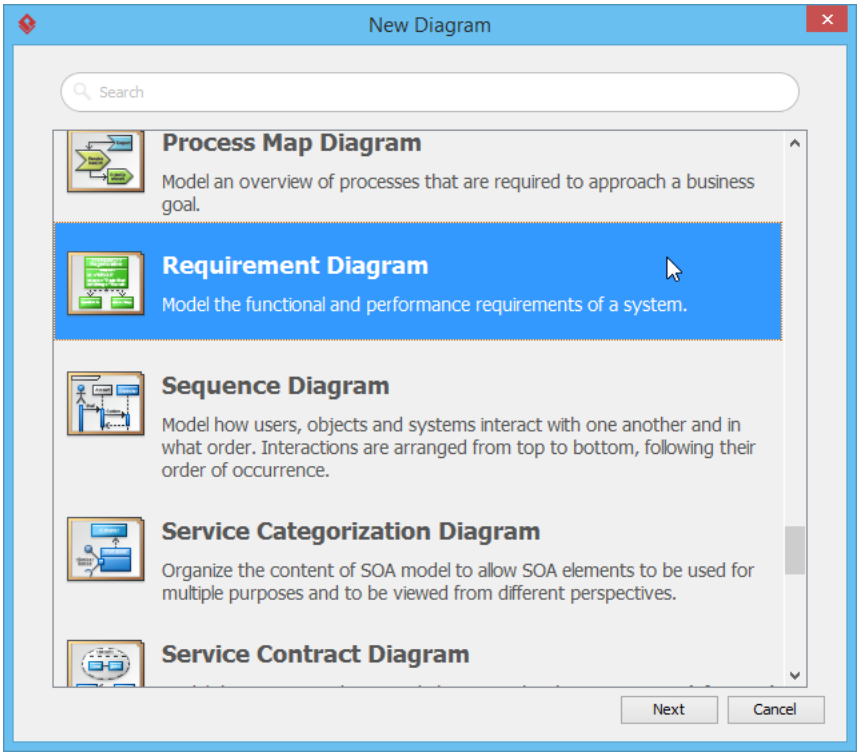
How to Customize Requirement Types?

Written Date : February 4, 2016

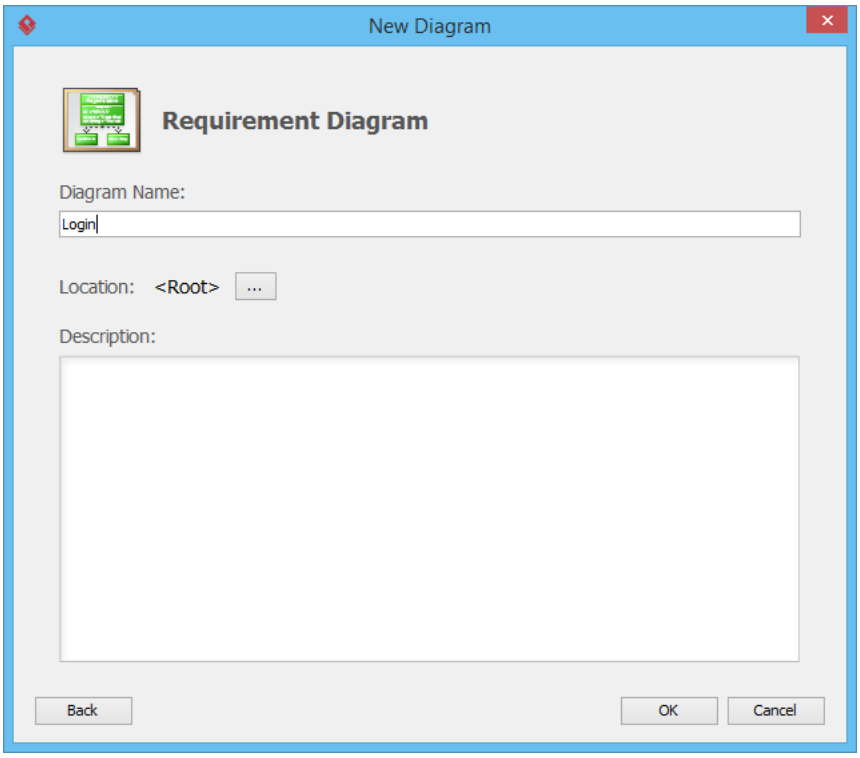
When you need to record system requirements, both functional and non-functional, requirement modeling will be helpful. Through requirement modeling, requirements are recorded and presented visually as boxes, with a name that summarize the requirement and a set of attributes that define the requirement. The default requirement 'box' allows specifying general attributes like ID, source, kind, verify method, risk and status. You may, however, define your own requirement types that contain attributes related to your domain.

---

1. Create a new project by selecting **Project > New** from the application toolbar.
2. In the **New Project** window, enter *Tutorial* as project name and click **Create Blank Project**.
3. We are going to create several requirements in this tutorial. Let's create a requirement diagram first. To create a requirement diagram, select **Diagram > New** from the application toolbar.
4. In the **New Diagram** window, select **Requirement Diagram** and click **Next**.



5. Enter *Login* as diagram name and click **OK**.

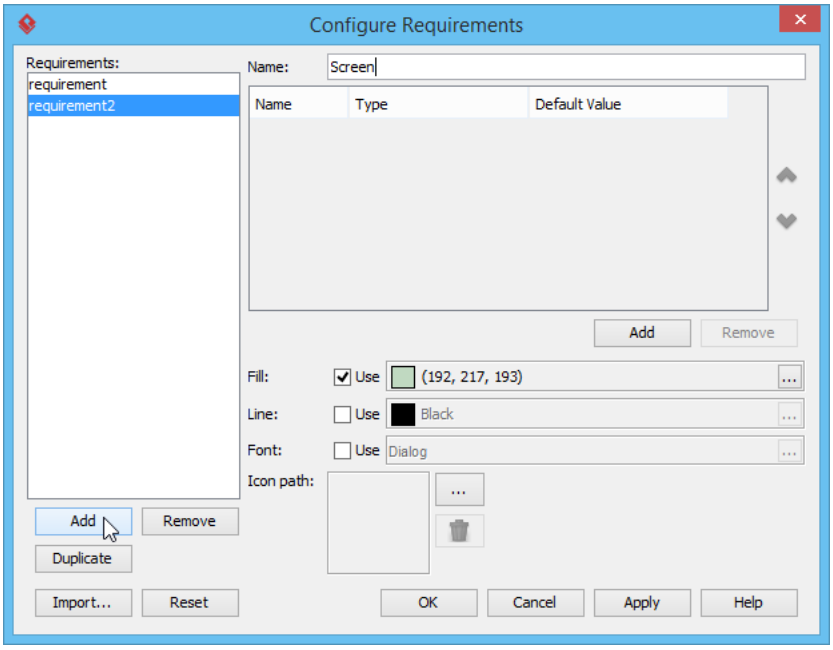


6. On the left hand side of the diagram you can see the diagram toolbar, where you can pickup a tool like a **Requirement**, a **Model** or a **Test Case** and click on the diagram to create a shape. Normally you will make use of the **Requirement** tool to create requirements. But in this tutorial, we are going to define our own types first. Select **Window > Configuration > Configure Requirements...** from the application toolbar.

7. We are going to create two requirement type:

Type	Description
Screen	A type of requirement which consists of properties about a screen design.
Checking	A type of requirement which consists of rules for validating input and the proper response to bad input.

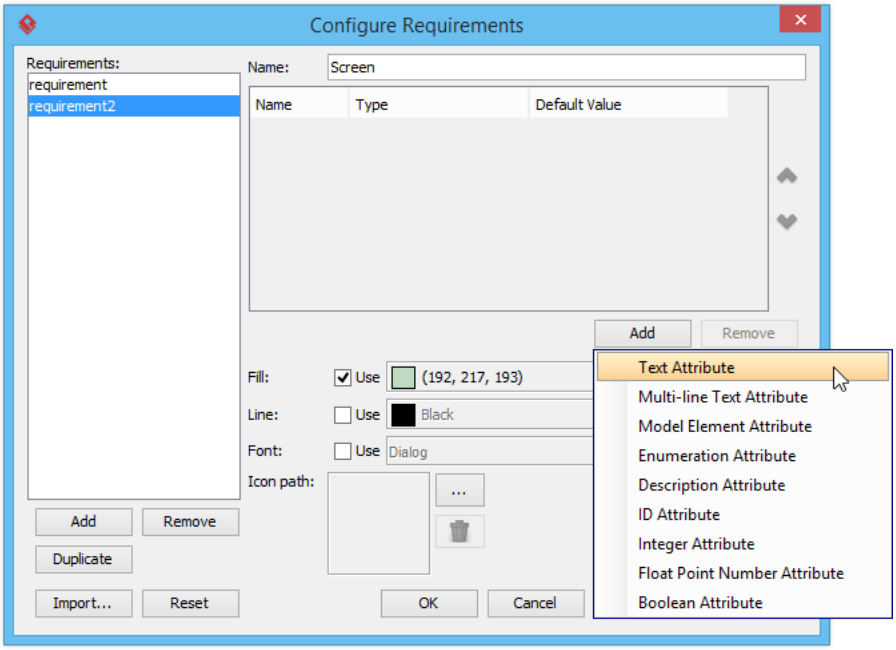
Create the *Screen* type first. Click **Add** at the bottom left of the **Configure Requirements**. Then, enter *Screen* as name.



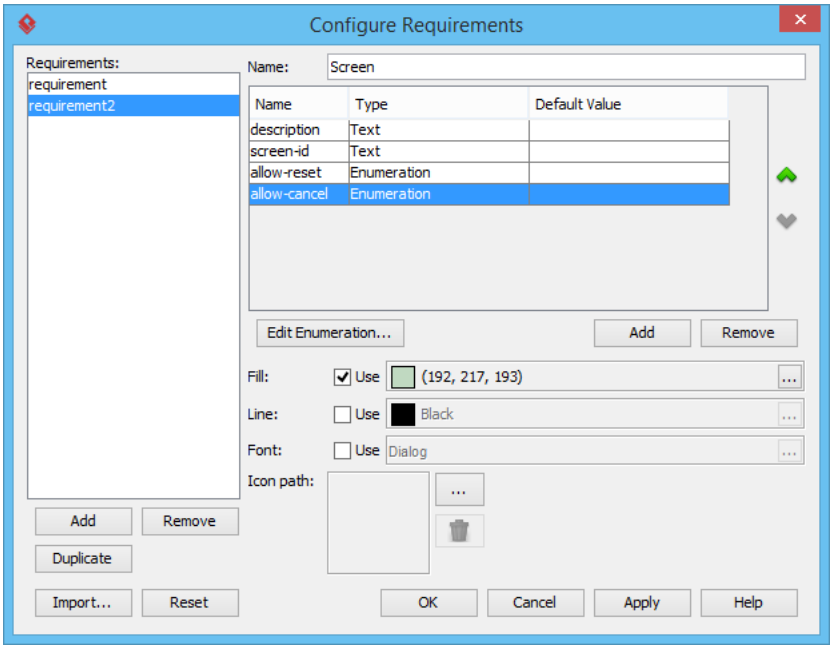
8. What makes the type *Screen* meaningful is its unique attributes (i.e. properties). For type *Screen*, the following attributes are needed:

Attribute	Type	Description
description	Text	A description of the screen being documented.
screen-id	Text	A unique and internal value for identifying the screen being documented.
allow-reset	Enumeration	Determine whether there is a Reset button for clearing fields on the screen.
allow-cancel	Enumeration	Determine whether there is a Cancel button to close the screen without proceeding.

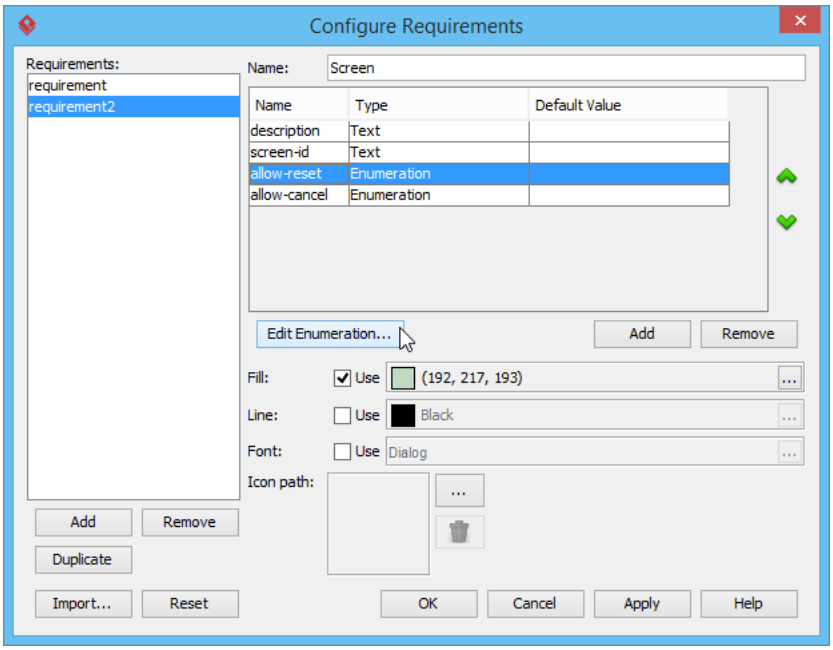
We need to define those properties above. Click **Add** at the right hand side of the dialog box and select **Text Attribute** from the popup menu.



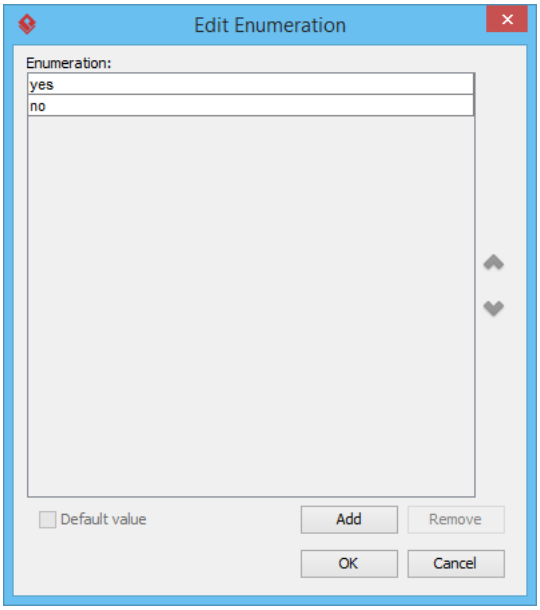
9. Enter *description* as name. Repeatedly create the remaining properties. Make sure you have chosen **Enumeration** as the type of *allow-reset* and *allow-cancel*.



10. The third attribute allow-reset is an enumeration attribute which enables the selection *yes* or *no*. Select the attribute and click **Edit Enumeration...**.

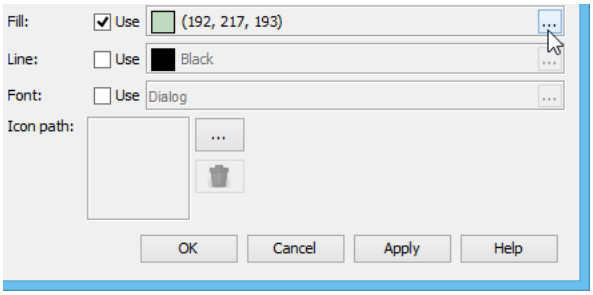


11. In the **Edit Enumeration** window, click **Add** and enter *yes*. Click **Add** again and enter *no*. This creates the two allowed values, *yes* or *no*, for this attribute. Click **OK** to go back to requirement configuration.

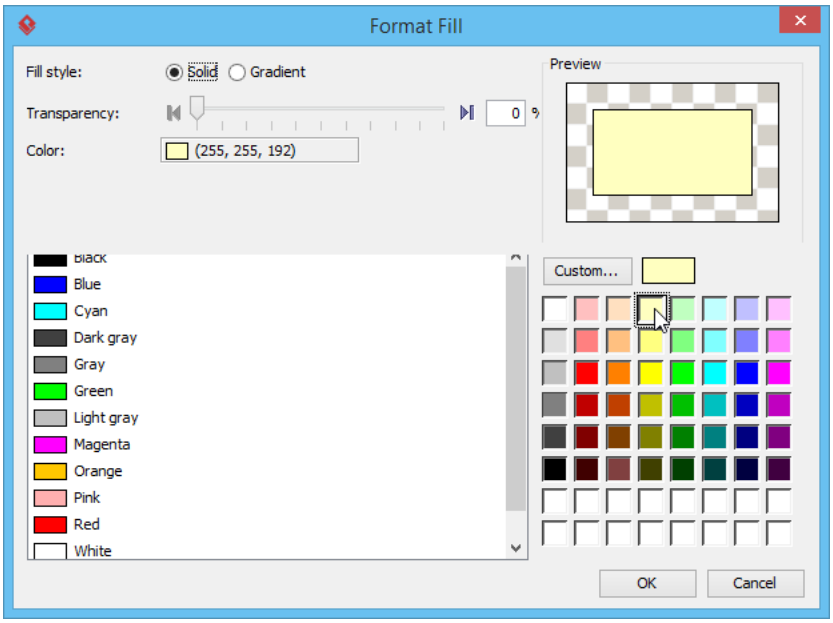


12. Similarly, add the enumeration values *yes* and *no* for the attribute *allow-cancel*.

13. The four attributes are all added. Besides defining requirement attributes, you can also set the formatting properties like fill, line and font styles. These setting will affect how this type of requirement will look like in diagram. Let's try setting a different fill color. Click the **...** button for **Fill**.



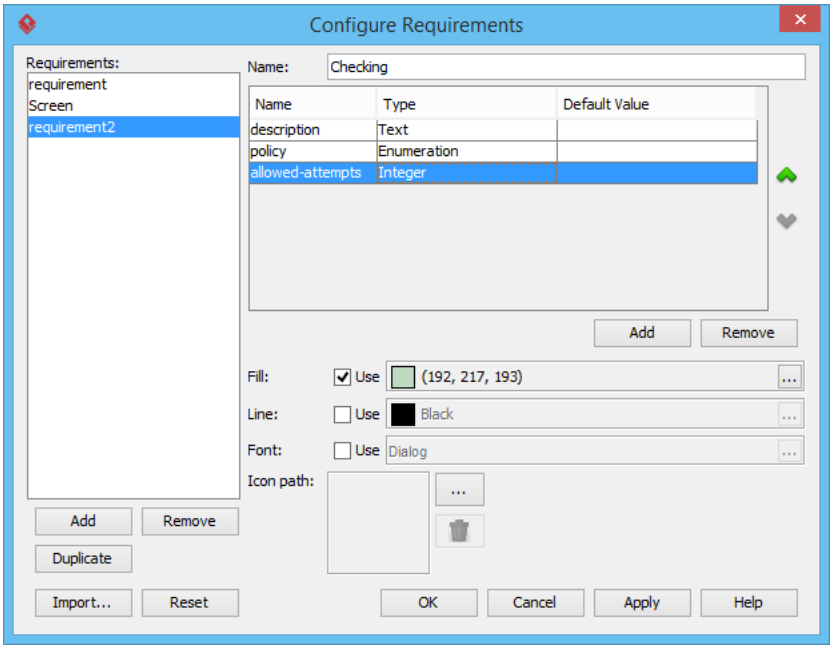
14. In the **Format Fill** window, select yellow and click **OK**.



15. This ends the configuration of type *Screen*. Apply the same technique to create another requirement type *Checking*. Here is a list of attributes:

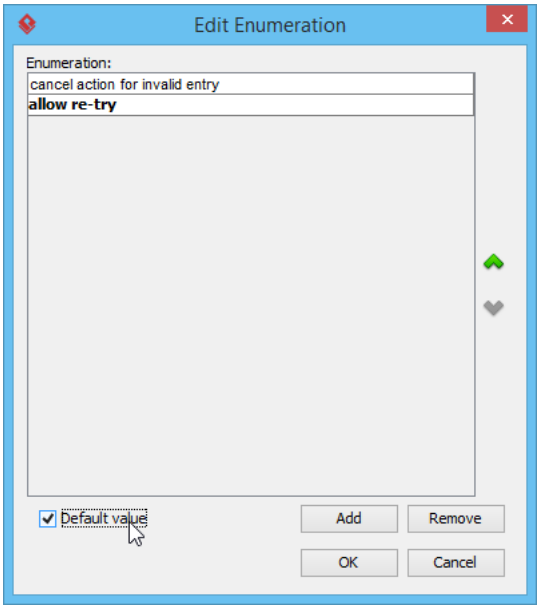
Attribute	Type	Description
description	Type	A description of checking needed to perform
policy	Enumeration	Determine the action to take when a bad input is detected upon checking. We may allow user to try again, or just cancel the action immediately.
allowed-attempts	Integer	The number of attempt user can make.

16. Add the enumeration values *cancel action for invalid entry* and *allow re-try* for the attribute *policy*. In most checking, we allow user to try again. Therefore, select allow re-try and check **Default value** at the bottom. Therefore, set *allow re-try* as the default value. This will cause *allow re-try* be selected when you create a <<Checking>> requirement.

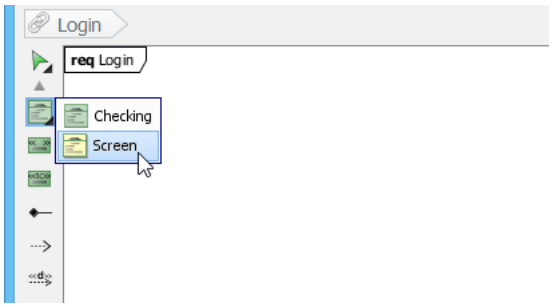


17. Requirement types are configured. Click **OK** to return to requirements configuration. Click **OK** again to confirm all the changes and return to the diagram.

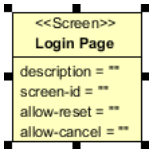
18. Now comes the final attribute - *allowed attempts*. It is an attribute that requires a numeric input. Therefore, click **Add** and select **Integer Attribute**. Enter *allowed attempts* as name.



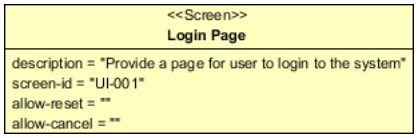
19. We can now create requirements with the new types. In the diagram toolbar, press on the tool **Requirement** and select *Screen*.



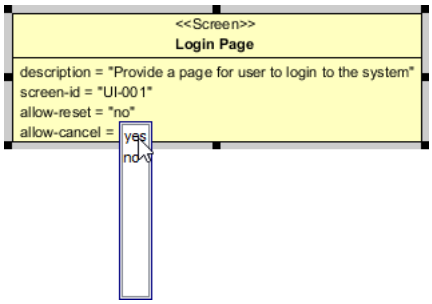
20. Click on diagram to create a requirement. Name it as *Login Page*. You got a requirement like this:



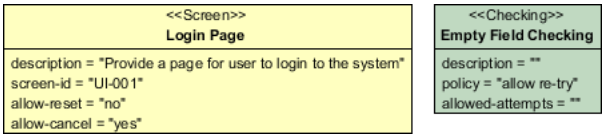
21. Double click on the attribute *description* and enter *Provide a page for user to login to the system* as description. Similarly, double click on *screen-id* and enter *UI-001*.



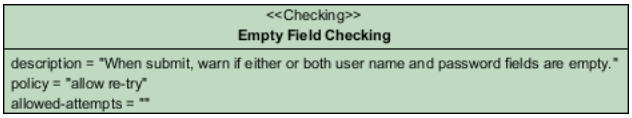
22. Remember *allow-reset* and *allow-cancel* are enumeration attribute? Double click on *allow-reset*, you can see that the value is restricted to either *yes* or *no*, as defined. Select *no* for *allow-reset* and *yes* for *allow-cancel*.



23. Now, make use of type *Checking* to specify a requirement for validating form input. Press on requirement type *Screen* in the diagram toolbar and select *Checking*. Click on diagram to create a requirement. Name it as *Form Field Checking*.

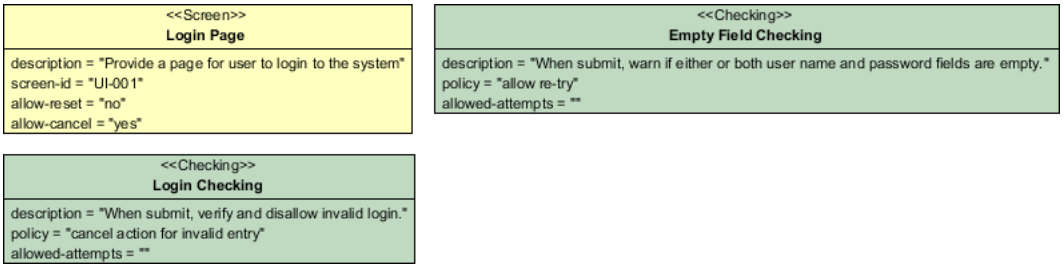


24. Double click on attribute *description* and enter *When submit, warn if either or both user name and password fields are empty*.



25. It is fine to let user re-enter the correct email address. Therefore, just keep the *policy* as *allow re-try*. Leave *allowed-attempts* empty to indicate that there is no restriction on the number of attempts.

26. Create another <<Checking>> requirement *Login Checking*. Set the description to *When submit, verify and disallow invalid login.*, policy to *cancel action for invalid entry*. Finally, you should have a diagram like this:



Visual Paradigm home page
(https://www.visual-paradigm.com/)

Visual Paradigm tutorials
(https://www.visual-paradigm.com/tutorials/)