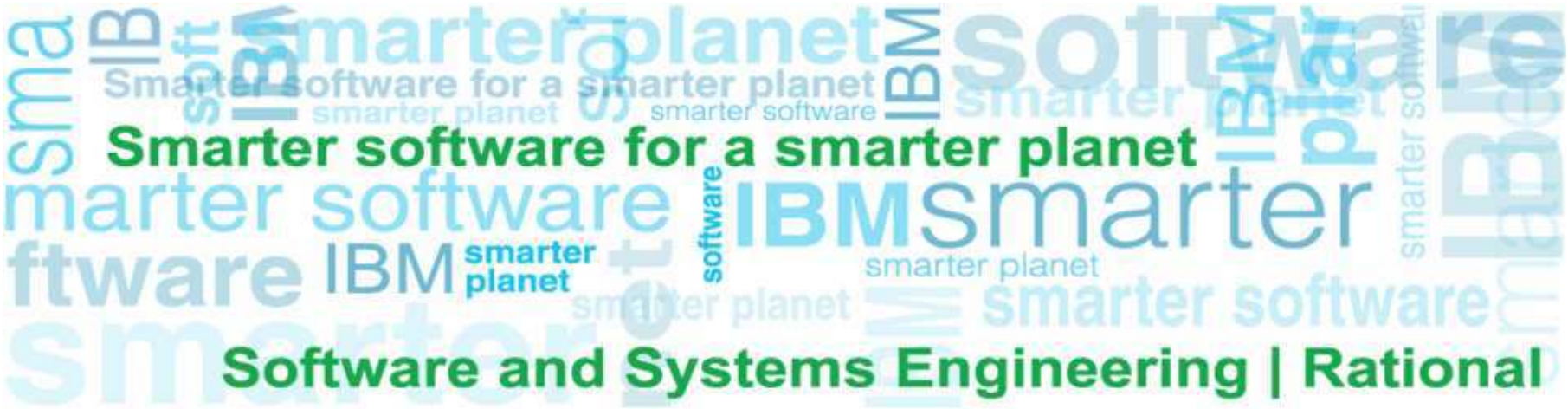
Yves Holvoet
Consultant, IBM Rational

Jean-Michel Oliva
Consultant, IBM Rational

La matinale du test logiciel

Virtualisation, automatisation et intégration continue des environnements de test



Agenda

- Panorama des solutions de test de Rational
- Les tests d'intégration et les environnements de virtualisation des tests
- Démonstration
- Questions / réponses

Contexte & Enjeux

- La rupture technologique actuelle (le tout numérique, l'internet, le web 2.0, la téléphonie mobile…) touche et change profondément les entreprises en créant de nouveaux schémas relationnels, de nouveaux besoins, de nouvelles exigences de simplicité.
- Le système d'information de l'entreprise doit s'adapter pour permettre le développement de nouveaux produits et de nouveaux services.
- La réduction des délais de mise à disposition sur le marché devient une priorité. Elle nécessite une meilleure synergie & synchronisation entre les directions métier et la direction des systèmes d'information

L'évolution toujours plus rapide des besoins des consommateurs contribue par ailleurs à accélérer l'axe temps. Tous les cycles de l'entreprise se raccourcissent : la R&D, la production, le marketing, la commercialisation.

Les DSI sont soumis à des pressions qui les conduisent aussi à raccourcir la durée de leurs projets.

Source : Baromètre CIO 2011

Quel est l'impact sur les phases de test ?

Réflexion sur les outils de test et sur la démarche de qualité logiciel ?

Les 4 axes de réflexion…

- ***Besoin de travailler en équipe => Souplesse et efficacité des projets***

  • Plate-forme collaborative de développement

- ***Traçabilité de l'expression des besoins jusqu'aux tests***

  • Chaîne industrielle de développement

- ***Un même processus / outillage, quelle que soit la technologie (Cobol, Pacbase, Java, …)***

  • Plate-forme commune de développement

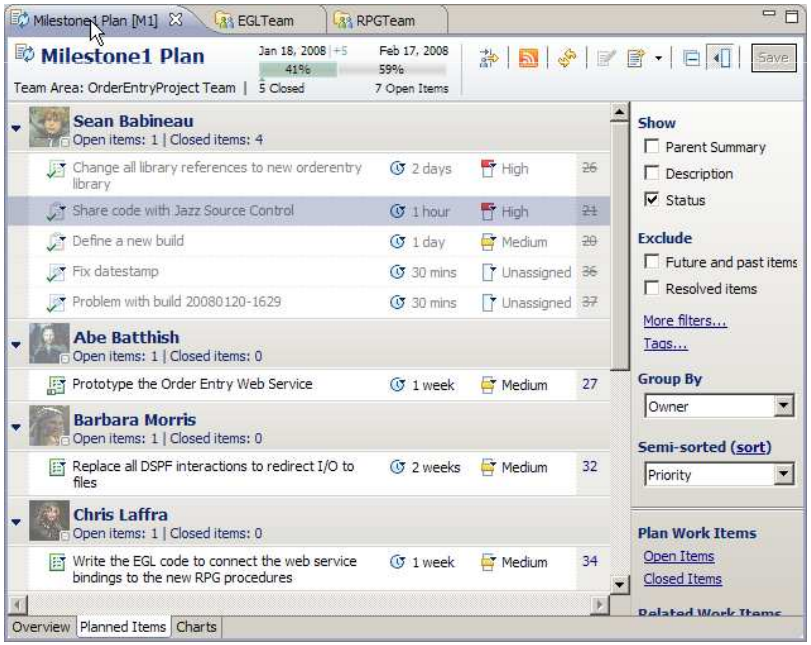
- ***Adapter facilement le processus aux différentes typologies de projets***

  • Gouvernance & Méthodologie

• Plate-forme collaborative de développement

Travail en équipe - Souplesse et efficacité des projets

- Favoriser la collaboration entre les métiers et les équipes de développement afin
  - D'identifier au plus tôt les écarts fonctionnels
  - D'accroitre la réactivité des études aux évolutions « métier »

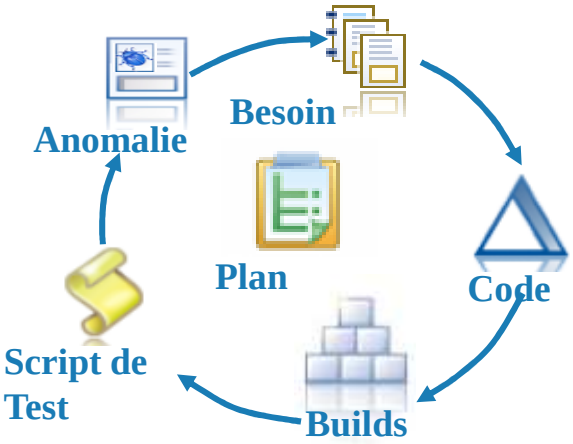


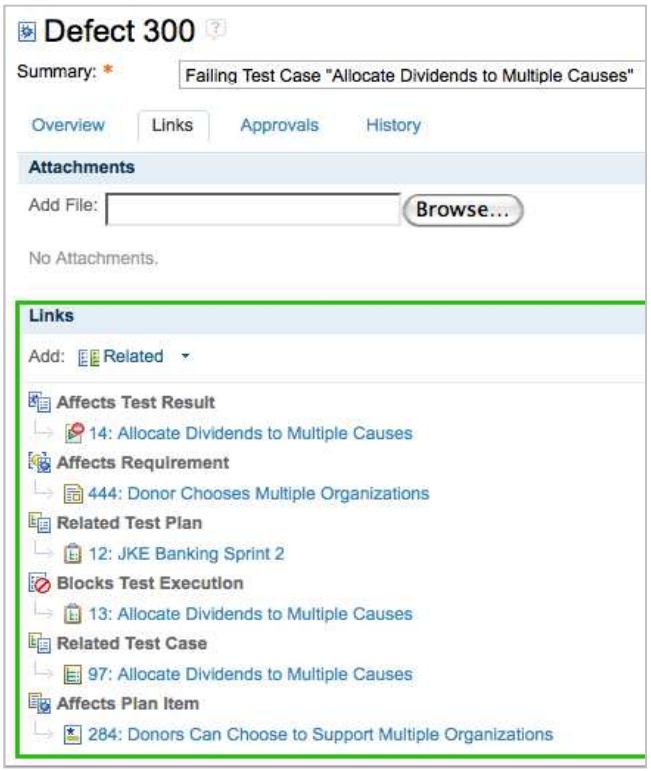
- Une collaboration active entre les métiers et les études nécessite une orientation vers une gestion des exigences
- Apporter de la transparence dans les activités de développement, de planification, de tests, etc..
- **Passer d'une efficacité individuelle à une productivité collective**

• Chaîne industrielle de développement

Traçabilité de l'expression des besoins jusqu'aux tests

- Gestion du cycle de développement de bout en bout
  - Le cycle de production logiciel doit être considéré dans son ensemble : de l'expression du besoin jusqu'à sa mise à disposition auprès des utilisateurs finaux
  - Supprimer les silos et favoriser l'intégration « sans couture » des outils, des artéfacts et des individus
  - Renforcer les liens de traçabilité



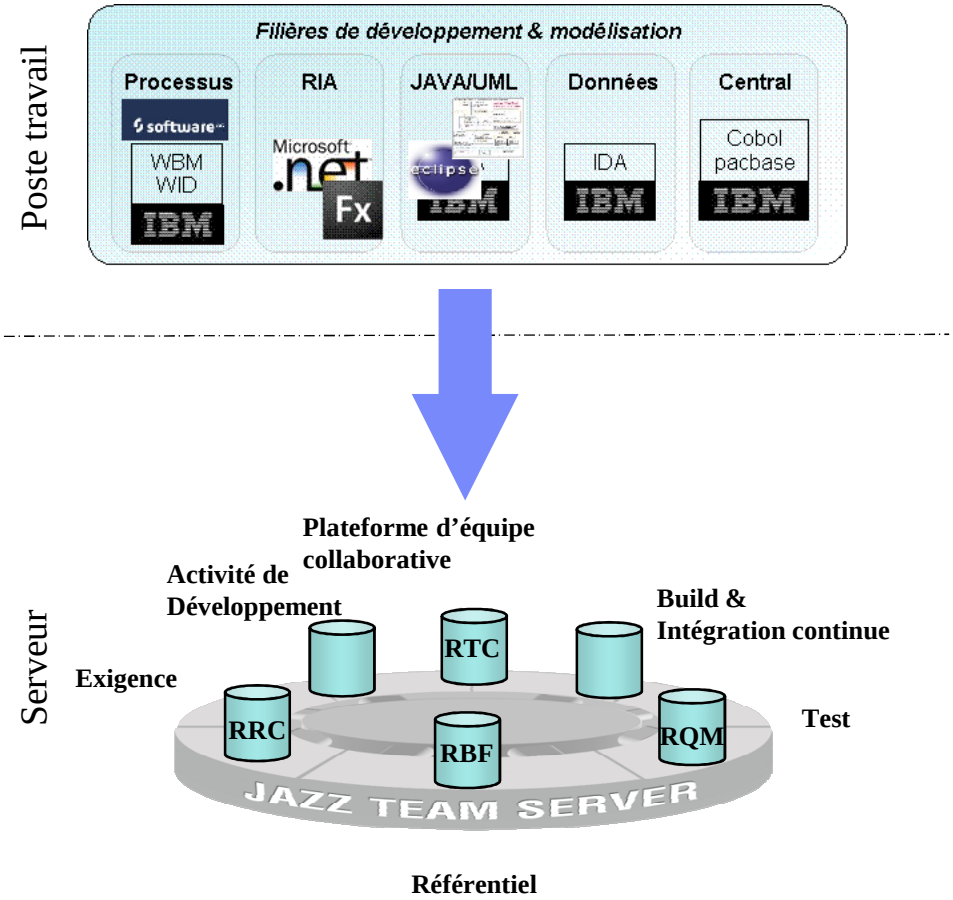


- **La modification des priorités, l'évolution des besoins du métier imposent une traçabilité forte entre les exigences, les activités de développement et les activités de tests.**
  - Quel est l'impact d'un changement (métier) sur le projet ?
  - Quels tests doivent être modifiés ?
  - Quelles activités de développement doivent être modifiées ?
  - Quels sont les besoins métier pour les lesquels les développements et les tests ont été réalisés ?
  - Etc..

• Plate-forme commune de développement

Un même référentiel, quelle que soit la technologie (Cobol, Pacbase, Java, .NET, Flex)

- Favoriser le partage d'informations entre des mondes technologiques différents (Pacbase, cobol, Java, .net, autres…)
- Favoriser la mutualisation des compétences
- Banaliser le poste de travail des équipes de développement en favorisant l'adoption du socle Eclipse
- Gérer des applications intégrant des filières technologiques différentes
  - Pacbase, Cobol, PL1
  - Java, RIA, …
- **Intégrer les outils et les processus**

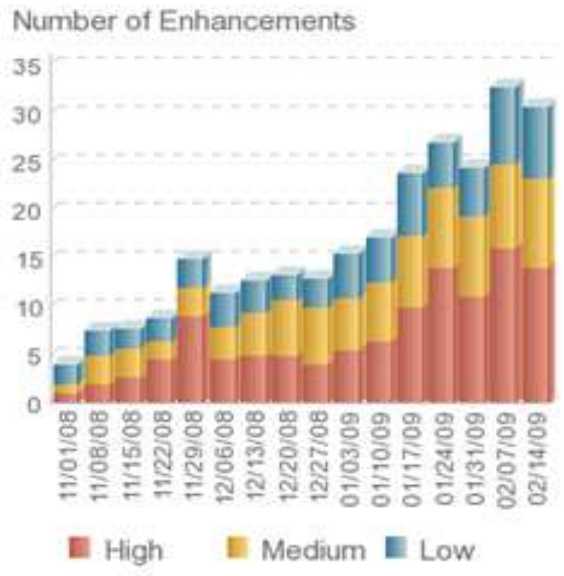


• Gouvernance & Méthodologique

Adapter le processus aux différentes typologies de projets

- Proposer des processus adaptées aux différentes typologies de projets. Identifier les critères d'éligibilité :
  - Méthode formelle (cycle de développement en V). Exemple : adaptée au projet « réglementaire »
  - Méthode Agile : projet dont la cible n'est pas clairement définie ou peut évoluer durant la vie du projet

- Outiller les méthodes pour faciliter leur adoption



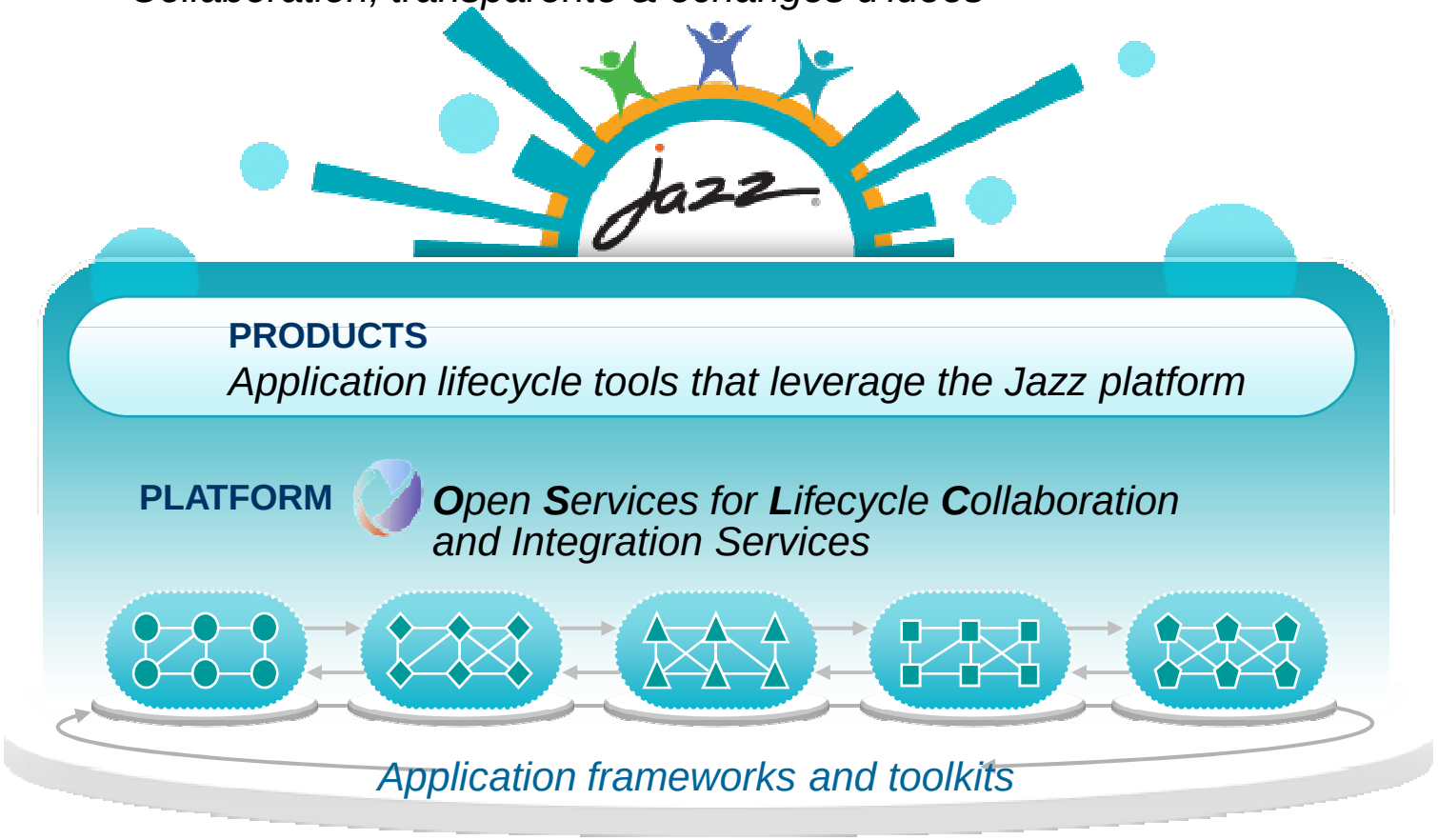
- Gérer correctement la gouvernance des projets
  - Indicateurs sur le cycle de vie des applications adaptés à chaque population (management, chef de projet, développeur…)
    - Time to value (planification)
    - Valeur produite
    - Coût,..
  - Démarche d'amélioration continue des processus de développement

Consolider automatiquement toutes les informations pertinentes pour suivre en temps réels les projets

Jazz, une plate-forme collaborative pour transformer la manière de produire des logiciels

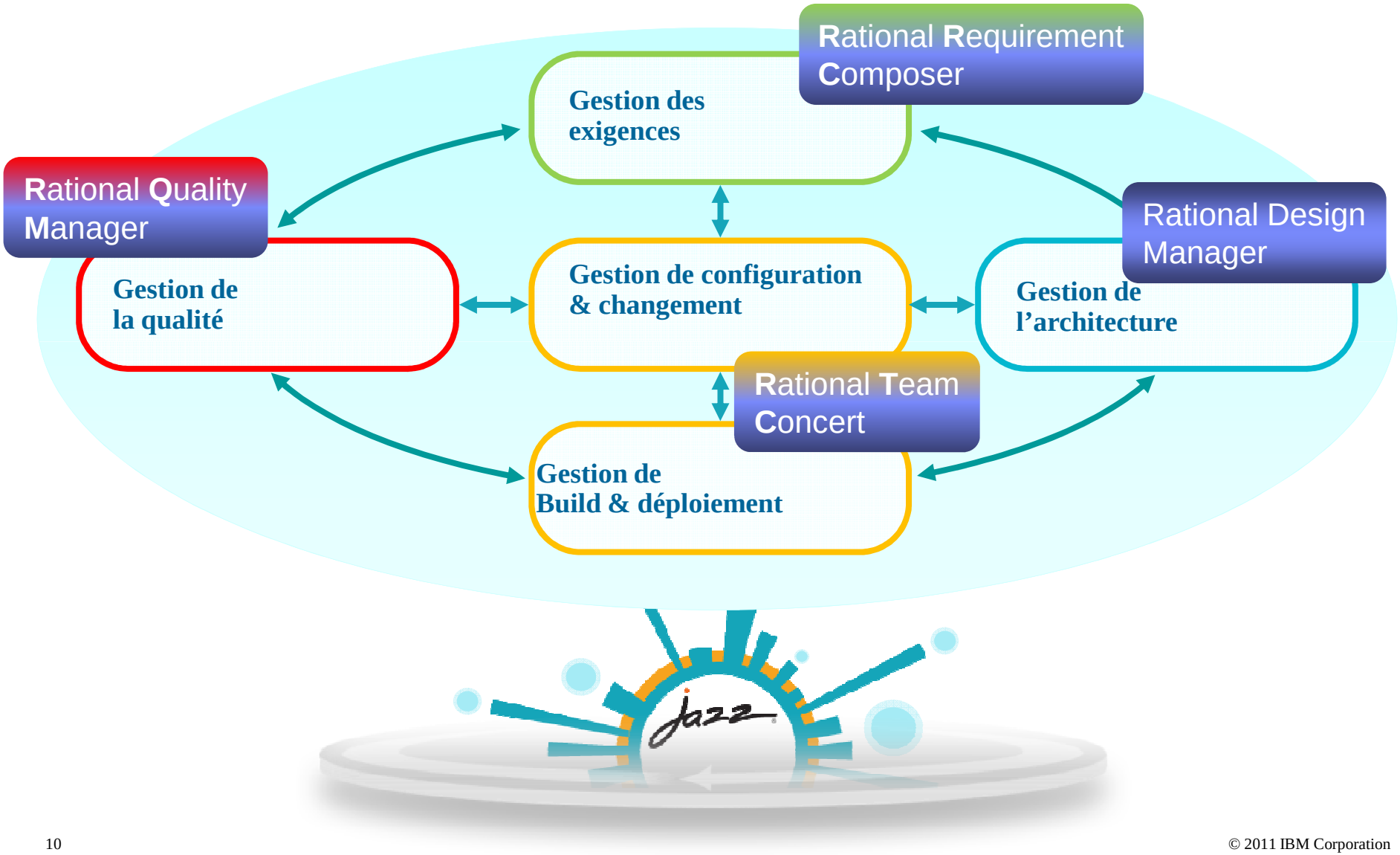
Communauté jazz.net

Collaboration, transparente & échanges d'idées



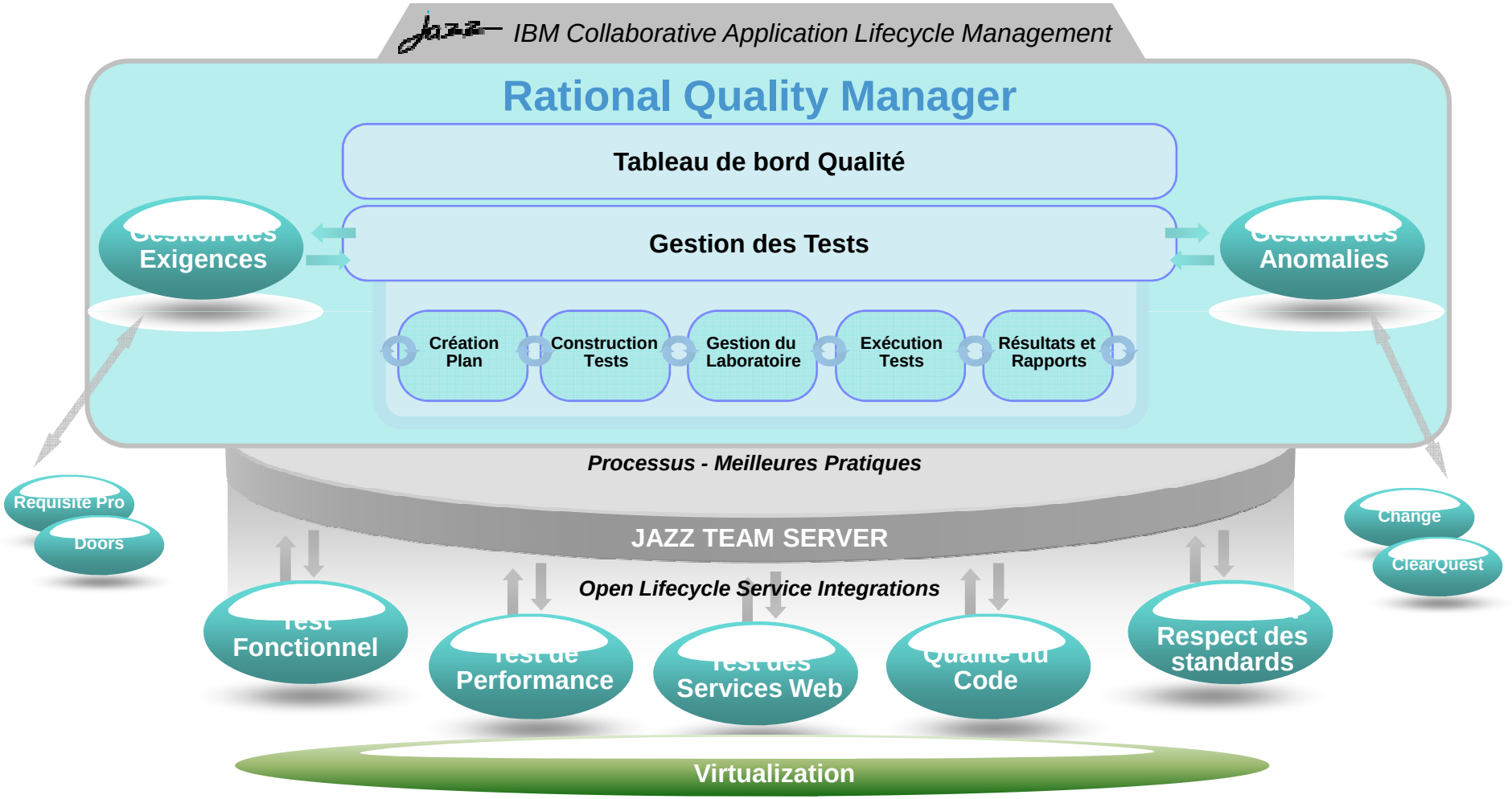
Plus d'information sur https://jazz.net/about/

Gestion du cycle de vie des applications
Modulaire, ouvert et extensible

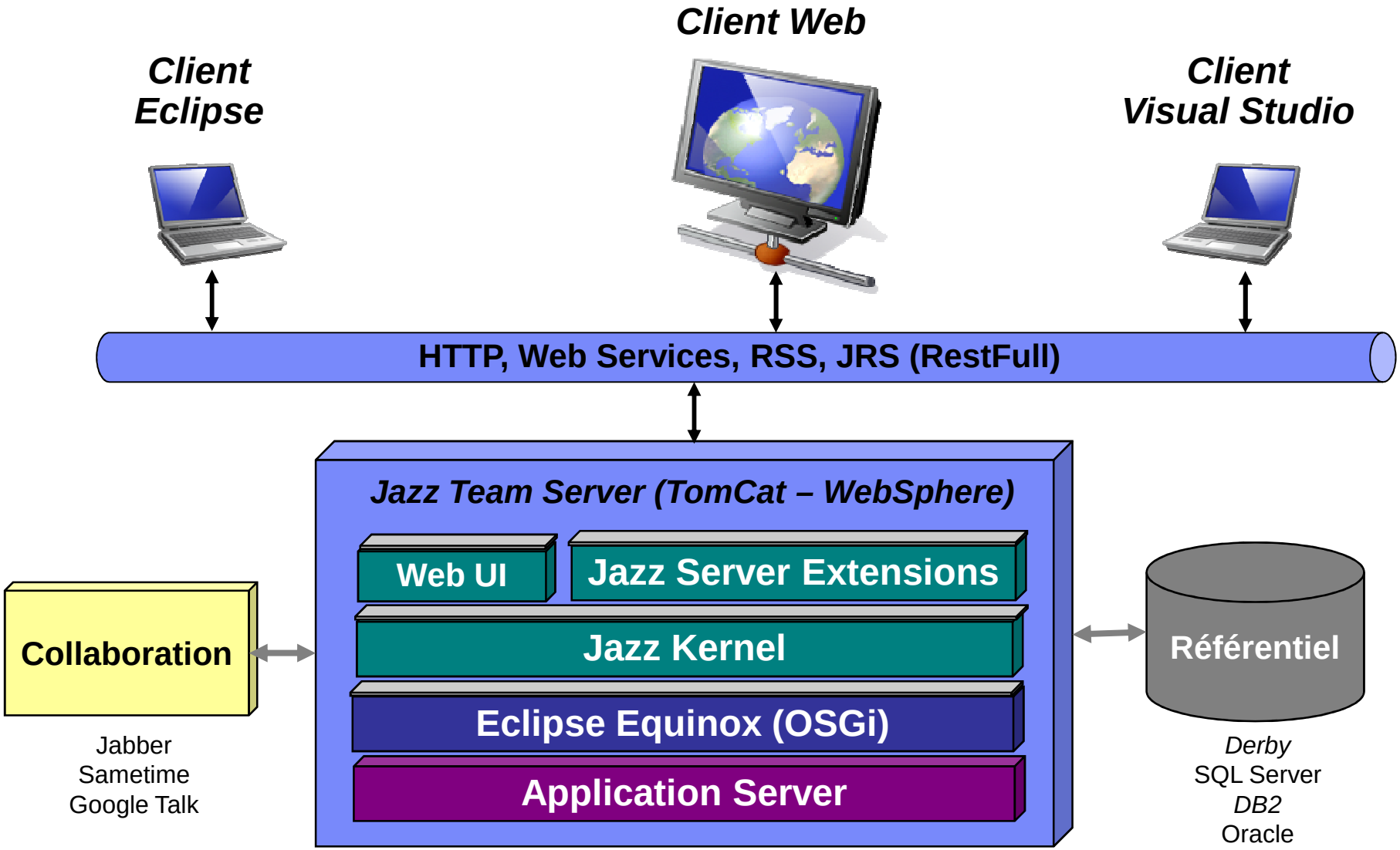


IBM Rational Quality Manager

Une gestion collaborative des tests basée sur la plateforme Jazz

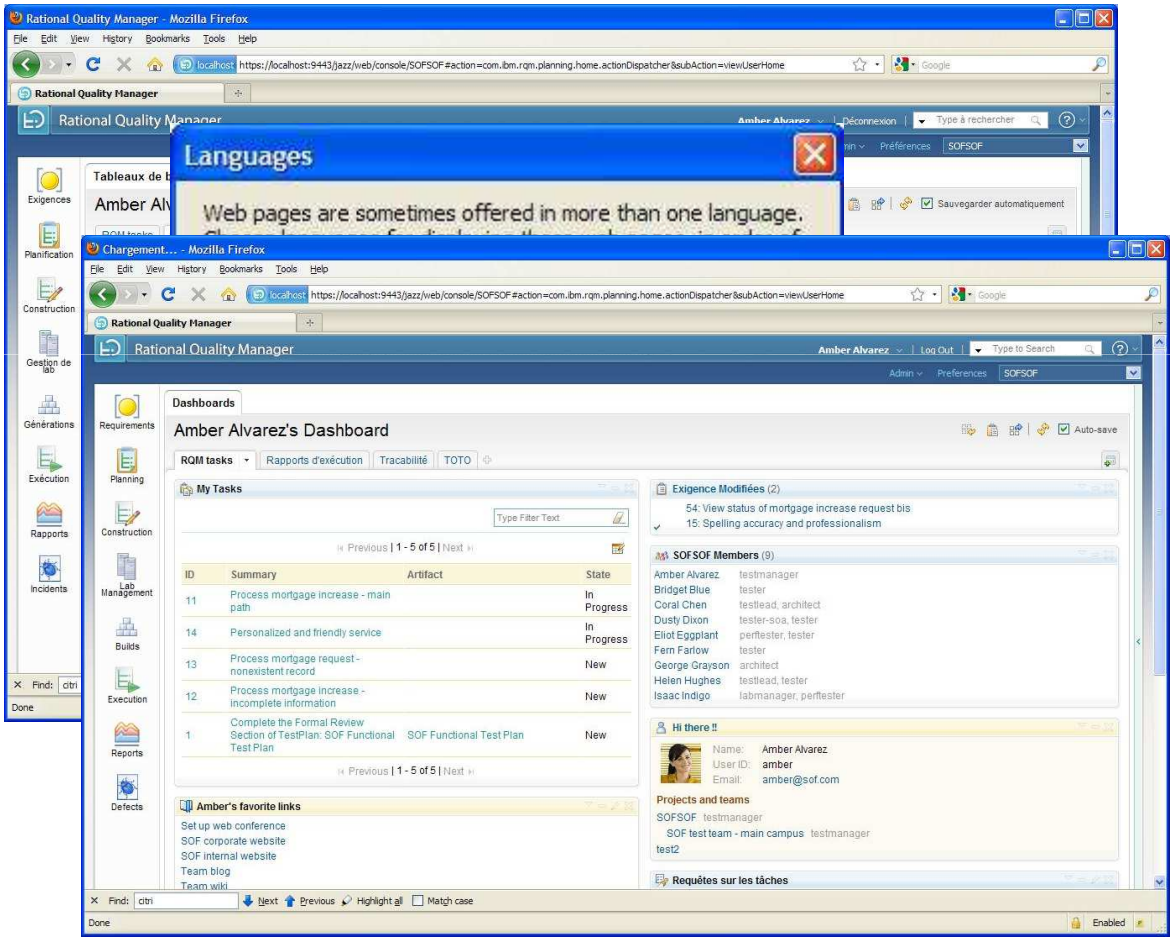


L'architecture de Rational Quality Manager

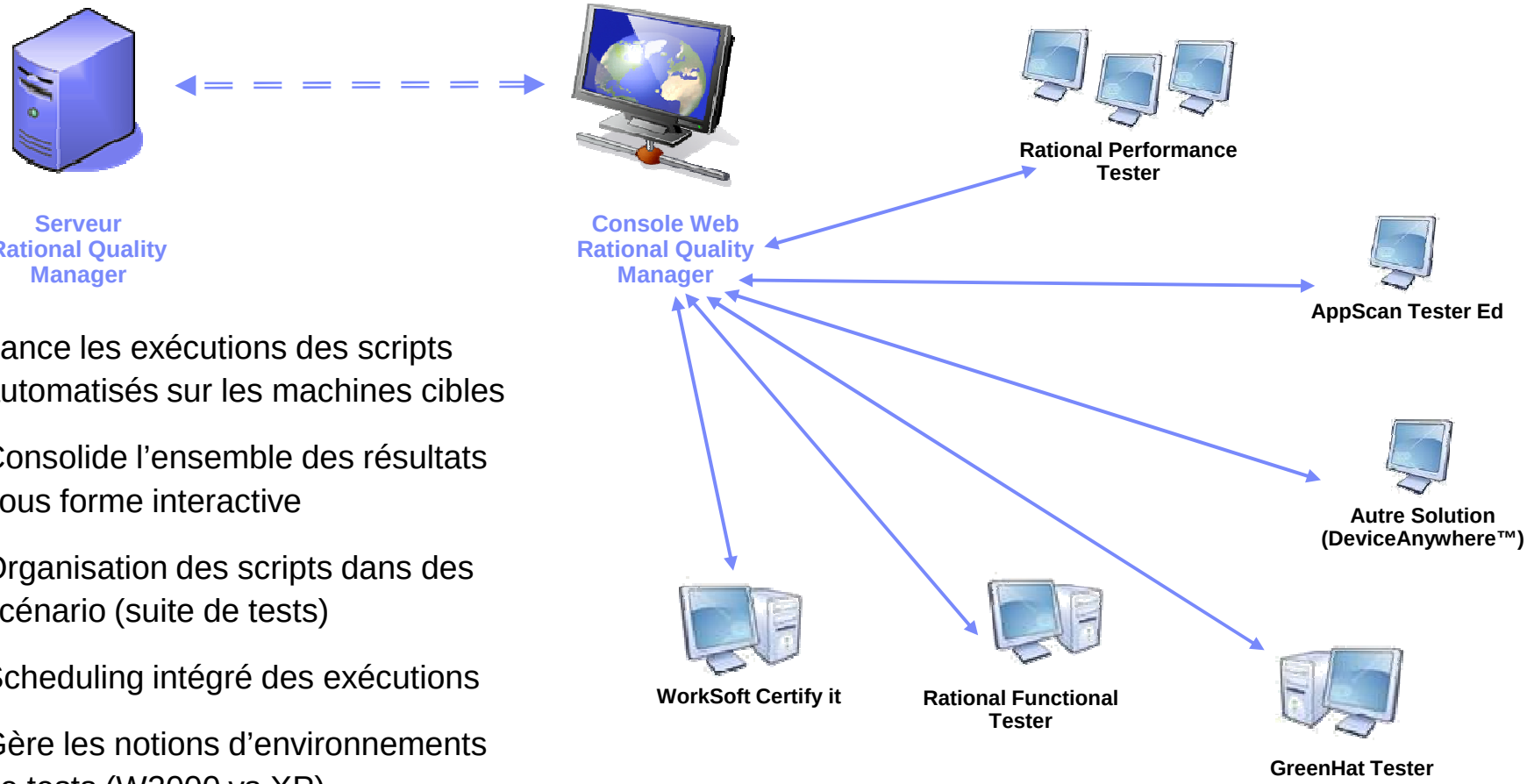


Rational Quality Manager est un client 100% web

- **100% web – sans activeX**
- Utilisation d'un poste banalisé
  - Pas d'installation sur les postes
  - Pas de mise à jour lors des changements de version
  - Pas besoin de droit d'administration local
- Support de Firefox et d'IE
- Changement de langue en fonction du choix du navigateur



Pilotage des automates de test



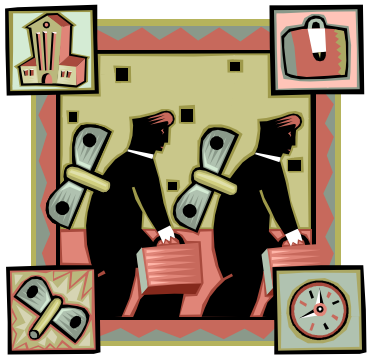
- Lance les exécutions des scripts automatisés sur les machines cibles
- Consolide l'ensemble des résultats sous forme interactive
- Organisation des scripts dans des scénario (suite de tests)
- Scheduling intégré des exécutions
- Gère les notions d'environnements de tests (W2000 vs XP)
- Intégré avec le Test Lab Management (machines virtuelles)

Test fonctionnel avec **IBM Rational Functional Tester**

Un outil polyvalent pour le test fonctionnel automatisé

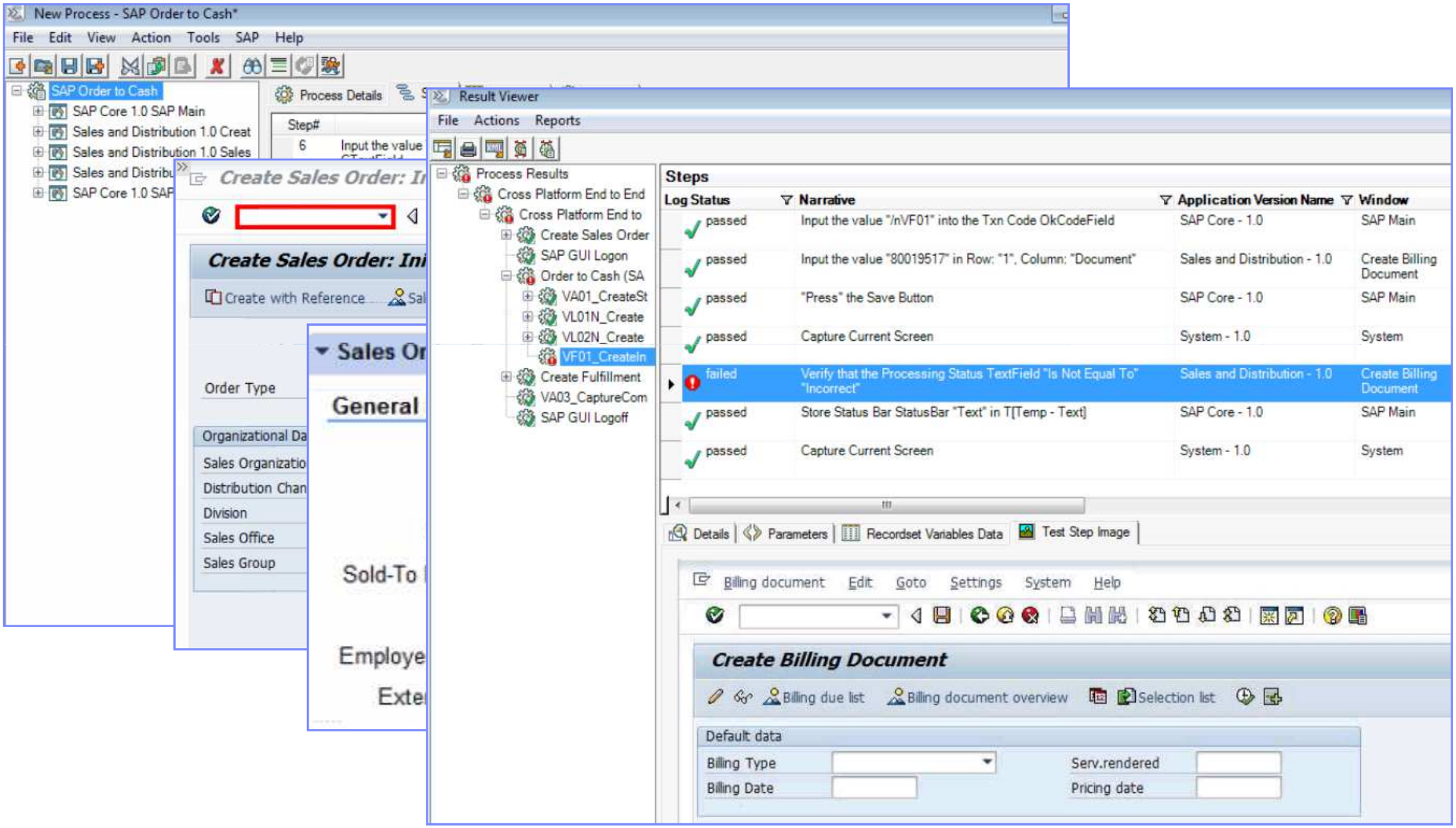
- **Domaines supportées par RFT**
  - Web, J2EE, AWT, SWT, AJAX, Adobe Flex 3.5, Dojo 1.3 Toolkit
  - WinForms and Windows .Net, VB 5.0 et 6.0, Oracle Forms
  - PowerBuilder 10.5 à 11.5
  - Siebel, Citrix 5.0
  - Terminaux **3270** et **5250** (Mainframe or zSeries - TN3270 - TN3270E AS/400 or iSeries - TN5250 Virtual Terminals or pSeries - VT default, - VT100 - VT420-7 - VT420-8 - VT UTF-8)
  - Couplage avec IBM Optim
  - **La licence RFT comprend l'ensemble des extensions**
  - **Pas de surcoût additionnel**
- **Principaux bénéfices**
  - Minimiser la maintenance des tests en s'affranchissant au mieux des changements de l'application
  - Tests pilotés par les données à partir de Wizards
  - Prise en main rapide pour les nouveaux utilisateurs grâce à la description des scripts en langage naturel
  - Langages de scripts standards et puissants pour les utilisateurs avancés (Java (**Eclipse**), VB(**VS .Net**))
  - Gestion des versions couplée avec **Rational ClearCase** et **Rational Team Concert**

Worksoft Certify for SAP



- **Simple**
  - La solution est conçue pour les utilisateurs métiers
  - La génération des scénarios découle de l'interaction avec l'application SAP
- **Sans script**
  - Concept d'objet et d'action
  - Pas de génération de script
  - Pas de compétence de programmation nécessaire
- **Réutilisable**
  - Les tests peuvent être réutilisés entre plusieurs projets
  - Tous les tests peuvent être utilisés à chaque étape de validation (non regression, migration, etc…)
- **Maintenance améliorée**
  - Modification d'un objet ou d'un comportement par un clic

Worksoft Certify for SAP



IBM Rational Performance Tester

Automatisation des tests pour les novices et les experts

- **IBM Rational Performance Tester**
  - Test de performance des applications Web, Citrix, socket, SAP, SIP, Siebel, Web Services (SOA)
- **Principaux bénéfices**
  - Productivité immédiate
    - Masque la complexité pour permettre de réaliser simplement le travail
  - Accès et manipulation de données avancés
    - Variation de données automatisée et synchronisation
  - Réduction du coût du test de performance
    - Besoin limité de ressource
    - Intégration à l'environnement de développement

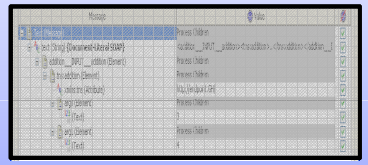
Agenda

- Panorama des solutions de test de Rational
- Les tests d'intégration et les environnements de virtualisation des tests
- Démonstration
- Questions / réponses

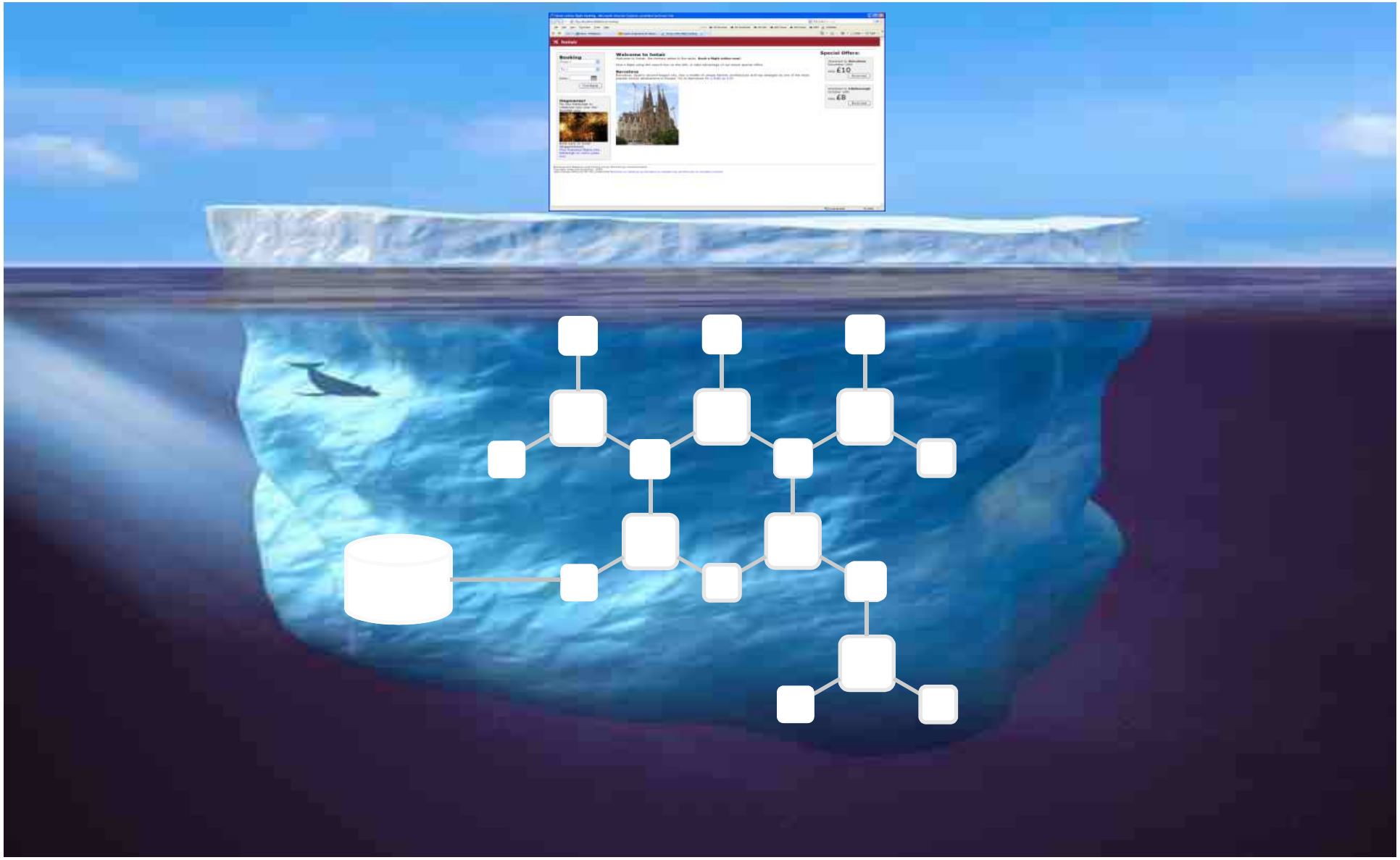
Types of Testing

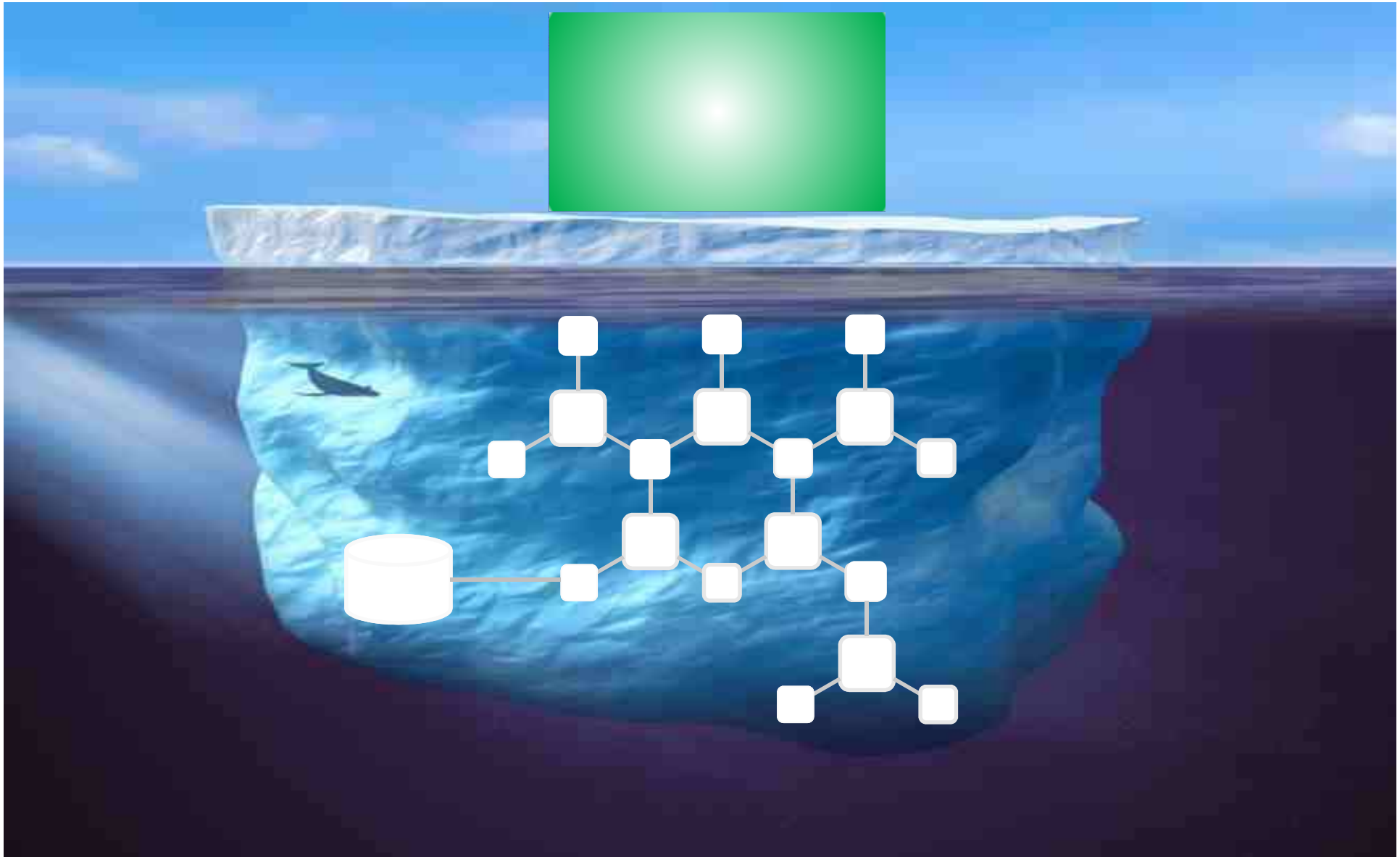
White box – testing concerned with the internal structure of the program

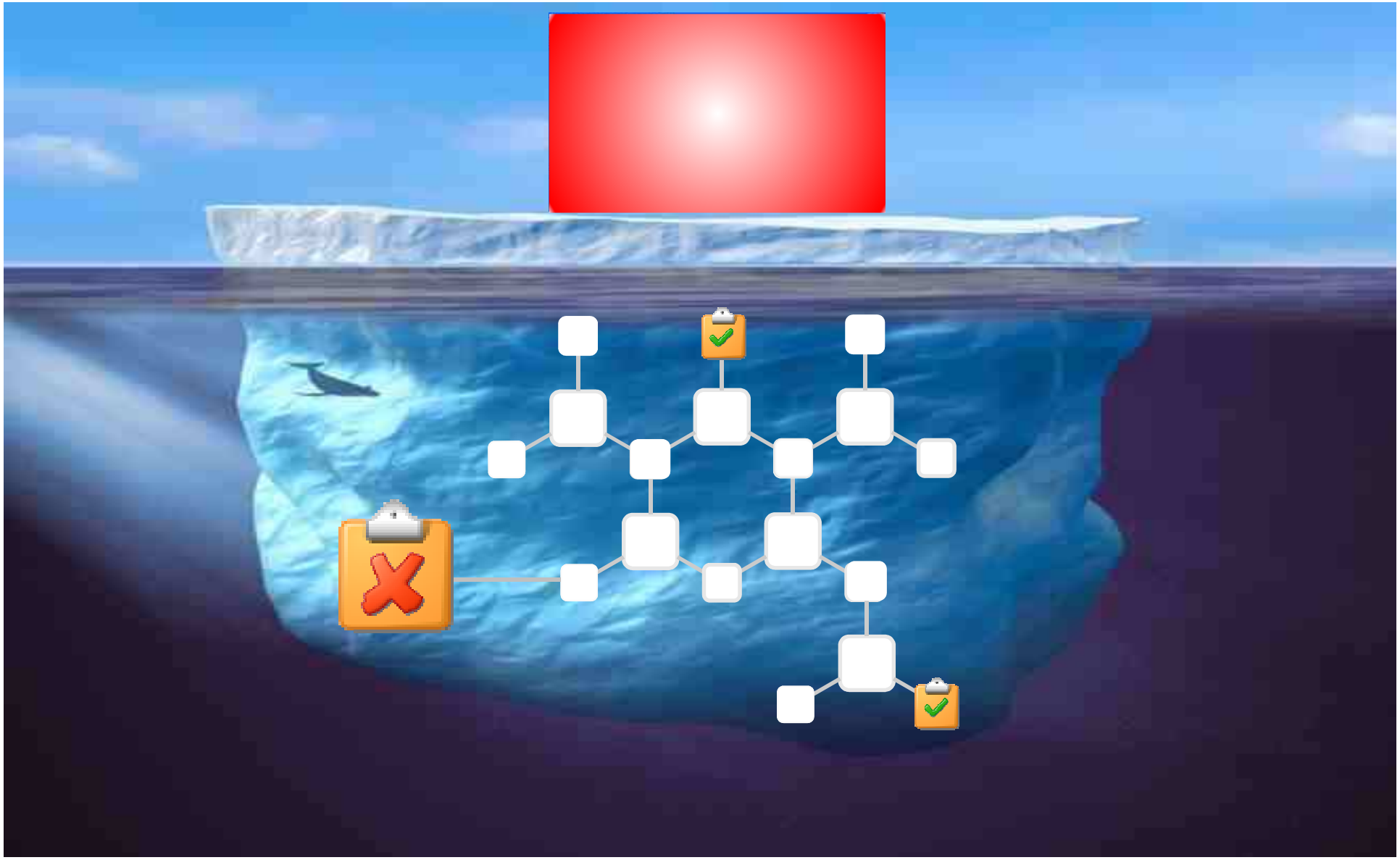
Grey box – Examination of logs, databases, message schemas, design outputs etc.



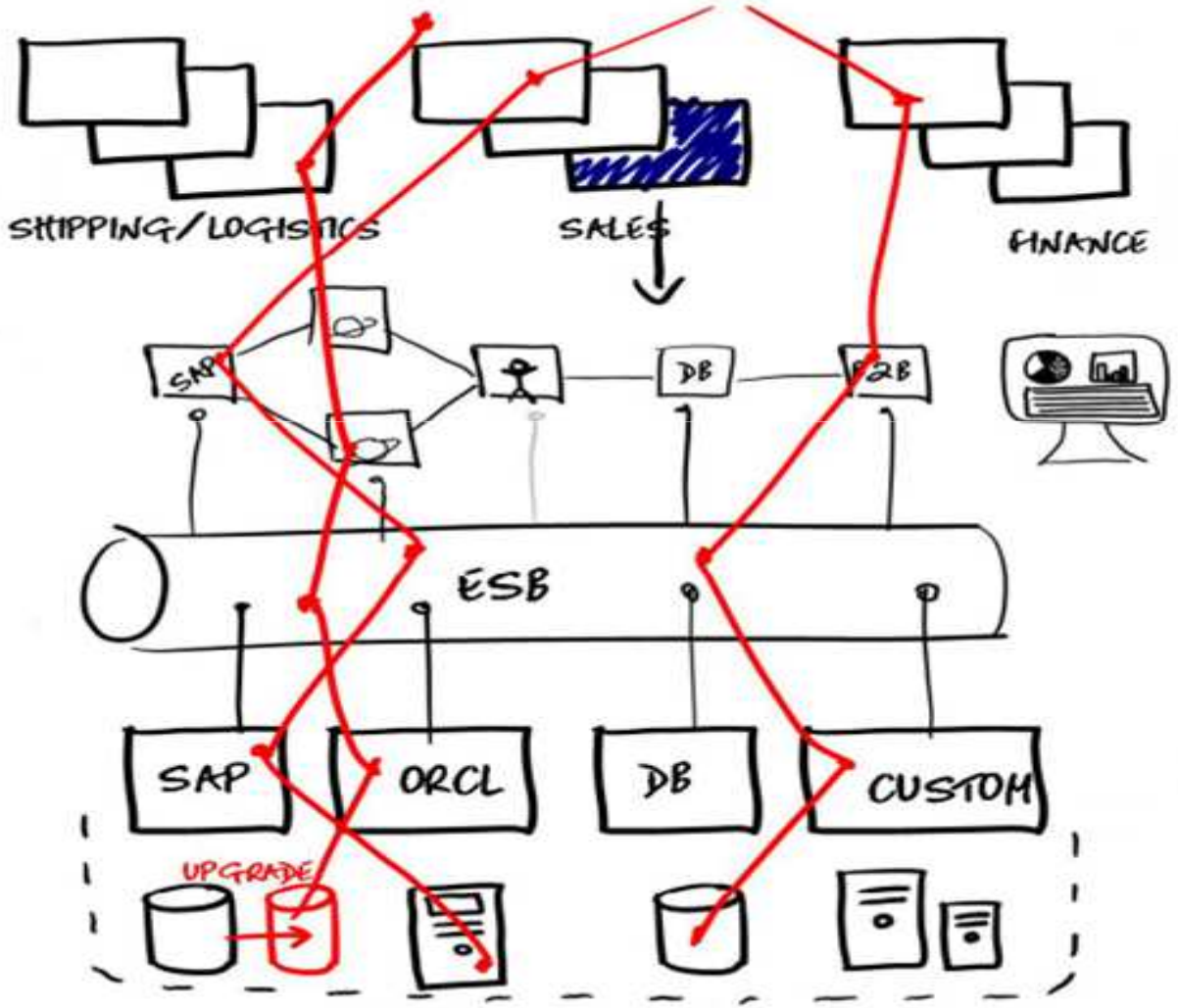
Black box – testing concerned with input/output of the program







Complexity…. Made Simple!



- Multiple layers of technology
- Multiple vendor platforms
- Complex transactions
- Complex dependencies
- Multiple stakeholders

Supported Environments & Technologies

Messaging Protocols

- ActiveMQ
- Email (SMTP, IMAP)
- Files
- FTP/S
- HTTP/S
  JMS (JBOSS et al)
  IBM WebSphere MQ
- JBoss MQ
- SAP IDoc, BAPI, RFC & XI/PI
- Software AG's IB & IS
- Solace
- Sonic MQ
- TCP
  TIBCO Rendezvous, Smart Sockets & EMS
- Custom

SOA, ESB, Others

- CentraSite
- Oracle Fusion
- SCA Domain
- Software AG IS, BPMS
- Sonic ESB
- TIBCO ActiveMatrix
- UDDI
- Web Services
- WebSphere RR
- WSDL

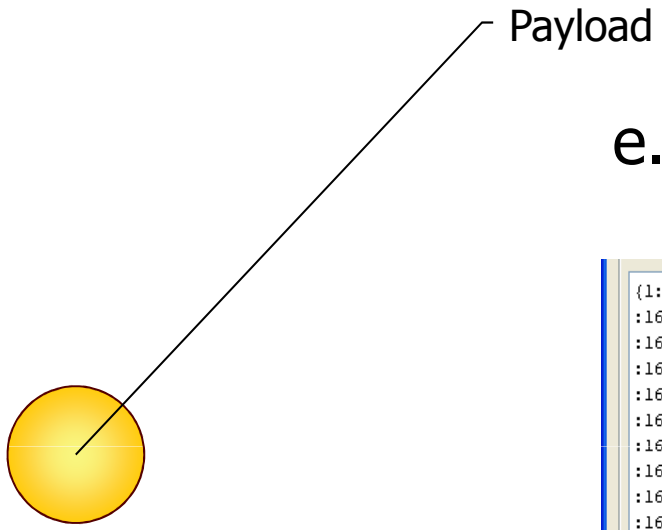
- BPM
- Databases
- Log Files

Message Formats

- .Net Objects
- Bytes
- COBOL Copybook
- ebXML
- EDI
- Fixed Width
- HL7
- IATA
- Java Objects
- MIME
- OAG
- SOAP
- Software AG Broker Docs
- SWIFT
- TIBCO ActiveEnterprise
- XML (DTD, XSD, WSDL)
- Custom

Functional Test Automation

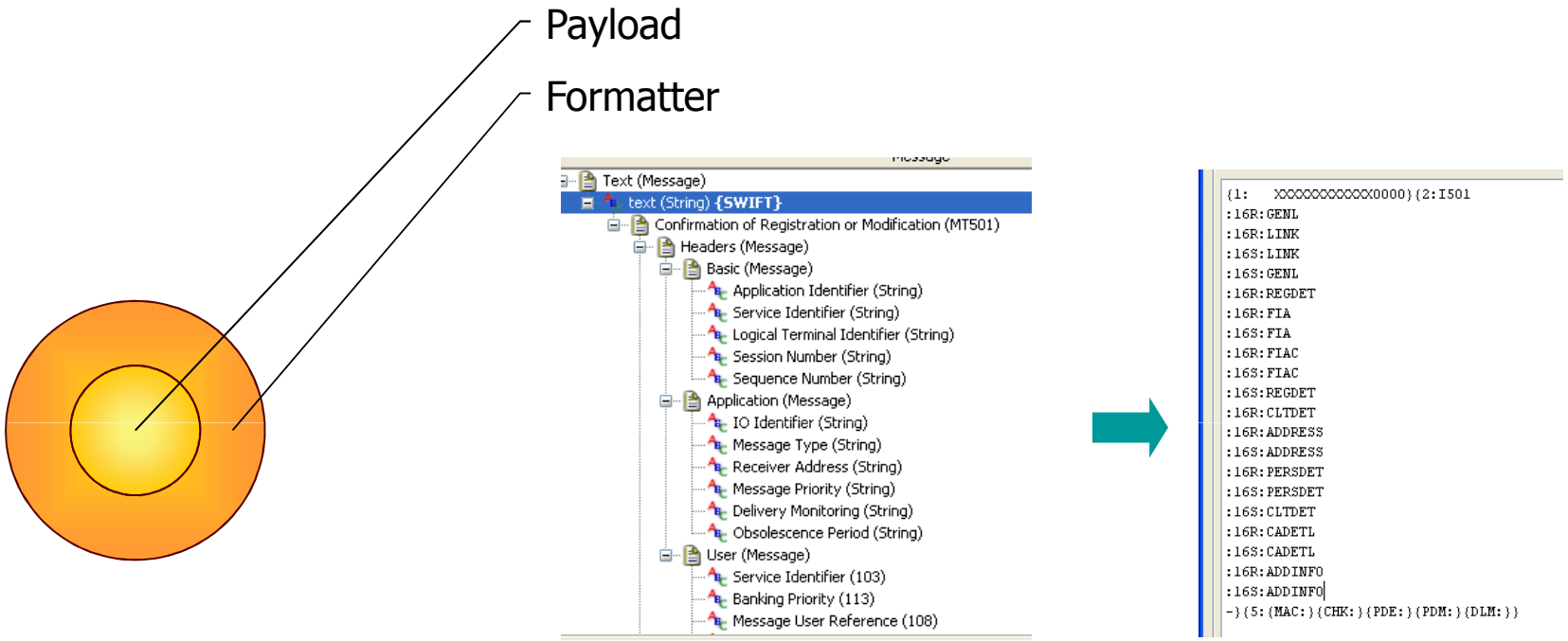
Build and Send Payloads 1



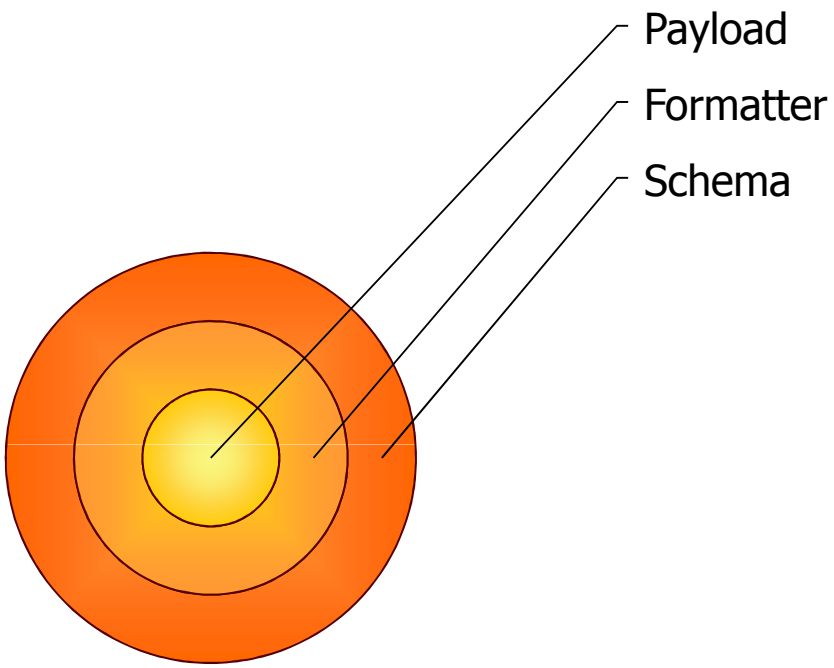
e.g. SWIFT, FIX, OFAC, Custom

```
{1:   XXXXXXXXXXX0000}{2:I501            }{3:{103:}{113:}{108:}{119:}{115:}}{4:
:16R:GENL
:16R:LINK
:16S:LINK
:16S:GENL
:16R:REGDET
:16R:FIA
:16S:FIA
:16R:FIAC
:16S:FIAC
:16S:REGDET
:16R:CLTDET
:16R:ADDRESS
:16S:ADDRESS
:16R:PERSDET
:16S:PERSDET
:16S:CLTDET
:16R:CADETL
:16S:CADETL
:16R:ADDINFO
:16S:ADDINFO
-}{5:{MAC:}{CHK:}{PDE:}{PDM:}{DLM:}}
```

Build and Send Payloads 2



Build and Send Payloads 3



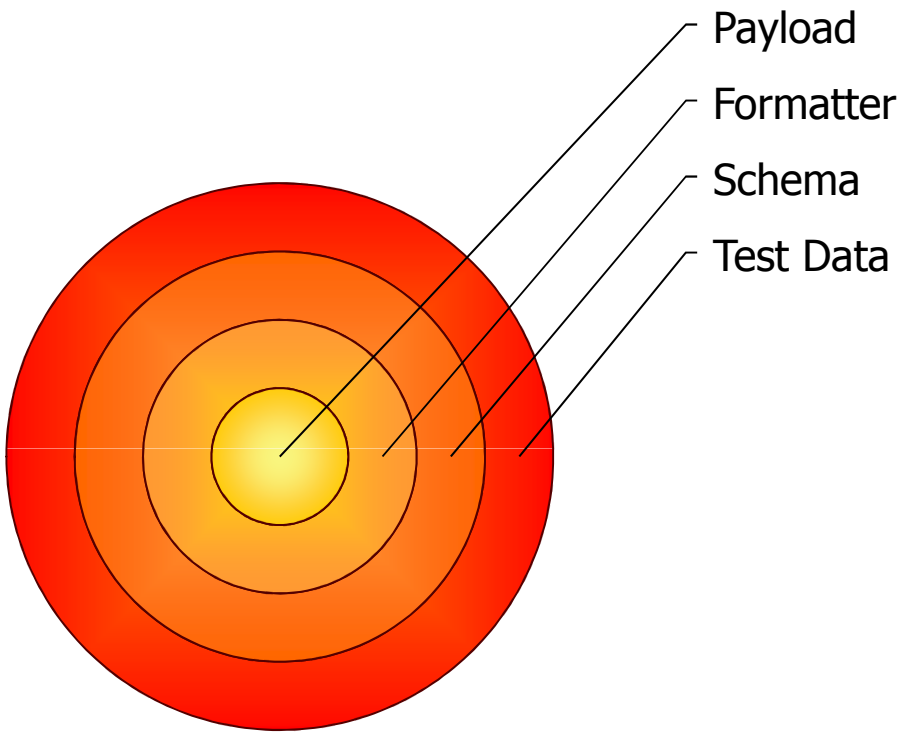
SOAP	XML
SWIFT	COBOL Copybook
DTD	XSD
WSDL	HL7
EDI	IATA
Text	MIME
Byte Array	Java Objects
OAG	SAP BAP/RFC
FIX	.Net Objects

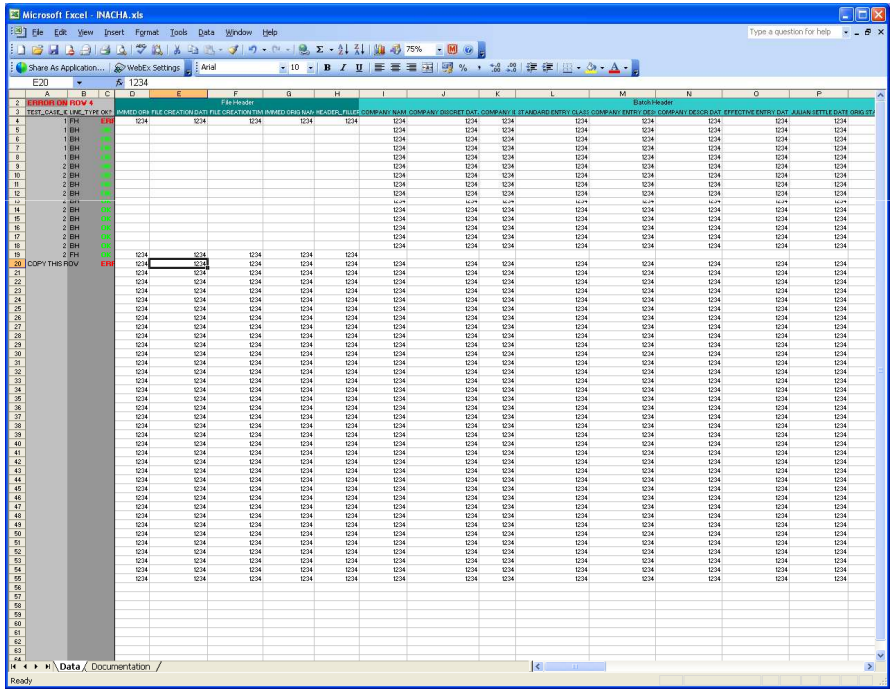
TIBCO ActiveEnterprise

webMethods IB and IS Documents

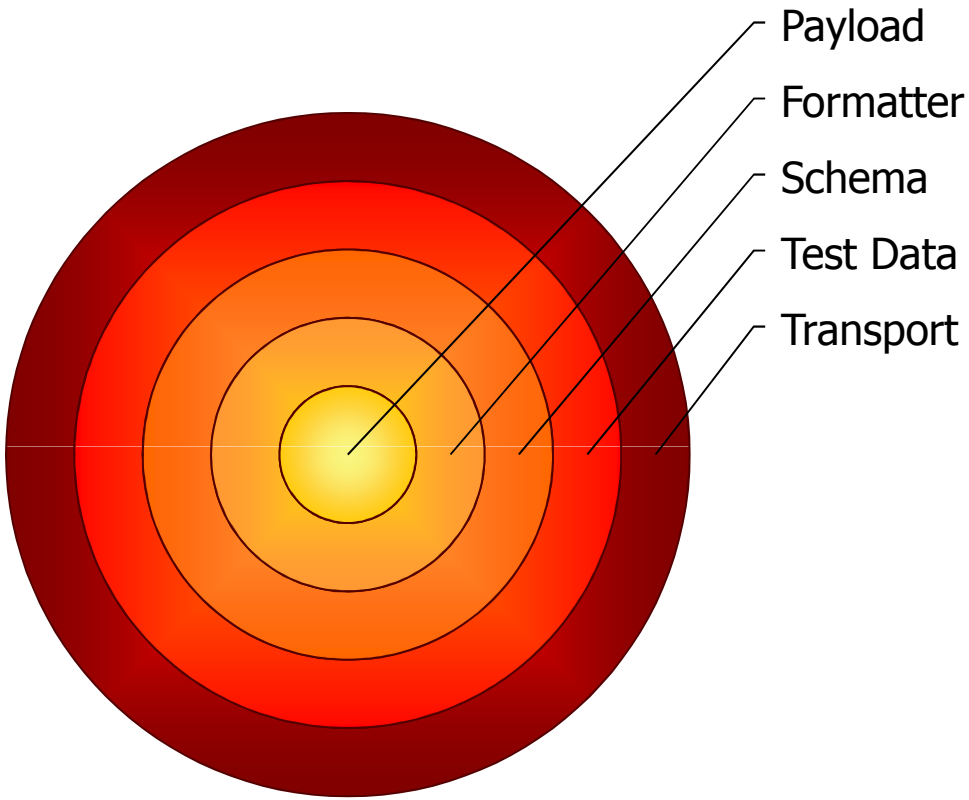
Custom…

Build and Send Payloads 4



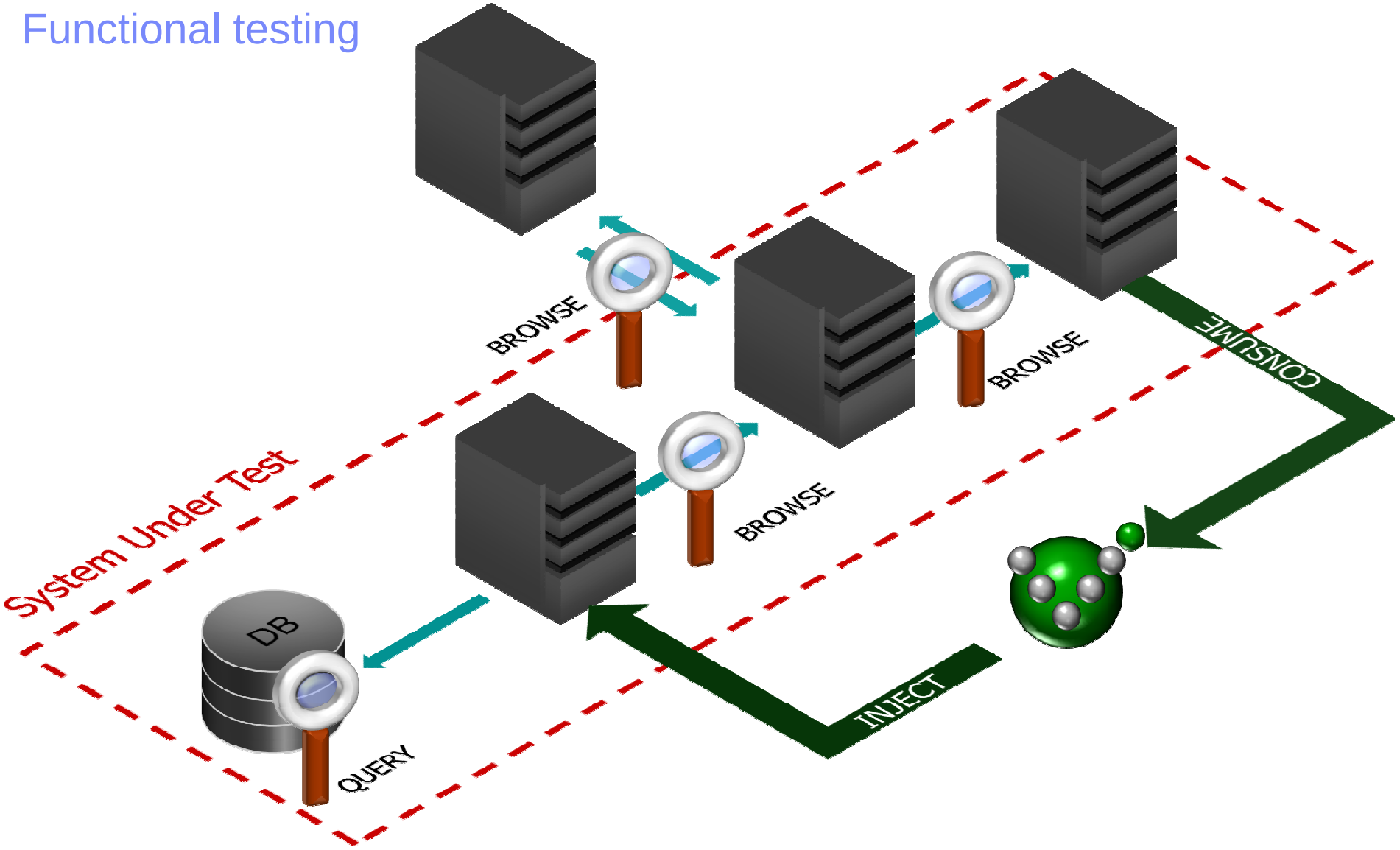


Build and Send Payloads 5

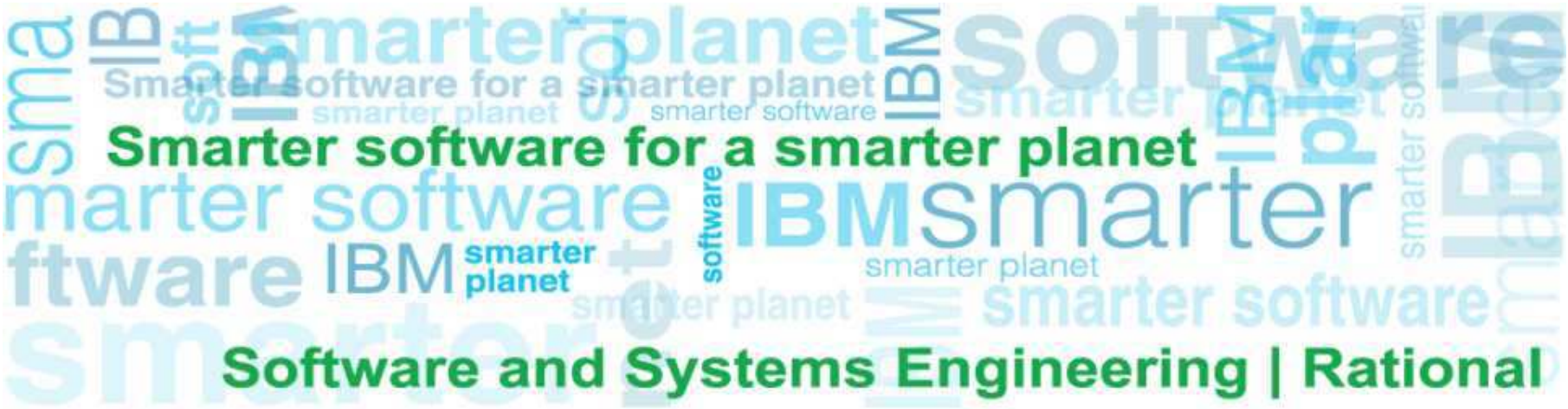


JMS	webMethods
HTTP/S	FTP
NDM	Oracle/BEA
JDBC	Sonic MQ
TCP/UDP	PL/SQL
Flat Files	Shell commands
TIBCO EMS	TIBCO iProcess
TIBCO Rendezvous	
IBM WebSphere MQ	
Custom Transports…	

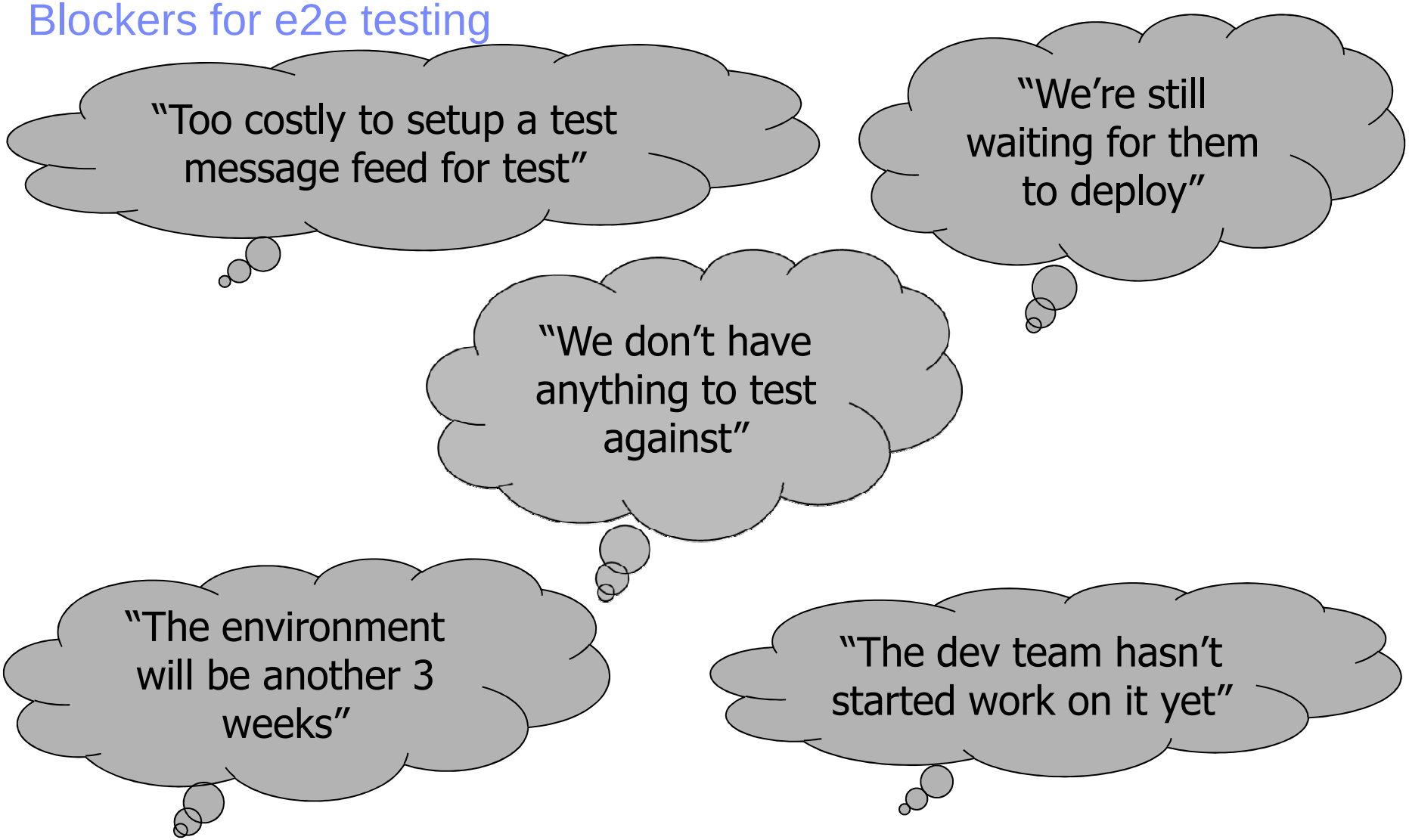
Functional testing



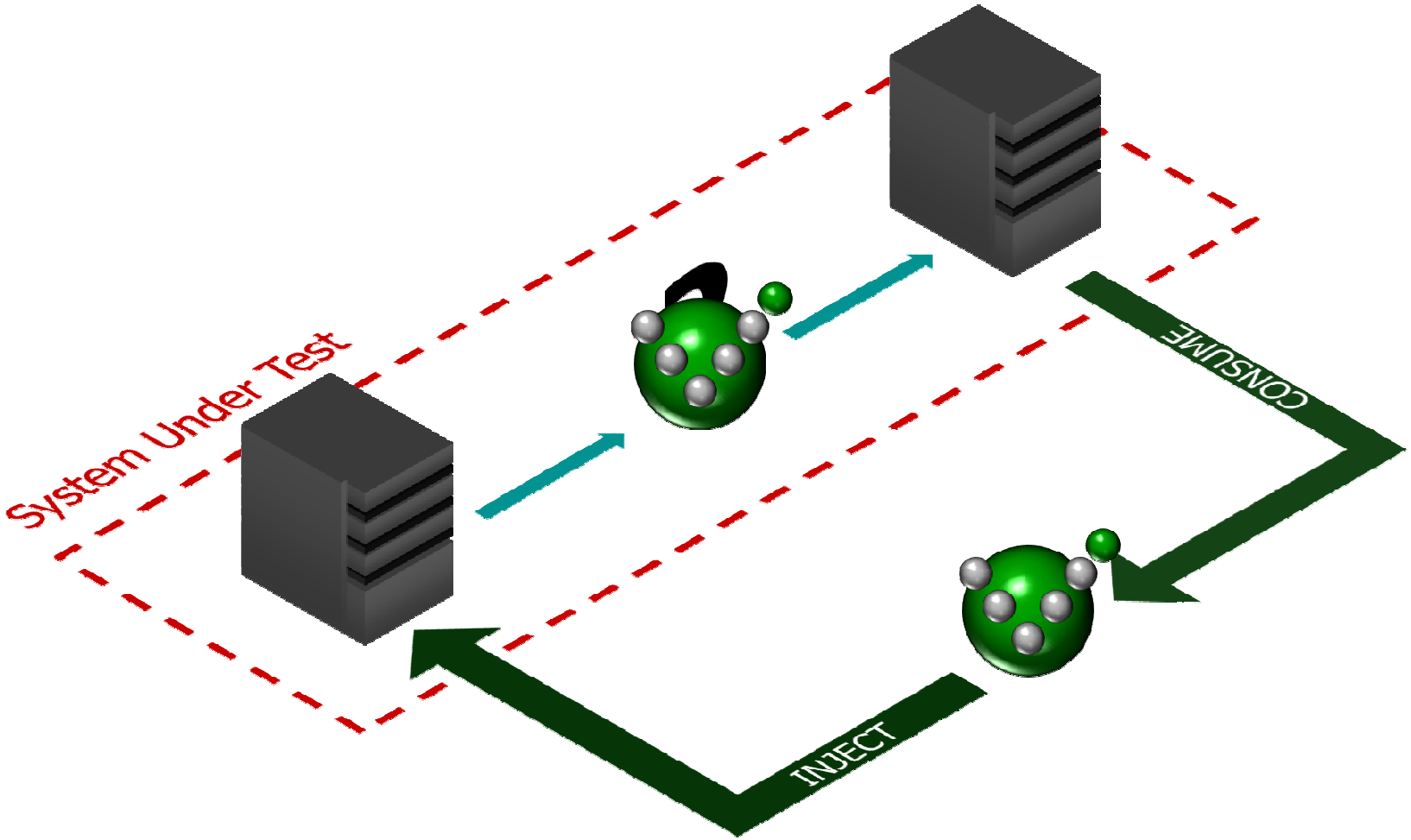
Virtualization



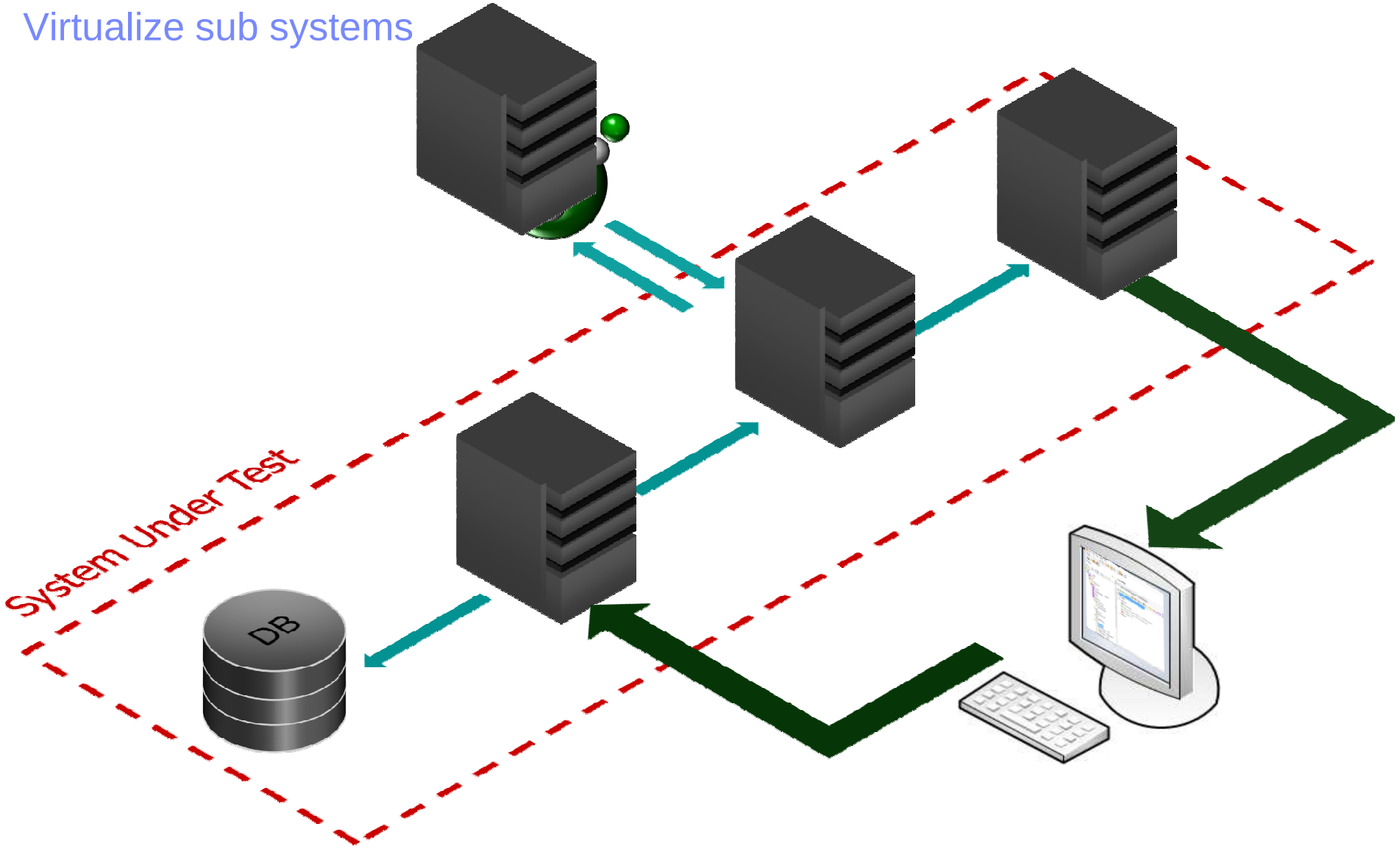
Blockers for e2e testing



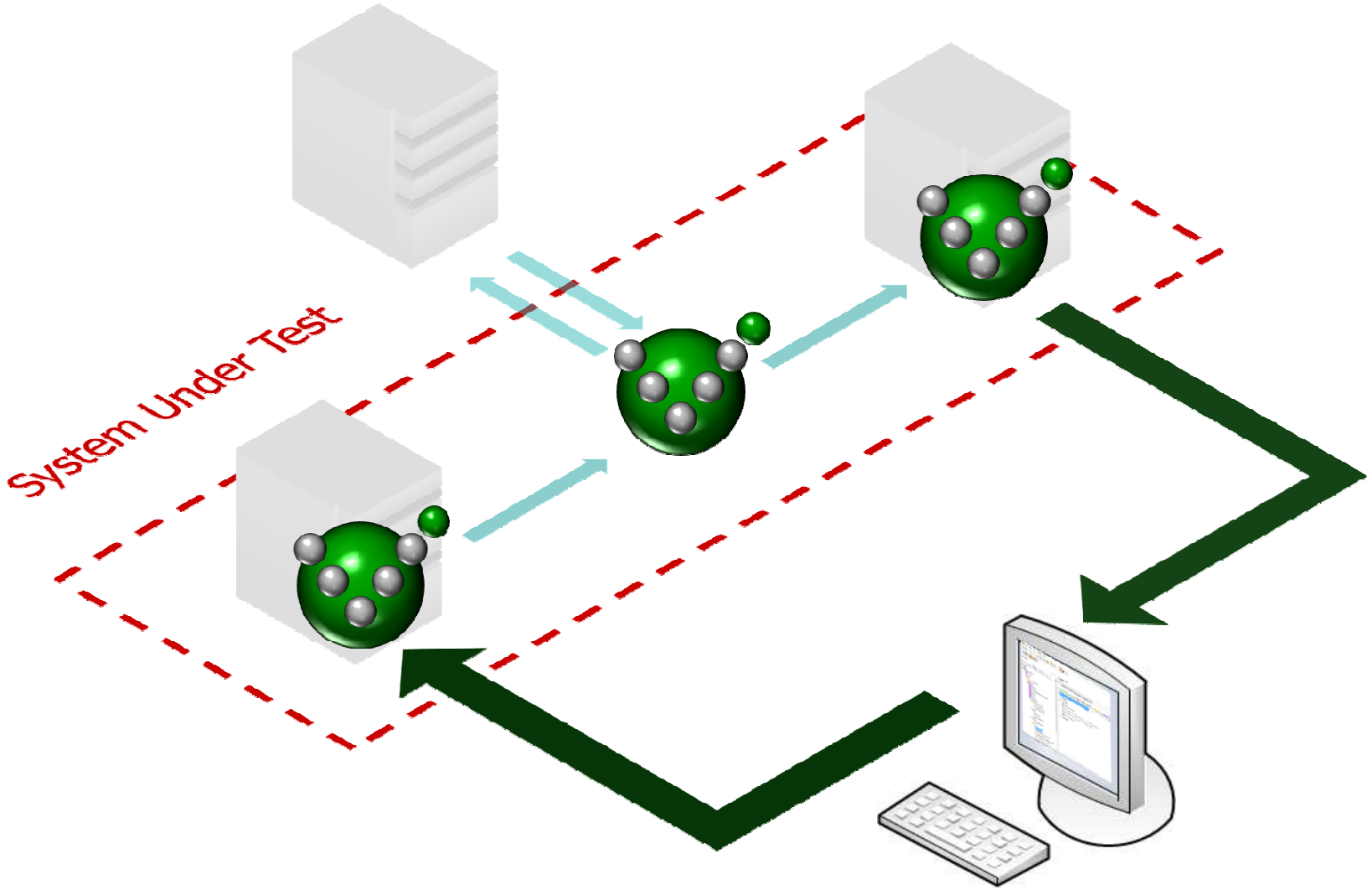
Stubbing



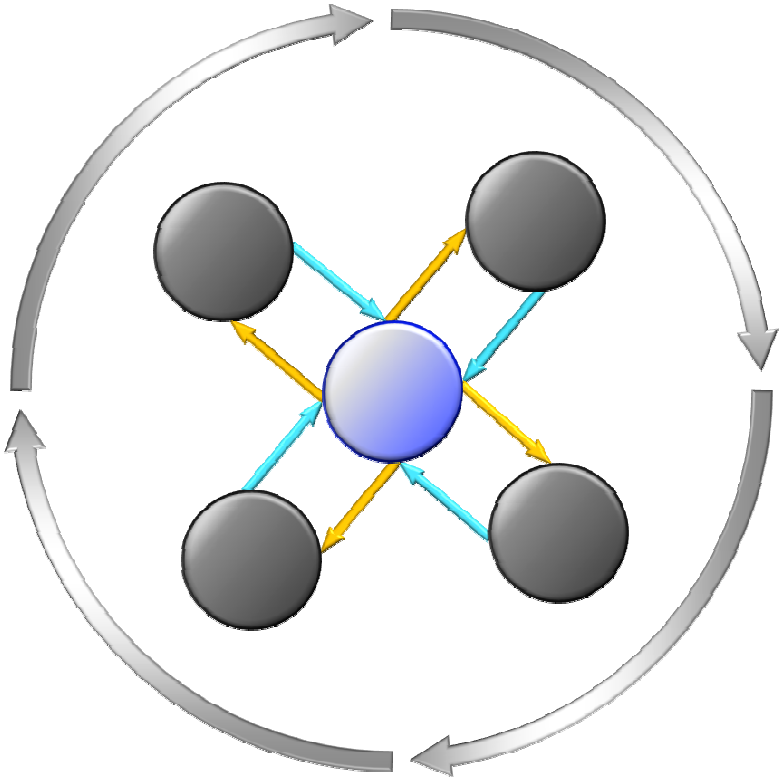
Virtualize sub systems



Virtualize entire application/layers



Incremental Testing



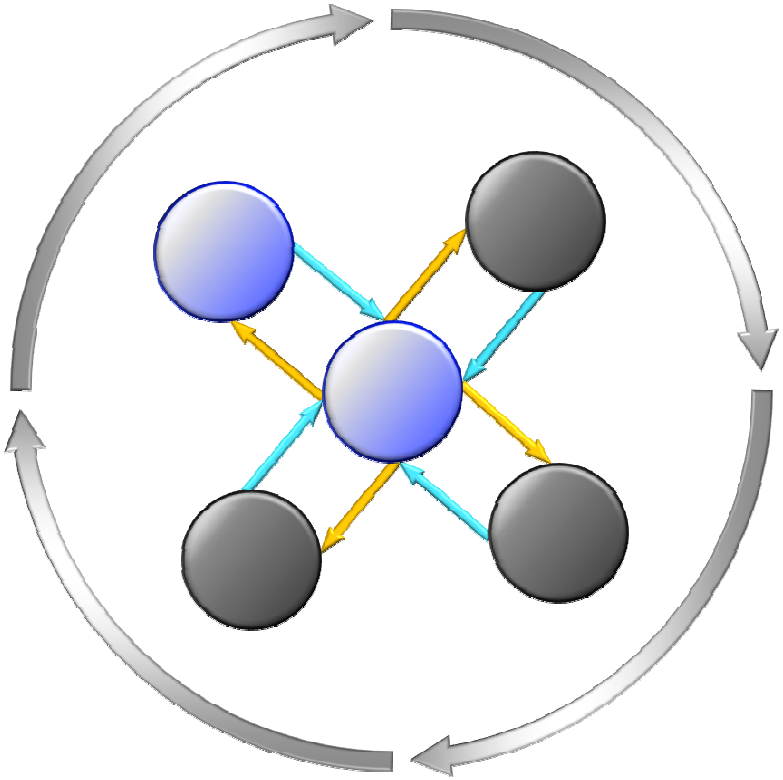
Actual Component

Virtualized Component

Initial Stage:

A single component can be tested in the context of an end to end environment, supplied by a set of virtualized components.

Incremental Testing



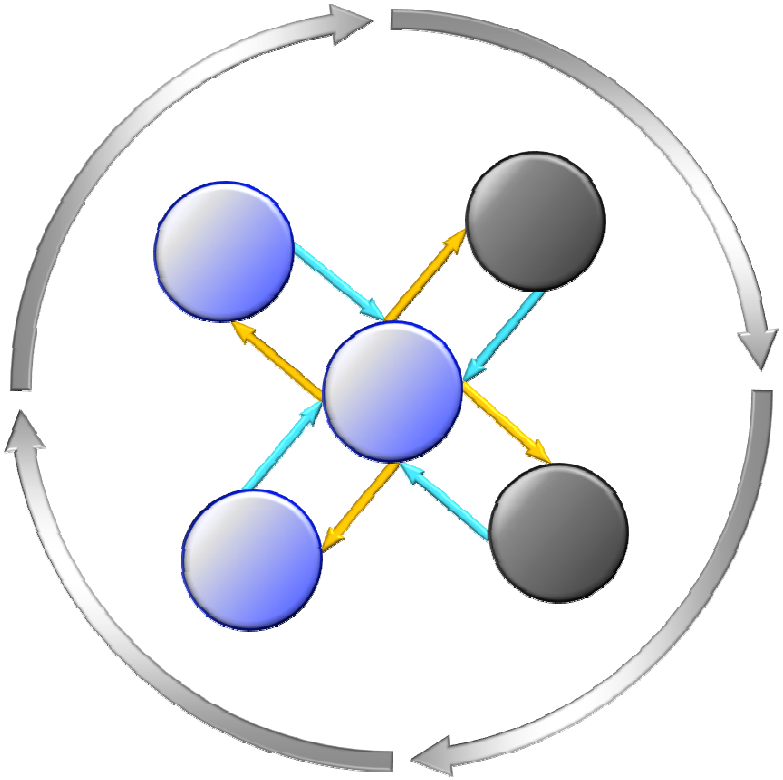
Actual Component

Virtualized Component

As components are built:

The same end to end tests can be run, replacing virtualized components with actual components.

Incremental Testing



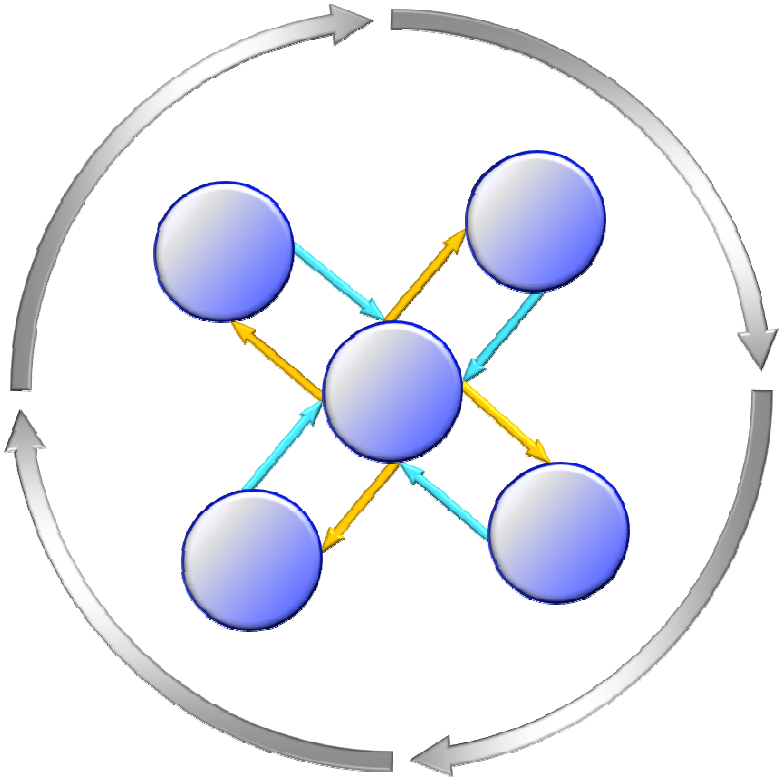
Actual Component

Virtualized Component

As components are built:

This enables us to test downstream dependencies as they are built.

Incremental Testing



Actual Component

Virtualized Component

When the system is complete:

End to end testing can be carried out with fewer surprises and lower risk

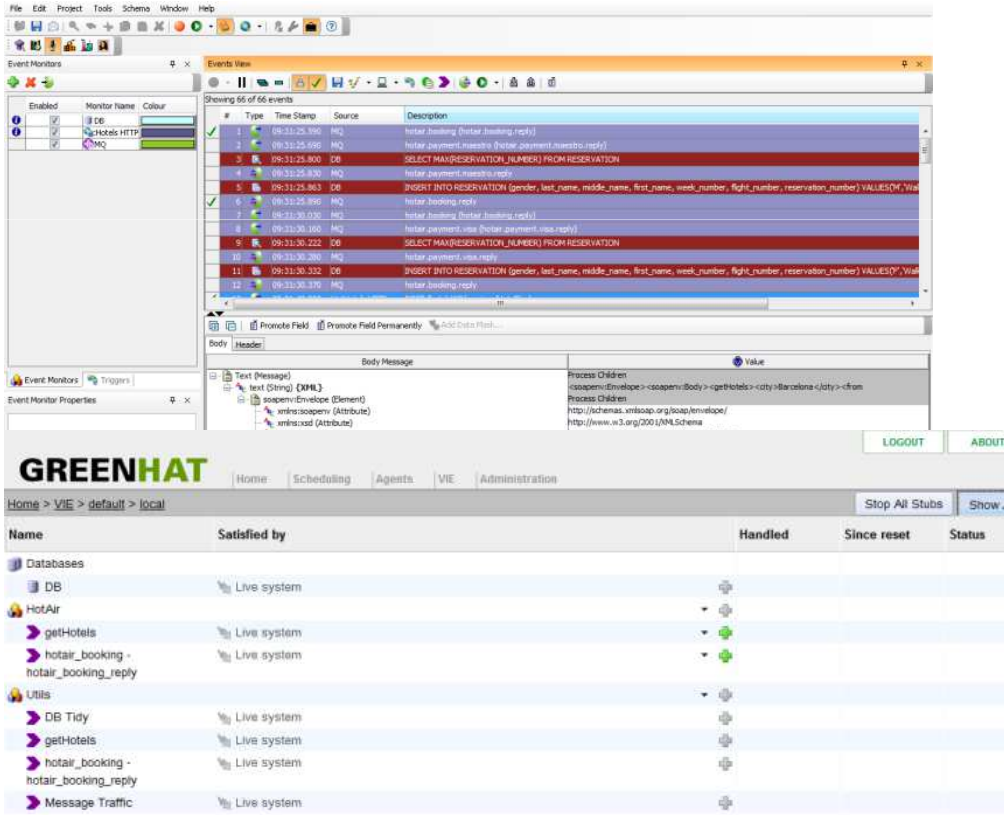
Virtual Application Types

Simple	Hard-coded response returned for given input
Non-deterministic	One-of-n hard-coded responses
Data driven	Input and/or output data specified in external data source (Excel, file, database)
Model driven, stateful	Input and/or output data kept in data model with complex relationships. Supports CRUD and other stateful behavior
Behavioural	Extends model-driven to provide pre-packaged functionality, e.g. shopping basket, real-time data feed, trading exchange, order matching

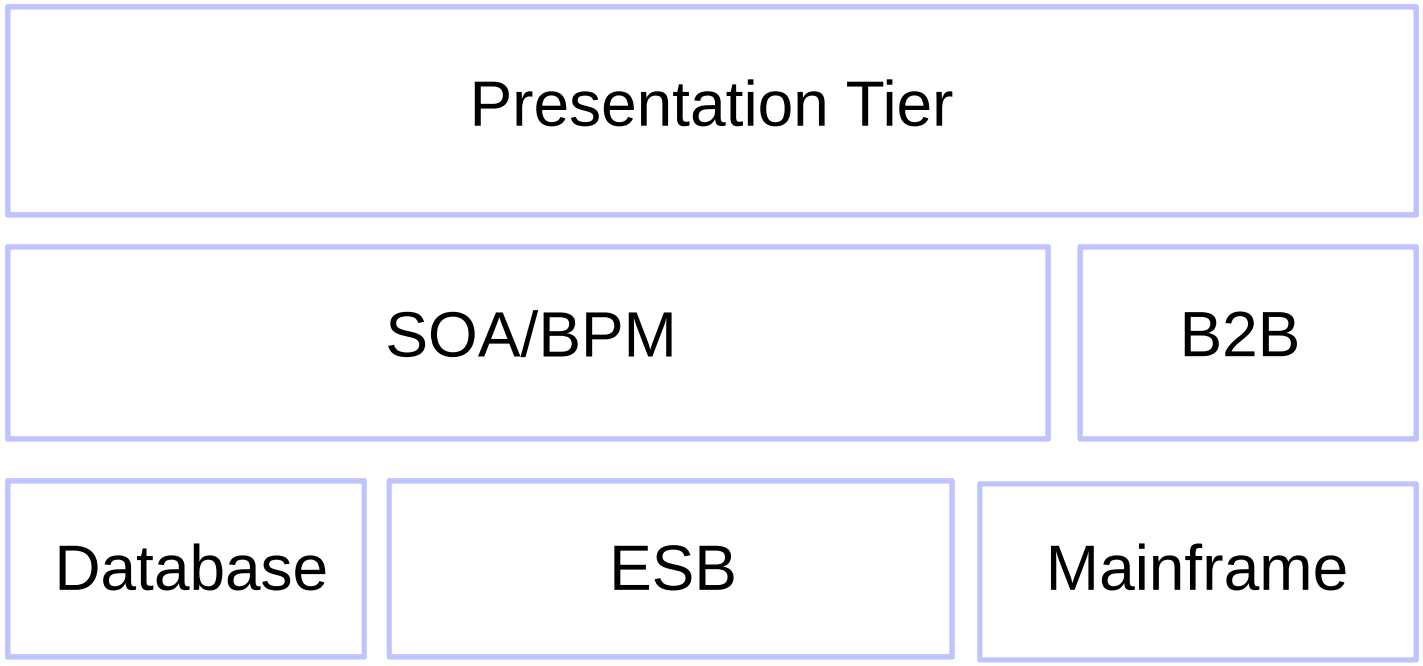
What is GH VIE?

"VIE is a means of virtualising individual or complete application domains and dependent services that are understandable and can be built without relying on teams of developers for coding"

- Tools for capture/analysis of the business domain
- Tools for automated generation of virtual applications
- Extensible & code free (Stateful, Stateless, Behavioural, Model Driven)
- Management interfaces to enable remote administration
- Repository to enable reuse and collaboration

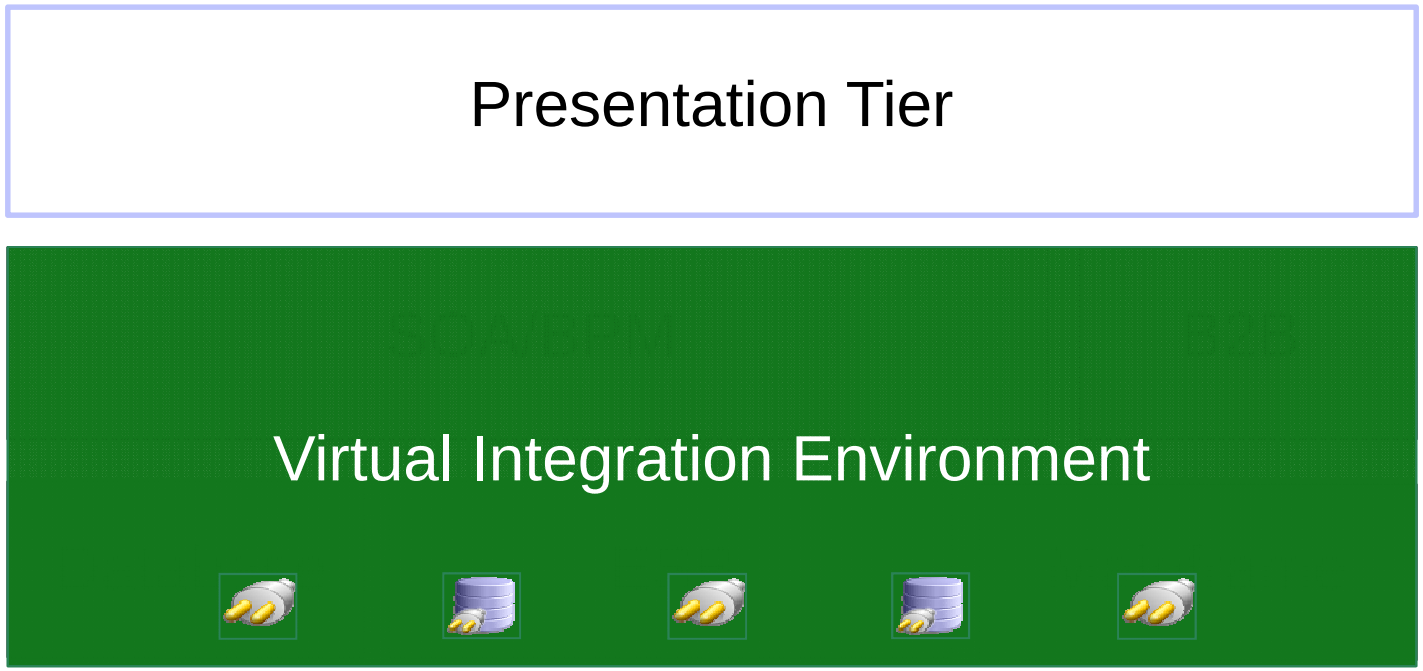


Candidate system



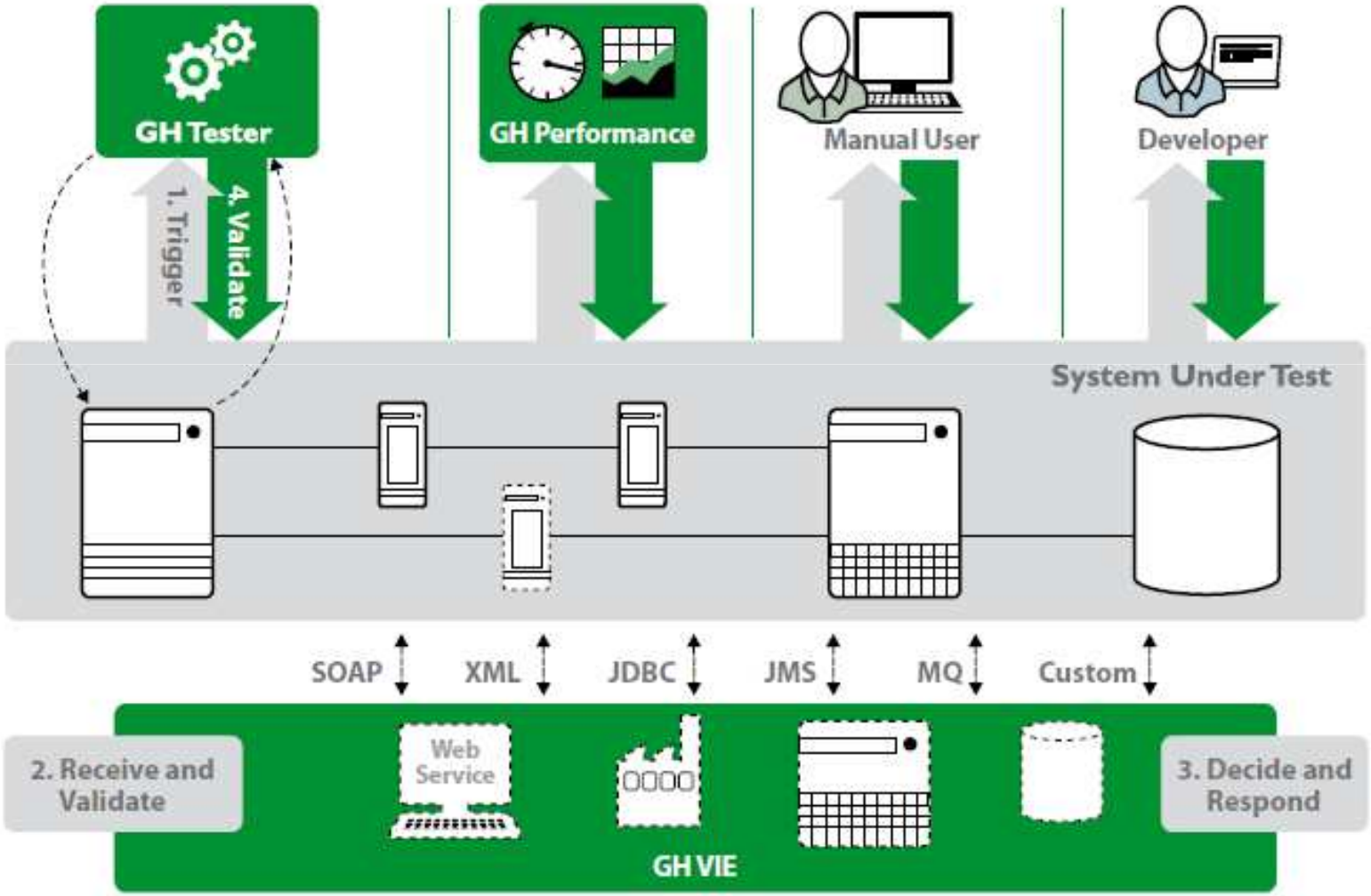
- Typical multi-tiered environment
- Problems with provisioning/access
- Problems with data consistency

Full environment virtualisation



- Virtualise entire set of applications
- Control data being returned to other systems
- Users unaware of virtualisation

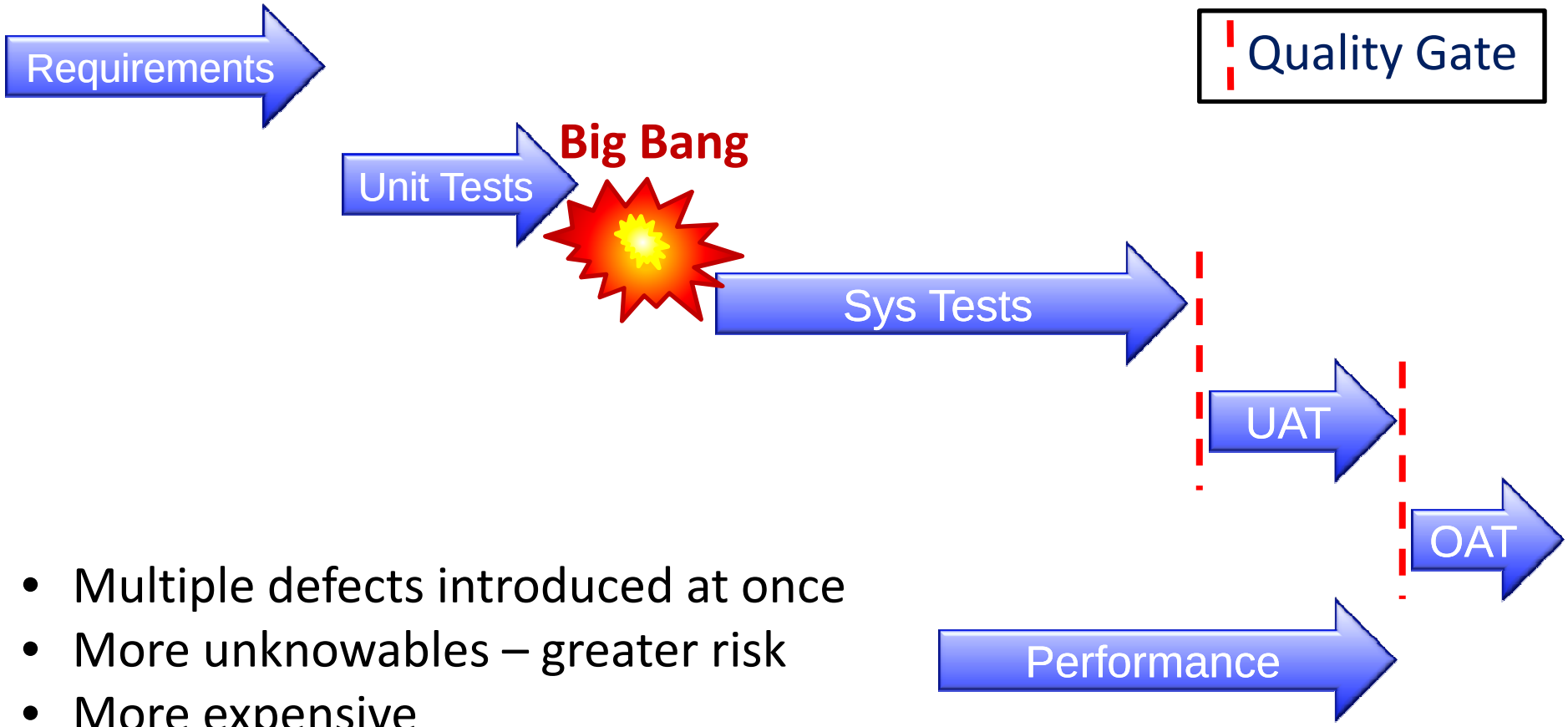
How it all fits together…



VIE Benefits…

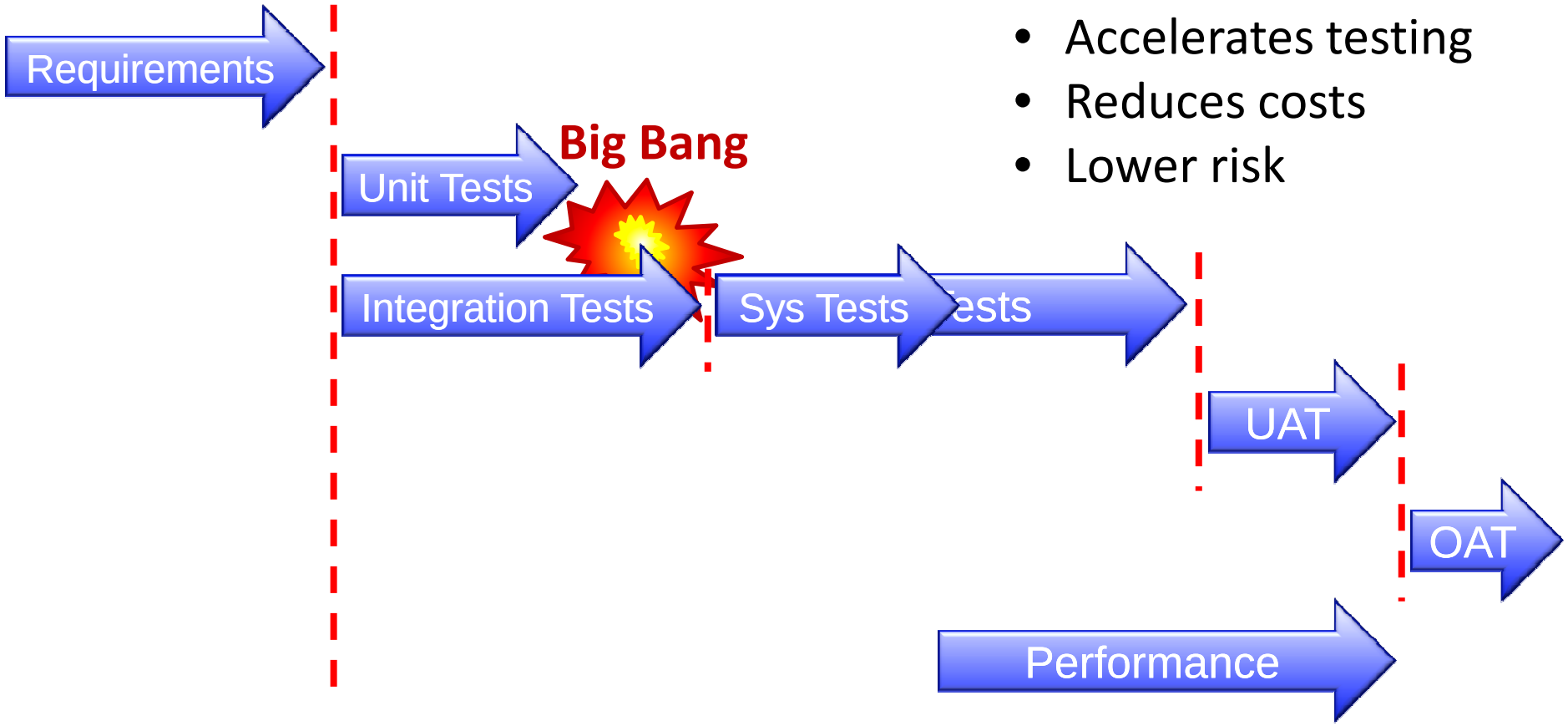
No need to modify application code	Variety of tools to enable creation of virtual service eg recording	Frees projects from external constraints eg databases
Simplifies management of Stub execution	Enables traceability over where stubs are running	Provides audit of who ran stubs and when
Enables simple versioning of stubs	Removes delays caused by late delivery of external interfaces	Speeds up development of stubs for testers and developers
Stubs can be reused and embellished to support different test cycles	Allows control over datasets being used across an environment	Enables erroneous data to be played back into systems
Removes the need for costly 3rd party interface leasing	Let's developers get on with developing code not stubs	Puts testers back in control and removes dependency on other teams
Easily simulate "+1" changes to interfaces and environments	Enables multi interface, complex and stateful simulation easily	**Limits risk and lowers integration issues when going into production**

The Old World



- Multiple defects introduced at once
- More unknowables – greater risk
- More expensive

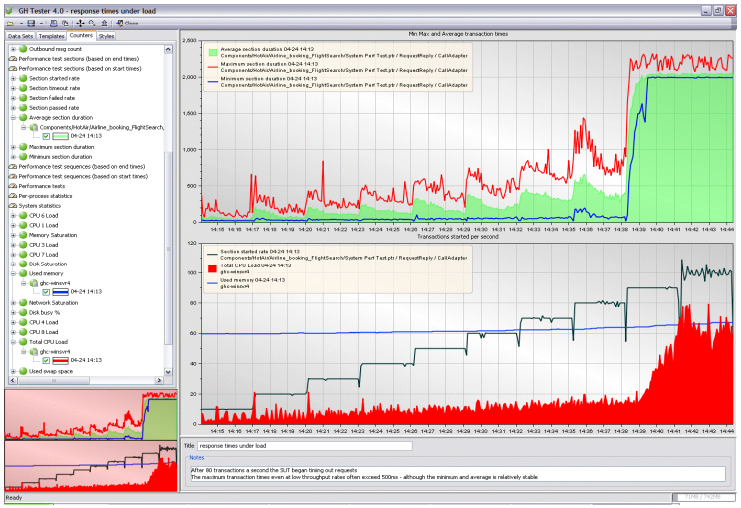
The New World



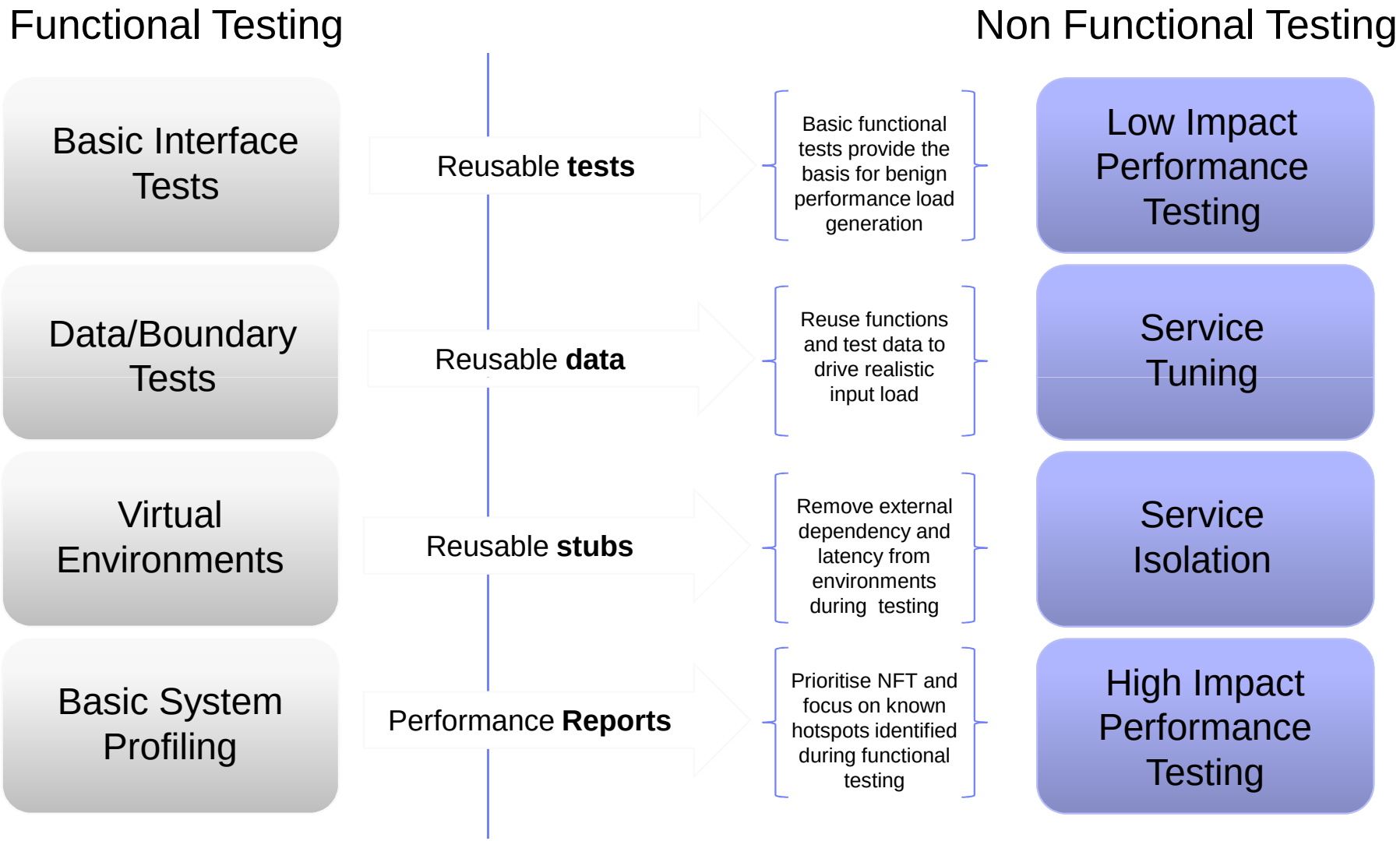
Performance Testing

Why you need a performance tool

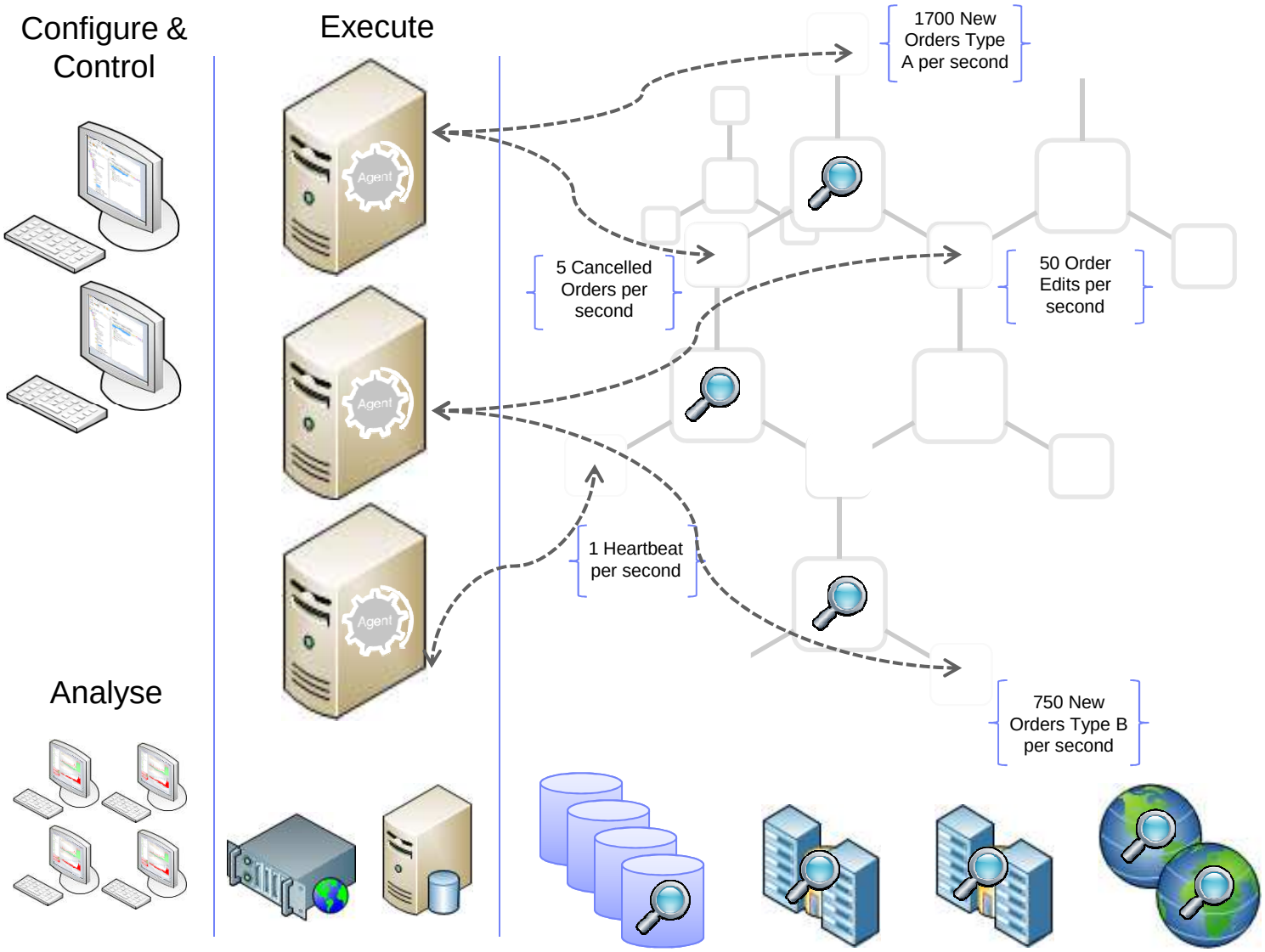
- Traditional performance testing happens too late!
- Individual component performance is important for service re-use and governance
- Realistic system performance testing is important to measure performance of full environments on shared infrastructure
  - Validate performance SLA's
  - Find and locate performance bottlenecks
  - Monitor changes over time – validate performance improvements
  - Validate performance across multiple operations concurrently
  - Simulate conditions for capacity planning
  - High performance service simulation



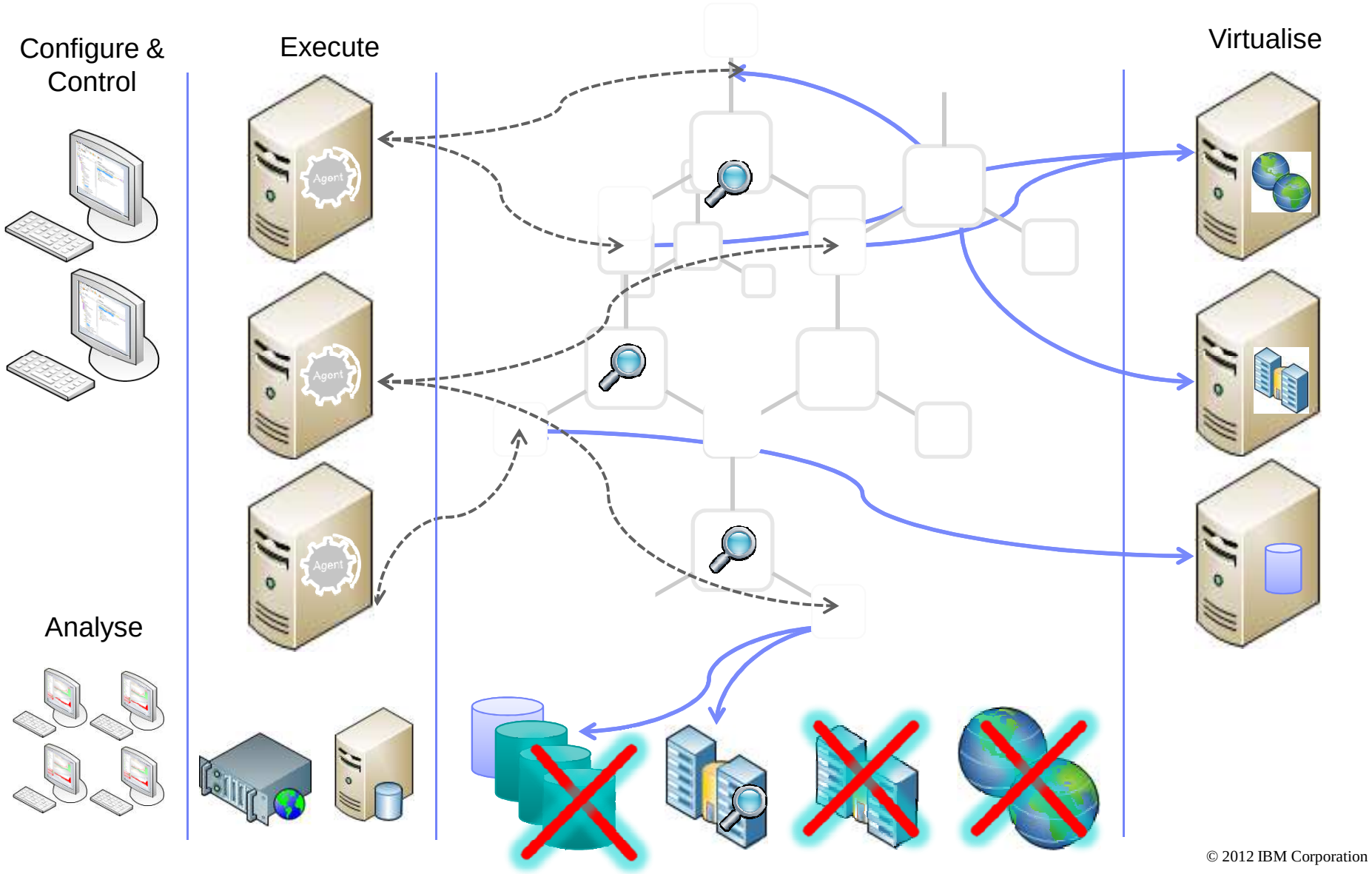
Where can you save time…

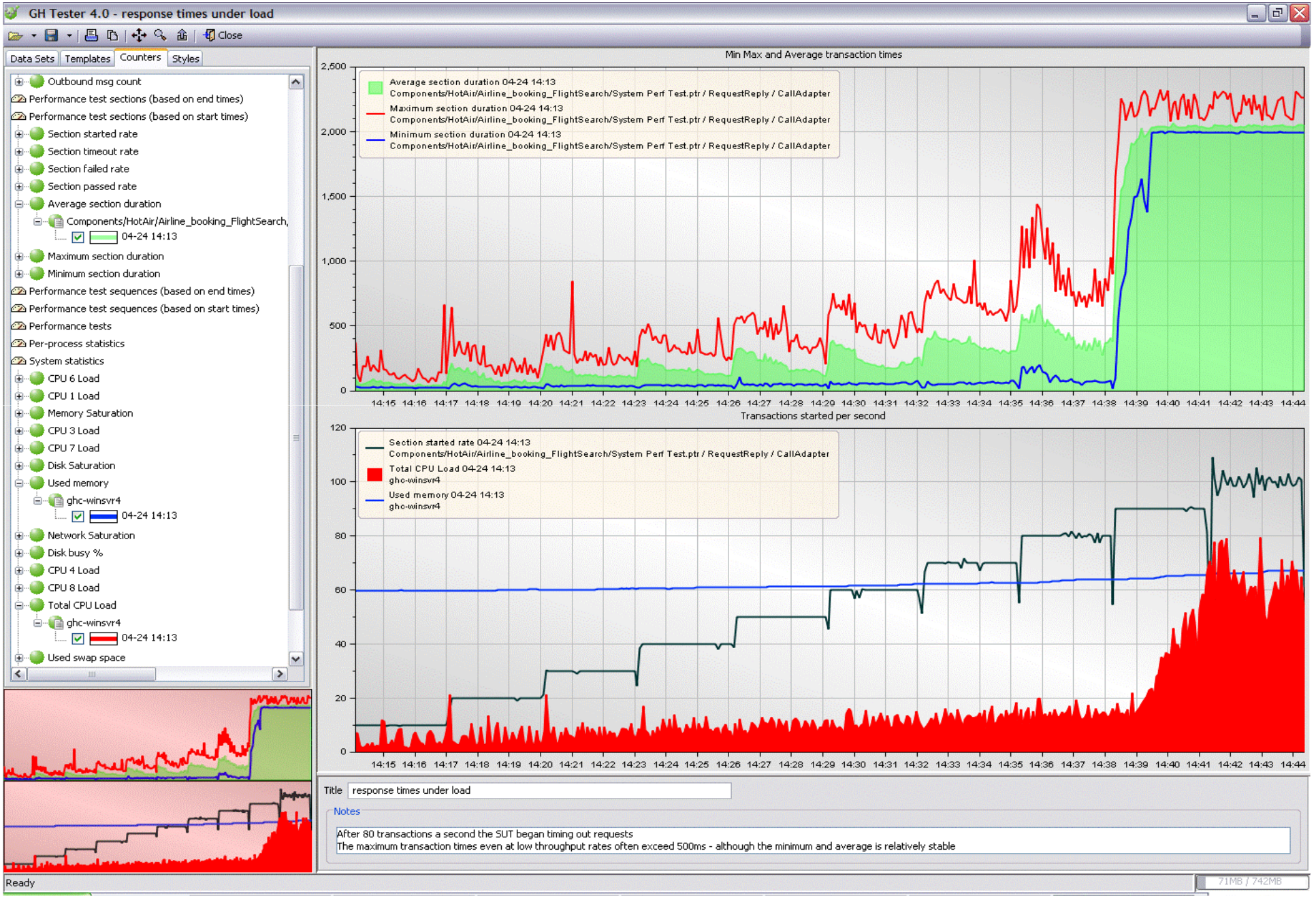


Execute and Analyse



Virtualise Dependencies




GH Tester 4.0 - response times under load
Close
Data Sets
Templates
Counters
Styles
Outbound msg count
Performance test sections (based on end times)
Performance test sections (based on start times)
Section started rate
Section timeout rate
Section failed rate
Section passed rate
Average section duration
Components/HotAir/Airline_booking_FlightSearch,
04-24 14:13
Maximum section duration
Minimum section duration
Performance test sequences (based on end times)
Performance test sequences (based on start times)
Performance tests
Per-process statistics
System statistics
CPU 6 Load
CPU 1 Load
Memory Saturation
CPU 3 Load
CPU 7 Load
Disk Saturation
Used memory
ghc-winsvr4
04-24 14:13
Network Saturation
Disk busy %
CPU 4 Load
CPU 8 Load
Total CPU Load
ghc-winsvr4
04-24 14:13
Used swap space
Min Max and Average transaction times
Average section duration 04-24 14:13
Components/HotAir/Airline_booking_FlightSearch/System Perf Test.ptr / RequestReply / CallAdapter
Maximum section duration 04-24 14:13
Components/HotAir/Airline_booking_FlightSearch/System Perf Test.ptr / RequestReply / CallAdapter
Minimum section duration 04-24 14:13
Components/HotAir/Airline_booking_FlightSearch/System Perf Test.ptr / RequestReply / CallAdapter
2,500
2,000
1,500
1,000
500
0
14:15 14:16 14:17 14:18 14:19 14:20 14:21 14:22 14:23 14:24 14:25 14:26 14:27 14:28 14:29 14:30 14:31 14:32 14:33 14:34 14:35 14:36 14:37 14:38 14:39 14:40 14:41 14:42 14:43 14:44
Transactions started per second
Section started rate 04-24 14:13
Components/HotAir/Airline_booking_FlightSearch/System Perf Test.ptr / RequestReply / CallAdapter
Total CPU Load 04-24 14:13
ghc-winsvr4
Used memory 04-24 14:13
ghc-winsvr4
120
100
80
60
40
20
0
14:15 14:16 14:17 14:18 14:19 14:20 14:21 14:22 14:23 14:24 14:25 14:26 14:27 14:28 14:29 14:30 14:31 14:32 14:33 14:34 14:35 14:36 14:37 14:38 14:39 14:40 14:41 14:42 14:43 14:44
Title
response times under load
Notes
After 80 transactions a second the SUT began timing out requests
The maximum transaction times even at low throughput rates often exceed 500ms - although the minimum and average is relatively stable
Ready
71MB / 742MB

Agenda

- Panorama des solutions de test de Rational
- Les tests d'intégration et les environnements de virtualisation des tests
- Démonstration
- Questions / réponses

Démonstration GreenHat

Slides de présentation

Agenda

- Panorama des solutions de test de Rational
- Les tests d'intégration et les environnements de virtualisation des tests
- Démonstration
- Questions / réponses

Green Hat customers include…

Yves Holvoet
Consultant, IBM Rational

Jean-Michel Oliva
Consultant, IBM Rational

La matinale du test logiciel

Virtualisation, automatisation et intégration continue des environnements de test