أكاديمية الحكومة الإلكترونية الفلسطينية
The Palestinian eGovernment Academy
www.egovacademy.ps

Tutorial III:
Process Integration and Service Oriented Architectures

Session 11
Integration Patterns

Prepared By

Mohammed Melhem

European Commission
TEMPUS

About

This tutorial is part of the PalGov project, funded by the TEMPUS IV program of the Commission of the European Communities, grant agreement 511159-TEMPUS-1-2010-1-PS-TEMPUS-JPHES. The project website: www.egovacademy.ps

Project Consortium:

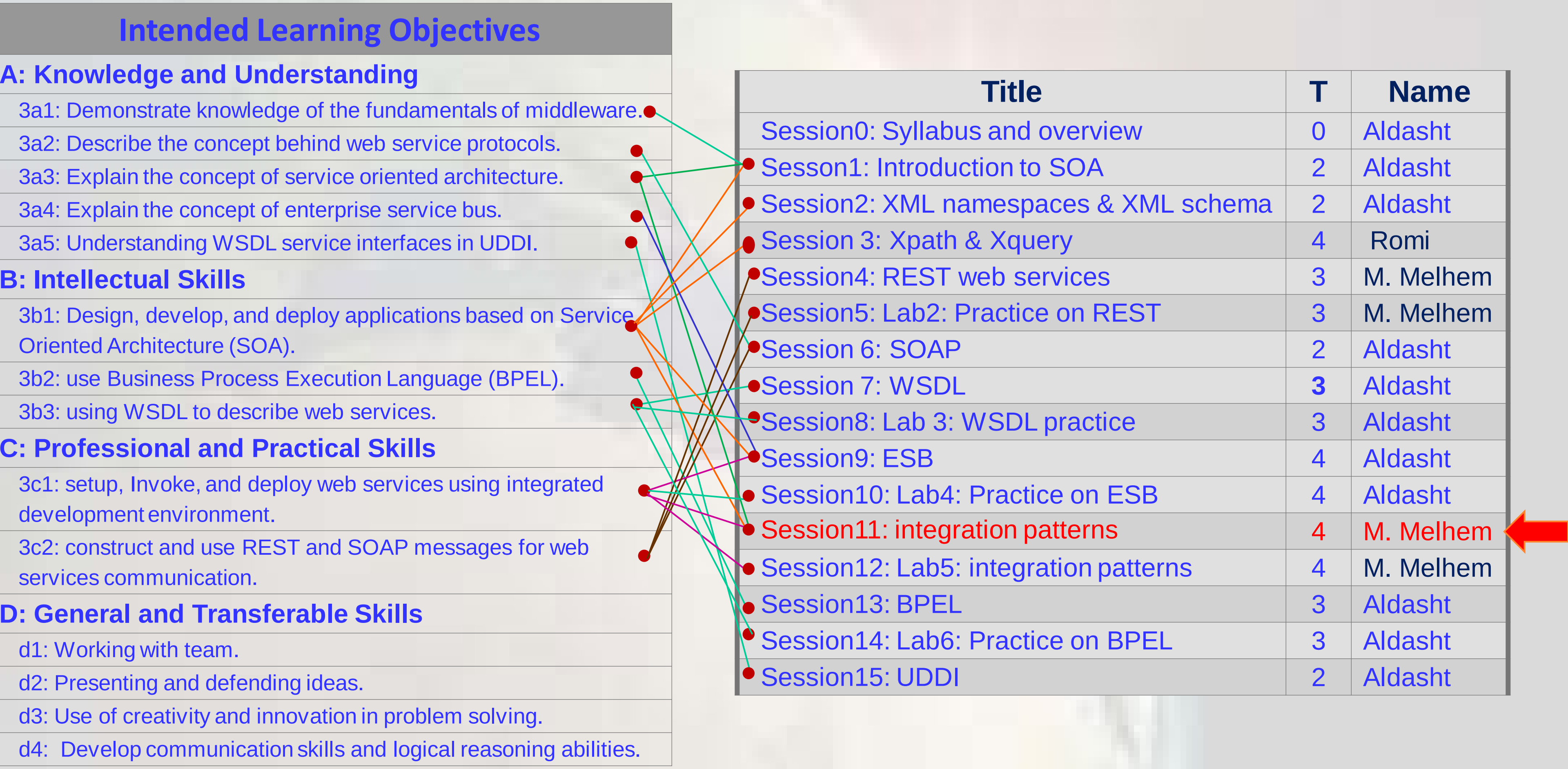
	Birzeit University, Palestine (**Coordinator**)		University of Trento, Italy
	Palestine Polytechnic University, Palestine		Vrije Universiteit Brussel, Belgium
	Palestine Technical University, Palestine		Université de Savoie, France
	Ministry of Telecom and IT, Palestine		University of Namur, Belgium
	Ministry of Interior, Palestine		TrueTrust, UK
	Ministry of Local Government, Palestine		

Coordinator:
Dr. Mustafa Jarrar
Birzeit University, P.O.Box 14- Birzeit, Palestine
Telfax:+972 2 2982935 mjarrar@birzeit.edu

The Palestinian e-government academy

© Copyright Notes

Tutorial Map



Intended Learning Objectives
A: Knowledge and Understanding
3a1: Demonstrate knowledge of the fundamentals of middleware.
3a2: Describe the concept behind web service protocols.
3a3: Explain the concept of service oriented architecture.
3a4: Explain the concept of enterprise service bus.
3a5: Understanding WSDL service interfaces in UDDI.
B: Intellectual Skills
3b1: Design, develop, and deploy applications based on Service Oriented Architecture (SOA).
3b2: use Business Process Execution Language (BPEL).
3b3: using WSDL to describe web services.
C: Professional and Practical Skills
3c1: setup, Invoke, and deploy web services using integrated development environment.
3c2: construct and use REST and SOAP messages for web services communication.
D: General and Transferable Skills
d1: Working with team.
d2: Presenting and defending ideas.
d3: Use of creativity and innovation in problem solving.
d4: Develop communication skills and logical reasoning abilities.

Title	T	Name
Session0: Syllabus and overview	0	Aldasht
Sesson1: Introduction to SOA	2	Aldasht
Session2: XML namespaces & XML schema	2	Aldasht
Session 3: Xpath & Xquery	4	Romi
Session4: REST web services	3	M. Melhem
Session5: Lab2: Practice on REST	3	M. Melhem
Session 6: SOAP	2	Aldasht
Session 7: WSDL	**3**	Aldasht
Session8: Lab 3: WSDL practice	3	Aldasht
Session9: ESB	4	Aldasht
Session10: Lab4: Practice on ESB	4	Aldasht
Session11: integration patterns	4	M. Melhem
Session12: Lab5: integration patterns	4	M. Melhem
Session13: BPEL	3	Aldasht
Session14: Lab6: Practice on BPEL	3	Aldasht
Session15: UDDI	2	Aldasht

Session 11: SOA design and Integration Patterns.

This session aims to cover Service oriented architecture and how to utilize it in systems integration using different patterns and methodologies of best practice.

After completing this session students will be able to:

1. Link SOA principles to integration
2. Open doors to integration patterns and understand how to use it in Enterprise Applications

Session contents

1. Review of Service Oriented Architecture
2. SOA – Principles
3. Integration
    1. The challenge
    2. Patterns
    3. Enterprise integration patterns
    4. Messaging
    5. Why web services
    6. Additional Patterns
4. SOA Integration Patterns
5. Summery

SOA “Service Oriented Architecture” Motivation

- Challenges in enterprise application development
  - Many users and backend systems interact with the system.
  - Quality of services (QoS) requirements and other Non-functional requirements .
  - Development and integration projects costly and long running
    - For example point – to – point connections, often developed from scratch.

SOA “Service Oriented Architecture” Motivation (cont.)

- Solution evolved into Service Oriented Architecture
  - Message Backbone
    - Point to point connection between applications
    - Simple, basic connectivity
  - Enterprise Application Integration
    - Connect application through a centralized hub
    - Easier to manage larger number of connections
  - Service Oriented Architecture
    - Integration and choreography of services through an Enterprise Service Bus
    - Flexible connections with well defined, standards-based interfaces

The Palestinian e-government academy

SOA "Service Oriented Architecture" Definition

- No Single definition, different sectors different meaning
- Architect
  - It is an Architectural style that aims to achieve loose coupling among interacting software agents.
- Business Domain
  - Set of services exposed to customers an
- Developers
  - A programming and deployment model realized by standards, tools and technologies such as Web services.

SOA Principles

- Loose Coupling
  Refers to the number of dependencies between modules
- Abstraction
  Contain essential information
- Reusability
- Autonomy
  Level of control over its underlying runtime execution
- Statelessness
- Discoverability (meta data)

INTEGRATION PATTERNS

The challenge

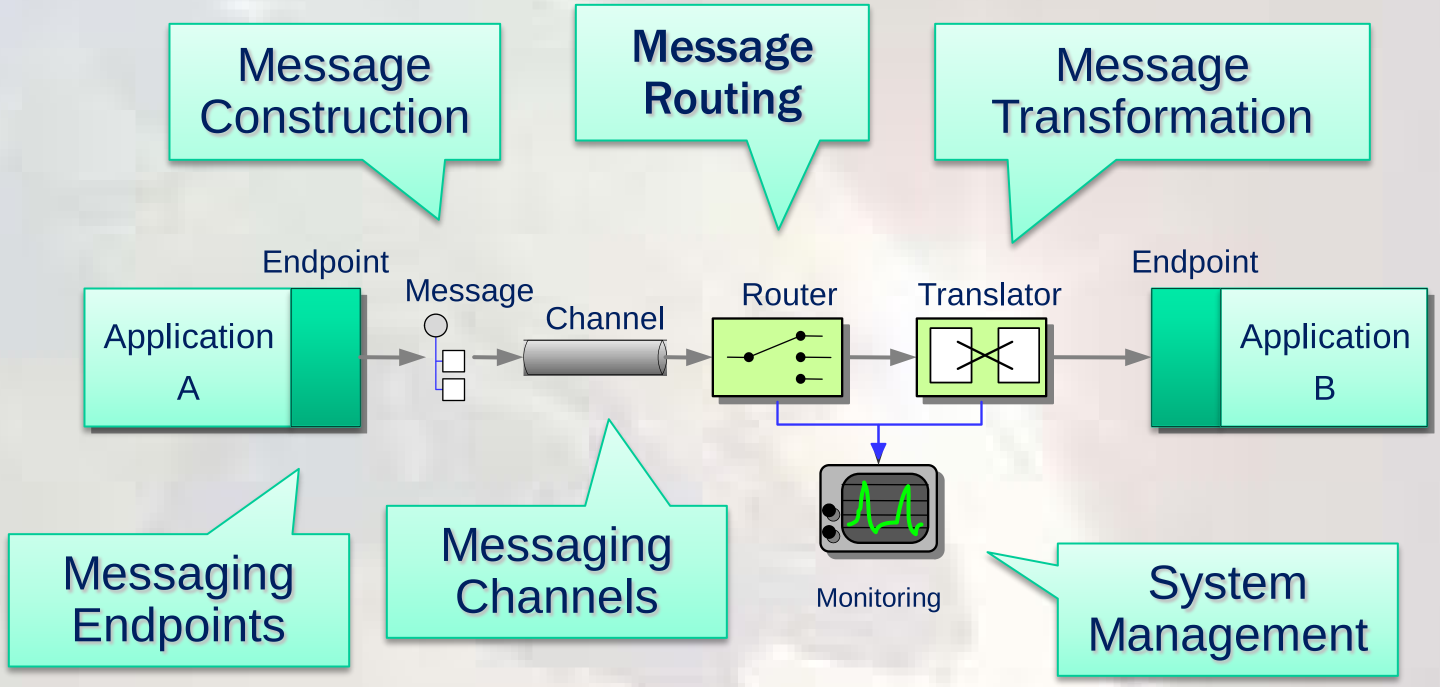
- Organizations increase its dependency on IT systems.
- Systems must provide an <u>integrated</u> solutions.
- Systems must <u>interoperate</u> with each other.
- Architectural trends of technology.
- Changing business needs and requirements

The Palestinian e-government academy

Design Patterns, Why?

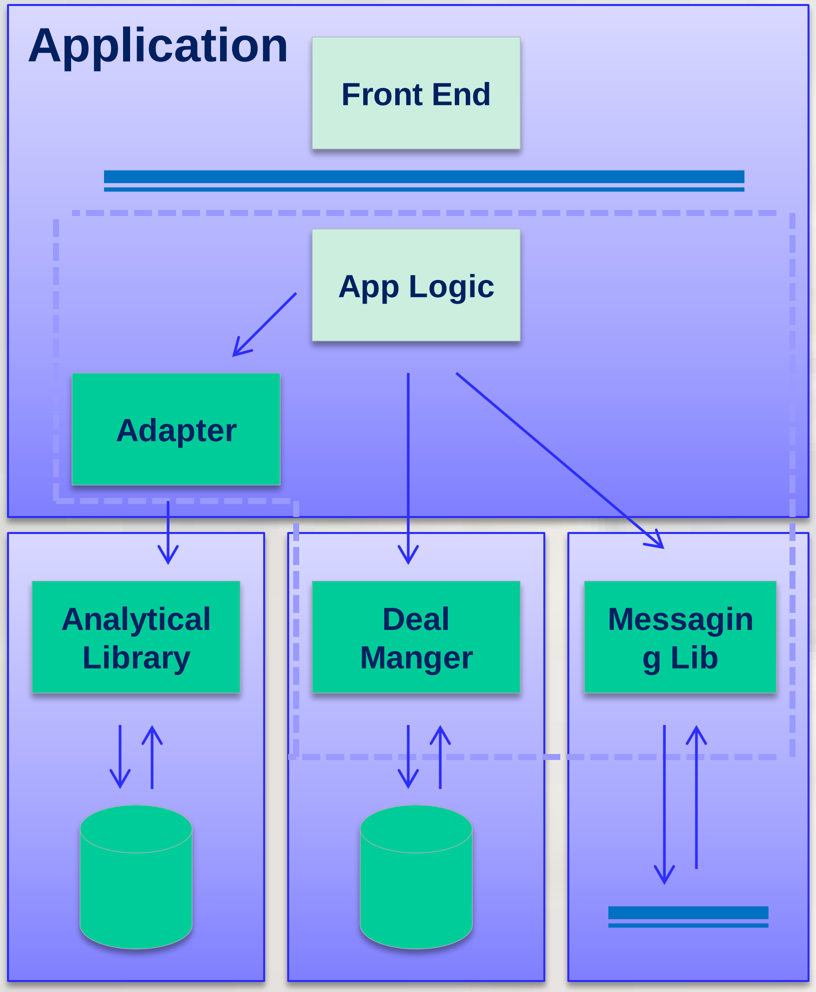
- Knowledge reuse.
- Patterns encapsulate knowledge of successful designs
- Shows good solution to a common problem within a specific context.
- Observed from real experience.
- No repetition (~~copy-paste~~)

Enterprise integration Patterns



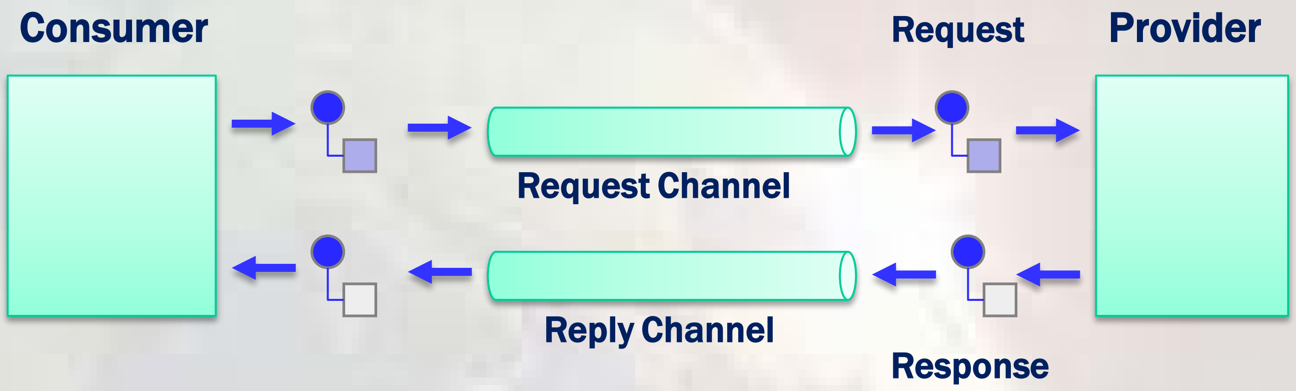
- Gregor Hohpe [1] defined a visual pattern language describing message-based enterprise integration solutions
- 6 categories of patterns contains around 65 patterns

Application Sample



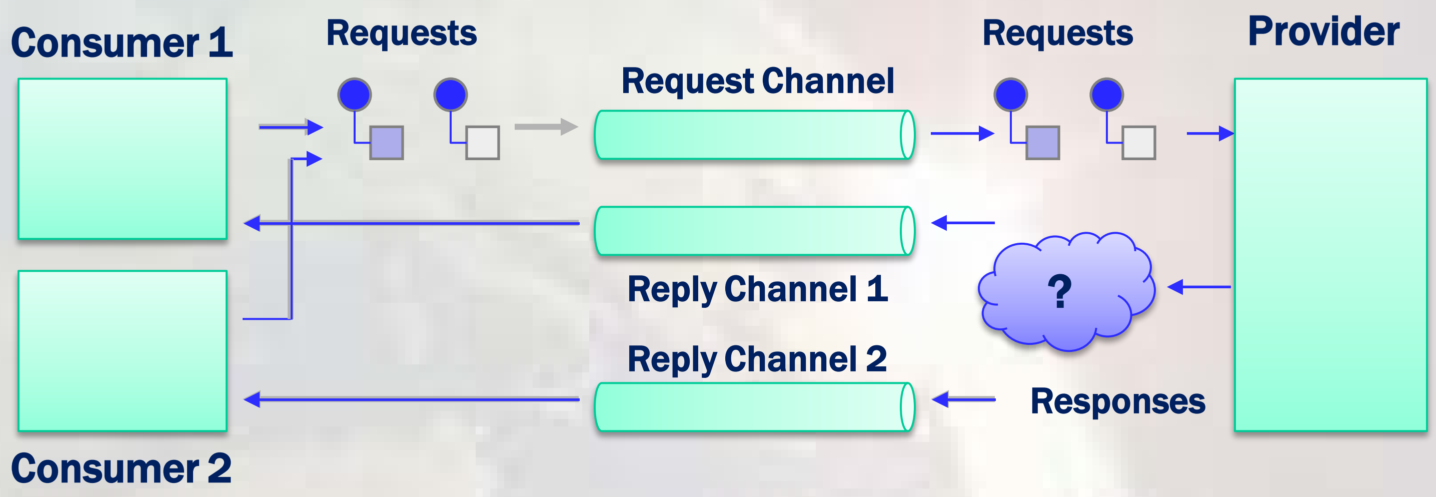
- Front end can uses web service to interact.
- Messages format defined in SOAP.
- Access analytics service using system com adapter.
- Access deals from existing services
- Access service bus through messaging abstraction library

Request – Response Pattern



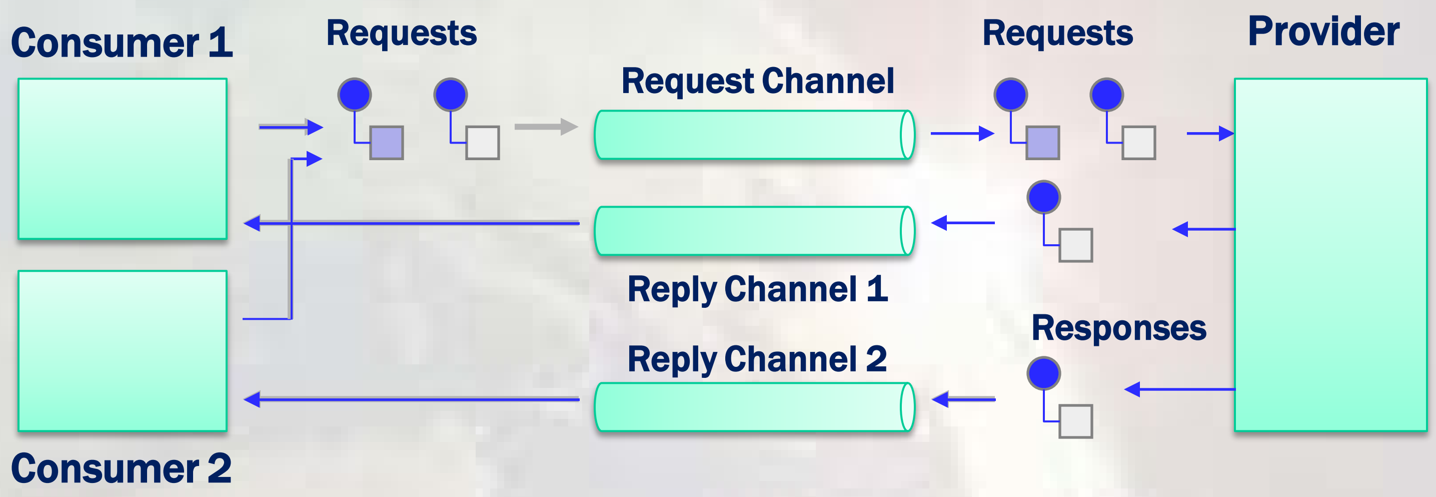
- Similar to RPC
- Asynchronous point – to – point channels
- Separate messaging

Multiple Consumer



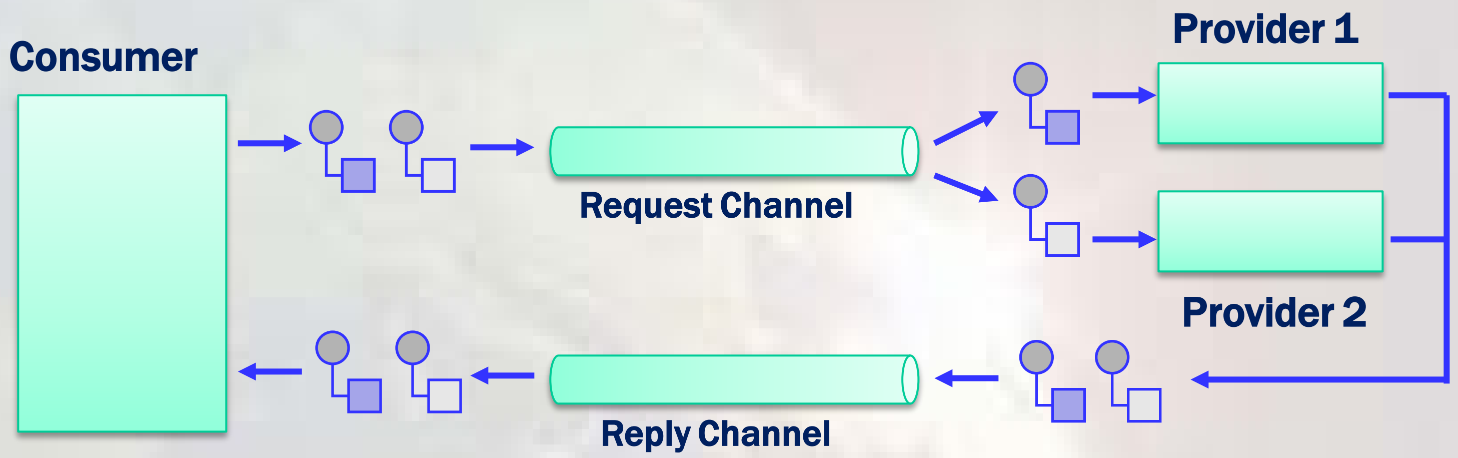
- Each consumer has its own queue
- But how does the provider send the response?
  - Could send to all consumers
  - Hard code

Return Address



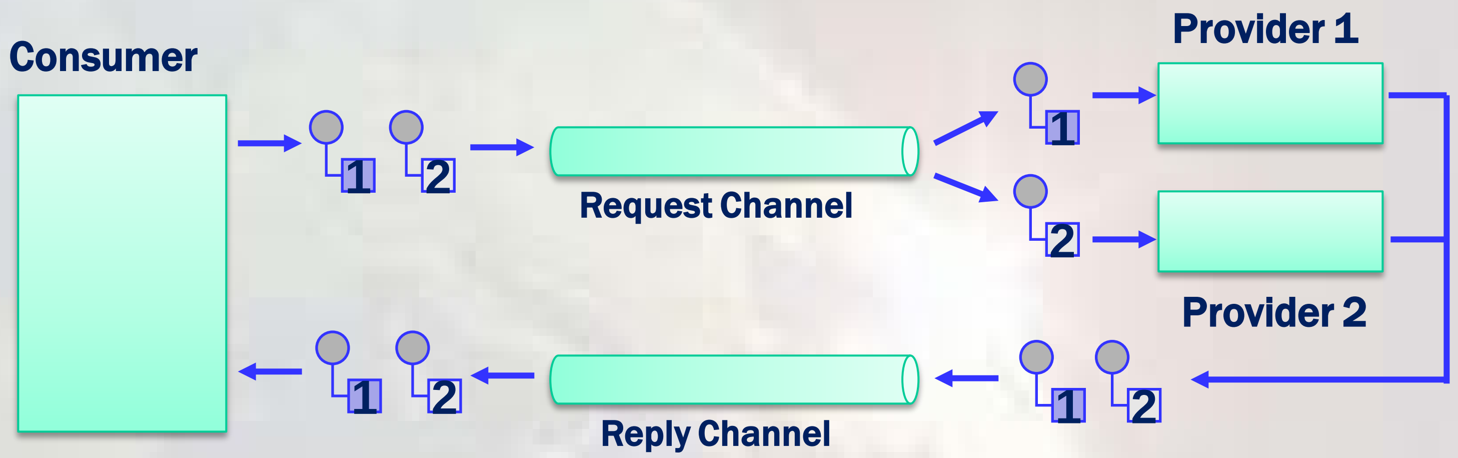
- Consumer send return address in request Header (WS-Addressing).
- Service provide send response message to consumer.

Load balancing service providers



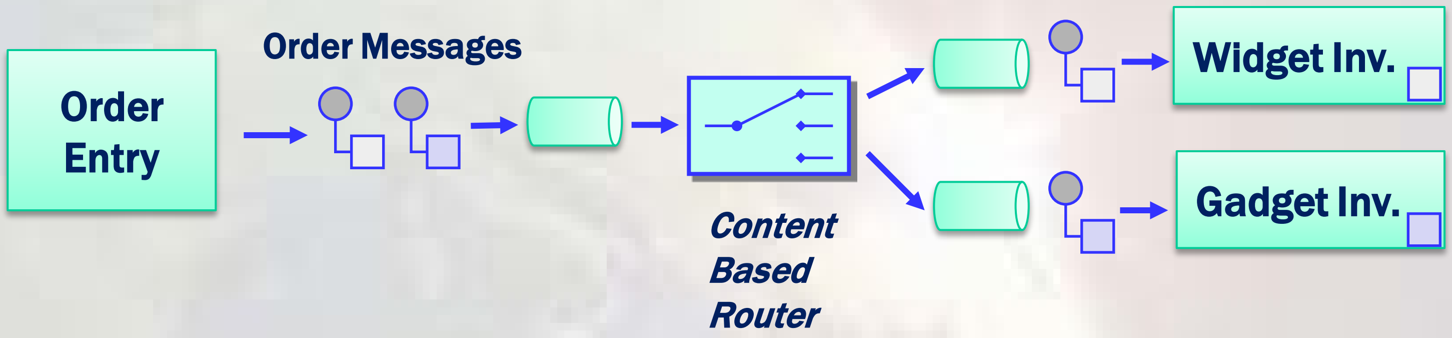
- Handle request by multiple providers.
- One service receive requests
- How to handle message order?

Correlation Identifier



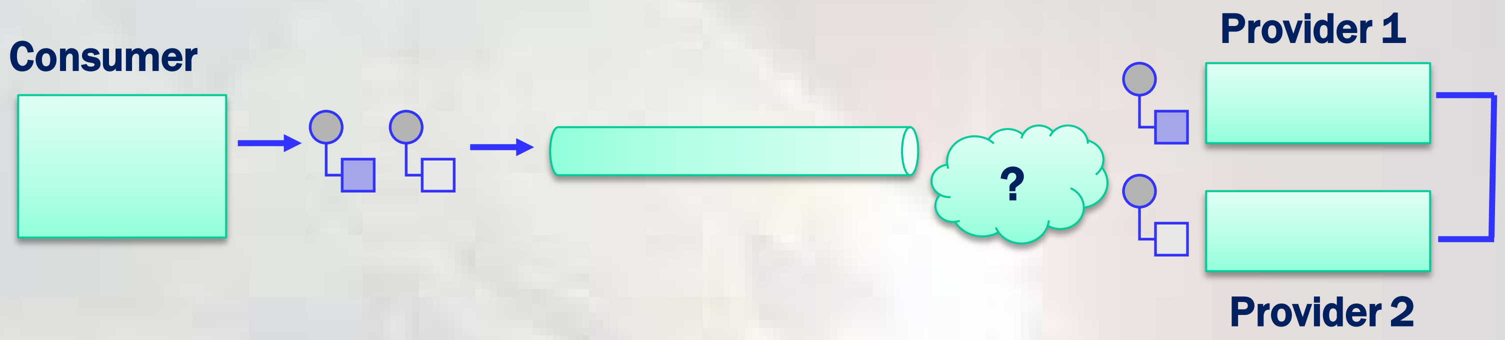
- Consumer assign an identifier to each message
    - Message Id
    - GUID
    - Business Key
- Provider include identifier in response message
- Sender Match request and response

Content-Based Router



- Add a content based router
- Router handle message forwarding
- Content remain intact

Multiple specialized provider

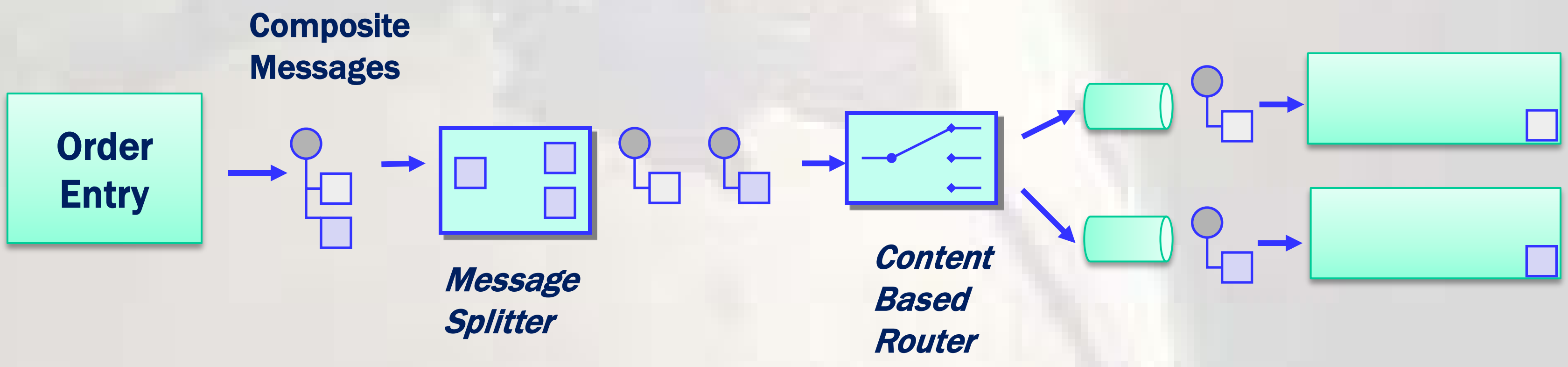


- Request message can be consumed by more than one service provider.
- Each provider can handle specific type of message
- How to route the request to the appreciate provider?
  - Sender should not worry about this task.
  - Find a coordination system

The Palestinian e-government academy

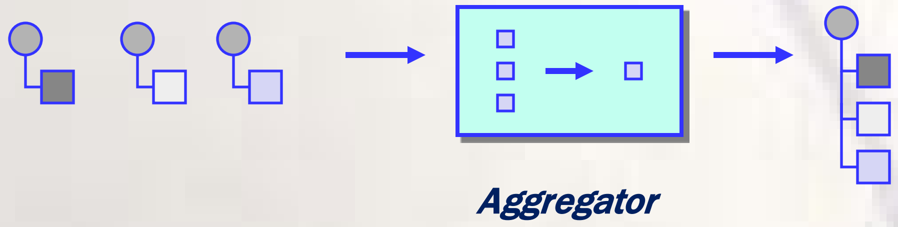
Composite messages

- How to process a message with multiple targets.
- Use a splitter to break out the message into separate messages.
- Next use router to send messages to its targets.



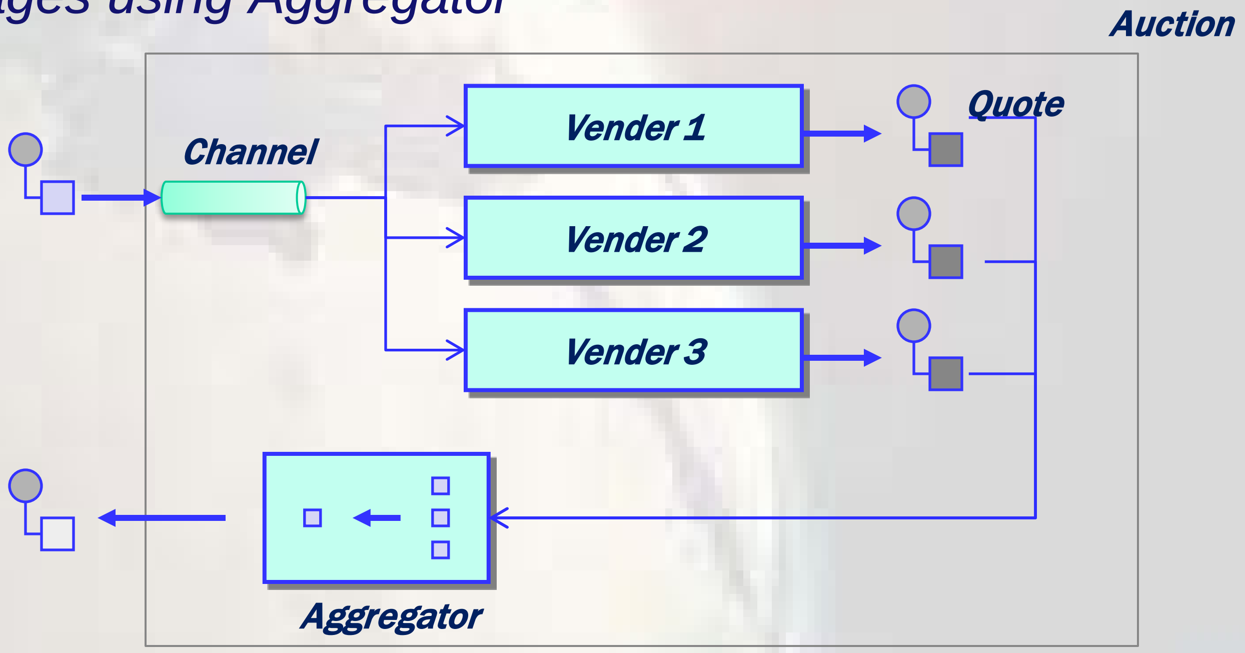
Aggregator

- How to combine results messages?
  - Messages can be out of order
  - Conversations can be intermixed
- Use a stateful filter, an Aggregator
- Collects and stores messages until a complete set has been received
- Publishes a single message created from the individual messages



Communicating with multiple parties (Auction)

- How can we send a message to dynamic set of providers? And return a single response message?
- *Scatter-Gather*
  - *Send message to pub-sub channel*
  - *Forward message to target vendor*
  - *Collect messages using Aggregator*

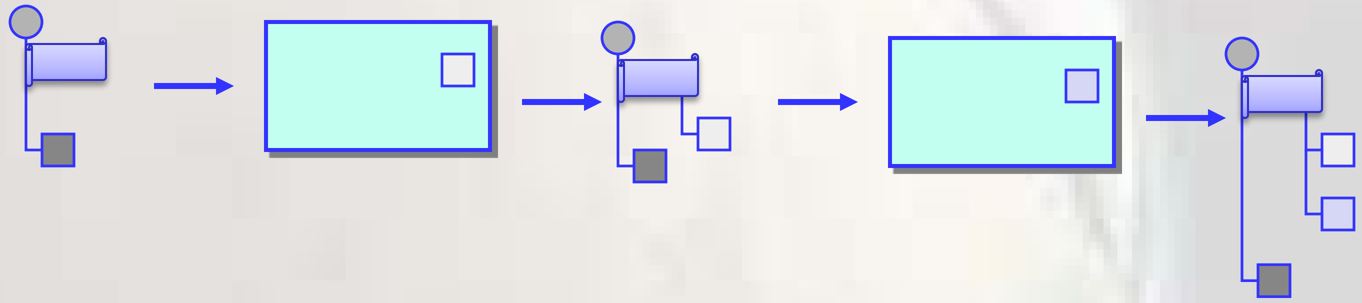


Messaging Channels

- Channels are:
  - Separate from applications
  - Asynchronous & reliable
  - Data is exchanged in self-contained messages
  - Loosely coupled integration

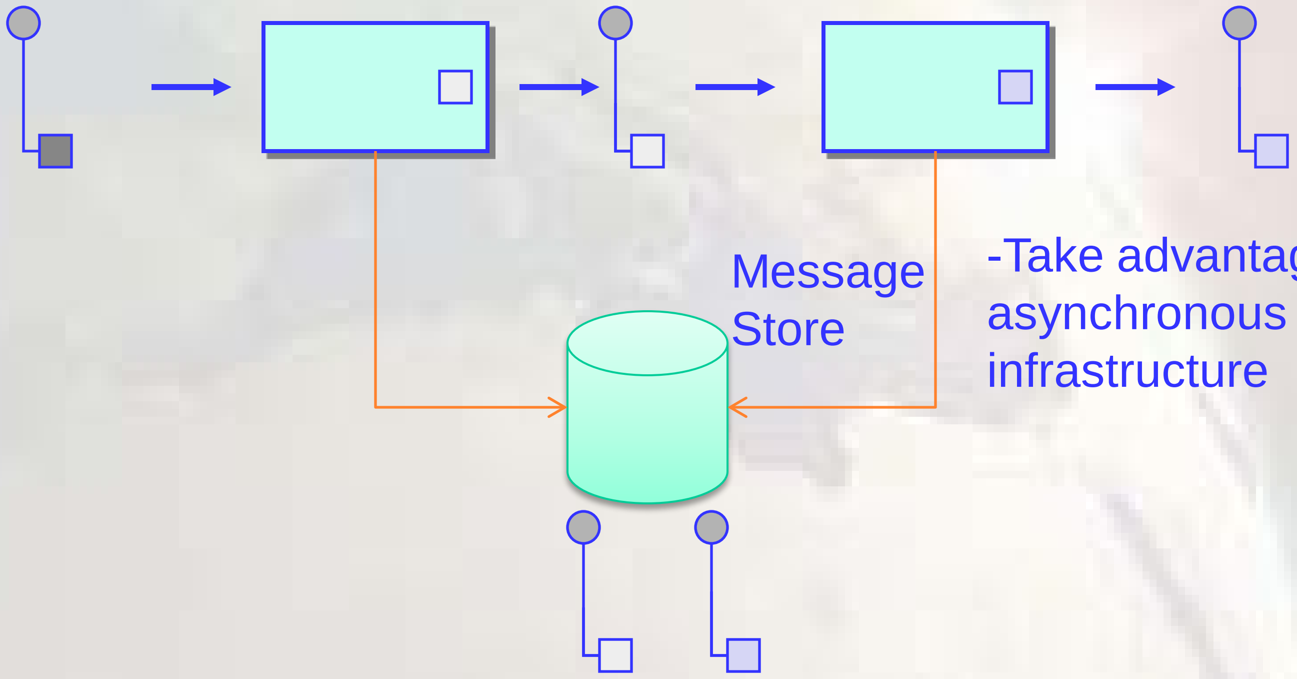
Message History

- The key benefit of message based system is the loose coupling between participants, however the same property makes debugging and analyzing dependence not an easy task.
  - If we are not ware where message goes, we can't assess the impact of the change in the message.
  - Another issue if we don't know which application produce the message, it become difficult to solve problems in messages.
- How can we effectively analyze and debug the flow of messages in a loosely coupled system?



Message Store

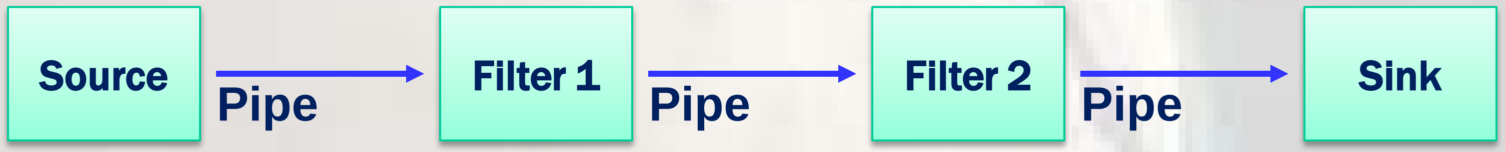
- How we can report of message status, without loosely coupled and transient nature of messaging?



-Take advantage of asynchronous nature of messaging infrastructure

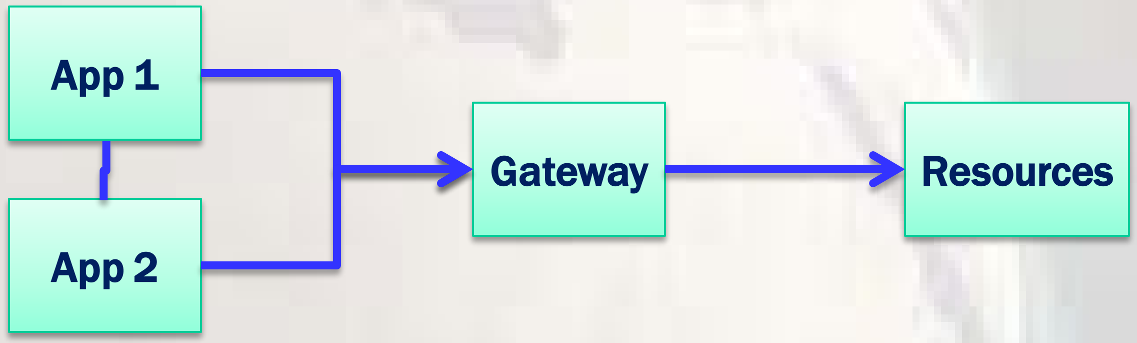
Additional Patterns (Pipe and Filter)

- Provides a solution to move an output from one system to another.
- Pipes are the connection between the source system and the receiving system (sink)
- Filters responsible of data transformation to the receiver for processing.
- This pattern useful for data transfer from one system to another, into different formats. i.e. converting data from local university storage format into National Student Registry format.
- Further reading: Microsoft Integration Patterns, patterns and practice 2005, CH6



Additional Patterns (Gateway)

– Abstracts the access to an external systems to a single interface, by eliminating the need for multiple systems.
– Simplifies the development and maintenance processes that are related to accessing external systems.
– Common uses accessing mainframe programs and processing credit card transactions. Pattern replaces direct access to resources.
– Further reading: Microsoft Integration Patterns, patterns and practice 2005, CH6



Why Web Services?

- Web service benefit:
  - Loose-coupling
  - Service oriented
  - Reliable communication
  - Vender independent
  - Bypass firewalls (HTTP/HTTPS)

The Palestinian e-government academy

SOA Patterns

Categories:

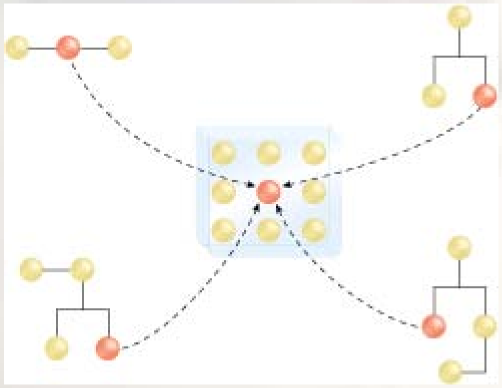
- Foundational Inventory Patterns
- Logical Inventory Layer Patterns
- Inventory Centralization Patterns
- Inventory Implementation Patterns
- Inventory Governance Patterns
- Service Implementation Patterns
- Service Security Patterns
- Service Messaging Patterns

SOA Patterns: Logic Centralization

How can the misuse of redundant service logic be avoided?

Problem:

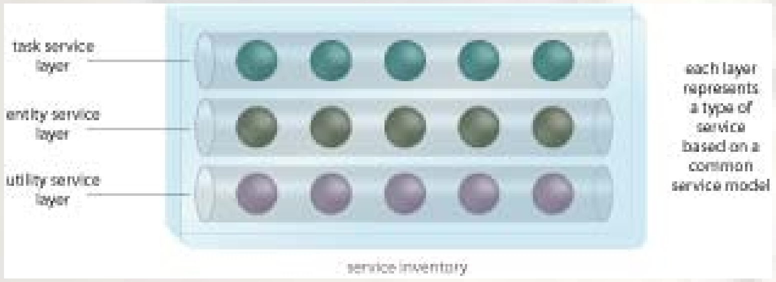
If National student services are not consistently reused, redundant functionality can be delivered in other services, resulting in problems associated with inventory denormalization and service ownership and governance.



SOA Patterns: Service Layers

How can group of services be organized based on functional commonality?

Problem:
Arbitrarily services delivered and governed by different teams can lead to design inconsistency and inadvertent functional redundancy.



SOA Patterns: Composition Autonomy

How can compositions be implemented to minimize loss of autonomy?

Problem:
Composition controller services naturally lose autonomy when delegating processing tasks to composed services, some of which may be shared across multiple compositions.

Summary

- Systems become more complex
- Patterns help us to design robust applications
- Interoperability and integration is key to build enterprise software
- Next session will cover the business process execution language

The Palestinian e-government academy

References

1. http://eaipatterns.com/gregor.html
2. http://en.wikipedia.org/wiki/Creational_pattern
3. http://en.wikipedia.org/wiki/Structural_pattern
4. http://en.wikipedia.org/wiki/Behavioral_pattern
5. http://dictionary.reference.com/browse/Integration
6. http://en.wikipedia.org/wiki/Systems_integration
7. http://www.designpatternsfor.net/Presentations/AsynchronousMessagingPatternsWithWCF.pdf
8. http://drops.dagstuhl.de/volltexte/2006/828/pdf/06291.SWM.Paper.828.pdf
9. http://www.soapatterns.org/orchestration.php
10. Enterprise Integration Patterns, Gregor Hohpe, Bobby Woolf Addison-Wesley, 2004
11. Patterns of Enterprise Application Architecture, Martin Fowler Addison-Wesley, 2003 Enterprise SOA, Dirk Krafzig, Karl Banke, Dirk SlamaPrentice Hall, 2004
12. Integration Patterns 3ed, Microsoft, 2004

Thanks

Mohammed Melhem