User Manual

AutoCRUD is basically a CRUD scaffolding tool for WebRatio. It increases development productivity by providing WebRatio engineers with a pattern-based development tool for IFML. Therefore, engineers basically select a data entity of their project and automatically generate the IFML specification for a CRUD operation by instantiating the right pattern. Although this work focuses on CRUD patterns, every development team may define new patterns or modified them to build its own repository.

Contents of this manual:

Installation

Installation steps:

1. AutoCRUD is a WebRatio plugin, so you must install previously WebRatio Web Platform Community Edition.
2. Download AutoCRUD v3.0.0 jar file and move it into WebRatio plugins folder.
3. (Re)start WebRatio.

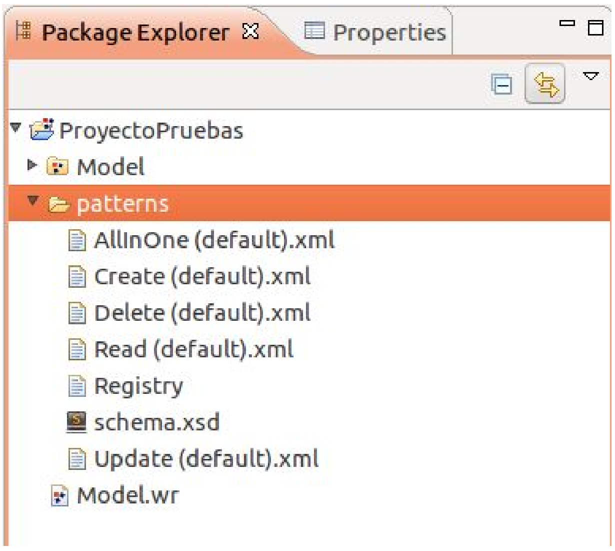
Project setup

First, you need to create a Web Project in Web Ratio with default settings, because this tool works inside a Web project scope.

AutoCRUD loads dynamically, when launched, the available patterns for the working project. Therefore, it expects to find a folder named “patterns” that contains an XMI file for each pattern specification.

Inside the plugin download bundle you may find a patterns folder with the specification of CRUD patterns. As a starting point, it is recommended copying this folder into the Web project folder in WebRatio, as the next figure shows.

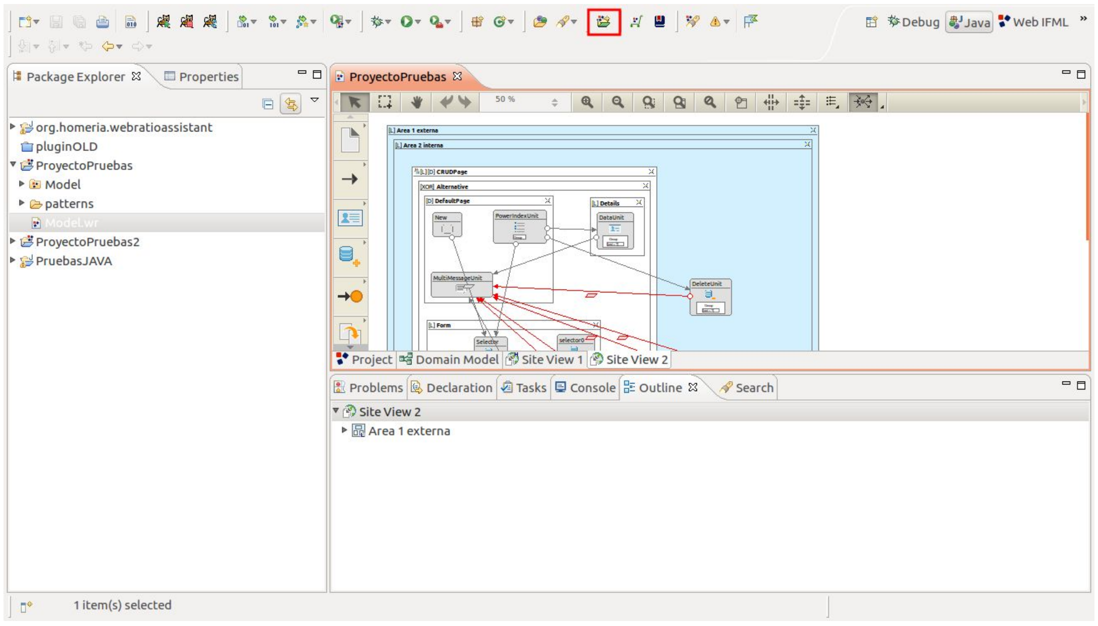
Additionally, note that the Web Project should posses one container at least. If there is none, then you have to create a new Site View.



Using the tool

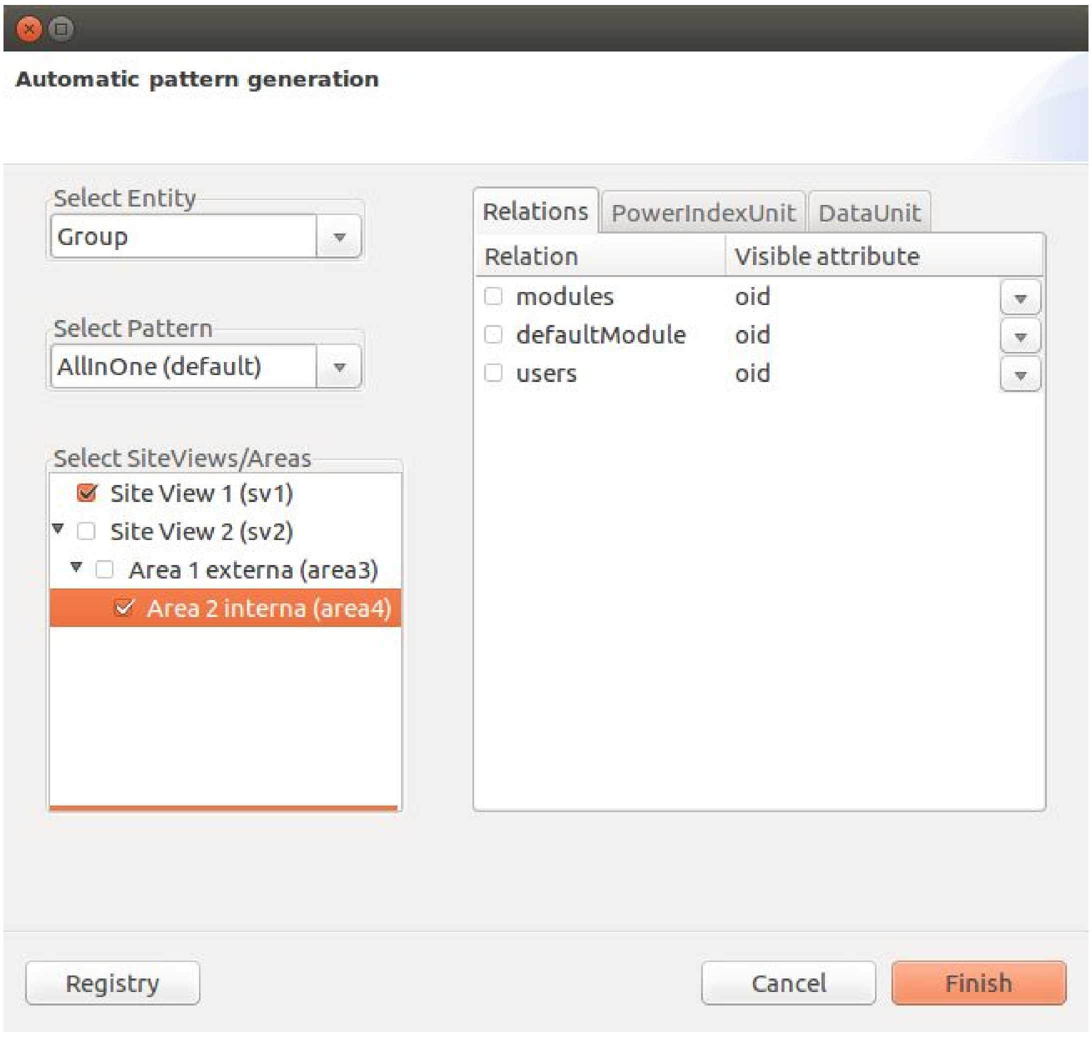
Example 1: Generate All CRUD operations for a data entity

The default data model of a Web Project is used for this example, which contains three main entities: user, group and ...
By clicking AutoCRUD icon at WebRatio toolbar (highlighted in the figure), this tool is launched (or Ctrl+F3).



When launched, AutoCRUD displays its pattern instantiation dialog, which is formed by two main sections: (1) at the left side users can select the target data entity, the pattern to instantiate (this dropdown list is dynamically populated) and the IFML containers (Areas or Site Views) in which the elements of the pattern must be generated; and (2) at the right side users may provide the desired parameters for pattern instantiation according to the data entity relations and the attributes to include in the Power and Data units (tabs are dynamically generated according to the selected pattern). For this example, Group data entity, AllInOne pattern and several containers (Site View 1 and Area 2) have been selected at the left side. At the right side, three tabs have been created to allow users to parameterize conveniently the instance of the selected pattern. Those three tabs are:

- Relations tab displays data relations of the selected data entity. This tab appears when the selected pattern contains units depending on data relations.
- PowerIndexUnit tab allows selecting which data entity attributes will be displayed on the listing. This tab appears when the selected pattern contains a Power Index Unit.
- DataUnit tab allows selecting which data entity attributes will be displayed on the detail view. This tab appears when the selected pattern contains a Data Unit.


Automatic pattern generation

Select Entity: Group

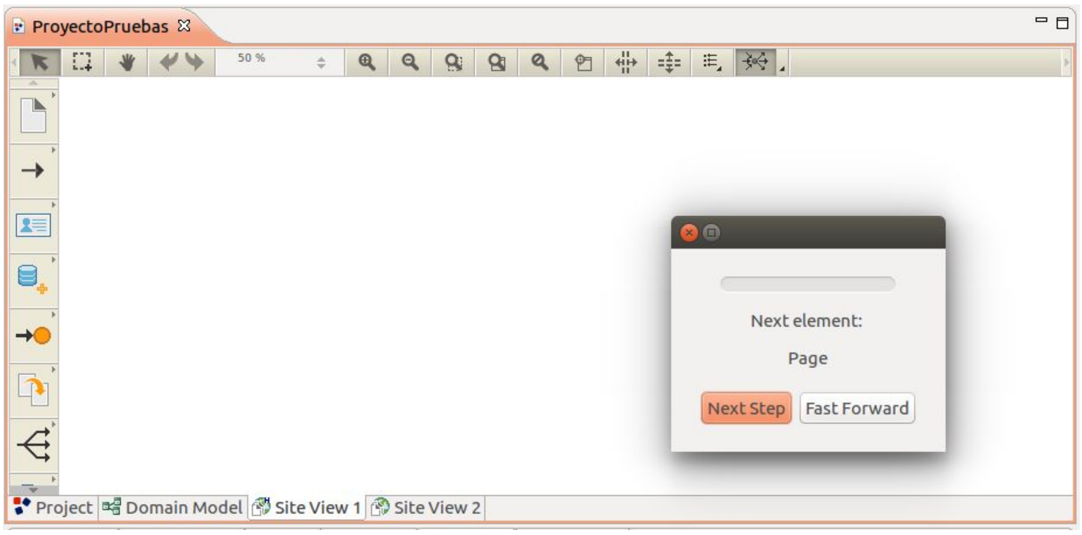
Select Pattern: AllInOne (default)

Select SiteViews/Areas
- [✓] Site View 1 (sv1)
- [ ] Site View 2 (sv2)
  - [ ] Area 1 externa (area3)
    - [✓] Area 2 interna (area4)

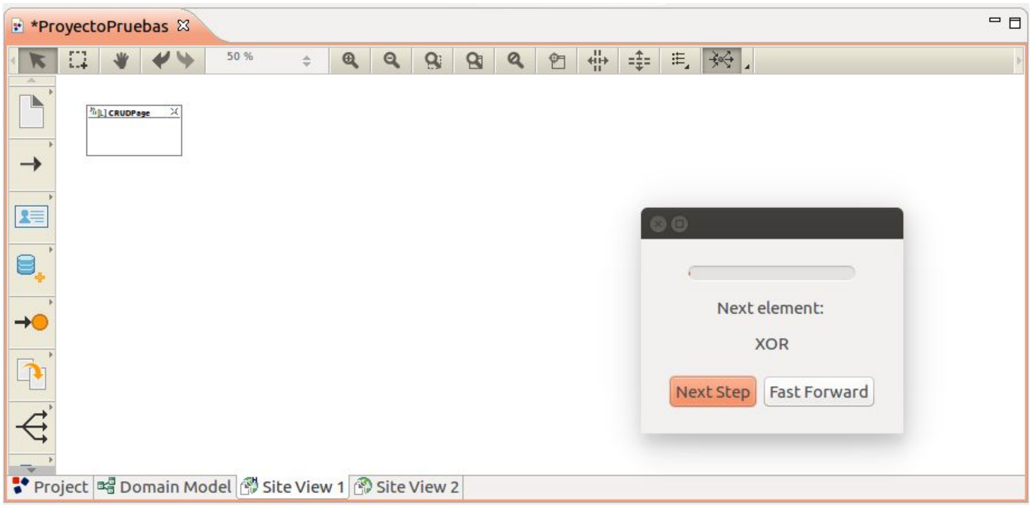
Relations | PowerIndexUnit | DataUnit

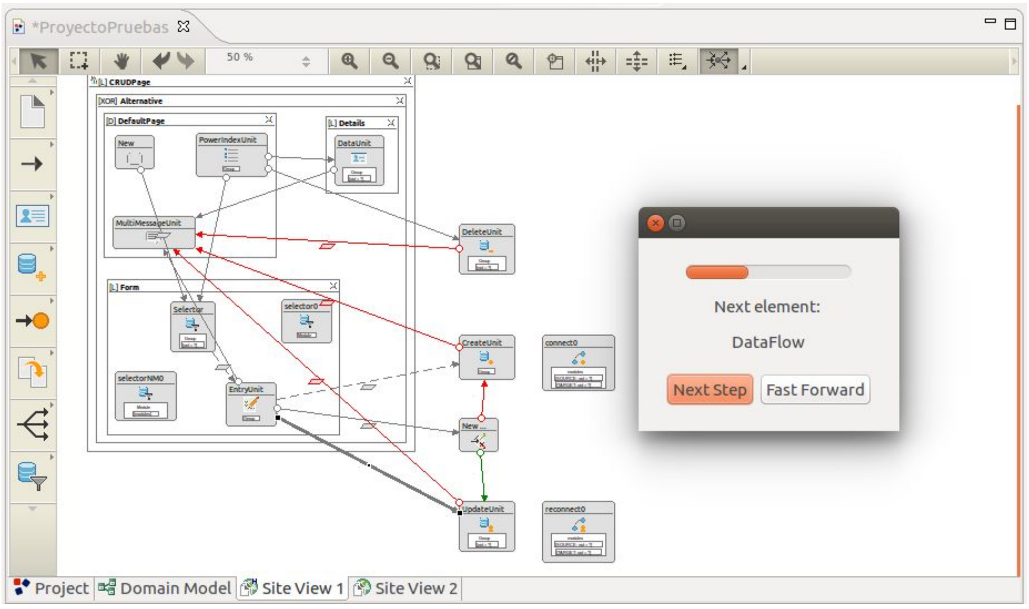
Relation	Visible attribute
[ ] modules	oid
[ ] defaultModule	oid
[ ] users	oid

When all desired attributes and relations are selected, you have to click Finish button in order to proceed with the IFML elements generation. Next figure shows the IFML generation dialog that allows users to select between two different generation modes: step-by-step (IFML elements are generated one by one) or fast forward (IFML elements are generated in one step). That dialog also shows a progress bar and the next element to generate.

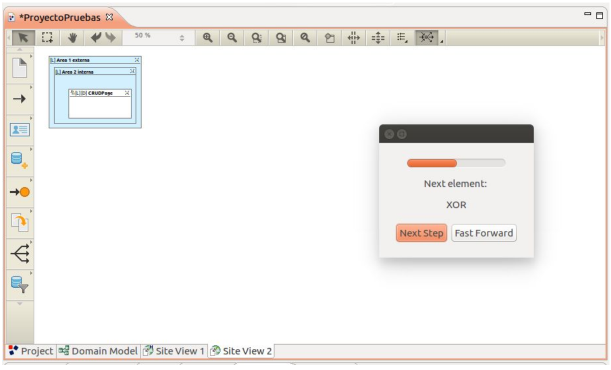


The following figures present different stages of the generation process for the selected pattern inside Site View 1.

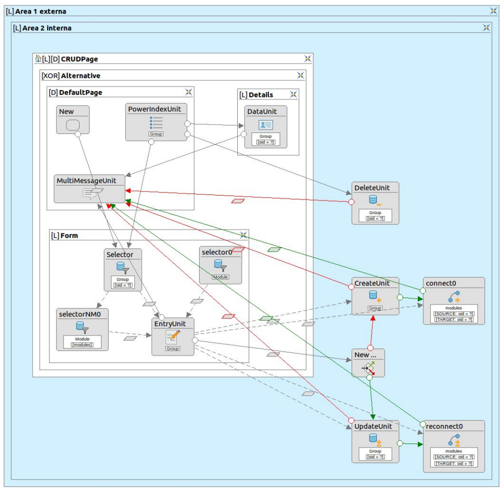




Once this first instance of the selected pattern is finished, the generation process starts again for the following container (Site View 2 > Area 2).

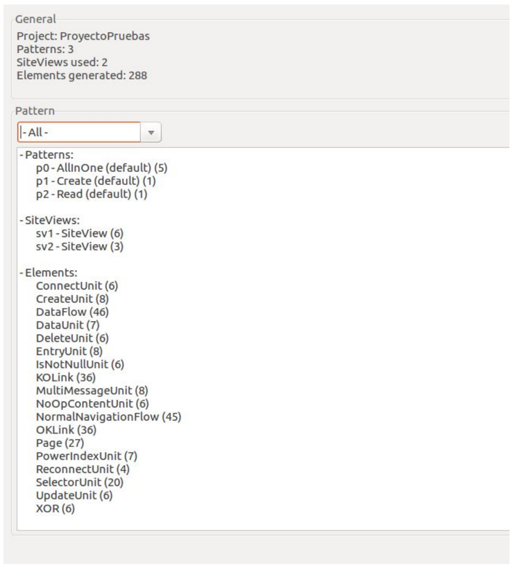


Next figure shows, once the generation is over and the dialog has been closed, the resulting IFML model for the instantiated pattern.



Example 2: Registry

AutoCRUD maintains a registry of all the IFML elements generated grouped by the pattern instantiation they belong. Just click Registry button to launch it. The registry dialog (shown in the next figure) presents, firstly, overall information about the project (name, number of patterns, siteviews involved, number of IFML elements generated) and, secondly, a listing with all the IFML elements (just their name) generated for a concrete pattern (or for all of them).


General
Project: ProyectoPruebas
Patterns: 3
SiteViews used: 2
Elements generated: 288

Pattern
|- All -

```
- Patterns:
    p0 - AllInOne (default) (5)
    p1 - Create (default) (1)
    p2 - Read (default) (1)

- SiteViews:
    sv1 - SiteView (6)
    sv2 - SiteView (3)

- Elements:
    ConnectUnit (6)
    CreateUnit (8)
    DataFlow (46)
    DataUnit (7)
    DeleteUnit (6)
    EntryUnit (8)
    IsNotNullUnit (6)
    KOLink (36)
    MultiMessageUnit (8)
    NoOpContentUnit (6)
    NormalNavigationFlow (45)
    OKLink (36)
    Page (27)
    PowerIndexUnit (7)
    ReconnectUnit (4)
    SelectorUnit (20)
    UpdateUnit (6)
    XOR (6)
```

CRUD patterns

Patterns are stored as XML files because this is the technology used by WebRatio. Therefore, no additional technology is necessary for pattern definition and usage. They include IFML elements fully or partially specified. AutoCRUD uses its own XML Schema to specify patterns. Visual pattern specification is not yet provided by this tool and manual pattern specification is out of the scope of this manual.

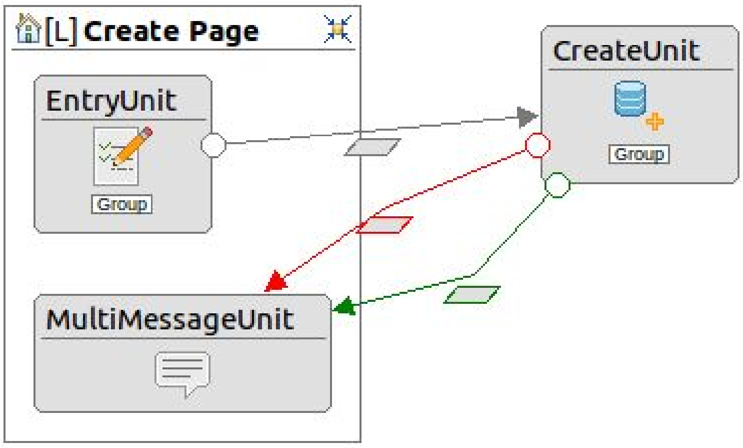
In this section, the provided XML specifications of CRUD patterns are explained, so they may serve as guides for users who want to create their own patterns.

Create pattern

It is structured as follows:

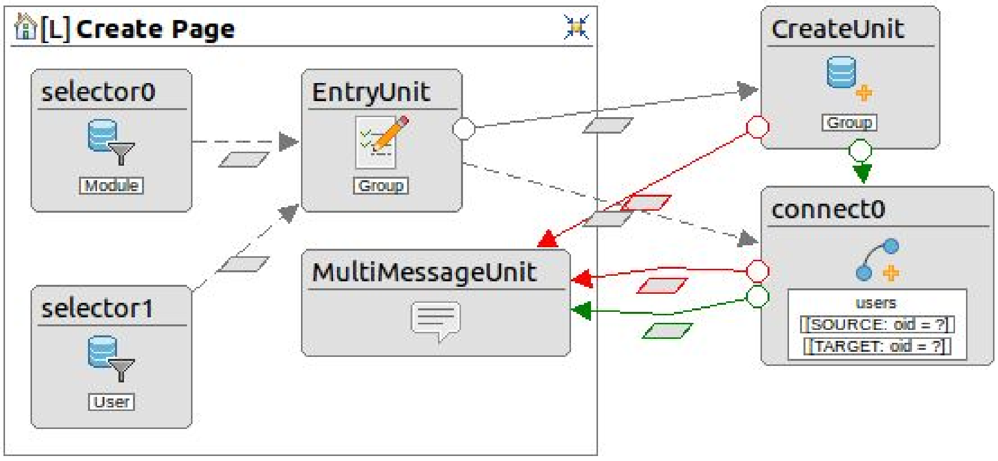
- Containers: A Page with an Entry Unit (EU) and a Multi Message Unit (MMU).
- Free units (not in a container): Create Unit (CU)
- Flows: (Page) - data flow - (CU), (CU) - OK - (MMU), (CU) - KO - (MMU)
- Relations:
    - All. A Selector Unit in the Page and a transport flow (SU) - (EU)
    - N:M. A Connector Unit (CU), a data flow (EU) - (CU) and a OK/KO flows (CU) - (MMU)

Next figure shows an example of instantiation without selecting relations:



Next figure shows an example of instantiation with two relations selected:

- A N:M relation linking User and Group
- A 1:N relation linking Module and Group

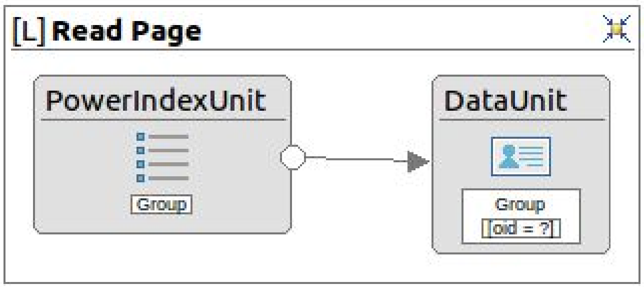


Read pattern

It is structured as follows:

- Containers: A Page with a Power Index Unit (PIU) and a Data Unit (DU).
- Free units (not in a container): none
- Flows: (PIU) - navigation flow - (DU)
- Relations: none

Next figure shows an example of instantiation:

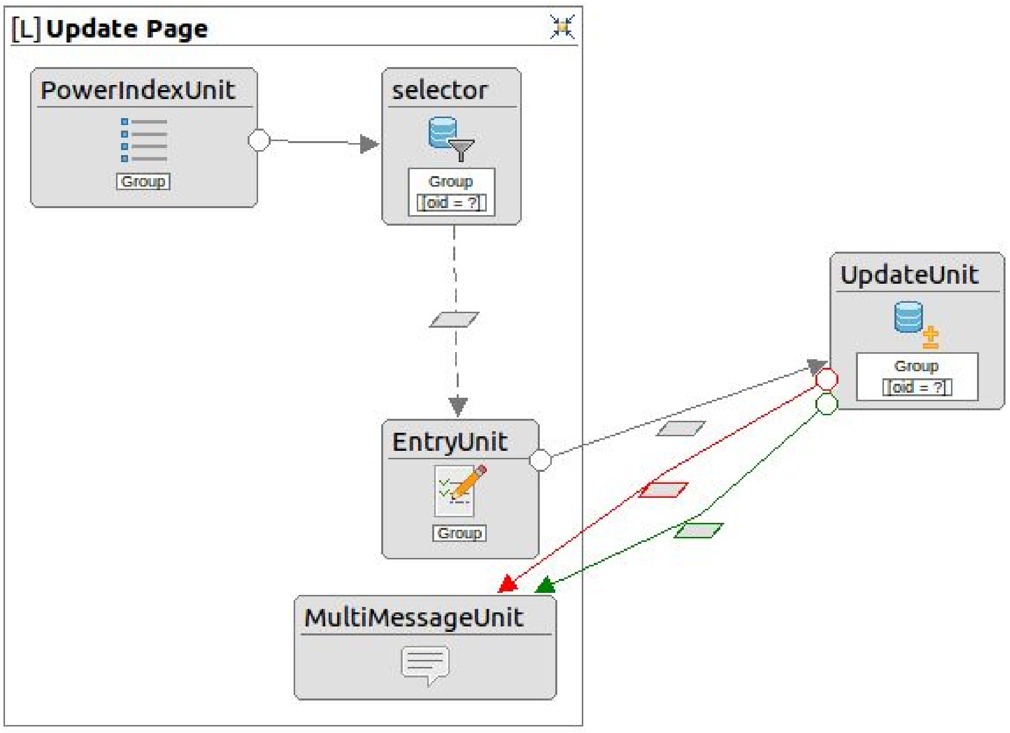


Update pattern

It is structured as follows:

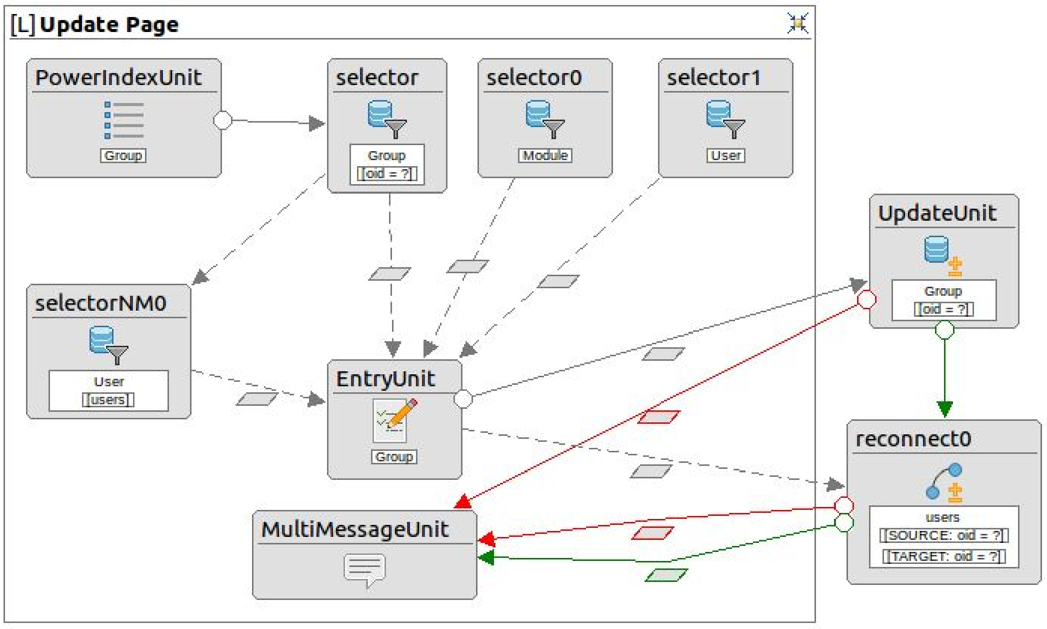
- Containers: A Page with an Entry Unit (EU), a Power Index Unit (PIU), a Multi Message Unit (MMU) and a Selector Unit (SU).
- Free units (not in a container): Update Unit (UU)
- Flows: (PIU) - navigation flow - (SU), (EU) - data flow - (UU), (UU) - OK/KO - (MMU), (SU) - transport flow - (EU)
- Relations:
  - All. A Selector Unit in the Page and a transport flow (SU) - (EU)
  - N:M. A Selector Unit (SUr), a data flow (SU) - (SUr), a data flow (SUr) - (EU), a Reconnect Unit (RU), a data flow (EU) - (RU), OK/KO flows (RU) - (MMU)

Next figure shows an example of instantiation without relation:



Next figure shows an example of instantiation with two relations selected:

- A N:M relation linking User and Group
- A 1:N relation linking Module and Group

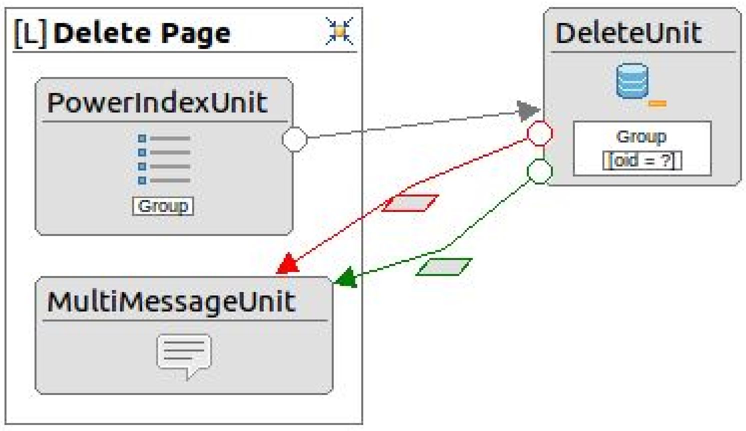


Delete pattern

It is structured as follows:

- Containers: A Page with a Power Index Unit (PIU) and a Multi Message Unit (MMU).
- Free units (not in a container): Delete Unit
- Flows: (PIU) - navigation flow - (DU), (DU) - OK/KO - (MMU)
- Relations: none

Next figure shows an example of instantiation:



Patterns XML Schema

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!-- Declaration -->

    <!-- Elements -->
    <xs:element name="page" type="pageType"/>

    <xs:element name="selectorUnit">
        <xs:complexType>
            <xs:attributeGroup ref="stdUnitAttributes"/>
            <xs:attribute name="type">
                <xs:simpleType>
                    <xs:restriction base="xs:token">
                        <xs:enumeration value="keyCondition"/>
                        <xs:enumeration value="roleCondition"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="parentId" type="xs:token"/>
        </xs:complexType>
    </xs:element>

    <xs:group name="contentUnits">
        <xs:choice>
            <xs:element name="powerIndexUnit"
type="stdUnitAttributesType"/>
            <xs:element name="dataUnit" type="stdUnitAttributesType"/>
            <xs:element name="entryUnit">
                <xs:complexType>
```

```
                    <xs:attributeGroup ref="stdUnitAttributes"/>
                    <xs:attribute name="type">
                        <xs:simpleType>
                            <xs:restriction base="xs:token">
                                <xs:enumeration value="preloaded"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:attribute>
                </xs:complexType>
            </xs:element>
            <xs:element name="multiMessageUnit"
type="stdUnitAttributesType"/>
            <xs:element name="noOpContentUnit"
type="stdUnitAttributesType"/>
            <xs:element ref="selectorUnit"/>
        </xs:choice>
    </xs:group>

    <xs:group name="links">
        <xs:choice>
            <xs:element name="okLink" type="okKoType"/>
            <xs:element name="koLink" type="okKoType"/>
            <xs:element name="normalNavigationFlow">
                <xs:complexType>
                    <xs:attributeGroup ref="stdLinkAttributes"/>
                    <xs:attribute name="validate" type="xs:boolean"/>
                    <xs:attribute name="type">
                        <xs:simpleType>
                            <xs:restriction base="xs:token">
                                <xs:enumeration value="isNotNull"/>
                                <xs:enumeration value="fixedValue"/>
                                <xs:enumeration value="entryToCreate"/>
                                <xs:enumeration value="entryToUpdate"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:attribute>
                </xs:complexType>
            </xs:element>
            <xs:element name="dataFlow">
                <xs:complexType>
                    <xs:attributeGroup ref="stdLinkAttributes"/>
                    <xs:attribute name="type">
                        <xs:simpleType>
                            <xs:restriction base="xs:token">
                                <xs:enumeration value="preload"/>
                                <xs:enumeration value="entryToConnect"/>
                                <xs:enumeration value="entryToReconnect"/>
                                <xs:enumeration value="unitToEntry"/>
                                <xs:enumeration value="unitToEntryRole"/>
                                <xs:enumeration value="entryToCreate"/>
                                <xs:enumeration value="entryToUpdate"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:attribute>
                </xs:complexType>
            </xs:element>
        </xs:choice>
    </xs:group>
```

```xml
<!-- Attributes -->

<xs:attribute name="id">
    <xs:simpleType>
        <xs:restriction base="xs:token">
            <xs:minLength value="1"/>
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
<xs:attribute name="name" type="xs:token"/>

<!-- AttributeGroups -->
<xs:attributeGroup name="stdLinkAttributes">
    <xs:attribute ref="id" use="required"/>
    <xs:attribute ref="name"/>
    <xs:attribute name="sourceId" type="xs:token" use="required"/>
    <xs:attribute name="targetId" type="xs:token" use="required"/>
</xs:attributeGroup>

<xs:attributeGroup name="stdUnitAttributes">
    <xs:attribute ref="id" use="required"/>
    <xs:attribute ref="name"/>
    <xs:attributeGroup ref="coordinates"/>
</xs:attributeGroup>

<xs:attributeGroup name="coordinates">
    <xs:attribute name="x" type="xs:token" use="required"/>
    <xs:attribute name="y" type="xs:token" use="required"/>
</xs:attributeGroup>

<!-- Complex types -->
<xs:complexType name="stdUnitAttributesType">
    <xs:attributeGroup ref="stdUnitAttributes"/>
</xs:complexType>

<xs:complexType name="stdLinkAttributesType">
    <xs:attributeGroup ref="stdLinkAttributes"/>
</xs:complexType>

<xs:complexType name="pagesSectionType">
    <xs:sequence>
        <xs:element ref="page" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="pageType">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="page"/>
        <xs:element name="xor" type="xorType"/>
        <xs:group ref="contentUnits"/>
    </xs:choice>
    <xs:attributeGroup ref="stdUnitAttributes"/>
    <xs:attribute name="landmark" type="xs:boolean"/>
    <xs:attribute name="default" type="xs:boolean"/>
</xs:complexType>

<xs:complexType name="xorType">
```

```
        <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element ref="page"/>
            <xs:group ref="contentUnits"/>
        </xs:choice>
        <xs:attributeGroup ref="stdUnitAttributes"/>
    </xs:complexType>

    <xs:complexType name="outsideunitsType">
        <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element name="createUnit" type="stdUnitAttributesType"/>
            <xs:element name="deleteUnit" type="stdUnitAttributesType"/>
            <xs:element name="updateUnit" type="stdUnitAttributesType"/>
            <xs:element name="isNotNullUnit" type="stdUnitAttributesType"/>
            <xs:element ref="selectorUnit"/>
        </xs:choice>
    </xs:complexType>

    <xs:complexType name="okKoType">
        <xs:attributeGroup ref="stdLinkAttributes"/>
        <xs:attribute name="message" type="xs:string"/>
        <xs:attribute name="type">
            <xs:simpleType>
                <xs:restriction base="xs:token">
                    <xs:enumeration value="noCoupling"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>

    <xs:complexType name="linksType">
        <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:group ref="links"/>
        </xs:choice>
    </xs:complexType>

    <xs:complexType name="relSubsectionType">
        <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:group ref="links"/>
            <xs:element ref="selectorUnit"/>
            <xs:element name="connectUnit" type="stdUnitAttributesType"/>
            <xs:element name="disconnectUnit"
type="stdUnitAttributesType"/>
            <xs:element name="reconnectUnit" type="stdUnitAttributesType"/>
        </xs:choice>
    </xs:complexType>

    <xs:complexType name="relationsType">
        <xs:sequence>
            <xs:element name="ALL" type="relSubsectionType" minOccurs="0"/>
            <xs:element name="LAST" type="linksType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="nmrelationsType">
        <xs:sequence>
            <xs:element name="ALL" type="relSubsectionType" minOccurs="0"/>
            <xs:element name="FIRST" type="linksType" minOccurs="0"/>
            <xs:element name="REMAINING" type="linksType" minOccurs="0"/>
            <xs:element name="LAST" type="linksType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

<!-- Root definition -->

    <xs:element name="PATTERN">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="PAGES" type="pagesSectionType"
minOccurs="0"/>
                <xs:element name="OUTSIDEUNITS" type="outsideunitsType"
minOccurs="0"/>
                <xs:element name="LINKS" type="linksType" minOccurs="0"/>
                <xs:element name="RELATIONS" type="relationsType"
minOccurs="0"/>
                <xs:element name="NMRELATIONS" type="nmrelationsType"
minOccurs="0"/>
            </xs:sequence>
            <xs:attribute ref="id" use="required"/>
            <xs:attribute ref="name" use="required"/>
        </xs:complexType>

        <xs:unique name="uniqueID">
            <xs:selector xpath=".//*" />
            <xs:field xpath="@id" />
        </xs:unique>
    </xs:element>

</xs:schema>
```