LabWork №.1

Ivan Zhilin

September 2024

npm package manager has a component library (npm Registry) and a command -line user interface. The source code for the package manager and related npm tools can be found at https://github.com/npm. The most commonly used npm commands are:

- initializing the project (creating the package.json file);
- install all packages from the package.json file;
- install a package by name;
- deleting a package by name;
- check for obsolete packages;
- display help;
- view installed packages;
- search for packages;
- update packages.

The component approach to the design of computer systems can be implemented within various languages, platforms, and applications. Let us consider some of them.

The ontology implemented in ***OWL*** (Web Ontology Language) is a set of declarative statements about the entities of the dictionary of a subject domain (discussed in more detail in [9]). OWL assumes the concept of an "open world", according to which the applicability of subject domain descriptions placed in a specific physical document is not limited only to the scope of this document — the contents of the ontology can be used and supplemented by other documents adding new facts about the same entities or describing another subject domain in terms of this one. The "openness of the world" is achieved by adding a URI to each element of the ontology, which makes it possible to understand the ontology described in OWL as part of a universal unified knowledge.

The ***IACaaS platform*** is designed to support the development, management, and remote use of applied and instrumental multi-agent cloud services (primarily intelligent ones) and their components for various subject domains [10].

The IACPaaS platform supports:

- the basic technology for the development of applied and specialized instrumental (intelligent) services using the basic instrumental services of the platform that support this technology;
- a variety of specialized technologies for the development of applied and specialized instrumental (intelligent) services, using specialized platform tool services that support these technologies.

The IACPaaS platform also does not contain means for a unified representation of the components of intelligent computer systems and means for their specification and automatic integration.

Based on the analysis carried out, it can be said that at the current state of development of information technologies, there is no comprehensive library of reusable semantically compatible components of computer systems and corresponding component management tools. Thus, it is proposed to implement a library and an appropriate component management tool that will implement seamless integration of components, ensure semantic compatibility of systems and their components, and significantly simplify the design of new systems and their components.

III. Proposed approach

Within this article, it is proposed to take the OSTIS Technology [11] as a basis, the principles of which make it possible to implement a library of semantically compatible components of intelligent computer systems and, accordingly, provide the ability to quickly create knowledge-driven systems using ready-made compatible components.

The systems developed on the basis of the OSTIS Technology are called *ostis-systems*. The *OSTIS Technology* is based on a universal method of semantic representation (encoding) of information in the memory of intelligent computer systems, called an *SC-code*. Texts of the *SC-code* (sc-texts) are unified semantic networks with a basic set-theoretic interpretation, which allows solving the problem of compatibility of various knowledge types. The elements of such semantic networks are called *sc-elements* (*sc-nodes* and *sc-connectors*, which, in turn, depending on orientation, can be *sc-arcs* or *sc-edges*). The *Alphabet of the SC-code* consists of five main elements, on the basis of which SC-code constructions of any complexity are built, including more specific types of scelements (for example, new concepts). The memory that stores SC-code constructions is called semantic memory, or *sc-memory*.

or sc-memory. Within this article, fragments of structured texts in the SCn code [12] will often be used, which are simultaneously fragments of the source texts of the knowledge base, understandable to both human and machine. This allows making the text more structured and formalized, while maintaining its readability. The symbol ":=" in such texts indicates alternative (synonymous) names of the described entity, revealing in more detail certain of its features.

The basis of the knowledge base within the *OSTIS Technology* is a hierarchical system of subject domains and ontologies.

In order to solve the problems that have arisen in the design of intelligent systems and libraries of their reusable components, it is necessary to adhere to the general principles of the technology for intelligent computer systems design, as well as meet the following requirements:

- ensuring compatibility (intelligent) of components of intelligent computer systems based on the unifying representation of these components;
- clear separation of the process of developing formal descriptions of intelligent computer systems and the 51 process of their implementation according to this description;
- availability of an ontology for component design of intelligent computer systems, including (1) a description of component design methods, (2) a model of a *library of reusable components*, (3) a model of a *specification of reusable components*, (4) a complete *classification of reusable components*, (5) a description of means for interaction of the developed intelligent computer system with *libraries of reusable components*;
- availability of *libraries of reusable components of intelligent computer systems*, including component specifications;
- availability of means for interaction of the developed intelligent computer system with libraries of reusable components for installation of any types of components and their management in the created system. The installation of a component means not only its transportation to the system (copying scelements and/or downloading component files) but also the subsequent execution of auxiliary actions so that the component can operate in the system being created.

Based on this, in order to solve the problems set within this article, it is proposed to develop the following system of subject domains and corresponding ontologies:

- Subject domain of reusable ostis-systems components
- Subject domain of a library of reusable ostis-systems components
- Subject domain of the manager of reusable ostis-systems components

IV. Concept of reusable component of ostis-systems

The Subject *domain of reusable ostis-systems components* describes the concept of a reusable component, the classification of components, and their general specification. This subject domain allows creating new and specifying existing components to add them to the library

As a *reusable ostis-systems component*, a component of some ostis-system that can be used within another ostis-system is understood (see [13]). This is a component of the ostis-system that can be used in other ostis-systems (***child ostis-systems***) and contains all those and only those sc-elements that are necessary for the functioning of the component in the child ostis-system. In other words, it is a component of some ***maternal ostis-system***, which can be used in some child ostis-system. To include a reusable component in some system, it must be installed in this system, that is, all the sc-elements of the component should be copied into it and, if necessary, auxiliary files, such as the source or compiled component files. *Reusable components* must have a unified specification and hierarchy to support compatibility with other components. The compatibility of *reusable components* leads the system to a new quality, to an additional extension of the set of problems to be solved when integrating components.

reusable ostis-systems component
:= [typical ostis-systems component]
:= [reused ostis-systems component]
:= [reusable OSTIS component]
:= [ostis-systems ip-component]
:= *frequently used sc-identifier**:
[reusable component]
⊂ *ostis-system component*
⊂ *ostis-system component*

The requirements for *reusable ostis-systems components* inherit the common requirements for the design of software components and also include the following ones [14]:

- there is a technical possibility to embed a reusable component into a child ostis-system;
- a reusable component should perform its functions in the most general way, so that the range of possible systems in which it can be embedded is the widest;
- compatibility of a reusable component: the component should strive to increase the level of negotiability of ostis-systems in which it is embedded and be able to be automatically integrated into other systems;
- self-sufficiency of components, that is, their ability to operate separately from other components without losing the appropriateness of their use.

In the *Subject domain of the library of reusable ostis-systems components*, the most common concepts and principles are described, which are valid for any *library of reusable components*. This subject domain allows building many libraries, each of which will be semantically compatible with any other built according to the proposed principles. Such libraries store components and their specifications for use in child ostis-systems. An example of a specification of a reusable ostis-systems component is shown in Figure 2.

Versions for the full contents of the *Subject domain of reusable ostis-systems components* and the *Subject domain of the library of reusable ostis-systems components* are represented in the work [15].

The ***manager of reusable ostis-systems components*** is the main means of supporting component design of intelligent computer systems built by the *OSTIS Technology* ([16]). It allows installing reusable components in ostis-systems and controlling them. The *Subject domain of the manager of reusable ostis-systems components* contains the full specification for the manager of ostis-systems components, the requirements for the component

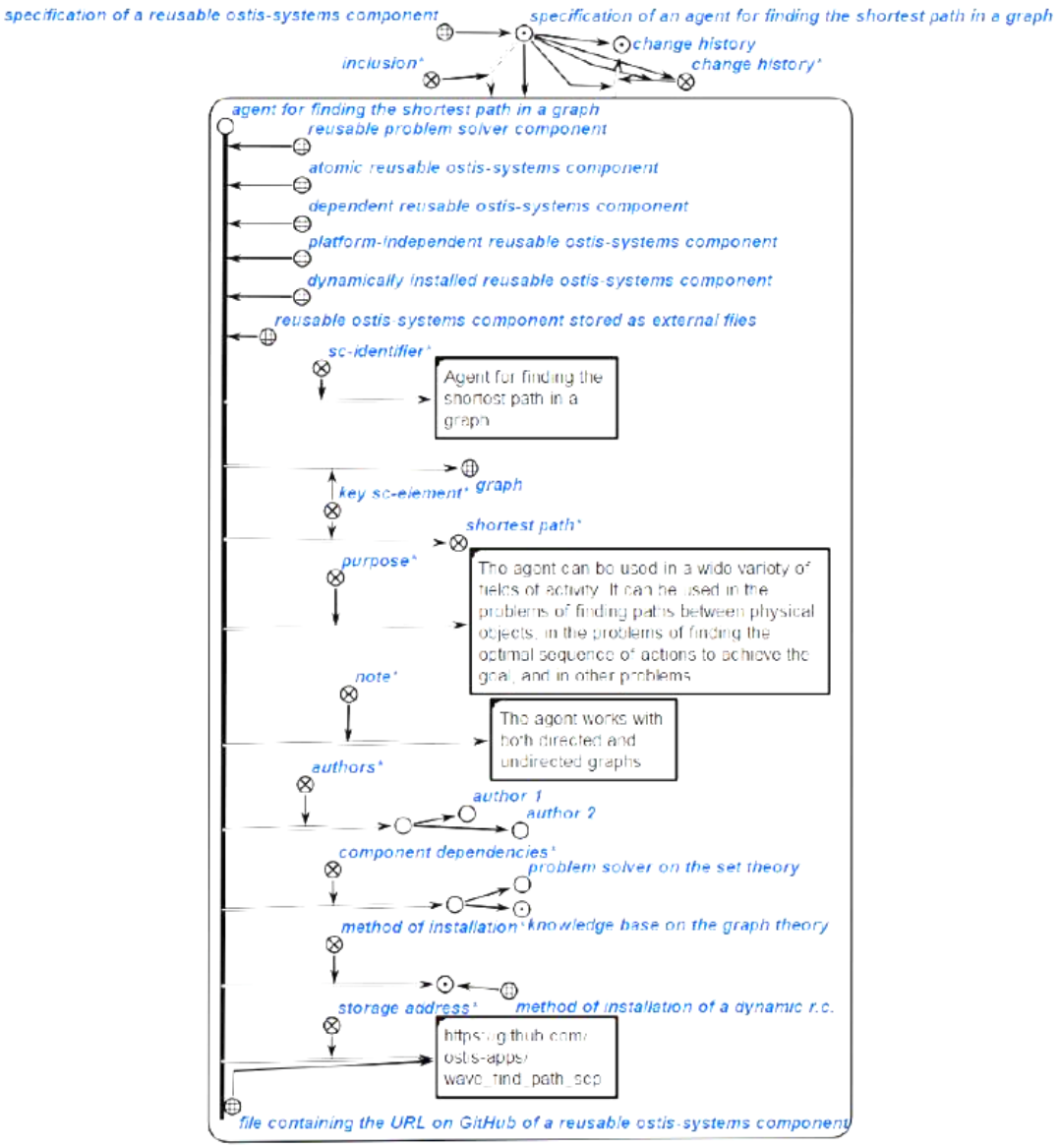

Figure 2: An example of a specification of a reusable ostis-systems component

manager, its functionality, the specification of the implementation option for the manager of ostis-systems components, including the sc-model of the knowledge base, the problem solver, and the interface

V. Architecture of component manager and library of reusable components of ostis-systems

To install reusable components of ostis-systems it is necessary to have a special subsystem in the system: a manager of reusable components of ostis-systems. The component manager interacts with the user and with the library of reusable components. To clarify the specifics of such interaction, diagrams are developed and SC-constructions necessary to initiate actions when working with the component manager are depicted.

Fig. 3 shows the entity-relationship diagram for the component manager, describing the current state of the component manager functionality. The diagrams (Fig. 3 and Fig. 4) use the following concepts:

- of entities
  - Developer — a developer (of their local system);
  - OSTIS Metasystem Developer;
  - Component manager — manager of reusable components of ostis-systems;
  - Library of components;
  - System — user's system