Chapter 3

Event-driven Service-Oriented Architecture

Event-driven Service-oriented Architecture (EDSOA) is a type of SOA where services have the capability to react to events. Events can be in the form of beginning or end of execution of services, changes in the value of critical variables or exceptions generated at the run time. An event is a change in the state of a service, start or end of an activity. Event is a primary entity in Event-driven Architecture (EDA) and it depends on time and location. Business processes are event-driven and requires a set of tasks to be executed to fulfill the goal. It requires composition and execution of services as per the events. Events are responsible to change the state of a process. We can model the business process based on event and state. EDSOA defines event-driven business process model by representing process as conceptual process flows. Process is represented as a set of services and completion of each service as an event which is responsible to trigger next service. Each service taking part in the execution of a process is a stateful service and supports event and notification. To monitor the event on services, properties and life cycle of services should be defined. Property describes the parameter and state and it is responsible for state transition of a process. Lifecycle is responsible to define the start, end and termination time of a service, which is required to maintain and control the execution of a process.

In this chapter the life cycle of EDSOA is presented in section 3.1. In section 3.2 EDSOA is proposed. Section 3.3 briefly discusses the Agro-produce marketing business process use case scenario. We have considered Agro-produce marketing as a proof of concept and prototype implementation to demonstrate the proposed approach. Section 3.5 shows the realization of proposed architecture based on the use case scenario.

3.1 Life Cycle of Event-driven Service-oriented Architecture

Business processes are represented as composition of services. Business processes are dynamic in nature due to changes in policies, rules, partners and events. There is a need for a formal
framework and event-driven service-oriented architecture to model, compose and execute the
dynamic business processes.

Event-driven business process developed using EDSOA follows the following EDSOA
development lifecycle.

**Model:** The model phase includes analysis and design of business processes, services, events and
messages. In model phase, actual services are modeled which can perform single business
function and take part in composition. Ontology should be defined to provide common domain
vocabulary, knowledge and business service templates, which we can leverage during the Model
phase.

**Compose:** In the composition phase, one needs to create or reuse existing services and combine
it to create composite services. Business rules, business policy, and business metadata should be
defined at composition phase. It will describe the control and data flow, mapping input, output of
services, which are required for generating dynamic composition schema.

**Deploy:** The deploy phase involves activities like creating environment for deployment,
registering endpoints, deploying the business services, provisioning business services to
organizations and subscribers. Components and services are to be deployed in an integrated way
so they can participate and get executed in a proper sequence at the time of execution of a
composite business process.

**Execution and Monitoring:** The execution and monitoring phase involves managing,
entitlements and subscriptions of the business services to organizations and users. Event and
state provides business context visibility for composite services. For example, you can determine
how a business service is utilized on various channels for different consumers, as well as
determining how different endpoints are performing for a specific business context along with
which services are in execution and in which state.
3.2 Proposed Event-driven Service-oriented Architecture

Event-driven Service-oriented architecture is proposed to provide state, transaction management, notification, execution and monitoring, and scalability. The role of the targeted system is to provide dynamic composition of a business process with interoperable integration of scattered services and resources. Our architecture is domain and tool independent. The proposal is mainly dependent upon existing grid computing, Semantic web and WS-* standards and specifications to achieve compliance. Our architecture presented in figure-15, contains various components to decompose the parts of an enterprise application based on the principles of SOA.

The architecture shown in figure-15 comprises of seven components:
3.2.1 Information Providers

In this experiment, Web Services acts as data service providers, maintained and published by respective organizations. These services provide access to different data sources with different mechanisms such as JDBC connection, stored procedure etc. It also provides access to existing applications or legacy applications, by using wrapper and exposing them as services. In this way they are allowing reuse of existing applications.

3.2.2 Event Manager

This component is the main controlling component of the architecture. Event manager is designed to communicate dynamically with the distributed heterogeneous services and other components such as ontology server, rules engine and composition engine. It is configured to furnish precise and event specific composition. It helps in the selection of services for composition. When any event is received by event manager, it will check the precondition of services, required input, and output parameters. Event manager uses backward chain algorithm for verifying above mentioned parameters. It works in coordination with the composition engine to generate the composition schema at runtime.
3.2.3 Composition Engine
Composition engine controls and manages the composition of services and allows orchestration and automated execution of a business process. It contains the composition logic and uses business rules to manage the flow of a business process. It works with the event manager and uses backward chain algorithm for checking the precondition, input and output of services. Forward chain algorithm is used with ECA rules to get the next sequence of services. According to events, composition engine generates composition schema at runtime based on events. The composition engine will be based on BPEL and will be executed by execution engine.

3.2.4 Ontology Server
Ontology server contains two levels of ontology, one provides the knowledge about application domain and another provides the knowledge about business processes and services involved in an organization. Ontology represents knowledge about the domain in terms of semantic graph, where concepts are represented as a node of the graph and relationships are represented as arch among the nodes.

To enable composition of business process, one of the important challenges is the utilization of uniform terms across the process. We have to cover all the terms, relations among them and the logical expression of the legal provision of business process. Functions and rules can be defined to capture the behavior, properties, and constraint of the concepts [89].

3.2.5 Rules Engine
Rules enhance the functionality of application to be developed based on the principle of Semantic web. It provides behavioural knowledge, constraints expression, and reaction to events. It defines flow and constraints of a business process and controls the behavior of a business process. It is also used to model the process, to define the policy, to handle the exception and to define the plan of a process [90]. There are four categories of rules, but I have considered mainly ECA rules and inferences rules. Rules engine contains ECA rules, which are triggered based on occurrence of an event. To achieve interoperability, ontology server will verify events and conditions, and corresponding to that an action will be taken. An action contains a set of
tasks to be executed and required to compose the BPEL schema [46]. Inference rules are of if-then type and used to represent facts and required conditions.

3.2.6 Execution Engine
Execution and management of distributed and heterogeneous services is more complex than traditional enterprise applications. Execution engine utilizes grid middleware services and working as an integrated part of grid environment. Schema generated by composition engine will be executed at centralized execution engine. It is supported by grid middleware services for monitoring, event management, automation, orchestration, security, performance management, transaction management and notification services. It will take care of execution and monitoring of a business process. Any exception or final output will be forwarded to the event manager; from there it will be sent to the user. For the execution of BPEL schema in a grid environment two types of grid services are developed: business services and grid services.

3.2.7 Business Services
Business services implement the core business functionality and participate in the composition and execution of a business process. These services are modeled at the business process design time. It represents the decomposed business logic of a process and performs individual business function and combines to fulfill the requirement of a process. These services are crucial services and require support for notification, state and transaction management. These services need to be implemented as grid services and exposed as stateful resources by following WSRF specifications [45]. Notification is achieved by following WSN specifications. These services are part of grid environment to get the benefits of grid like security, execution, monitoring, resource sharing and scalability. These services are interacting with other external services to get the required information and knowledge for the execution of a business process.

3.2.8 Grid Services
These services provide resource management, virtualization capability, security, scalability, management, performance etc. It is deployed to manage the execution and control the behavior of grid. Inbuilt grid services like, service correlation, service isolation, self-discovery,
virtualization, dynamic provisioning, resource pooling and persistent storage are utilized to fulfill the functionalities of middleware [53]. It offers deployment infrastructure services to manage hardware, software, and resources.

3.3 Motivating Scenario

As a proof of concept, I have taken an example of Agro-Produce Marketing Process in India. The Model Act [91] is expected to bring reforms in the Agro-produce Marketing Process. As indicated in the Model Act, a typical trade can span across the markets located at various places. I have considered a trading scenario to sell agro-produce across the markets. A seller (or farmer) joins the market place and expresses his intention to start a trade of agricultural produce. An authorized market functionary carries out measurement and grading of the produce. Seller needs to pay transportation fees in case vehicle is used to transport an agro-produce. Price of the produce can be set by tender bid, auction or any other transparent system. In case of direct sale, a seller is exempted to pay market fee or commission; whereas in case of indirect sale the market fee is imposed on the seller for using different utility services. Only license holders are allowed to carry out trade in the market area. If they fail to pay fees to the Agro-Produce Marketing Committee (APMC) or fail to pay the agreed-upon price of the purchased good, the APMC may cancel the license of the trader. If the trade is carried out by license holders in a manner explained above, the bill will be issued and the transaction will be recorded in the APMC database. Figure-16 represents a simplified workflow capturing few aspects of a typical trade. A framework is also evaluated by applying to another use case scenario of agricultural recommendation system [47, 52].
3.4 Realization of Event-driven Service-oriented Architecture

The business logic adopted for this experiment follows the proposed legal framework for marketing of agricultural produce. Business processes like product discovery, negotiation, agreement, validation, payment, and transaction management are covered for implementation. We have applied these convergence and architecture on Agricultural Marketing Process and with this approach, we are expected to achieve following objectives:

- Enabling Direct and Indirect Marketing for farmers to sell agro-produce.
- Enabling effective product discovery for buyers.
- Enabling simultaneous composition and negotiations at different markets.
- Implementation of transparent business process and transaction management for APMCs.
- Providing effective implementation framework and architecture to realize the proposed legislation for the government.
- Enabling decision support for traders by providing access to services mapping to actual business processes.
Here events are generating from different sources such as trader, buyer, seller and different APMCs. Events generally reflect a change in an agricultural marketing business process and also trigger other event. Events are fine grained and represent data, message, location or context. For e.g. farmer located in the city of Visnagar wants to sell Mango in Ahmedabad APMC. EDSOA captures these diverse events dynamically from multiple sources. It associates this information using declarative rules and taking corresponding action based on conditions. EDSOA reduces the complexity using dynamic modeling and composition when information is coming in real-time and in parallel from diverse sources. SOA can address reliable sequential process, as it does not allow multiple events to execute an action and to attack above requirements with traditional service architecture. EDSOA and SOA can work together to achieve dynamic modeling, composition and execution of business processes by capturing event and correlating them using rules.

Different components of EDSOA are described below:

### 3.4.1 Information Providers

Agro-market and Agro-business processes are collection of services of APMC, like receive trade offer, calculate price, get current market price, market mediation, price calculation, offer price, pay price, issue bill and so on [49]. Thus, this component is planned as a collection of web-enabled Agribusiness services that can serve as information and service providers. These services help in storing and retrieving information from APMC database and external applications.

### 3.4.2 Event Manager

Event manager maintains the farmer subscription by retrieving his preference from the APMC database. When a farmer put forward any buy or sell request, event manager will treat as event and start the composition process. It will interact with ontology server, rule engine, composition engine, and based on the farmer marketing policy and rules it will generate the composition schema. It also generates recommendation to the farmer by maintaining subscription, preferences and with event correlation [47, 52].
3.4.3 Composition Engine
When farmer starts the Agro-produce marketing process, event manager interacts with composition engine and orchestrates the services. Composition engine follows the Agro marketing business rules and starts the backward and forward chain algorithms to manage the sequence of services. It generates the Agro marketing business process composition schema as per the event generated during the execution of a business process.

3.4.4 Ontology Server
Ontology service contains the agriculture domain ontology. Agricultural ontology provides common vocabulary about agricultural and marketing of agricultural produce. AGROVOC [92] is used as a base vocabulary to build Agricultural ontology. We have tried to cover all the terms, relations among them and the logical expression of the legal provision. The ontology serves as a building block to an agriculture information system. It answers the queries of farmers, helps to make decisions about the crop production, helps to generate the recommendation and helps in discovery, selection and composition of agro marketing services.

3.4.5 Rules Engine
Rules engine is the rules repository, which contains information in the form of rules. Rules are added by the agricultural experts. Agricultural Experts are agriculture scientists and APMC authorities, who add agriculture related and marketing related rules. An example of a rule is “If sale is direct, then seller is exempted to pay market fee”. This rule will be triggered against corresponding event like “A seller enters the market place” or “A farmer starts the trade”.

3.4.6 Execution Engine
Execution and management of distributed and heterogeneous services is more complex than traditional enterprise applications. Execution engine utilizes the grid middleware services and working as an integrated part of grid environment. Schema generated by composition engine will be executed by centralized execution engine. It is supported by grid middleware services for monitoring, event management, automation, orchestration, security, performance management, transaction management and notification services. It will take care of execution and monitoring
of a business process. Any exception or final output will be forwarded to the event manager; from there it will be sent to the user.

### 3.4.7 Business Services

In Agro Marketing business process, core business functionalities are implemented as business services such as TradeOfferGridService, PayPriceService, NetCostComputationService. These services are developed by following WSRF, WSN and grid computing standards. Thus, they have support for state, notification, resource sharing and execution monitoring. These services work with Agricultural information provider services to get and store information about farmers and crops.

### 3.4.8 Grid Services

Various grid components and APIs are used to develop services which provide security, data management, resource management, virtualization and execution monitoring facilities. Grid Security Infrastructure (GSI) and Certification Authority (CA) are used to control the access of services and resources. OGSA-DAI (Data Access and Integration) are used to access and integrate datasets in a grid environment. Grid Resource Allocation and Management (GRAM) helped in execution management. Various other services are also used to achieve different functionalities such as index service is used to aggregate the resources.

The proposal for Event-driven Service-oriented Architecture, its lifecycle and realization of architecture based on agro-produce marketing use case scenario is discussed in this chapter. In the rest of the thesis, phases of EDSOA lifecycle are discussed in subsequent chapters. Chapter 4 discusses modeling and composition of event-driven business process. Chapter 5 discusses research experiments based on the development of dynamic business process with state, notification, service grouping and policy support. Chapter 6 describes the execution of direct and indirect agro-produce marketing process with notification support.