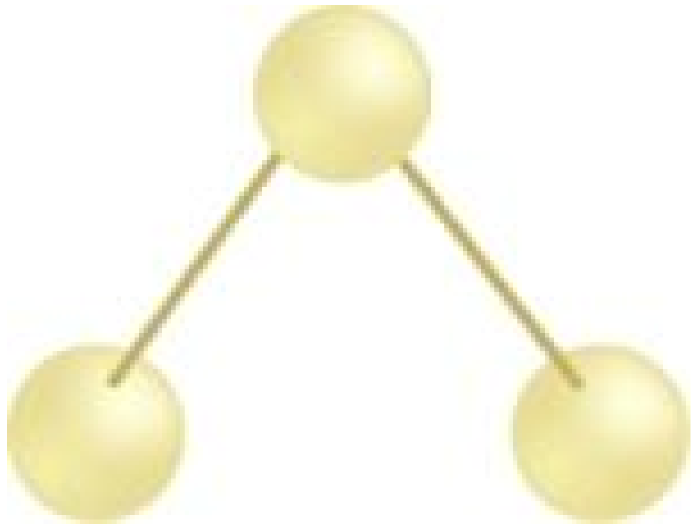
SERVICE COMPOSITION



WHAT ???

- A *service composition* is an aggregate of services collectively composed to automate a particular task or business process.
- To qualify as a composition, at least two participating services plus one composition initiator need to be present.
- Otherwise, the service interaction only represents a point-to-point exchange.

- These composite services can be in turn recursively composed with other services into higher level solutions, and so on.
- Such recursive composition of business services is one of the most important features of SOA, allowing to rapidly build new solutions based on the existing business services.

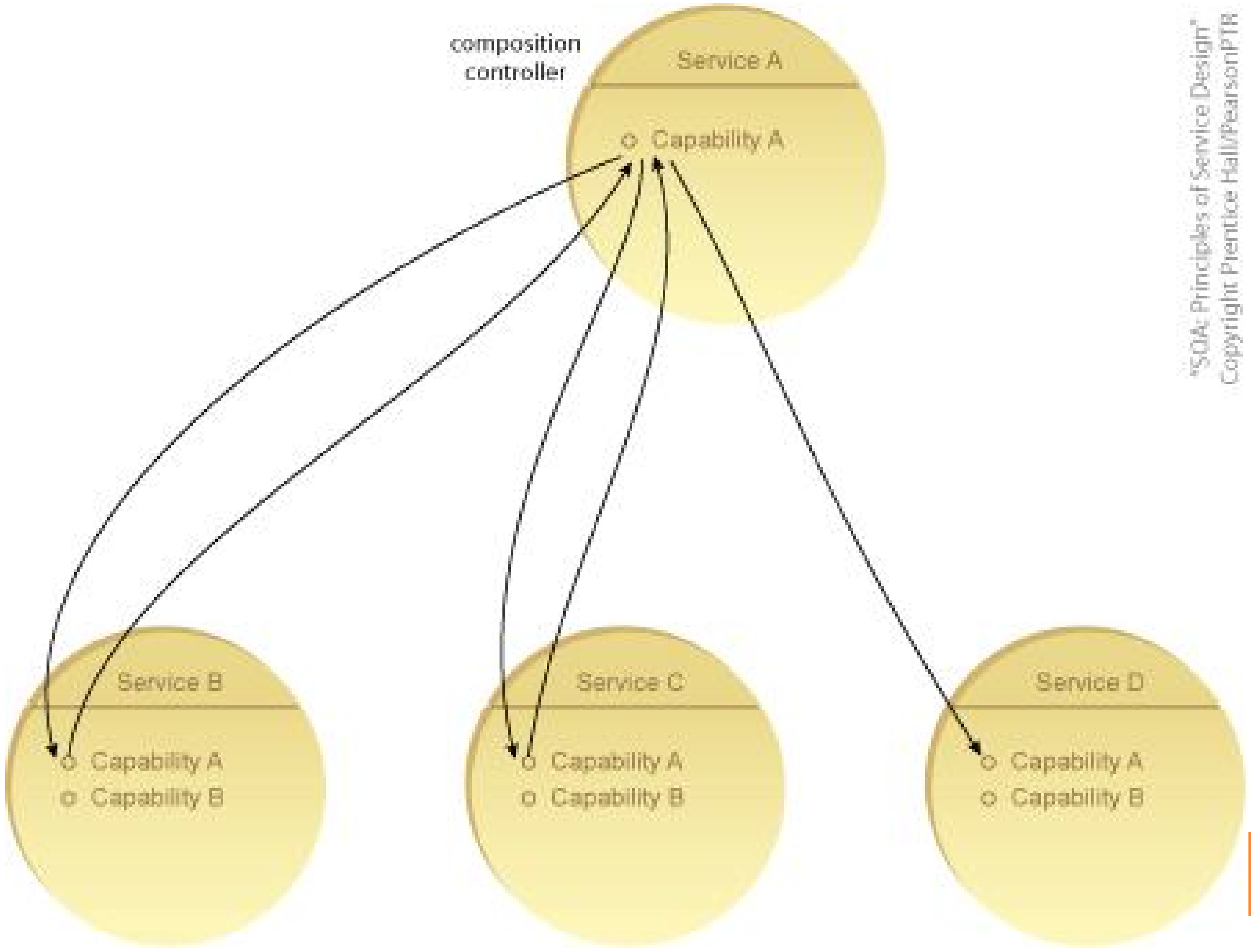
MOTIVATION

- Usage simplicity
- Improved reusability
- Solution partitioning, visibility, control and change management

COMPONENTS

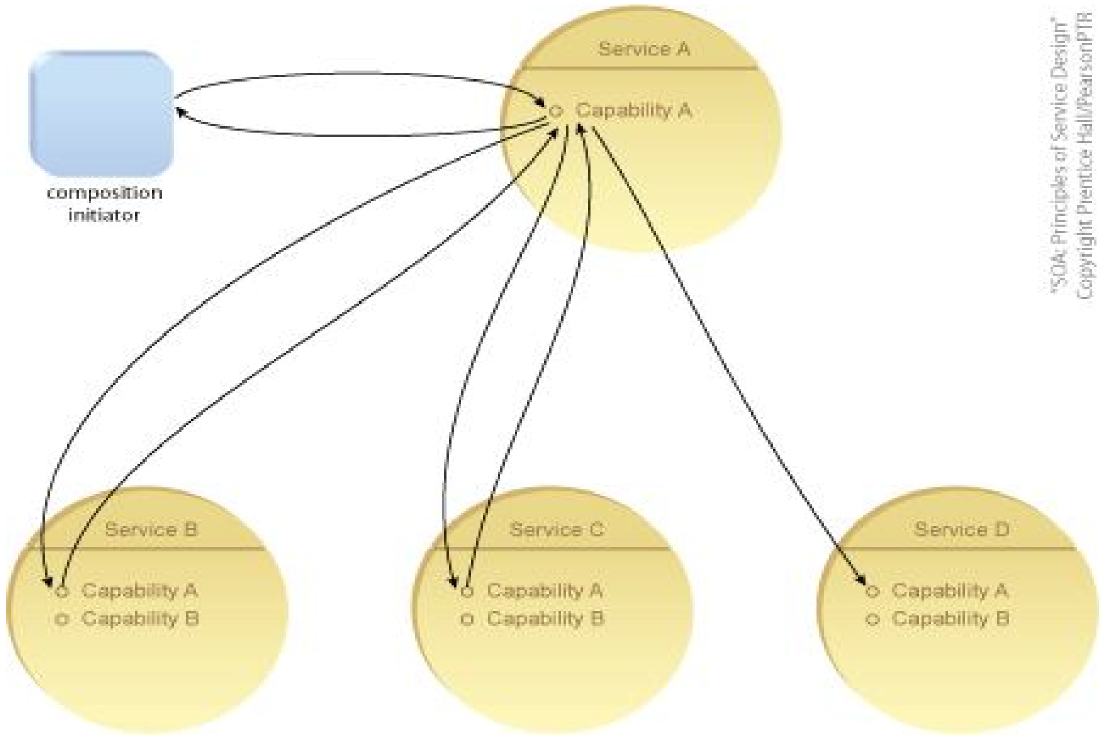
- COMPOSITION CONTROLLER

  - Within a runtime service activity being executed by a composition of services, the composition controller (often just referred to as "controller") represents a service with a capability that is executing the parent composition logic required to compose capabilities within other services.


composition controller
Service A
Capability A
Service B
Capability A
Capability B
Service C
Capability A
Capability B
Service D
Capability A
Capability B

COMPOSITION INITIATOR

- Often a separate service consumer program exists that acts as the initial sender of a message path by sending a command or input values to a composition controller.

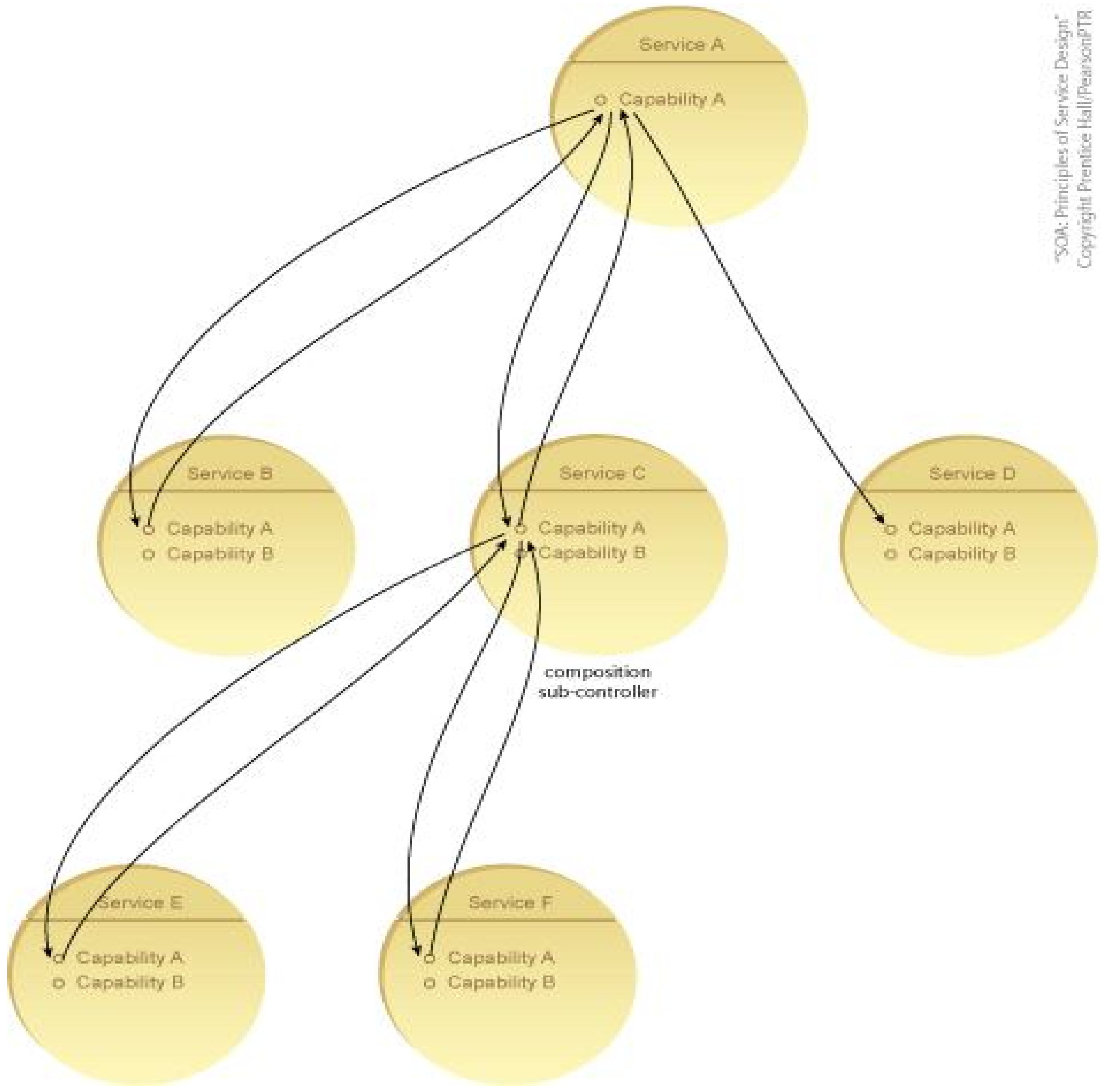


COMPOSITION MEMBER

- a service that participates in a service composition by being composed by another service.
- Service Composability design principle emphasizes the need for services to be designed as effective composition members, regardless of whether they need to be initially positioned within a composition.
- This is fundamental to the long-term goal of establishing a service inventory from which agnostic services can be repeatedly repurposed into multiple service compositions capable of fulfilling new and changing business requirements.

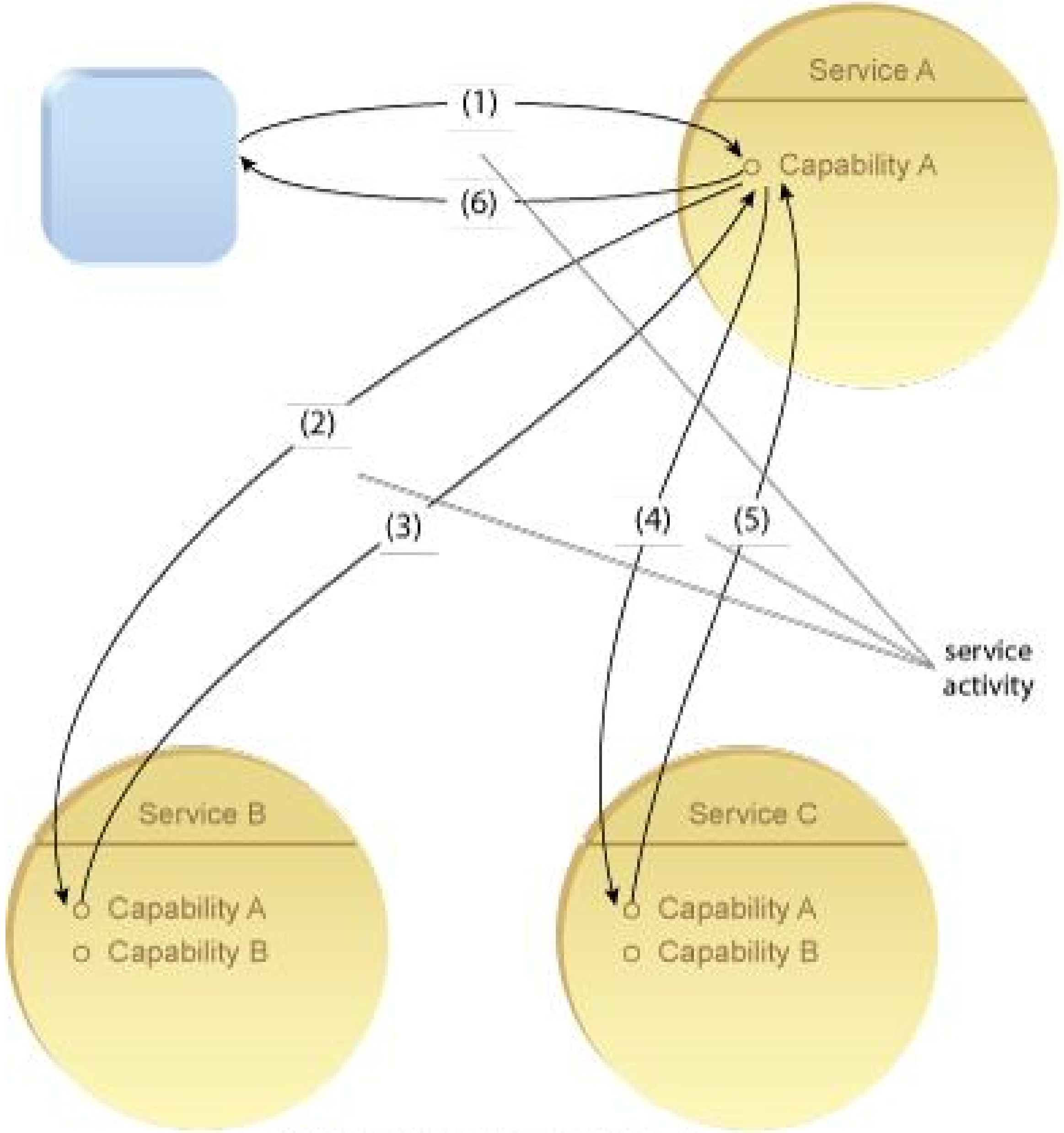
COMPOSITION SUB CONTROLLER

- variation of the composition controller role that represents nested composition logic.
- Whereas a regular composition controller is at the top of a typical composition hierarchy, a sub-controller generally contains a capability that is composing other service capabilities while this capability itself is also being composed by the parent composition controller.
- sub-controller is a temporary runtime role assumed by a service when its capability is composed by another service.
- It is also considered a composition member, as it represents one of the composition participants composed by a parent controller.


Service A
Capability A
Service B
Capability A
Capability B
Service C
Capability A
Capability B
Service D
Capability A
Capability B
composition sub-controller
Service E
Capability A
Capability B
Service F
Capability A
Capability B

SERVICE ACTIVITY

- The chain of message exchanges carried out in support of the execution of a specific task or business process is referred to as a *service activity*.
- A primitive service activity generally maps to a single data exchange, much like a point-to-point interaction.
- A complex service activity is usually associated with the message exchanges that occur across a composition of services.



"SOA: Principles of Service Design"
Copyright Prentice Hall/PearsonPTR

ASPECTS

- COMPOSITION DESIGN
  - concerned with synthesizing a specification of how to coordinate the component services to fulfill the client request.

- COMPOSITION IMPLEMENTATION
  - concerned with how to actually achieve the coordination among services, by executing the specification produced by the composition design.

WS-BPEL

- Stands for Web Services Business Process Execution Language.
- XML based language(defined by grammar) enabling users to describe business process activities as Web Services and define how they can be connected to accomplish specific tasks

- An *executable business process*: models an actual behaviour of a participant in a business interaction.

- An *abstract business process*: is a partially specified process that is not intended to be executed, may hide some of the required concrete operational details.

- WS-BPEL aims to model the behaviour of processes, via a language for the specification of both Executable and Abstract Business Processes. By doing so, it extends the Web Services interaction model and enables it to support business transactions.

TERMINOLOGY

- Activities:
  - Message exchange or intermediate result transformation
- Process:
  - The composition result, consists of a set of activities

THE STRUCTURE OF WS-BPEL PROCESS

- A WS-BPEL process definition is represented at runtime by the process service

- Services that participate in the WS-BPEL defined processes are considered as partner services and are established as a part of the process definition.

- Numerous activity elements are provided to implement various types of process logic.

WS BPEL PROCESS DEFINITION

```
<process>

    <partnerLinks>
        .....
    </partnerLinks>
    <variables>
        ....
    </variables>
    <faultHandlers>
        ....
    </faultHandlers>
    <sequence>
        <receive....>
        <invoke.....>
        <reply........>
        .....
    </sequence>
   .......

 </process>
```

- **The process element**

  <process> element: root element and must have a name attribute for assigning the name value. It is used to establish the process definition-related namespaces.

- **partnerLink and partnerLinks element**

  partnerLinks define the services that are orchestrated by process. It contains a set of <partnerLink> elements each of which represent the communication link between the two partners.

  the partnerLink element contains attributes:

  1. myRole
  2. partnerRole

- <variables element>
  - Hold the data that constitute the state of a BPEL business process during runtime.
  - Attributes:
    - Message type: allow for the variable to contain an entire WSDL message
    - Element: refer to an xsd element construct
    - Type: used to just represent an XSD simpleType, such as a string or integer.

WS-BPEL FUNCTIONS

- getVariableProperty(variable name, property name)
  - Retrieve global property values from variables.

- getVariableData(variable name, port name, location path)
  - Has a mandatory variable name parameter and two optional parameters to specify a part of the variable data.

BASIC ACTIVITIES

- Invoke element
  - <invoke> activity is used to invoke the ws operations provided by the partners.
- Receive element
  - <receive> activity is used to receive input requests in a BPEL business process to provide services to its partners. The process blocks until the message is received.
- Reply element
  - <reply> is used to send a response to a request previously accepted. They are used to synchronous request-reply interactions.

STRUCTURED ACTIVITIES

- <sequence> element
  - Used to organize a series of activities so that they are executed in a predefined, sequential order. Allows for nesting.
  - Eg.

```
<sequence>
    <receive>........</receive>
    <assign>..........</assign>
    <invoke>.........</invoke>
    <reply>...........</reply>
 </sequence>
```

- Switch case
- Flow
- Pick
- If/else
- Scope
- Assign, copy-from & to

THE FAULTHANDLERS, CATCH AND CATCHALL

- React to faults while executing business process activities.

- <catch> activity
  - Used to specify faults that are to be caught and handled
- <catchAll> activity
  - Used to catch all faults. It is optional.

- Syntax:

```
    <faultHandlers>
       <catch faultName="SomethingBadHappened"
                  faultVariable="TimesheetFault">
       ......
       </catch>
       <catchAll>
                  activity
       </catchAll>
 </faultHandlers>
```

CompensationHandler element

- Used to define compensation activities: gather all activities that have to be carried out to compensate another activity.

- Syntax:

```
        <compensationHandler>
                activity
        </compensationHandler>
```

- **<empty>** element: an activity that does nothing.
- **<wait>** element: specify a delay for a certain amount of time or wait until a certain deadline is reached.