

# COMET - UML profile for service architecture model

<!-- -->





## Table of contents




1 Introduction.....	2
2 Structuring Concepts.....	2
3 Modelling Concepts.....	3
4 Validation rules.....	9
5 Predefined views.....	9
5.1 Examples.....	10

## 1. Introduction



This UML profile defines the stereotypes that COMET will use for modelling the system architecture, which includes the components, their interfaces and relationships as well as their properties.

## 2. Structuring Concepts


Concept	Extended Property	Description	UML Mapping	Icon
Application Specification		Application specification.	Package <<ApplicationSpec>>	
Business Service Specification		Business service specification.	Package <<BusinessServiceSp	
Protocol Specification		Shared or standard protocol specification. (Modelled as role models.)	Package <<ProtocolSpec>>	
Resource Service Specification		Resource service specification.	Package <<ResourceServiceS	
Service Architecture Model		The model package that contains the service architecture model.	Model <<ServiceArchitectur	
Tool Specification		Tool specification.	Package <<ToolSpec>>	
Type Library		Type libraries define various datatypes and primitive classes that are used in the architecture extending the simple, primitive	Package <<TypeLibrary>>	


		types supported by UML such as string, float, integer, etc.		
User Interface Specification		User interface specification.	Package <<UserInterfaceSpec	
User Resource Service Specification		User resource service specification.	Package <<UserResourceServ	
User Service Specification		User service specification.	Package <<UserServiceSpec>	


### 3. Modelling Concepts



Concept	Extended Property	Description	UML Mapping	Icon
Application Component		Application component is a business level component, which provides end-user functionality within a business do-main. An application aggregates sets of tool, business service, interworking work-flow and resource service components.	Component <<Application	
Business Service Component		A Business Service Component provides business functionality for User Service Components and thus implements business logic. A Business Service Component has	Component <<BusinessService>>	

		transactional responsibility and has ownership of the information available through that service. Information is represented by entities (business entities).		
Composite Data		A composite data structure.	Class <<CompositeData>>	
Entity		An entity is a persistent data object that is managed by a component. We can define components that are solely responsible for a specific entity type.	Class <<Entity>>	
EntityData		Persistent data managed by an entity.	Class <<EntityData>>	
Event		The description of an event.	Class <<Event>>	
Event Interface		An event interface publishes or subscribes to events. Events in this context are descriptions of an actual event having occurred.	Interface <<EventInterface>>	
Message		The description of a message.	Class <<Message>>	
Message Interface		A message interface operates on messages, which unlike events that	Interface <<MessageInterface>>	

		are read-only, also can be modified. In addition to the publish and subscribe mechanism, messages can also be written to directly using put and get operations.		
notification		Operation that indicates oneway notification of event, where a client is notified of an event. Implies a receiving notification/listener interface on the client side.	Operation <<notification>>	
oneway		Operation that indicates directed, oneway request with no return values.	Operation <<oneway>>	
PrimaryKey		A primary key defines a persistent unique identifier for a specific entity. The primary key may also be defined as a PrimaryKeyField attribute on the Entity itself.	Class <<PrimaryKey>>	
PrimaryKey	Primary Key Field	Part of primary key for an entity.	Property <<PrimaryKeyField>	
Protocol		Protocols define the interaction between components. A protocol is associated with some behavioural specification, e.g.	Class <<Protocol>>	

		state machines and/or sequence diagrams.		
publishes		Publishes from a business service component to an event or message.	Dependency <<publishes>>	
request-response		Operation that indicates directed request with return value.	Operation <<request-response>>	
Resource Service Component		A Resource Service Component provides access to underlying resource providers, e.g. a data store. It provides the necessary interfaces to store, retrieve, and query information from a resource provider. These may also be described, if it seems necessary to explicitly capture specifics of persistent resource storage design, e.g. because the logical business entity model differs greatly from the data storage design.	Component <<ResourceService>>	
solicit-response		Operation that indicates a two-way notification with response, where a client is notified of an event and can respond to that event. Implied a receiving notification/listener interface on the	Operation <<solicit-response>>	

		client side.		
Subscribes		Subscribes from a business service component to an event.	Dependency <<subscribes>>	
Tool Component		Tool component represents a logical client-side component, consisting of user interface, user service, and user resource service components. A Tool Component is a composition or usage of a set of user interface and user service components. A tool component implicitly accesses a set of business service components via its user service components. A tool component represents functionality that supports a particular business role in a set of use cases; as such, it is closely linked to the business and the business specification.	Component <<Tool>>	
User Interface Component		User interface components provide presentation and/or user dialog logic. A User Interface Component is any component used by a tool to provide interactive services for an end user. A	Component <<UserInterface>>	

		User Interface Components accesses User Service Components to provide the necessary business functionality for the end user.		
User Resource Service Component		A User Resource Component provides resource (typically data) services to a given User Service Component.	Component <<UserResourceServ	
User Service Component		User service components provide user-specific abstractions of a set of business services. A User Service Component provides an abstraction from the business service tier. It provides middleware transparency for the client-side. This decreases complexity for client developers. A User Service Component packages a service and a set of entities. The service represents the functionality available to a client. The entities represent the information abstractions available through the service. The entity within a user service is a local,	Component <<UserService>>	

		session-specific representation of business entity concepts.		
ValueType		A simple value type.	Class <<ValueType>>	

## 4. Validation rules

Validation rules are composed of a set of metrics (m), which qualifies the model, and absolute constraints (c), which can invalidate the system if the model does not meet the criteria.

Check User Interface Specification	Package	A UserInterface can only access UserService or an interface thereof.
Check User Service Specification	Package	A UserService can only access BusinessService or an interface thereof.
Check Business Service Specification	Package	A BusinessService can only access other BusinessService components and ResourceService component or an interface thereof.
Check Component Structure	Package	Checks that the component is modelled according to the component model management structure.

## 5. Predefined views

Name	MetaClass	Description
Component Interface View (detailed & non-detailed)	Package	Displays the interfaces of the selected component specification package.
Interface Information View	interface	Displays the information model of the selected interface class (return values, parameters, ...)

Collaboration Specification View	Package	Displays the collaborations and dependencies between the subcomponents of the component package.
Component Dependencies View	Package	Displays the external dependencies of the selected component specification.
Component Specification View	Package	Displays all visible properties (interfaces, return values, ...) of the selected component specification.

## 5.1. Examples

---

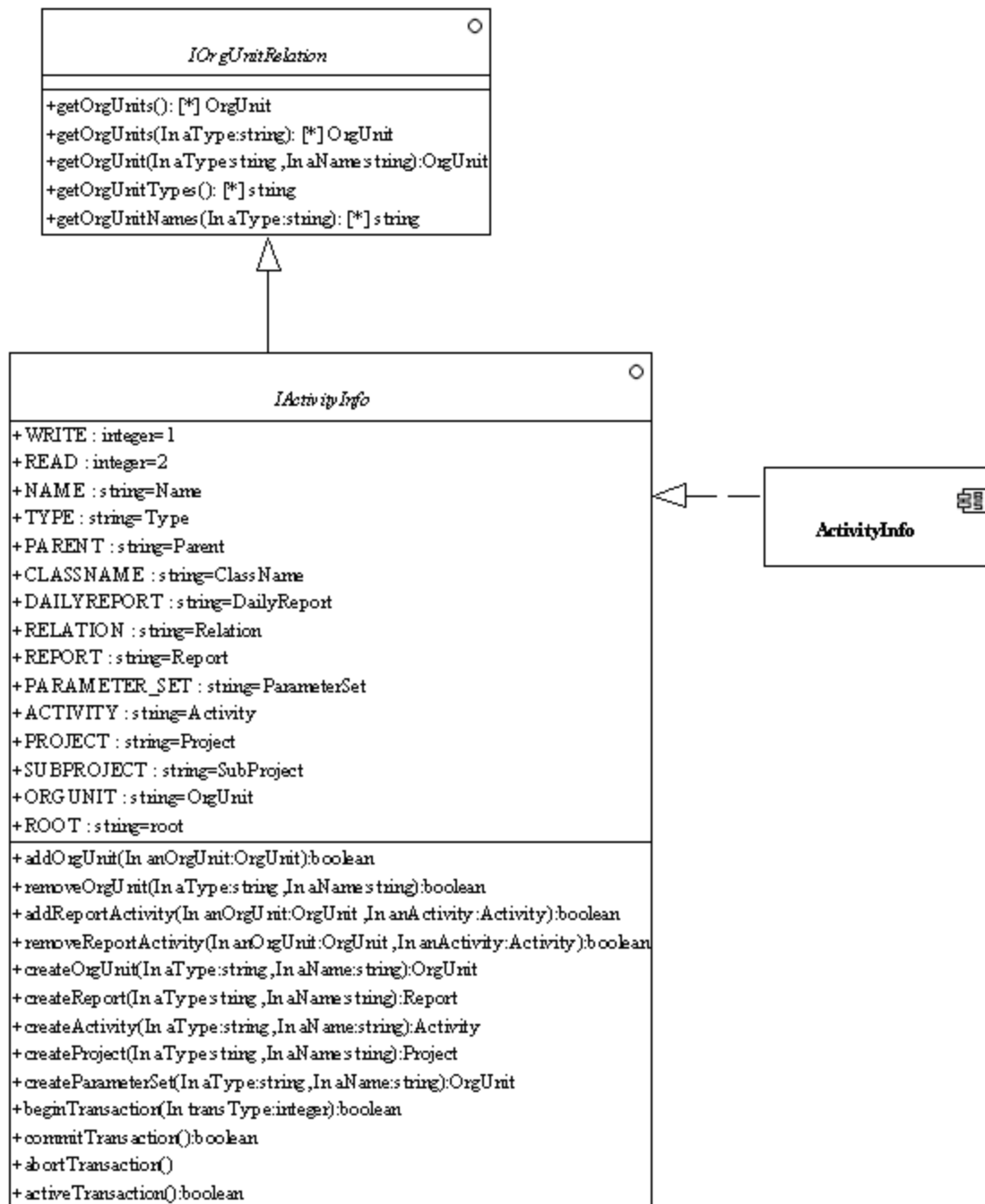


Figure 1: Detailed Interface View

Figure 1 shows a detailed view of the provided interfaces.

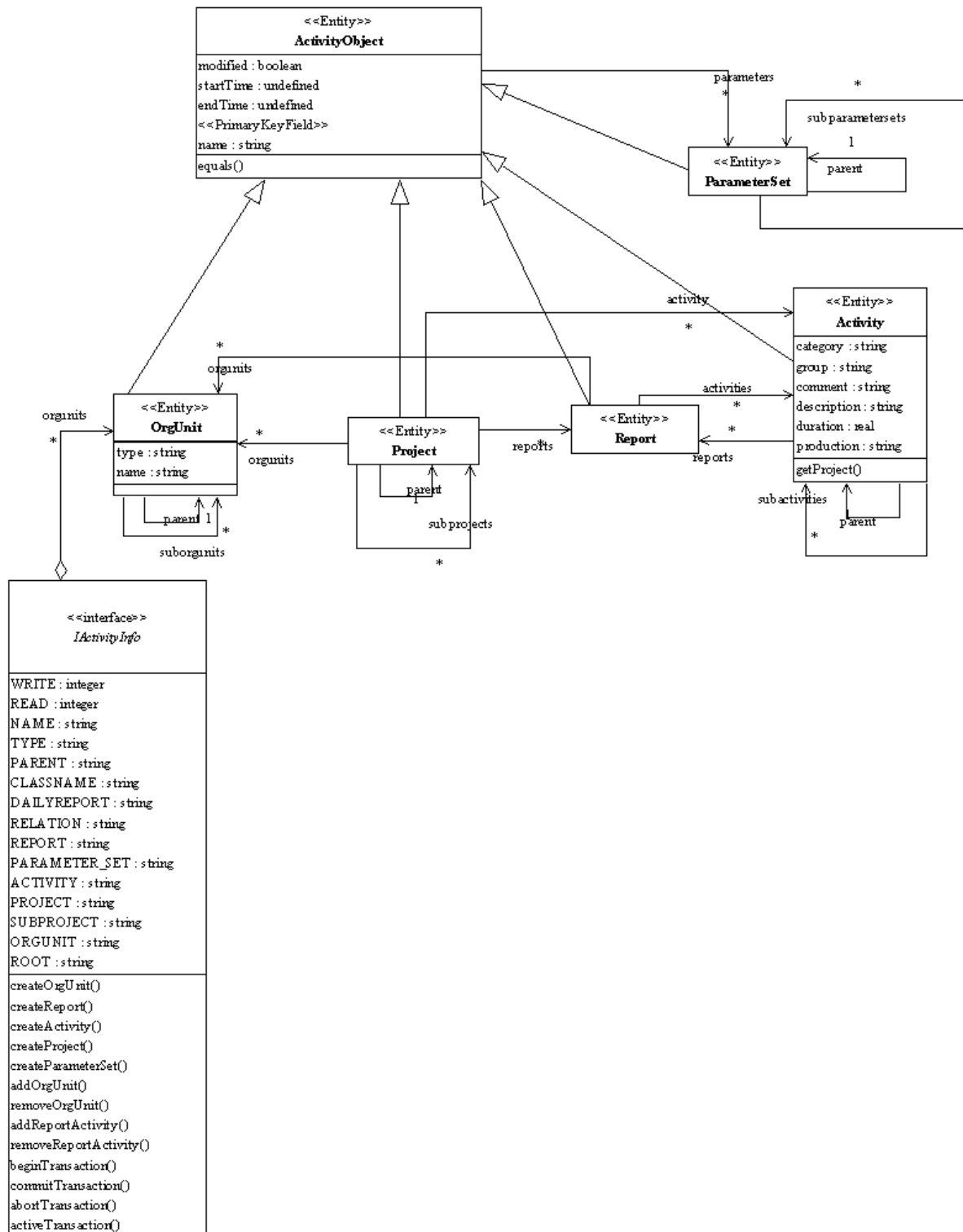


Figure 2: Interface Information Example

Figure 2 shows the information elements exposed by an interface.

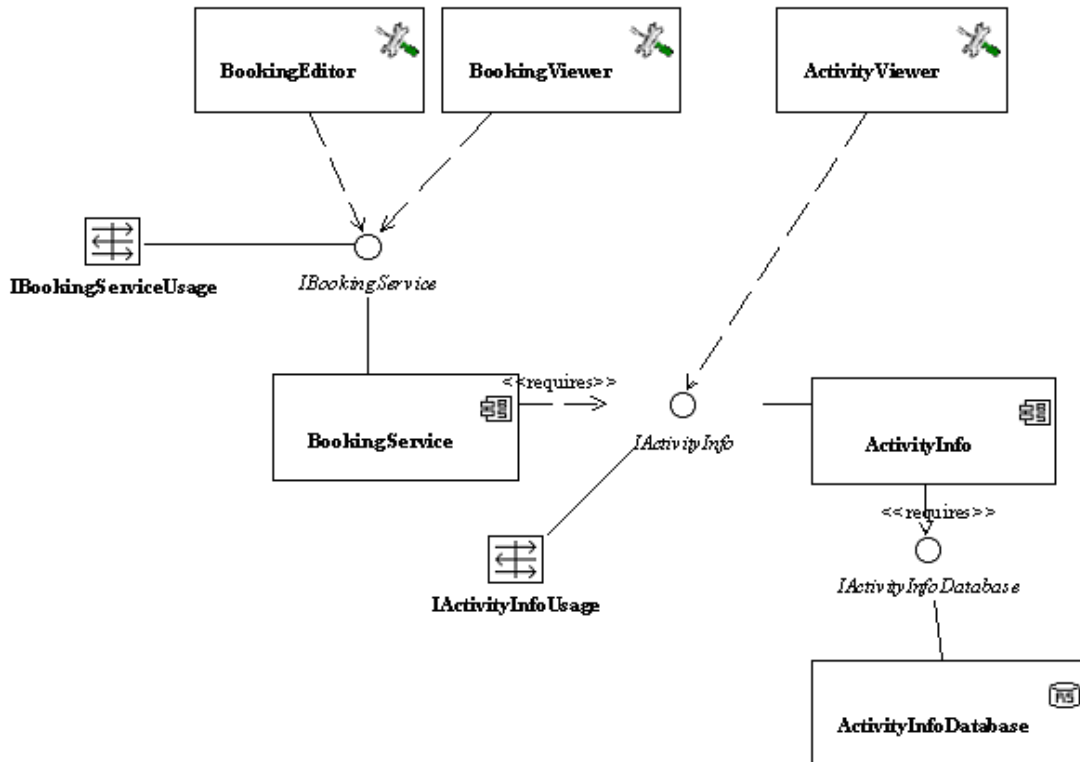


Figure 3: Component Collaboration View Example

Figure 3 shows an example of Component Collaboration View.