Service-Oriented Architecture: One Size fits Nobody

Christoph Bussler

Keynote, ICEIS Conference
June 13th, 2007

bea
Think liquid.™

Agenda

- The Promise of SOA and One SOA Approach
- Common Architecture Patterns
- Abstractions
- SOA vs. SOI
- Conclusions

The Promise of Service-oriented Architecture (SOA)

- Flexibility
  - Adjust to business need of the day
  - E. g., availability of new product starting next Monday

The Promise of Service-oriented Architecture (SOA)

- Flexibility
  - Adjust to business need of the day
  - E. g., availability of new product starting next Monday
- Agility
  - Fast change
  - E. g., product promotion change for one day (tomorrow) only

The Promise of Service-oriented Architecture (SOA)

- Flexibility
  - Adjust to business need of the day
  - E. g., availability of new product starting next Monday
- Agility
  - Fast change
  - E. g., product promotion change for one day (tomorrow) only
- Reuse
  - Reuse existing functionality
  - E. g., customer record retrieval needed by helpdesk as well as product ordering system

The Promise of Service-oriented Architecture (SOA)

- Flexibility
  - Adjust to business need of the day
  - E. g., availability of new product starting next Monday
- Agility
  - Fast change
  - E. g., product promotion change for one day (tomorrow) only
- Reuse
  - Reuse existing functionality
  - E. g., customer record retrieval needed by helpdesk as well as product ordering system
- Integration / Composition
  - Packaged application systems, (web) services available over the Internet, user interactions
  - E. g., retrieval of order status by customer

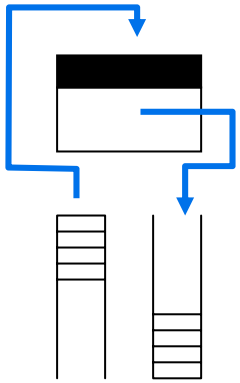
The Promise of Service-oriented Architecture (SOA)

- Flexibility
  - Adjust to business need of the day
  - E. g., availability of new product starting next Monday
- Agility
  - Fast change
  - E. g., product promotion change for one day (tomorrow) only
- Reuse
  - Reuse existing functionality
  - E. g., customer record retrieval needed by helpdesk as well as product ordering system
- Integration / Composition
  - Packaged application systems, (web) services available over the Internet, user interactions
  - E. g., retrieval of order status by customer
- Dynamic and automatic discovery
  - Finding existing services (automatically) inside and outside organizations
  - E. g., list all web services that check if a given address is a real address

One SOA Approach

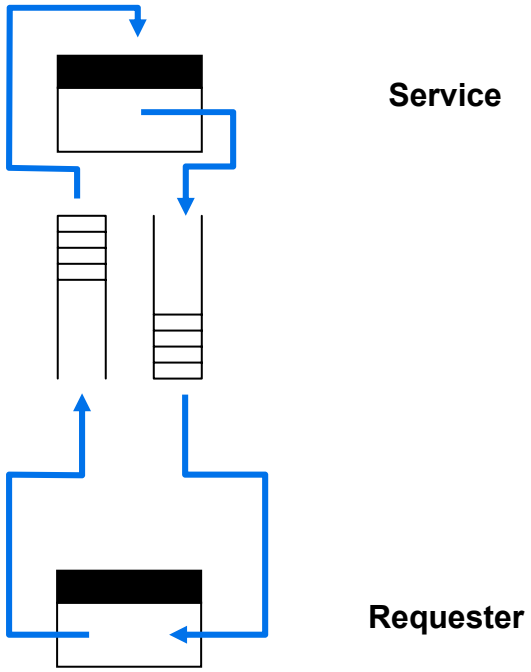
One SOA Approach

- Put all functionality on a queue
  - IN queue for requesting service
  - OUT queue for providing service results



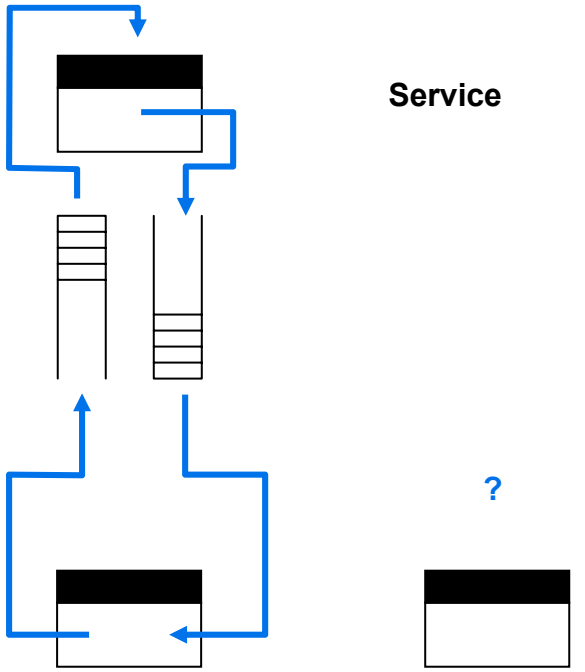
One SOA Approach

- Put all functionality on a queue
  - IN queue for requesting service
  - OUT queue for providing service results
- Necessary invocations over queuing interface by sending and receiving messages
  - Service requester sends and receives messages
  - Service provider receives and sends messages



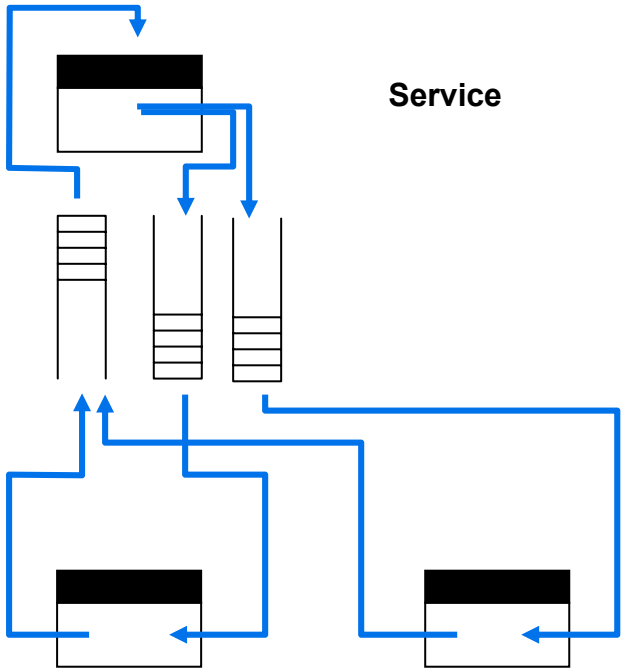
One SOA Approach

- Put all functionality on a queue
  - IN queue for requesting service
  - OUT queue for providing service results
- Necessary invocations over queuing interface by sending and receiving messages
  - Service requester sends and receives messages
  - Service provider receives and sends messages



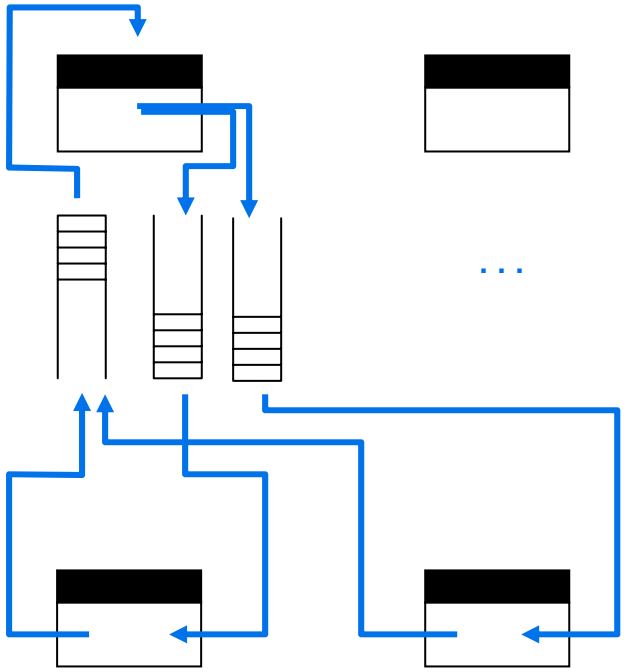
One SOA Approach

- Put all functionality on a queue
  - IN queue for requesting service
  - OUT queue for providing service results
- Necessary invocations over queuing interface by sending and receiving messages
  - Service requester sends and receives messages
  - Service provider receives and sends messages



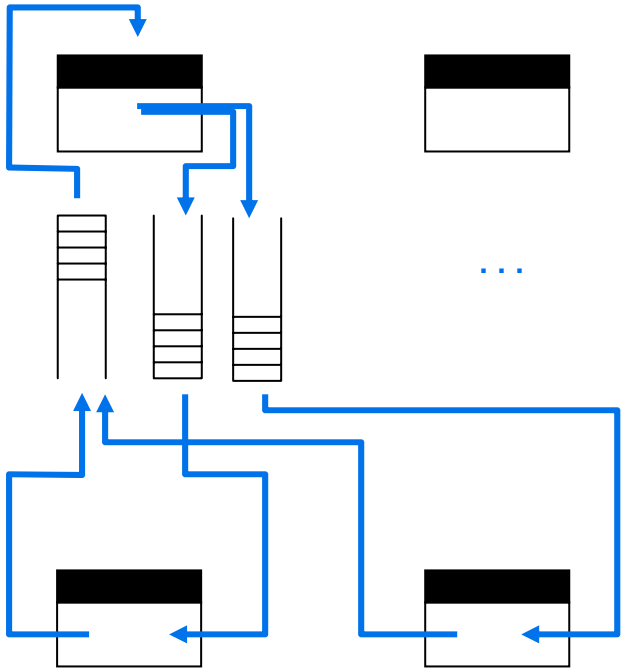
One SOA Approach

- Put all functionality on a queue
  - IN queue for requesting service
  - OUT queue for providing service results
- Necessary invocations over queuing interface by sending and receiving messages
  - Service requester sends and receives messages
  - Service provider receives and sends messages



One SOA Approach

- Put all functionality on a queue
  - IN queue for requesting service
  - OUT queue for providing service results
- Necessary invocations over queuing interface by sending and receiving messages
  - Service requester sends and receives messages
  - Service provider receives and sends messages
- “Message-based” approach
  - All interactions between service requester and service provider take place over messages



One SOA Approach

- Wait a minute …

One SOA Approach

- Heterogeneity?
  - Data type system mismatches?

One SOA Approach

- Heterogeneity?
  - Data type system mismatches?
  - Data syntax mismatches?

One SOA Approach

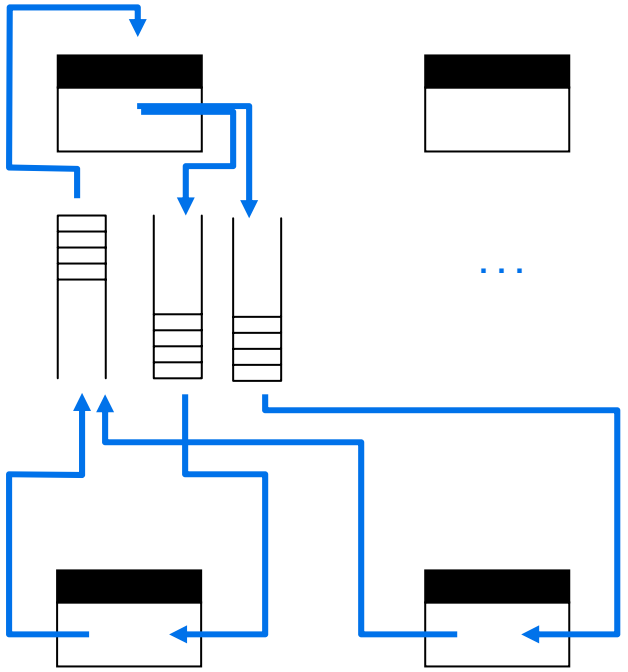
- Heterogeneity?
  - Data type system mismatches?
  - Data syntax mismatches?
  - Data semantics mismatches?

One SOA Approach

- Heterogeneity?
  - Data type system mismatches?
  - Data syntax mismatches?
  - Data semantics mismatches?
  - Let's add transformation to the approach

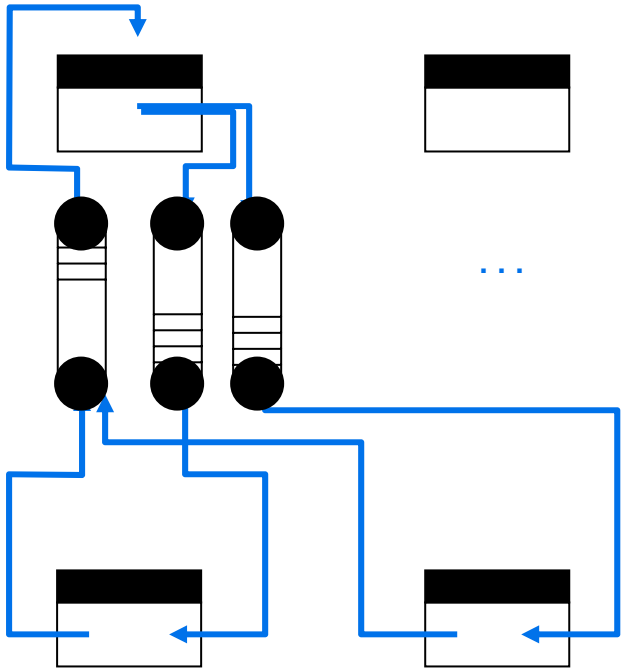
One SOA Approach

- Transformation
  - Syntax
  - Semantics
    - Vocabulary



One SOA Approach

- Transformation
  - Syntax
  - Semantics
    - Vocabulary



Example of Service Composition

- A service
  - That calls another service
  - That in turn composes two other services
  - Using the SOA approach based on queues

Example of Service Composition

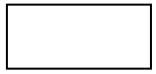
- Preface: Symbols

Example of Service Composition

- Interface

Example of Service Composition

- Implementation

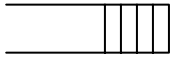


Example of Service Composition

- Service

Example of Service Composition

- Queue (ESB)



Example of Service Composition

- Transformation

Example of Service Composition

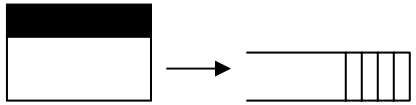
- State

Example: Story

- Story: Once upon a time …

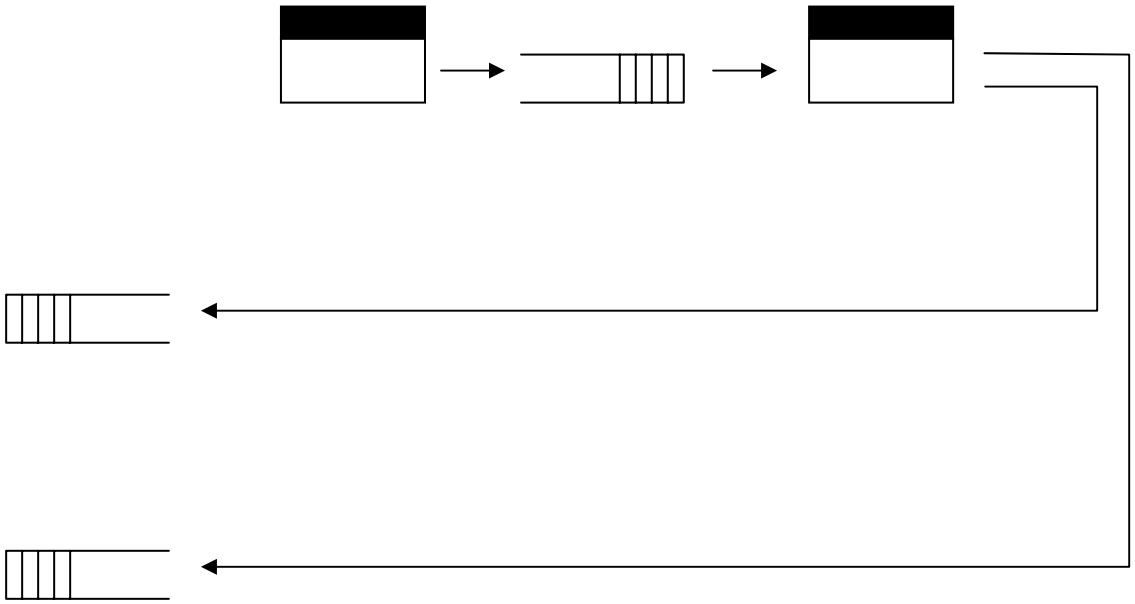
Example: Story

Example: Story

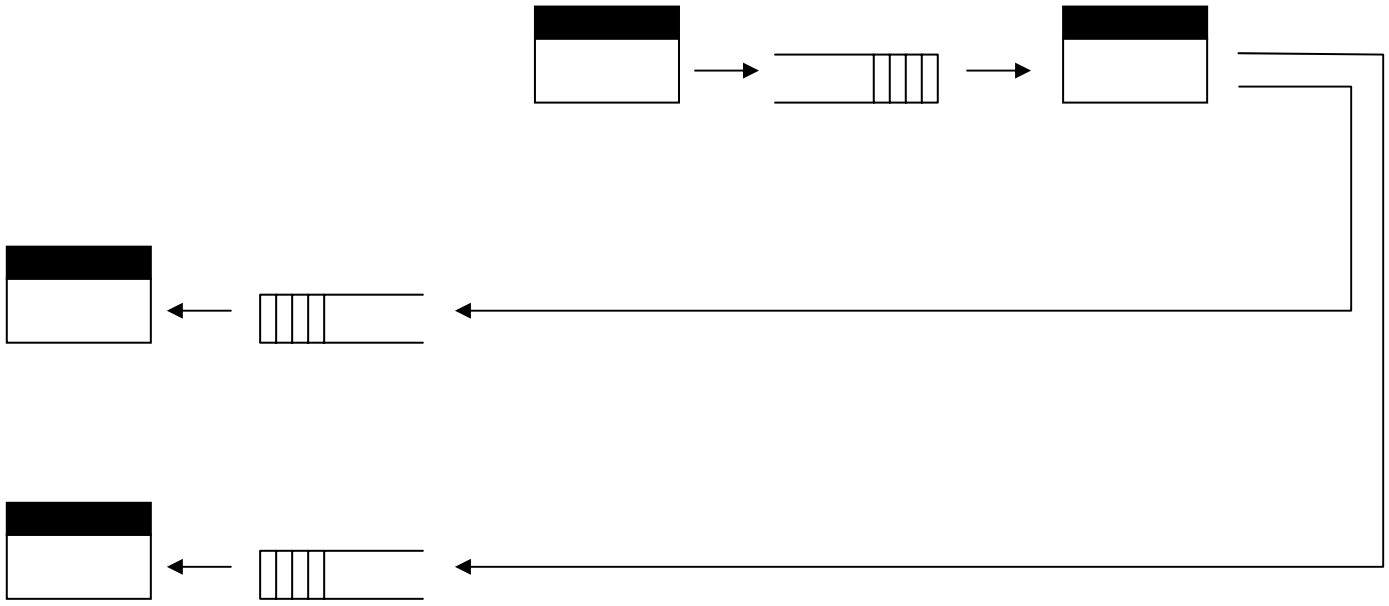


Example: Story

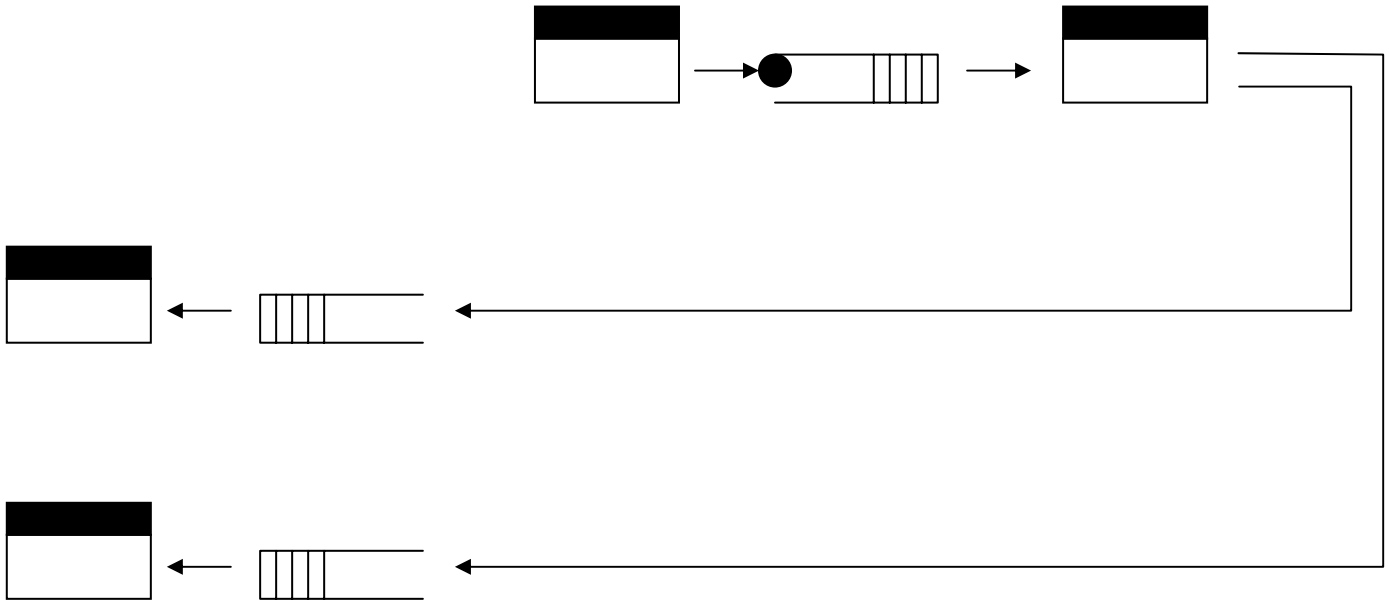
Example: Story



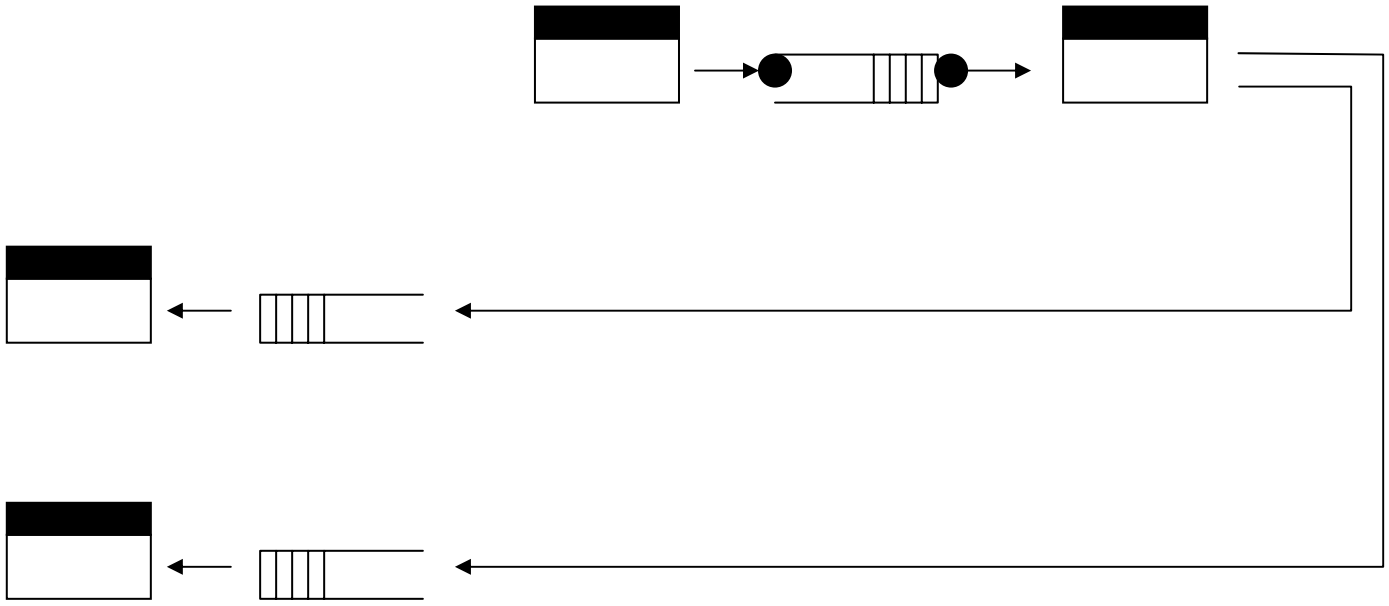
Example: Story



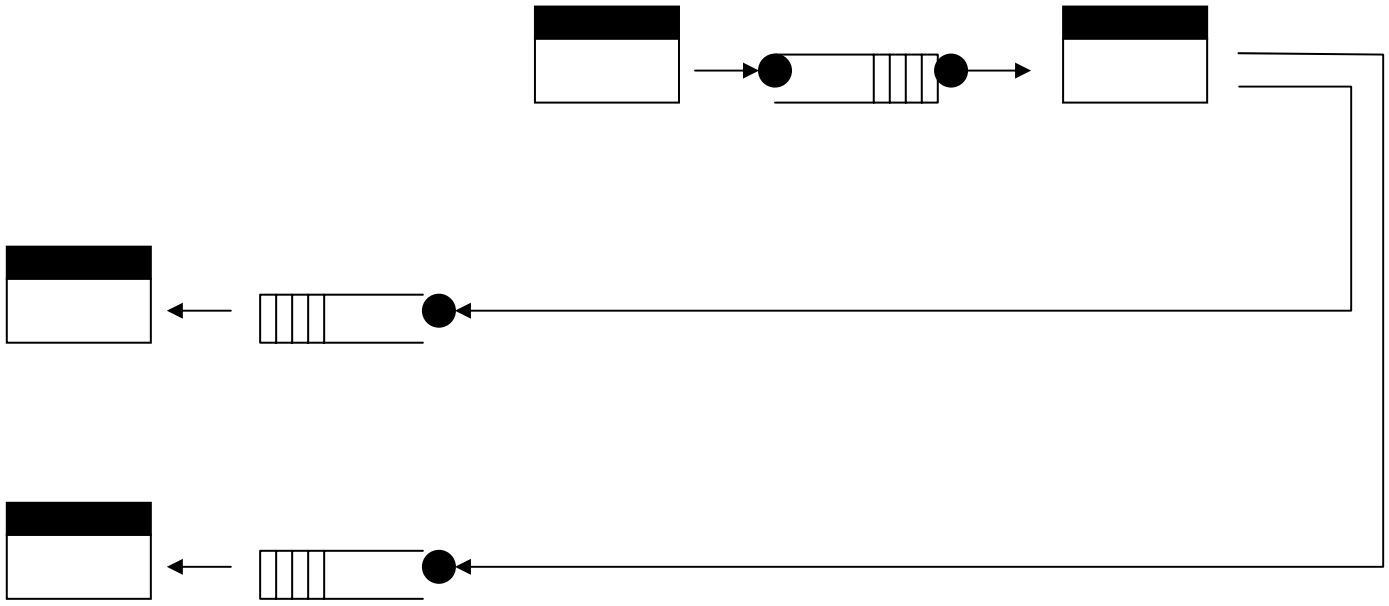
Example: Story



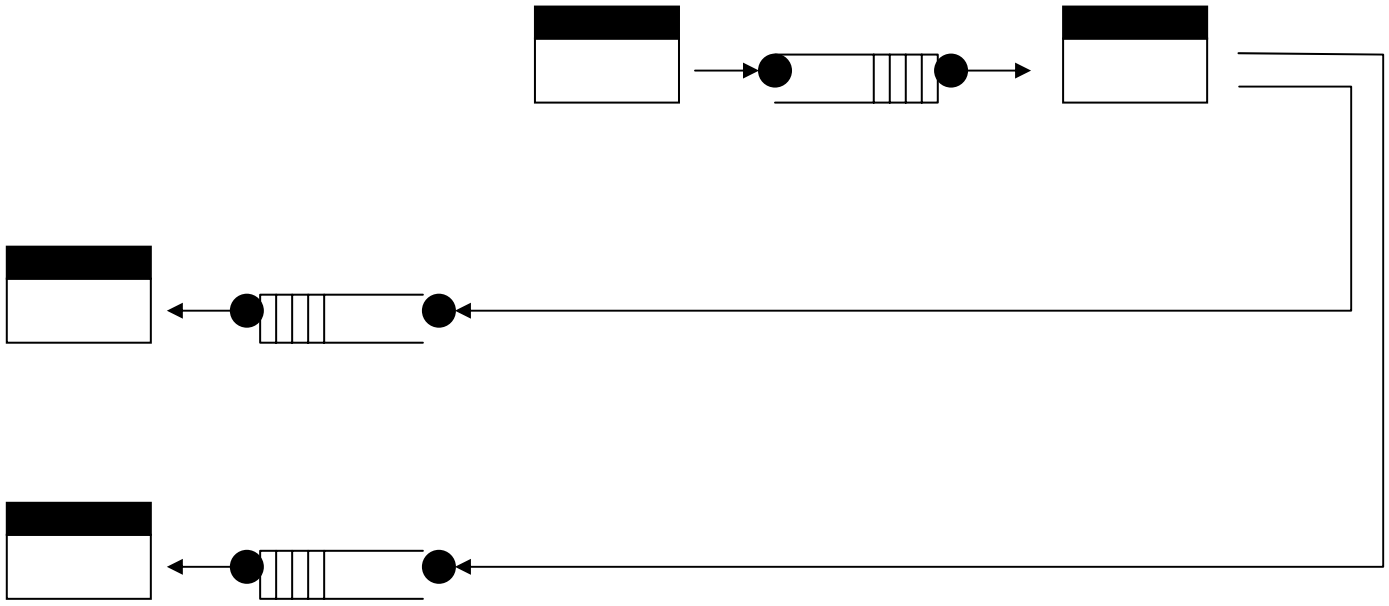
Example: Story



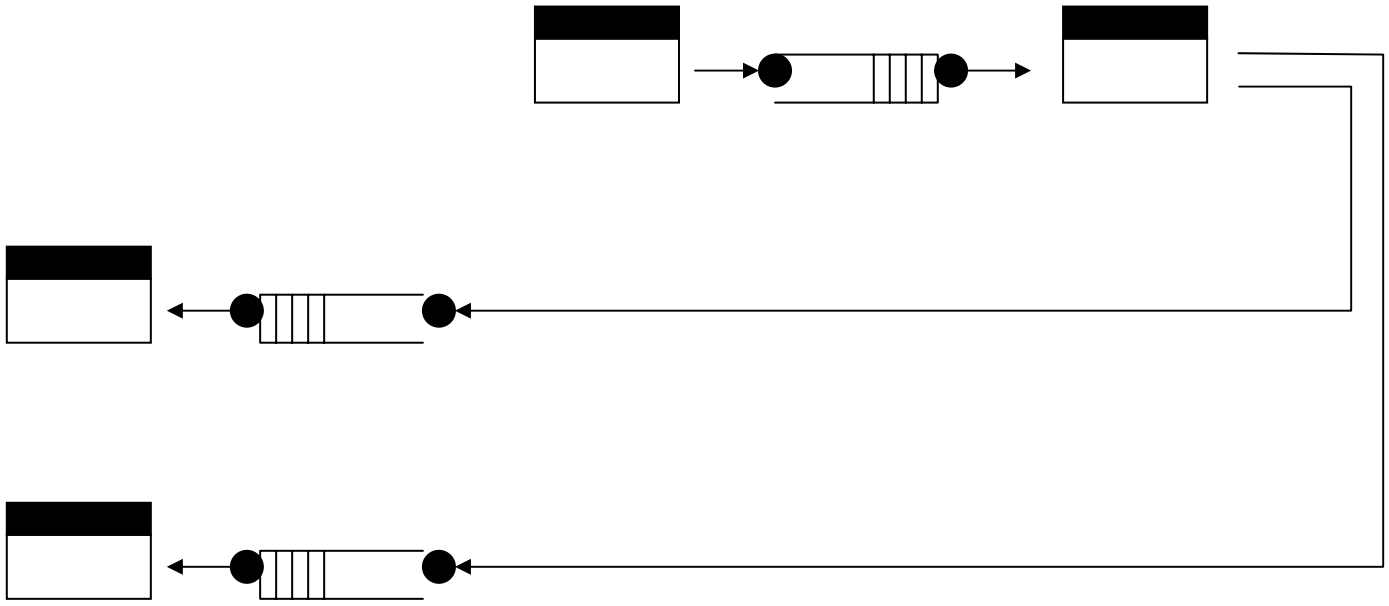
Example: Story



Example: Story

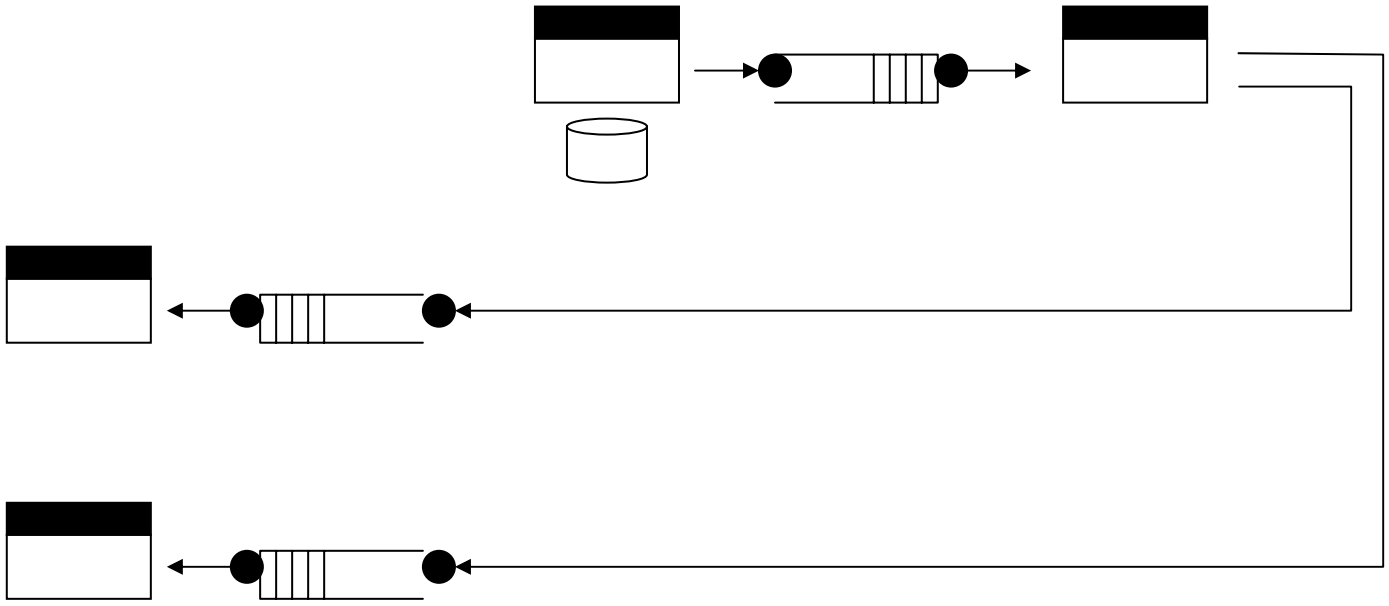


Example: Story

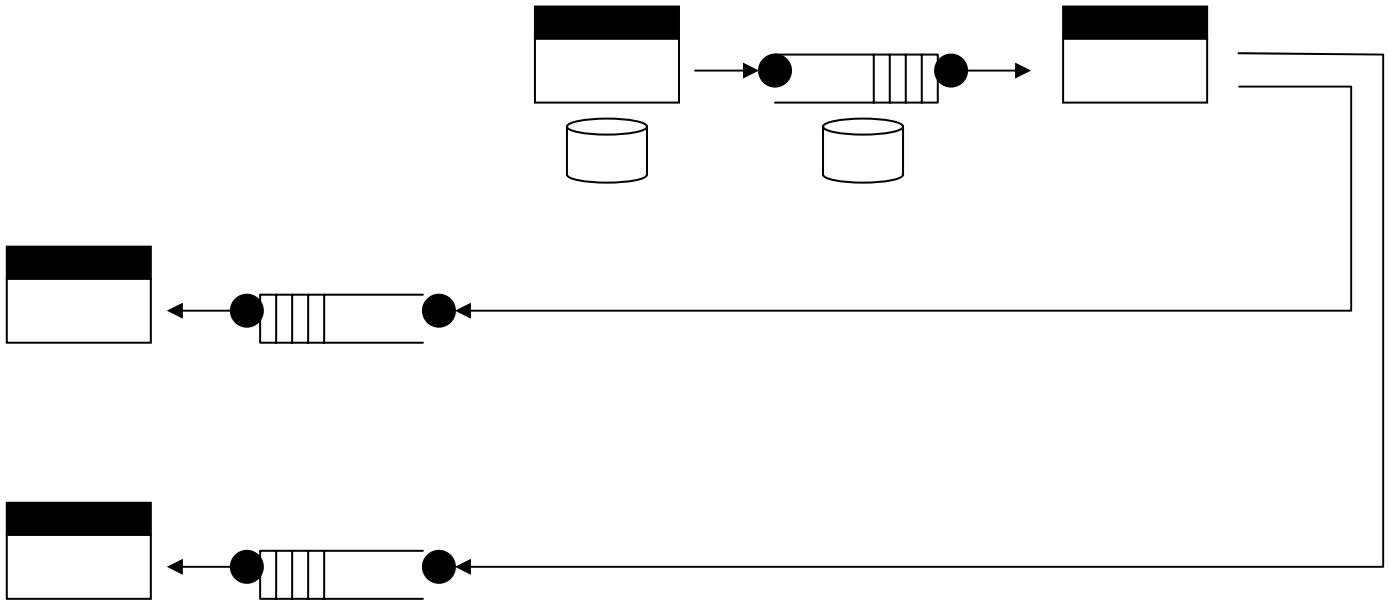


And the same for the way back (omitted)

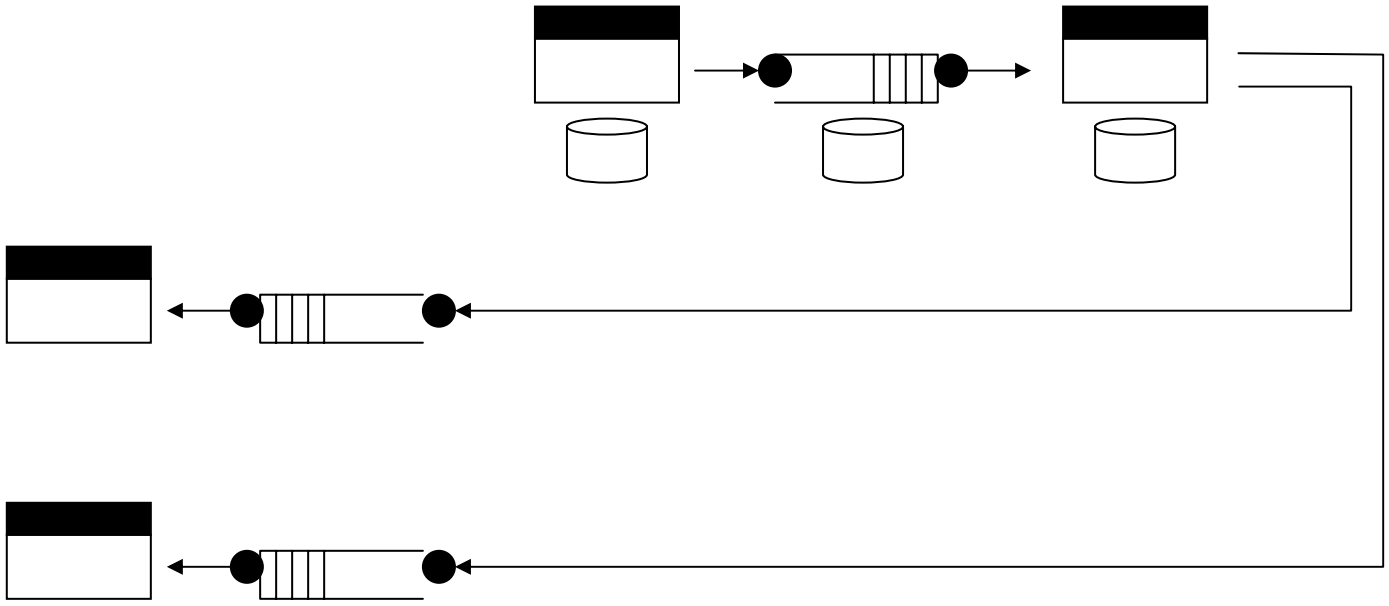
Example: Story



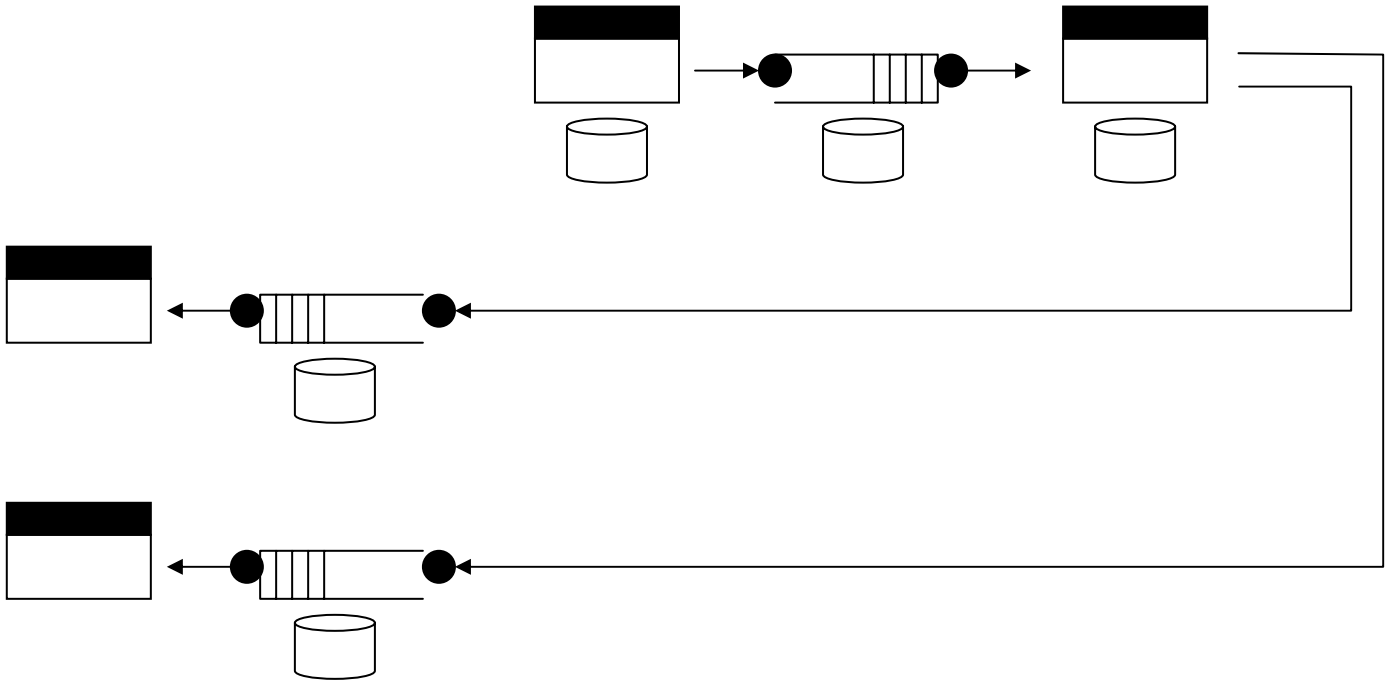
Example: Story



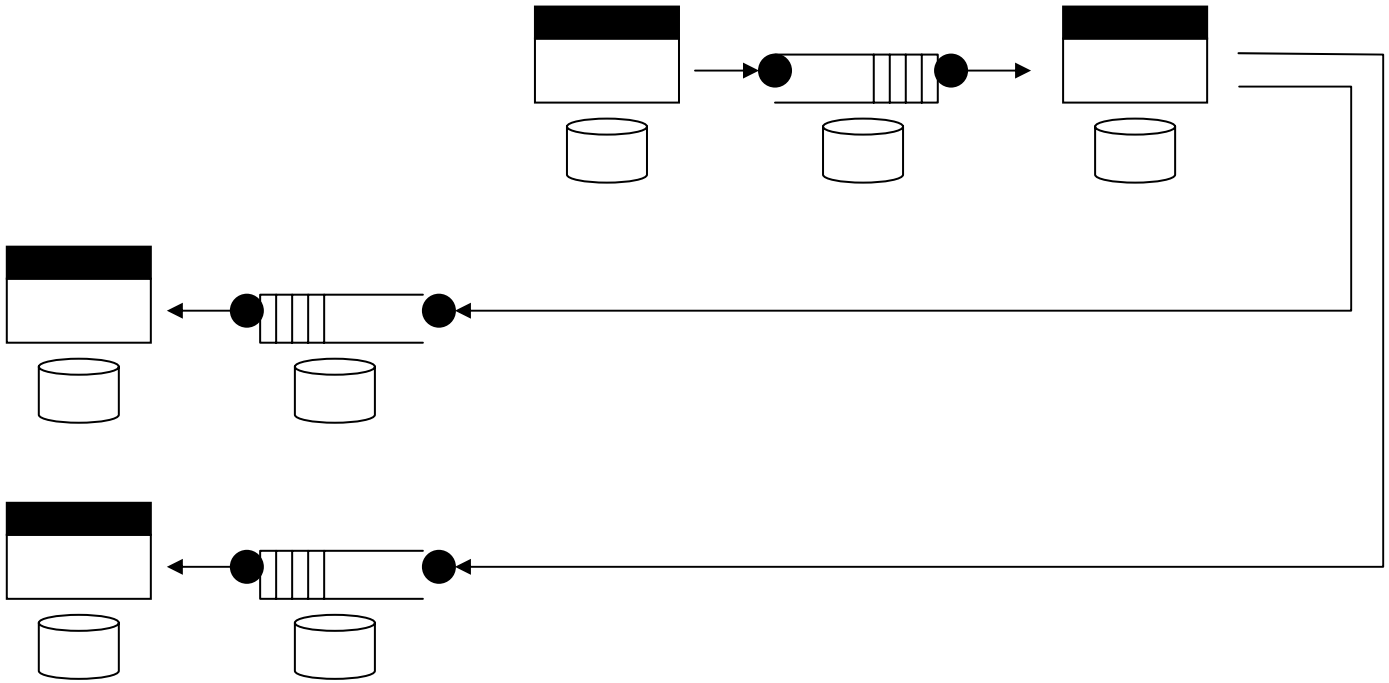
Example: Story



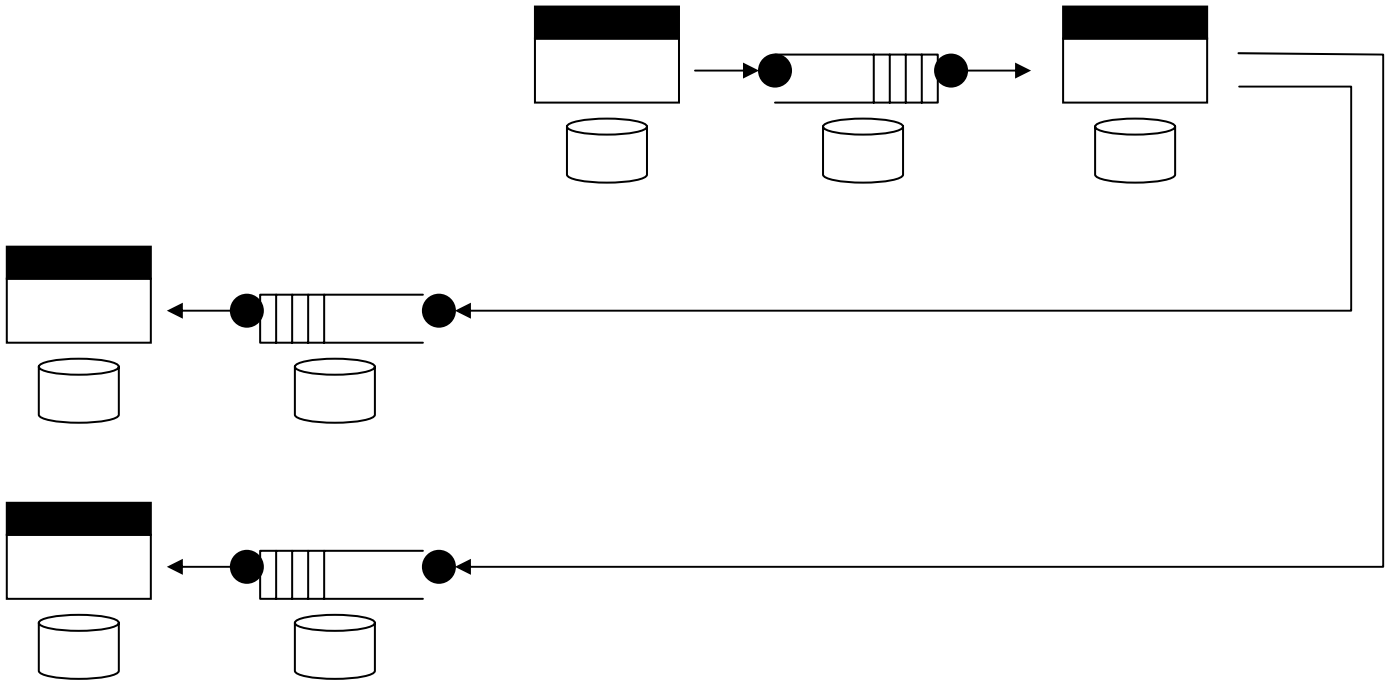
Example: Story



Example: Story

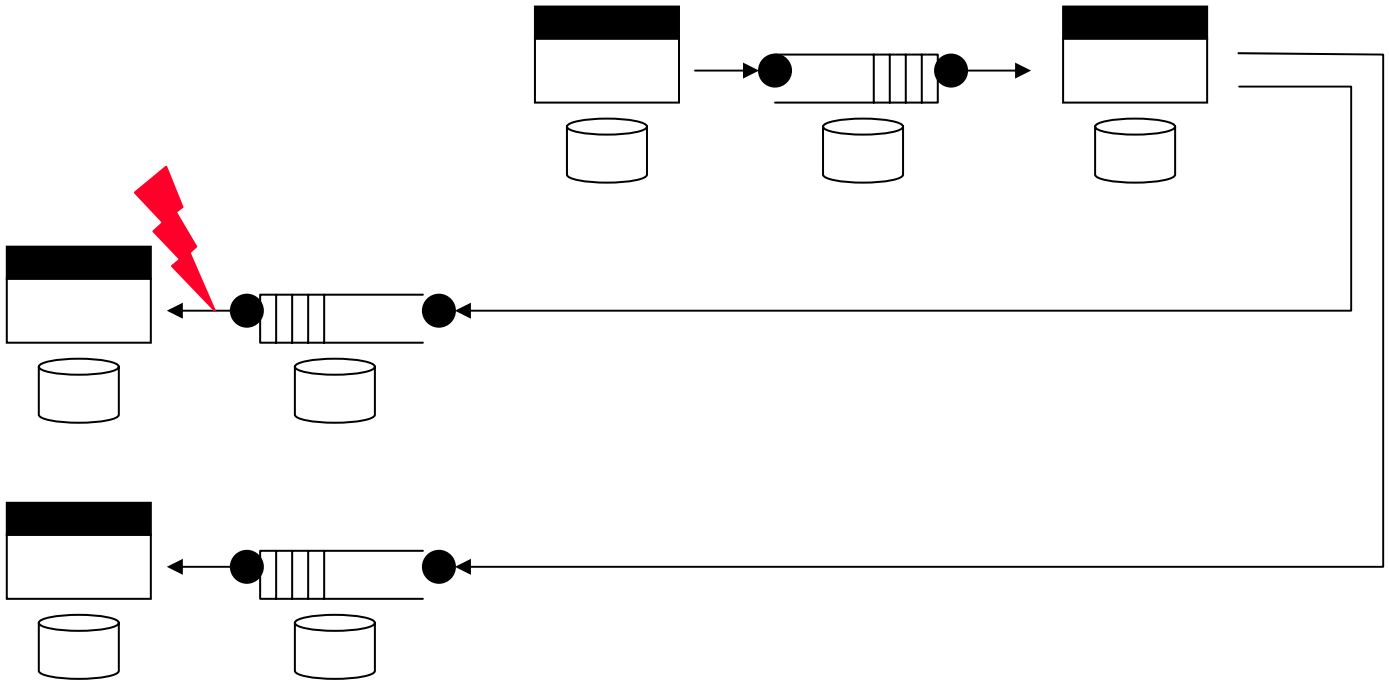


Example: Story

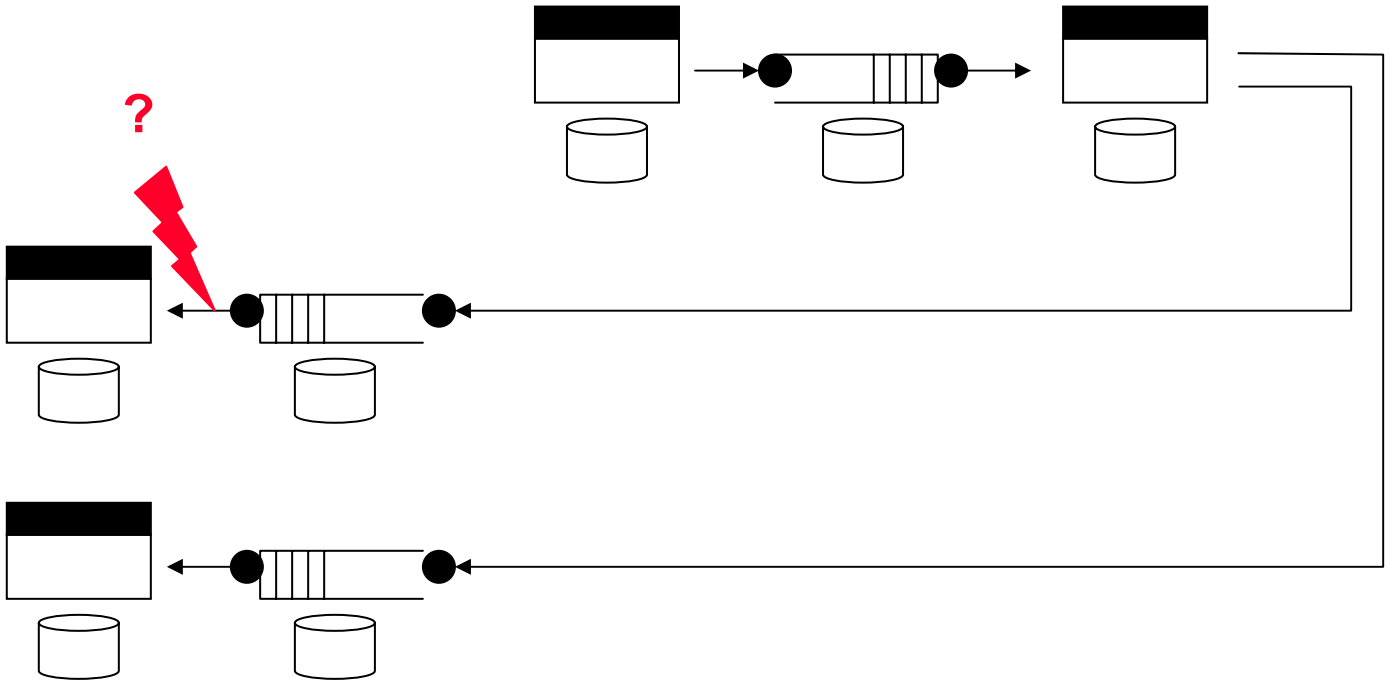


And the same for the way back (omitted)

Example: Story



Example: Story



Example of Service Composition

- Some potentially nagging questions for you ☺

Example of Service Composition

- Some potentially nagging questions for you ☺
  - How is state of composition managed?

Example of Service Composition

- Some potentially nagging questions for you ☺
  - How is state of composition managed?
  - What happens in case of failures? Assume (a) transactional queuing systems (b) non-transactional, but persistent queuing system (c) non-transactional and non-persistent queuing system?

Example of Service Composition

- Some potentially nagging questions for you ☺
  - How is state of composition managed?
  - What happens in case of failures? Assume (a) transactional queuing systems (b) non-transactional, but persistent queuing system (c) non-transactional and non-persistent queuing system?
  - How does recovery from those failures work?

Example of Service Composition

- Some potentially nagging questions for you ☺
  - How is state of composition managed?
  - What happens in case of failures? Assume (a) transactional queuing systems (b) non-transactional, but persistent queuing system (c) non-transactional and non-persistent queuing system?
  - How does recovery from those failures work?
  - How does this approach scale (performance, throughput)? Like one queue is filling up?

Example of Service Composition

- Some potentially nagging questions for you ☺
  - How is state of composition managed?
  - What happens in case of failures? Assume (a) transactional queuing systems (b) non-transactional, but persistent queuing system (c) non-transactional and non-persistent queuing system?
  - How does recovery from those failures work?
  - How does this approach scale (performance, throughput)? Like one queue is filling up?
  - How does this approach adjust for various job mixes of online-transactions (human user over UI) and off-line transactions (B2B users over LANs and WANs)? Performance vs. throughput?

Example of Service Composition

- Some potentially nagging questions for you ☺
  - How is state of composition managed?
  - What happens in case of failures? Assume (a) transactional queuing systems (b) non-transactional, but persistent queuing system (c) non-transactional and non-persistent queuing system?
  - How does recovery from those failures work?
  - How does this approach scale (performance, throughput)? Like one queue is filling up?
  - How does this approach adjust for various job mixes of online-transactions (human user over UI) and off-line transactions (B2B users over LANs and WANs)? Performance vs. throughput?
  - How is the status and the history of execution tracked?

Example of Service Composition

- Some potentially nagging questions for you ☺
  - How is state of composition managed?
  - What happens in case of failures? Assume (a) transactional queuing systems (b) non-transactional, but persistent queuing system (c) non-transactional and non-persistent queuing system?
  - How does recovery from those failures work?
  - How does this approach scale (performance, throughput)? Like one queue is filling up?
  - How does this approach adjust for various job mixes of online-transactions (human user over UI) and off-line transactions (B2B users over LANs and WANs)? Performance vs. throughput?
  - How is the status and the history of execution tracked?
  - …

Switch Perspective

- From
  - SOA view
  - = Computer science

Switch Perspective

- From
  - SOA view
  - = Computer science
- to
  - IT view
  - = Information technology (i.e., implementation of business functionality)

Business Processes

Business Processes

- Bad news: today still not implemented with workflow / process management systems in general

Business Processes

- Bad news: today still not implemented with workflow / process management systems in general
- Good news: does not make difference to following discussion

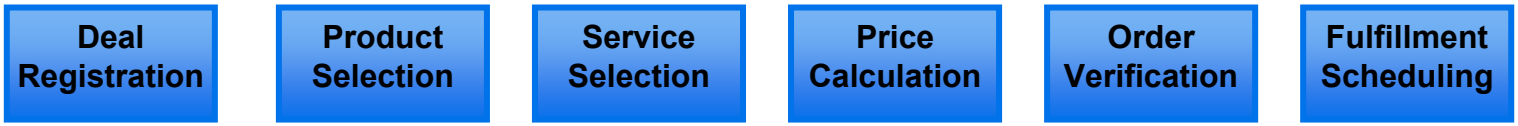
Business Processes

- Bad news: today still not implemented with workflow / process management systems in general
- Good news: does not make difference to following discussion
- Implementation?

Business Processes

- Bad news: today still not implemented with workflow / process management systems in general
- Good news: does not make difference to following discussion
- Implementation!
  - Implicit by
    - Data attribute states
    - Messages in specific queues
    - Knowledge of people
    - ‘Process’ in packaged application systems

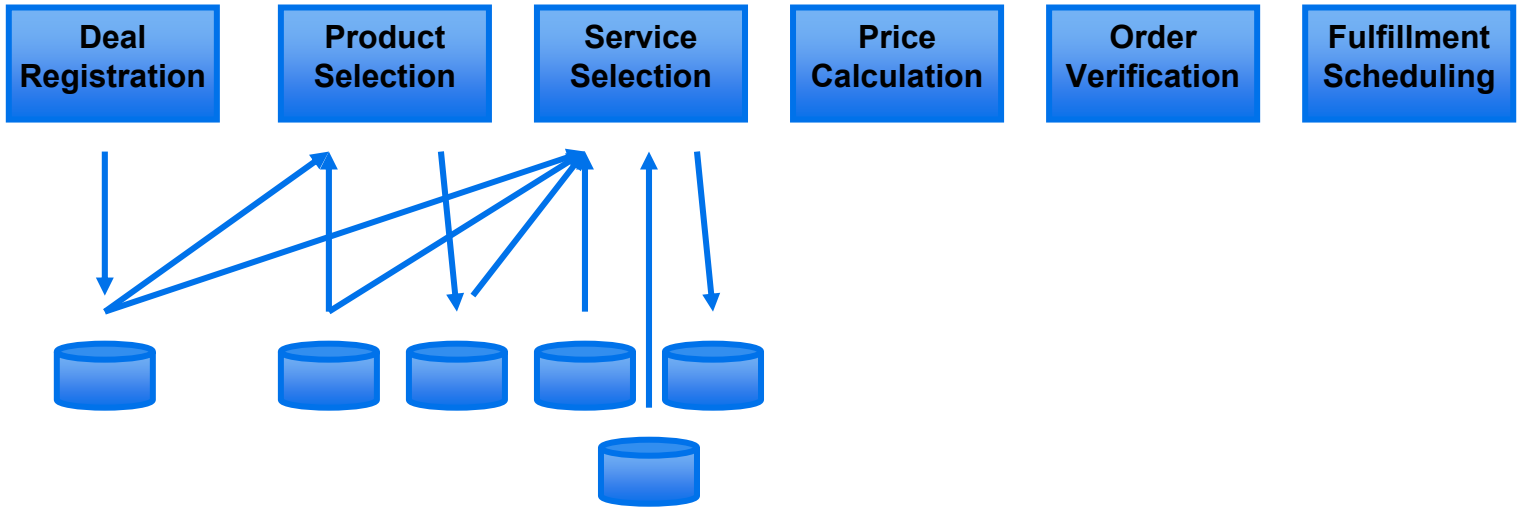
(Simplified) Business Process Example



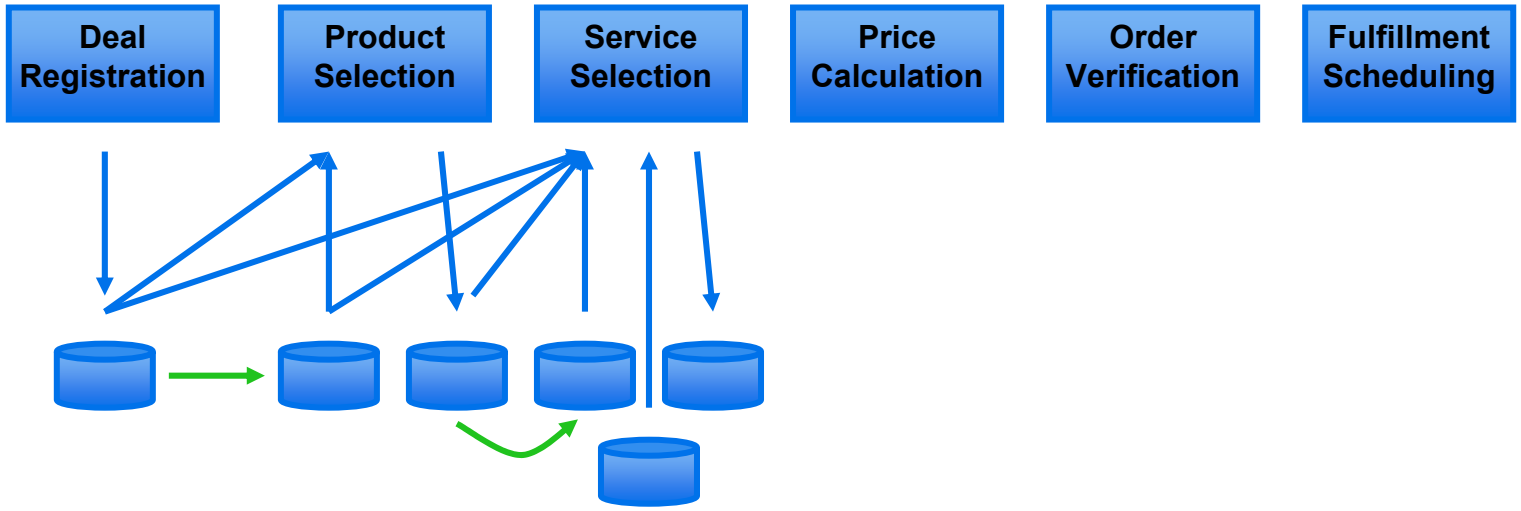
(Simplified) Business Process Example

(Simplified) Business Process Example

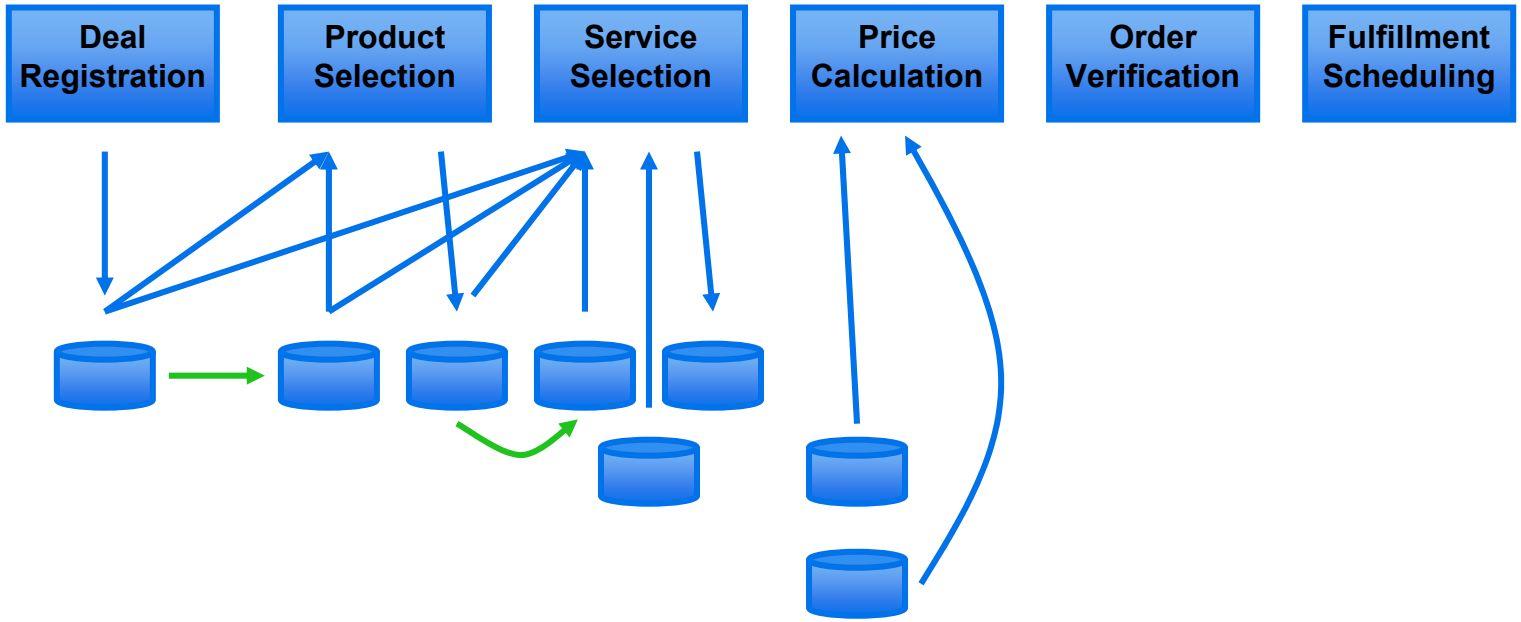
(Simplified) Business Process Example



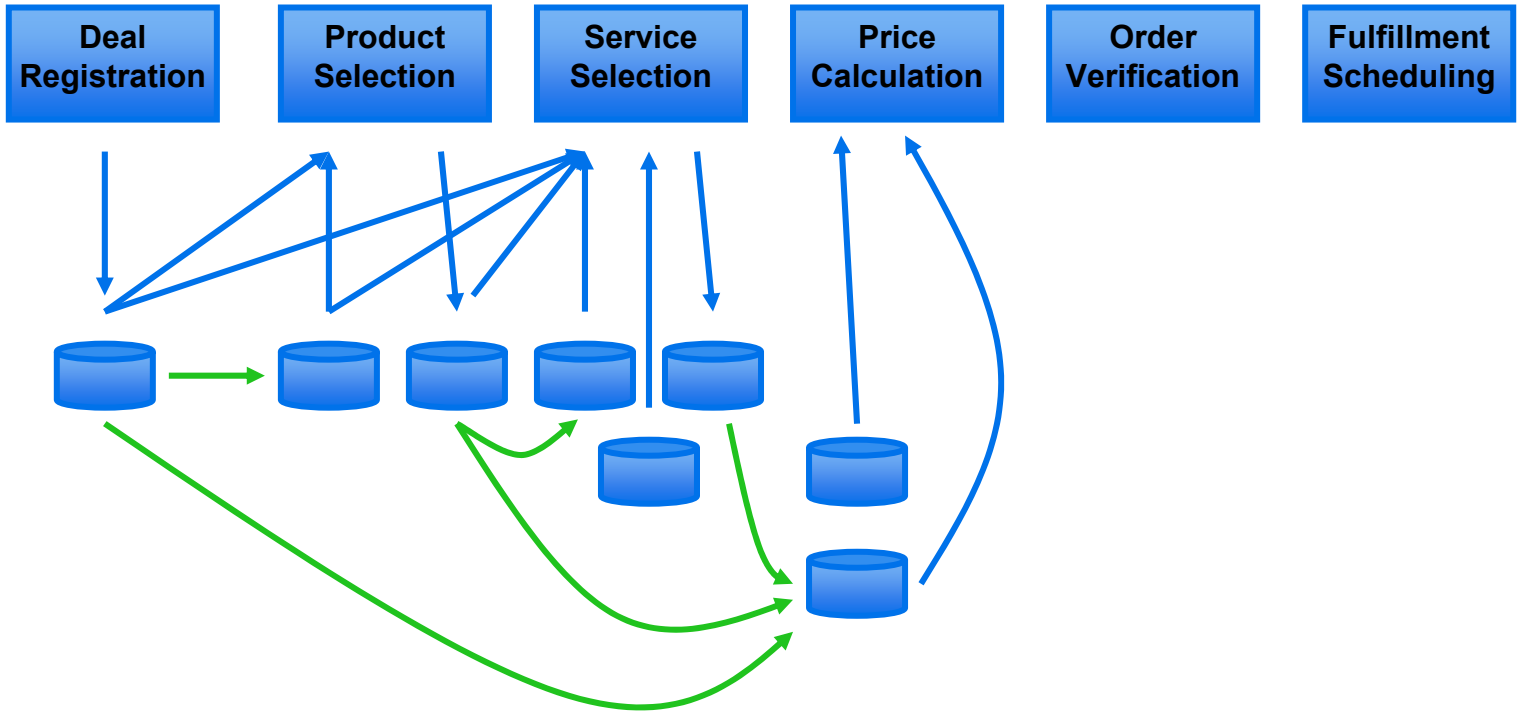
(Simplified) Business Process Example



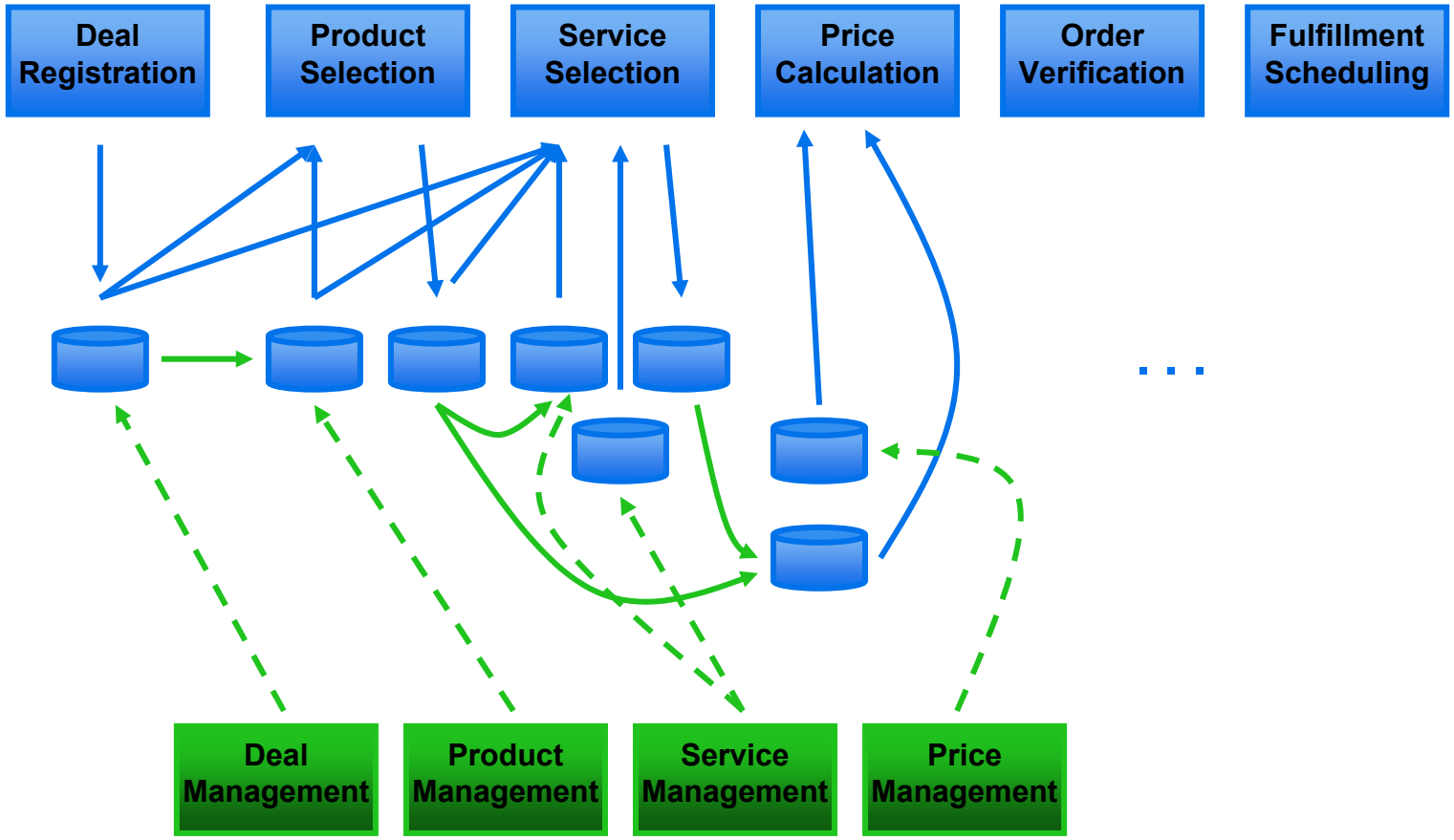
(Simplified) Business Process Example



(Simplified) Business Process Example



(Simplified) Business Process Example



Big Question

- How to implement business functionality with a SOA?

Switch Perspective

- From
  - IT view
  - = Information technology (i.e., implementation of business functionality)

Switch Perspective

- From
  - IT view
  - = Information technology (i.e., implementation of business functionality)
- to
  - Architecture view
  - = System Architecture (i.e., complex system abstractions)

Common Architecture Patterns

- UI read
  - Data in heterogeneous sources

Common Architecture Patterns

- UI read
  - Data in heterogeneous sources
- UI write
  - Data in heterogeneous sources

Common Architecture Patterns

- UI read
  - Data in heterogeneous sources
- UI write
  - Data in heterogeneous sources
- Automated read / write
  - Non-user processing of heterogeneous data

Common Architecture Patterns

- UI read
  - Data in heterogeneous sources
- UI write
  - Data in heterogeneous sources
- Automated read / write
  - Non-user processing of heterogeneous data
- Task and data (by value) flow

Common Architecture Patterns

- UI read
  - Data in heterogeneous sources
- UI write
  - Data in heterogeneous sources
- Automated read / write
  - Non-user processing of heterogeneous data
- Task and data (by value) flow
- Task and data (by reference) flow

Common Architecture Patterns

- UI read
  - Data in heterogeneous sources
- UI write
  - Data in heterogeneous sources
- Automated read / write
  - Non-user processing of heterogeneous data
- Task and data (by value) flow
- Task and data (by reference) flow
- Data flow
  - Data in exactly one place at any given time

Common Architecture Patterns

- UI read
  - Data in heterogeneous sources
- UI write
  - Data in heterogeneous sources
- Automated read / write
  - Non-user processing of heterogeneous data
- Task and data (by value) flow
- Task and data (by reference) flow
- Data flow
  - Data in exactly one place at any given time
- Data replication
  - Data replicas consistent at any point in time everywhere

Common Architecture Patterns

- UI read
  - Data in heterogeneous sources
- UI write
  - Data in heterogeneous sources
- Automated read / write
  - Non-user processing of heterogeneous data
- Task and data (by value) flow
- Task and data (by reference) flow
- Data flow
  - Data in exactly one place at any given time
- Data replication
  - Data replicas consistent at any point in time everywhere
- Data master-slave
  - Data slaves lag data master state

Common Architecture Patterns

- UI read
  - Data in heterogeneous sources
- UI write
  - Data in heterogeneous sources
- Automated read / write
  - Non-user processing of heterogeneous data
- Task and data (by value) flow
- Task and data (by reference) flow
- Data flow
  - Data in exactly one place at any given time
- Data replication
  - Data replicas consistent at any point in time everywhere
- Data master-slave
  - Data slaves lag data master state
- Packaged application system access
  - Remote access
  - Data flow/replication/master-slave within application systems

Common Architecture Patterns

- Concurrent data management
  - Data replication / flow / master-slave of data

Common Architecture Patterns

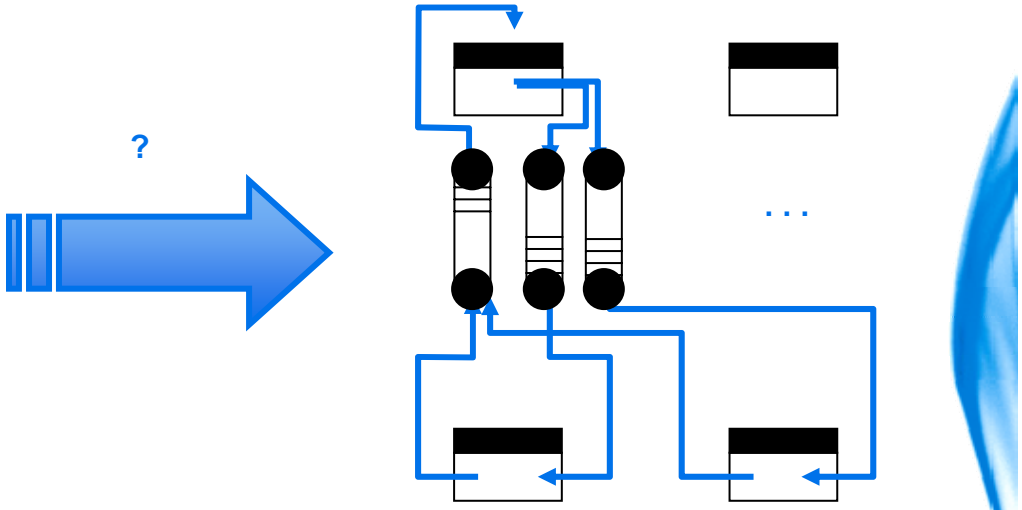
- Concurrent data management
  - Data replication / flow / master-slave of data
- Concurrent processes
  - Creation, update, deletion of sets of complex data objects

Common Architecture Patterns

- Concurrent data management
  - Data replication / flow / master-slave of data
- Concurrent processes
  - Creation, update, deletion of sets of complex data objects
- Dependencies
  - E. g., do not allow process until data is in a specific state

One SOA Approach Revisited

- UI read
- UI write
- Automated read / write
- Task and data (by value) flow
- Task and data (by reference) flow
- Data flow
- Data replication
- Data master-slave
- Packaged application system access
- Concurrent data management
- Concurrent processes
- Dependencies



bea
Think liquid.

What's Wrong?

Horizontal Abstraction

- Computer science view
  - Layered architecture
  - Abstraction of invocation infrastructure and functionality

Horizontal Abstraction

- Computer science view
  - Layered architecture
  - Abstraction of invocation infrastructure and functionality
- The message-based approach to SOA tries to provide a single abstraction to all architecture patterns
  - SX, S, M, L, XL, XXL

Horizontal Abstraction

- Computer science view
  - Layered architecture
  - Abstraction of invocation infrastructure and functionality
- The message-based approach to SOA tries to provide a single abstraction to all architecture patterns
  - SX, S, M, L, XL, XXL
  - Uniformity
    - (Comfortable and convenient)

Functionality Abstraction

- Needed:

Functionality Abstraction

- Needed:
- Building blocks that implement higher-level functionality supporting architecture patterns

Functionality Abstraction

- Needed:
- Building blocks that implement higher-level functionality supporting architecture patterns
- Building blocks that implement appropriate (horizontal) layering for the architecture pattern they implement

Functionality Abstraction

- Needed:
- Building blocks that implement higher-level functionality supporting architecture patterns
- Building blocks that implement appropriate (horizontal) layering for the architecture pattern they implement
- Building blocks that have (vertical) layering so that they can combined for business process implementation

Functionality Abstraction

Functionality Abstraction

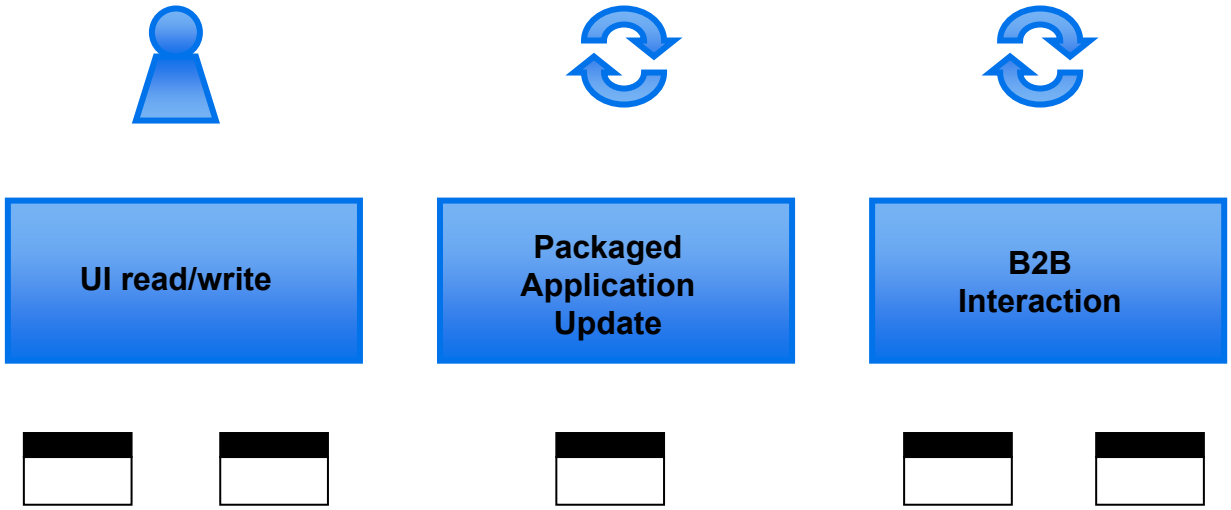
UI read/write

Packaged Application Update

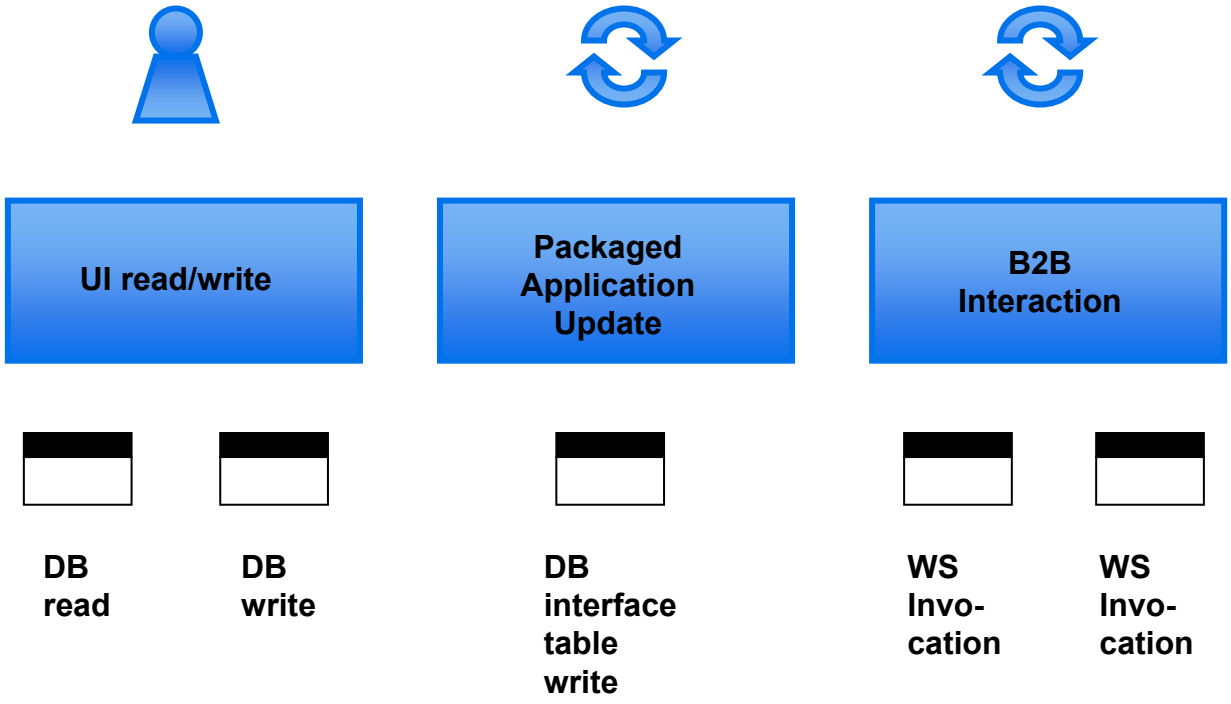
B2B Interaction

bea Think liquid.

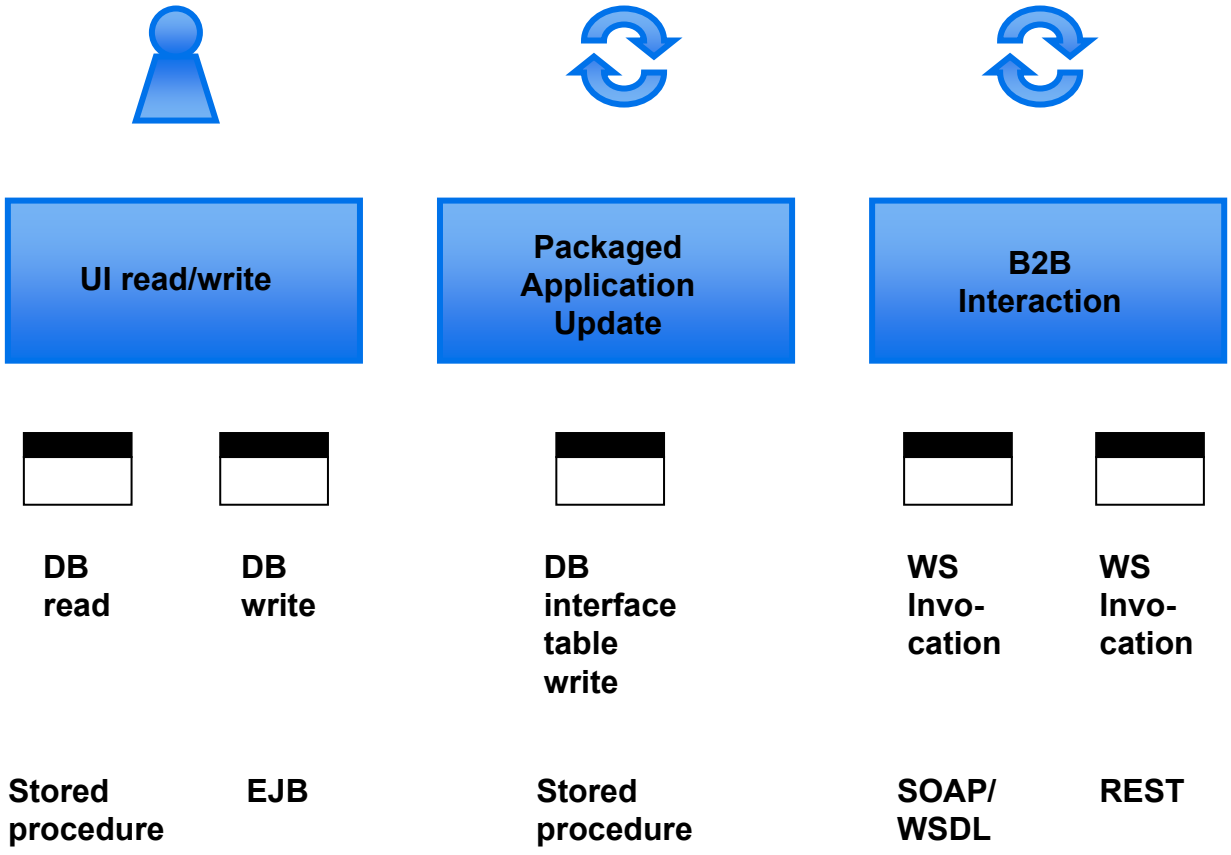
Functionality Abstraction



Functionality Abstraction



Functionality Abstraction

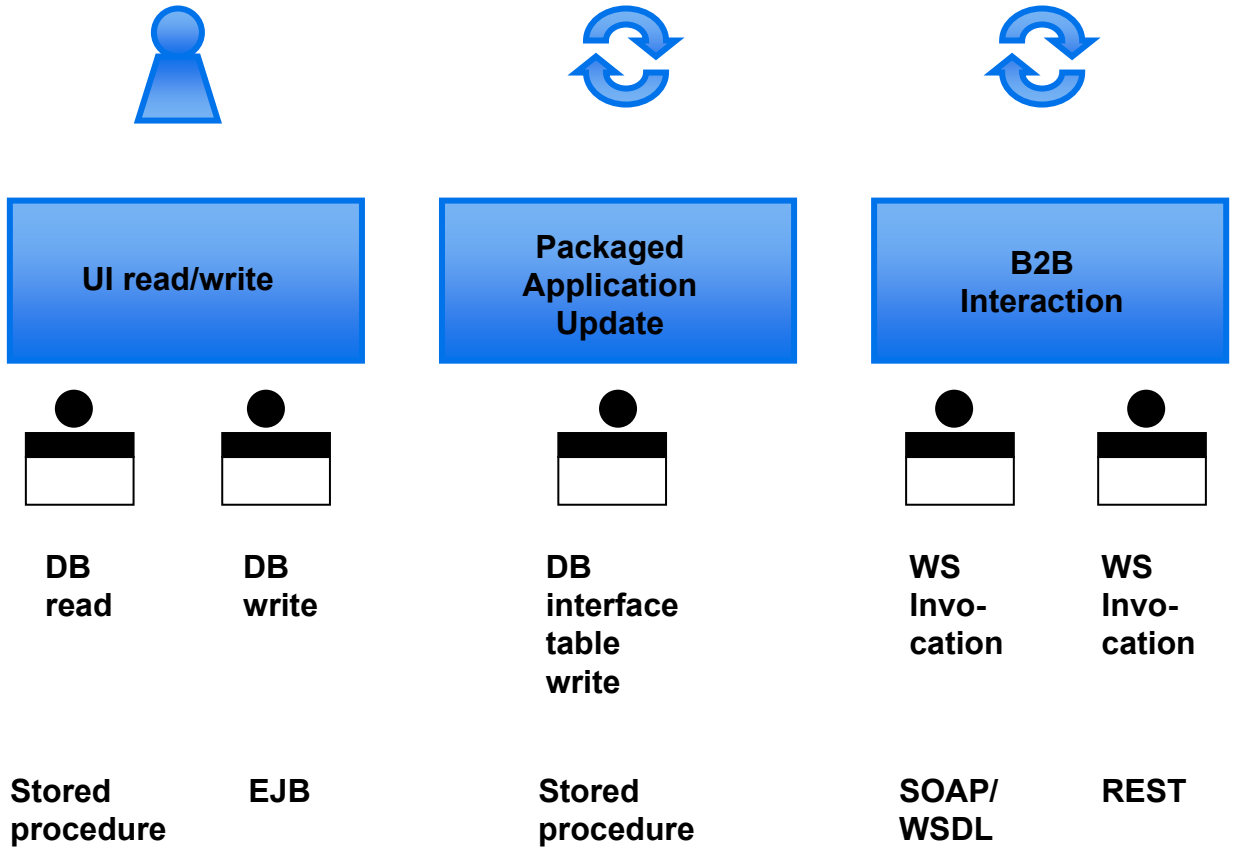


Functionality Abstraction

Definition						
	DB read	DB write	DB interface table write	WS Invo-cation	WS Invo-cation	
Implementa-tion	Stored procedure	EJB	Stored procedure	SOAP/ WSDL	REST	

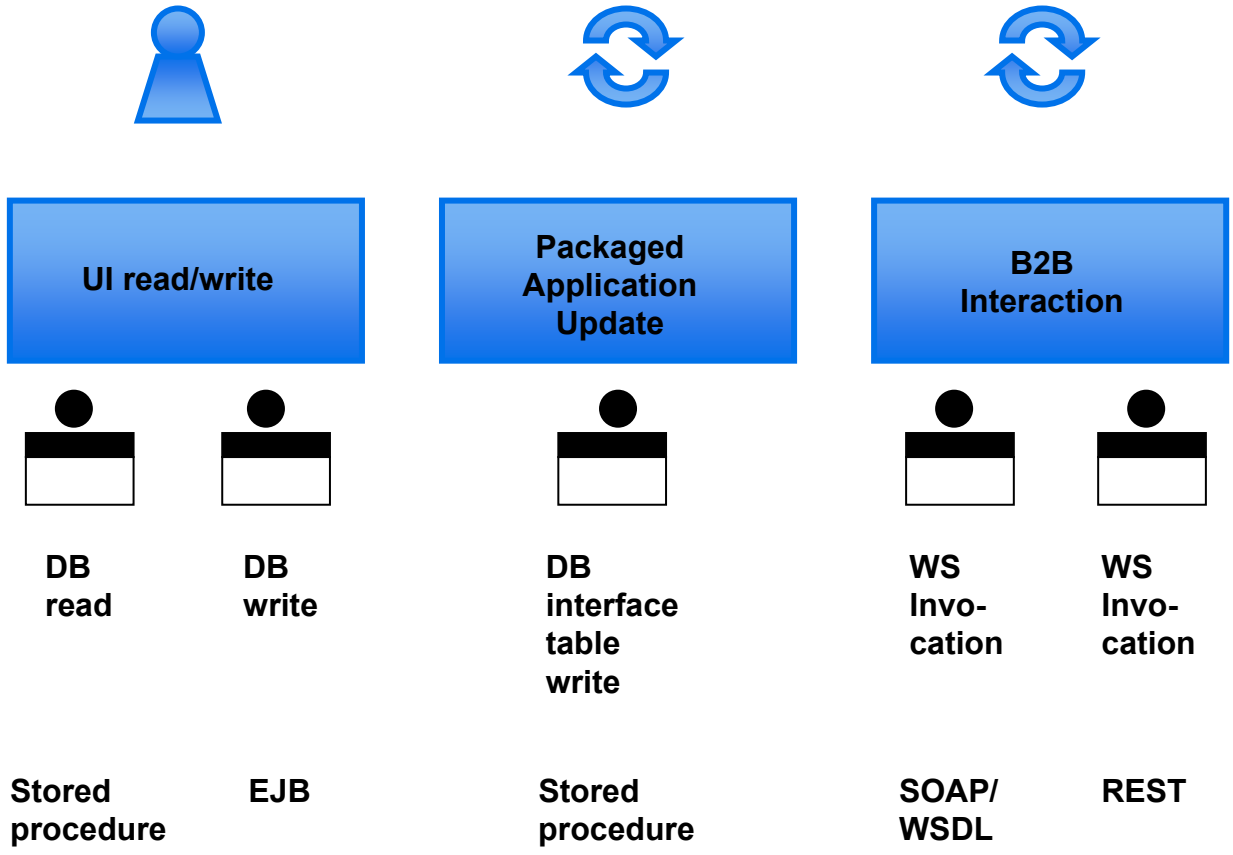
bea Think liquid.™

Functionality Abstraction



bea Think liquid.

Functionality Abstraction

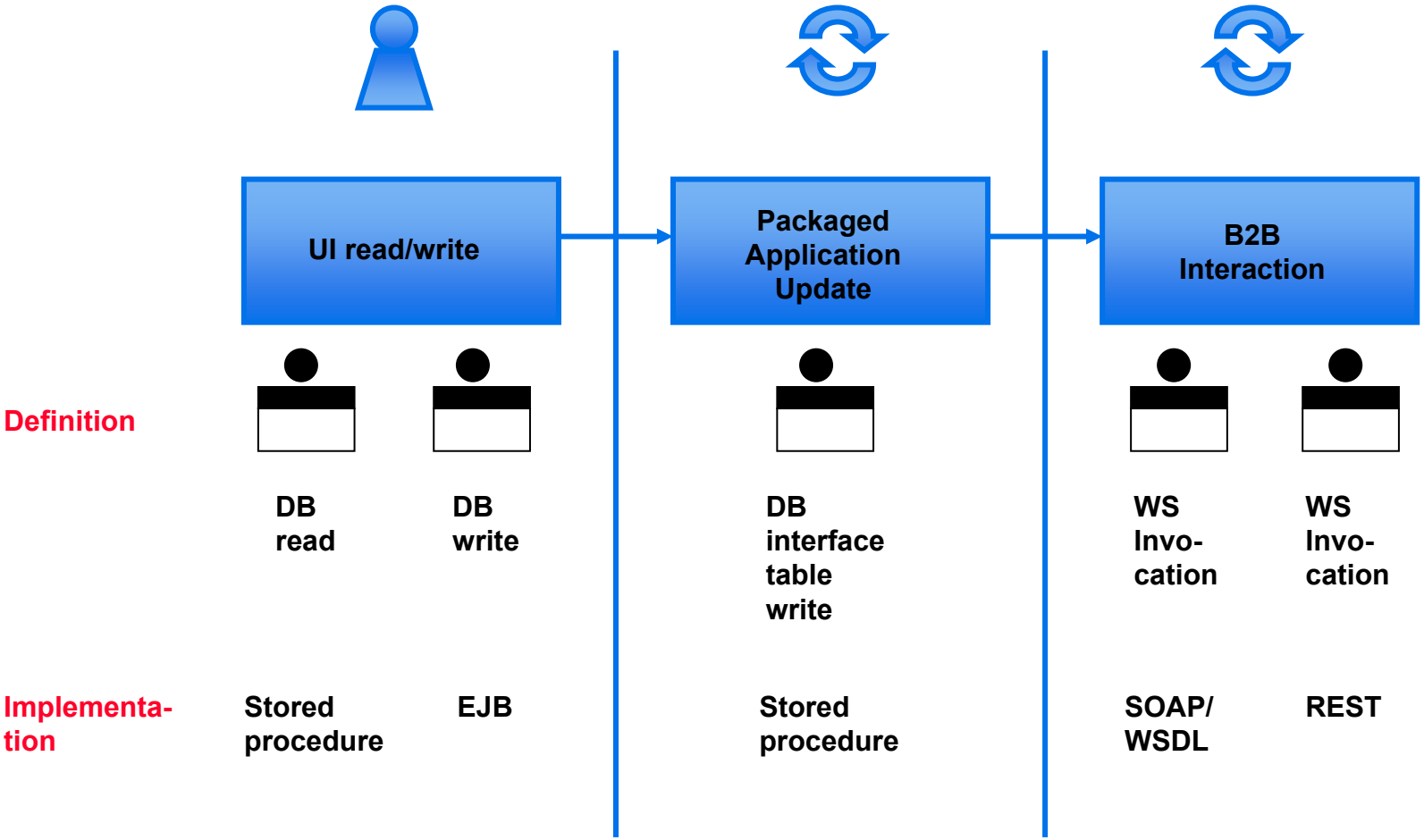


bea Think liquid.

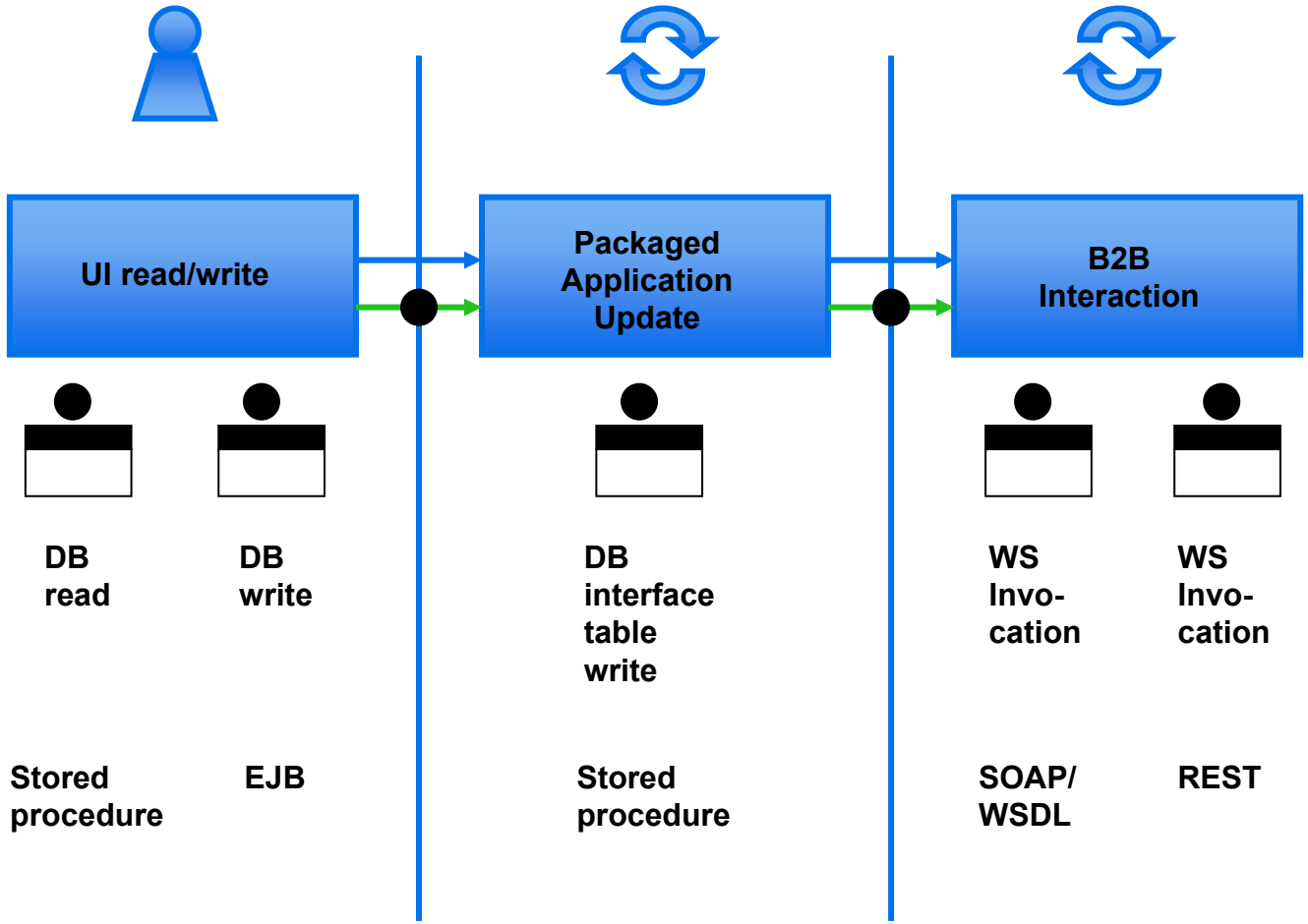
Functionality Abstraction

	UI read/write		Packaged Application Update	B2B Interaction	
Definition	DB read	DB write	DB interface table write	WS Invo-cation	WS Invo-cation
Implementa-tion	Stored procedure	EJB	Stored procedure	SOAP/ WSDL	REST

Functionality Abstraction



Functionality Abstraction



SOA vs. SOI

- “A” Architecture
- Architecture
  - Building blocks to satisfy architecture patterns
  - Independent of implementation (OASIS SOA RM)

SOA vs. SOI

- “A” Architecture, “I” Implementation
- Architecture
  - Building blocks to satisfy architecture patterns
  - Independent of implementation (OASIS SOA RM)
- Implementation
  - Building blocks to address architecture patterns and computer science abstractions
  - Building blocks have two interfaces
    - Horizontal one into infrastructure
    - Vertical one to each other

Conclusions

- SOA
  - One size fits nobody
  - Single horizontal abstraction too simplified

Conclusions

- SOA
  - One size fits nobody
  - Single horizontal abstraction too simplified
- SOA with functionality abstraction
  - Many sizes fit 80%
  - Specific building blocks target specific functionality