

COMBINE - Service architecture model example

<!-- -->

Table of contents

1 Reference Architecture Analysis.....	2
1.1 Survey Booking Subsystem Grouping Use Case Model.....	2
1.2 Component Identification.....	2
1.3 Component Interfaces.....	3
2 The Booking Editor Tool description.....	4
2.1 User Service.....	6
3 The Booking Viewer Tool description.....	9
3.1 User Service.....	10
4 The Subscription Editor Tool description.....	11
4.1 User Service.....	11
5 The Booking Service Business Service description.....	12
5.1 Interface Specification.....	13
6 The Subscription Service Business Service description.....	14
6.1 Interface Specification.....	15
7 The Activity Info Resource Service description.....	16
7.1 Activity Info Model.....	16
7.2 The Activity Info Resource Service description.....	16

1. Reference Architecture Analysis

1.1. Survey Booking Subsystem Grouping Use Case Model

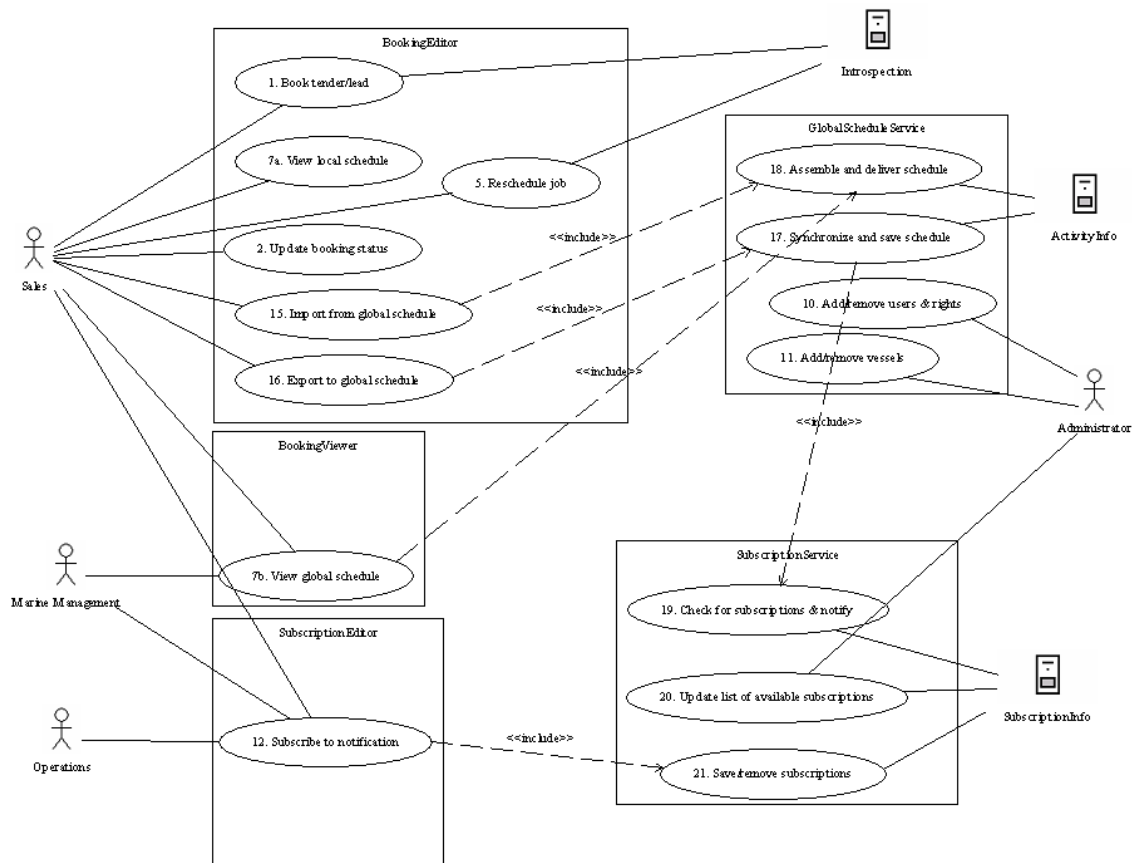


Figure 1: Survey Booking subsystem grouping use cases

Figure 1 is a use case diagram that shows the subsystems of the Survey Booking Application. Each subsystem has a set of use cases that must be realized and offered as required to its users. Human users interact with the system using tool components that provide graphical user interfaces. Each of these tools collaborates with business service components. Business services are network visible and can be shared between multiple tools. Persistent storage for the business services is provided by resource services.

1.2. Component Identification

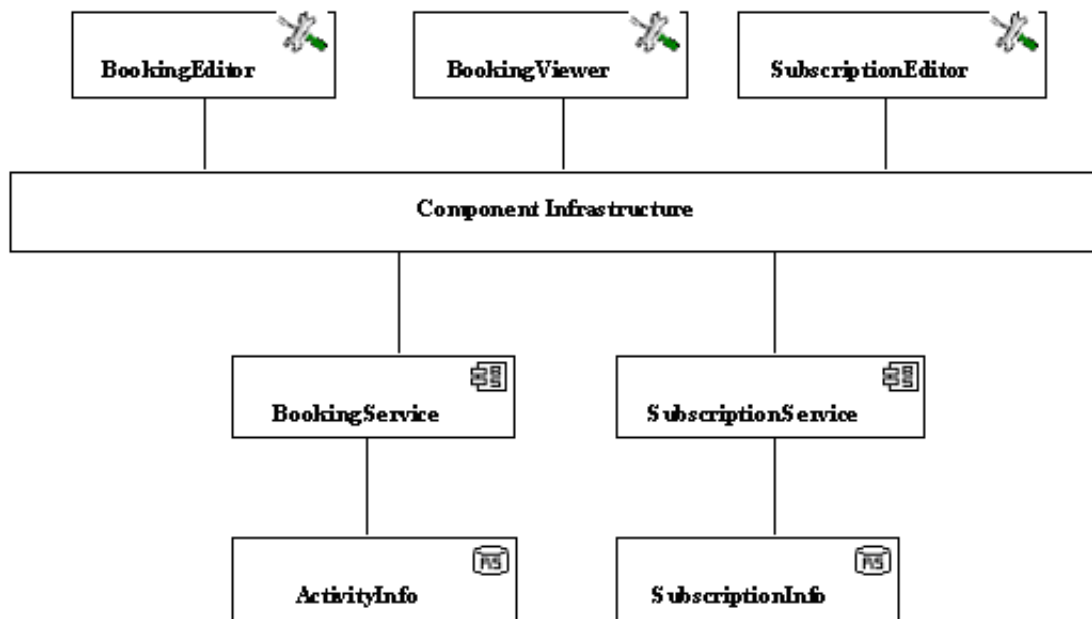


Figure 2: Survey Booking reference architecture analysis

Figure 2 is a class diagram that shows the mapping of the subsystems to corresponding Tool, BusinessService and ResourceService components.

- The subsystems **BookingEditor**, **BookingViewer** and **SubscriptionEditor** are used by human users. They map to Tool components according to the reference architecture. The **BookingEditor** is to be installed locally on a client machine, while the **BookingViewer** and the **SubscriptionEditor** will be available through a web server.
- The subsystems **GlobalScheduleService** and **SubscriptionService** offers services to the tools and maps to BusinessService components. The tools communicate with these business services over a network. The **GlobalScheduleService** corresponds to the component **BookingService** in the figure above.
- The subsystems **ActivityInfo** and **SubscriptionInfo** map to ResourceService components that provide persistent data storage to the **BookingService** and **SubscriptionService** respectively.

The structure of the Survey Booking Application component is shown in the diagram below. Components are modelled as UML classes. Each component class is modelled within a UML package that holds the full specification of the component.

1.3. Component Interfaces

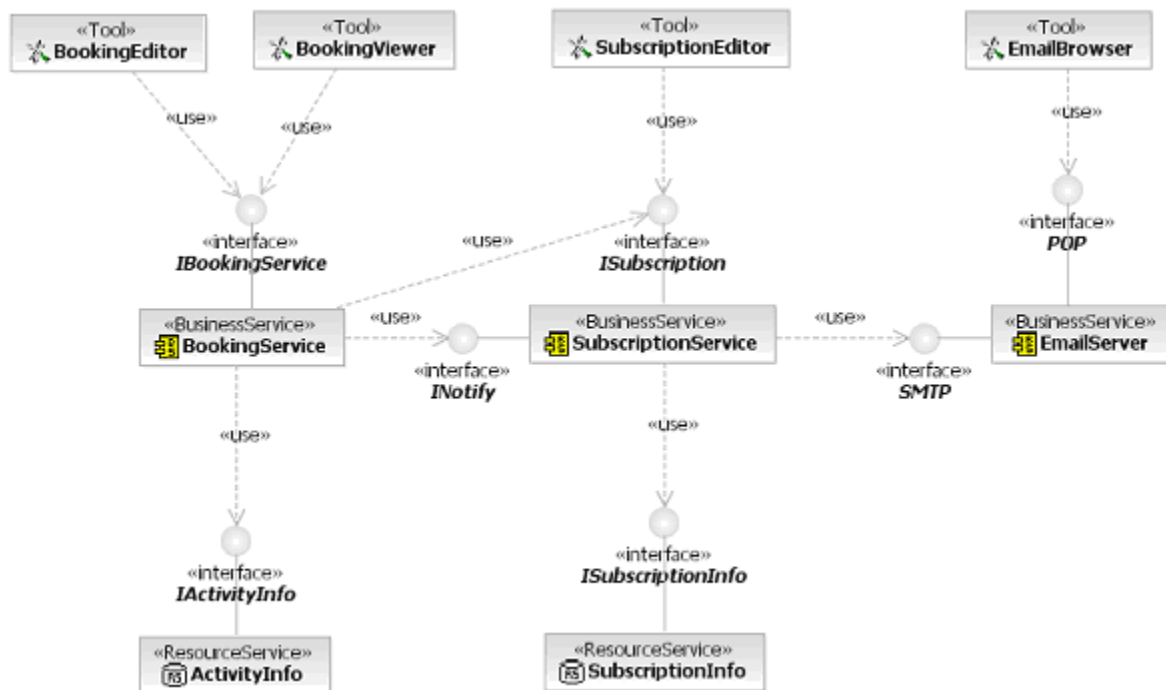


Figure 3: Survey Booking component structure

Components collaborate with each other through well-defined interfaces. Figure 3 is a class diagram that shows the components and interfaces of the Survey Booking Application. Two additional components are shown in this diagram. The **EmailBrowser** and **EmailServer**. The **EmailBrowser** represents a standard Email reader tool that is used in the Survey Booking context for retrieving and reading booking notifications. The **EmailServer** represents the standard email server business service that is deployed within the company. As can be seen, the **SubscriptionService** requires the **EmailServer** in order to send notifications.

All of the components presented here will be handled in separate chapter. For each tool, the use cases involved will be broken down to define user services. For each business service, the use cases involved will be broken down to define and interfaces offered to the tools.

2. The Booking Editor Tool description

A tool consists of a User Interface component and a User Service component. The Booking Editor must also be able to operate in an offline modus. This means that local persistent storage must be provided.

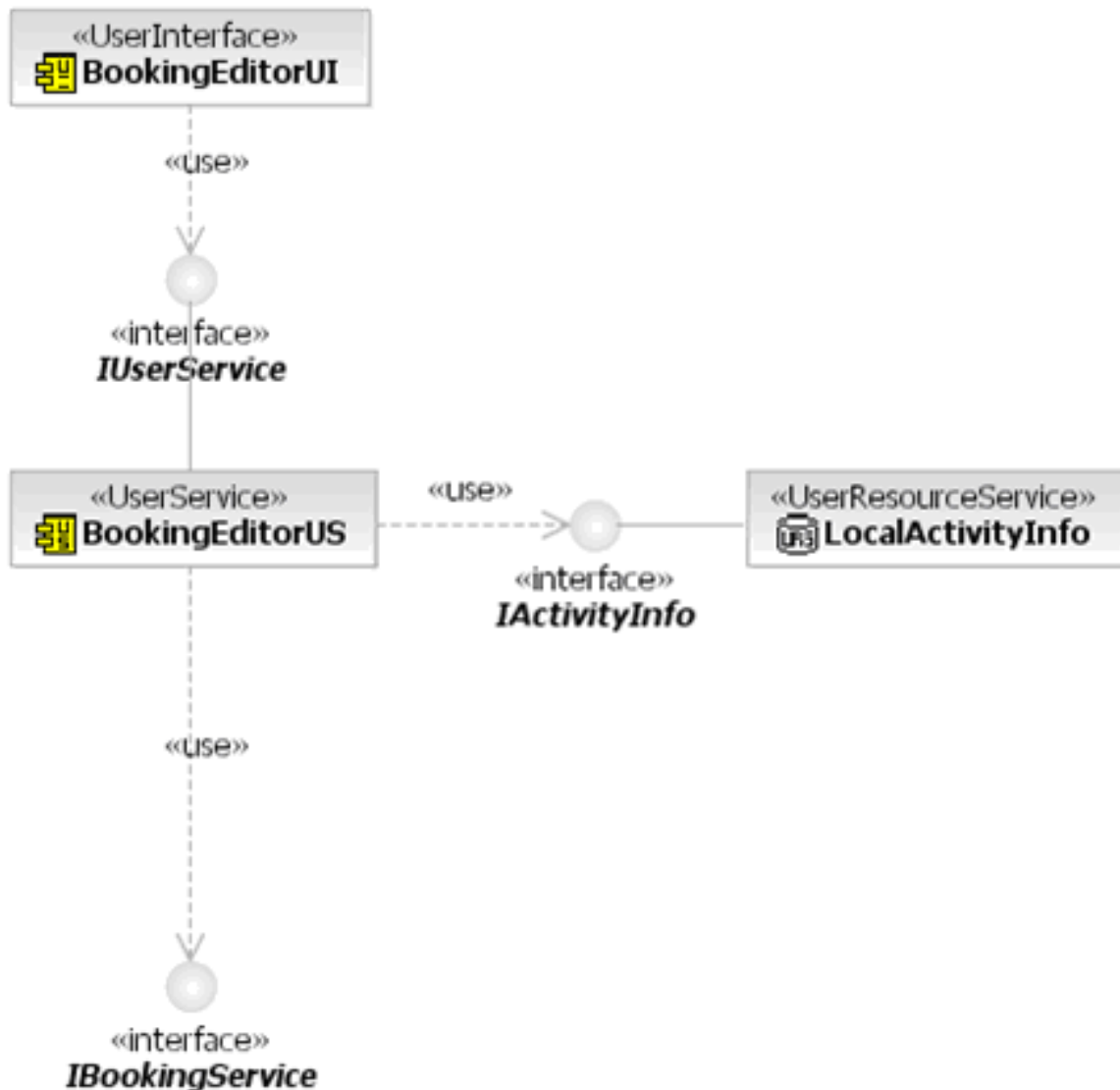


Figure 4: Booking Editor component structure

Figure 4 is a class diagram that shows the internal components of the Booking Editor Tool. **BookingEditorUS** is a User Service that provides a façade to the Tool. The **BookingEditorUS** uses **LocalActivityInfo**, which is a User Resource Service that provides the local data persistent storage used in offline modus.

The **LocalActivityInfo** component is provided by a lightweight implementation of the **ActivityInfo** component described in appendix C. The User Service is described in details below. The User Interface component will be described in the forthcoming design document.

2.1. User Service

In order to specify the User Service for the Booking Editor we need to do a use case analysis. The use case analysis takes the Business Resource Model as input and analyses the information and operations that must be provided through the user service.

2.1.1. Business Resource Analysis

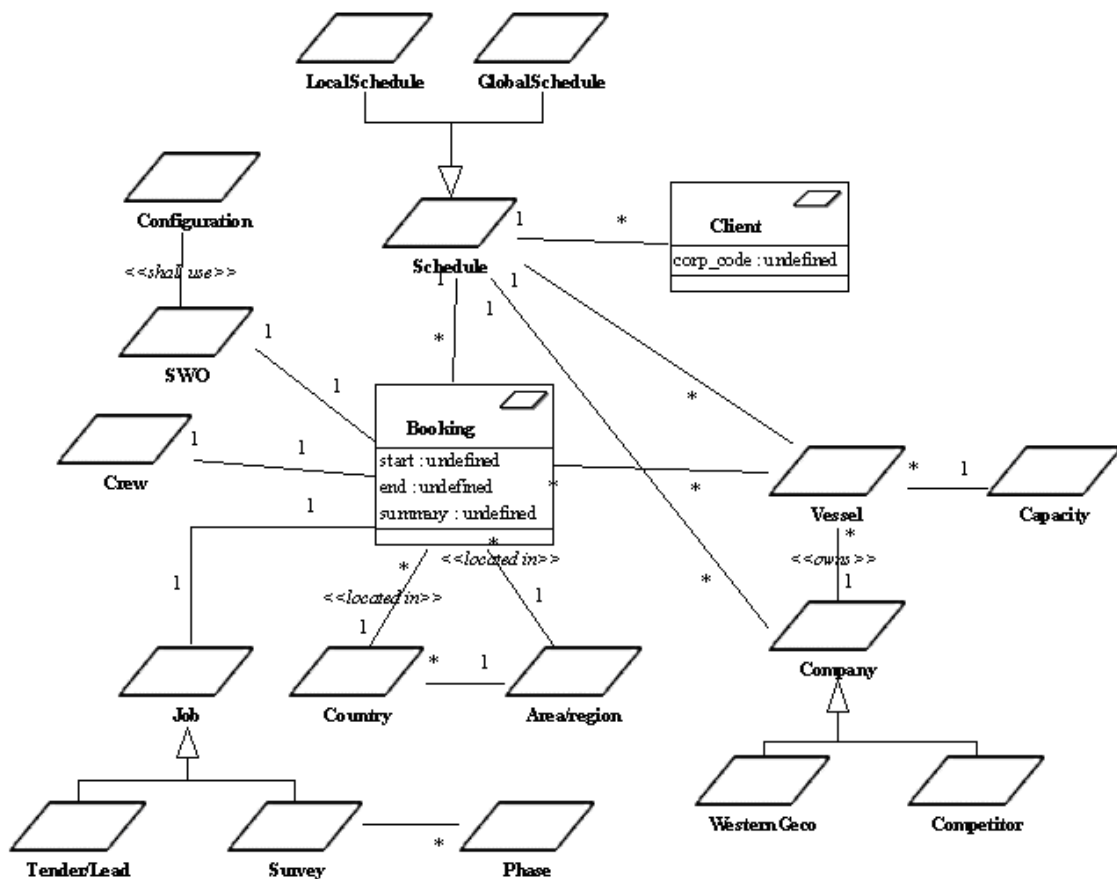


Figure 5: Booking Editor business resource analysis

Figure 5 shows the business information that must be presented to the user through the User Interface component. This information must be accessible to the User Interface component from the User Service.

The information provided by the User Service is based on the generic ActivityInfo model, mapping the business information to appropriate activity elements. This information is provided to the User Interface, which is to be implemented as a rich client.

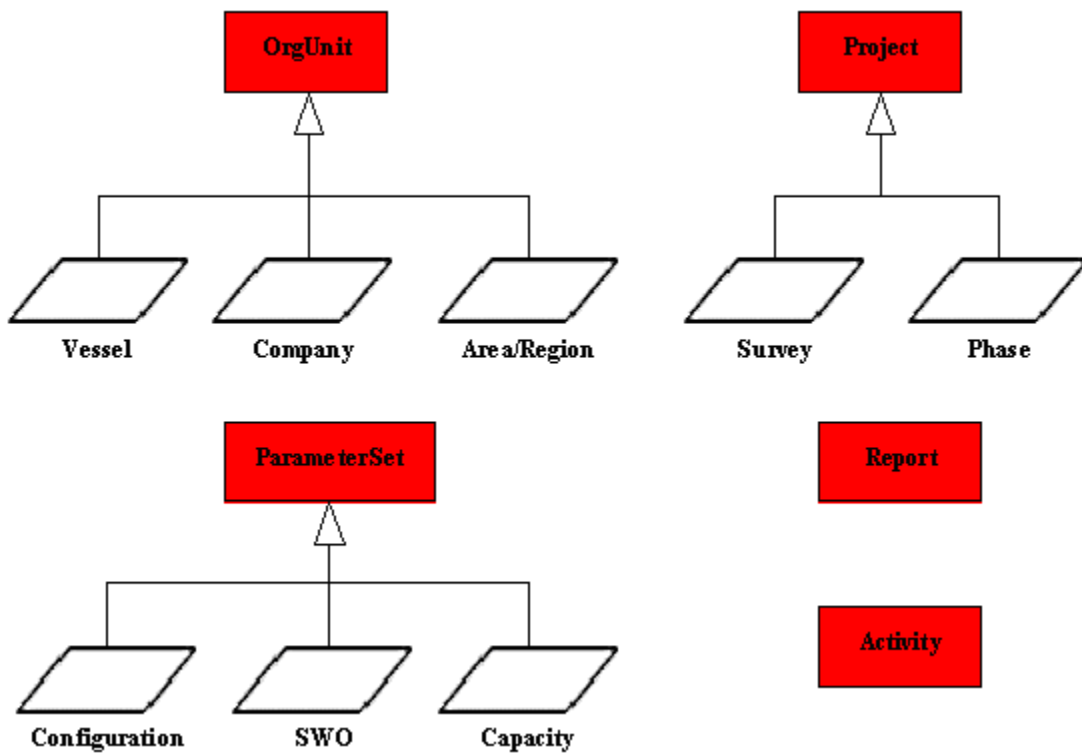


Figure 6: Business Resource Model mapping to Activity Info Model

Figure 6 shows the mappings from the Business Resource Model to the Activity Info Model.

2.1.2. Interface Operations

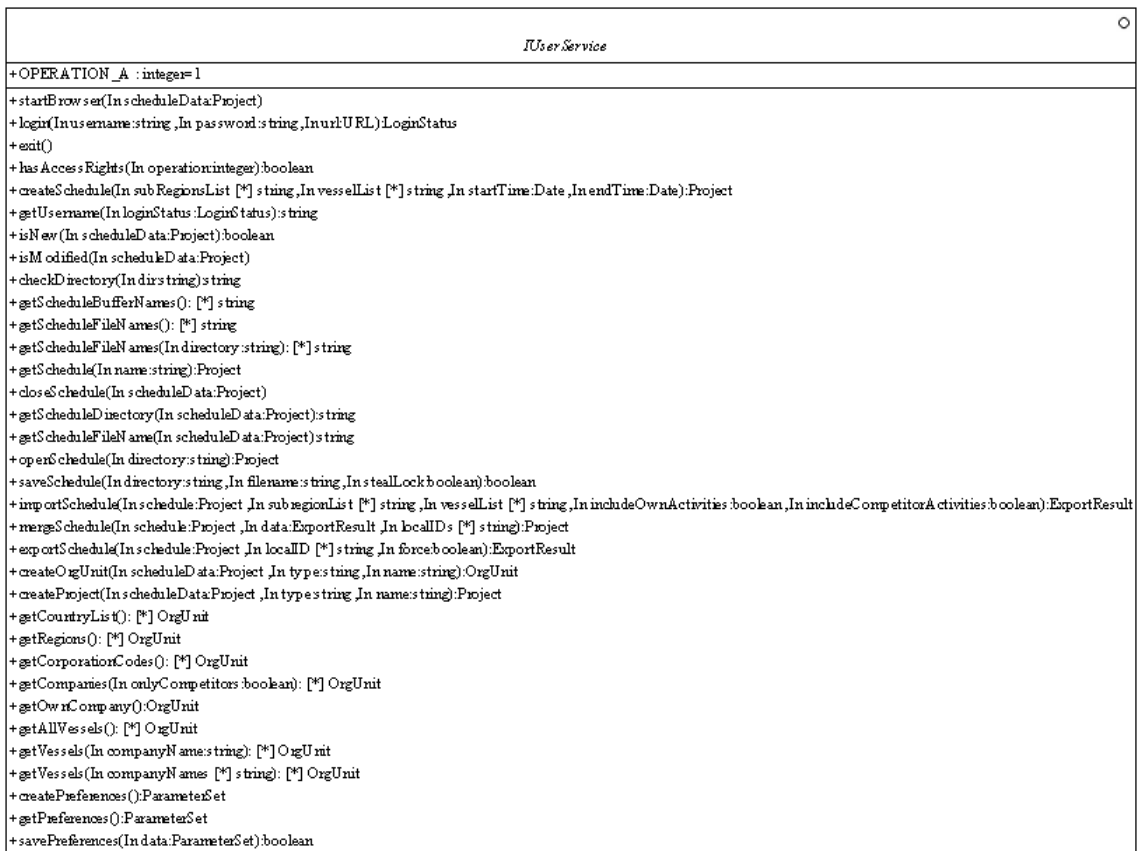


Figure 7: User Service interface operations

Figure 7 shows the interface operations provided by the **IUserService** interface.

2.1.3. Interface Information

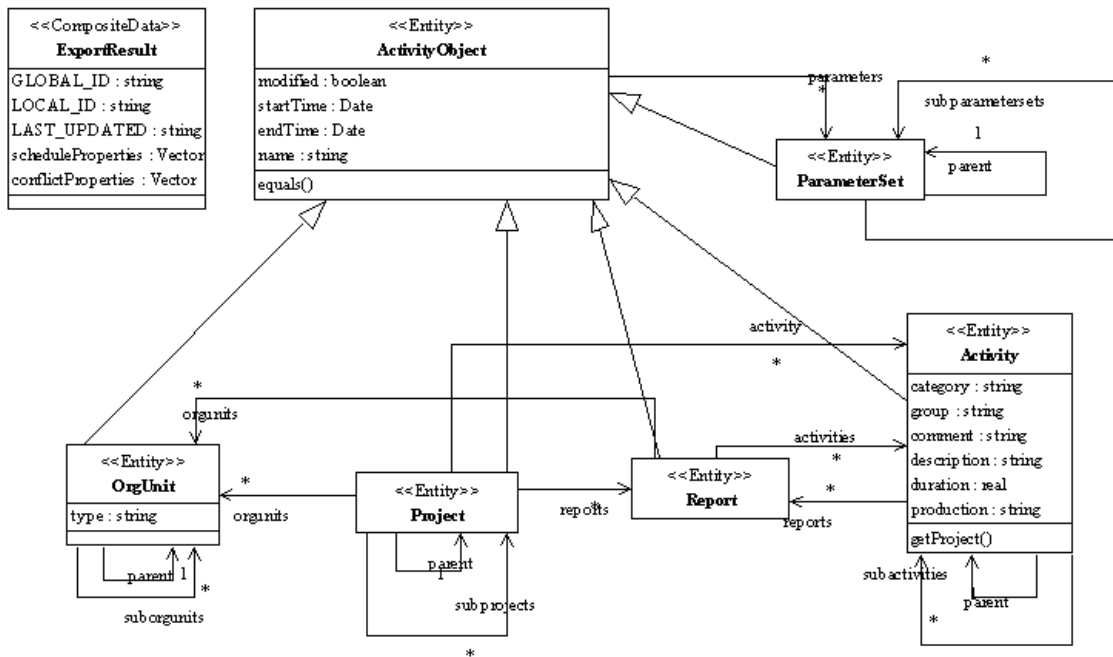


Figure 8: User Service interface information

Figure 8 shows the information that is available through the **IUserService** interface. The entities correspond to the ActivityInfo information model described in section 0. The composite data corresponds to the exported data type provided by the **BookingService** component.

3. The Booking Viewer Tool description

This tool resembles the **BookingEditor** tool. However, it is a viewer only, so it should not support editing and should only access the global schedule.

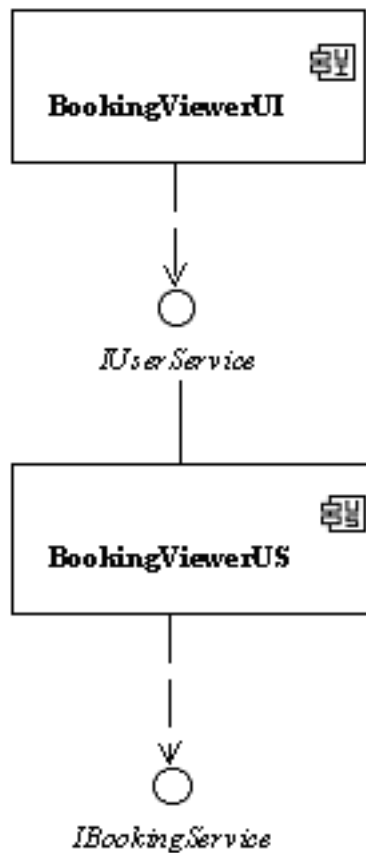


Figure 9: Booking Viewer component structure

Figure 9 shows the component structure for the **BookingViewer** component. Since no offline modus is required, a **LocalActivityInfo** component is not needed as in the **BookingEditor** case.

3.1. User Service

The business resource analysis and the interface information are the same as for the Booking Editor tool. However, the interface operations for the Booking Viewer are more lightweight, as can be seen in the diagram below.

```

IUser Service
+OPERATION_A : integer= 1
+login(In username:string ,In password:string ,In url:URL):LoginStatus
+exit()
+hasAccessRights(In operation:integer):boolean
+createSchedule(In subRegions:List [*] string ,In vesselList [*] string ,In startTime:Date ,In endTime:Date):Project
+getUsername(In loginStatus:LoginStatus):string
+createOrgUnit(In scheduleData:Project ,In type:string ,In name:string):OrgUnit
+createProject(In scheduleData:Project ,In type:string ,In name:string):Project
+importSchedule(In schedule:Project ,In subregionList [*] string ,In vesselList [*] string ,In includeOwnActivities:boolean ,In includeCompetitorActivities:boolean):ExportResult
+getCountryList(): [*] OrgUnit
+getRegions(): [*] OrgUnit
+getCorporationCodes(): [*] OrgUnit
+getCompanies(In onlyCompetitors:boolean): [*] OrgUnit
+getOwnCompany():OrgUnit
+getAllVessels(): [*] OrgUnit
+getVessels(In companyNames [*] string): [*] OrgUnit
    
```

Figure 10: User Service interface operations

4. The Subscription Editor Tool description

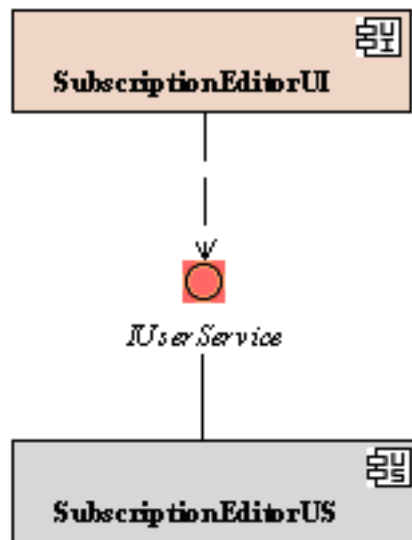


Figure 11: Subscription Editor component structure

Figure 11 shows the component structure for the Subscription Editor tool.

4.1. User Service

The Subscription Editor is a generic notification framework that can be used in many different business contexts. An analysis of the use case gives the following user service specification.

4.1.1. Interface Operations

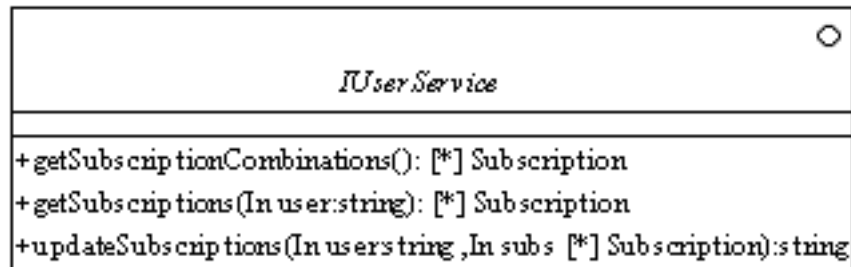


Figure 12: User Service interface operations

Figure 12 shows the operations provided by the **IUserService** interface.

4.1.2. Interface Information

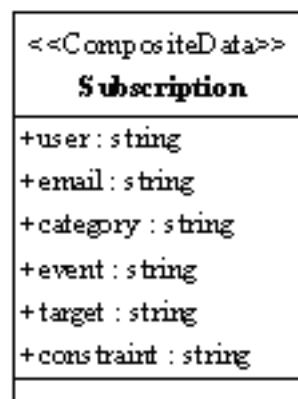


Figure 13: User Service interface information

Figure 13 shows the information available through the **IUserService** interface.

5. The Booking Service Business Service description

The **BookingService** component is used by the **BookingEditor** and the **BookingViewer** tools. The communication between the tools and **BookingService** occurs over a network, so it is important to specify an interface and protocol that minimises the communication overhead. This is done by providing composite data, which are structured by-value data objects that are sent over the network.

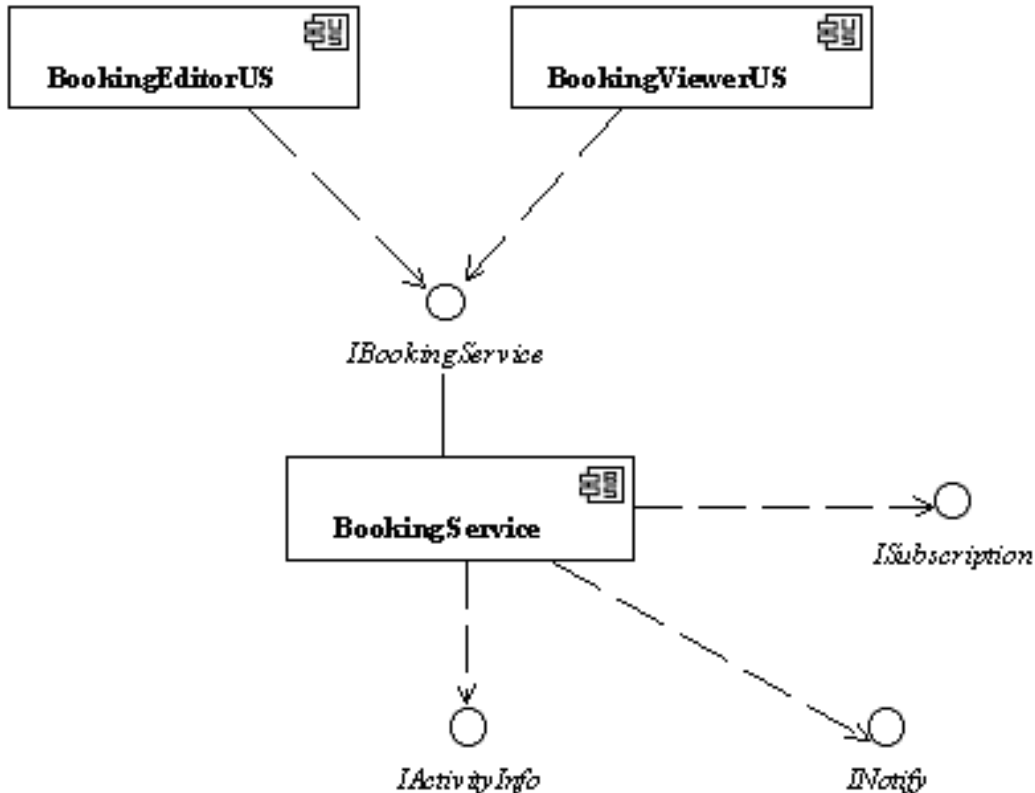


Figure 14: Booking Service component structure

Figure 14 shows the provided and required interfaces of the **BookingService** component. The **IBookingService** interface provides composite data to the tools. The composite data provided by the **IBookingService** encapsulates an activity graph instance according to the Activity Info model.

5.1. Interface Specification

5.1.1. Interface Operations

<i>IBookingService</i>
+userName : string
+userEmailAddress : string
+importSchedule(In subregionList [*] List(string) ,In vesselList:List(string) ,In startTimeDate,In endTimeDate,In existingGlobalIds:Vector ,In companyFilter:CompanyEnum):ExportResult
+exportSchedule(In _scheduleProperties:Vector ,In _force:boolean):ExportResult

Figure 15: Booking Service interface operations

Figure 15 shows the operations provided by the **IBookingService** interface. This interface supports the use cases 1) *synchronize and save schedule*, and 2) *assemble and deliver schedule* of the Global Schedule Service. The vessel and user management use cases will be provided by an administration tool.

5.1.2. Interface Information

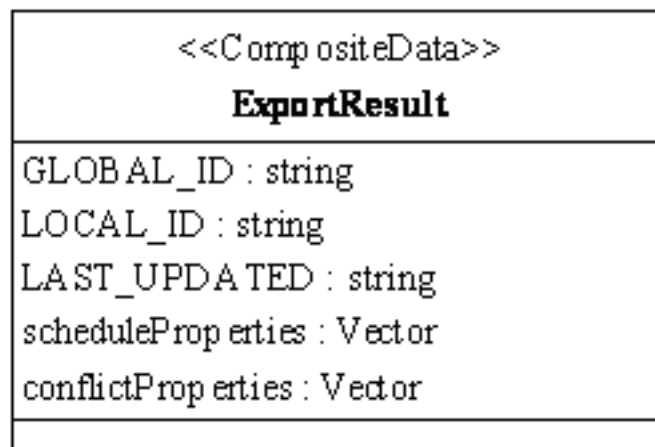


Figure 16: Booking Service interface information

Figure 16 shows the information available through the **IBookingService** interface. As discussed above, **ExportResult** is the composite data representation of an activity graph instance of an Activity Info model.

6. The Subscription Service Business Service description

The **SubscriptionService** component is used by the **SubscriptionEditor** and **BookingService**. Figure 17 shows the required and provided interfaces of the **SubscriptionService** component.

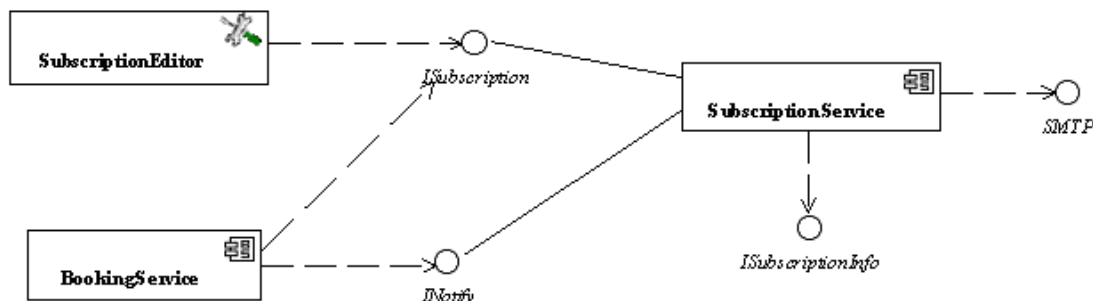


Figure 17: Subscription Service component structure

An analysis of the use cases gives the interface specification detailed below.

6.1. Interface Specification

6.1.1. Interface Operations

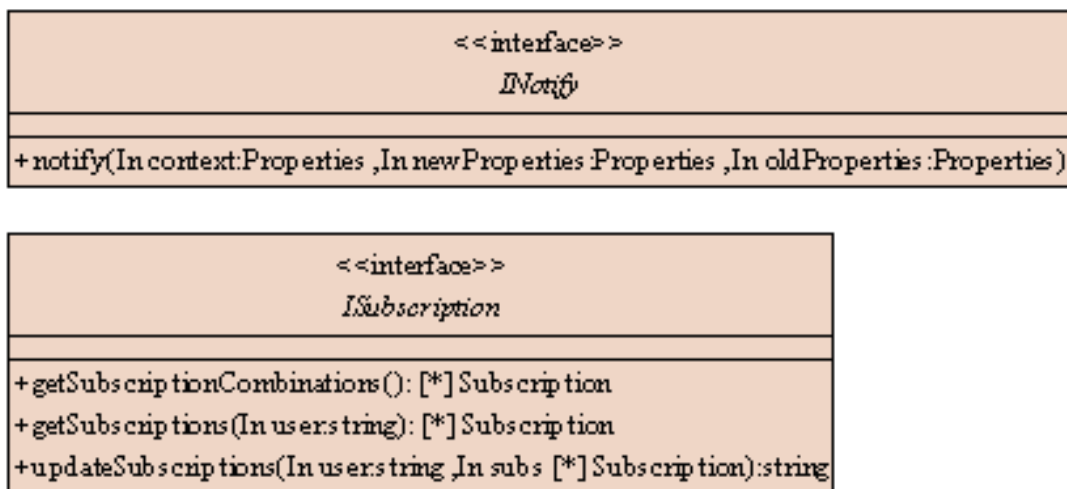


Figure 18: Interface operations for INotify and ISubscription

Figure 18 shows the operations provided by the **INotify** and **ISubscription** interfaces.

6.1.2. Interface Information

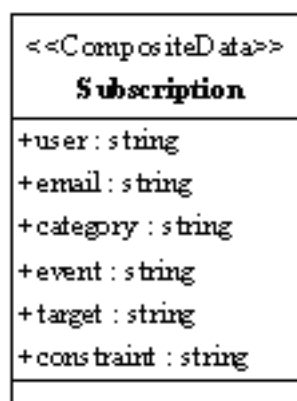


Figure 19: Subscription Service interface information

Figure 19 shows the information available through the interfaces provided by the **SubscriptionService** component.

7. The Activity Info Resource Service description

This section describes the Activity Info Model and the **ActivityInfo** component that implements it.

7.1. Activity Info Model

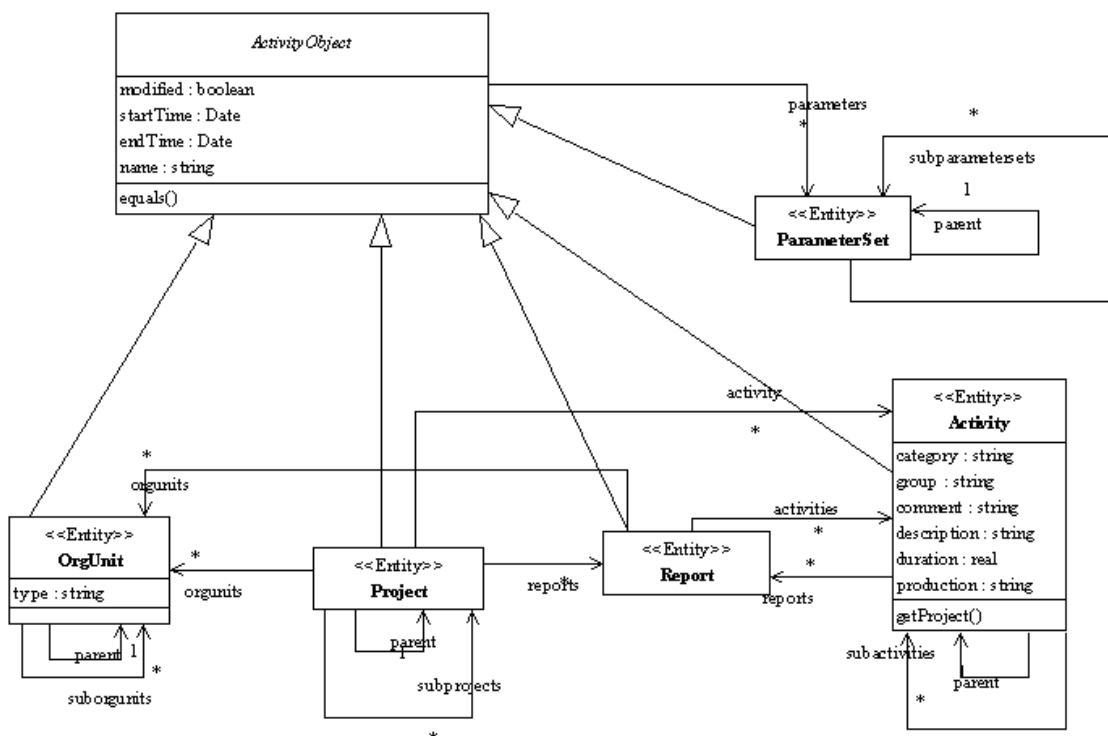


Figure 20: Activity Info Model

The Activity Info Model defines five different activity types that can be used to implement system representation of business resources:

- **OrgUnit** can be used to represent an organisational unit.
- **Project** can be used to represent a project in business domain.
- **Activity** can be used to represent an activity in the business domain.
- **Report** can be used to represent a document which can be generated and printed.
- **ParameterSet** can be used to represent various attribute information.

7.2. The Activity Info Resource Service description

The **ActivityInfo** component is a resource service that implements the Activity Info Model and provides access to other business service components through a well-defined interface. The **ActivityInfo** component is also available in a lightweight implementation, called **LocalActivityInfo**, which can be used by tool components to provide offline storage capabilities.

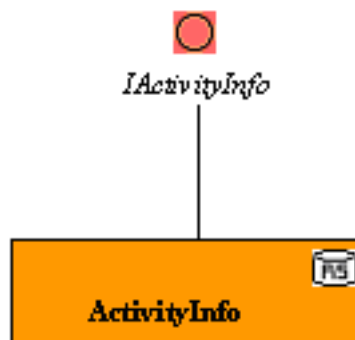


Figure 21: Activity Info component structure

Figure 21 shows the component structure for the **ActivityInfo** component.

7.2.1. Interface Specification

The interface information model exposed by the **IActivityInfo** interface is the Activity Info Model presented above. The interface operations of the interface are shown in the diagram below.

<i>ActivityInfo</i>
<pre> +SUBPROJECT : string=SubProject +WRITE : integer=1 +READ : integer=2 +NAME : string=Name +TYPE : string=Type +PARENT : string=Parent +CLASSNAME : string=ClassName +DAILYREPORT : string=DailyReport +RELATION : string=Relation +REPORT : string=Report +PARAMETER_SET : string=ParameterSet +ACTIVITY : string=Activity +PROJECT : string=Project +ORGUNIT : string=OrgUnit +ROOT : string=root +addOrgUnit(In anOrgUnit:OrgUnit) boolean +removeOrgUnit(In aType:string ,In aName:string)boolean +addReportActivity(In anOrgUnit:OrgUnit ,In anActivity:Activity)boolean +removeReportActivity(In anOrgUnit:OrgUnit ,In anActivity:Activity)boolean +createOrgUnit(In aType:string ,In aName:string):OrgUnit +createReport(In aType:string ,In aName:string)Report +createActivity(In aType:string ,In aName:string):Activity +createProject(In aType:string ,In aName:string)Project +createParameterSet(In aType:string ,In aName:string):OrgUnit +beginTransaction(In transType:integer)boolean +commitTransaction() boolean +abortTransaction() +activeTransaction():boolean </pre>

Figure 22: ActivityInfo interface operations