CHAPTER

IFML extensions

7

7.1 DESKTOP EXTENSIONS

7.1.1 EVENT EXTENSIONS

ONFOCUSLOST

The **OnFocusLost** event is an extension of ViewElementEvent that captures the loss of focus of a SimpleField in a Form. The event is triggered when the user moves away from the field (e.g., by using the tab key or by clicking on another field). It can be associated with a SimpleField or with an entire Form. Its outgoing InteractionFlow can have any ViewElement as a target and a ParameterBindingGroup comprising as input parameter the value of the SimpleField or the values of all the SimpleFields of the Form.

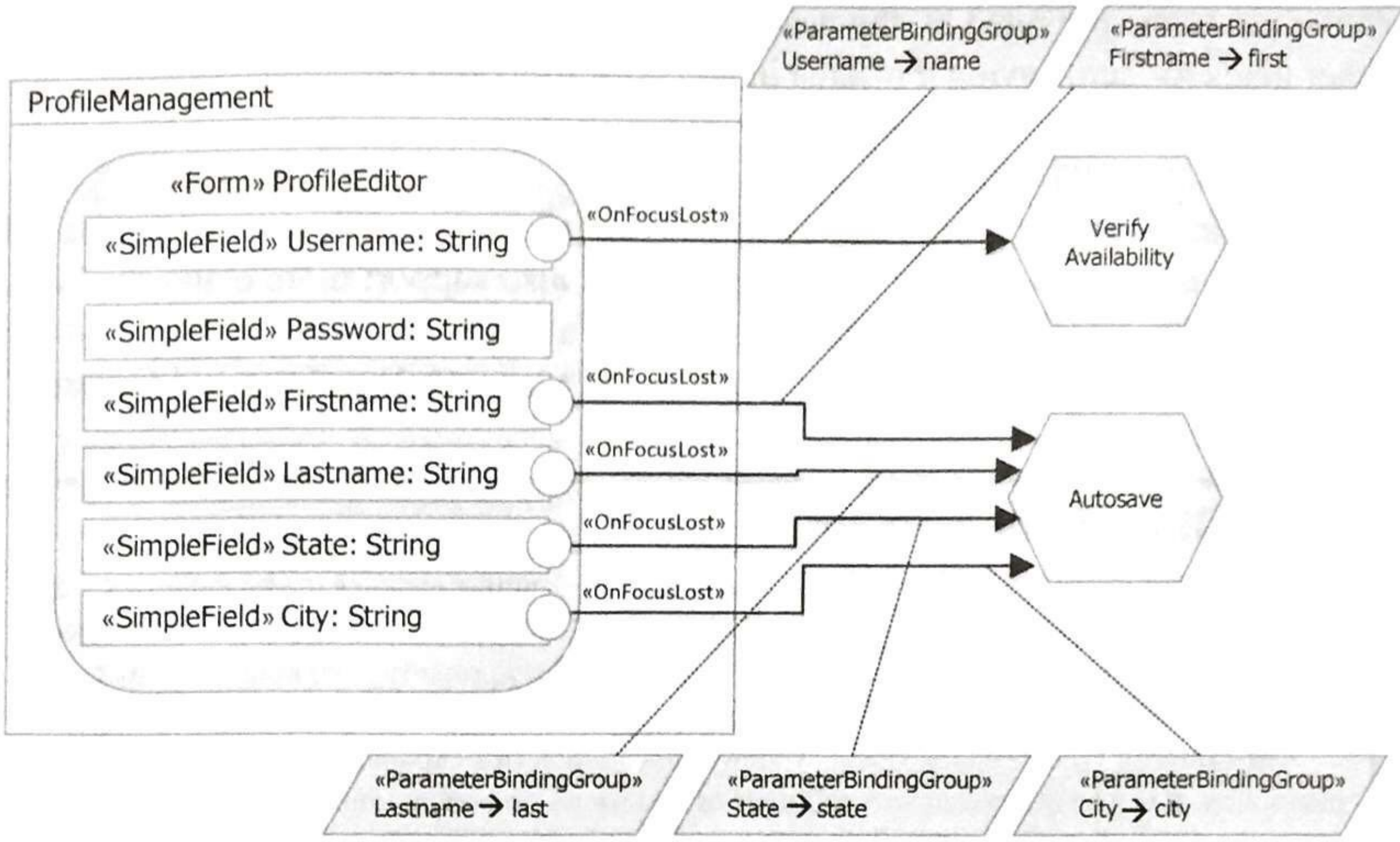


FIGURE 7.2

Example of extended event for Form and SimpleField ViewElements.

7.1.1.1 Drag and Drop

ONDRAGSTART AND ONDROP

The **OnDragStart** event is an extension of ViewElementEvent that captures the beginning of a drag interaction. It can be associated with Details or List ViewComponents (and specializations thereof). It has no outgoing InteractionFlow element. It has a mandatory property "OnDropEvent" that denotes an event of type OnDrop, which is the target of the OnDragStart event.

The **OnDrop** event is an extension of ViewElementEvent that captures the termination of a drop interaction. It can be associated with a Details or List ViewComponent (and specializations thereof). It must appear as the value of the OnDropEvent property of an event of type OnDragStart, which is the source of the OnDrop event. It has one outgoing InteractionFlow element. Such InteractionFlow can have any ViewElement as a target and a DataBindingGroup comprising two input parameters: (1) the value of one or more class instances of the ViewComponent associated with the source OnDragStart event and (2) the value of one or more class instances of the ViewComponent associated with the OnDrop target event.

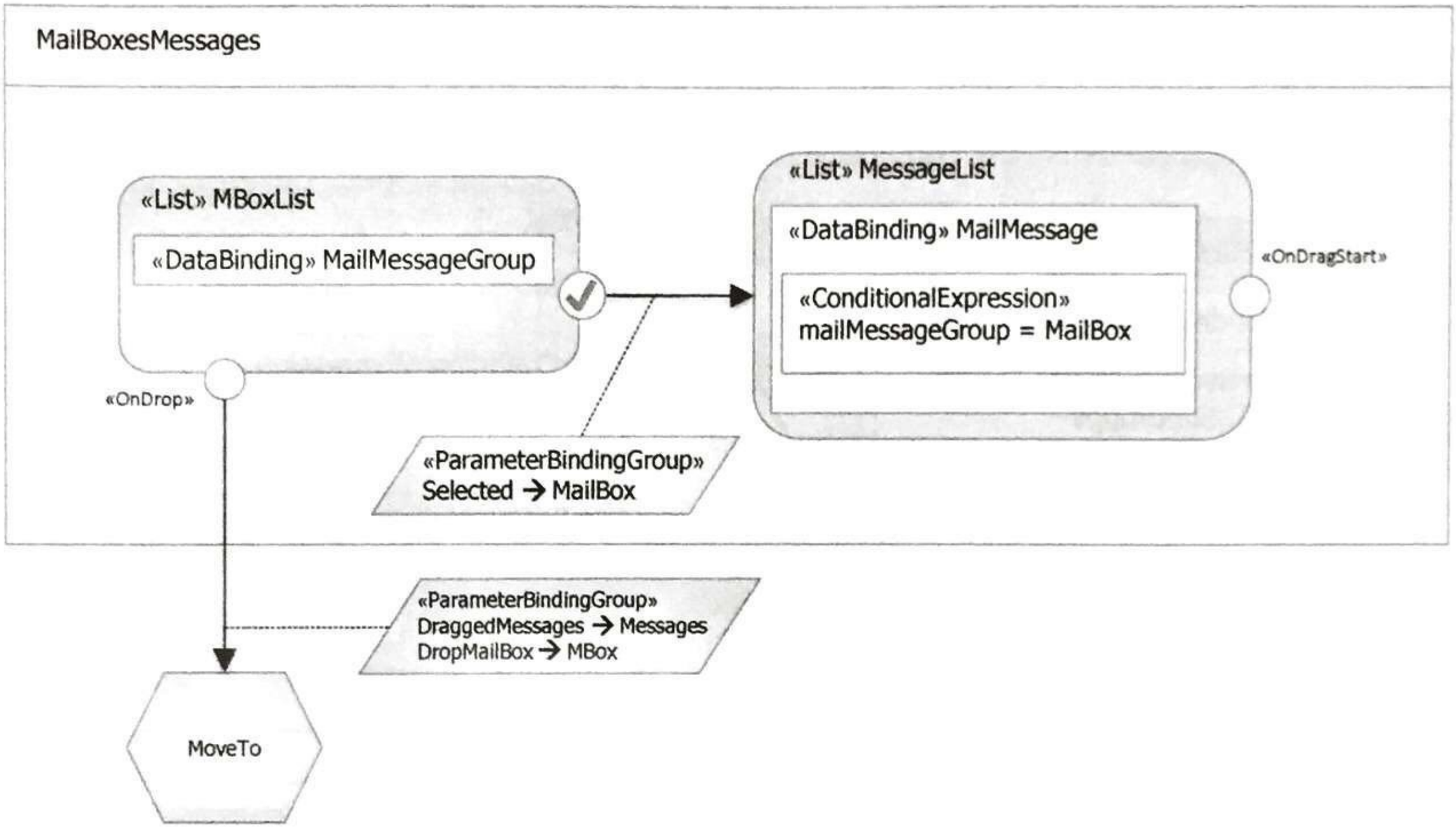


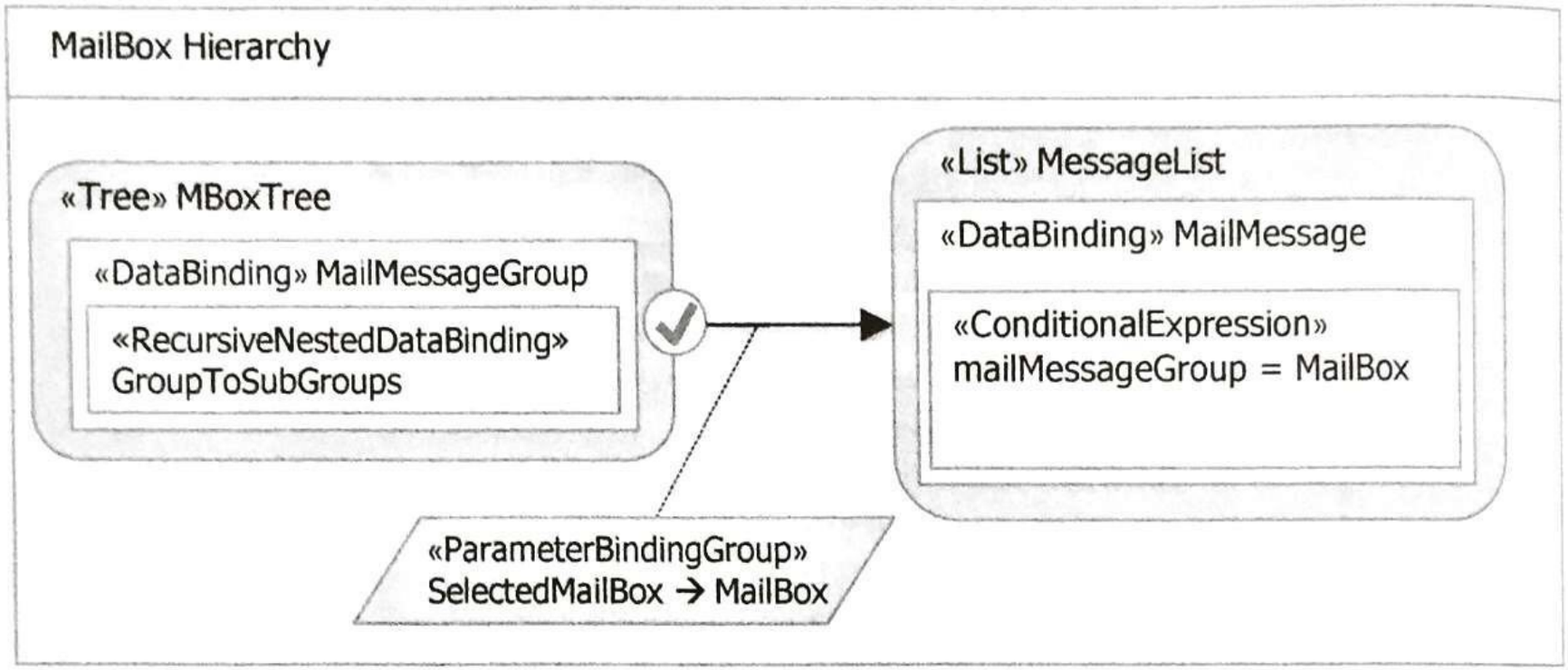
FIGURE 7.3

Extending IFML with drag and drop events.

7.1.2.2 Table

TABLE VIEWCOMPONENT

A **Table** is an extension of ViewComponent that displays tabular data and allows the user to edit them. It has a DataBinding element that typically refers to a class of the domain model. The attributes of the class are mapped to the columns of the table using the ColumnAttribute ViewComponentPart. The Table component can be associated with events of type CellUpdate, RowInsertion, and RowDeletion.



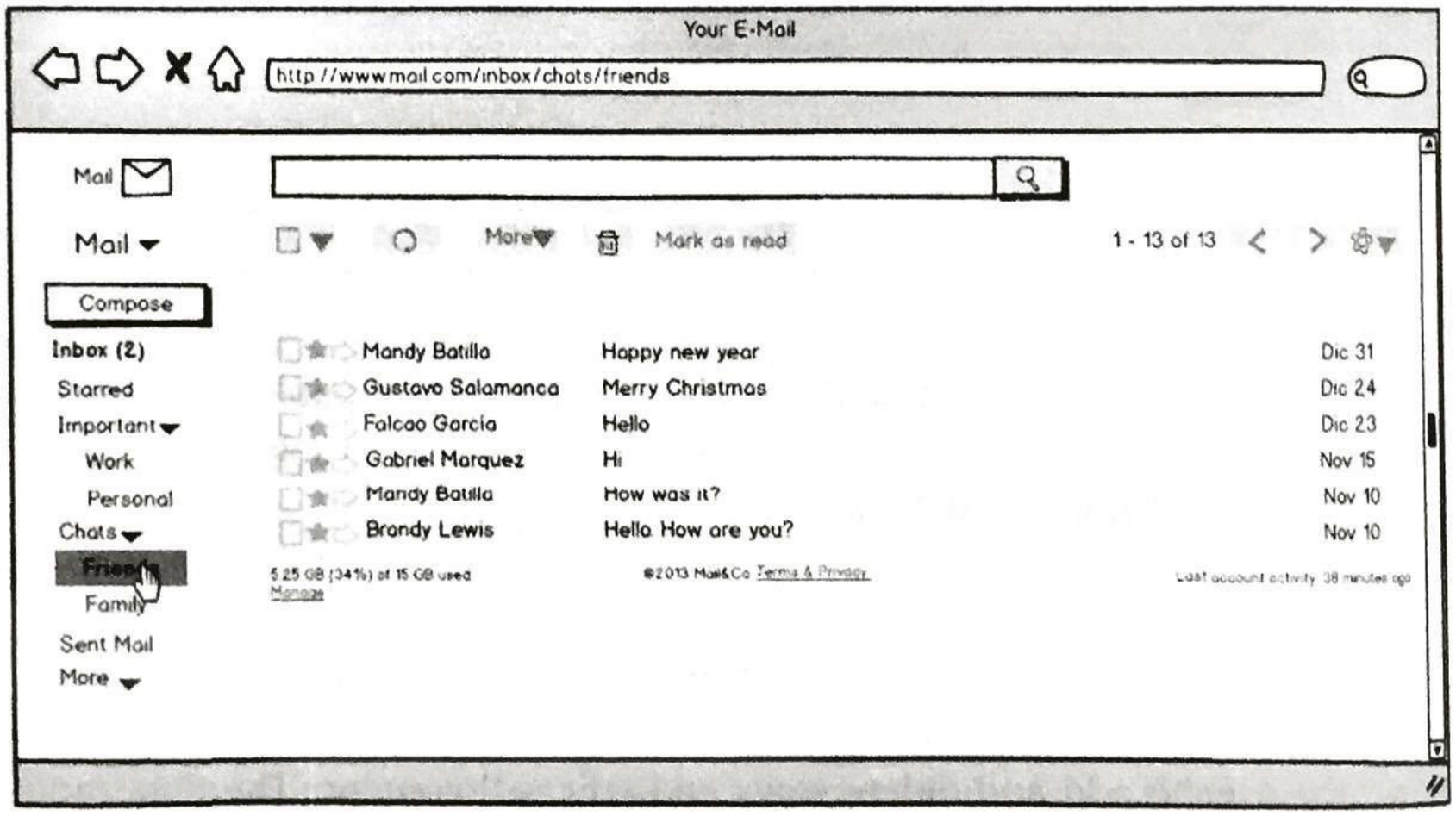
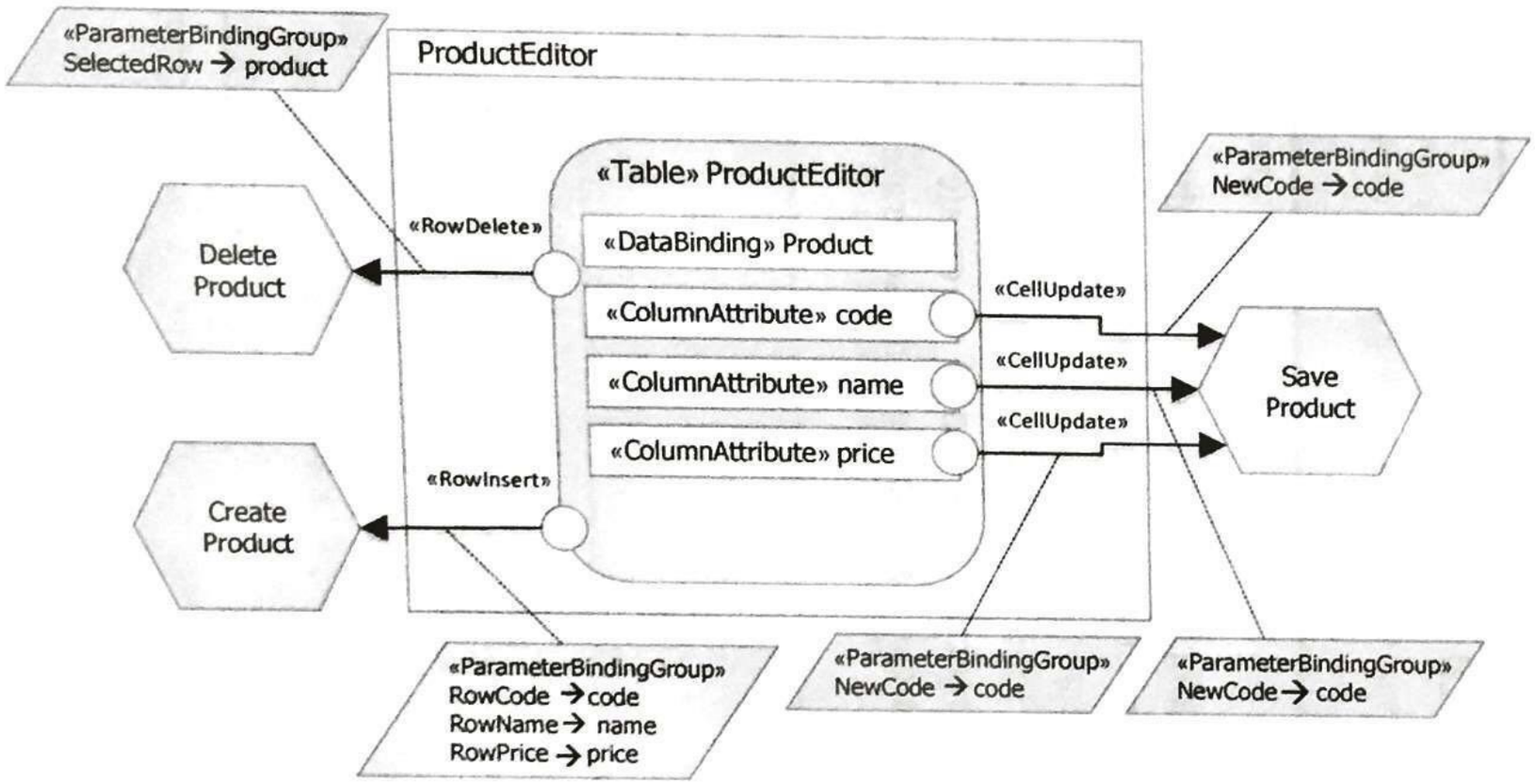


FIGURE 7.4

Example of usage of the Tree ViewComponent



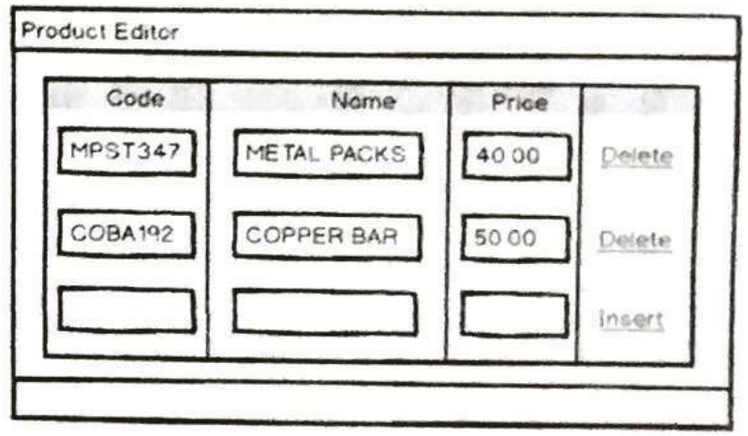


FIGURE 7.5

Example of usage of the Table ViewComponent.

7.1.3 COMPONENTPART EXTENSIONS

EDITABLESELECTIONFIELD

An **EditableSelectionField** extends the Field element and denotes an input field that is both editable and selectable.

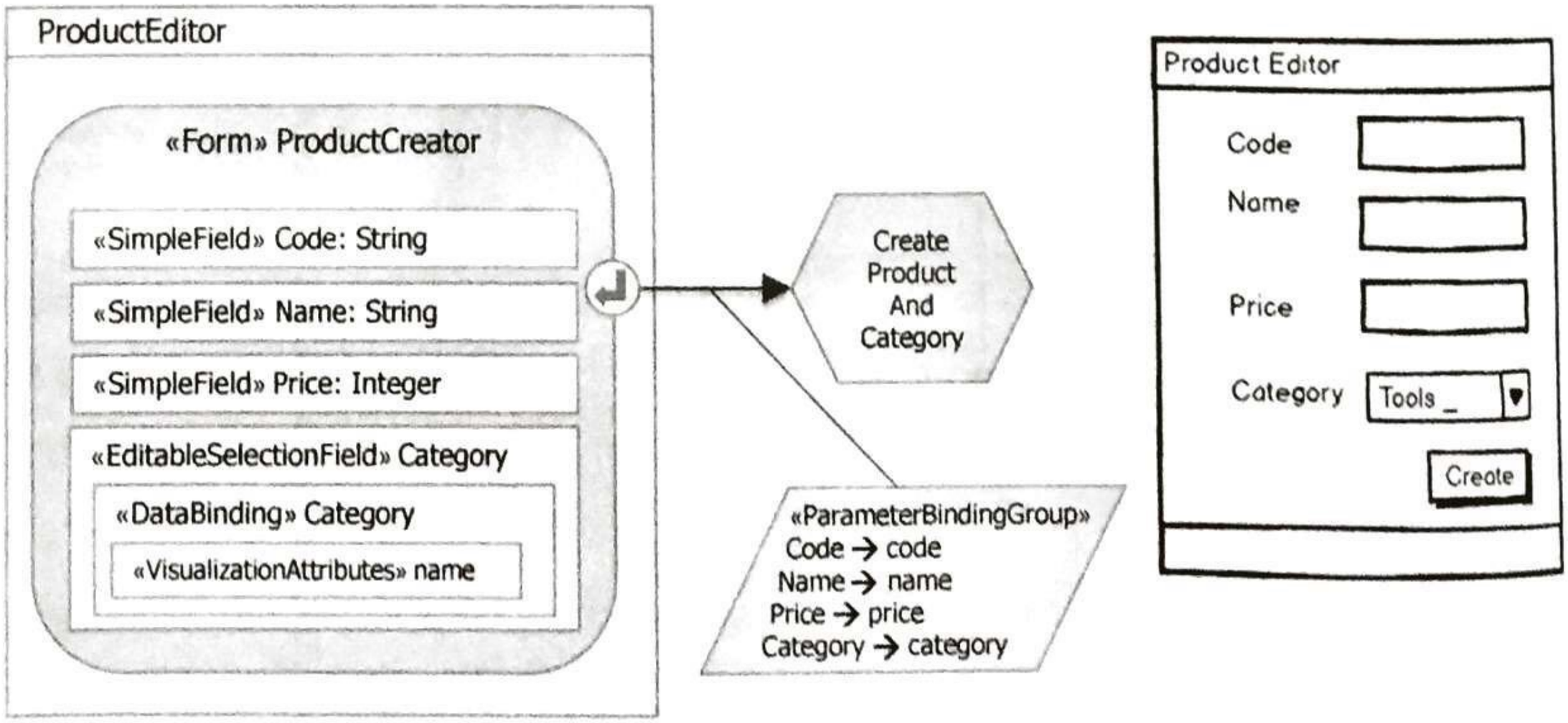


FIGURE 7.6

An example of usage of the EditableSelectionField.

7.2 WEB EXTENSIONS

7.2.1 CONTAINER EXTENSIONS: PAGES, AREAS, AND SITE VIEWS

PAGE

A **page** is an extension of ViewContainer that denotes an addressable web interface unit.

AREA

An **Area** is an extension of a disjunctive (XOR) ViewContainer that denotes a collection of pages or other areas, grouped according to an application-specific purpose.

SITEVIEW

A **SiteView** is an extension of a disjunctive (XOR) ViewContainer that denotes web application areas and pages grouped together according to an application-specific purpose, typically because they serve the needs of a UserRole.

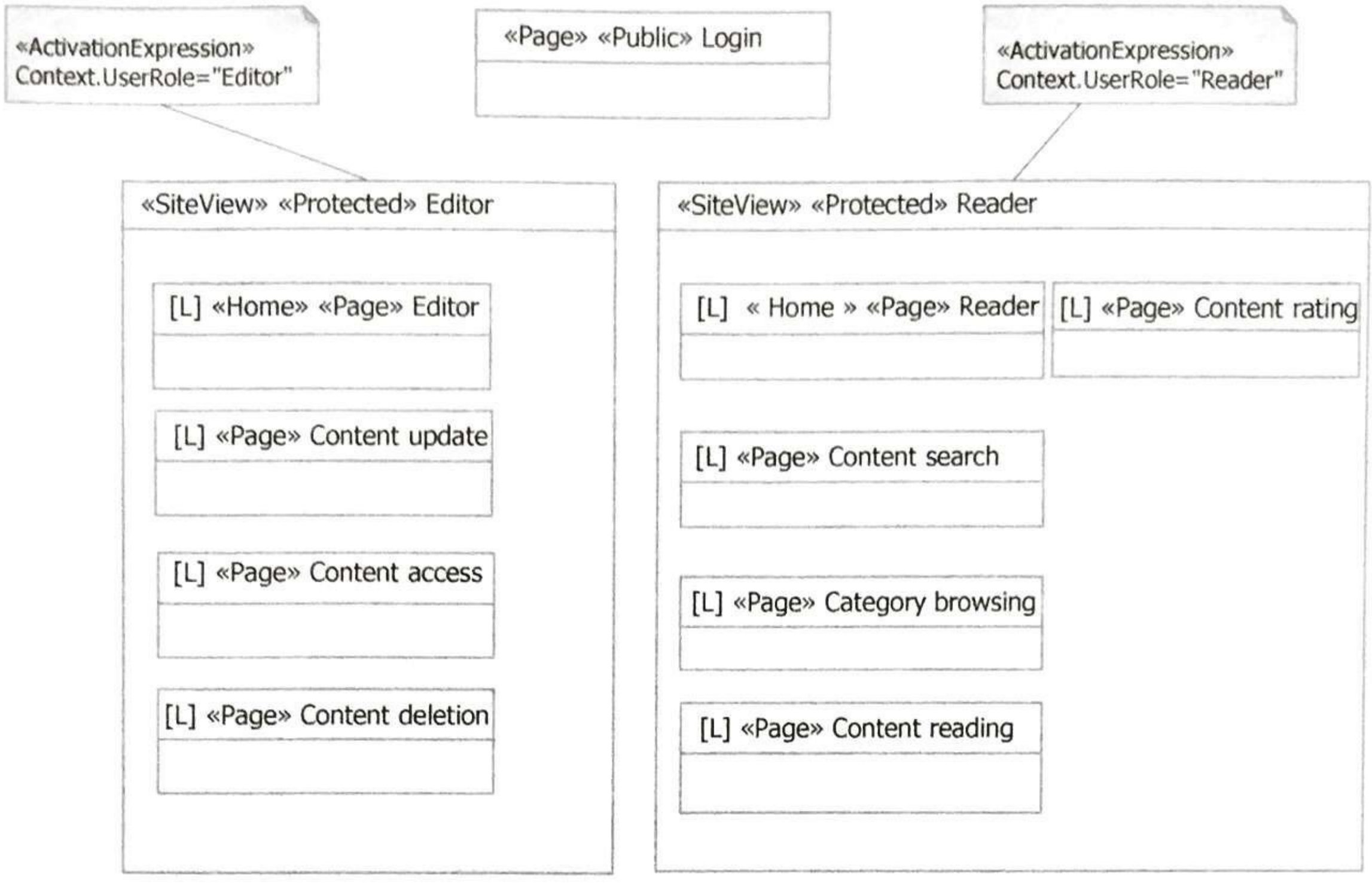


FIGURE 7.7

An IFML model of a typical web application for e-commerce.

7.2.2 EVENT AND INTERACTION FLOW EXTENSIONS

LINK

A **WebNavigationFlow** is an extension of a NavigationFlow that incorporates additional properties specific to hypertext links on the web.

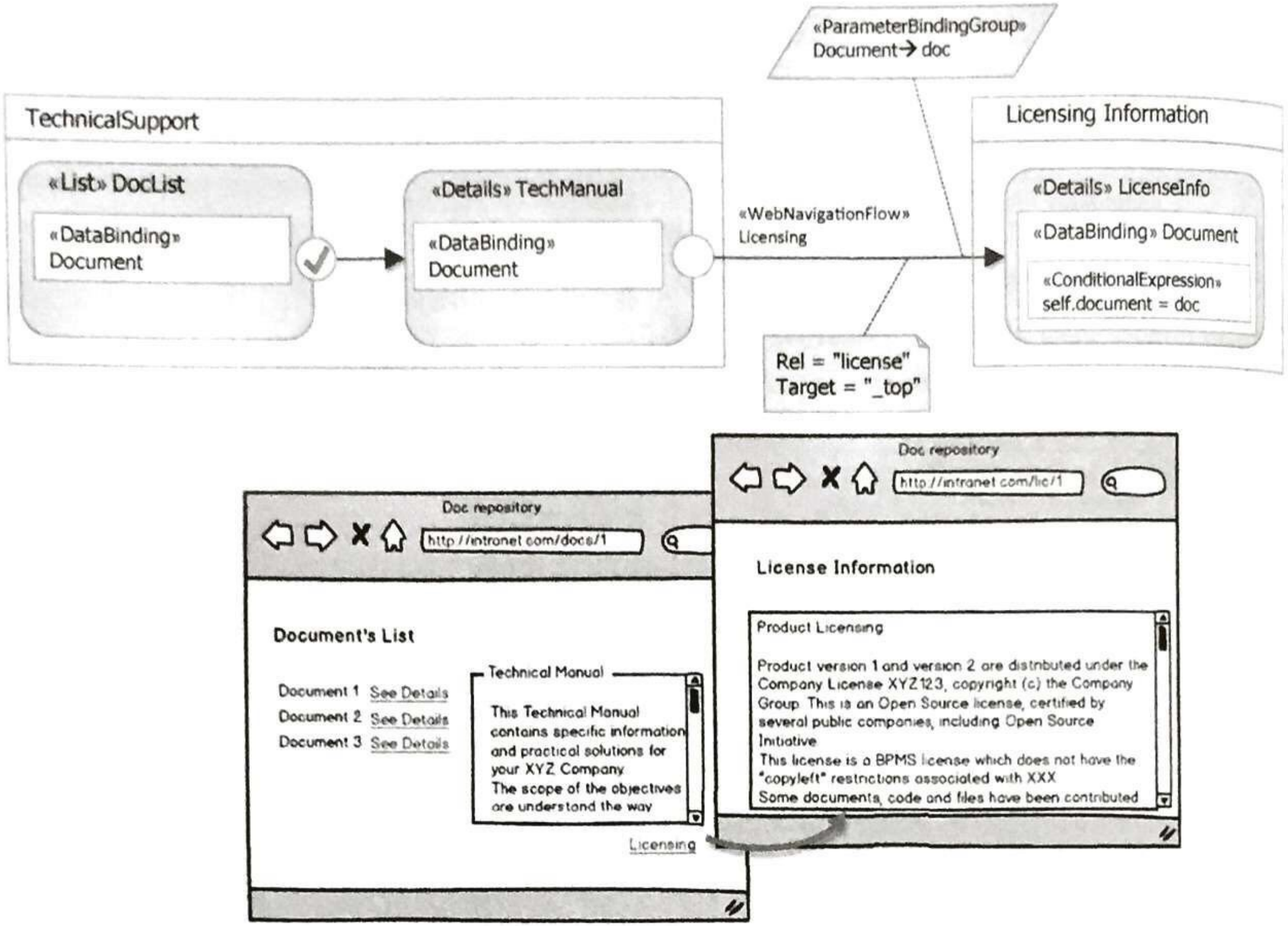


FIGURE 7.8

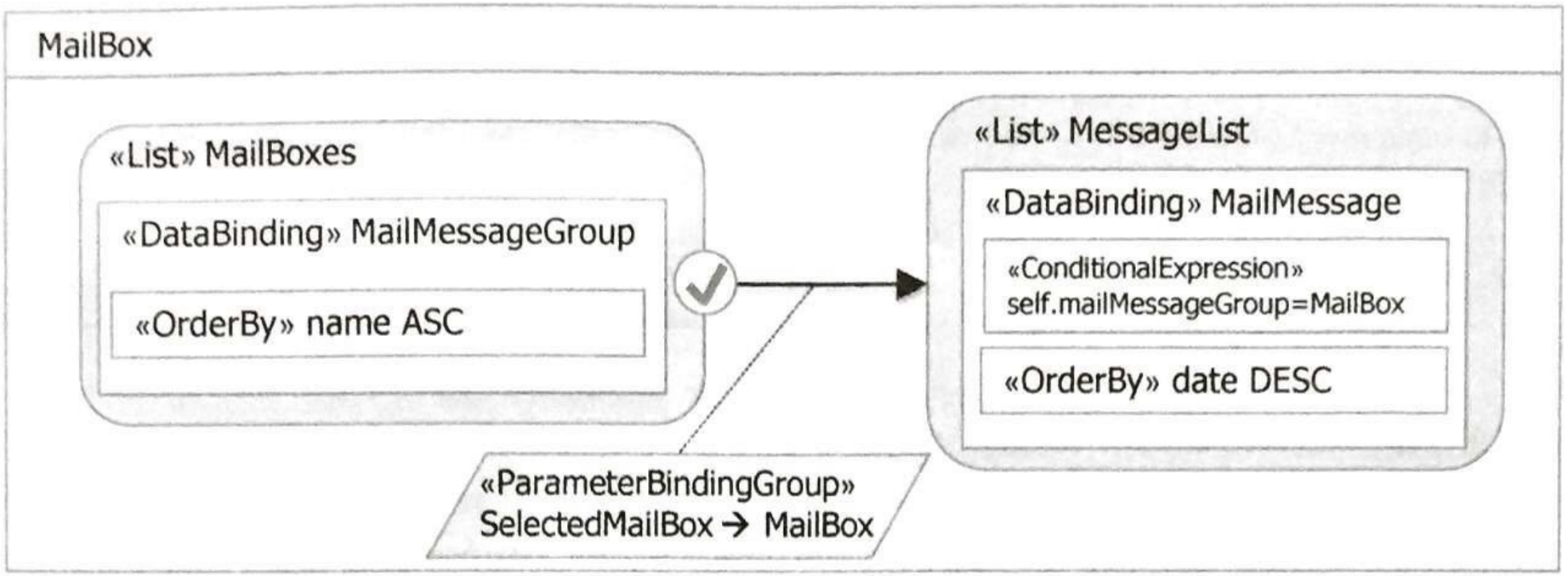
Example of usage of WebNavigationFlow.

7.2.3 COMPONENT EXTENSIONS

7.2.3.1 Dynamically-sorted list

DYNAMICSORTEDLIST

The **DynamicSortedList** is an extension of the List ViewComponent that allows the user to sort data using visualization attributes. The DynamicSortedList has a one-to-many association, named "SortAttributes," with the metaclass "VisualizationAttribute," which denotes the subset of the visualization attributes usable for sorting.



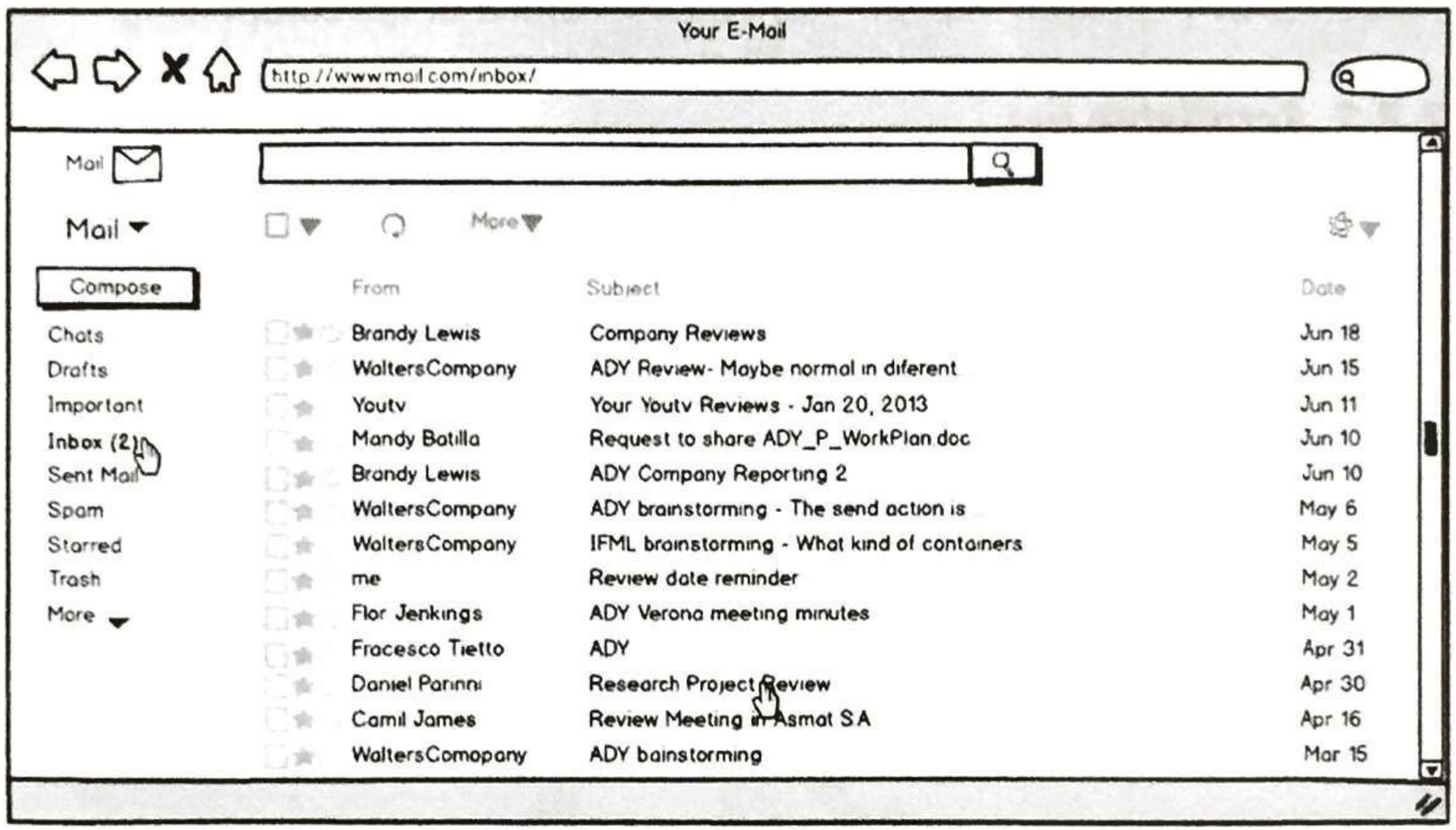


FIGURE 7.9

Example of usage of sorted list.

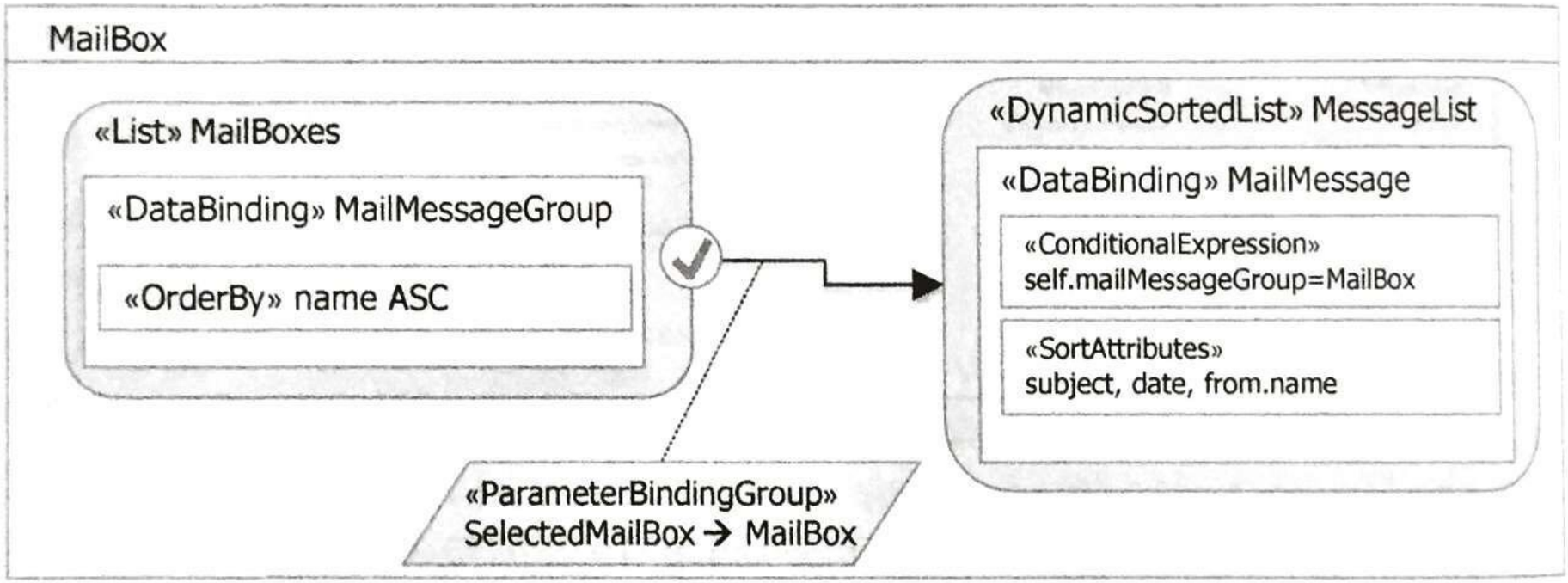


FIGURE 7.10

Example of usage of the DynamicSortedList.

7.2.3.2 Scrollable list

SCROLLABLELIST

The **ScrollableList** is an extension of the List ViewComponent that allows the user access ordered DataBinding instances grouped in blocks. The **ScrollableList** ViewComponent has an attribute called "block size" that specifies how many instances constitute a block. It also has an implicit parameter (named current), which holds the block currently in view, and implicit events for moving to the first, last, i-th, next, and previous block.

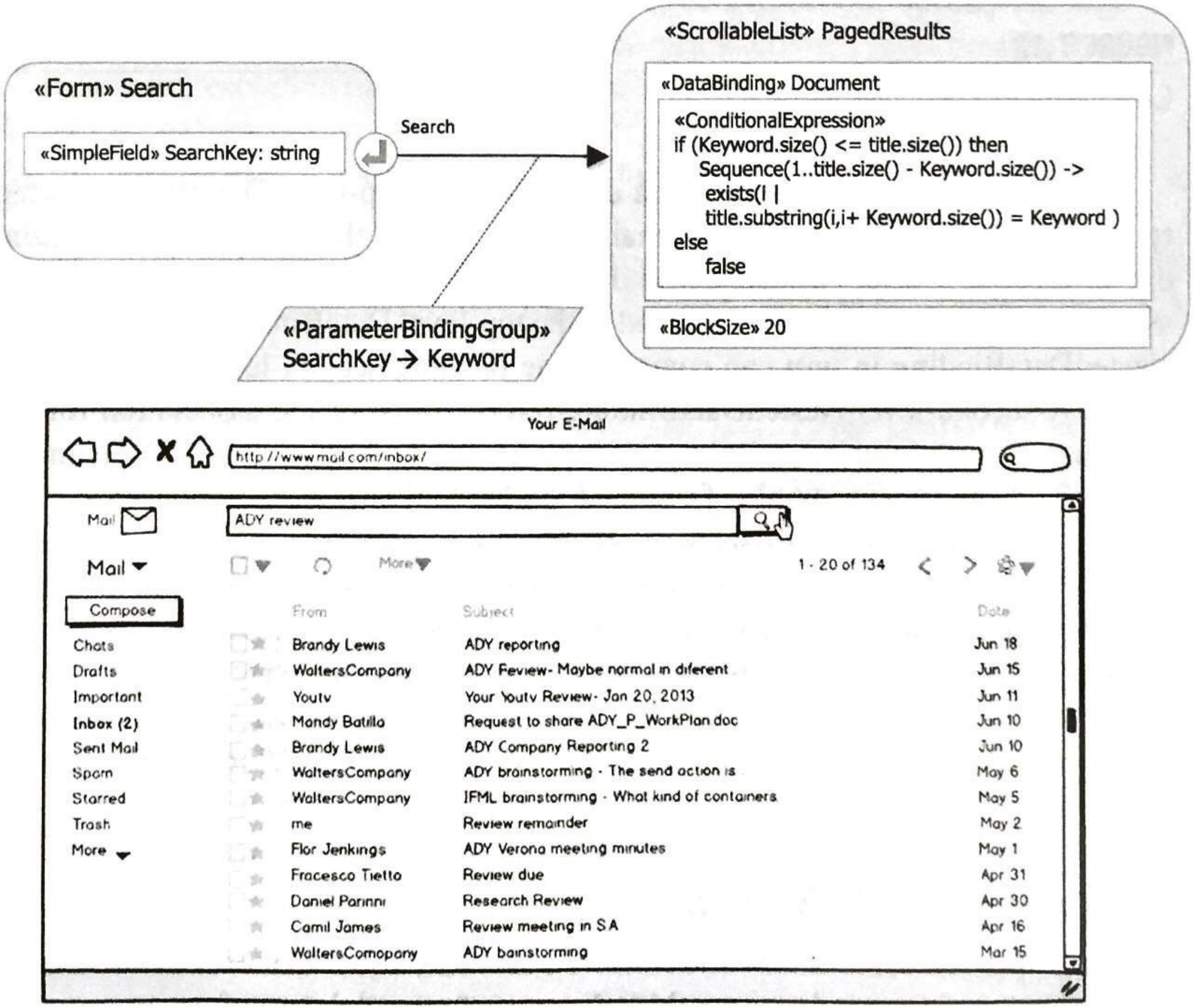


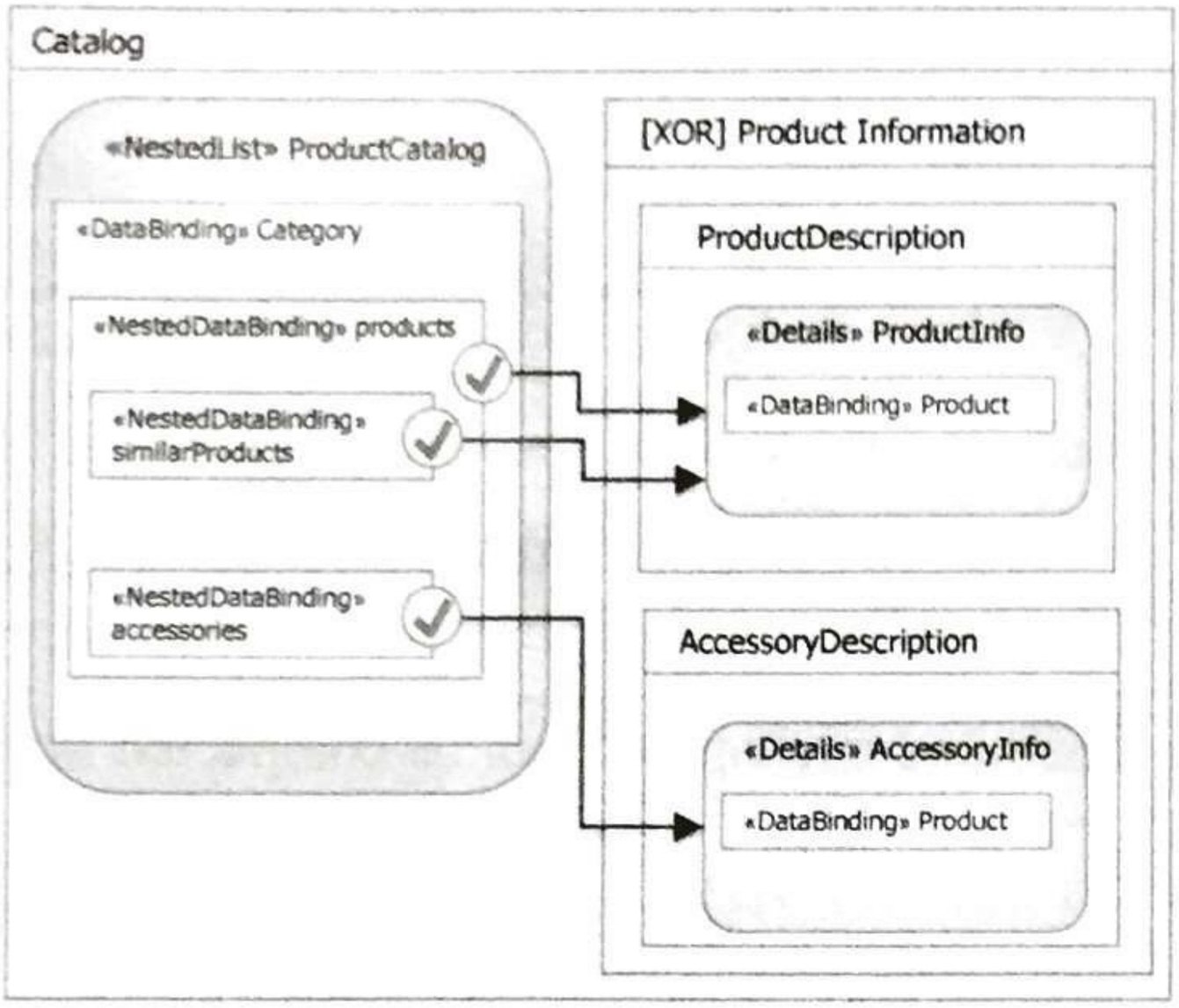
FIGURE 7.11

Example of usage of the ScrollableList.

7.2.3.3 Nested list

NESTEDLIST

The **NestedList** is an extension of the List ViewComponent that denotes the nesting of multiple lists, one inside another.



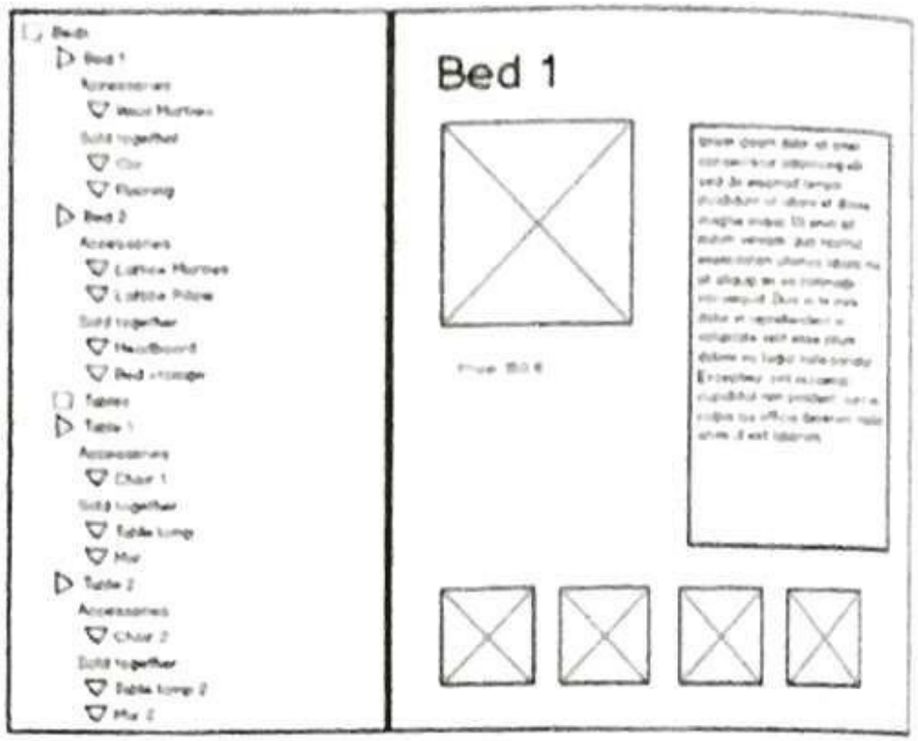


FIGURE 7.12

Example of usage of NestedList.

7.2.3.2 Scrollable list

SCROLLABLELIST

The **ScrollableList** is an extension of the List ViewComponent that allows the user access ordered DataBinding instances grouped in blocks. The **ScrollableList** ViewComponent has an attribute called "block size" that specifies how many instances constitute a block. It also has an implicit parameter (named current), which holds the block currently in view, and implicit events for moving to the first, last, i-th, next, and previous block.

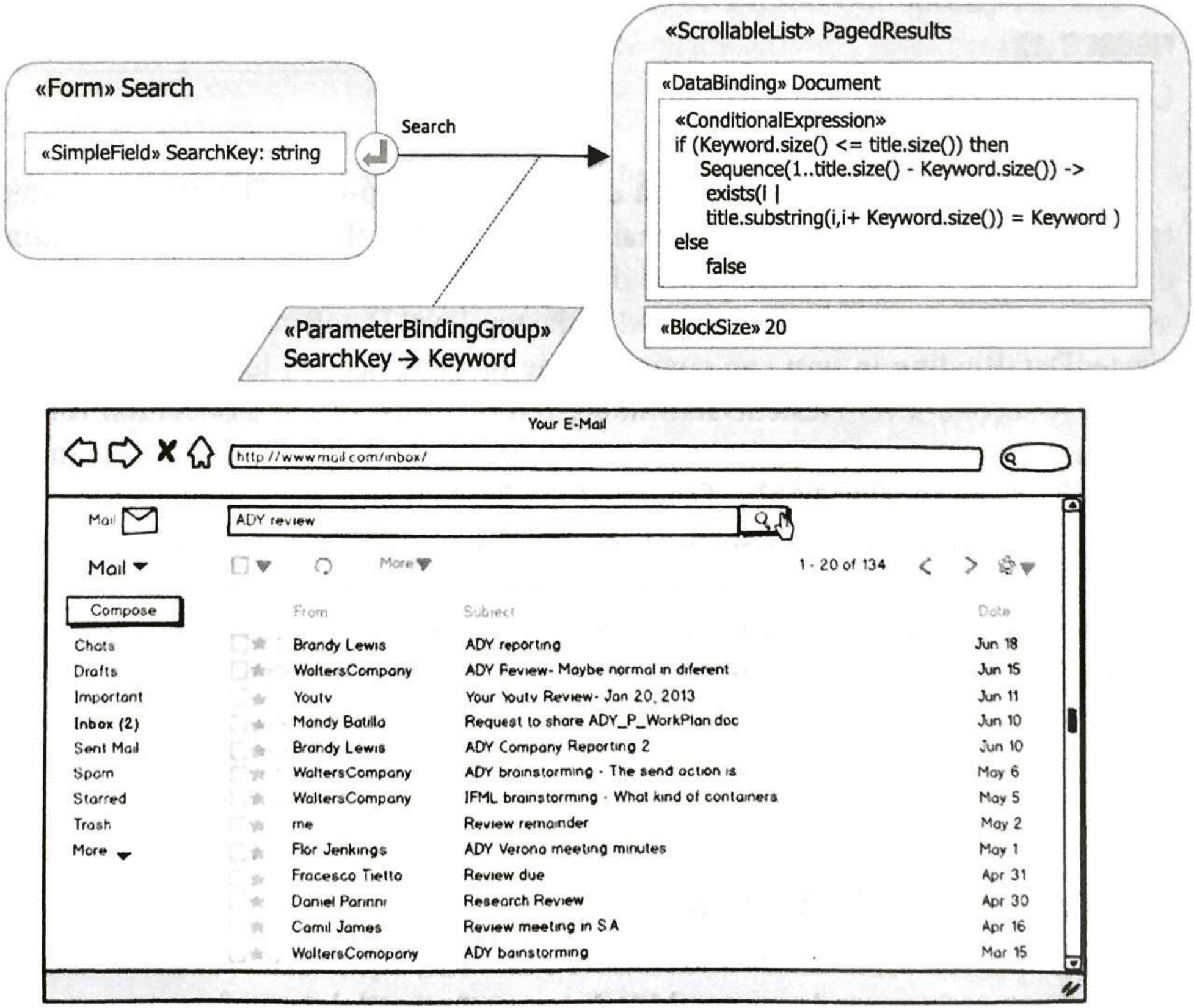


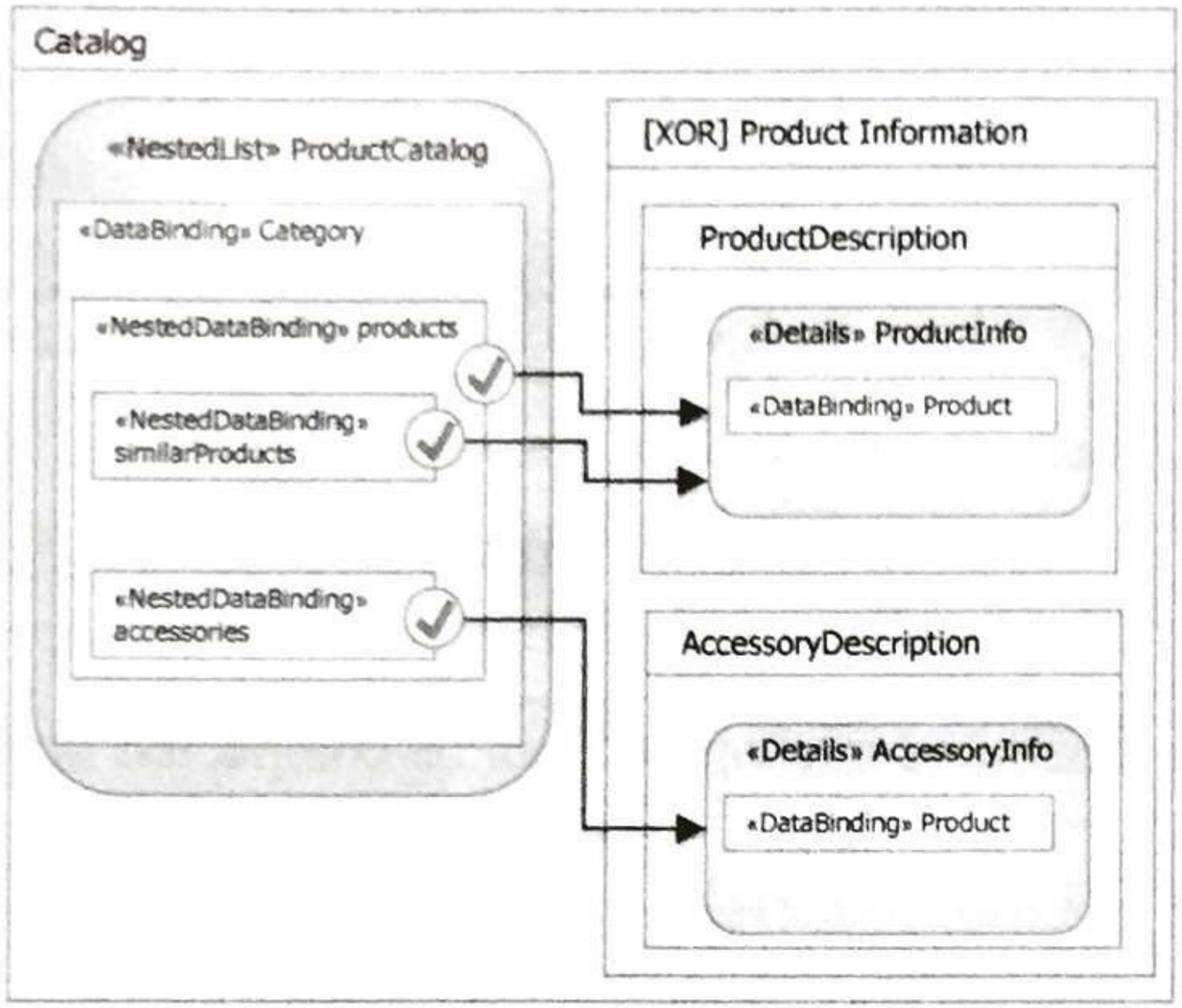
FIGURE 7.11

Example of usage of the ScrollableList.

7.2.3.3 Nested list

NESTEDLIST

The **NestedList** is an extension of the List ViewComponent that denotes the nesting of multiple lists, one inside another.



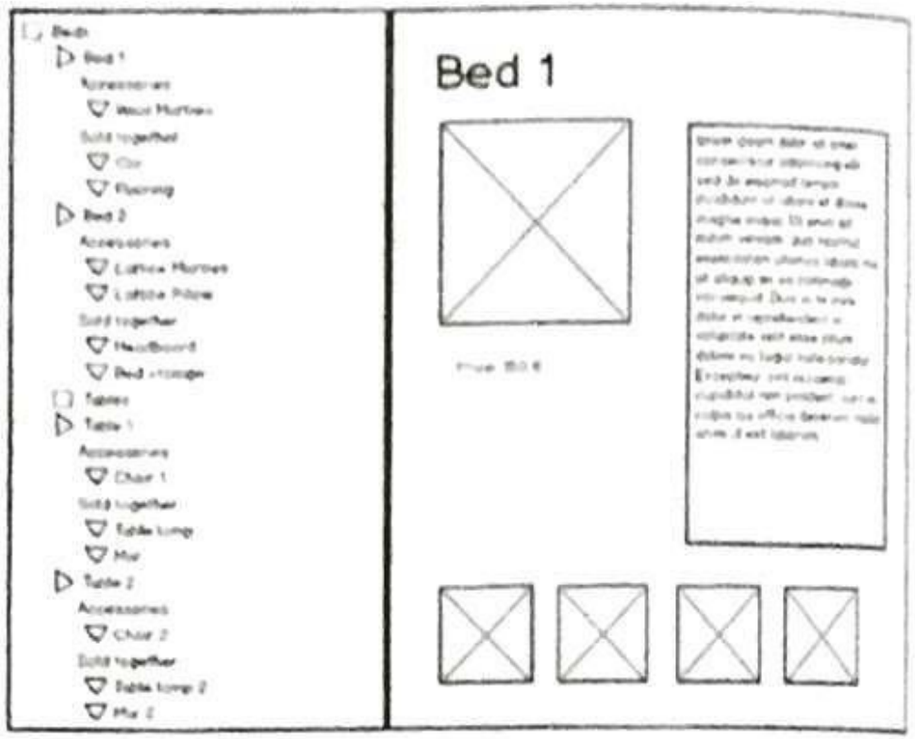


FIGURE 7.12

Example of usage of NestedList.

7.3 MOBILE EXTENSIONS

7.3.2 CONTAINERS EXTENSIONS

SYSTEM VIEWCONTAINER

A ViewContainer stereotyped as **«system»** denotes a fixed region of the interface, managed by the operating system or by another interface framework in a cross-application way.

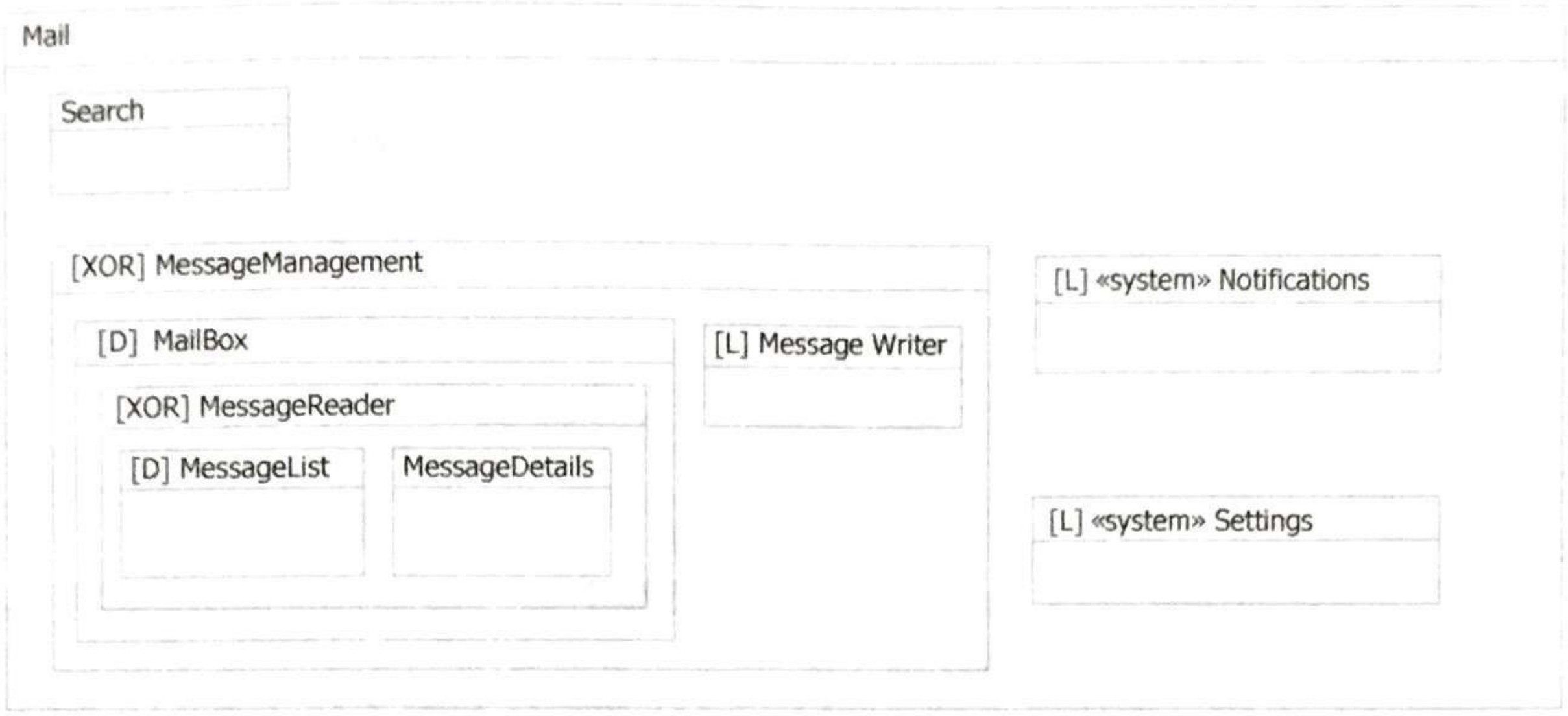


FIGURE 7.13

Example of «system» ViewContainers.

7.3.4 CAMERAS AND SENSORS

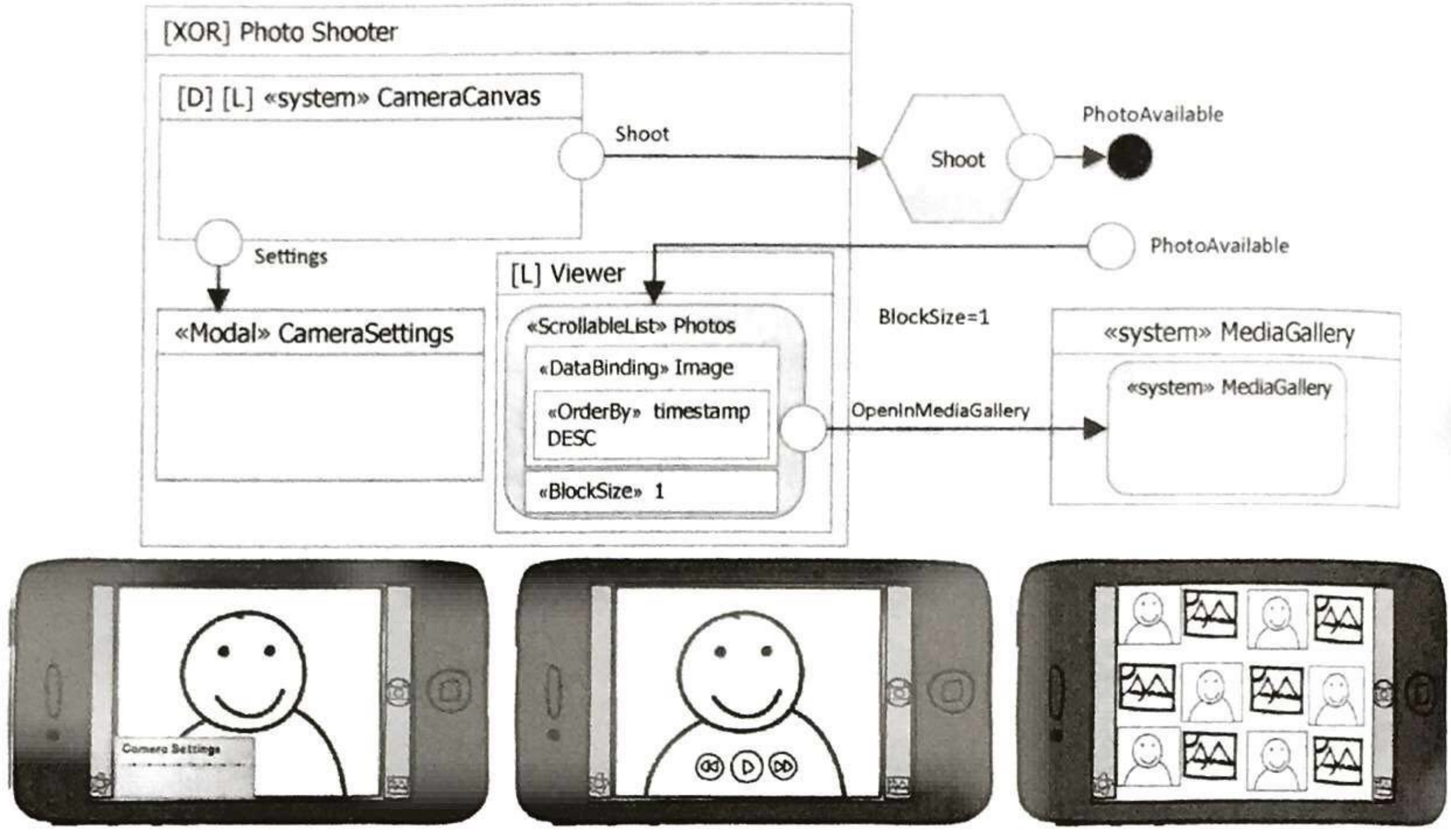


FIGURE 7.14

Example of usage of the camera and media gallery.

7.3.5 COMMUNICATION

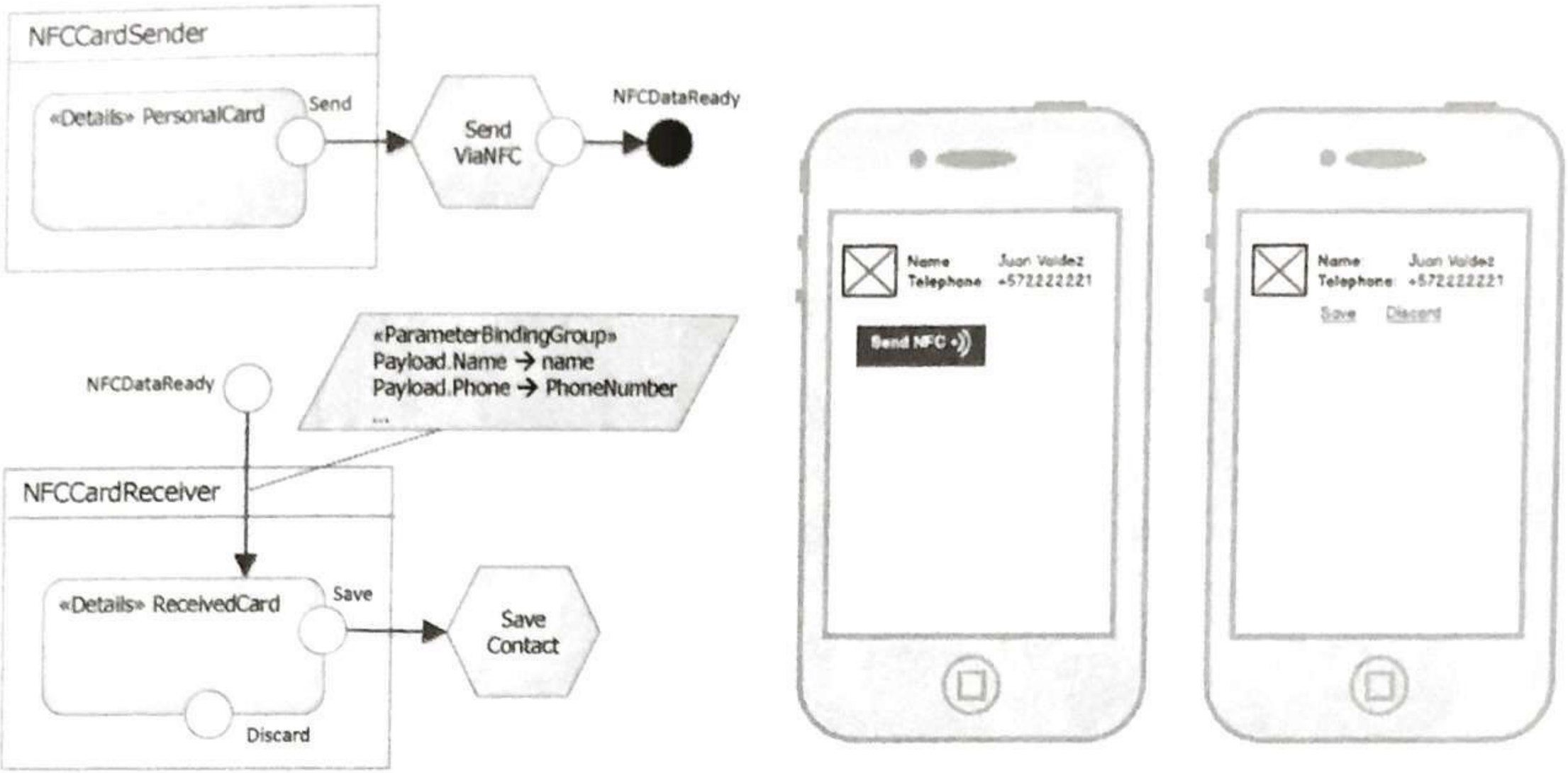


FIGURE 7.15

Example of usage of NFC data exchange

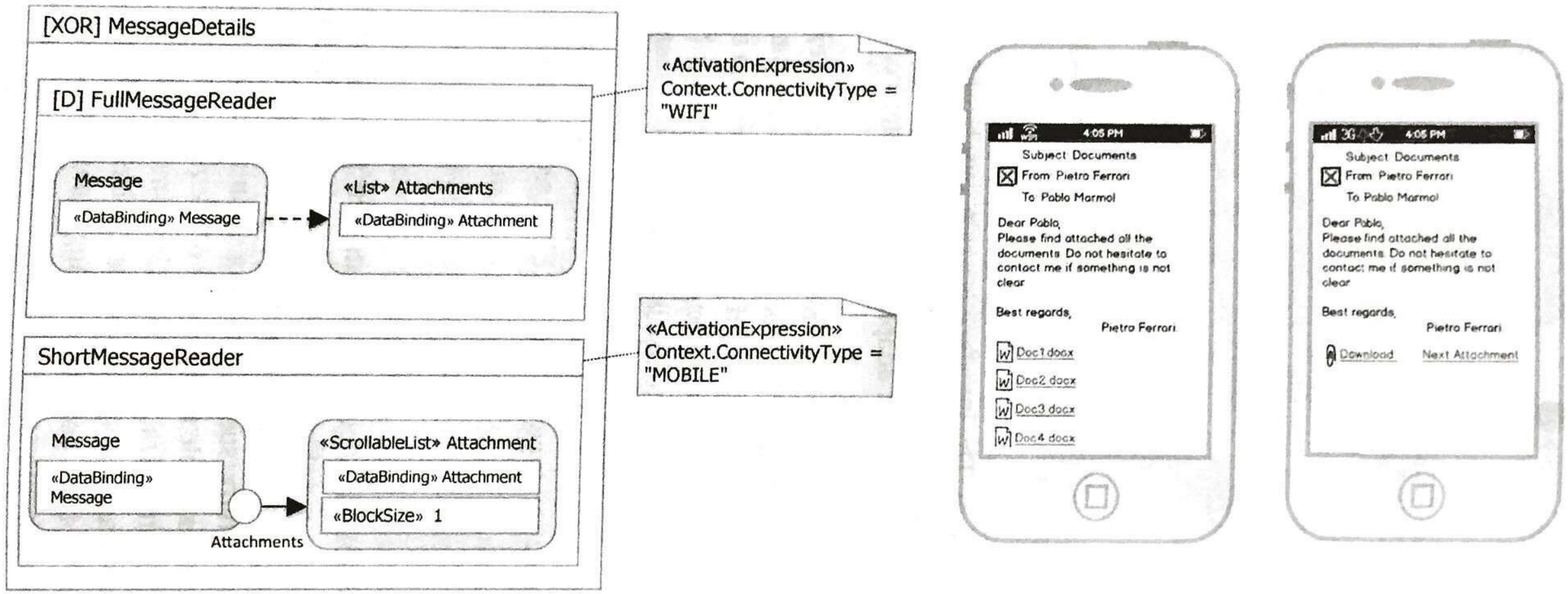


FIGURE 7.16

example of interface adaptation to network capacity

7.3.6 POSITION

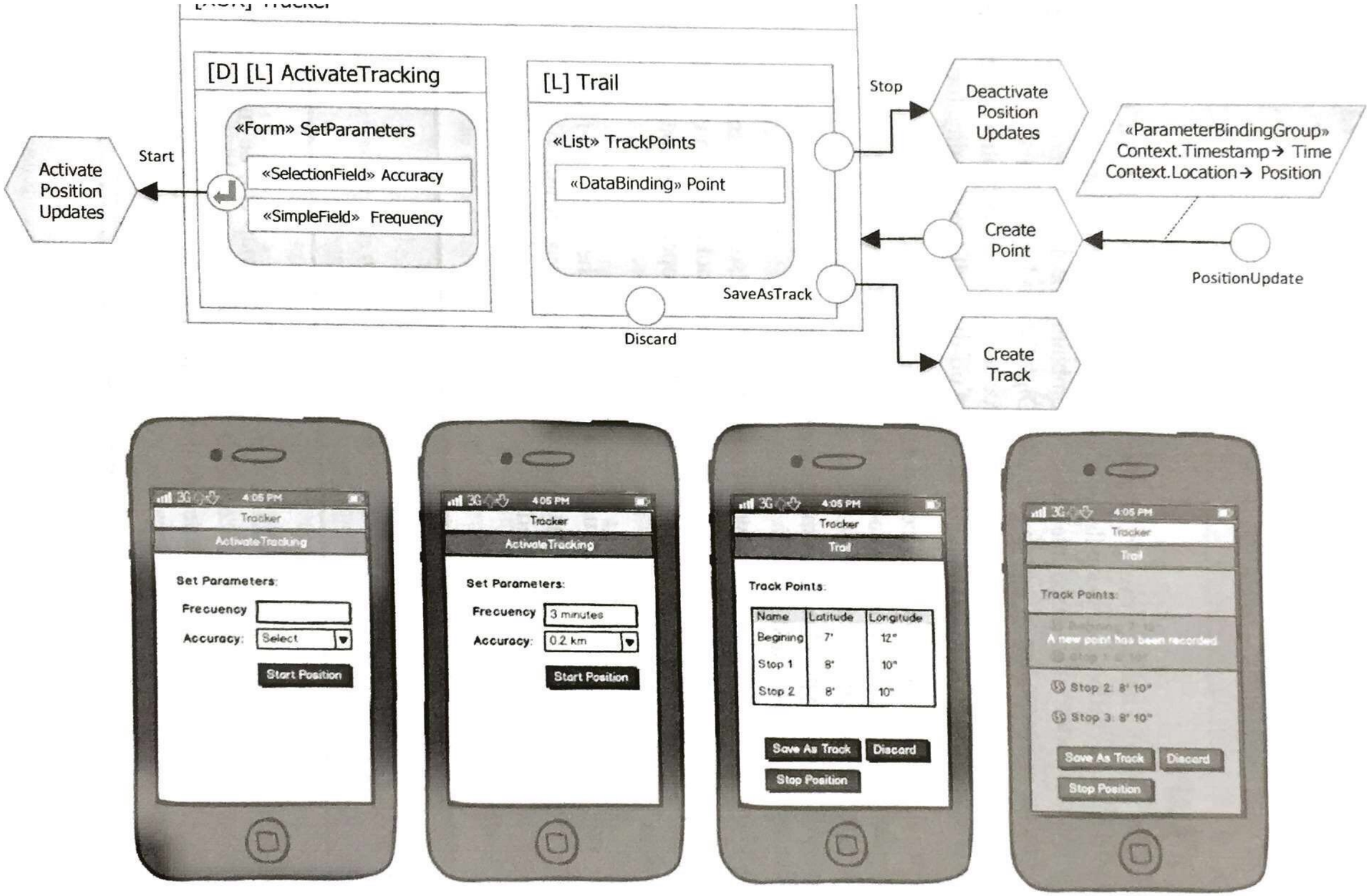


FIGURE 7.17

Example of usage of the position

7.3.7 MAPS

MAPVIEW

A **MapView** is an extension of ViewContainer that denotes a map view. It supports the events for panning and zooming and for changing the map type and the camera parameters.

MARKER

A **Marker** is an extension of ViewComponent usable in MapView containers that denotes that the underlying DataBinding instances possess a location attribute that is displayable in a map view. It supports the events for selecting, dragging, and dropping.

PATH

A **Path** is an extension of the List ViewComponent usable in MapView containers that presents underlying DataBinding instances (that must possess a location attribute) as a polyline in a map view. It supports events for selecting the entire path or a single point on it.

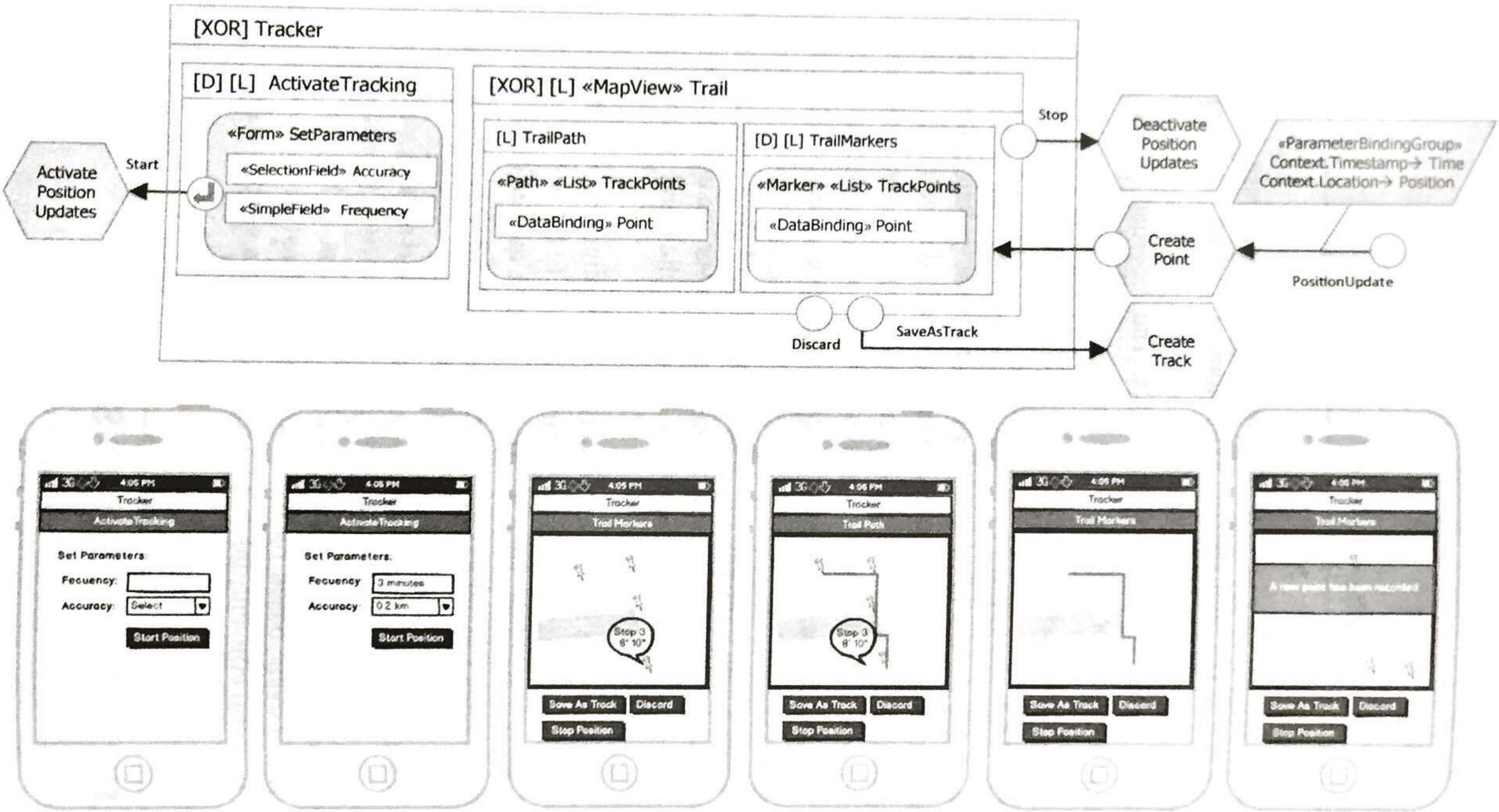


FIGURE 7.18

Example of usage of the MapView ViewContainer.

7.3.8 GESTURES

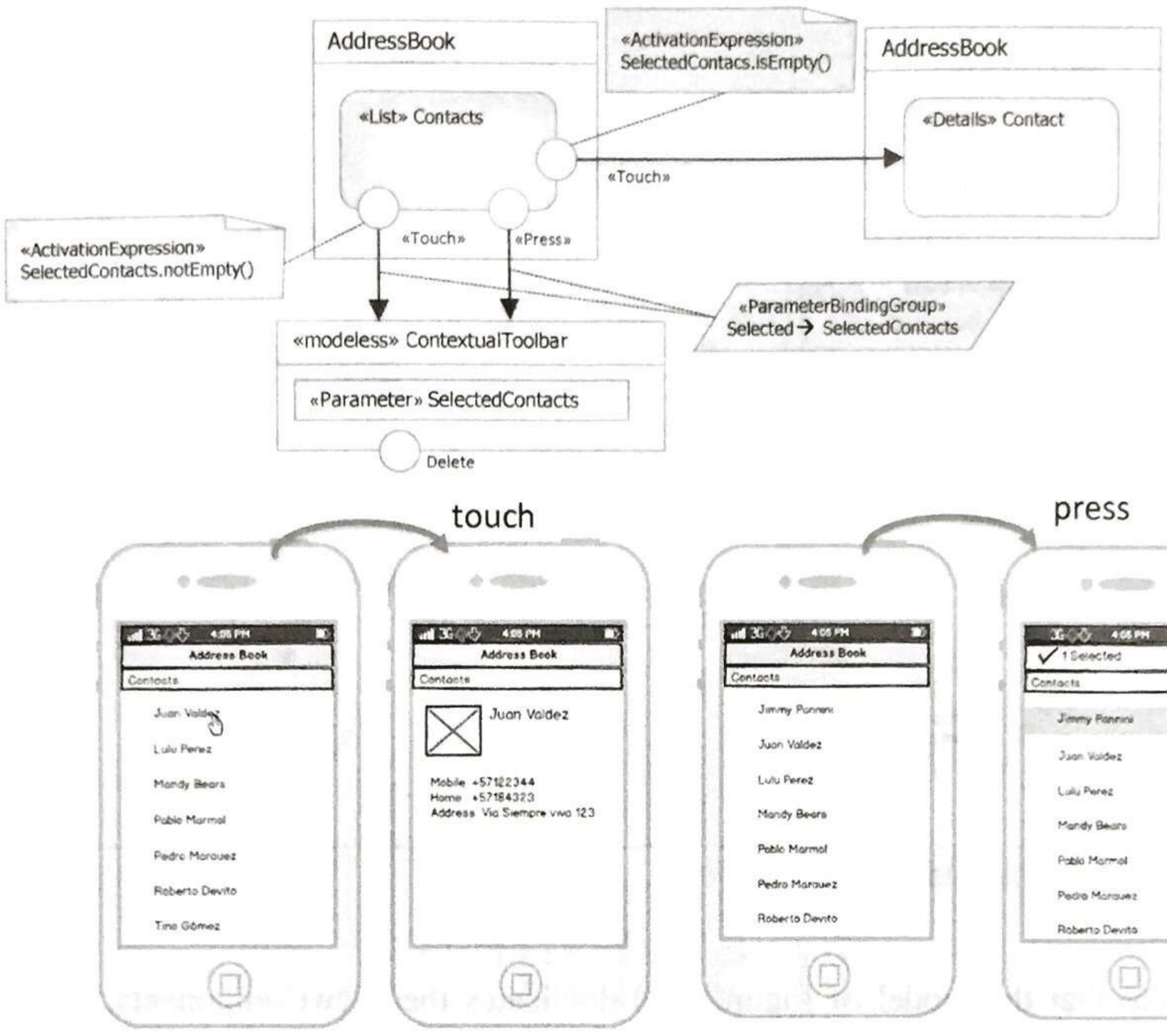


FIGURE 7.19

Example of touch and press event handling.

7.4 MULTISCREEN EXTENSIONS

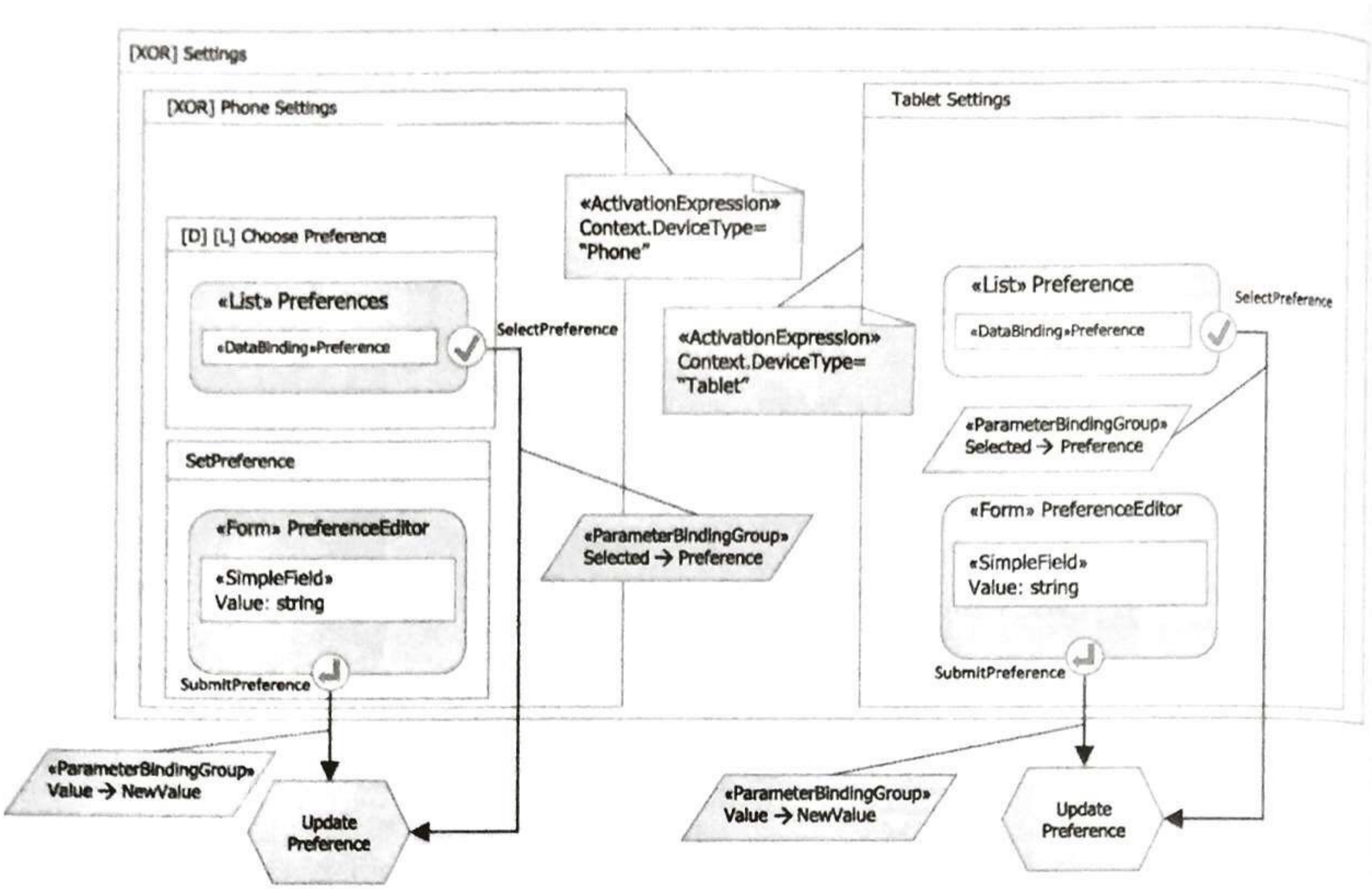


FIGURE 7.20

Example of flexible interface composition.

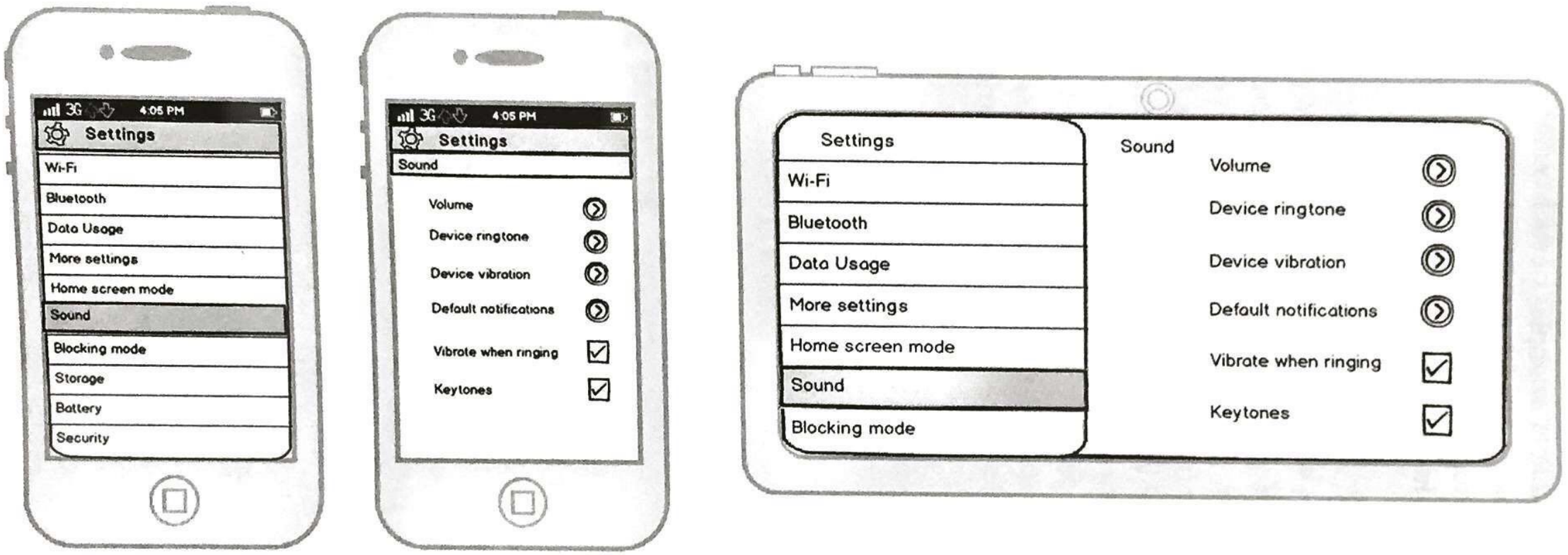


FIGURE 7.21

Mock-up of the adaptable composition of Figure 7.20.