

# *EDOC Vision*

(Extracted from the EDOC specification)

The vision of the EDOC Profile is to simplify the development of component based EDOC systems by means of a modeling framework, based on UML 1.4 and conforming to the OMG Model Driven Architecture ([www.omg.org/mda](http://www.omg.org/mda)) that provides:

6. A platform independent, recursive collaboration based modeling approach that can be used at different levels of granularity and different degrees of coupling, for both business and systems modeling and encompasses:
  - A loosely coupled, re-useable business collaboration architecture that can be leveraged by business-to-business (b2b) and business-to-customer (b2c) applications, as well as for enterprise application integration;
  - A business component architecture that provides interoperable business components and services, re-use and composability of components and re-use of designs and patterns, while being independent of choice of technology (e.g. component models), independent of choice of middleware (e.g. message services) and independent of choice of paradigms (e.g. synchronous or asynchronous interactions);
  - Modeling concepts for describing clearly the business processes and associated rules that the systems support, the application structure and use of infrastructure services, and the breakdown of the system into configurable components;
  - An architectural approach that allows the integration of “process models” and “information models”
7. A development approach that allows two-way traceability between the specification, implementation and operation of Enterprise computing systems and the business functions that they are designed to support;
8. Support for system evolution and the specification of collaboration between systems;
9. A notation that is accessible and coherent.

The vision addresses key business needs by enabling the development of tools that support:

10. Business collaborations as a central concern – covering alliances, outsourcing, supply chains, and internet commerce, and dealing with relationships that are in constant flux where what is inside the enterprise today is outside tomorrow, and vice versa;
11. Process engineering by assembling services – so that basic business functions can remain relatively constant while who performs them and in what sequence changes, and services themselves can become proactive;

12. The ability for parts of the enterprise to react quickly and reliably to change through:
  - shorter development time and improved quality of applications meeting market needs, improved interoperability between systems and support for distributed computing;
  - reduced lead time and improved quality resulting from the ability to generate a substantial portion of application code;
  - more robust specification by removing ambiguity and enabling more rigorous analysis of designs;
13. A new marketplace for interoperable collaboration based infrastructures and business components.

The EDOC Profile provides this modeling framework by defining:

14. A set of Profile Elements comprising:
  - a technology independent profile, the Enterprise Collaboration Architecture (ECA) allowing the definition of Platform Independent Models as defined by the MDA;
  - a Patterns Profile that can be applied in specifications that use the ECA;
  - a set of Technology specific Models allowing the definition of Platform Dependent Models as defined by the MDA.
15. A structure for the application of the Profile Elements in the specification of EDOC systems that conforms to the MDA.

This remainder of this chapter:

16. provides an overview of the Profile Elements (Section 2);
17. defines how the Profile Elements are applied in the specification of an EDOC system(Section 3)
18. defines the conventions for the definition of UML profiles used in the EDOC Profile (Section 4).

The ECA is fully defined in Chapter 3, the Patterns Profile in Chapter 4 and the Technology specific Models in Chapter 5. Non-normative mappings from the ECA to the Technology specific Models defined in Chapter 5 are described in Part II.

# *The EDOC Profile Elements*

## *The Enterprise Collaboration Architecture*

The Enterprise Collaboration Architecture (ECA) comprises a set of five UML profiles:

- the Component Collaboration Architecture (CCA) which details how the UML concepts of classes, collaborations and activity graphs can be used to model, at varying and mixed levels of granularity, the structure and behavior of the components that comprise a system;
- the Entities profile, which describes a set of UML extensions that may be used to model entity objects that are representations of concepts in the application problem domain and define them as composable components;
- the Events profile, which describes a set of UML extensions that may be used on their own, or in combination with the other EDOC elements, to model event driven systems;
- the Business Process profile, which specializes the CCA, and describes a set of UML extensions that may be used on their own, or in combination with the other EDOC elements, to model system behavior in the context of the business it supports;
- the Relationships profile, which describes the extensions to the UML core facilities to meet the need for rigorous relationship specification in general and in business modeling and software modeling in particular.

Each profile consists of a set of UML extensions that represent concepts needed to model specific aspects of EDOC systems. The concepts are described in terms of UML profiles.

The semantics of each profile (except for the Relationships Profile) are also expressed in a UML-independent MOF metamodel.

The ECA profiles are technology independent and are used together to define platform independent models of EDOC systems in conformance with the MDA. In particular, they enable the modeling of the concepts that until now have had to be specified programmatically in terms of the use of services such as events/ notifications, support for relationships and persistence.

## *Component Collaboration Architecture*

The Component Collaboration Architecture (CCA) details how the UML concepts of classes, collaborations and activity graphs can be used to model, at varying and mixed levels of granularity, the structure and behavior of the components that comprise a system. It defines an architecture of recursive decomposition and assembly of parts, which may be applied to many domains.

The term component is used here to designate a logical concept – a “part”, something that can be incorporated in a logical composition. It is referred to in the CCA as a Process Component. In many cases Process Components will correspond, and have a mapping, to physical components and/or deployment units in a particular technology.

A Process Component is a processing component: it collaborates with other Process Components within a CCA Composition, interacting with them through Ports, where Ports are an abstraction of interfaces of various types (e.g., synchronous, asynchronous). Process Components can be used to build other Process Components or to implement roles in a process – such as a vendor in a buy-sell process.

Process Components collaborate at a given level of specification collaborate and are themselves decomposed at the next lower level of specification. Thus the concepts of Process Component and Composition are interdependent.

The recursive decomposition of Process Components utilizes two constructs in parallel: Composition (using UML Collaboration) to show what Process Components must be assembled and how they are put together to achieve the goal, and Choreography (using UML Activity Graph) to show the flow of activities to achieve a goal. The CCA integrates these concepts of “what” and “when” at each level.

Since CCA, by its very nature, may be applied at many levels and the specification requirements at these various levels are not exactly the same, the CCA can be further specialized with profiles for each level using the same profile mechanisms. Thus Process Components exposed on the Internet will require features of security and distribution, while more local Process Components will only require a way to communicate, and there may be requirements for Process Components for specific purposes such as business-2-business e-commerce, enterprise application integration, distributed objects, real-time etc.

It is specifically intended that different kinds and granularities of Process Components at different levels will be joined by the recursive nature of the CCA. Thus Process Components describing a worldwide B2B business process can decompose into application level Process Components integrated across the enterprise and these can decompose into program level Process Components within a single system. However, this capability for recursive decomposition is not always required. Any Process Component may be implemented directly in the technology of choice without requiring decomposition into other Process Components.

## *Entities profile*

The Entities profile describes a set of UML extensions that may be used to model entity objects that are representations of concepts in the application problem domain and define them as composable components.

The goal is to define the entities with their attributes, relationships, operations, constraints and dependencies at a technology-independent level as components within system modeled using the CCA. The component determines the unit of distribution and interfaces that must be complemented by other components.

The profile includes declarative elements for placing constraints on the profile and for rules that will propagate the effects of changes and events.

The Entities profile is used with the Events and Business Process profiles to allow definition of the logic of automated business processes and of events that may be exchanged to achieve more loosely coupled integration. These three profiles together support the design of an EDOC system on the foundation provided by the CCA.

The Entities profile is used to define a representation of the business and operations that effect changes in state of the business model. Business processes modeled using the Events profile and the Business Process profile operate on this model where the process flow determines when operations should occur as a result of inputs from other systems, the occurrence of business events or the actions of human participants.

## *Events profile*

The Events profile describes a set of UML extensions that may be used on their own, or in combination with the other EDOC elements, to model event driven systems

An event driven system is a system in which actions result from business events. Whenever a business event happens anywhere in the enterprise, some person or thing, somewhere, may react to it by taking some action. Business rules determine what event leads to what action. Usually the action is a business activity that changes the state of one or more business entities. Any state change to a business entity may constitute a new business event, to which, in turn, some other person or thing, somewhere else, may react by taking some action. The purpose of the Event Profile is to define the use of the concepts in the CCA, Entity and Event profiles, and to extend them in order to support the design of event-driven business systems.

The main concepts in event driven business models are the business entity, business event, business process, business activity and business rule. So the basic building blocks are the business process and the business entity. The two are 'wired together' by a flow of actions from process to entity, and by a flow of events from entity to process. In a component framework, therefore, business processes have event inflow and action outflow, and entities have action inflow and event outflow.

This means that CCA business process components and CCA business entity components can be created by modeling

19. A business process as a set of rules of the type notification/condition/activity (This is the event-driven equivalent of the commonly known event/condition/action rule).
20. A business entity as set of operation/state/event causalities.

The connection from business process to business entity is a configurable mapping of activity to operation.

The connection from business entity to business process is a configurable set of subscriptions.

With these building blocks it is possible to model a number of event-based interactions. Furthermore, by reconfiguring the activity to operation mapping and/or the subscriptions, it is possible to re-engineer the business process and its execution in the system.

However, neither the business world, nor the computing world applies only one paradigm to their problem space. Businesses use a combination of loosely coupled and tightly coupled processes and computing solutions deploy a combination of loosely coupled and tightly coupled styles of communication and interaction between distributed components. Consequently, while the Events profile is defined to support the event-driven flavor of loosely coupled business and systems models, it allows such models to co-habit with more tightly coupled models.

### ***2.1.4 Business Process profile***

The Business Process profile specializes the CCA, and describes a set of UML extensions that may be used on their own, or in combination with the other EDOC elements, to model system behavior in the context of the business it supports

The Business Process profile provides modeling concepts that allow the description of business processes in terms of a composition of business activities, selection criteria for the entities that carry out these activities, and their communication and coordination. In particular, the Business Process profile provides the ability to express:

21. complex dependencies between individual business tasks (i.e. logical units of work) constituting a business process, as well as rich concurrency semantics;
22. representation of several business tasks at one level of abstraction as a single business task at a higher level of abstraction and precisely defining relationships between such tasks, covering activation and termination semantics for these tasks;
23. representation of iteration in business tasks;
24. various time expressions, such as duration of a task and support for expression of deadlines;
25. support for the detection of unexpected occurrences while performing business tasks that need to be acted upon, i.e. exceptional situations;
26. associations between the specifications of business tasks and business roles that perform these tasks and also those roles that are needed for task execution;
27. initiation of specific tasks in response to the occurrence of business events;
28. the exposure of actions that take place during a business process as business events.

## Relationships profile

The Relationships profile describes the extensions to the UML core facilities to meet the need for rigorous relationship specification in general and in business modeling and software modeling in particular

Relationships are fundamental to behavior because they are the paths over which actions occur, therefore clear, concise and rigorous specification of relationship semantics is of utmost importance. Furthermore, it should be noted that multiplicities are not the most important or most interesting properties of relationships. Property determinations are much more important for the semantics of a relationship, and distinguish among different kinds of relationships. The fragments of relationship invariants about property determination represent an essential fragment of those elusive “business rules” that are the backbone of a good specification and that should never be only “in the code.”

At the same time, it is very desirable to discover and specify – rather than reinvent – those kinds of relationships that are encountered in all specifications, so that reuse at the specification level becomes possible. Such generic relationships extend the set of reusable constructs that already exist in UML.

The Relationships profile defines generic relationships that provide concepts and constructs that permit UML to be used for specification of businesses and systems in a more rigorous manner than (and without restrictions currently imposed by) the base UML 1.4. Generic relationships provide for *explicit* specification of relationship *semantics* in class diagrams using invariants, in accordance with UML 1.4 Section 2.3.2: “The static semantics ... are defined as a set of invariants of an instance of the [association].... These invariants have to be satisfied for the construct to be meaningful.”

The approach presented is extensible, and if it appears that in a particular business (or a set of applications) additional generic relationships are needed and useful, then they may be precisely and explicitly defined and added in a manner similar to the definitions provided here.

The profile also provides advice for choosing and using a subset of UML for business modeling such a that the business models represented in terms of this subset will be readable and *understandable* by all stakeholders, specifically, business subject matter experts, analysts, and developers (as well as managers). The generic relationships described here are among the most important constructs of this subset.

## Patterns

A key element of the EDOC Profile design rationale is the ability to exploit the capability of patterns to capture modeling know-how or techniques and help developers to maintain efficiency and consistency in products. Patterns allow standard models to be reused to build good object models for EDOC systems

Many approaches to the use of patterns have been proposed, for example “Design Pattern” proposed by E.Gamma et.al, “Analysis Patterns” proposed by M. Fowler or “Catalysis Approach” proposed by D. D’Souza. In its use of

patterns the EDOC Profile focuses on improving sharability and reusability of object models rather than on assisting modeling efforts by illustrating good modeling techniques.

EDOC Patterns improve the sharability and reusability of models, by supporting the following features:

29. Models are made consistent with predefined normative modeling constructs, not only with modeling manners and notations.
30. Modeling constructs for common atomic objects, such as, Date, Currency, Country-code are predefined.
31. Common aggregated objects, such as Customer, Company, or Order, which represent business entities, are predefined as normative modeling constructs, using normative atomic objects.
32. Business concepts, such as Trade, Invoice, or Settlement, which are typically represented as relationships among objects, are defined as aggregations of the common elementary aggregated objects or simple objects, and are predefined as normative modeling constructs.
33. Aggregations that can be predefined using the more basic and elementary patterns as base, are defined as object patterns.
34. Patterns can represent business concepts where they provide for aggregation of more elementary patterns, thus aggregation or composition mechanisms are provided in patterns.
35. Business rules that govern a business concept are represented with a pattern with encapsulated constraints and a mechanism for constraint inheritance among patterns is provided.

A pattern is a set of types that can be instantiated to create an object model. Making a pattern from a set of object models requires identifying and defining the common types among those object models as their metamodel as in the ECA. Identifying and specifying many reusable business object patterns enables quick and high quality model development by selecting appropriate patterns to use in the project as a template.

The Patterns profile defines a standard means, Business Function Object Patterns (BFOP), for expressing object models using the UML package notation, together with the mechanisms for applying patterns that are required to describe models.

BFOP is a set of object patterns laid out in a hierarchical multi-layer structure, the Basic, Unit, Basic Model, Product (application systems) and Option layers. For example, **Error! Reference source not found.** illustrates how “Sales/Purchase Pattern” is composed from “Sales Order & Purchase Order Pattern”, “Closing Pattern” and so on. The UML parameterized collaboration mechanism is used to materialize the pattern integration.

One of the major benefits of using this multi-layered structure is that it enables reuse (inheritance) of the constraints which have been defined and encapsulated in patterns in the layers. It provides a normalized way to define constraints and is effective in maintaining consistency within the object model.

The concepts of Business Pattern Package (defining a pattern) and Business Pattern Binding (applying a pattern) have the features of pattern inheritance

and pattern composition. This capability is useful for expressing patterns that include the objects constructed by recursive component composition as defined by the ECA.

The Patterns Profile defines three basic forms of pattern:

36. a simple pattern, which is a pattern consisting of minimal elements needed to form a pattern;
37. an inherited pattern, which is a pattern defined by inheriting from another pattern;
38. a composite pattern, which is a pattern defined as a result of combining more than two patterns: the composite pattern concept is an extension of the inherited pattern.

Using the above three basic forms of pattern as the base, notations for expressing patterns and their metamodel are defined.

The instantiation of a composite pattern in a hierarchical structure becomes possible by resolving pattern inheritance and collaboration by performing "unfold". When composite patterns are granular enough to include implementation details, it is possible to use them to describe a component concept such as that defined in the CCA, each pattern package can be implemented with real components instead of unfolding it into a components pattern. In short, the proposed pattern concept and mechanism can be applied to the components based development that is required for EDOC systems.

The Patterns profile also includes standard models from the ECA such as Business Entities, Business Processes, Business Events and Business Rules, together with , together with a set of common and reusable patterns of relationship properties that occur in business modeling.

The profile does not define any new metamodel elements. The pattern notation uses currently available metamodel elements and patterns are described using the UML pattern notation.

### 38.1.1 Mappings to Technology - Platform Independence

Viewpoint abstractions in the context of model-based development provide mechanisms for specifying platform independent models of business applications.

Such platform independence is an important element in systems that can adapt to change and, hence, is a fundamental element of the EDOC vision (Figure 1). The rate of change of today's enterprises and their requirements generates demands for flexible and dynamic systems that are capable of coping with the ever changing business requirements and with changes in software and hardware technologies.

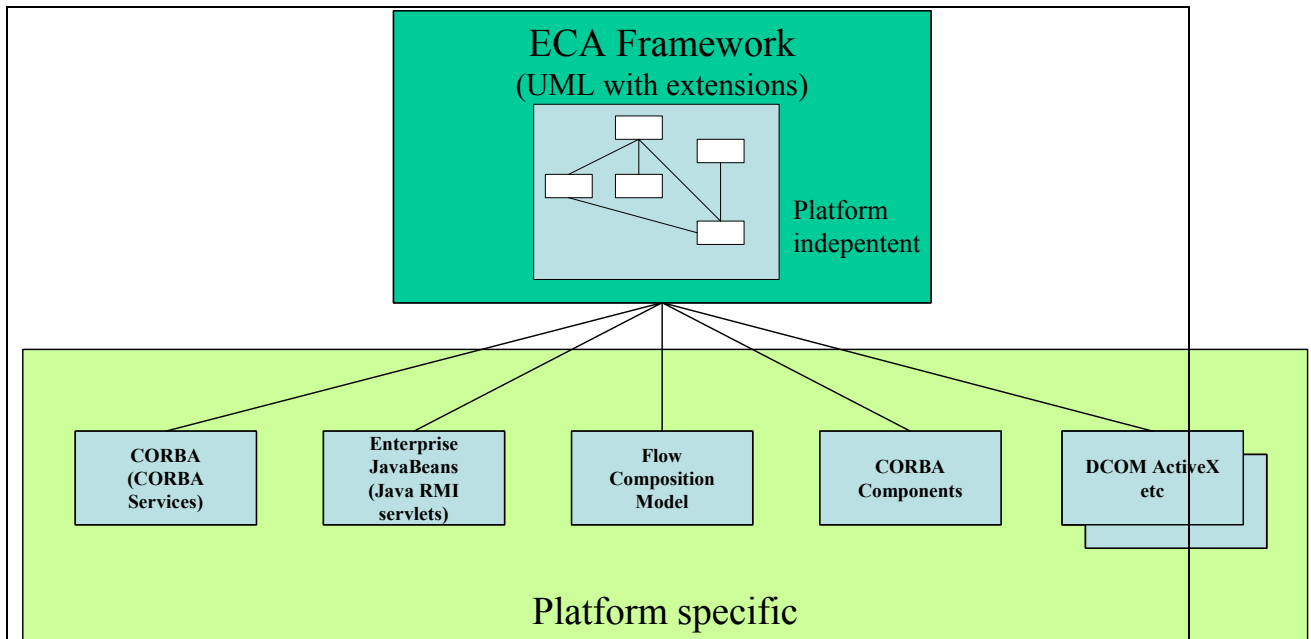


Figure 1: EDOC framework vision