QuickRDA

RDA Domain Language

Erik Eidt

Version 2.7

March, 2011

***Copyright (c) 2011-2012 Hewlett-Packard Development Company, L.P.***

[Introduction 5](#_Toc336066988)

[An Alternative Paradigm 6](#_Toc336066989)

[Understanding Layering 6](#_Toc336066990)

[Architectural Description Layers 7](#_Toc336066991)

[Mapping Layers 7](#_Toc336066992)

[Business Contextual Architecture 8](#_Toc336066993)

[The Concepts of Business Contextual Architecture 9](#_Toc336066994)

[The Concepts of Business Contextual Architecture 10](#_Toc336066995)

[Conceptual Service Architecture 12](#_Toc336066996)

[The Concepts of Conceptual Service Architecture 12](#_Toc336066997)

[The Relationships of Conceptual Service Architecture 14](#_Toc336066998)

[Contextual to Conceptual Mapping 15](#_Toc336066999)

[References 16](#_Toc336067000)

[QuickRDA: Introduction & Overview 16](#_Toc336067001)

[QuickRDA: Best Practices Modeling Guide 16](#_Toc336067002)

***Table of Tables***

[Table 1. The Concepts in Business Contextual Architecture 9](#_Toc336067003)

[Table 2. The Relationships in Business Contextual Architecture 10](#_Toc336067004)

[Table 3. Concepts in Conceptual Service Architecture 12](#_Toc336067005)

***Table of Figures***

*No table of figures entries found.*

# Introduction

This guide formally defines the abstractions of Role-based Domain Architecture, RDA.

Role-based Domain Architecture is named after the top layer, which captures architecture at the highest level of abstraction using roles and responsibilities. As the name suggests, RDA is specifically oriented toward capturing domain architecture — the big picture of a domain covering the ecosystem some business or organizations find themselves in; it is meant to be used in a larger context of architectural architecture and design complementing RDA’s domain models by providing IT platform architecture, addressing implementation, configuration, deployment, and operational concerns. RDA’s lowest layers, Logical Designs and Physical Technologies, bridge some of the gap with IT platform architecture, though its primary intent is as domain specification for implementation.

Familiarity is encouraged with the QuickRDA: Introduction & Overview, which provides background on the layering methodology.

This document describes the RDA Domain Model Definition, which is also known as the RDA Domain Metamodel. Using the definition or metamodel, domain models can be captured.

## An Alternative Paradigm

For those familiar with process-, sequence-, and invocation-based models and methodologies, it should be pointed out that none of the RDA abstractions imply order, sequencing, or steps.

Interactions between modeled elements are captured though specific relationships, such as responsibilities in the Contextual layer, and service dependencies in the Conceptual layer not through transfer of control flow or invocations, not through timings or sequence ordering.

## Understanding Layering

Each layer of RDA has a particular focus — each layer is an abstraction that enables reasoning over the whole architecture at a consistent level.

Understanding any given layer is as much about understanding what’s in that layer as it is about understanding what’s not in the layer, and, to understand what’s not in a layer we also need to understand concerns are the purpose of the other layers.

Those who have read the QuickRDA: Introduction & Overview will recall that we informally defined the layers as having abstractions that concern themselves with *entities* interacting according to connecting *relationships* in which they share *content*. For example, the Contextual layer provides abstractions that concerns themselves with *roles* interacting with each fulfilling *responsibilities* manifesting in exchanges of *business artifacts*, and, the Conceptual layer provides abstractions that concern themselves with individually replaceable conceptual *services* each concentrating on their unique value and *depending upon* each other in a prescriptive architecture of *document* exchanges.

Here we’ll give the specific abstractions of each layer formal definitions in terms of concepts and relationships that are available for the modeler to use.

RDA defines two kinds of layers: Architectural Description Layers, and Mapping Layers. Each Architectural Description Layer focuses its own set of concerns for which it defines its own particular set of abstractions, both concepts and relationships. Mapping Layers provide the necessary overlap that connects the layers of architectural description into a single architecture, generally defining only relationships that map between concepts (and relationships) defined in the adjacent layers.

The next document, QuickRDA: Best Practices Modeling Guide discusses experiences in applications of the methodology.

For each layer, this document describes the abstractions in terms of available:

* Concepts, and,
* Relationships

The two kinds of layers — Architectural Description Layers and Mapping Layers — are discussed in turn.

## Architectural Description Layers

The Architectural Description Layers are the:

* Business Contextual Architecture
* Conceptual Service Architecture
* Logical Design Architecture
* Physical Technologies Architecture

Below which comes:

* Service Implementation (code & data)

## Mapping Layers

Mapping layers connect concepts and relationships in an upper layer to concepts and relationships in adjacent lower layer. Mapping layers introduce relationships (of refinement) but otherwise don’t introduce new concepts: they target existing concepts from the two adjacent layers.

# Business Contextual Architecture

As its name implies, the contextual architecture is about capture of business context. The business context architecture has multiple potential uses and audiences; some of the concepts and relationships are applicable across all of these uses; others are specific to one or the other.

One feature of RDA is furthering the development of a Service-Oriented Architecture (SOA) by creating isolation between the various layers of RDA (business contextual architecture, conceptual service architecture, logical design architecture, and physical technologies architecture), this isolation allows for the service interfaces to be defined cleanly, minimizing unnecessary interdependence that makes software solutions brittle and hard to support over time.

Another is organizational design function, Org Design, in which the goal is defining, maintaining, and rolling out of roles and responsibilities, or simulation or execution of them. Activities of organizational design involve defining roles & responsibilities in planning new organizations, as well as simulation and execution of an organizational design.

### The Concepts of Business Contextual Architecture

The concepts used to capture business context are:

|  |  |  |  |
| --- | --- | --- | --- |
| ***Concept*** | ***Function*** | ***Definition*** | ***Intended Usages*** |
| Role | *Interacting Entities* | Abstract or Concrete Business Entity | SOA & Org Design |
| Responsibility | *Connecting Relationships* | Expectations by the neighboring Roles | SOA & Org Design |
| Artifact | *Shared Content* | Tangible Artifacts, Outcomes | SOA & Org Design |
| Plan | *Shared Content* | Placeholder for Roles & Responsibilities | Org Design, Differentiation in SOA |
| Actor | *Interacting Entities* | Concrete Parties that Play Roles | Org Design, Examples in SOA |

Table 1. The Concepts in Business Contextual Architecture

***Roles*** are business *entities* that interact by sharing *content*, namely ***Business Artifacts***, via connecting *relationships*, which are ***Responsibilities***.

Roles — in RDA we use the term Role to represent the generic notion that ranges from the very concrete (i.e. actors, e.g. business parties & corporations, groups, organizations, and individuals) to the very abstract (i.e. roles in opposition, e.g. provider, consumer) in an ecosystem. Roles can be composed from other roles, or they can represent simple abstract functions with responsibilities.

Responsibilities — are the expectations that the other parties in an ecosystem or the roles in small groups have of each other.

Artifacts — or business artifacts — are the ultimate manifestation of responsibilities. Business Artifacts can describe more tangible items such as widgets or money. They can describe business documents, such as tickets or trip/record locators, contracts, requirements, etc… They can also describe outcomes, results or business conditions.

Plan — A Plan is a kind of Artifact. Sometimes the same abstract role needs to occur in different settings of one domain architecture: a good example is a middleman situation, where the middleman plays opposing roles each in different networks. Plans are used in capturing these descriptions. Plans express the notion of blueprints — artifacts that represent assemblies of roles with their attendant responsibilities, i.e. plans represent domain models themselves, and, planning responsibilities can be described in terms of generating such blueprints, while execution responsibilities can be described in terms of consuming such blueprints, staffing the roles and managing the execution of the plan.

Actor — An Actor is a kind of Role. In order to simulate or execute a network of roles & responsibilities, concrete actors should be identified to play the roles. Also, sometimes identifying exemplary actors helps with the clarity of domain models even when not intended for direct execution.

RDA allows domain models to start abstract with generic roles, and lead to more concrete domain parties, or start with concrete domain parties and lead to more abstract roles. To facilitate this, actors are not only capable of playing roles (that have responsibilities and relationships with other actors) but can also be directly assigned responsibilities and have relationships with other actors.

### The Concepts of Business Contextual Architecture

The relationships are used to capture business context are:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Relationship*** | ***Subject Concept*** | ***Object Concept*** | ***Definition*** | ***Intended Usage*** |
| Plays | Role | Role | Role Composition | SOA |
| Plays | Actor | Role | Assignment of Actor to Role | Org Design |
| Example Of | Actor | Role | Sample | SOA |
| Member Of | Role | Role | Role Composition | Org Design |
| For | Plays | Role | Qualifies Role Play | Org Design |
| Of | Role | Role | Qualifies Role | Org Design |
| Mentors | Role | Role | Assistance Relationship | Org Design |
| Interacts With | Role | Role | Abstraction | Internal Usage by Tooling |
| Assigned To | Responsibility | Role | Assignment of Responsibility | SOA & Org Design |
| Has Parent | Responsibility | Responsibility | Responsibility decomposition | SOA & Org Design |
| About | Responsibility | Responsibility | One responsibilities can be about others | SOA & Org Design |
| Consumes | Responsibility | Artifact | Artifact consumption | SOA & Org Design |
| Provides | Responsibility | Artifact | Artifact providing | SOA & Org Design |
| Manages | Responsibility | Artifact | Manages the artifact | SOA & Org Design |
| Consumes With Input | Responsibility | Artifact | Artifact consumption, though with round-trip input from the consumer | SOA & Org Design |
| Provides With Input | Responsibility | Artifact | Artifact providing, though with round-trip input from the consumer | SOA & Org Design |
| Kind Of | Artifact | Artifact | Kind, type, or class | SOA & Org Design |
| Component Of | Artifact | Artifact | Artifact composition | SOA & Org Design |
| View on | Artifact | Artifact | A report of sorts on an artifact | SOA & Org Design |
| Reference To | Artifact | Artifact | Contains a component that is a reference | SOA & Org Design |
| Reply To | Artifact | Artifact | Messages that are related | SOA & Org Design |
| Informs | Artifact | Artifact | Artifacts that need to inform other artifacts | SOA & Org Design |
| List Of | Artifact | Artifact | Collection of artifacts | SOA & Org Design |
| Involves | Artifact | Artifact | Otherwise unspecified dependency | SOA & Org Design |
| One Of | Artifact | Artifact |  | SOA & Org Design |
| Same | Artifact | Artifact |  | SOA & Org Design |
| In Plan | Plays | Plan | Indicates the plan (network) that a role is played in | SOA & Org Design |

Table 2. The Relationships in Business Contextual Architecture

Plays — between roles, the Plays relationship allows role composition. The subject role incorporates the responsibilities of the object role. The incorporated responsibilities represent externally visible characterization of a subject role.

Plays — between an Actor and a role, the Plays relationship indicates identification of the concrete player.

Example Of — is used to link an actor identified for exemplary purposes to a role; these are considered as examples only, and otherwise without bearing on the domain model.

Member Of — between roles, the Member Of relationship allows role aggregation. The subject role incorporates the object role. Member Of is used for internal decomposition of a role.

# Conceptual Service Architecture

As its name implies, the contextual architecture is about capture of business context. The business context architecture has multiple potential uses and audiences; some of the concepts and relationships are applicable across all of these uses; others are specific to one or the other.

One is furthering the development of Service-Oriented Architecture, SOA, by creating a business contextual architecture, along with the other layers of RDA, conceptual service architecture, logical design architecture, and physical technologies architecture

Another is organizational design function, Org Design, in which the goal is defining, maintaining, and sharing of roles and responsibilities, or simulation or execution of them. Activities of organizational design involve defining roles & responsibilities in planning new organizations, as well as simulation and execution of an organizational design.

### The Concepts of Conceptual Service Architecture

The concepts used to capture conceptual services are:

|  |  |  |  |
| --- | --- | --- | --- |
| ***Concept*** | ***Function*** | ***Definition*** | ***Intended Usages*** |
| Service | *Interacting Entities* | The Conceptual Information Service | SOA & Org Design |
| Operation | *Interacting Entities* | Accessible point of a service | SOA & Org Design |
| Tower | *Interacting Entities* | Arrangements of Services | SOA & Org Design |
| Fundamental Objects | *Shared Content* | Persistence for Services & Towers | Org Design, Differentiation in SOA |
| User Interface | *Interacting Entities* | Non-service accessor of operations |  |
| Information Documents | *Shared Content* | Refinement of Artifact for this layer | Org Design, Examples in SOA |
| Message | *Shared Content* | Operation Request or Response package |  |

Table 3. Concepts in Conceptual Service Architecture

Service — this is the primary abstraction for the conceptual services layer. Contains operations, may be organized in towers; has dependencies on other services.

Tower — an organizational unit composed of services or other towers.

Operation — an accessible point for a service; often one of C, R, U, or D when the service has persistence; associated with input message and reply message.

Fundamental Object — this is the primary abstraction for persistence in the conceptual services layer. One tower is the authority of record (system of record) for each fundamental object.

Information Document — an electronic document that is a refinement of a contextual business artifact; often the payload of a message

Message — an artifact that is related to operation invocation

### The Relationships of Conceptual Service Architecture

The relationships used to capture business context are:

|  |  |  |  |
| --- | --- | --- | --- |
| ***Relationship*** | ***Subject Concept*** | ***Object Concept*** | ***Definition*** |
| Is In Service | Operation | Service | Links operation as being to service |
| Is In Tower | Service | Tower |  |
| Depends On | Service | Operation | Service to Service dependency |
| Is Owned By | Fundamental Object | Service | Ownership of Fundamental Object by Tower |
| Has Request | Operation | Message | Associates a Message with an Operation |
| Has Response | Operation | Message | Associates a Message with an Operation |

# Contextual to Conceptual Mapping

|  |  |  |  |
| --- | --- | --- | --- |
| ***Relationship*** | ***Subject Concept*** | ***Object Concept*** | ***Definition*** |
| Supports | Operation | Responsibility->Artifact | Refinement linking the providing or consuming of an artifact with an Operation |
| Implements | Service | Responsibility | Refinement linking of a Responsibility with a Service |
|  | Tower | Role | Refinement linking Role to Tower |
| Elaborates | Document | Artifact | Refinement linking Artifact to Document |

These properties establish linkage by refinement between subject concepts, which are in the conceptual model, and, object concepts, which are in the contextual model.

Is Supported By — describes a mapping between the contextual providing and consuming of artifacts and conceptual service operations. Thus, a responsibility providing an artifact may be supported by a read operation on a service.

Is Implemented By — describes a mapping between contextual responsibilities and conceptual services; the service is said to carry out the responsibility.

Is Responsible For — describes a mapping between a contextual role and a conceptual tower. The tower is run by this role and carrys out the role’s function.

Is Elaborated By — describes a mapping between contextual business artifacts and conceptual information documents. The document elaborates the artifact.

# References

##### QuickRDA: Introduction & Overview

##### QuickRDA: Best Practices Modeling Guide