

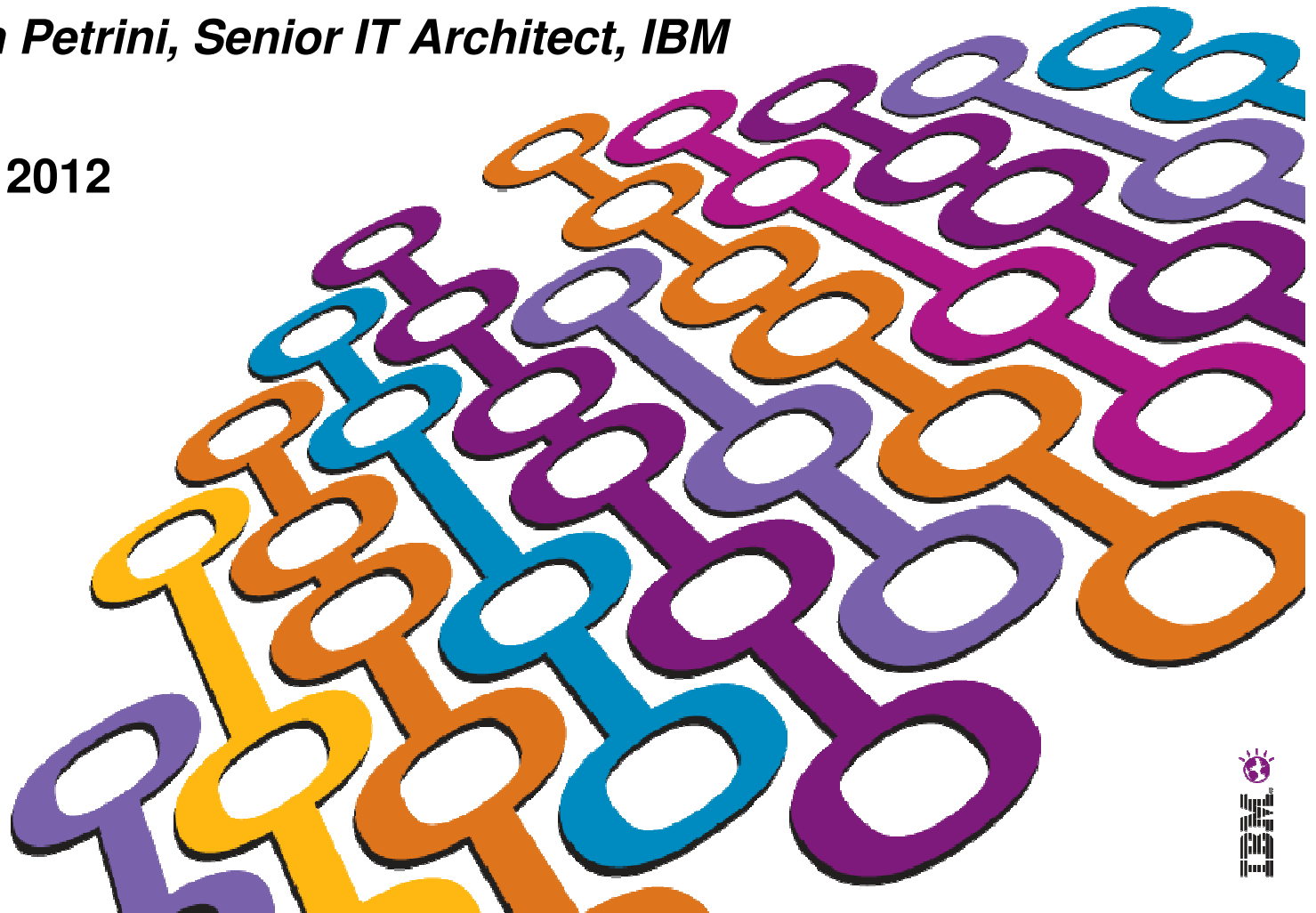
## Transitioning from integration and SOA to BPM

**Presenter:** Kim Clark, Senior IT Specialist, IBM ([kim.clark@uk.ibm.com](mailto:kim.clark@uk.ibm.com))

***Co-author: Brian Petrini, Senior IT Architect, IBM***

**Date:** 21<sup>st</sup> March 2012

**Version:** 1.1





# Agenda

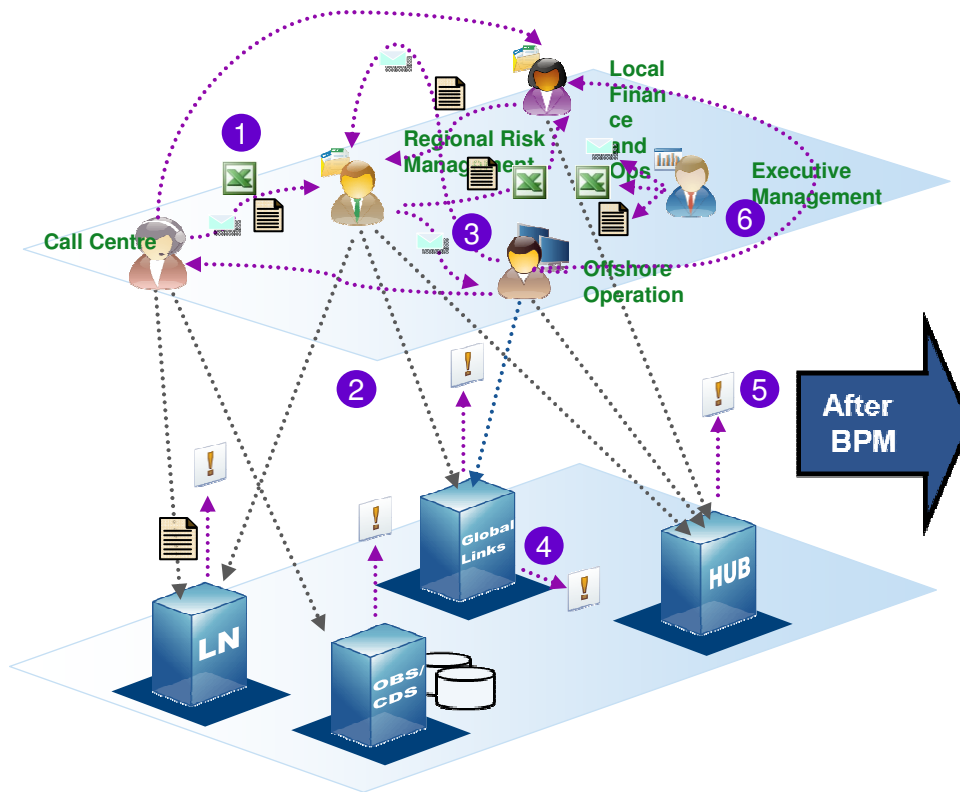
- Brief introduction to Business Process Management (BPM)
- Where have companies got to on their SOA journey?
- Is BPM a different type of service consumer?
- How do we technically characterise services?
- How would we expose services differently for BPM requesters?
- Resources and assistance



# BPM eases process execution across disparate systems providing a process oriented view

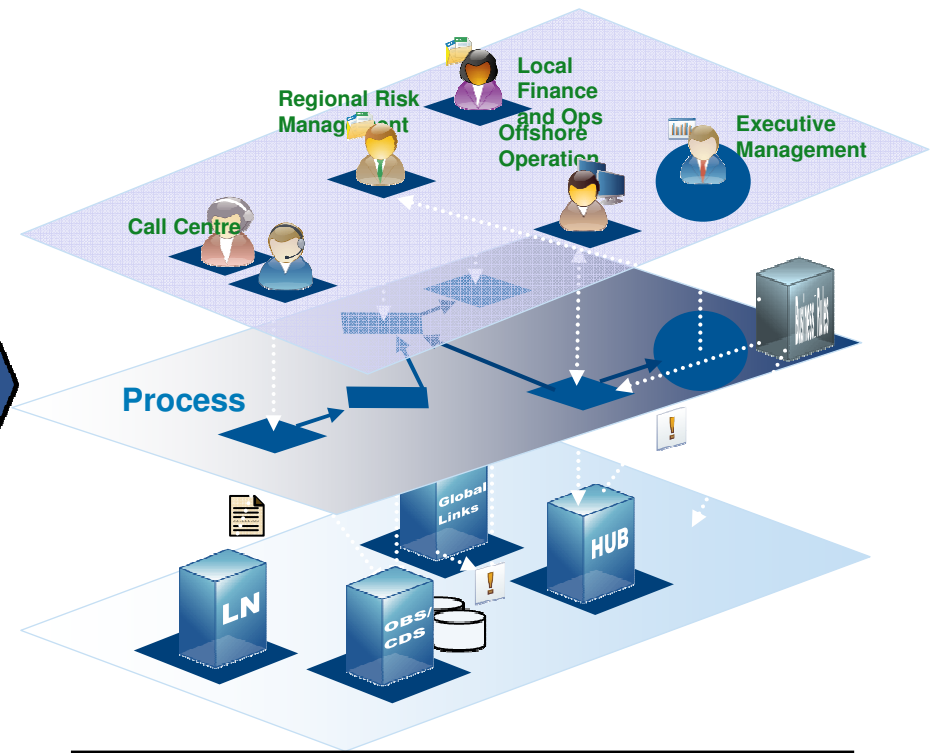


Current State



*Hidden work → Process Variation → Complexity → Missing Information → Chaos*

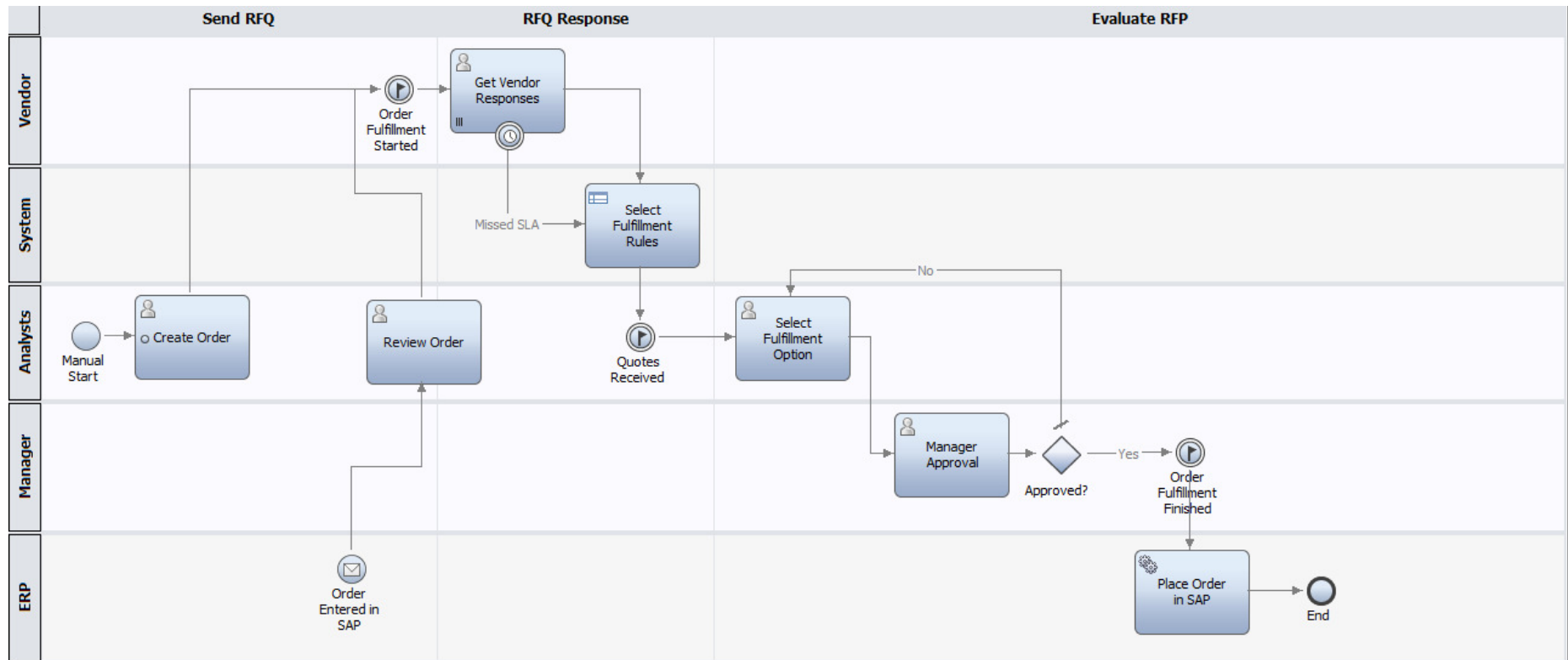
Future State



*Process Definition → Management Orchestration → Defined Work Segments → Measurable Results*



# Business Process Modelling Notation (BPMN)



More than just a diagramming standard  
- an real executable, monitored, scalable process



# Although “*process*” is at the core, there is more to BPM

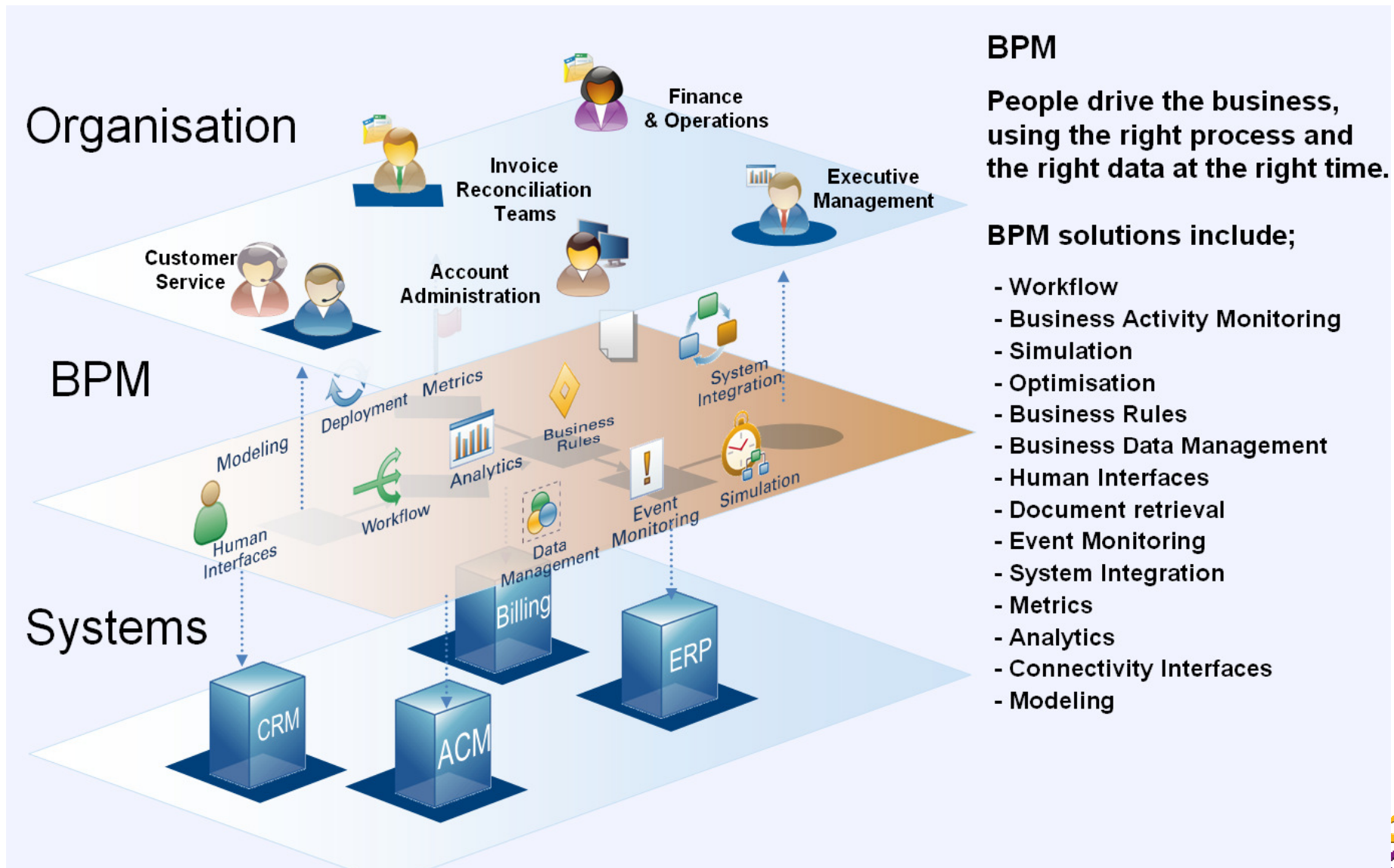


## BPM

People drive the business, using the right process and the right data at the right time.

BPM solutions include;

- Workflow
- Business Activity Monitoring
- Simulation
- Optimisation
- Business Rules
- Business Data Management
- Human Interfaces
- Document retrieval
- Event Monitoring
- System Integration
- Metrics
- Analytics
- Connectivity Interfaces
- Modeling

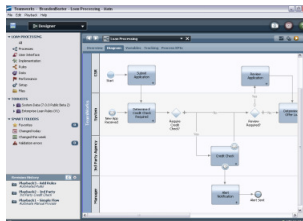




# IBM Business Process Manager



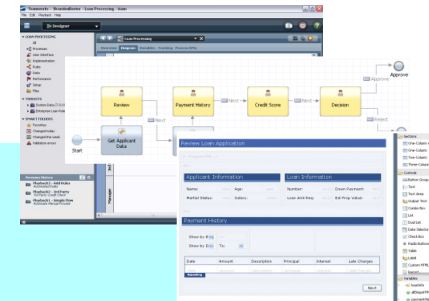
Business Modelers



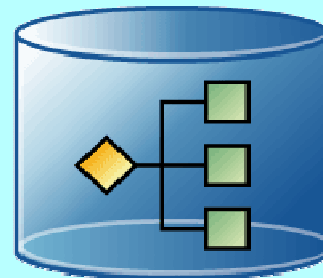
Business Modeler



Process Developers



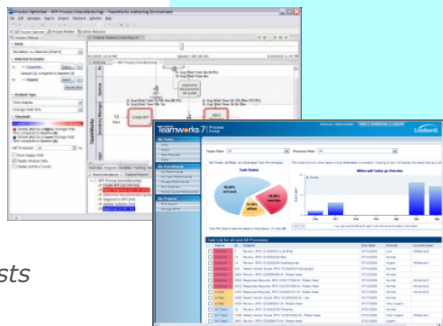
Graphical Implementation



Shared Process Model

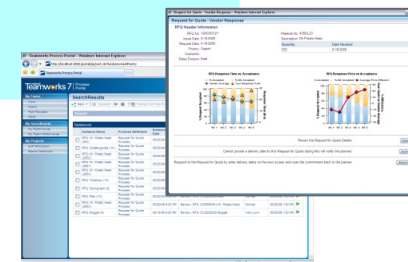


Managers / Analysts



Reports / Analytics

**"... the picture IS the process ..."**



User Portal / Forms



Users







# Cloud-based platform for collaboration on processes

## BlueworksLive

The screenshot displays the BlueworksLive web application interface. The top navigation bar includes 'Work', 'Community', and 'Library' tabs, along with a search bar and user options. The main content area shows a process diagram for 'Human Resources > Onboarding'. The diagram is a flowchart with steps like 'Add Candidate To HR Records', 'Orientation', 'New Hire Orientation Class', 'Benefits Overview Class', 'Watch Orientation CD', 'Input Employee Information into Database', 'Request Equipment', 'Set Employee Status To Active', 'Request Workspace', 'Request Access Badge', 'Provide Orientation CD', and 'plan'. The diagram is organized into lanes for different roles: Background, Candidate, HR Docs Team, HR Specialist, HR Team, Facilities Team, and Unassigned. A details panel on the right for the 'Set Employee Status To Active' task lists participants (HR Specialist, PeopleSoft), business owners (Human Resources), cycle time (30 Minutes), and inputs (Cleared Checklist, Employee Address, New Hire Package). A comment section at the bottom shows a conversation between Chaz Ripoll and Jason Butler.





## Agenda

- Brief introduction to Business Process Management (BPM)
- Where have companies got to on their SOA journey?
- Is BPM a different type of service consumer?
- How do we technically characterise services?
- How would we expose services differently for BPM requesters?
- Resources and assistance

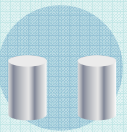
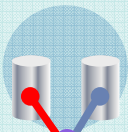
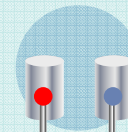
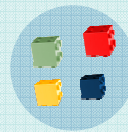
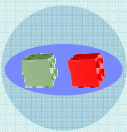
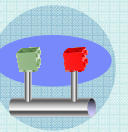
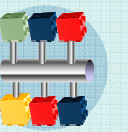




# Service Integration Maturity Model (SIMM)

## *Where are you?*



							
	<b>Silo</b>	<b>Integrated</b>	<b>Componentized</b>	<b>Services</b>	<b>Composite Services</b>	<b>Virtualized Services</b>	<b>Dynamically Re-Configurable Services</b>
<b>Business View</b>	Function Oriented	Function Oriented	Function Oriented	Service Oriented	Service Oriented	Service Oriented	Service Oriented
<b>Organization</b>	Ad hoc IT Governance	Ad hoc IT Governance	Ad hoc IT Governance	Emerging SOA Governance	SOA and IT Governance Alignment	SOA and IT Governance Alignment	SOA and IT Governance Alignment
<b>Methods</b>	Structured Analysis & Design	Object Oriented Modeling	Component Based Development	Service Oriented Modeling	Service Oriented Modeling	Service Oriented Modeling	Grammar Oriented Modeling
<b>Applications</b>	Modules	Objects	Components	Services	Process Integration via Services	Process Integration via Services	Dynamic Application Assembly
<b>Architecture</b>	Monolithic Architecture	Layered Architecture	Component Architecture	Emerging SOA	SOA	Grid Enabled SOA	Dynamically Re-Configurable Architecture
<b>Infrastructure</b>	Platform Specific	Platform Specific	Platform Specific	Platform Specific	Platform Specific	Platform Neutral	Dynamic Sense & Respond
	<b>Level 1</b>	<b>Level 2</b>	<b>Level 3</b>	<b>Level 4</b>	<b>Level 5</b>	<b>Level 6</b>	<b>Level 7</b>

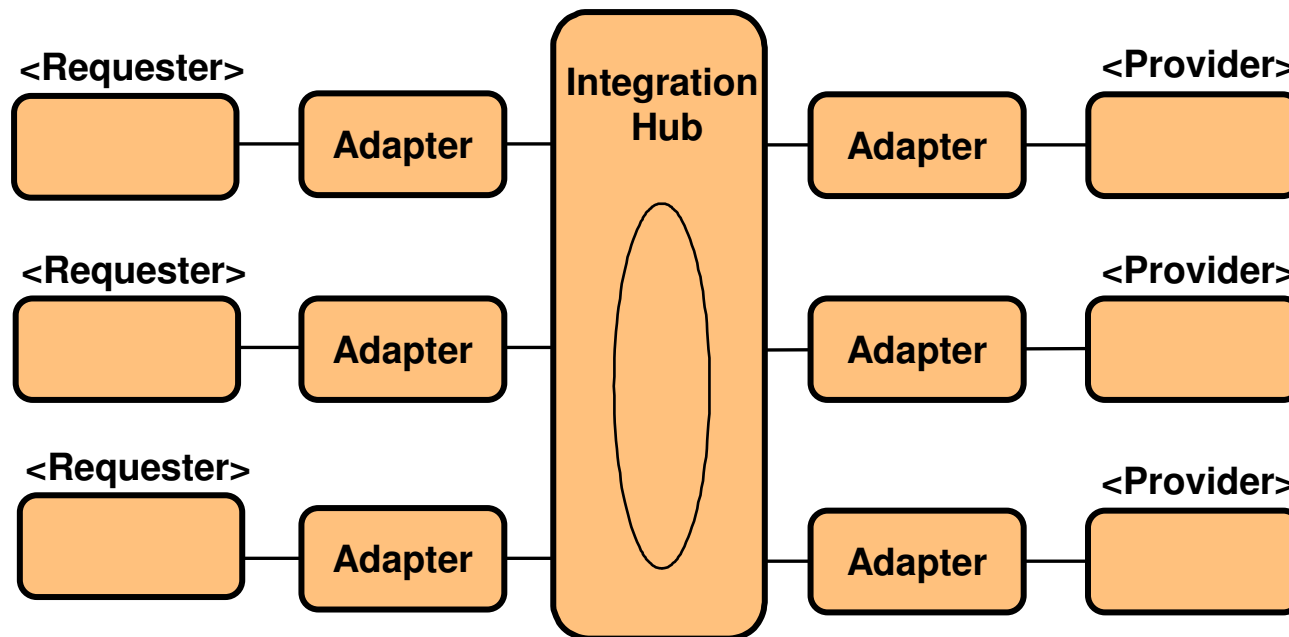
“What does a maturing SOA look like...”

[http://www.ibm.com/developerworks/websphere/library/techarticles/0904\\_clark/0904\\_clark.html](http://www.ibm.com/developerworks/websphere/library/techarticles/0904_clark/0904_clark.html)

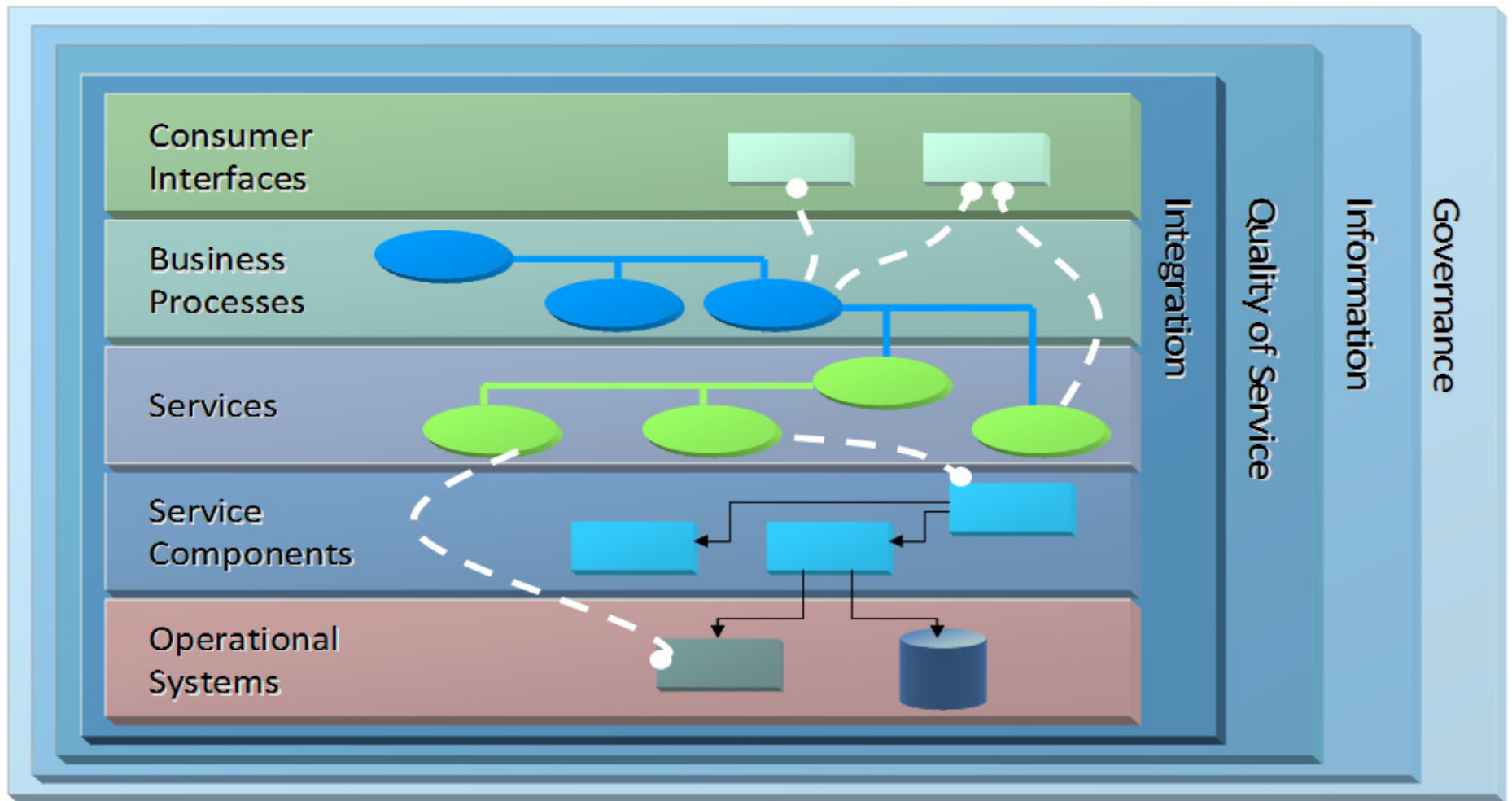


# Enterprise Application Integration (EAI)

## – Hub and Spoke Pattern



# Service Oriented Architecture (SOA) Reference Architecture

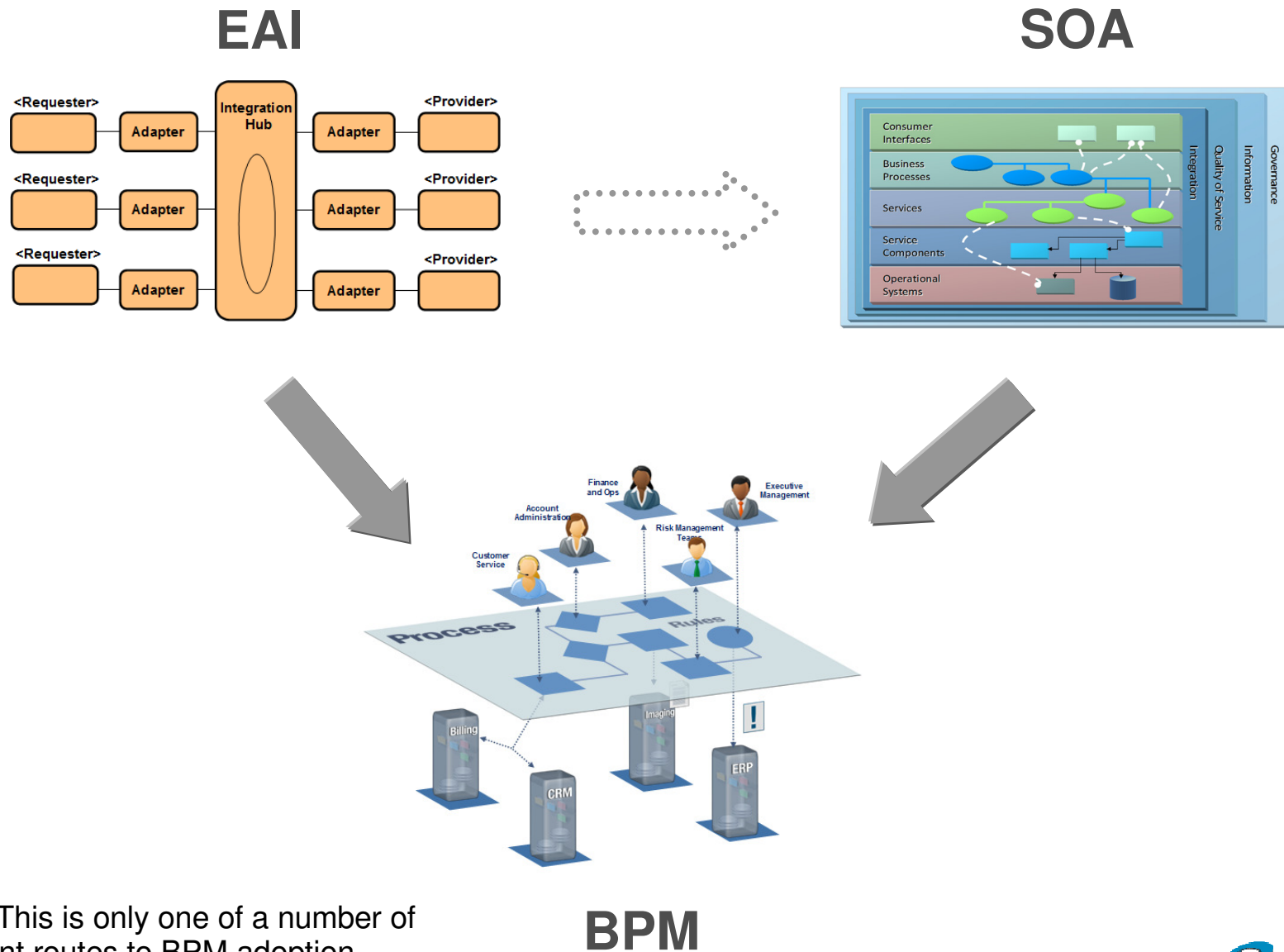


(C) The Open Group 2009

<https://collaboration.opengroup.org/projects/soa-ref-arch>



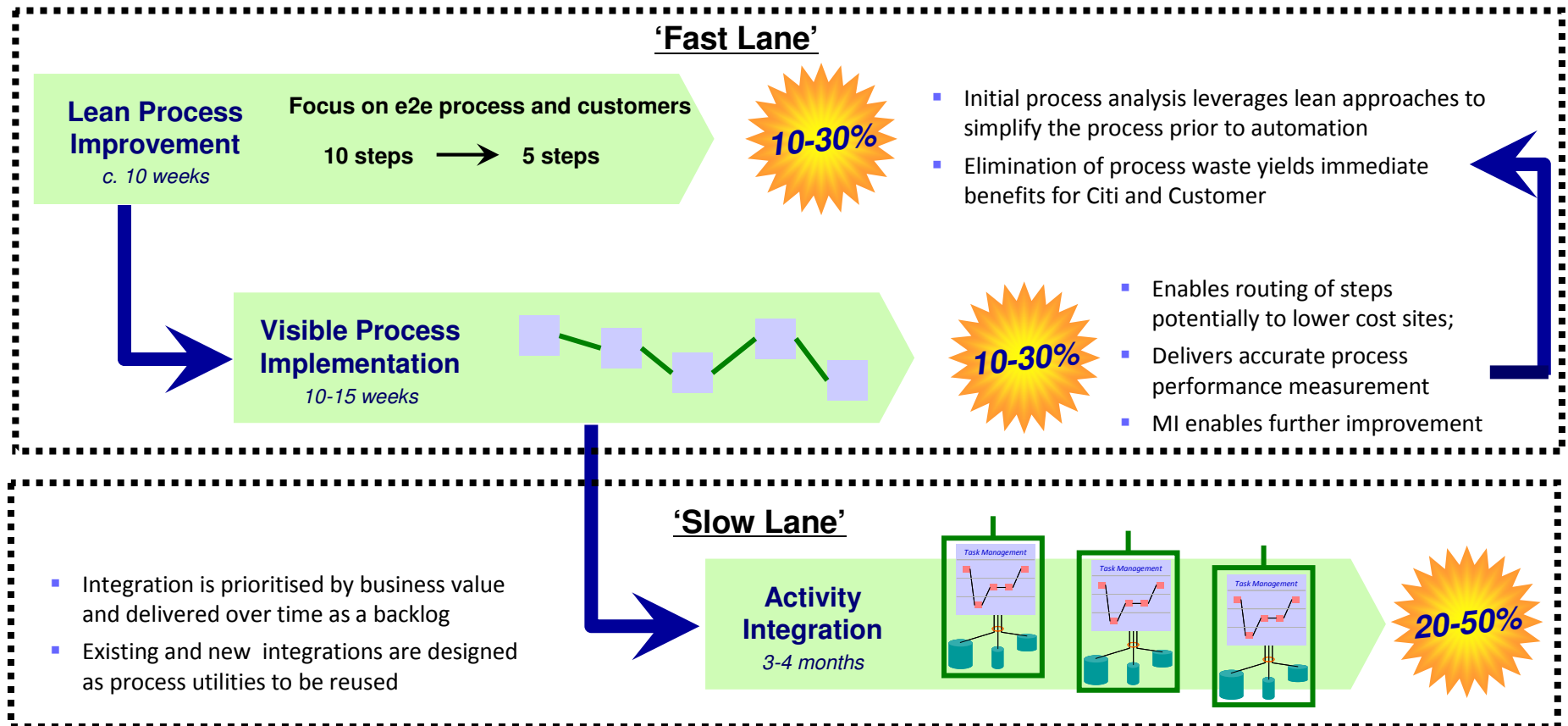
# Moving from integration and SOA to BPM



*Note:* This is only one of a number of different routes to BPM adoption



# Separate streams of activity for short and long term goals



- Each step delivers real benefit in its own right
- The combined benefit delivered by the combination of the three stages is higher than any can deliver alone



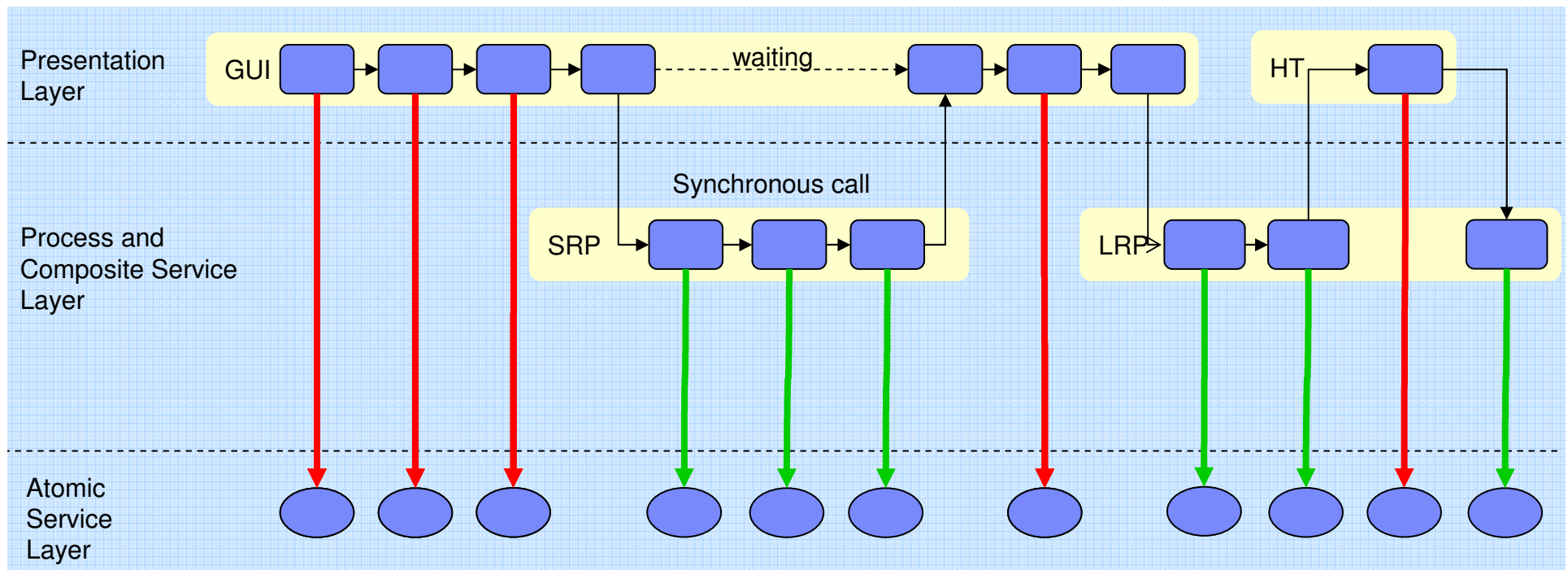


# Agenda

- Brief introduction to Business Process Management (BPM)
- Where have companies got to on their SOA journey?
- Is BPM a different type of service consumer?
- How do we technically characterise services?
- How would we expose services differently for BPM requesters?
- Resources and assistance



# How different are UIs and Processes as requesters?



GUI Graphical User Interface    HT Human Task    SRP Short-running Process    LRP Long-running Process





# Differences in requesters requirements between users and automated processes



- Who's in control
  - In traditional systems, humans (via graphical interfaces) are requesters and back end systems are providers.
  - In BPM, the BPM engine running the process is the requester. Providers can either be back end systems or humans. Humans are given tasks by the BPM system.
  - Note however, that when working on the task, the human may be working with UIs that call services.
- Time/Resource Management
  - If users own the flow, bad at managing their time. They usually wait for a response.
  - A process is very efficient at managing time. Always distributes work asynchronously.
- Exception handling
  - Humans are good at retaining a dialog despite unexpected situations. Great at improvising and handling diverse exception scenarios.
  - Processes are structured – only handle the exception scenarios that are defined in advance. Can't think for themselves.
- Versioning of data structures
  - Humans can dynamically handle changes in version that perhaps provide more or less data, or place data in different places or present it in an unstructured form.
  - Processes fail if core data structures alter. Resilience to changing data structures has to be designed in.
- Quality of output
  - Humans work with data inconsistently and are only (just!) bounded by the validation.
  - Processes provide a consistent predictable behavior for a given set of data. Important for automation of the next step in line, for management information and for process compliance.
- Context switching
  - Humans are bad at context switching and can get confused and cause data leakage errors.
  - Process engines switch context with precision and therefore can ensure high throughput across multiple processes.



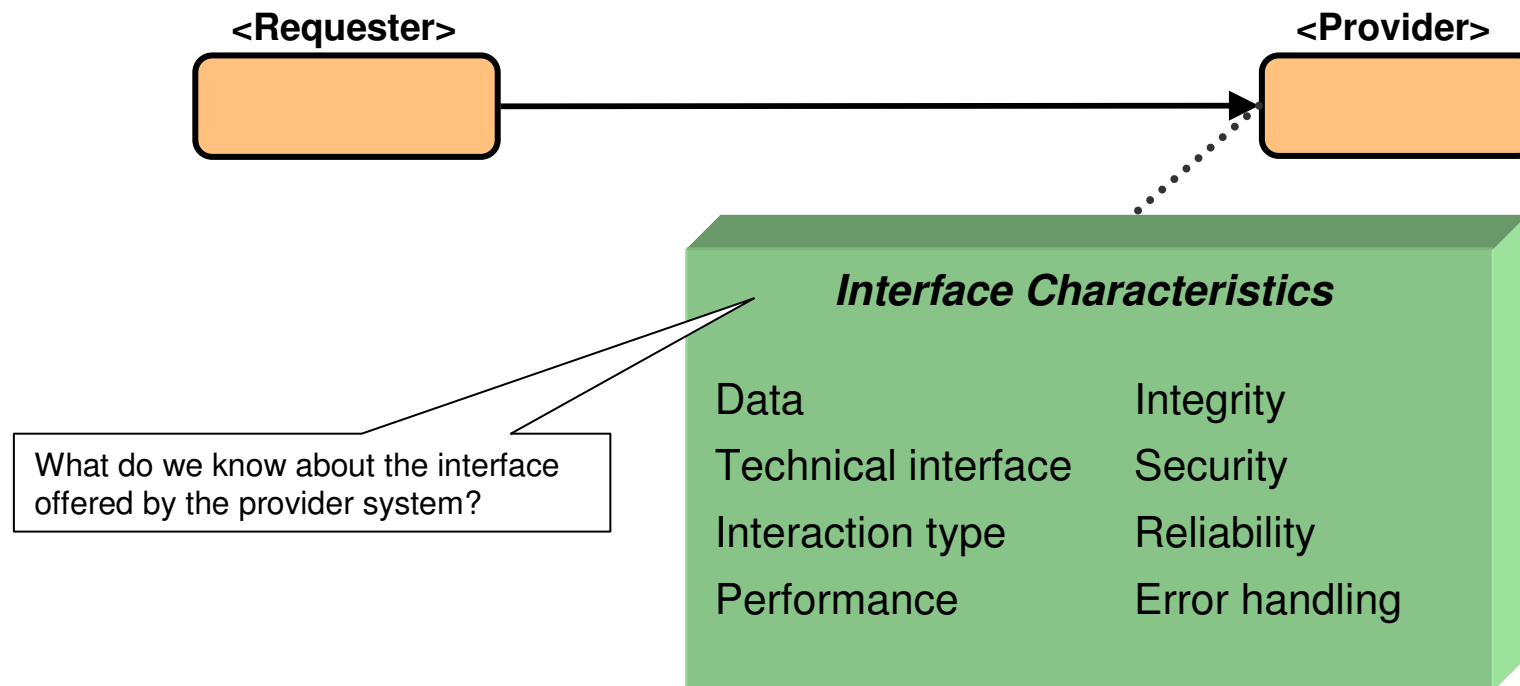


## Agenda

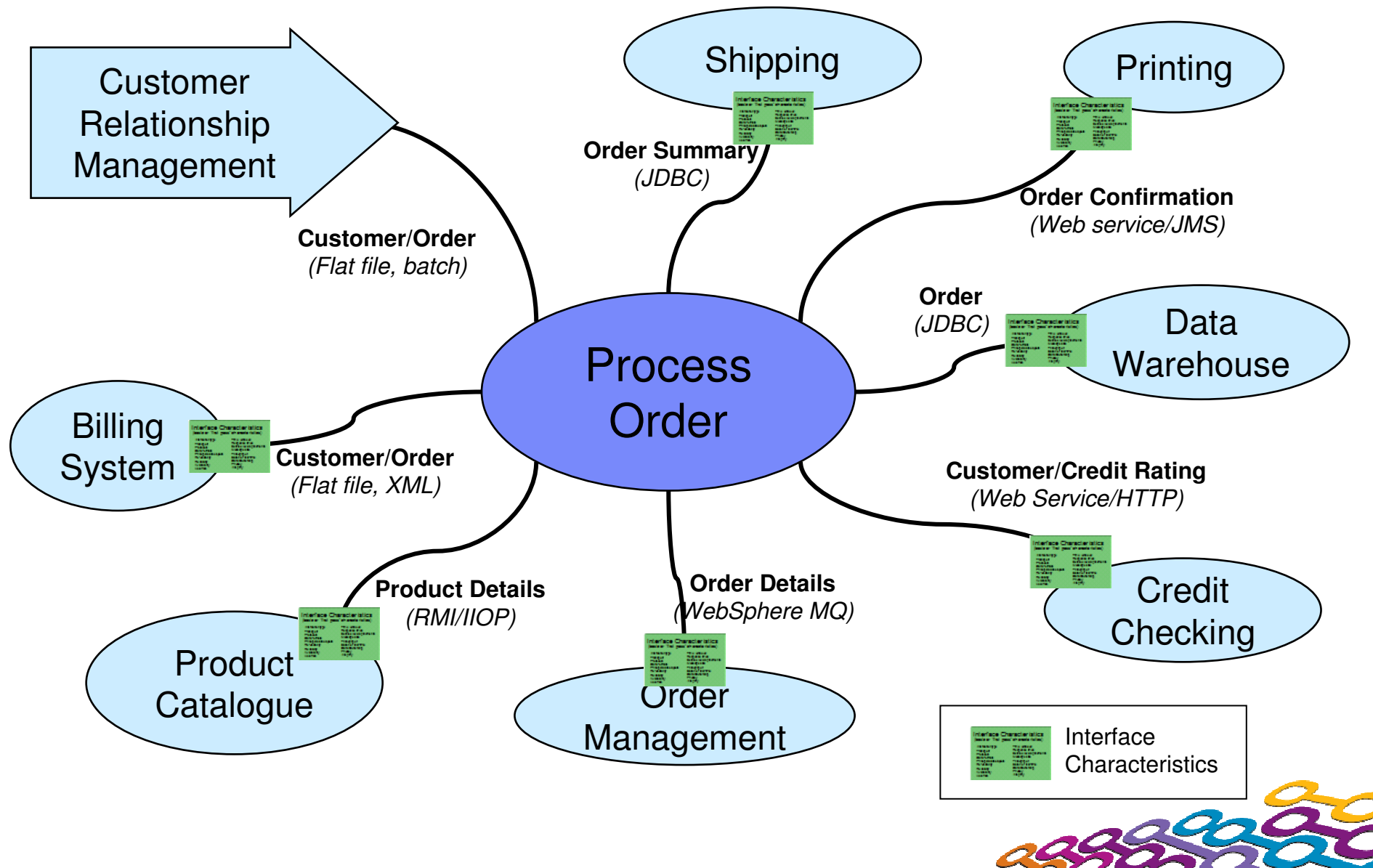
- Brief introduction to Business Process Management (BPM)
- Where have companies got to on their SOA journey?
- Is BPM a different type of service consumer?
- How do we technically characterise services?
- How would we expose services differently for BPM requesters?
- Resources and assistance



# Using interface characteristics to analyse interactions



# Solution Context Diagram with Integration Detail



# Interface Characteristics

## *Functional Definition*

- Principal data objects
- Operation/function
- Read or change
- Request/response objects

## *Technical Interface*

- Transport
- Protocol
- Data format

## *Interaction type*

- Request-response or fire-forget
- Thread-blocking or asynchronous
- Batch or individual

## *Performance*

- Response times
- Throughput
- Volumes
- Concurrency
- Message size

## *Integrity*

- Validation
- Transactionality
- Statefulness
- Event Sequence
- Idempotence

## *Security*

- Identity/Authentication
- Authorisation
- Data Ownership
- Privacy

## *Reliability*

- Availability
- Delivery assurance

## *Error Handling*

- Error Management capabilities
- Known exception conditions

“Interface Characteristics: Capturing Integration Complexity for BPM and SOA”

[http://www.ibm.com/developerworks/websphere/techjournal/1112\\_clark/1112\\_clark.html](http://www.ibm.com/developerworks/websphere/techjournal/1112_clark/1112_clark.html)





## Agenda

- Brief introduction to Business Process Management (BPM)
- Where have companies got to on their SOA journey?
- Is BPM a different type of service consumer?
- How do we technically characterise services?
- How would we expose services differently for BPM requesters?
- Resources and assistance



# Exposing services for BPM

## *Functional definition*

### Interface Characteristics (basic or "first pass" characteristics)


- Principal data objects
- Request/response objects
- Operation/function
- Validation
- Transactionality
- Statefulness
- Event Sequence
- Idempotence
- Data Ownership
- Privacy
- Mean time between failures
- Maintenance schedule
- Delivery assurance
- Synchronous or Asynchronous
- Request/Response or Fire/Forget
- Batch or real-time
- Response times
- Throughput
- Availability windows
- Volumes
- Message size
- Known exception conditions
- Error management capabilities
- Transport
- Protocols
- Data formats





# Exposing services for BPM

## *Functional definition*



Interface Characteristics (basic or "first pass" characteristics)	
• Principal data objects	• Synchronous or Asynchronous
• Request/response objects	• Request/Response or Fire/Forget
• Operation/function	• Batch or real-time
• Validation	• Response times
• Transactionality	• Throughput
• Statefulness	• Availability windows
• Event Sequence	• Volumes
• Idempotence	• Message size
• Data Ownership	• Known exception conditions
• Privacy	• Error management capabilities
• Mean time between failures	• Transport
• Maintenance schedule	• Protocols
• Delivery assurance	• Data formats

### ■ Human Requester of a Service

#### — Operation/function

- Synchronously exposed service may only have time to trigger work and provide acknowledgement.

#### — Request/response data

- Need enough data for the human Requester and may retrieve all details in a single call to reduce network hops.

### ■ Automated Process requesting a Service

#### — Operation/function

- Asynchronously exposed service may have time to complete it.

#### — Request/response data

- Need only contain keys and primary metadata. Services has time to retrieve detail. Process may need to retrieve latest data on response anyway. Etc.





#### Interface Characteristics (basic or "first pass" characteristics)

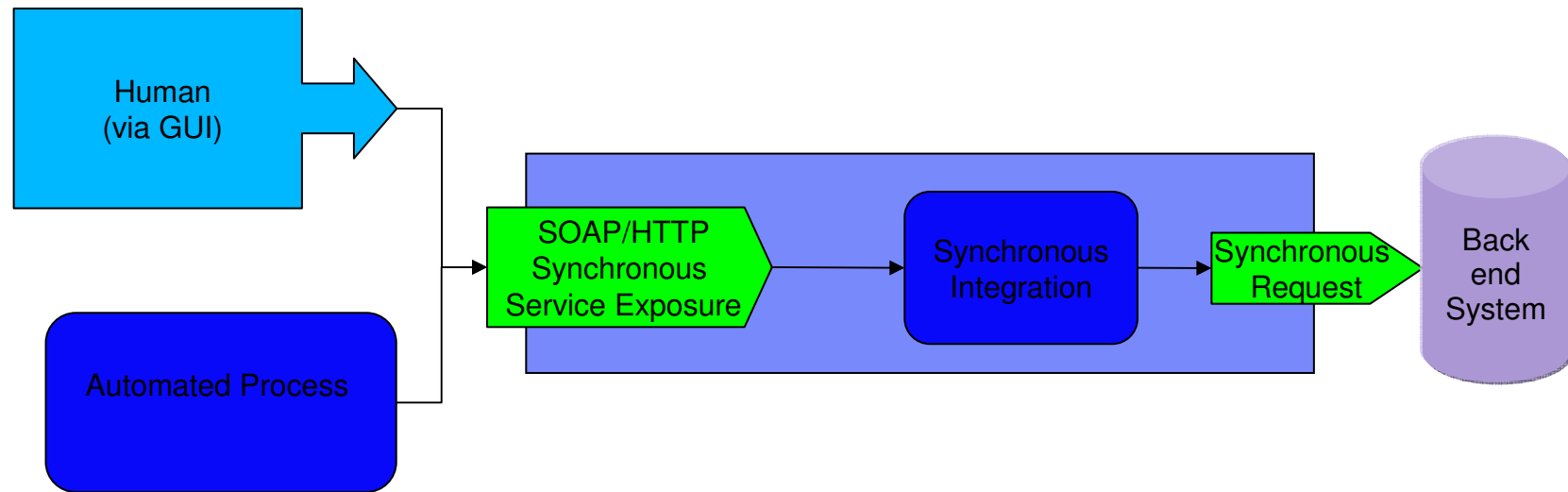
- Principal data objects
- Request/response objects
- Operation/function
- Validation
- Transactionality
- Statefulness
- Event Sequence
- Idempotence
- Data Ownership
- Privacy
- Mean time between failures
- Maintenance schedule
- Delivery assurance
- Synchronous or Asynchronous
- Request/Response or Fire/Forget
- Batch or real-time
- Response times
- Throughput
- Availability windows
- Volumes
- Message size
- Known exception conditions
- Error management capabilities
- Transport
- Protocols
- Data formats

# Exposing services for BPM

## *Interaction Style/Transport/Protocol*



# Synchronously Exposed Service

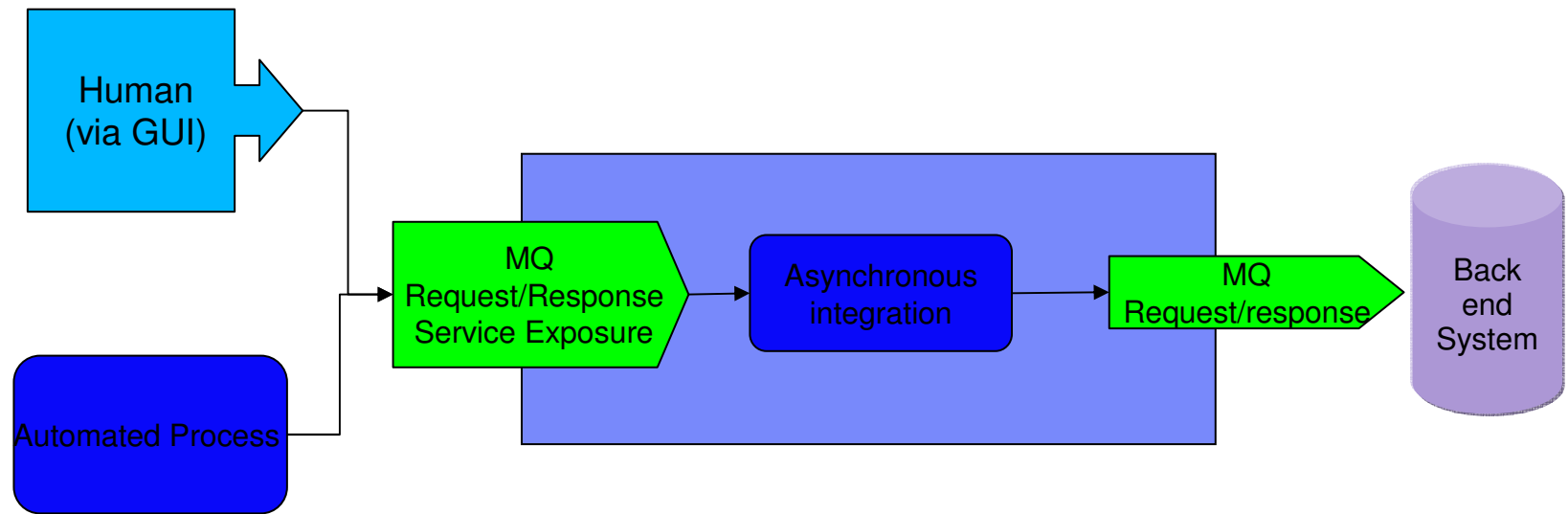


- Where will the error be surfaced if the service is unavailable? How will it be managed?
- How transactional is the interface? Could duplicates occur?





# Asynchronously exposed service

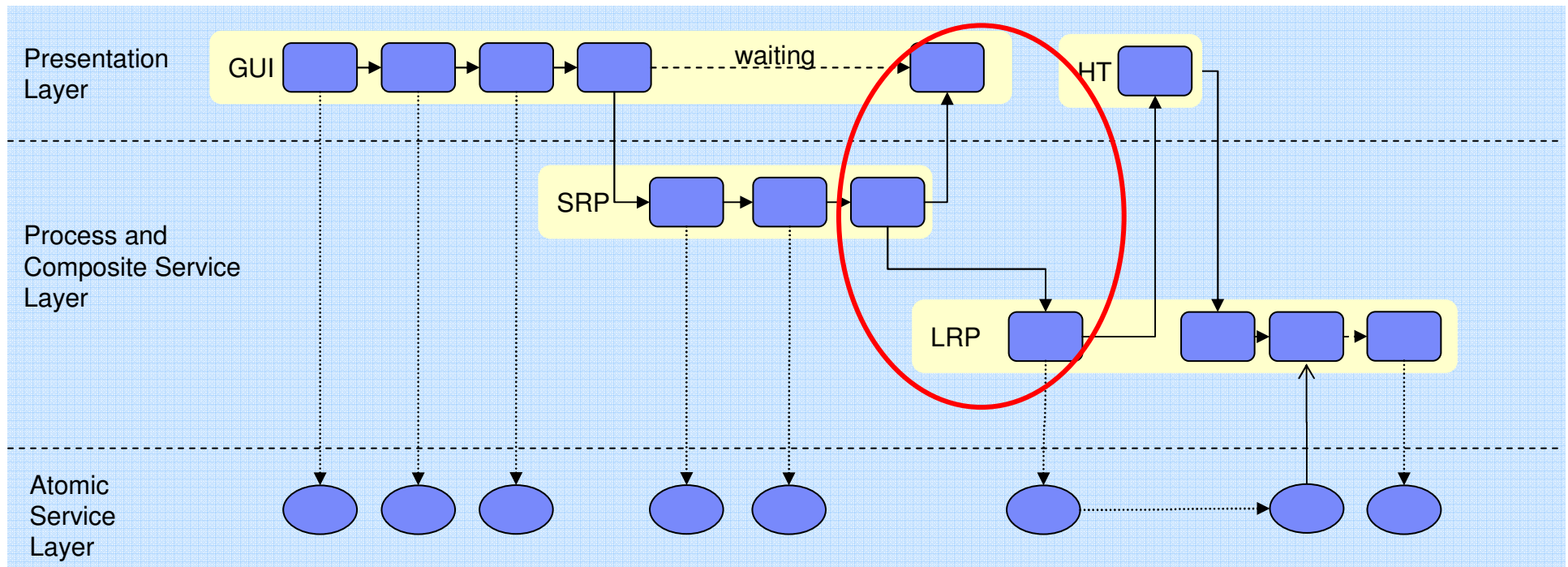


- What happens if the asynchronous request takes longer to complete than the requesters are prepared to wait?
- How would a graphical user interface manage late responses?



# Common pattern for complex services exposed to GUIs

*Triggering* long running work,  
with *acknowledgement* response

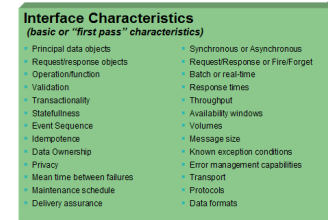


GUI Graphical User Interface    HT Human Task    SRP Short-running Process    LRP Long-running Process



# Exposing services for BPM

## *Transport/Protocol*



- Human Requester of a Service
  - UIs prefer synchronous interaction
- Automated Process Requesting a Service
  - Processes prefer asynchronous, especially for updates.
  - Message based exposure will be preferred
    - WebSphere MQ
    - JMS
    - WS-ReliableMessaging?

**Caution:** Exposing over an asynchronous transport is only half the story. E.g.

- Are the messages persistent?
- Is the interaction pattern long lived, or does it have early timeouts?



# Exposing services for BPM

## *Integrity/Transactionality*

### Interface Characteristics (basic or "first pass" characteristics)

- Principal data objects
- Request/response objects
- Operation/function
- Validation
- Transactionality
- Statefulness
- Event Sequence
- Idempotence
- Data Ownership
- Privacy
- Mean time between failures
- Maintenance schedule
- Delivery assurance
- Synchronous or Asynchronous
- Request/Response or Fire/Forget
- Batch or real-time
- Response times
- Throughput
- Availability windows
- Volumes
- Message size
- Known exception conditions
- Error management capabilities
- Transport
- Protocols
- Data formats

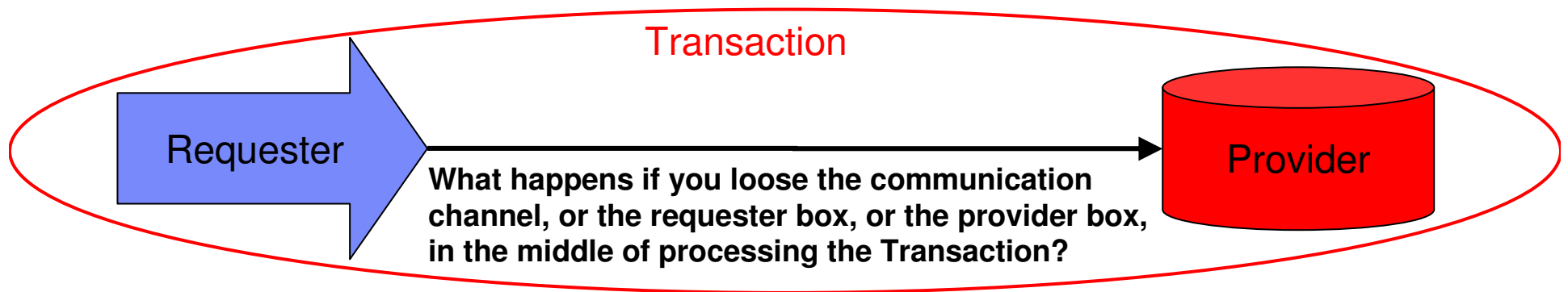




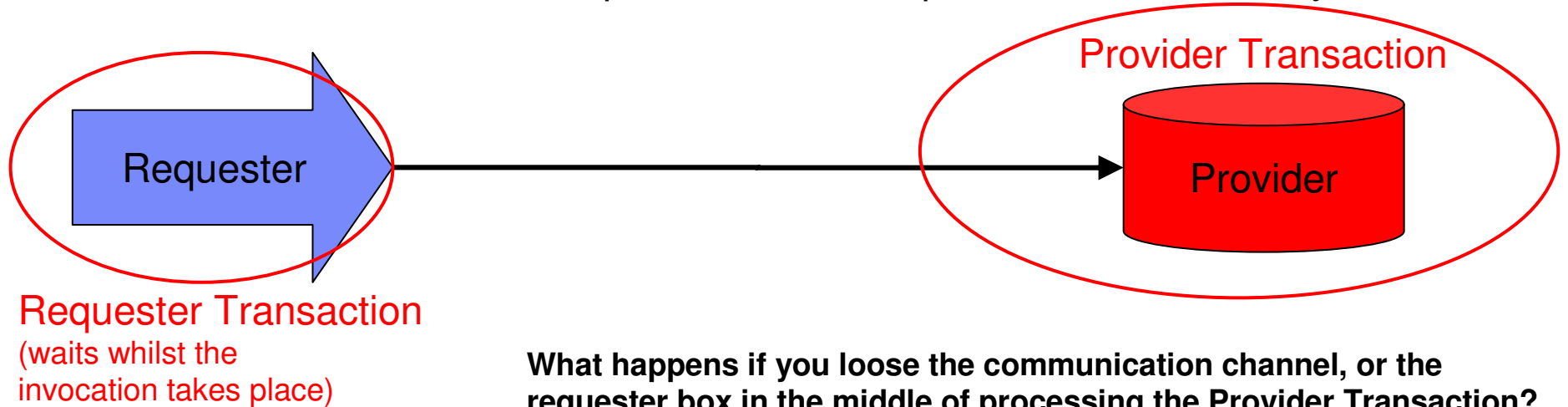


# Transactions within synchronous interactions

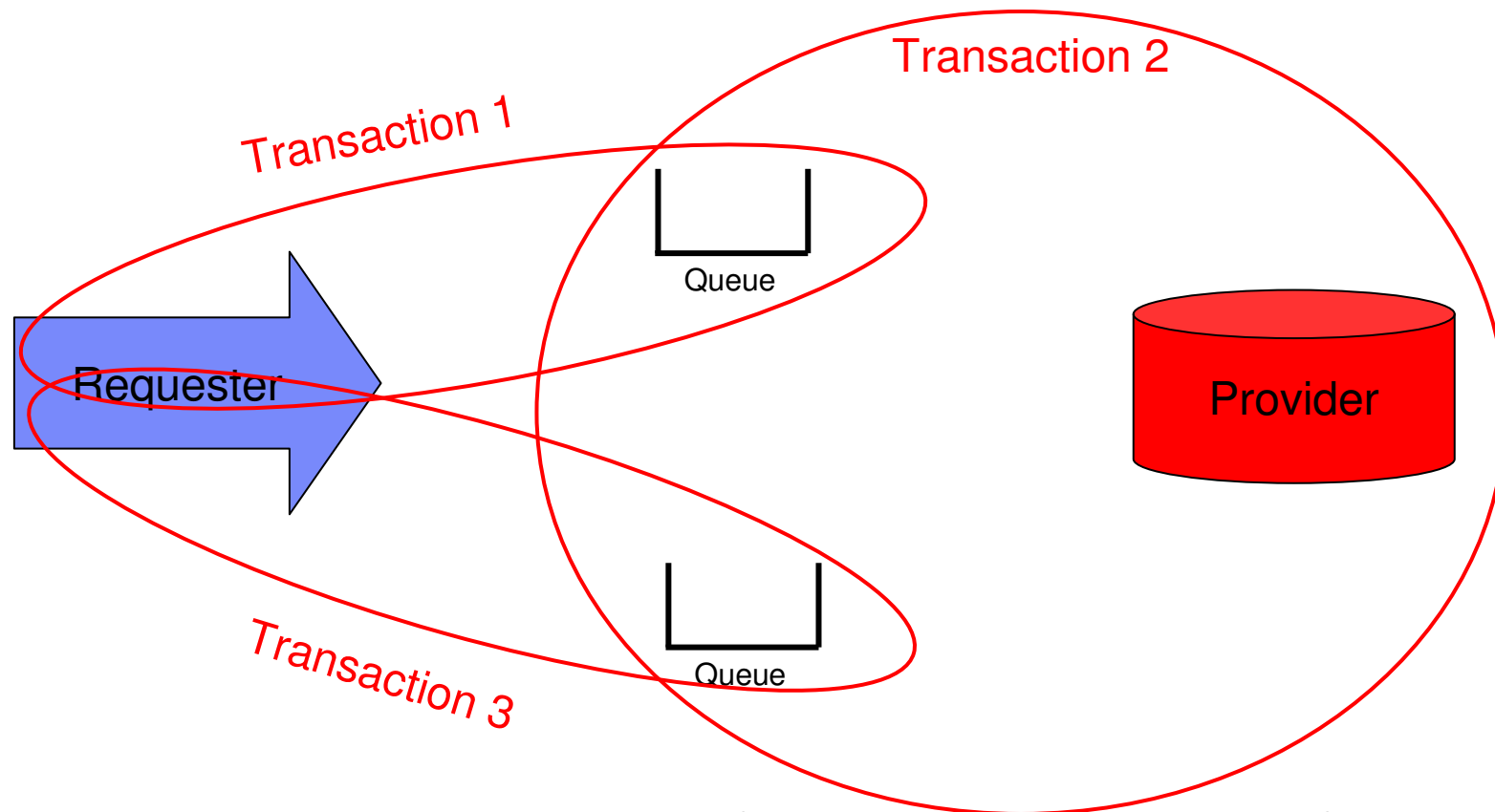
What is the minimum number of transactions required to invoke synchronously?



What about when the interaction protocol does not provide transactionality?



## Transactions within an asynchronous interaction



What is the minimum number of transactions required to perform a two way asynchronous interaction?





## Retry

- Is it safe to retry?
  - Is the request transactional?
  - Do duplicates matter?
  - Is the request idempotent?
- Who should take responsibility for performing the re-try
  - The user? The BPM engine? The service implementation?
- To what extent should re-try be alerted/logged?
  - If it ultimately works on retry, who needs to know about occasional errors on first try.
  - If it regularly fails, how do we know which alternate endpoint is failing, and how/who gets alerted?





# Idempotence

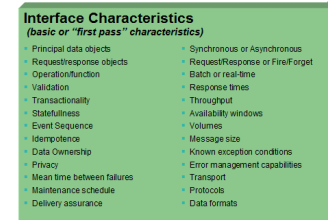
## *Definition and examples*

- An idempotent request means that no matter how many times the request is made, the result will be the same.
- Idempotence in low level transactions (“IT” transactions)
  - Read-only requests are nearly always idempotent.
  - Updates are idempotent unless they retain some level of change history (e.g. auditing).
  - Deletes are idempotent from a data perspective (you can’t delete something twice), but often not behaviourally as deletion of a non-existent entity often results in an error.
  - Inserts are not naturally idempotent. If they have a unique key constraint then they may be idempotent from a data perspective, but behaviourally are different on the second call due to the unique constraint error.
- Idempotence in business transactions
  - Nearly all business transactions involve inserts – the most difficult to make idempotent – even if they appear to be only performing a simple update or delete from a business perspective. This is either because they involve the creation of a new data item (a money transaction, purchase order etc.), or because they are audited in some way.
  - Most business transactions are therefore not naturally idempotent – unless you explicitly design it in.
  - The only exception is “reads”, but these are often costly in CPU so double submissions would be best avoided anyway.



# Exposing services for BPM

## *Integrity/Transactionality*



## Human Requester of a Service

### ■ Transactionality

- User can improvise upon in-doubt transactions.
- Transactionality often not available.
- In-doubt transaction more likely due to synchronous interaction over non-transactional medium.

### ■ Idempotence

- Must consider idempotence for business transactions since the user can retry.

### ■ Event Sequencing

- There tends to be an appreciable time gap between each of a user's requests, so event sequencing is less likely to be an issue. It can still happen though; for example across channels.

### ■ Validation

- Data must not all be present since can return to the human consumer for further validation.

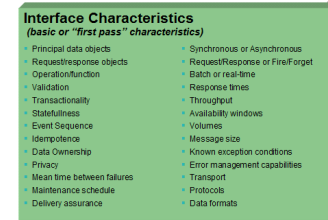
### ■ Locking

- Optimistic locking conflicts can be passed back to the human consumer.



# Exposing services for BPM

## *Integrity/Transactionality continued*



The diagram is a green rectangular box with the title "Interface Characteristics (basic or 'first pass' characteristics)" in bold. It contains two columns of bulleted items. The left column lists: Principal data objects, Request/Response objects, Operation/function, Validation, Transactionality, Statefulness, Event Sequence, Idempotence, Data Ownership, Privacy, Mean time between failures, Maintenance schedule, and Delivery assurance. The right column lists: Synchronous or Asynchronous, Request/Response or Fire/Forget, Batch or real-time, Response times, Throughput, Availability windows, Volumes, Message size, Known exception conditions, Error management capabilities, Transport, Protocols, and Data formats.

• Principal data objects	• Synchronous or Asynchronous
• Request/Response objects	• Request/Response or Fire/Forget
• Operation/function	• Batch or real-time
• Validation	• Response times
• Transactionality	• Throughput
• Statefulness	• Availability windows
• Event Sequence	• Volumes
• Idempotence	• Message size
• Data Ownership	• Known exception conditions
• Privacy	• Error management capabilities
• Mean time between failures	• Transport
• Maintenance schedule	• Protocols
• Delivery assurance	• Data formats

## Automated Process Requesting a Service

### ■ Transactionality

- Transactionality may will make coding of exception handling much simpler.
- In-doubt transactions can be more effectively avoided by waiting longer periods for a response.
- **However**, how often can transactional interaction be achieved with back end systems

### ■ Idempotence

- Coding for processes to manage the effects of duplicates create as a result of re-tries would be extremely complex. Services must either be transactional, idempotent, or use a store/forward pattern so re-try is not necessary.

### ■ Event Sequencing

- Exceptions resulting from out of sequence events are extremely difficult to code for. Event sequencing may therefore be critical.

### ■ Validation

- All data must be present. No original requester present from whom to request further data. Validation should perhaps therefore be stronger to ensure invalid data situations don't penetrate the service.

### ■ Locking

- Pessimistic locking may actually be preferred, though there are scalability issues with this.



# Exposing services for BPM

## *Security*

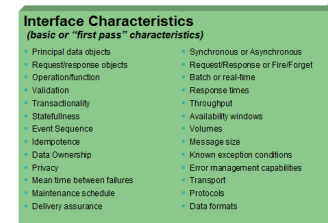
Interface Characteristics (basic or "first pass" characteristics)	
• Principal data objects	• Synchronous or Asynchronous
• Request/response objects	• Request/Response or Fire/Forget
• Operation/function	• Batch or real-time
• Validation	• Response times
• Transactionality	• Throughput
• Statefulness	• Availability windows
• Event Sequence	• Volumes
• Idempotence	• Message size
• Data Ownership	• Known exception conditions
• Privacy	• Error management capabilities
• Mean time between failures	• Transport
• Maintenance schedule	• Protocols
• Delivery assurance	• Data formats





# Exposing services for BPM

## Security



### ■ Human Requester of a Service

#### — Authorization

- Service often acts on the identity of the human requester.

#### — Data ownership and privacy

- The data is often owned by the requester.

### ■ Automated Process Requesting a Service

#### — Authorization

- When the process performs work, under who's credentials will it act? The originator? Would they have access to all the services called? How do we keep their session alive? Under a system id? If so where's the audit trail. What happens if a support user has to re-submit a request – what id should we use then?

#### — Data ownership and privacy

- What data can re-submitters of asynchronous events see? Does temporary storage of the request data outside of the system of record pose a data protection act breach?



# Exposing services for BPM

## *Availability*

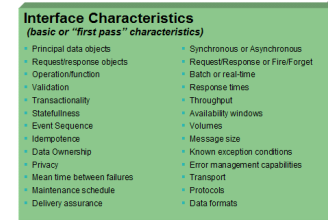
### Interface Characteristics (basic or "first pass" characteristics)

- Principal data objects
- Request/response objects
- Operation/function
- Validation
- Transactionality
- Statefulness
- Event Sequence
- Idempotence
- Data Ownership
- Privacy
- Mean time between failures
- Maintenance schedule
- Delivery assurance
- Synchronous or Asynchronous
- Request/Response or Fire/Forget
- Batch or real-time
- Response times
- Throughput
- Availability windows
- Volumes
- Message size
- Known exception conditions
- Error management capabilities
- Transport
- Protocols
- Data formats



# Exposing services for BPM

## Availability



### ■ Human Requester of a Service

- Availability windows
  - Humans often only work part of the time. Service availability should match.
- Outages
  - How do we tell human requesters about service outages?

### ■ Automated Process Requesting a Service

- Availability windows
  - Processes can work 24/7. Are the services they talk to available over same time periods?
- Outages
  - How do we tell processes to stop trying during outages?

Healthcheck and/or store/forward pattern required.

Caution: Don't assume that if an asynchronous transport is in place, that a sensible store/forward pattern is implemented



# Exposing services for BPM

## *Performance*

### Interface Characteristics (basic or "first pass" characteristics)

- Principal data objects
- Request/response objects
- Operation/function
- Validation
- Transactionality
- Statefulness
- Event Sequence
- Idempotence
- Data Ownership
- Privacy
- Mean time between failures
- Maintenance schedule
- Delivery assurance
- Synchronous or Asynchronous
- Request/Response or Fire/Forget
- Batch or real-time
- Response times
- Throughput
- Availability windows
- Volumes
- Message size
- Known exception conditions
- Error management capabilities
- Transport
- Protocols
- Data formats





# Throughput vs. Response time

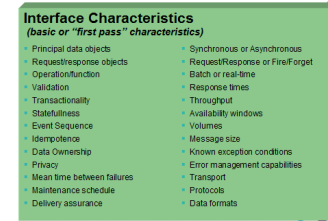
**Throughput** based applications are architected, designed, implemented, optimised and managed differently to **Response Time** based applications

- **Throughput based (Automated Process)**
  - Designed to maximize throughput, so typically asynchronous using messages
  - No user awaiting response
  - SLA Measured in events per time period (e.g. processes 100,000 messages per hour)
  - Pace of an individual process irrelevant
  - Performance tuned for high CPU utilisation
  - Thread count should be minimised.
  - Concurrency only related to performance tuning.
  - Errors must be handled/resolved
  - Servers handle retries
  - Typically stateful. Survives server restart.
- **Response time based (Human)**
  - Designed to minimize response time, so typically synchronous
  - Typically a user awaits the response
  - SLA Measured in time taken for response to arrive (e.g. user waits no more than 3 seconds)
  - Performance tuned to handle peaks
  - Queue depths should be minimised
  - Concurrency relates closely to the number of expected clients.
  - Errors can often be thrown away
  - Clients handle re-tries
  - Largely stateless. Transaction doesn't survive server restarts – although the session might.



# Exposing services for BPM

## Performance

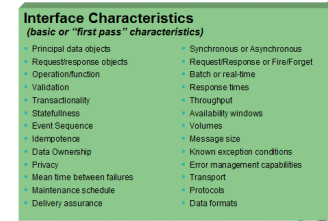


- Human Requester of a Service
  - Response times
    - Only faster response times can be tolerated.
  - Throughput/Volumes
    - Capacity has to be sufficient to handle peaks.
  - Batch or real-time
    - Humans work with real-time and not batch.
  - Message size
    - Messages should remain small to keep response times fast.



# Exposing services for BPM

## *Performance continued*



## ■ Automated Process Requesting a Service

### — Response times

- Slower synchronous response times can be accommodated since no user is waiting, but at the cost of threads.
- If an asynchronous interaction pattern is used, very long response times can be accommodated.

### — Throughput/Volumes

- Throughput can be managed more effectively since peaks and troughs present in user interaction can be levelled in throughput based work.

### — Batch or real-time

- Processes can take advantage of existing batch based processes to perform work.
- Will bulk processing volumes affect existing online user response times. Need to consider how throughput flow control is achieved.

### — Message size

- Larger messages can be handled due to acceptance of slower response times, and potential to perform flow control. However, usual concerns around memory etc. apply.



# Exposing services for BPM

## *Error and exception management*

### Interface Characteristics (basic or "first pass" characteristics)


- Principal data objects
- Request/response objects
- Operation/function
- Validation
- Transactionality
- Statefulness
- Event Sequence
- Idempotence
- Data Ownership
- Privacy
- Mean time between failures
- Maintenance schedule
- Delivery assurance
- Synchronous or Asynchronous
- Request/Response or Fire/Forget
- Batch or real-time
- Response times
- Throughput
- Availability windows
- Volumes
- Message size
- Known exception conditions
- Error management capabilities
- Transport
- Protocols
- Data formats





# Exposing services for BPM

## *Error and exception management*



Interface Characteristics (basic or "first pass" characteristics)	
• Principal data objects	• Synchronous or Asynchronous
• Request/Response objects	• Request/Response or Fire/Forget
• Operation/function	• Batch or real-time
• Validation	• Response times
• Transactionality	• Throughput
• Statefulness	• Availability windows
• Event Sequence	• Volumes
• Idempotence	• Message size
• Data Ownership	• Known exception conditions
• Privacy	• Error management capabilities
• Mean time between failures	• Transport
• Maintenance schedule	• Protocols
• Delivery assurance	• Data formats

### ■ Human Requester of a Service

- Exception paths
  - Coding for exceptions relatively simple users.
- Online error management
  - Exceptions can be returned to requesters for handling.
- Diagnosis
  - Visibility is not as important since services respond quickly.

### ■ Automated Process Requesting a Service

- Exception paths
  - Coding for exceptions must be much more explicit than would be necessary with users.
- Offline error management
  - Need to be able to manage exceptions offline. i.e. the original source of data and/or business domain expert may no longer be available.
- Diagnosis
  - Due to longer running nature of asynchronous interactions a greater degree of in-flight visibility will be required for diagnosis of “where’s my message” scenarios.



## Why do we need two services when they provide the same functionality?



- If services used by GUIs are different from those used by processes/applications, then aren't we creating requester specific services? What about re-use!
- Specific to a **category of requester**, not specific to a requester.
  - We should always avoid creating services specific to a single requester. However, services specific to a category of requesters may be necessary.
  - Two likely requester categories might be **GUIs**, and **processes**.
  - Differences relating to functional requirements may be easier to tease out in the analysis. Differences in non-functional requirements might not be clear unless sufficient analysis is done into the interface characteristics.





## Agenda

- Brief introduction to Business Process Management (BPM)
- Where have companies got to on their SOA journey?
- Is BPM a different type of service consumer?
- How do we technically characterise services?
- How would we expose services differently for BPM requesters?
- Resources and assistance





# **Getting started:** *Resources and assistance*

**Process Discovery Workshop**

**Quick Win Pilot**

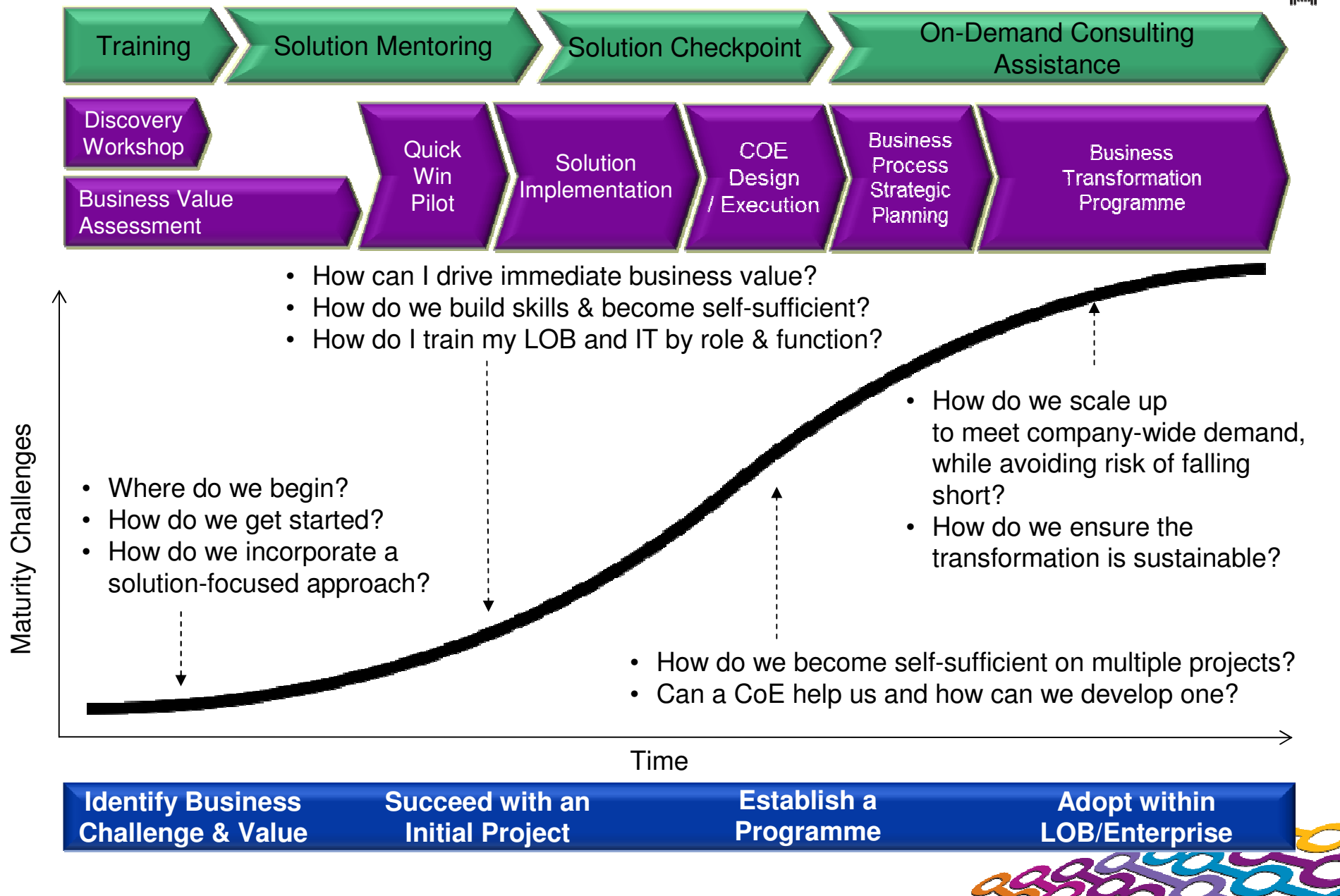
**Application Infrastructure Assessment**

***IBM Redbook: BPM Adoption***

***developerWorks: BPM Journal***



# Establishing & evolving excellence in service delivery





## Free on-site “Process Discovery Workshop”

- Joint collaboration with business and IT stakeholders to develop well-defined approach to improve a specific business challenge in a targeted business area that could use the features of IBM BPM
- Work together with IBM or our Business Partners to
  - review the end-to-end business challenge.
  - identify and assess potential risks.
  - assess high-level implementation effort and solution phasing.
  - define baseline solution architecture and implementation approach.
  - assess realistic business impacts of proposed solution, evaluate best place to start.
  - clarify business case justification for proposed solution, timings & activities.





# Application Infrastructure Assessment

- Structured workshops to assess an organisation's IT maturity across a broad range of competencies.
- Examples areas include:
  - Integration
  - Service Oriented Architecture
  - Business Process Management
- Provides recommendations on actions to perform to reach target architecture.





## Quick Win Pilot

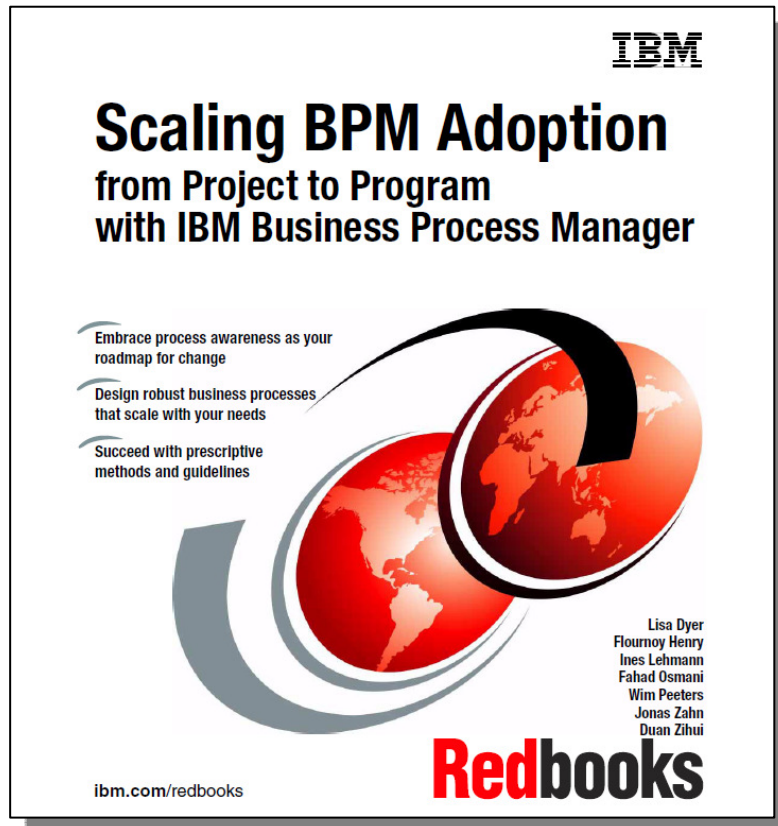
- Eight-to-ten week engagement with IBM or our Business Partner using a proven, iterative approach that collaboratively builds a deployable solution.
- Each specific project is unique, but IBM has structured this offering to ensure consistent delivery of the following outcomes:
  - Define and prioritise your future-state business process solution
  - Define a quick-win scope that delivers a high-value subset of the future-state solution
  - Gain process owner and stakeholder acceptance for the quick-win solution scope
  - Build and deploy the quick-win solution within your pilot environment
  - Establish an internal proof-point for the value of iterative process development within your business
  - Provide a jump start for the enablement of your internal project team
- During/following first Quick Win Pilot, IBM can help you scale quickly and move into programmatic BPM adoption.







# IBM Redbook: BPM Adoption



- BPM method
  - Process discovery
  - Implementing a BPM project
  - Deploying/managing a process
  - Business process governance
  - BPM common practice
  - Scaling from project to program
- ...and more!

<http://www.redbooks.ibm.com/abstracts/sg247973.html>



# developerWorks: BPM Journal



IBM

English

Sign in (or register)

developerWorks

Technical topics

Evaluation software

Community

Events

Search developerWorks

WebSphere

New to WebSphere

Products

Downloads

Technical library

Community & forums

Events

Related links

- How to buy
- News
- Training
- Services
- Support

developerWorks > Technical topics > WebSphere > Technical library >

## IBM Business Process Management Journal

Issue 4.1: February 15, 2012



### IBM Business Process Management Journal

Current issue

Other formats

Archive

About the editor

**A word from [the editor](#):**

The buzz words for this edition are *integration*, *agility*, *business rules*, and *mobile*! The quarterly IBM BPM Journal is jam-packed with content from some of IBM's leading experts on business process management and decision management. These are the folks on the front lines, developing and deploying IBM's BPM solutions in the real world.

This month's contributors will share with you their expertise and hands-on experience with integrating IBM BPM products with CICS, Host On-Demand, WebSphere Portal and IBM Case Manager; introduce you to a new pattern for business rules to simplify and accelerate your rules projects; tell you how to avoid common pitfalls in business object model development, recommend an agile approach to BPM that has been tested successfully with many BPM clients; demonstrate how you can create a mobile BPM app that enables users to access, claim, and complete tasks remotely, and lots more.

Start reading now!

Top story

#### Introducing the Generic Ruleset Signature pattern for WebSphere Operational Decision Management V7.5

by Nigel Crowther, Kallol Ghosh, and Jonathon Carr

This two part article series introduces the Generic Ruleset Signature pattern. This pattern can

Update My dW interests

(Log in | What's this?)

IBM Business Process Manager



Process and profit unite, IBM Business Process Manager helps companies boost productivity

→ Watch video (00:03:26)

IBM Blueworks Live



Document & Run Processes in the Cloud.



# This presentation uses content from the following:

- Interface Characteristics:  
*Capturing Integration Complexity for BPM and SOA*
  - [http://www.ibm.com/developerworks/websphere/techjournal/1112\\_clark/1112\\_clark.html](http://www.ibm.com/developerworks/websphere/techjournal/1112_clark/1112_clark.html)
- SOA maturity in relation to solution design
  - [http://www.ibm.com/developerworks/websphere/library/techarticles/0904\\_clark/0904\\_clark.html](http://www.ibm.com/developerworks/websphere/library/techarticles/0904_clark/0904_clark.html)
- Process implementation types:  
*Patterns based design for process-based solutions*
  - [http://www.ibm.com/developerworks/websphere/library/techarticles/1004\\_clark/1004\\_clark.html](http://www.ibm.com/developerworks/websphere/library/techarticles/1004_clark/1004_clark.html)
- “Process-oriented modeling for SOA”
  - <http://www.ibm.com/developerworks/architecture/library/ar-procmod1>
- The Enterprise Service Bus, re-examined  
*Updating concepts and terminology for an evolved technology*
  - [http://www.ibm.com/developerworks/websphere/techjournal/1105\\_flurry/1105\\_flurry.html](http://www.ibm.com/developerworks/websphere/techjournal/1105_flurry/1105_flurry.html)





## Other useful references

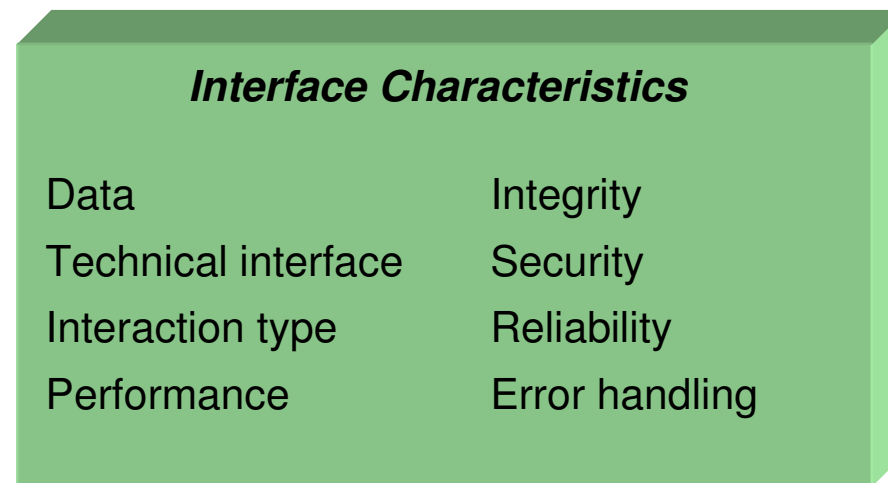
- Scaling BPM Adoption from Project to Program with IBM BPM
  - <http://www.redbooks.ibm.com/abstracts/sg247973.html>
- Five Guidelines to Better Process Modeling
  - <http://wiki.lombardi.com/display/commwiki/Five+Guidelines+to+Better+Process+Modeling>





## Key Messages

- Integration maturity will continue to be critical to IT scalability and agility.
- Interface characteristics provide a structured approach to assessing integration complexity and maturity.



*“Interface Characteristics: Capturing Integration Complexity for BPM and SOA”*

[http://www.ibm.com/developerworks/websphere/techjournal/1112\\_clark/1112\\_clark.html](http://www.ibm.com/developerworks/websphere/techjournal/1112_clark/1112_clark.html)



