Such an sc-memory model can be easily described in the *sc-language*, that is, in the sublanguage of the *SCcode*. Such a language allows describing how texts of a language are represented inside the memory of the *ostis-platform* in the same language. At the same time, not only the unification of representing information processed by the *ostis-platform* and information describing the *ostis-platform* itself is observed, but also opportunities are given for expanding and using the language in the process of evolution of the *ostis-platform* and its components, including those in the process of evolution of *Implementation of ostis-platform sc-memory.*

SCin-code
:= [Semantic Code interior]
:= [Language for describing the representation of the SC-code inside ostis-platform sc-memory]
:= [Metalanguage for describing the representation of sc-constructions in ostis-platform sc-memory]
⇒ *frequently used non-primary external identifier of the sc-element**:
[scin-text]
∈ *common noun*
∈ *abstract language*
∈ *metalanguage*
∈ *sc-language*
⊂ *SC-code*
⊃ *sc-memory*

should be distinguished*
∋ { • *SC-code*
:= [Universal language of internal semantic representation of knowledge in memory of ostis-systems]
• *SCin-code*
:= [Metalanguage for describing the representation of the SC-code in ostis- platform sc-memory]
⊂ *SC-code*
}

Software interface of Implementation of ostis platform sc-memory
⇐ *software interface*:*
Implementation of ostis-platform sc-memory
∈ *software interface*
∈ *reusable ostis-systems component stored as source files*
∈ *atomic reusable ostis-systems component*
∈ *dependent reusable ostis-systems component*
⇒ *component dependencies*:*
{• GLib library of methods and data structures
• C++ Standard Library of methods and data structures
}
⇒ *method representation language used*:*
• C
• C++
⊃ *Software interface for information-forming methods of Implementation of ostis-platform sc-memory*
:= [information-forming methods of Implementation of ostis-platform sc-memory]
:= [subsystem that is part of the implementation of ostis-platform sc-memory, which allows creating, modifying, and deleting constructions of sc-memory]
⇐ *software interface*:*
Implementation of the information-generating subsystem of Implementation of ostis-platform sc-memory
⊂ *Implementation of ostis-platform sc-memory*
⊃ *Software interface for information-forming methods of Implementation of ostis-platform sc-memory*
:= [information retrieval methods of Implementation of ostis-platform sc-memory]
:= [subsystem that is part of Implementation of ostis-platform sc-memory that allows finding constructions in sc-memory]
⇐ *software interface*:*
Implementation of the information retrieval subsystem of Implementation of ostis-platform sc-memory
⊂ *Implementation of ostis-platform sc-memory*

Implementation of ostis-platform file memory
∈ *file memory implementation based on the prefix tree*
⇐ *software model*:*
ostis-platform file memory
∈ *reusable ostis-systems component stored as source files*
∈ *atomic reusable ostis-systems component*
∈ *dependent reusable ostis-systems component*
⇒ *component dependencies*:*
{• *GLib library of methods and data structures*
}
⇒ *method representation language*:*
• C
⇒ *internal language*:*
• SCfin code

Implementation of ostis-platform file memory
:= [Semantic Code file interior]

:= [Language for describing the representation of information constructions that do not belong to the SC-code inside the ostis-platform file memory]

:= [Metalanguage for describing the representation of information constructions that do not belong to the SC-code inside the ostis-platform file memory]

⇒ *frequently used sc-identifier*:*:*
[sc.fin-text]
∈ *common noun*

∈ *abstract language*

∈ *metalanguage*

∈ *sc-language*

⊂ *SC-code*

⊃ *ostis-platform file memory*

should be distinguished*

∋ { • *SC-code*
:= [Universal language of internal semantic representation of knowledge in memory of ostis-systems]

• *SCfin-code*
:= [Metalanguage for describing the representation of external information constructions that do not belong to the SC-code in ostisplatform file memory]
⊂ *SC-code*
}

should be distinguished*

∋ { • *SC-code*
:= [Metalanguage for describing the representation of the SC-code in ostis-platform sc-memory]
⊂ *SC-code*

• *SCfin-code*
:= [Metalanguage for describing the representation of external information constructions that do not belong to the SC-code in ostisplatform file memory]
⊂ *SC-code*
}

Implementation of the subsystem of interaction with the external environment using languages of network interaction

⇒ *software system decomposition*:*
{• *Implementation of the subsystem of interaction with the external environment using languages of network interaction based on the JSON language*
}

Implementation of the network interaction subsystem with sc-memory based on JSON in the ostis-platform

:= [Subsystem for interacting with sc-memory based on the JSON format]

:= [Network software interface of Implementation of ostis-platform sc-memory]

:= [Our proposed option of implementing the mechanism for accessing the ostis-platform sc-memory in a distributed collective of ostis-systems]

∈ *reusable ostis-systems component stored as source files*

∈ *non-atomic reusable ostis-systems component*

∈ *dependent reusable ostis-systems component*

∈ *client-server system*

⇒ *method representation language used*:*

- C
- C++
- Python
- TypeScript
- C
- Java

⇒ *language used*:*

- SC-JSON-code

⇒ *software system decomposition*:*
{• *Implementation of the Server System based on Websocket and JSON, providing network access to memory of the ostis-platform*
{ }
= **{**• *Implementation of the client system in the Python programming language*
• *Implementation of the client system in the TypeScript programming language*
• *Implementation of the client system in the C programming language*
• *Implementation of the client system in the Java programming language*
}
}

SC-JSON-code

:= [Semantic JSON-code]

:= [Semantic JavaScript Object Notation code]

:= [Metalanguage for describing the representation of messages between subsystems of the ostisplatform]

⇒ *frequently used sc-id*:*
[sc-json-text]

:= [The language we propose for interaction in a distributed collective of ostis-systems]
∈ *common noun*

∈ *abstract language*
⊂ *SC-code*
⊂ *JSON*

Implementation of the Server System based on Websocket and JSON, providing network access to memory of the ostis-platform

:= [Implementation of a Websocket-based system that provides parallel-asynchronous multi-client access to sc-memory of the sc-model interpretation platform using the SC-JSON code]

:= [sc-json-server]

⇒ *frequently used sc-identifier*:*
[sc-server]

:= [sc-server]

∈ *reusable ostis-systems component stored as source files*

∈ *atomic reusable ostis-systems component*

∈ *dependent reusable ostis-systems component*

⇒ *method representation language used*:*
- C
- C++

⇒ *language used*:*
- SC-JSON-code

⇒ *component address*:*
`[https://github.com/ostis-ai/sc-machine/sctools/sc-server]`

⇒ *component dependencies*:*
{● *Library of software components for processing json texts*
● *Library of cross-platform software components for implementing server applications based on Websocket*
● *Software component for setting up software components of ostis-systems*
● *Implementation of sc-memory*
}

Implementation of the interpreter for sc-models of user interfaces

:= [Our proposed interpreter for interpreting scmodels of ostis-systems user interfaces]

∈ *reusable ostis-systems component stored as source files*

∈ *non-atomic reusable ostis-systems component*

∈ *dependent reusable ostis-systems component*

⇒ *method representation language used*:*
- *JavaScript*
- *TypeScript*
- *Python*
- *HTML*
- *CSS*

⇒ *component address*:*
url[https://github.com/ostis-ai/sc-web]

⇒ *component dependencies*:*
{● *Library of standard interface components in the JavaScript programming language*
● *Library for implementing server applications in the Python Tornado programming language*
● *Implementation of the client system in the TypeScript programming language*
● *Implementation of the client system in the Python programming language*
}

VI. PROSPECTS FOR DEVELOPING THE SOFTWARE PLATFORM OF OSTIS-SYSTEMS

Software implementation of the ostis-platform

⇒ *prospects for development*:*
- Despite the fact that the *Implementation of ostisplatform sc-memory* is functionally complete for the development of *semantically compatible interoperable ostis-systems* and is *multi-user*, i.e. it can execute actions of different users in parallel, significant restrictions are imposed on the *actions* of these users. First of all, these restrictions are connected not so much with the memory model underlying the implementation but with the model of asynchronous access to it. The implemented model of asynchronous memory access requires blocking access to a group of related *sc-elements* and not to a particular one of these *sc-elements*. For example, to create an *outgoing sc-arc* from a given *sc-element*, it is necessary to lock not only the cell in memory in which this sc-element is stored but also the initial *incoming* and *outgoing sc-connectors* from the list of incoming and outgoing sc-connectors of this sc-element, respectively. In the process of parallel operation of sc-memory, blunders can often occur: deadlocking of processes performing actions on the same sc-elements, resource races on the same sc-elements, etc. To eliminate these problems, a transition to a new model of asynchronous access to sc-memory is required or a transition to a new implementation of sc-memory without changing the existing programming interface for the implementation of sc-memory.
- The current *Software implementation of the ostis-platform* is *customized* and does not include the *Implementation of the SCP Language interpreter* (that is, when the *ostis-platform* is running, the SCP *Language interpreter* is not used), which hinders the development of *platform-independent ostis-systems*. This is in no way related to the complexity of developing