-
  - Reusable component — Reusable component of ostis-system;
  - (Reusable) component specification — Specification of reusable component of ostis-system;
  - Storage (GitHub) — A repository of components and specifications, such as GitHub;
  - Other library — a third-party library of reusable components.
- relationship
  - update;
  - use;
  - subsystem;
  - search;
  - connect;
  - store;
  - link;
  - installation
- attributes
  - OSTIS Metasystem library — OSTIS Metasystem library of reusable components of ostissystem;
  - OSTIS Metasystem manager — OSTIS Metasystem manager of reusable components of ostissystem;
  - search arguments — search arguments for reusable components
    - Author — the author of the component;
    - Class — the class of the component;
    - Identifier — the name of the component;
    - Explanation — explanation of the component.

Component Library is a library of reusable components of ostis-systems, which is a subsystem of them.The library's knowledge base is a repository of reusablecomponent specifications, and the library also providesan interface to visualise and manage component specifications of the user's system.

Component Manager — is a reusable componentmanager for ostis systems that is a subsystem for installing, downloading, and tracking components and theirspecifications for both the user's system and other systems that store reusable components.

The entity-relationship diagram for the component-manager from the point of view of the ostis-system useron the example of the OSTIS Metasystem Library (Fig. 3)contains the following information. The developer uses some ostis-system, a subsystem ofwhich is a reusable component manager and optionallya library of reusable components. The developer can update a reusable component from the OSTIS MetasystemLibrary *using the OSTIS Metasystem* Reusable Component Manager. The developer can use the component-manager to search for components in the OSTIS Metasystem and third-party libraries known tothe managerby criteria such as component author, class, identifier,and component explanation fragment. The developer can connect to other component libraries. The developer canalso install the found components into his system.

The entity-relationship diagram for a component library depicts the main relationships between a system,in this case the OSTIS Metasystem, and its subsystems(manager and library) in terms of the storage of components and their specifications.(Fig. 4). The diagramcontains the following information.

OSTIS Metasystem has a subsystem in the form of a library and a component manager. The OSTIS Metasystem library stores many specifications of reusablecomponents. Since all components are stored via GitHub,the manager uses the links provided in the componentspecifications to access them. The component specifications have a link to the repository that stores thecomponent itself.

Updating reusable OSTIS Metasystem components inthe OSTIS Metasystem Library is done through the OSTIS Metasystem Manager and the GitHub repository.The manager allows you to select the required versions of components and install the corresponding component specifications in the OSTIS Metasystem Library. According to these specifications, users using the OSTIS Metasystem will be able to learn about components and install them in their systems. A component repository such as GitHub has many repositories, each of which can store any number of components and their specifications for installation on other systems using the reusable component manager.

VI. Reusable components installation process

Let's consider the functions of the manager of reusable components of ostis-systems.

reusable ostis-system components manager
:= [component manager]
⇒ functions * :
{● reusable component installation
⇒ partitioning * :
{● scnitem
scnitem component download
● setting component dependencies
● translating component scs files into system sc-memory
}
● search for specifications of reusable components
● downloading specifications of reusable components
}

In general, component installation consists of the following steps:

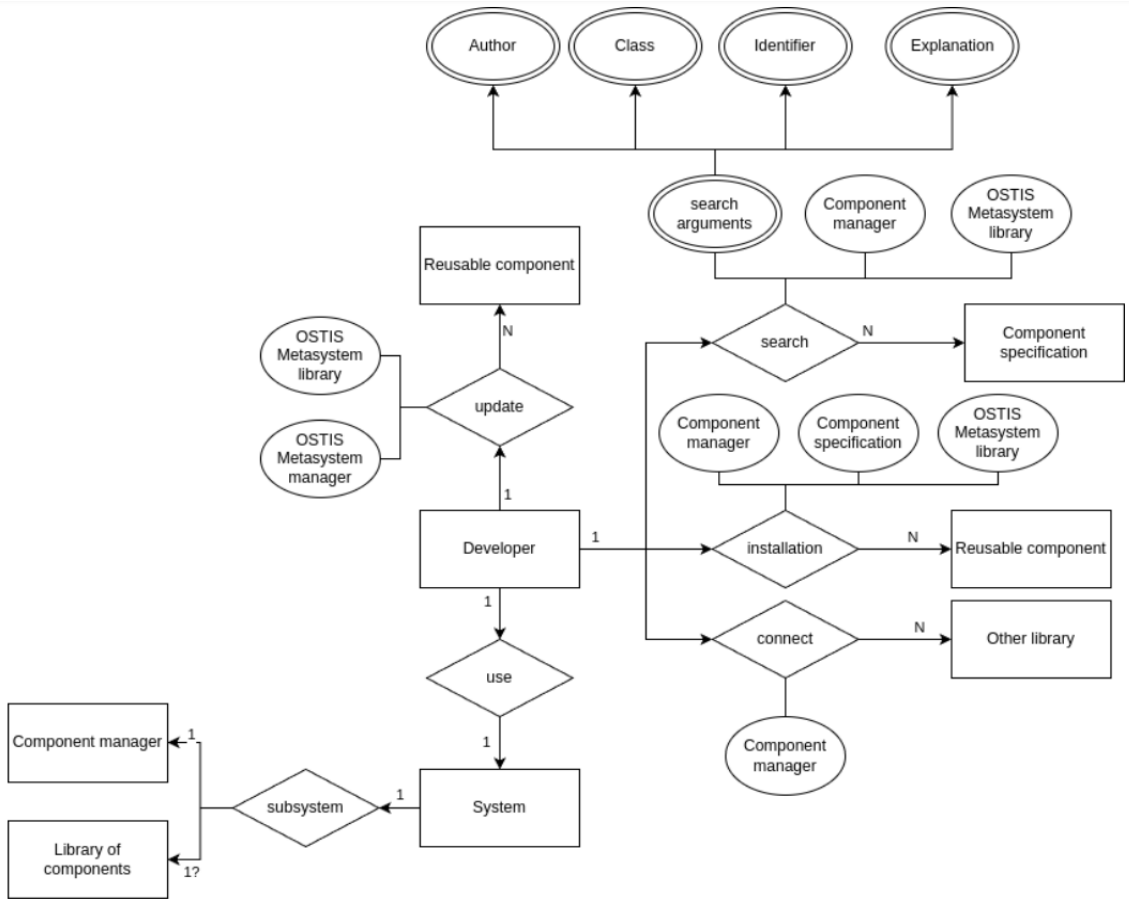

Figure 1: Entity-relationship diagram for a component library

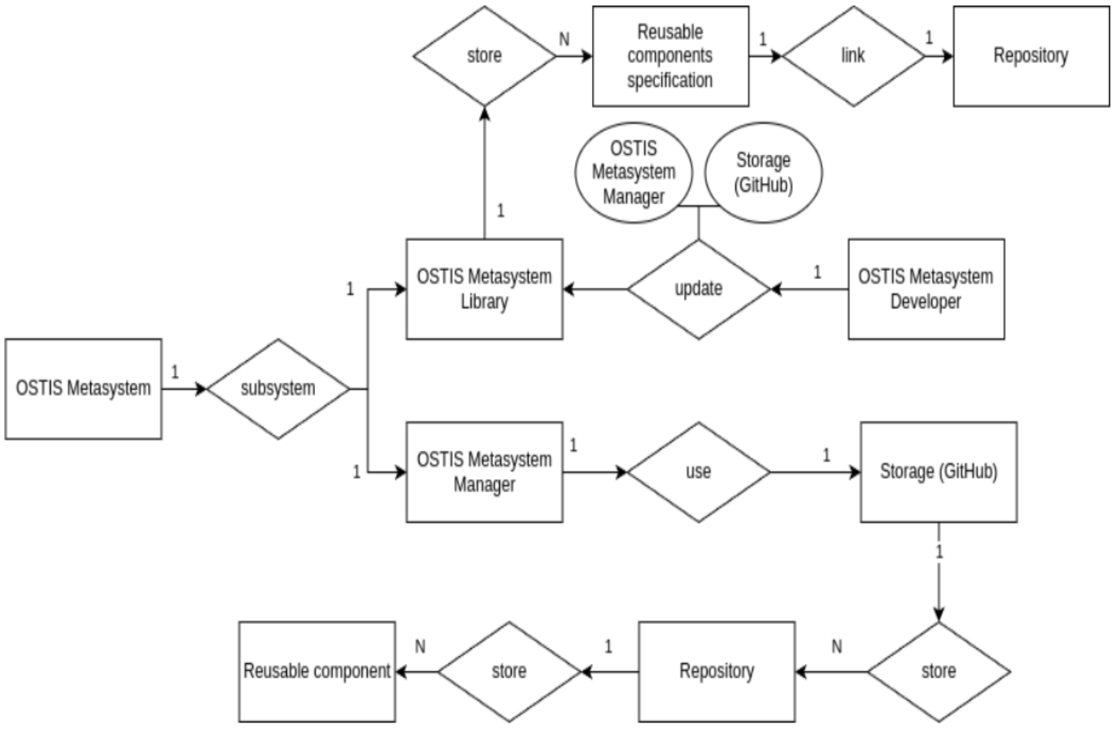

Figure 2: Entity-relationship diagram for a component library

- initiation of the agent to install all component specifications described in the knowledge base;
- to initiate the agent to search for the required component specifications in the knowledge base;
- initiating the agent to install the selected components.

After the user has initiated the component specification agent, the component manager will search the component specifications for references to the component specification repository. The specification file is called specifcation.scs and is stored in the folder with the reusable component itself. If the component manager was able to locate this file, it will load the file into sc memory. The component specification may include:

- identifier of the component;
- classes to which the component belongs;
- indicating the author of the component;
- indicating a note for the component;
- specifying how the component is installed;
- specifying the location (link) where the component is stored.

After the specifications are set, the user can search forcomponents or install them.

The design of initiating the action of searching forcomponent specifications is shown in Fig. 5.

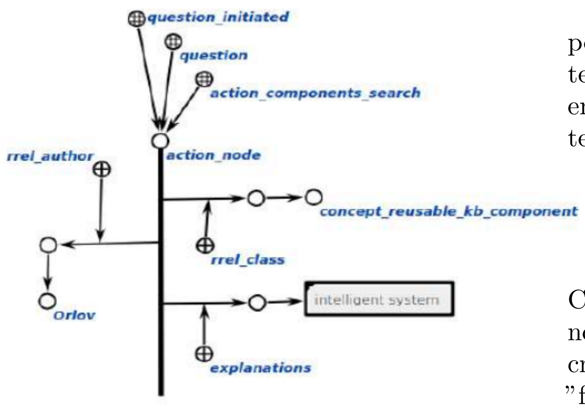

Figure 3: Example of calling the reusable components specification search agent

Three parameters are possible for the agent to search for component specifications: class, author, note. According to the above example, the manager will searchfor specifications of components created by Orlov M.K., belonging to the class multiple-used knowledge base component and having the substring "intelligent system" in the note.

For the agent to find all components known to the system, then class of reusable ostis-system components must be passed as a parameter, and then the agent will find all specifications of reusable components stored in the system.

In order to install a component, you need to pass it as a parameter when calling the component installation agent. The agent will find the required component and its semantic neighbourhood that specifies the storage location of the component and how to install it, then the agent will install the reusable component in the ostissystem.

The design of the component installation agent initiation is shown in Fig 6:

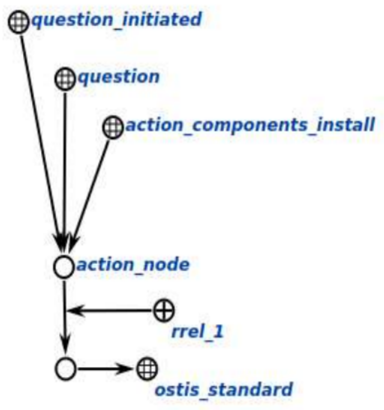

Figure 4: Example of calling the reusable components installation agent

Thus, the component manager and reusable component library allow systems to create and design intelligent systems based on off-the-shelf solutions, thus enhancing system interoperability and simplifying system development.

VII. Specification of ostis-system generation

Component-based design of computer systems means not only extending the functionality of a system already created in some form, but also creating an entire system "from scratch".

For the generation of ostis-systems the manager of reusable components of ostis-systems is used, which provides the possibility to assemble the system from the components available from the libraries of reusable components of ostis-systems.

The following typical sequence of user actions is used to generate ostis-systems.

generation of ostis-systems
:= [creation of ostis systems]
⇒ generalised sequence of user actions * :
⟨• search for ostis platform
• installing the ostis platform
• search for generic subsystems