtualization + Test Data Management



4) Service Virtualization



- Pros
  - Does not require underlying database infrastructure
  - Isolated test environments
  - Easily cover corner cases
  - Ease to share
  - Eliminates complexity of underlying database schema
  - Capture just the data you need to … and dynamically mask

- Cons
  - It's not a real database … virtualizing of INSERT/UPDATE scenarios increases complexity

Combining Service Virtualization with traditional TDM

Service Virtualization

Test Data Management

Simulate database interactions for "SELECT" operations and performance/corner-case scenarios

Model the data relationships and generate for expanded coverage and disposable test data

Subset and Mask existing data and leverage database infrastructure for "INSERT"/"UPDATE" operations

Test Data Lifecycle

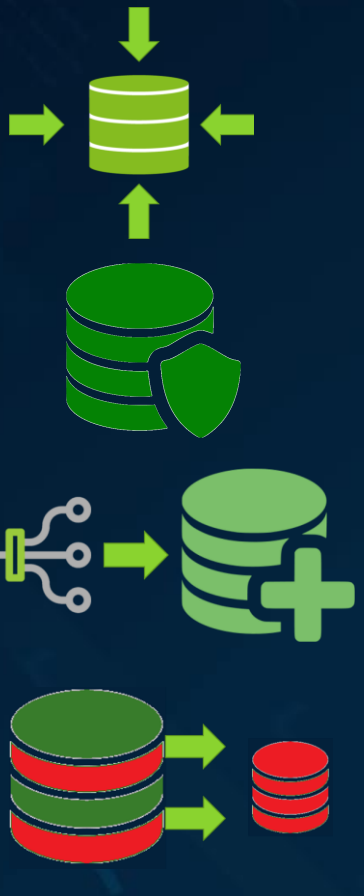
Make reusable data a reality with simple and intuitive workflows

Management
- Capture, Navigate, Edit, Snapshot

Masking
- Ensuring existing data is safe for use in testing environments

Model/Generation
- Extend and reshape the data you have for additional value

Sub-setting
- Carving out specific data sets from the now, abundance of data available



Capturing and Managing Test Data

How do you get your data into the testing infrastructure?

- What are my test data requirements?
- What Data can I capture
  - Database extraction
  - In use (Over the wire)
- Post Capture
  - Masking, Subsetting
- What tools exist
  - Wireshark, Fiddler, CA LISA, Parasoft Virtualize, HPSV, Charles Proxy, APM tools (Dynatrace, Appdynamics)

Masking Sensitive Data

Once we get Data into the testing Infrastructure, How much risk have we introduced

- Can we use the data we have?
- What can we do to remediate our risk

- Masking
  - Ensuring existing data is safe for use in testing environments

- What tools exist
  - Scripting, Arx, Jailer, Metadata Anonymization Toolkit, Talend, DatProf, CA TDM , Parasoft Virtualize, HPE Security IBM Optim, Informatica, Oracle Data Masking, MasterCraft

Don’t forget to mask the data

- Protects against unintended misuse
  - Privacy concerns, sensitive corporate and regularity requirements (HIPPA, PCI, GDPR)
- It’s not as a simple “XXXX” or scrambling values
  - 354-15-1400 > XXX-XX-XXXX
  - 354-15-1400 > 004-15-1453
- Need to consider
  - Validity and format of the data
  - Multiple copies of the same data need to be masked the same way
  - How is the masked data is used
    - Related or derived values; 354-15-1400 vs 1400 (i.e. last 4 digits)
    - Manipulated/changing data cannot be masked if validation is required

Expanding Data Coverage

How useful is your data

- Stagnate, obsolete, burned
- Limited data reusability due to uniqueness constraints
- Repurposing data

- Model/ Generation
  - Extend and reshape the data you have for additional value
  - Seed data

- What tools exist
  - Mockaroo, Data Factory, Spawner, Databene Benerator, The Data Generator, Toad, Open ModelSphere, Parasoft Virtualize, DatProf, IBM Infosphere, CA TDM, NORMA, DB Tools (SQL Server Management, MySQL, Erwin)

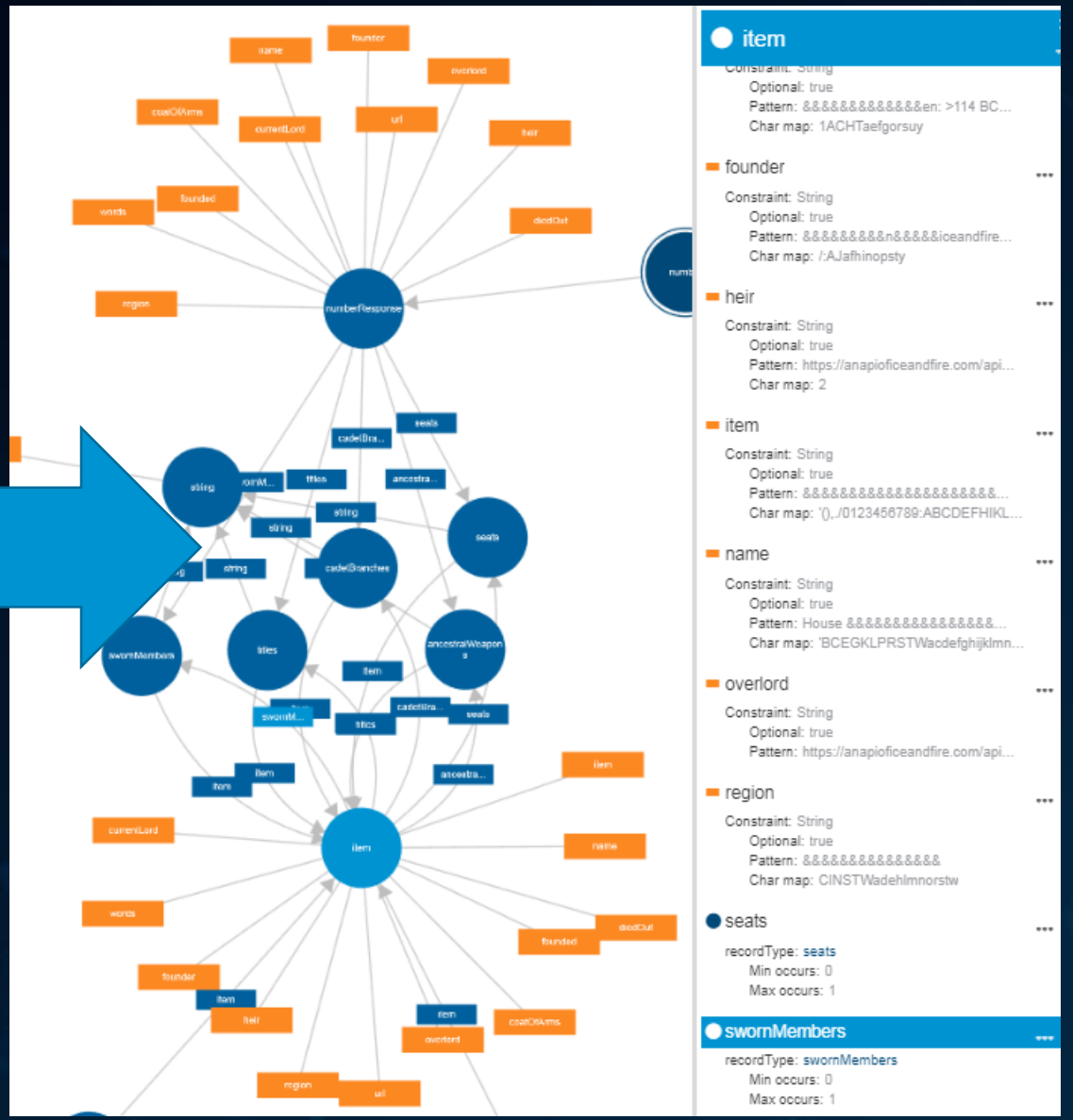
Data Warehouse: GameOfThrones

Data Sets

name

	Row #	name
	4	House Greyjoy of Pyke
	5	House Lannister of Casterly Rock


```
▼ array [1]
  ▼ 0 {4}
      _id : 593b376497292abc79f8b924
    ▼ item [1]
      ▼ 0 {18}
          _id : 593b376497292abc79f8b929
        ▼ swornMembers [1]
          ▶ 0 {3}
          founder : https://anapioficeandfire.com/api/characters/615
        ▼ cadetBranches [1]
          ▶ 0 {3}
          words : Hear Me Roar!
          founded : Age of Heroes
        ▼ titles [1]
          ▼ 0 {3}
              _id : 593b376497292abc79f8b92d
            ▼ string [3]
              ▼ 0 {2}
                  _id : 593b376497292abc79f8b930
                  item : King of the Rock (formerly) Lord of Casterly Rock
              ▶ 1 {2}
```

Finding the Right Data

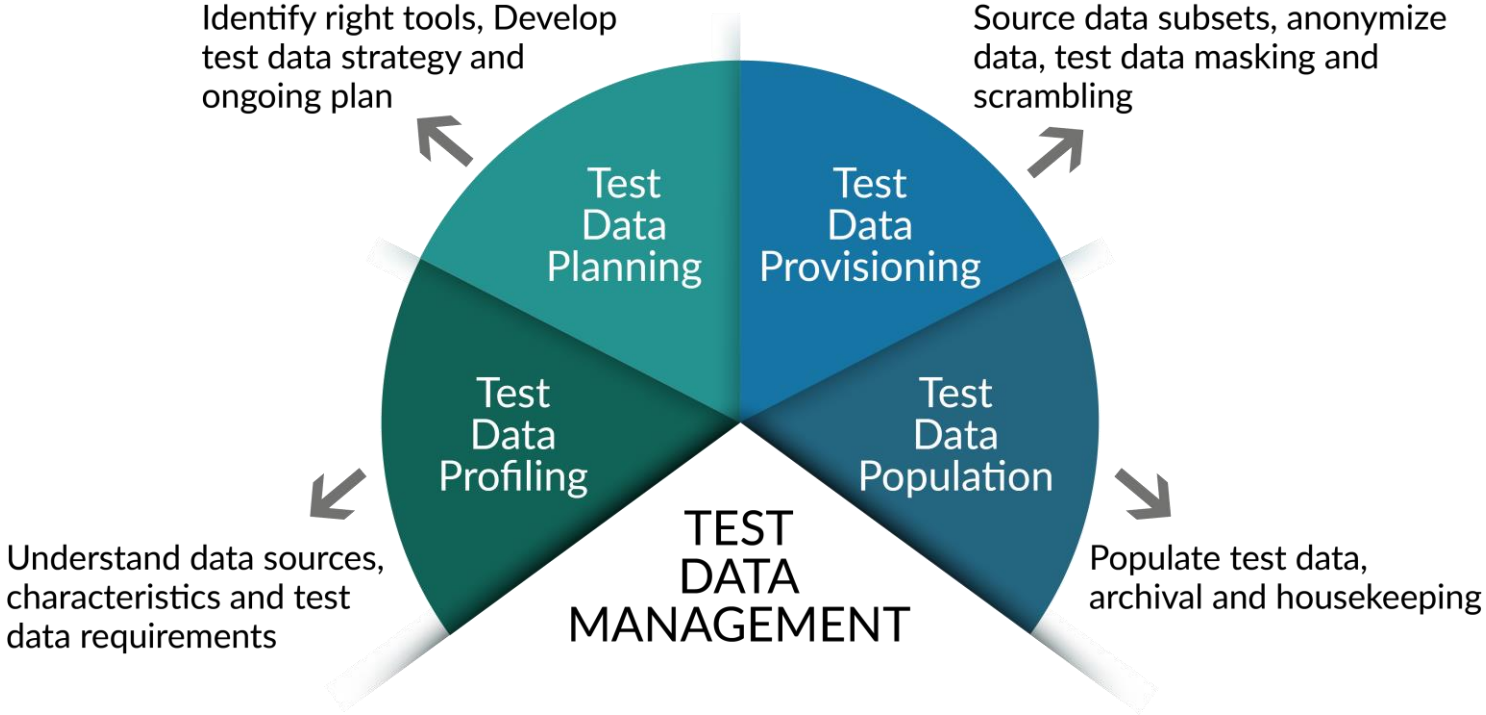
How do you filter the data you have amassed

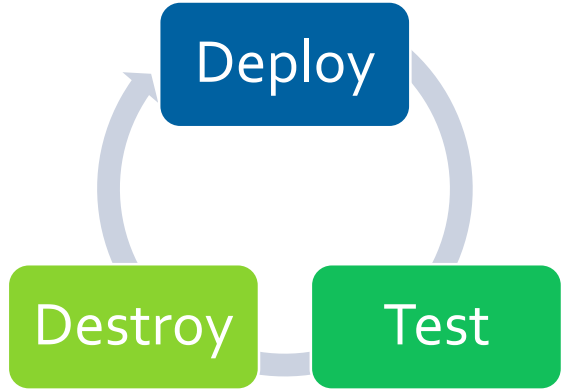
- Pull select data from a library to satisfy your unique testing requirements
- A good problem to have

- Sub-setting
  - Carving out specific data sets from the now, abundance of data available

- What tools exist
  - Db Tools, Scripting, DatProf, CA TDM, Parasoft Virtualize, Delphix, HPE Security, IBM Optim, Informatica, Oracle Data Masking, MasterCraft

Test Data Management Lifecycle





http://qatesting.rsystems.com/quality-engineering/

How Service Virtualization Helps

SV is the best data management technique for Agile

- Simplifies the TDM problem
  - Reduces back-end data requirements
  - Data-Graphs vs. Relational-Data
    - Scalable, Fast, Efficient dynamic storage
  - Removes complex table/key relationships

- Link Service Virtualization and Automated Testing together to close the loop
  - Link the data on the front end to the back end
  - More predictable, controllable data scenarios
  - Note: Any data validation should be to validate the AUT not the back-end behaviour. Data validation of shared data will be different in system test.

Conclusions

"Getting the right keys"

- Combination of Service Virtualization and TDM offers a simplified approach to data management
  - Capture > Mask > Model > Generate > Subset
  - Don't forget to Mask for privacy compliance

- Utilize Service Virtualization to 'shift left' integration testing
  - Share data between Test tools and Service Virtualization layer to fully test the AUT (not constrained by the back-end system)
  - Utilize simple data storage rather than 'full schemas' for rapid/agile prototyping

- Create different data sets for different purposes
  - Different use-cases scenarios (positive/negative)
  - Different types of testing (e.g. functional vs. performance)

LEARN MORE ABOUT PARASOFT'S

TECHNOLOGIES FROM THE FUTURE

REDUCE WAIT TIME WITH SIMULATION AND TEST DATA MANAGEMENT

October 3rd, 11:30am - 12:30pm (Monorail)

THE STATE OF API TESTING: NEW INDUSTRY SURVEY RESULTS HIGHLIGHT THE CHALLENGES OF TEST AUTOMATION

October 4th, 7:15 - 8:15 am (Magic Kingdom 1)

BACK TO THE FUTURE OF TEST AUTOMATION: TECHNOLOGIES FROM THE FUTURE THAT YOU CAN USE TODAY

October 4th, 11:15am - 12:00pm (Grand Ballroom)

PARASOFT®

Want more information?

Chris Colosimo

chris.colosimo@parasoft.com

Visit our booth # 16