

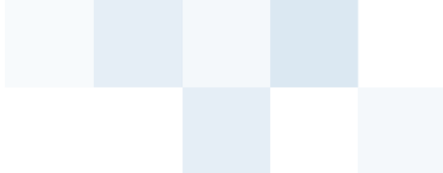


Effective Test Data Management in End-To-End and API Testing in Agile environment.

Excel isn't a long-term solution

Mark Lambert (VP Products and Support)
mark.lambert@parasoft.com

12/4/14

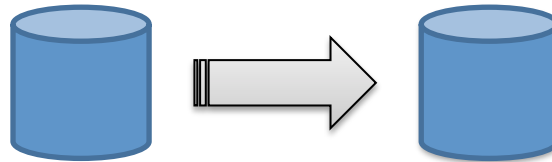
- 
- A decorative graphic consisting of a grid of squares in various shades of blue and white, arranged in a pattern that tapers to the right.
- The Test Data Management Problem
 - TDM Approaches
 - Data Masking
 - Using Service Virtualization to help with TDM
 - Conclusions

- 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**

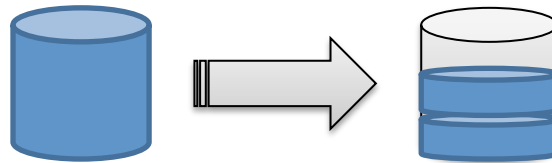
- “Significant time and delays due to acquiring, validating, organization and protecting test data”
 - 42 back-end systems with complex interdependencies
 - Lots of overruns due to lack of data readiness
 - 15% added to all schedules for time associated with TDM
 - Resulted in 10% of tests not executed because of missing data
 - Testing in production, leading to defects found in production

- 21% of project = hidden cost associated with TDM

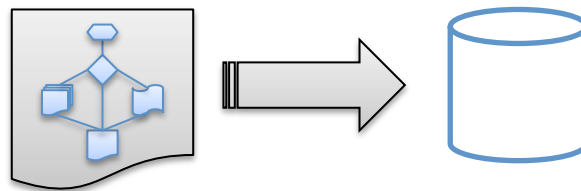
1. Clone/Copy the production database



2. Subset/Sample the production database



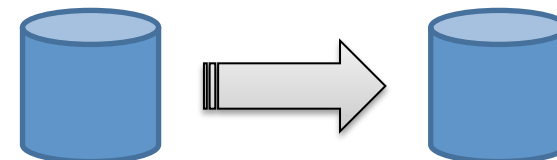
3. Generate/Synthesize data



1) Clone/Copy the production database

- Pro:

- Relatively simple to implement



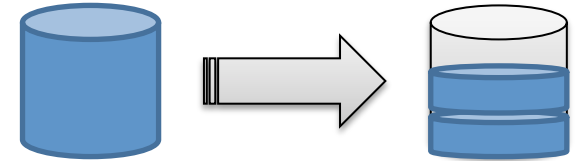
- Cons:

- 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

- Pro:

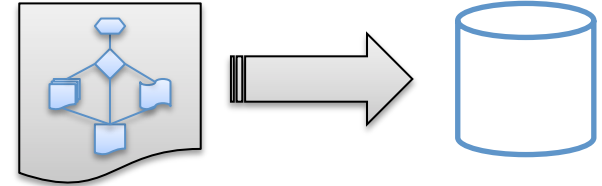
- Quick-win
- Less expensive compared to cloning or generating synthetic test data



- 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
- Dev/test spend 50-70% of time looking for useful data (20% of the SDLC cost)
- **DO NOT FORGET TO MASK!!!**

3) Generate/Synthesize data

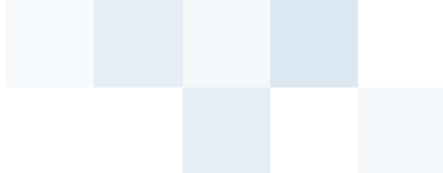


- Pro:

- 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
- 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

- 
- A decorative graphic consisting of several light blue squares of varying sizes arranged in a grid-like pattern, located in the upper left area of the slide.
- Combination of Sub-set and Synthesis
 - Quick hit of sub-set data
 - Use tools for automated extraction of related data
 - Mask data post-extraction
 - Use synthesis to generate large volumes
 - Create multiple datasets for different purposes

A decorative graphic consisting of several light blue squares of varying sizes arranged in a grid-like pattern in the upper left corner of the slide.

■ Functional Testing

- Extract a subset of production act as input values for data driven testing
- Use test and data modelling to generate corner cases

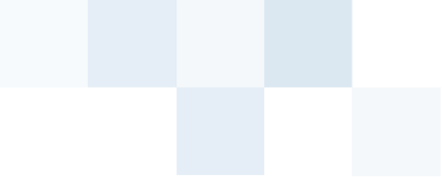
■ Performance

- 100's-1000's of test data over hours of execution
- Use Synthesis to generate large volumes

- Protects against unintended misuse
 - Privacy concerns, sensitive corporate and regularity requirements (HIPPA, PCI)
- 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

- Manage the data
 - The data becomes an asset
 - Backup/Versioning Strategy
 - Update/Keep in-sync with production
 - Snapshot-rollback as part of the automated test process

- Test and Requirements Modelling
 - Model the test or data
 - Reduce the number of tests/volume of data to cover critical paths
 - Example Vendors; GridTools, ConformIQ



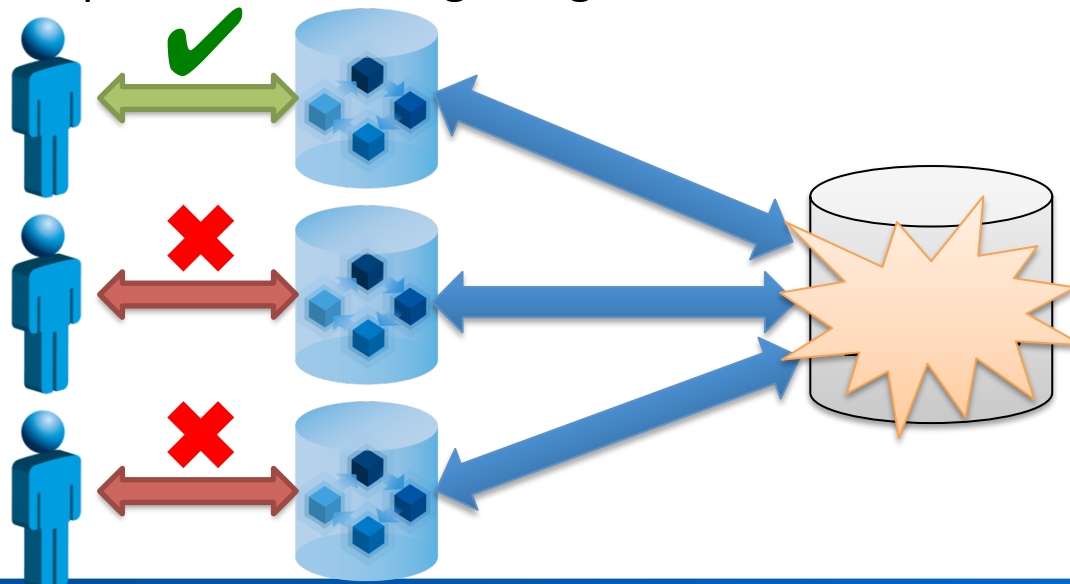
Service Virtualization delivers a simulated dev / test environment allowing an organization to test anytime or anywhere



Test data management for complex transactions

■ The Challenge

- Multiple teams using the same test database
- Teams not respecting data integrity or other team's test data records
- Regression tests consistently failing. Takes >1 hour to determine that it was due to "data changes".
- "Real problems" were getting lost in the noise

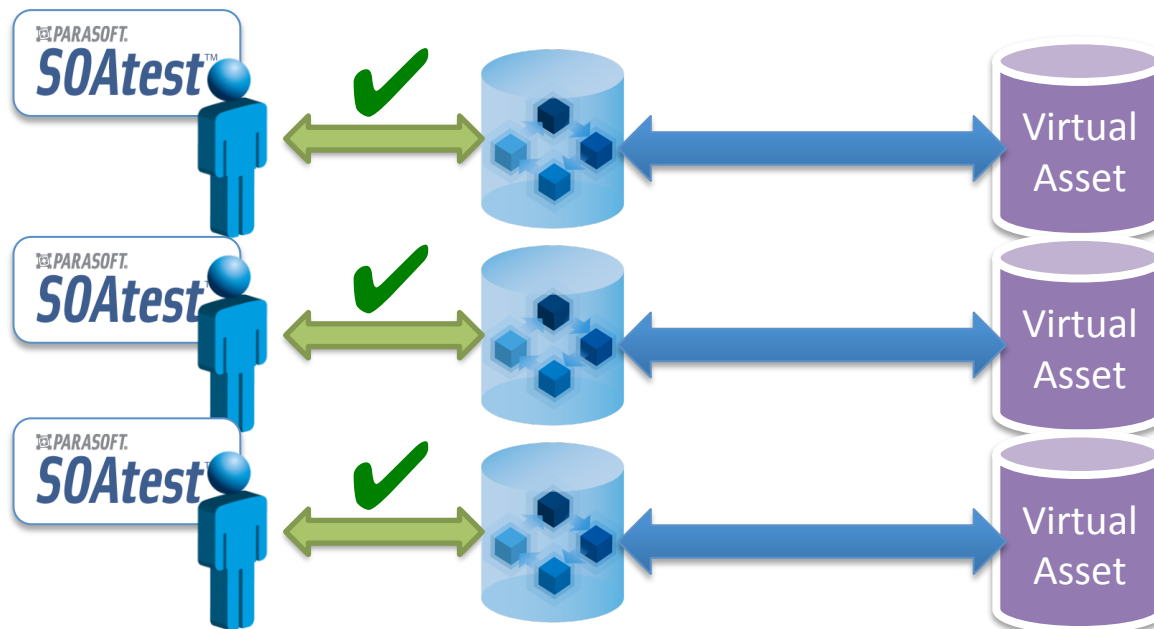




Test data management for complex transactions

■ The Solution

- Setup Virtual Assets to model the SQL queries and use SOAtest to manage automated nightly regressions against both virtual assets and live systems

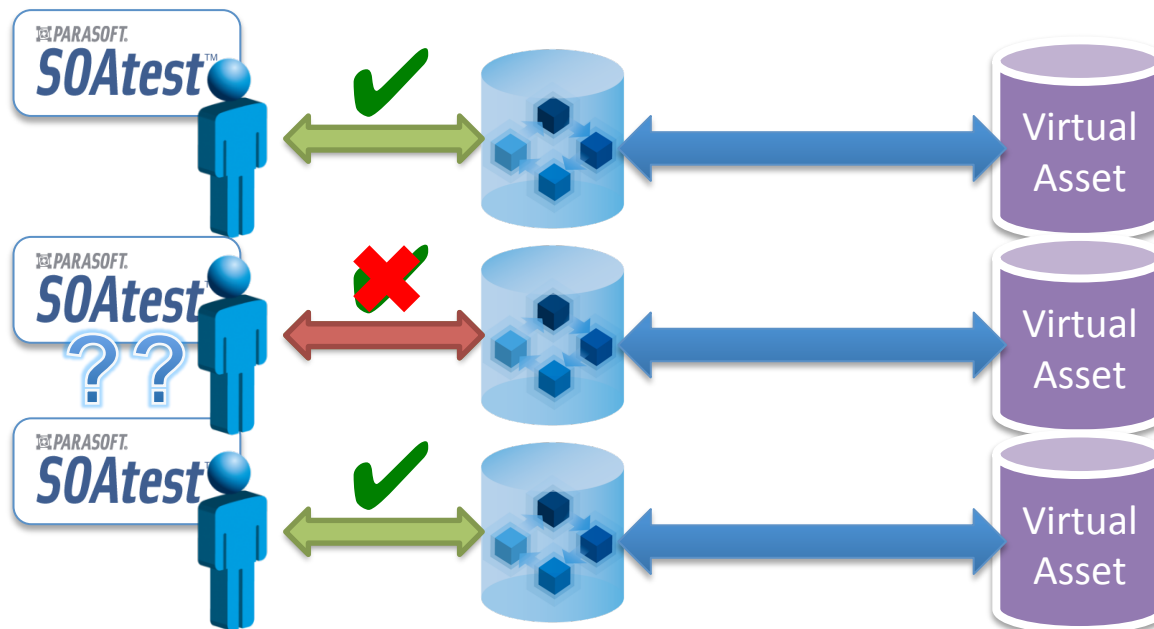




Test data management for complex transactions

■ The Business Benefit

- Test teams able to focus on 'real regressions' and separate out data integrity issues from functional test failure



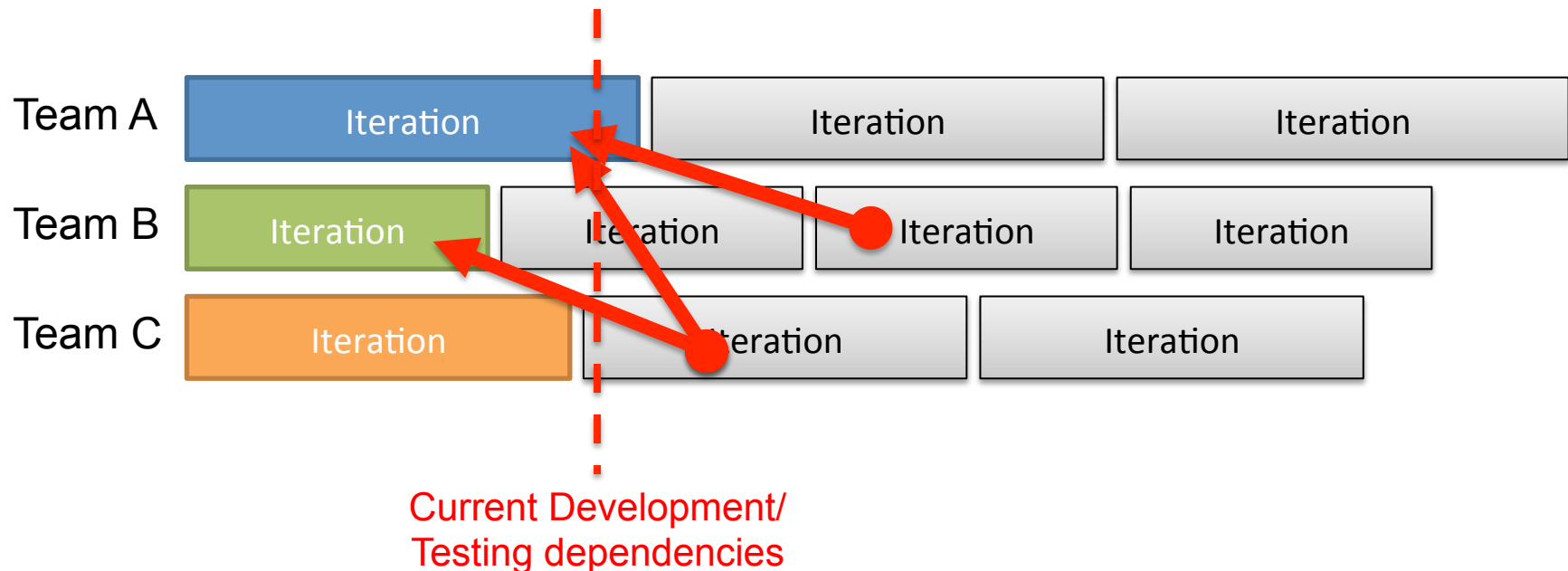
Reduced wait time for test team by 60% for a major media conglomerate



Agile/Parallel development limited by system dependencies

■ The Challenge

- Large agile development effort to adopt Service Oriented Architecture (SOA)
- High risk project but the Test team “stuck waiting for the first build”
- Development of functionality was not easy to coordinate as different teams had different schedules; not all finished at the same time



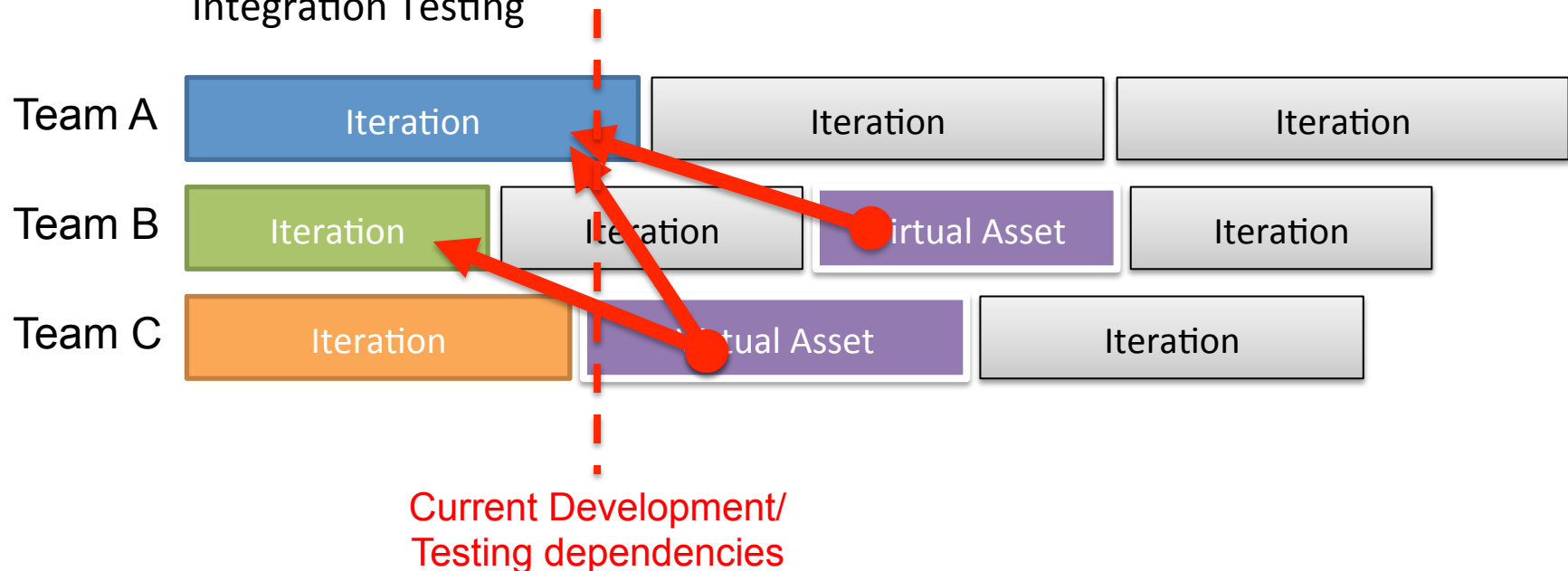
Reduced wait time for test team by 60% for a major media conglomerate



Agile/Parallel development limited by system dependencies

■ The Solution

- Use descriptions of the new services (WSDL, XSD, example JSON payloads) to build Virtual assets prototyping the new functionality.
- Test team builds tests with against the prototypes with SOAtest and the independent development tests use the prototypes to perform early stage Integration Testing



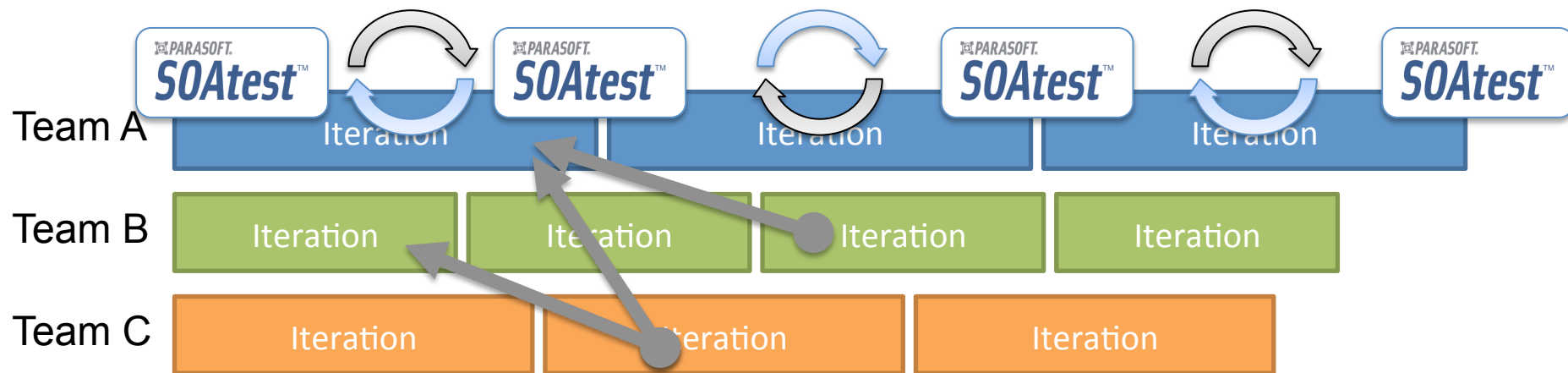
Reduced wait time for test team by 60% for a major media conglomerate

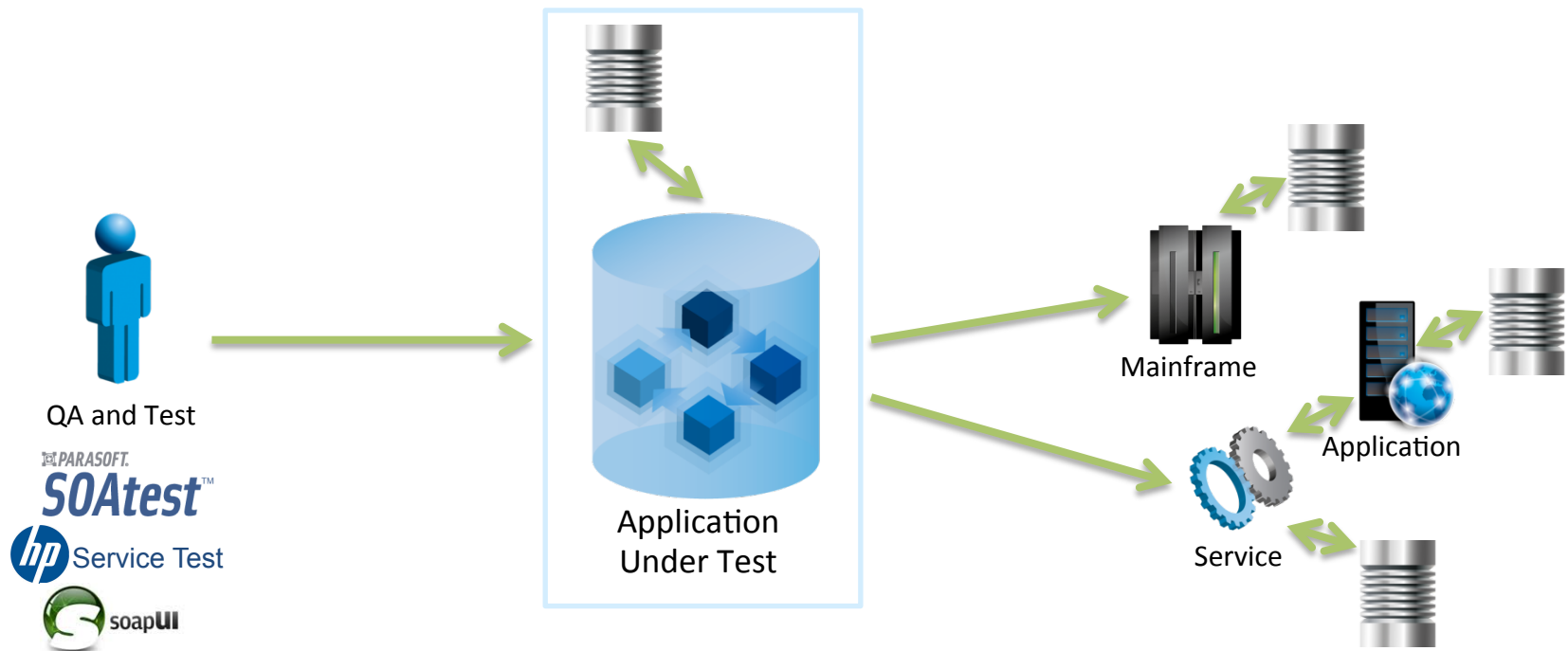


Agile/Parallel development limited by system dependencies

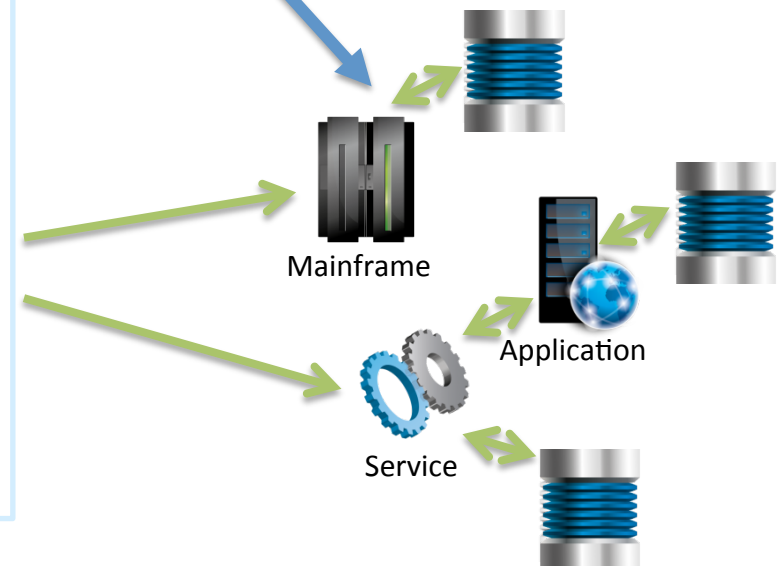
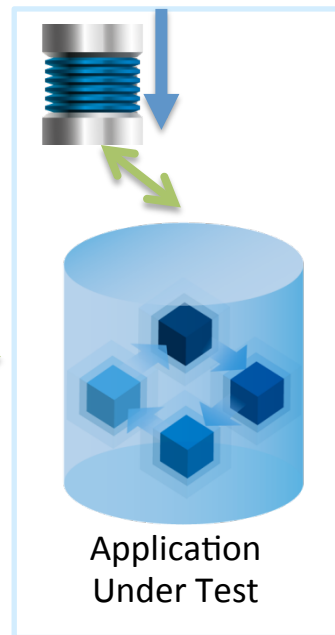
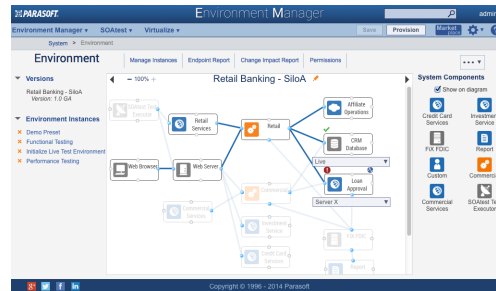
■ The Business Benefits

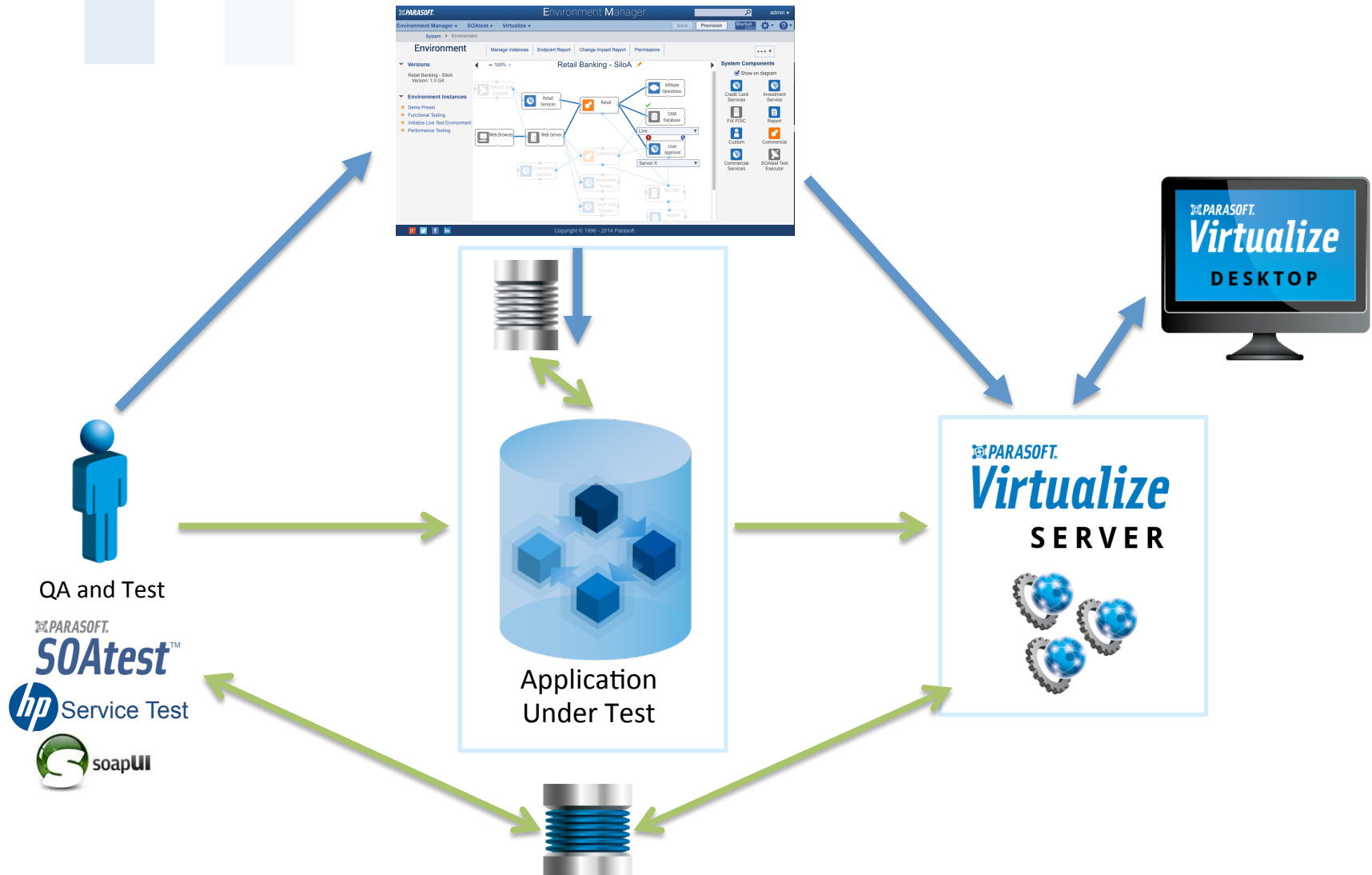
- Met business goals and timelines, were able to test functionality “as soon as” it was available. Practiced TDD against prototypes to get a head-start on ‘full system testing’





Environment Manager





Leveraging application behavior virtualization the team can reduce the complexity and the costs of managing multiple environments while providing on-demand access for development, test and training

Define & Capture

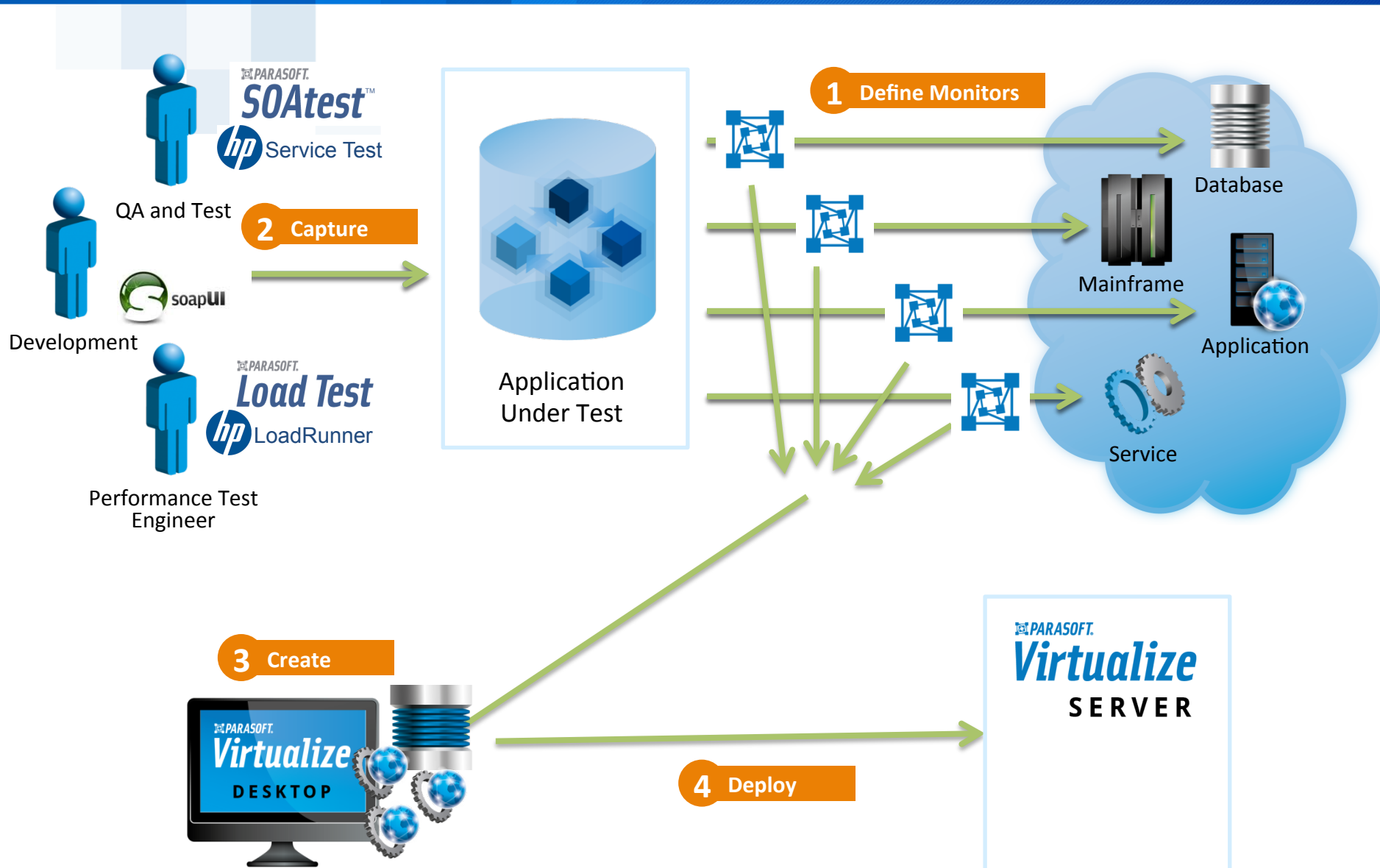
Initiated from the system under test, the user has the ability to capture detail from a live monitor that analyzes system traffic, from analyzing transaction logs or by modeling virtual behavior within the Parasoft Virtualized interface.

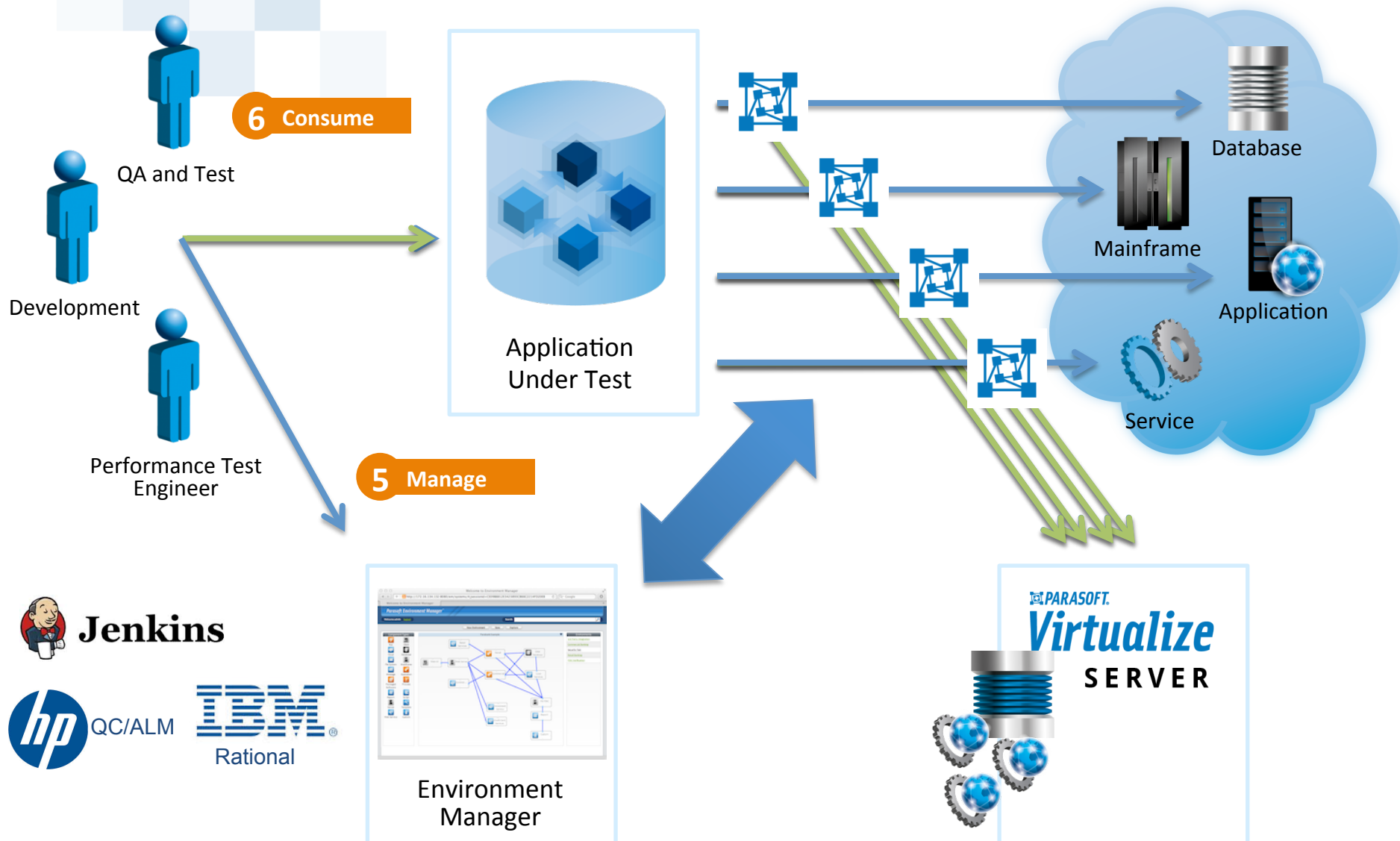
Model & Deploy

After the virtualized artifact has been captured, users can now instruct the details of the virtualized asset behavior. This includes: performance, data sources and conditional response criteria. The virtualized asset is then provisioned for simplified uniform access across teams and business partners.

Provision & Consume

The virtualized asset can now be called for unit, functional and performance tests. The virtualized asset can be leveraged by any test suite – including Parasoft Test.





- Simplifies the TDM problem
 - Reduces back-end data requirements
 - Data-Graphs vs. Relational-Data
 - Scalable, Fast, Efficient dynamic storage
 - Removes complex table/key relationships
 - Simplifies CRUD or stateful complexity of relational schema
- 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.

- Combination approach to TDM
 - Sample + Synthesis + Record
 - 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 data graphs 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)



Thank you!
Q&A

12/4/14