⇒ *commmand to call an agent*: [components search –class concept_reusable_kb_component]*

⇒ *result*:*

[Found all Knowledge Base components whose specifications have been installed]

- *Installing the Knowledge Base component*

⇒ *sc-agent*:*

[ScComponentManagerInstallAgent]

⇒ *commmand to call an agent*: [components install –idtf part_polygons]*

⇒ *result**

[A Knowledge Base component in the form of subject domain of polygons was established.]

⇒ *note*:*

[After performing this step, we can find the concept "multiple" in the web interface and browse its semantic neighbourhood. But it is worth noting that the subject domain of triangles, which is a private subject domain of polygons, is empty.]

- *Installing the Knowledge Base component*

⇒ *sc-agent*:*

[ScComponentManagerInstallAgent]

⇒ *commmand to call an agent*: [components install –idtf part_triangles]*

⇒ *result*:*

[The Knowledge Base component is installed in the form ofof subject domain of triangle]

⇒ *note*:*

[After performing this step, we can find the concept "triangle" in the web interface and browse its semantic neighbourhood. It is worth noting that the subject domain of triangles, which is a private subject domain of polygons, is fully described and compatible with the subject domain of polygons.]

- *Creating two sets of triangles*

⇒ *note*:*

[At this step it is necessary to find the class "triangle" in the webinterface, create two sets of triangles and add elements to them. It is necessary to specify that the sets and their elements belong to the class "triangle".]

⇒ *example*:*

[triangles_1 = {ABC, CDE, XYZ}, triangles_2 = {MNK, CDE, XYZ}]

⇒ *note*:*

[After performing this step, you can check that no operations on sets can be performed now. This can be verified by right-clicking on the node "triangles_1".]

- *Search for all available problem solver components in the library*

⇒ *sc-agent*:*

[ScComponentManagerSearchAgent]

⇒ *commmand to call an agent*: [components search –class concept_reusable-ps_component]*

⇒ *result*:*

[Found all components of the problem solver whose specifications are installed.]

- *Installing the components of the problem solver*

⇒ *sc-agent*:*

[ScComponentManagerInstallAgent]

⇒ *commmand to call an agent*: [components install –idtf agent_of_finding_intersection_of_sets]*

⇒ *result*:*

[A problem solver component for finding the intersection of two sets is established.]

⇒ *note*:*

[After this step, you can check that you can now perform an operation on sets. In the web interface, search for the concept "installed components" and select the node of the desired agent *agent_of_finding_intersection_of_sets)* and run the set intersection agent using the example of two previously created triangle sets. The intersection of the two sets will be found. But it should be noted that this way of launching the agent is long and inconvenient.]

- *Search for all available interface components in the library* *sc-agent*:*

[ScComponentManagerSearchAgent]

⇒ *commmand to call an agent*: [components search class concept_reusable_interface_component]*

⇒ *result*:*

[Found all interface components whose specifications have been downloaded.]

- *Installing the user interface component*

⇒ *sc-agent*:*

[ScComponentManagerInstallAgent]

⇒ *commmand to call an agent*: [components install idtf menu_of_agent_of_finding_intersection_of_sets]*

⇒ *result*:*

[Installed an interface component for an agent to find the intersection of two sets.]

⇒ *note*:*

[After this step, the intersection finder can be invoked using a button in the interface, which is much faster and more convenient than the first method. This can be checked by calling the agent to find the intersection of two sets using the example of triangle sets (triangles_1 and triangles_2).]

- *Setting a logical formula component*

⇒ *sc-agent*:*

[ScComponentManagerInstallAgent]

⇒ *commmand to call an agent*: [components install idtf lr_about_isosceles_triangle]*

⇒ *result*:*

[Established a component with a logical formula for determining whether a triangle is isosceles or not.]

⇒ *note*:*

[If you go to the web interface after performing this step, create the necessary fragment for the geometry logic formula parcels and try to run the logic output, it fails because the logic output component is missing.]

- *Setting the logic inference component*

⇒ *sc-agent*:*

[ScComponentManagerInstallAgent]

⇒ *commmand to call an agent*: [components install idtf scl_machine]*

⇒ *result*:*

[Logic inference machine is installed.]

⇒ *note*:*

[After performing this step go to the web-interface, create the necessary fragment to send a logical formula on geometry and try to run the logical output, then the formula will generate the necessary fragment of the Knowledge Base. However, this is still not very convenient.]

- *Installing the user interface component*

⇒ *sc-agent*:*

[ScComponentManagerInstallAgent]

⇒ *result*:*

[Interface component for logic output component installed]

⇒ *note*:*

[After performing this step in the web interface after creating the necessary fragment to send the formula on geometry, you can easily call the logical output agent through the interface component.]

⇒ *result*:* [The functionality of the system is extended. A system capable of logical inference and finding intersection of sets is obtained. The system has interface components corresponding to these agents. The Knowledge Base on geometrical figures (polygons and triangles) is also obtained.]