

RACI in Rational Method Composer

Contents

| | |
|--|----|
| Contents..... | 1 |
| 1 Introduction..... | 1 |
| 2 Creating a RACI User-Defined Type in RMC..... | 1 |
| 2.1 Making Modifications..... | 2 |
| 2.2 Variations of RACI..... | 4 |
| 2.3 Qualifiers..... | 5 |
| 3 Extending and renaming existing relationships..... | 6 |
| 3.1 Benefits and drawbacks..... | 6 |
| 3.2 Modifying the UDT Definition..... | 6 |
| 3.3 Customizing the skin..... | 7 |
| 4 Publishing..... | 9 |
| References..... | 10 |

1 Introduction

This article will show how to incorporate RACI into Rational Method Composer (RMC).

A responsibility assignment matrix (RACI matrix) describes the participation by various roles in completing tasks for a project or business process.

RACI is an acronym from the four key responsibilities used: *Responsible*, *Accountable*, *Consulted*, and *Informed*.

- **Responsible:** The person who does the work to achieve the task. They have responsibility for getting the work done or decision made. As a rule this is one person; examples might be a business analyst, application developer or technical architect.
- **Accountable:** The person who is accountable for the correct and thorough completion of the task. This must be one person and is often the project executive or project sponsor. This is the role that responsible is accountable to and approves their work.
- **Consulted:** The people who provide information for the project and with whom there is two-way communication. This is usually several people, often subject matter experts.
- **Informed:** The people who are kept informed about progress and with whom there is one-way communication. These are people that are affected by the outcome of the tasks so need to be kept up-to-date.

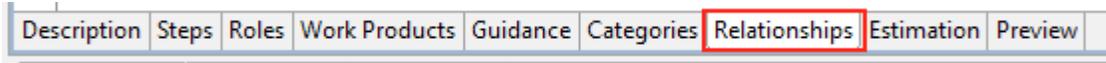
There are many variants of RACI. For example, RASCI is an acronym using an additional responsibility: *Supportive*. PACSI and CAIRO are other variants. Any of these variants can be supported in RMC.

2 Creating a RACI User-Defined Type in RMC

To begin, review the online help topic [Creating a user-defined type or modifying an existing type](#).

2.1 Making Modifications

We will be [modifying an existing user-defined types](#) to add user-defined relationships. We will be adding a new relationship for each of the R, A, C, I parts, which will add these relationships to the relationships tab of the task.



Modifying the user-defined type to add RACI relationships will also display the relationships on the task and role pages.

Task: Plan the project



Roles

| | |
|-------------|---|
| Responsible |  Project Manager |
| Accountable |  Business Analyst |
| Consulted |  Technical Architect |
| Informed |  Application Developer |

The relationships are displayed for the task Plan the project.

Role: Technical Architect



Relationships



| | |
|--------------|--|
| Consulted by |  Plan the project |
|--------------|--|

The role Technical Architect consults on the task Plan the project. Note that the Role description displays the opposite relationship. On the role page, the relationship is “consulted by”, while on the task page, the relationship is “consulted”. The text is customizable - you choose what to call the relationship and it’s opposite.

A sample library containing the user-defined type definitions is included for download. If you do not use the sample library, you can modify the UDT definition in RMC.

To [modify the existing user-defined types](#) to include RACI, go to File > Create UDT Definition... and add the following XML within the body of the text (Note the following code supports the RACI Variation):

```
<modifiedType id="org.eclipse.epf.uma.Task">
  <section id="com.abc.Task.addRoles" type="reference">
```

```

        <name>Roles</name>
        <reference name="Responsible" id="com.abc.Task.responsible"
contributeTo="roles" publish="true">
            <valueType>org.eclipse.epf.uma.Role</valueType>
            <oppositeReference publish="true">
                <name>Responsible for</name>
            </oppositeReference>
        </reference>
        <reference name="Accountable" id="com.abc.Task.accountable"
contributeTo="roles" publish="true">
            <valueType>org.eclipse.epf.uma.Role</valueType>
            <oppositeReference publish="true">
                <name>Accounted by</name>
            </oppositeReference>
            <reference_qualifiers>
                <qualifier
id="com.abc.Task.accountable.reviews">Reviews</qualifier>
                <qualifier
id="com.abc.Task.accountable.signs">Signs</qualifier>
            </reference_qualifiers>
        </reference>
        <reference name="Consulted" id="com.abc.Task.consulted" publish="true"
contributeTo="roles">
            <valueType>org.eclipse.epf.uma.Role</valueType>
            <oppositeReference publish="true">
                <name>Consulted by</name>
            </oppositeReference>
        </reference>
        <reference name="Informed" id="com.abc.Task.informed" publish="true"
contributeTo="roles">
            <valueType>org.eclipse.epf.uma.Role</valueType>
            <oppositeReference publish="true">
                <name>Informed by</name>
            </oppositeReference>
        </reference>
    </section>
</modifiedType>
<modifiedType id="org.eclipse.epf.uma.TaskDescriptor">
    <!-- This adds custom relationships and attributes from a task to a task
descriptor -->
</modifiedType>

```

Now select File > Update UDT Definition.

The result is new relationships in the task editor as follows:

2.2 Variations of RACI

You can customize the user-defined type definitions to support other RACI variants.

For example, to use the RASCI variant, add the Supportive responsibility by adding the following lines:

```
<reference name="Supportive" id="com.abc.Task.supportive" publish="true"
contributeTo="roles">
    <valueType>org.eclipse.epf.uma.Role</valueType>
    <oppositeReference publish="true">
        <name>Supported by</name>
    </oppositeReference>
</reference>
```

Explanation:

- Reference name: This is the name of the relationship that appears on the task page
- Id: Can be any text string, although convention is to use a string that identifies the company doing the modification, followed by the id of the type being modified, and then a name for the modification
- Publish: Whether to display the relationship on the published page
- contributeTo: Set to “roles”: this means the reference is a role relationship. When this task has been added to a process work breakdown structure, the role relationship name will appear in the the model info column, along with the standard role relationships (“performing” and “additional”).
- valueType: What types of elements can be assigned to the relationship (in this case, only Roles

can be assigned to the relationship

- oppositeReference: Refers the relationship that is to be published on the Role page – its name and whether or not to publish it.

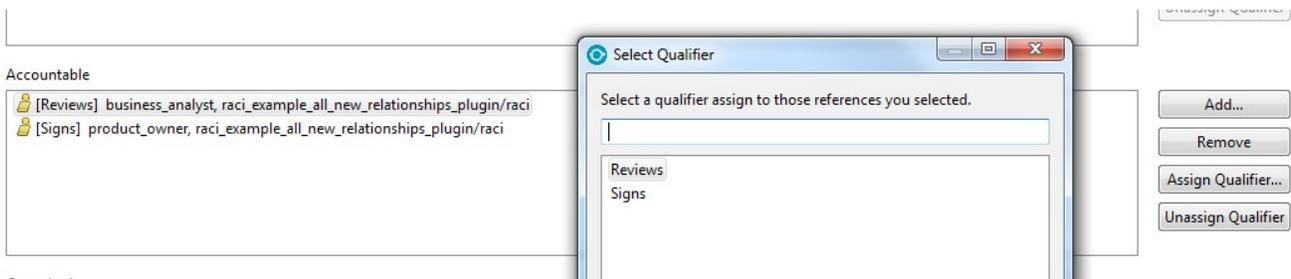
2.3 Qualifiers

We can also add “qualifiers” to a relationship. In the example, two qualifiers were added to the “Accountable” role to distinguish between “reviewers” and “signers”.

```
<reference_qualifiers>  
  <qualifier id="com.abc.Task.accountable.reviews">Reviews</qualifier>  
  <qualifier id="com.abc.Task.accountable.signs">Signs</qualifier>  
</reference_qualifiers>
```

- Qualifier Id: Can be any text string. Convention used here is to use the Id of the reference (relationship) being qualified, followed by a name for the qualifier.

The qualifier is selected as shown:



The task page changes based on the qualifier selected.

Task: Plan the project

➔

| Roles | |
|-------------|-----------------------------|
| Responsible | Project Manager |
| Accountable | Reviews Business Analyst |
| | Signs Product Owner |
| Consulted | Technical Architect |
| Informed | Application Developer |

At this time, however, the task descriptor page does not include qualifiers. An RFE has been logged to fix this.

Task: Plan the project



Relationships

| | | | |
|-------|---------------------------------|-----------------------------------|---|
| Roles | Responsible: Project Manager | Consulted: Technical Architect | : |
|-------|---------------------------------|-----------------------------------|---|

Additional roles

| | |
|-------------|-----------------------------------|
| Accountable | Business Analyst Product Owner |
| Informed | Application Developer |

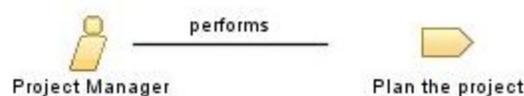
3 Extending and renaming existing relationships

Instead of defining a whole new set of relationships, it may be preferable to just extend the existing role relationships, and renaming the existing relationships.

For example, the “Primary performer” relationship can be renamed as “Responsible” and “Additional performers” can be renamed to “Consulted”. Then instead of adding 4 new relationships, you only need to add two, “Accountable” and “Informed”.

3.1 Benefits and drawbacks

The main benefit of extending the existing relationships is so that RMC will continue to generate diagrams on role pages like this one:



The drawbacks are:

- the text “performs” that appears on the diagram cannot be customized
- you can only customize the names of relationships as they appear in the published site – the authors that work in RMC will still see “Primary performer” and “Additional performers”.

These drawbacks are minor, so extending may be the best option.

3.2 Modifying the UDT Definition

Similar to the previous example, you need to modify the user-defined types. However, you only have to create 2 new relationships instead of 4. To [modify the existing user-defined types](#), go to File > Create UDT Definition... and add the following XML within the body of the text:

```
<modifiedType id="org.eclipse.epf.uma.Task">
  <section id="com.abc.Task.addRoles" type="reference">
    <name>Additional roles</name>
    <reference name="Accountable" id="com.abc.Task.accountable"
publish="true" contributeTo="roles">
      <valueType>org.eclipse.epf.uma.Role</valueType>
      <oppositeReference publish="true">
        <name>Accountable for</name>
      </oppositeReference>
      <reference_qualifiers>
        <qualifier
id="com.abc.Task.accountable.reviews">Reviews</qualifier>
        <qualifier
id="com.abc.Task.accountable.signs">Signs</qualifier>
      </reference_qualifiers>
    </reference>
    <reference name="Informed" id="com.abc.Task.informed" publish="true"
contributeTo="roles">
      <valueType>org.eclipse.epf.uma.Role</valueType>
      <oppositeReference publish="true">
        <name>Informed by</name>
      </oppositeReference>
    </reference>
  </section>
</modifiedType>
</types>
```

Now select File > Update UDT Definition.

The result is two new relationships in the task editor.

3.3 Customizing the skin

The relationships “Primary performer” and “Additional roles” can be renamed in the published sites by using a custom skin. You can create your own RACI skin by copying a Rational Method Composer default skin and then making changes to resource files (XSL).

The default skin location is C:\<user_home>\RMC\Skins.

Default locations for the *user_home* folder are as follows:

| Operating system | Default location of the <i>user_home</i> folder |
|---------------------------------|---|
| Microsoft Windows Vista, 7 or 8 | C:\Users\ <i>user_name</i> |
| Microsoft Windows XP | C:\Documents and Settings\ <i>user_name</i> |
| Linux | /home/ <i>user_name</i> |

Note that the *user_home*\RMC\Skins folder is created the first time that you publish or browse a configuration by using Rational Method Composer. The folder is initially populated with the skin that

is used to publish or browse. Adding a new skin folder to this location makes the new skin available in the publishing options. See [publishing configurations as websites](#).

Copy one of the default folders that appear here, such as “RMC”, and give it a new name, such as “RMC_RACI”.

To change the naming of the existing relationships:

1. Open **<skin folder>content_xsl/resources.properties**

2. Change the text that appears on the task page:

primaryPerformerText=Primary Performer

to

primaryPerformerText=Responsible

and

additionalText=Additional

to

additionalText=Consulted

3. Change the text that appears on the task descriptor page:

mainText=Primary

to

mainText=Responsible

and

additionalText=Additional

to

additionalText=Consulted

4. Change the text that appears on the Role page.

primaryPerformsText=Primary Performs

to

primaryPerformsText=Responsible for

and

additionallyPerformsText=Additionally Performs

to

additionallyPerformsText=Consulted by

5. Change the text that appears on the Role descriptor page.

performsText=Performs

to

performsText=Responsible for

6. Change the text that appears on the Team Allocation tab of the Delivery Process.

modelInfo_primaryTasksText=Performs as Owner

to

modelInfo_primaryTasksText=Performs as Responsible

and

modelInfo_additionalTasksText=Performs as Additional

to

modelInfo_additionalTasksText=Performs as Consulted

7. Remove the text for the task-descriptor “assisting” relationship.

assistingText=Assisting

to

assistingText=

When you publish, the pages will appear with the new text, as in the screenshots below.

This is an example of the Task page.

Task: Plan the project



Relationships

| | | |
|-------|---------------------------------|-----------------------------------|
| Roles | Responsible: Project Manager | Consulted: Technical Architect |
|-------|---------------------------------|-----------------------------------|

Roles

| | |
|-------------|---|
| Accountable | Reviews Business Analyst Signs Product Owner |
| Informed | Application Developer |

This is an example of the Task Descriptor page.

Task: Plan the project



Relationships

| | | |
|-------|---------------------------------|-----------------------------------|
| Roles | Responsible: Project Manager | Consulted: Technical Architect |
|-------|---------------------------------|-----------------------------------|

Additional roles

| | |
|-------------|-----------------------------------|
| Accountable | Business Analyst Product Owner |
| Informed | Application Developer |

Properties

| | |
|----------------------|--|
| Multiple Occurrences | |
| Event Driven | |
| Ongoing | |
| Optional | |
| Planned | |
| Repeatable | |

Note that in the default skins provided with RMC, the role descriptor pages will not display RACI

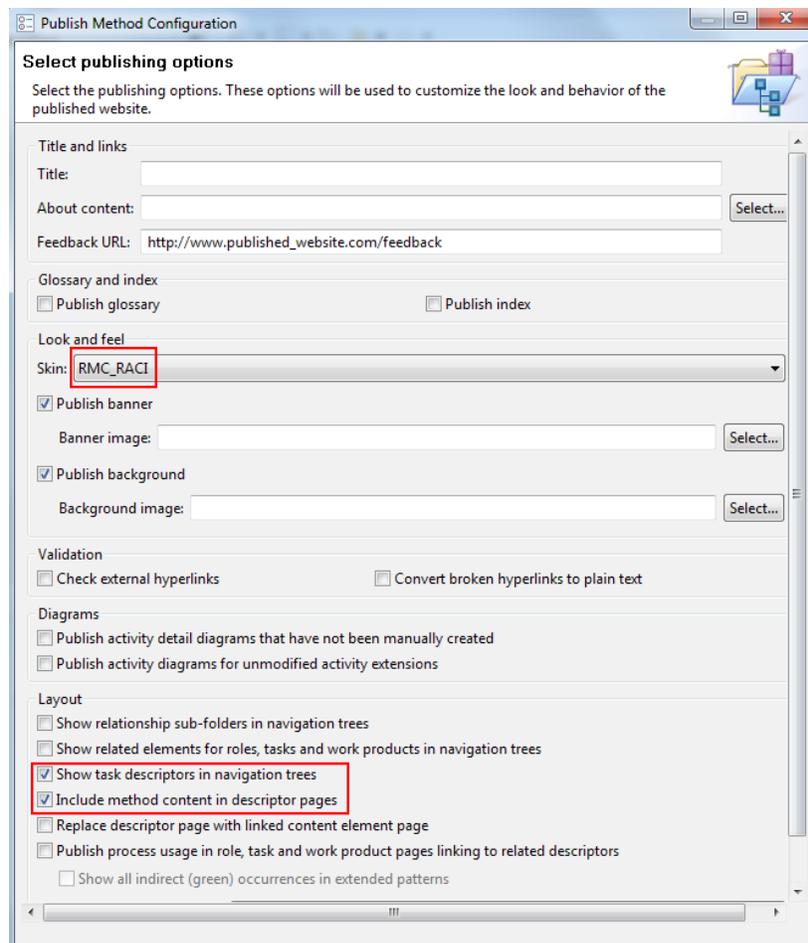
relationships. This can be fixed by replacing the file *role_descriptor.xsl* in your skins with the one provided with this article. We have also provided an updated *resources.properties* file that includes the name changes described above.

For more information see [Customizing Skins in Rational Method Composer](#).

4 Publishing

Go to Configuration > Publish.

1. Select HTML. Click Next.
2. Select the RACI configuration. Click Next.
3. Select Publish the entire configuration. Click Next.
4. Select the RMC_RACI skin. Select Show task descriptors in navigation trees and select Include method content in descriptor pages. Click Next.



5. Select Default. Click Next.
6. Select Static Web site. Click Finish.

References

- [1] http://en.wikipedia.org/wiki/Responsibility_assignment_matrix

[2] <http://www.projectsmart.co.uk/raci-matrix.php>