- *installing genetic subsystems*
- *search for reusable components of ostis-systems*
- *installing reusable components*
- *configure ostis system*

    := [configuration of the ostis system]

  ⟩

Whether installing reusable components into an already created system or creating a system from scratch, constructs are created in the ostis-system knowledge base to denote which components are installed into the system. Figure 7 shows an example of a construct specifying which components are installed in an ostis system.

Finding and installing an ostis-platform is necessary because different ostis-platforms may be suitable for different classes of tasks and components to be installed in the generated system.

By installing generic subsystems, the functionality of the ostis-system being generated can be greatly expanded. The OSTIS Metasystem Library contains many subsystems often used in other ostis-systems. Typical subsystems include, for example, the subsystem for collective design of ostis-systems, natural language interface, training subsystem, security subsystem and others.

Thanks to the extensive search functionality of reusable ostis-system components, it is possible to search for any components according to various criteria and combinations thereof.

Customising an ostis-system implies setting parameters to specify the peculiarities of the system operation, as well as specifying which users are administrators, developers, experts, and users of the created ostis-system.

In addition to user actions when creating an ostissystem, the ostis-system generation subsystem also registers the created ostis-system in the OSTIS Metasystem. Thus, the OSTIS Metasystem is able to monitor and update the status of the components of this system.

VIII.User interface of component manager and library of reusable components of ostis-systems

The multi-component manager for ostis-systems has a console interface. The component manager is connected to sc-memory as a dynamic component, so it does not require a restart, and you can immediately see the installed components in a running system. Let's look at the commands with which you can use the component manager. Each command calls the corresponding agent. Agent-based architecture allows you to implement any user interfaces for the component manager of ostis-systems. Any variant of the component manager user interface creates sc-constructs in scmemory that are needed to invoke the corresponding scagent.

action. Set the specifications of reusable ostis-system components

⇒ agent*:

. [ScComponentManagerInitAgent]

⇒ command to initiate an action*:

. [components init]

action.Find specifications for reusable ostis components

⇒ agent*:

. [ScComponentManagerSearchAgent]

⇒ command to initiate action*:

. [components search]

⇒ possible flags*:

- [*author*]
- [*class*]
- [*explanation*]

. When using the search command with the author flag, you must list the system identifiers of the sc nodes that denote the authors of the reusable component. The class flag is used to pass the class name to the component manager to search for components belonging to this class. The explanation flag is used to specify a natural-language fragment that is a substring of the component's explanation. If multiple search flags are listed, components that satisfy all search criteria simultaneously will be found. If you use the component search command without flags, all components whose specifications are downloaded will be found.

action. Install reusable ostis components

⇒ agent*:

. [ScComponentManagerInstallAgent]

⇒ command to initiate action*:

. [components install]

⇒ flags*:

- [*idtf*]

The component install command requires the mandatory idtf flag, which the component manager uses to search by system ID for the components to be installed and create the necessary construct to call the component install agent.
The interface of the reusable component library is graphical (Fig. 8. It displays the components that are in the library and provides access to search, browse, and install them.

As an example of a library of reusable components of ostis-systems for consideration of an example of work, let's take the OSTIS Metasystem Library.

Installation of the OSTIS Metasystem is performed using the following command sequence.

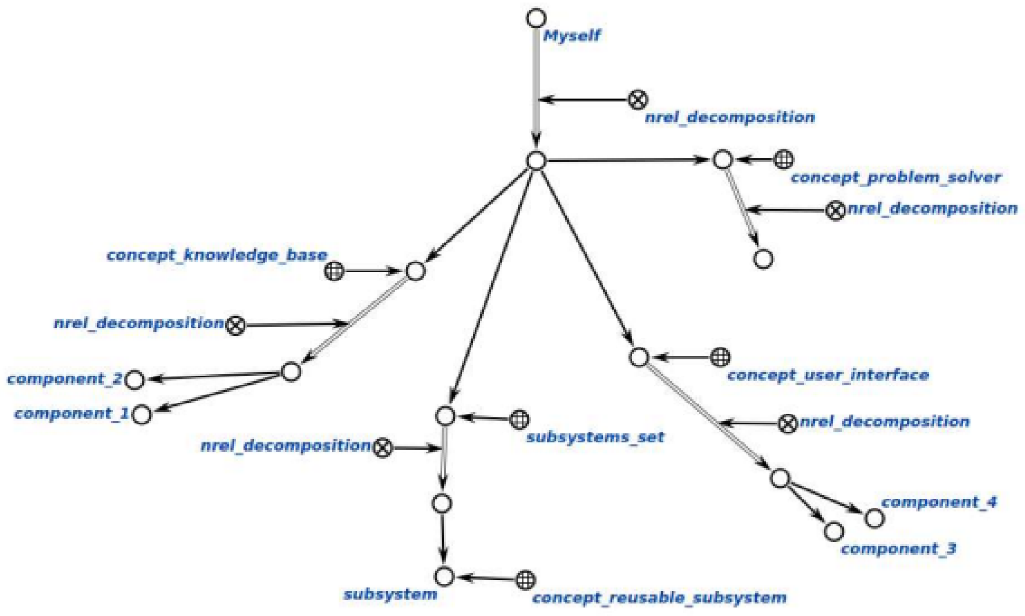

Figure 7. Formalisation of installed components to the ostis-system

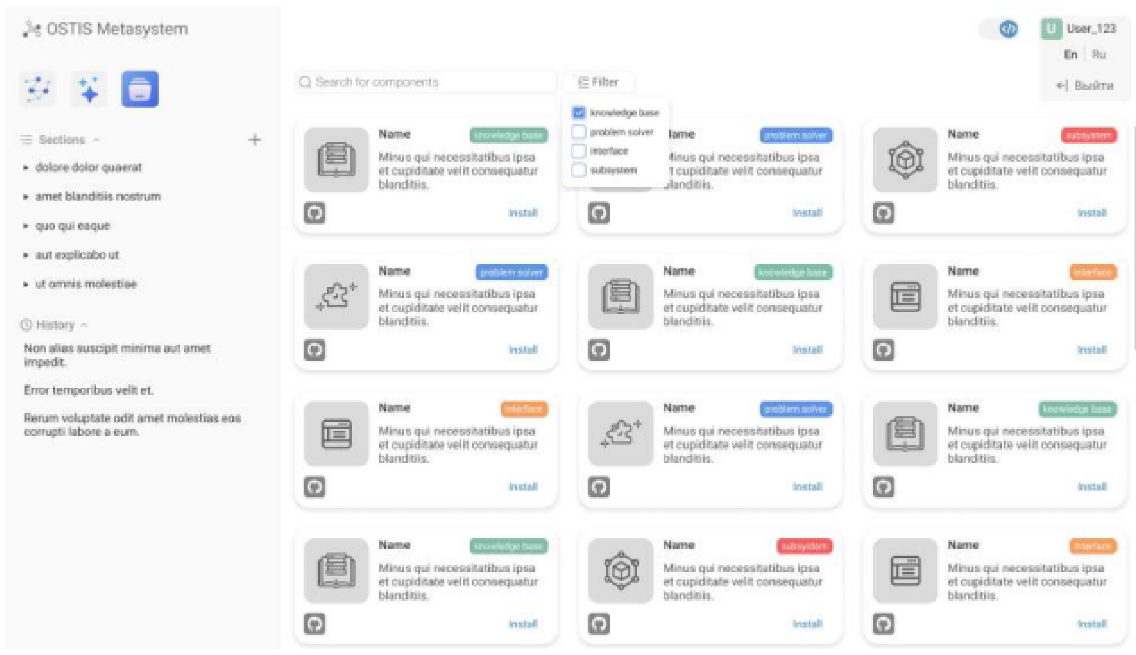

Figure 8. User interface of a library of reusable components of ostis-system

OSTIS Metasystem

⇒ installation stages*:

⟨

- Repository cloning
  - ⇒ terminal command*:
    [git clone https://github.com/ostis-ai/ostis-metasystem]
- Change dorectory to the project root
  - ⇒ terminal command*:
    [cd ostis-metasystem]
- OSTIS Metasystem installation
  - ⇒ *terminal command*:*
    [./scripts/install_metasystem.sh]
- Run sc-component-manager
  - ⇒ *terminal command*:*
    [./scripts/run_sc_component_manager.sh]

⟩

⇒ *component installation procedure*:*

- Install all the reusable components specifications
  - ⇒ *command*:*
    [component init]
- Reusable components searching
  - ⇒ *command*:*
    [components search]
- Installation of reusable component
  - ⇒ *command*:*
    [components install – – idtf <identifier>]

⟩

Creation of the core
⇒ stages*:

⟨ Install all the reusable components specifications
  - ⇒ sc-agent*:
    [ScComponentManagerInitAgent]
  - ⇒ command to call an agent*:
    [components init]
  - ⇒ result*:
    [All the reusable components specifications from OSTIS Library are installed.]
- Search reusable user interface components
  - ⇒ sc-agent*:
    [ScComponentManagerSearchAgent]
  - ⇒ commmand to call an agent*:
    [components search- -class concept-reusable-interface-component]
  - * Install sc-models of user interface interpreter
    - ⇒ *sc-agent*:*
      [ScComponentManagerInstallAgent]
    - ⇒ *commmand to call an agent*:*
      [components install idtf scweb]
  - ⇒ *result*:*
    [sc-web is intalled by specification.]

⇒ *note*:*
[If you start the web interface after this step, only the start page will load, because the knowledge base is currently empty.]]

- Searching for Knowledge Base components

⇒ *sc-agent*:*
[ScComponentManagerSearchAgen]

⇒ *commmand to call an agent*:*
[components search class concept-reusable-kb-component]

⇒ *result*:*
[Received all components for which their specification states that they are Knowledge Base components.]

⇒ *note*:*
[If you start the web interface after this step, only the start page will load, because the knowledge base is currently empty.]]

- Reusable components searching
  - ⇒ *command*:*
    [components search]

Installation of reusable component
⇒*command*:*
[components install – – idtf <identifier>]

- OSTIS Standard installation

components

⇒ *agent*:*
[ScComponentManagerInstallAgent]

⇒ *commmand to call an agent*:*
[components install idtf ostis-standard]

Extension of kernel functionality

⇒ *stages*:*
[ScComponentManagerInstallAgent]

- Search for all available Knowledge Base components in the library

⇒ *sc-agent*:*
[ScComponentManagerInstallAgent]

- ScComponentManagerSearchAgent