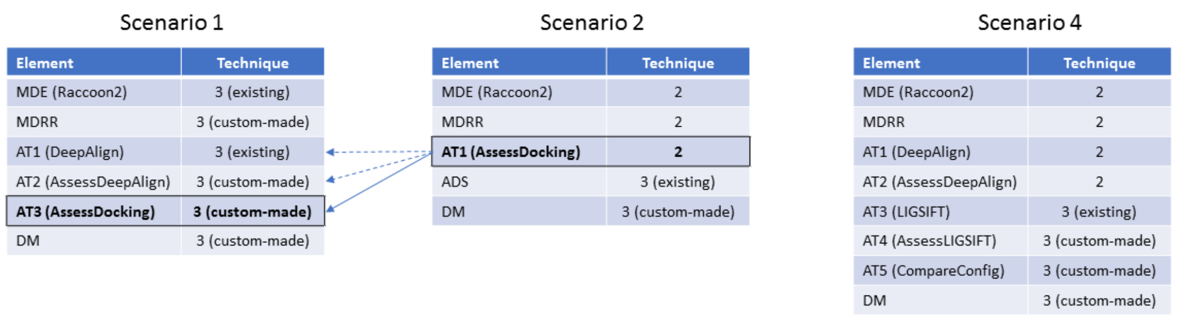

Scenario 1

Element	Technique
MDE (Raccoon2)	3 (existing)
MDRR	3 (custom-made)
AT1 (DeepAlign)	3 (existing)
AT2 (AssessDeepAlign)	3 (custom-made)
AT3 (AssessDocking)	**3 (custom-made)**
DM	3 (custom-made)

Scenario 2

Element	Technique
MDE (Raccoon2)	2
MDRR	2
AT1 (AssessDocking)	**2**
ADS	3 (existing)
DM	3 (custom-made)

Scenario 4

Element	Technique
MDE (Raccoon2)	2
MDRR	2
AT1 (DeepAlign)	2
AT2 (AssessDeepAlign)	2
AT3 (LIGSIFT)	3 (existing)
AT4 (AssessLIGSIFT)	3 (custom-made)
AT5 (CompareConfig)	3 (custom-made)
DM	3 (custom-made)

Figure 8.1: Overview of techniques used in the three selected scenarios.

the already implemented MDE and MDRR. The most interesting use of the framework can be observed when implementing the AT for assessing docking results which is required in Scenario 2. At this point, there are three already implemented ATs that are candidates for reuse. The AT: AssessDocking, which is AT3 in Scenario 1, can be reused since it has the same core computation and the same types of interfaces as the required AT in Scenario 2 (the dashed arrows in Figure 8.1 show the candidates which were considered but not chosen, and the full arrow shows the chosen AT). An analogous method can be used when implementing Scenario 4. The details of these procedures will be described in the remainder of this chapter.

8.2 Implementing Scenario 1

The title of Scenario 1 is: Suggest a ligand-protein pair that should be used in the next molecular docking, based on protein similarity and previous results. Scenario 1 starts with the user running a docking or VS simulation. Previous docking results should then be analysed to find receptors that are similar to the currently used receptor. Once a similar receptor has been found, the previous docking results of that receptor should be analysed to filter out ligands which have been successfully docked to it. This ligand (or multiple ligands) can be suggested as a candidate for the next docking with the currently used receptor. A basic diagram of Scenario 1 was shown in Figure 4.3. It proposes the use of six elements of these four element types:

- MDE: The extension of Raccoon2 presented in this thesis.
- MDRR: A custom-made MongoDB-based repository.
- 3 $\times$ AT: the structural alignment tool DeepAlign (AT1), a custom-made assessor of DeepAlign (AT2), and a custom-made assessor of docking results (AT3).
- DM: a custom-made DM.

This diagram was derived from the basic diagram of the framework, thus it is reasonable to assume that Scenario 1 fits the framework. However, to provide a more precise analysis, an attempt to derive a detailed diagram for each element and its interfaces can be made. Prior to starting the coding step, a textual and formal description of each element and its interfaces can be used to confirm that the proposed elements can be used.

8.2.1 Abstract descriptions of Scenario 1

MDE: The extended version of Raccoon2 Any existing docking tool can be used as an MDE, as long as it fits the description of the element type MDE of the framework. The extended version of Raccoon2, as described in Chapter 3, is one option. In order to to determine whether Raccoon2 can be the MDE, a diagrammatic, textual, and formal description of the interfaces of Raccoon2 as an MDE can be created.

If a diagram of Raccoon2 can be derived from the detailed diagram of the element type MDE (Figure 6.2), if the list of interfaces of Raccoon2 can be derived from the list of interfaces of an MDE, and if the formal description of Raccoon2 and its interfaces can be derived from the formal description of an MDE and its interfaces (Appendix B), then one can conclude that Raccoon2 can be used as an MDE.

Figure 8.2 shows the detailed diagram of the extended version of Raccoon2 with gUSE. This diagram shows that replacing the generic labels in Figure 6.2 with Raccoon2-specific ones is possible. The required user input for docking consists of one or more ligands, receptors and appropriate configuration files. The result of the docking can be provided to the user, and forwarded to an element representing an MDRR. For Scenario 1, two additional user input values are required: AutoDock Vina affinity threshold and DeepScore threshold. These can be entered into Raccoon2 and forwarded to the MDRR. The suggested candidate ligand for next docking, provided as a result by the MDRR, can be presented to the user. A list of interfaces, derived from the list of MDE interfaces, can also be created.

Raccoon2 interfaces

1a-c. Raccoon2 provides a user interface to obtain ligand, receptor, and config files.

1d-e. Raccoon2 should provide a user interface to obtain the AutoDock Vina and DeepScore thresholds.

2. Raccoon2 provides a user interface to view docking results which should be extended to include the suggested ligand for next docking.

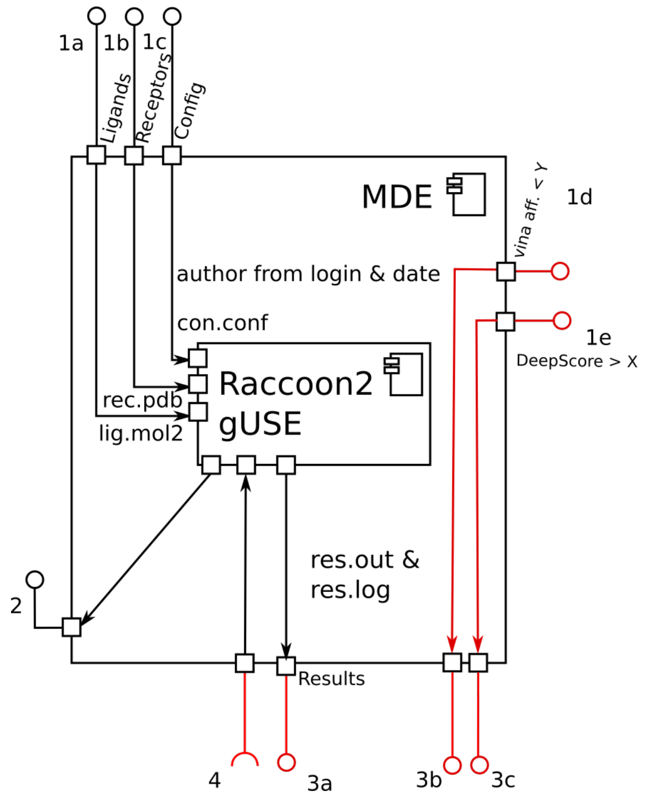

Figure 8.2: The diagram of the MDE: the cloud-enabled Raccoon2 (in red: segments that need to be implemented, in black: existing segments).

3a-c. Raccoon2 needs to provide an interface to the docking results, as well as AutoDock Vina and DeepScore thresholds which should be sent to the MDRR.

4. Raccoon2 requires an interface to receive the suggested ligand for next docking from the MDRR.

Formal description of Raccoon2 The formal description of Raccoon2, shown in Appendix C, was derived from the formal description of the element type MDE (Figure 6.7). The schema *Docking_AutoDockVina* uses *dockingWithConfig*, and has been included in the schema *MolecularDockingEnvironment_Raccoon2* (pages 172 - 173). This is equivalent to the *Docking* and *MolecularDockingEnvironment* schemas from the framework (page 163).

In the generic formal description the element type MDE uses the schema *Docking* to show that a docking can be conducted either without a configuration file (using *dockingWithoutConfig*) or with a configuration file (using *dockingWithConfig*). When describing the specific element Raccoon2 (Figure 8.3), only the definition for *dockingWithConfig* was used, because docking with Raccoon2 uses AutoDock Vina which requires a configuration file. The schema *Docking_AutoDockVina* is derived from *Docking* by leaving out the option to use *dockingWithoutConfig* and changing the schema's name. The schemas *MolecularDockingEnvironment_Raccoon2* and *ViewMolecularDockingEnvi-*

$$
\begin{array}{|l}
\mathit{dockingWithConfig} : (\mathit{LIGAND} \times \mathit{RECEPTOR} \times \mathit{CONFIG}) \rightarrowtail\!\!\!\!\!\!\!\!\!\;\;\rightarrow \mathit{RESULT} \\
\hline
\forall\, l : \mathit{LIGAND};\ r : \mathit{RECEPTOR} \mid l \neq \varnothing \wedge r \neq \varnothing \bullet \exists\, c : \mathit{CONFIG};\ \mathit{res} : \mathit{RESULT} \mid \\
c \neq \varnothing \bullet \mathit{dockingWithConfig}(l, r, c) = \mathit{res}
\end{array}
$$

$$
\begin{array}{|l}
\hline
\mathit{Docking\_AutoDockVina} \\
\mathit{ligand?} : \mathit{LIGAND} \\
\mathit{receptor?} : \mathit{RECEPTOR} \\
\mathit{config?} : \mathit{CONFIG} \\
\mathit{result!} : \mathit{RESULT} \\
\hline
\mathit{config?} \neq \varnothing \wedge \mathit{result!} = \mathit{dockingWithConfig}(\mathit{ligand?}, \mathit{receptor?}, \mathit{config?}) \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\mathit{MolecularDockingEnvironment\_Raccoon2} \\
\mathit{ligands?} : \mathit{LIGANDS} \\
\mathit{receptors?} : \mathit{RECEPTORS} \\
\mathit{config?} : \mathit{CONFIG} \\
\mathit{results!} : \mathit{RESULTS} \\
\mathit{date!} : \mathit{DATE} \\
\hline
\exists\, \mathit{ligand?} : \mathit{ligands?};\ \mathit{receptor?} : \mathit{receptors?};\ \mathit{result!} : \mathit{results!} \bullet \mathit{Docking\_AutoDockVina} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\mathit{ViewMolecularDockingResults\_Raccoon2} \\
\Xi \mathit{MolecularDockingEnvironment\_Raccoon2} \\
\hline
\mathit{results!} \neq \varnothing \\
\hline
\end{array}
$$

Figure 8.3: Excerpt of the Z notation describing Raccoon2 as element of Scenario 1.

ronmentResults_Raccoon2 are the same as the respective generic schemas in all but name.

MDRR: a custom-made MongoDB repository An existing repository can be used as an MDRR, as long as it fits the description of the MDRR element type of the framework. To the best of the candidate's knowledge, no such repository exists. Furthermore, since this is the first scenario implemented using the framework, there is no library of abstract descriptions of existing tools to use for comparison. Creating a custom-made repository that would be used as an MDRR is one solution. The abstract descriptions of the proposed custom-made tool is shown, while Sub-section 8.2.2.2 shows the benefits of using MongoDB as a database.

Figure 8.4 shows the detailed diagram of the proposed custom-made tool by replacing the generic labels of Figure 6.2 with specific ones. The MongoDB-based MDRR would require docking results. It would also require the AutoDock Vina and DeepScore threshold values, which should be forwarded to an element that uses them. Any other data stored in the repository should also be provided for the next elements. The MongoDB-based MDRR requires the input of the ATs in order to keep track of the process, and the DM in order to store the suggested ligand for next docking. This suggestion should be provided to the MDE and subsequently viewed by the user. A comprehensive list of interfaces, derived from the list of MDRR interfaces, can also be created.

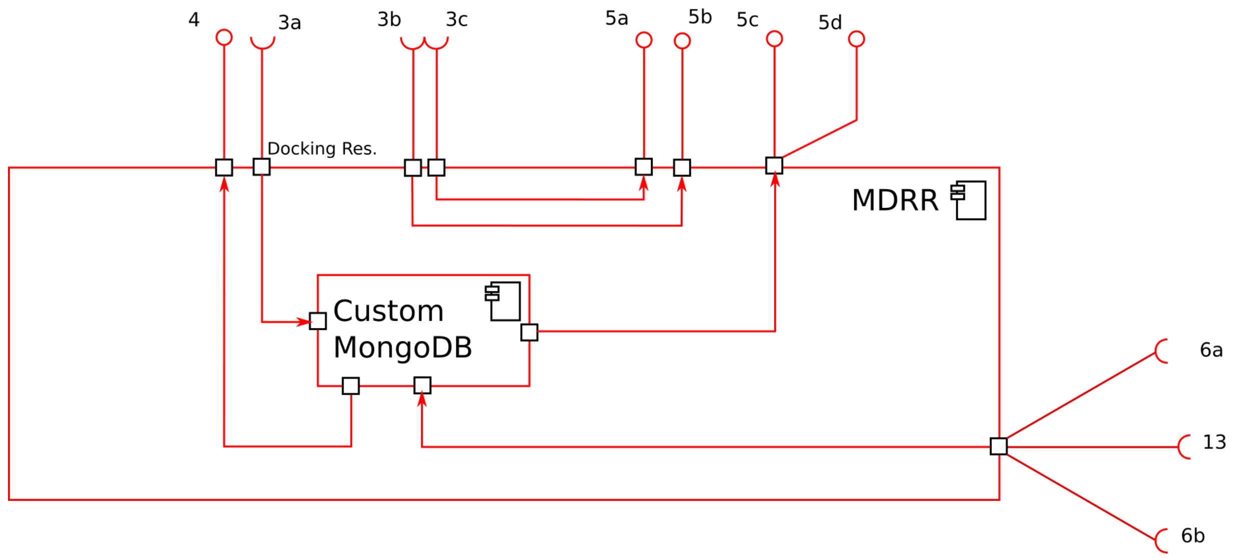

Figure 8.4: The diagram of the custom-made MongoDB-based MDRR.

Interfaces of the custom-made MongoDB-based MDRR

3a-c. The MDRR should require docking results, AutoDock Vina threshold and DeepScore threshold.

4. The MDRR needs to provide the suggested ligand for next docking to the MDE.

5a-b. The MDRR needs to provide the DeepScore threshold to the DeepAlign AT, and the AutoDock Vina threshold to the docking assessment AT.

5c. The MDRR needs to provide a list of all receptors stored in the repository, and the currently used receptor, to the DeepAlign AT.

5d. The MDRR needs to provide a list of previous docking results that have used receptors similar to the current receptor, to the docking assessment AT.

6a-b. The MDRR should require the results from the DeepAlign AT and the docking assessment AT to keep track of the process.

13. The MDRR needs to receive the suggested ligand for next docking from the DM.

Formal description of the custom-made MongoDB-based MDRR The formal description of the proposed custom-made MongoDB-based MDRR (Appendix C, pages 173 - 175), is virtually the same as the description of the element type MDRR. The schema *MolecularDockingResultsRepository_MongoDB* is a replica of *MolecularDockingResults-Repository* from the framework's description, albeit with an altered name. It shows that the repository should store data about the ligand, receptor, configuration file, date and docking result. The fact that the suggestion of ligand for next docking is also stored in the MDRR is modelled using the *decisionRepository*. The data can be inserted into or selected from these model repositories.

AT1: DeepAlign Any existing structural alignment tool can be used as an AT in Scenario 1, as long as it fits the description of the element type AT. Chapter 2 provided an overview of several existing tools, and proposed using the tool DeepAlign. An abstract description of DeepAlign can be created to determine whether it can be the AT. A diagrammatic, textual, and formal description of the interfaces of DeepAlign as an AT will be created. Similarly to the way it was concluded that Raccoon2 can be the MDE, if the diagram, list of interfaces, and formal description of DeepAlign can be derived from the abstract descriptions of the element type AT (Figure 6.4, and Appendix B), then one can conclude that DeepAlign can be used as an AT.

Figure 8.5 shows the detailed diagram of DeepAlign. It shows that replacing the generic labels in Figure 6.4 with DeepAlign-specific ones is possible. A user-provided threshold of the value of DeepScore is required, along with a list of all previous receptors and the currently used receptors which will be compared. The threshold and the results of the pairwise comparison should be sent to another AT which will assess whether the structural similarity score is sufficient to call two receptors similar. A more comprehensive list of interfaces, derived from the list of AT interfaces, can also be created.

Interfaces of DeepAlign

5a. The DeepAlign AT should require the DeepScore threshold.

5c. The DeepAlign AT should require a list of receptors, and a target receptor to calculate the structural alignment.

7a. The DeepAlign AT should provide the DeepScore threshold to an assessment AT.

7b. The DeepAlign AT should provide the structural alignment results along with any meta-data for assessment.

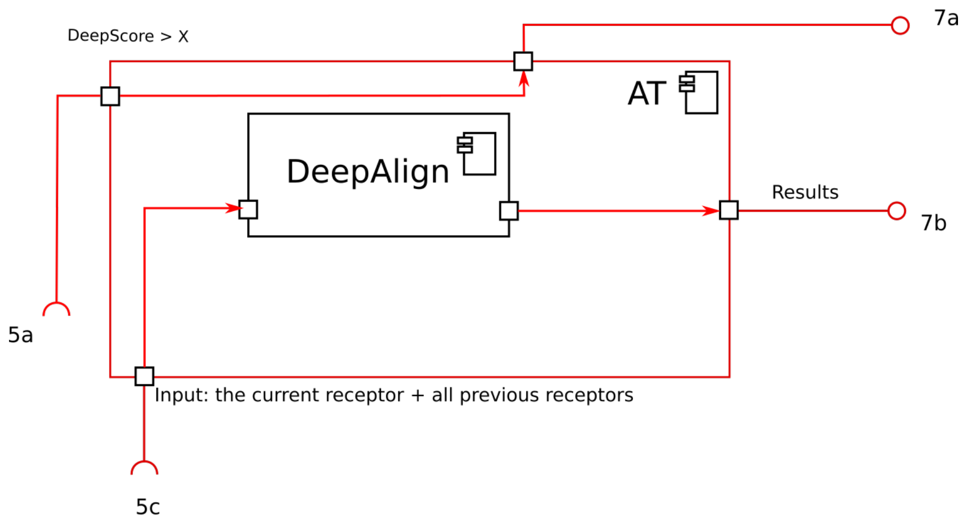

Figure 8.5: The diagram of the AT DeepAlign.

Formal description of DeepAlign The generic formal description of an AT included all possible classes of ATs based on the input they receive. The schema *DeepAlign* (Appendix C, page 176) is derived from the schema *AdditionalTool* of the framework's formal description (Appendix B, page 168). The schema *AdditionalTool* showed that the result of the AT can come from any combination of AT classes based on the input. The schema *DeepAlign* specifies that this needs to be *DeepAlignCore*, a tool that uses previous results, equivalent to *additionalTool_PR* of the framework. Instead of generic previous results, it specifically defines the input as a pair of receptors. In *DeepAlignCore*, it is shown that a *current_receptor* and a *previous_receptor* need to exist, in order for a DeepAlign result based on these two receptors to exist.

AT2: Assess DeepAlign results In an analogous manner to the above elements, it has been decided to use a custom-made tool to assess the results of DeepAlign and filter receptors that are "similar" to the currently used receptor. Since there is no library of already implemented elements, it is reasonable to propose creating a custom-made tool that will do the assessment based on a user-provided threshold of the DeepScore value. Abstract descriptions of this custom-made tool can be derived from the abstract descriptions of the element type AT. The detailed diagram is shown in Figure 8.6. The list of interfaces is as follows.

Interfaces of the custom-made threshold-based DeepAlign assessment AT

7a-b. This AT should require the DeepScore threshold and DeepAlign result (along with any meta-data) as input.

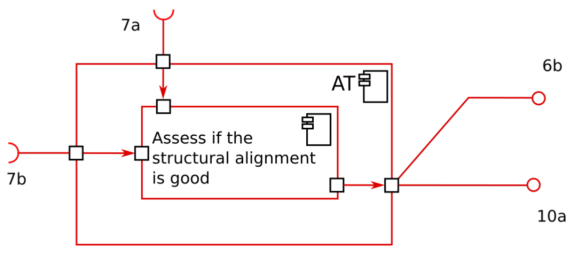

Figure 8.6: The diagram of the AT to assess DeepAlign.

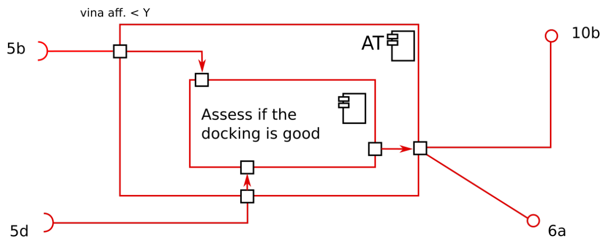

Figure 8.7: The diagram of the AT to assess docking results.

6b. This AT should provide the results of the assessment (whether the DeepAlign result is sufficient to label the two receptors as similar) to the MDRR for storage.

10a. This AT should provide the results of the assessment to the DM for summarising.

Formal description of the custom-made threshold-based DeepAlign assessment AT The schema *AssessDeepAlign* (Appendix C, page 176) is derived from the schema *AdditionalTool* of the framework's formal description (Appendix B, page 168), and *goodDeepAlignResult* is equivalent to *additionalTool_UI_ATR* which uses a user-provided input and results of another AT. In *goodDeepAlignResult* it is explained that the user-provided input needs to be a threshold which will be compared to the DeepScore value of the DeepAlign result. The schema *AssessDeepAlign* shows that a receptor r will be part of the receptors assessed as similar only if the result of *goodDeepAlignResult* is positive.

AT3: Assess docking results An analogous method was used to propose a custom-made tool to assess the docking results and filter "good" docking results. This is useful because the scenario requires a similar receptor which has been "successfully" docked with a ligand. In Scenario 1, there is no library of already implemented elements, so a custom-made tool could conduct the assessment based on a user-provided threshold of the AutoDock Vina affinity value. Figure 8.7 shows the detailed diagram which, along with the other abstract descriptions of this custom-made tool, can be derived from the abstract descriptions of the element type AT. The list of interfaces is as follows.

Interfaces of the custom-made threshold-based docking result assessment AT

5b, 5d. This AT should require the AutoDock Vina threshold and a list of docking results.

6b. This AT should provide the results of the assessment (whether the docking result is good) to the MDRR for storage.

10b. This AT should provide the results of the assessment to the DM for summarising.

Formal description of the custom-made threshold-based docking result assessment AT The schema *AssessPreviousDocking* (Appendix C, page 177) is derived from the schema *AdditionalTool* of the framework's formal description (Appendix B, page 168), and *goodDocking* is equivalent to *additionalTool_UI_PR* which uses a user-provided input and previous results. In *goodDocking*, it is shown that the user-provided input needs to be a threshold which will be compared to the docking score of the previous docking result. The schema *AssessPreviousDocking* shows that a previous docking *result* will be part of the docking results assessed as similar only if the result of *goodDocking* is positive.

DM: custom-made element The DM combines the result of AT2 (if receptors are similar) and AT3 (if the docking is good). It should create a list of receptors sorted by alignment score first (most similar to the current receptor), then by the AutoDock Vina score (part of best docking results). Based on this sorted list of receptors, the MDRR can select ligands that have been docked with them and suggest these ligands for a next docking. The DM is an element type that is very specific to every scenario, so it will likely be a custom-made tool. Nevertheless, abstract descriptions of this custom-made tool can be derived from the abstract descriptions of the DM element type. The detailed diagram is shown in Figure 8.8. The list of interfaces is as follows.

Interfaces of the custom-made DM

10a-b. The DM should require the result from AT2 and AT3.

13. The decision, in this case the sorted list of most similar receptors that were part of the best docking results, should be provided to the MDRR.

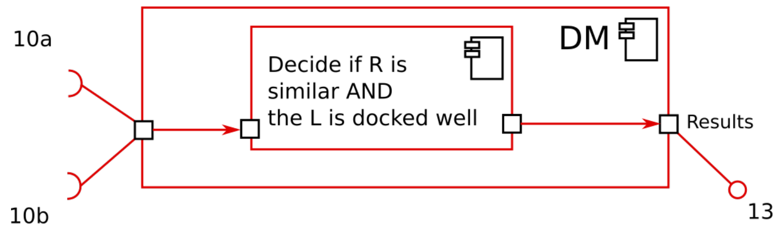

Figure 8.8: The diagram of the DM.

Formal description of the custom-made DM The formal description of the custom-made DM is derived from the formal description of the DM element type of the framework. The schema *DecisionMaker_Custom* (Appendix C, page 177) is equivalent to the schema *DecisionMaker* of the generic element type DM (Appendix B, page 171). It specifically uses the *makeADecisionAdditionalTool* which requires the results of two ATs as input. In this case the first result is a list of filtered receptors and the second is a list of filtered previous docking results.

Detailed diagram of entire Scenario 1 Based on the analysis shown above, a complete detailed diagram of Scenario 1 can be created (Figure 8.9). It confirms that Scenario 1 fits the framework because it is equivalent to the detailed diagram of the entire framework (Figure 6.1). The items drawn in red are ones that need to be implemented, while the ones in black are existing items that can be used. There are a total of 21 interfaces between these elements (numbered according to the numbering-scheme of the framework).

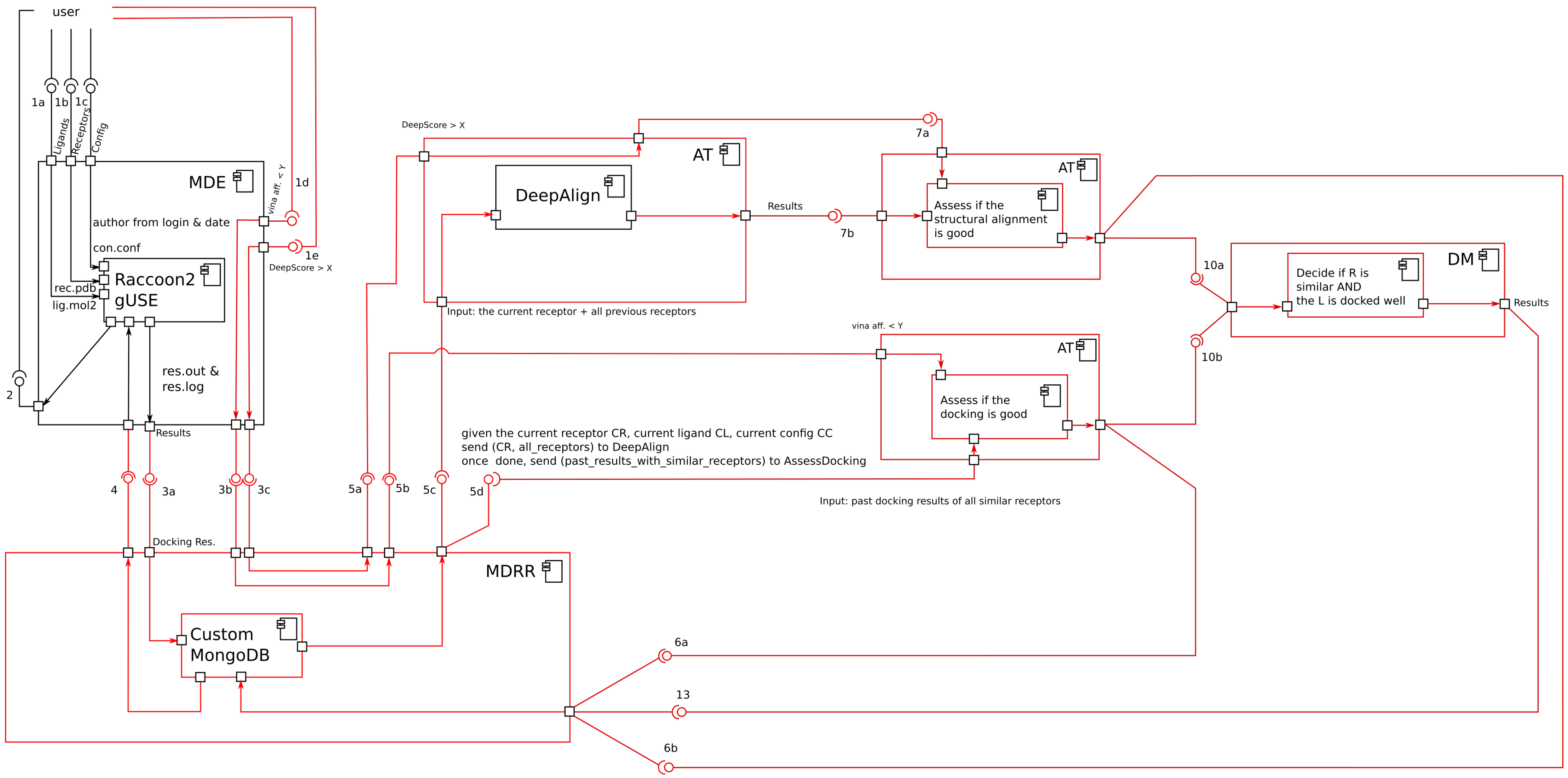

Figure 8.9: The detailed diagram of Scenario 1.