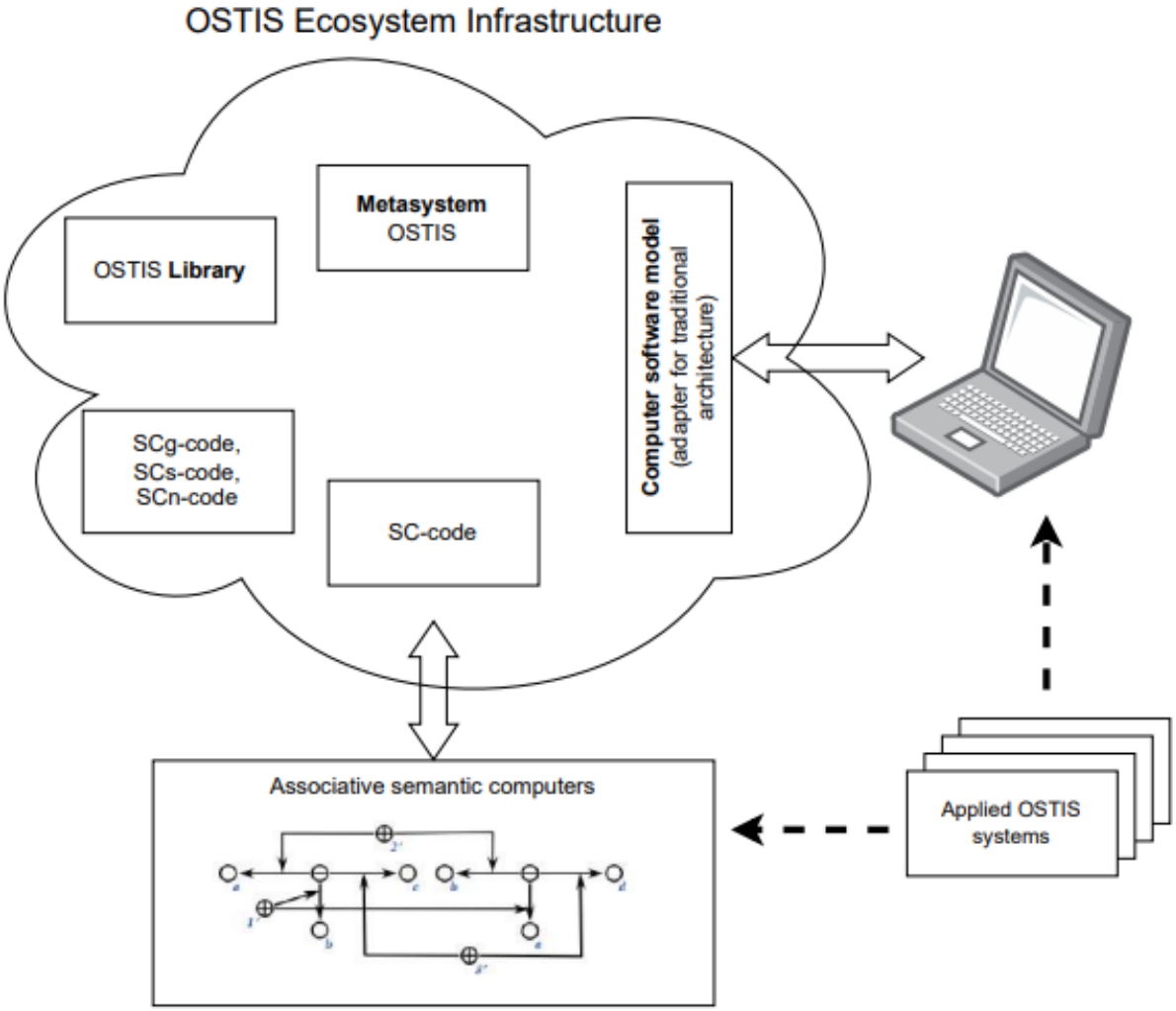

Figure 5: The OSTIS ecosystem and specialized computers place in it

[8] I. I. Levin, A. I. Dordopulo, I. A. Kalyaev, Yu. I. Doronchenko, M. K. Raskladkin Sovremennye i perspektivnye vysokoproizvoditel'nye vychislitel'nye sistemy s rekonfiguriruemoi arkhitekturoi [Modern and perspective high-performance computing systems with reconfigurable architecture]. *Vestnik YuUrGU [Bulletin of the South Ural State University]*, Chelyabinsk, Ser. Comp. Math. and Software Eng., 2015, Vol. 5, No. 3. pp. 24-39.

[9] V. Golenkov, N. Guliakina, D. Shunkevich, Otkrytaya tekhnologiya ontologicheskogo proektirovaniya, proizvodstva i ekspluatatsii semanticheski sovmestimykh gibridnykh intellektual'nykh komp'yuternykh sistem [Open technology of ontological design, production and operation of semantically compatible hybrid intelligent computer systems], Minsk, Bestprint, 2021, P. 690.

[10] D. Shunkevich Universal'naya model' interpretatsii logikosemanticheskikh modelei intellectual'nykh komp'yuternykh system novogo pokoleniya [Universal model of interpreting logicalsemantic models of intelligent computer systems of a new generation]. Otkrytye semanticheskie tekhnologii proektirovaniya intellectual'nykh system [Open semantic technologies for intelligent systems], 2022, pp. 285-296.

Открытая семантическая технология как фундамент нового поколения интеллектуальных систем

Татур М. М., Парамонов А. И

Рассматриваются вопросы развития нового поколения интеллектуальных систем. Приводятся примеры известных практических экспериментов по созданию специализированных платформ для достижения высокой реальной производительности вычислительной системы. Выдвигается гипотеза о необходимости создания принципиально новой архитектуры вычислительной машины для эффективной реализации проекта экосистемы ostis-платформ. Предлагается концепт инфраструктуры OSTIS экосистемы на базе специализированных компьютеров.

Received 13.03.2023

Design Principles, Structure, and Development Prospects of the Software Platform of ostis-systems

Nikita Zotov
Belarusian State University of Informatics and Radioelectronics
Minsk, Belarus
Email: nikita.zotov.belarus@gmail.com

***Abstract*—In the article, the principles of design and development of the basic *Software implementation of the ostis- platform* are described. The advantages of the ontological approach to documenting software systems of this type are shown. The structure, problems, and prospects of developing the *Software implementation of the ostis-platform* are described.**

***Keywords*—ontological design, automation tools for design and development of computer systems, knowledge base management system, universal interpreter, graph storage, ostis-platform**

I. Introduction

Modern *software computer systems* should operate not just with *data* but with *knowledge*. To understand the *meaning* of *knowledge*, it is necessary to represent this knowledge in an understandable form for any cybernetic *system*: as for any *human*, so for any *artificial system* [1]. At the same time, the form of representing this knowledge must be unified and independent of the platform on which this knowledge can be interpreted. Nevertheless, computer systems remain dependent on highly qualified specialists and experts in subject domains in which the automation of the design of these systems is carried out; therefore, the implementation of these systems requires significant resources [2]. One of the reasons for this is the need for a computer system to work on different platforms, each of which, in general, may have its own characteristics and limitations, which must be taken into account at the implementation stage.

The solution of these problems is **design and development of fundamentally new platforms**, which should provide:

- *unambiguity of interpretation* and representation of software system models provided by the unified knowledge representation language and platform design ontology used;
- *semantic compatibility* of software system models and their components [3], including interoperability between them [4];
- *platform independence* of software system models implemented and interpreted on it;
- *simplicity* and *extensibility* of their functionality;
- *functional completeness* for creating software system models due to the presence of a formal methodology for designing its implementation;
- segregation of duties between platform components.

II. Proposed approach to the design of systems for automating the design of software systems

The shortcomings of modern computer systems for design automation of other software systems, ways to solve them, as well as the approach to the solution described below were described earlier in the works [5] and [6].

Despite the vast variety of classical technologies used by mankind, there is no general solution that allows solving the problem in a complex. At the moment, the described problems can only be solved with the help of a general and universal solution — ***OSTIS Technology*** [7]. The *OSTIS Technology* is based on a unified variant of information encoding based on *semantic networks* with a basic set-theoretic interpretation, called *SC-code* [8]. The language of semantic representation of knowledge is based on two formalisms of discrete mathematics: *Set Theory* — defines the semantics of the language — and *Graph Theory* — defines the syntax of the language. Any types and models of knowledge can be described using *SC-code* [7].

One of the key principles of the *OSTIS Technology* [9] is providing ***platform independence*** of *ostis-systems* [10], i.e. strict separation of the *logical-semantic model of a cybernetic system (sc-model of a cybernetic system)* and the *platform for interpreting sc-models of a cybernetic system (ostis-platform)*. The advantages of such a strict separation are quite obvious:

- transfer of the *ostis-system* from one ostis-platform to another is carried out with minimum overhead

costs (in the ideal case – comes down to simply loading the *sc-model of a cybernetic system* onto the ostis-platform);

- components of *ostis-systems* become universal, that is, can be used in any ostis-systems where their use is appropriate;
- the development of the ostis-platform and the development of sc-models of systems can be carried out in parallel and independently of each other, in the general case by separate independent development teams according to their own rules and methods [11].

logical-semantic model of a cybernetic system

:= [formal model (formal description) for the functioning of a cybernetic system, consisting of (1) a formal model of information stored in the memory of a cybernetic system and (2) a formal model of a collective of agents that process this information.]

⊃ *sc-model of a cybernetic system*

:= [logical-semantic model of a cybernetic system represented in the SC-code]

:= [logical-semantic model of an ostis-system, which, in particular, can be a functionally equivalent model of any cybernetic system that is not an ostis-system]

ostis-system

⊂ *subject*

⇒ *generalized decomposition*:*

{• *sc-model of a cybernetic system*
• *ostis-platform*
}

sc-model of a cybernetic system

⇒ *generalized decomposition*:*

{• *sc-memory*
• *sc-model of the knowledge base*
• *sc-model of the problem solver*
• *sc-model of a cybernetic system interface*
}

ostis-platform

:= [platform for interpreting sc-models of computer systems]
:= [interpreter for sc-models of cybernetic systems]
:= [interpreter of unified logical-semantic models of computer systems]
:= [family of platforms for interpreting sc-models of computer systems]
:= [platform for implementing sc-models of computer systems]
:= ["empty"ostis-system]
:= [sc-machine implementation]
⊂ *platform-dependent reusable ostis-systems component [11]*

sc-memory

:= [abstract sc-memory]
:= [sc-storage]
:= [semantic memory storing SC-code constructions]
:= [storage of SC-code constructions]

In general, *sc-memory* implements the following functions:

- storage of *SC-code* constructions;
- storage of information constructions (files) external to *SC-code*. In general, file storage can be implemented differently from storage of *sc-constructions*;
- access (reading, creating, deleting) to *SC-code* constructions, implemented through the corresponding *software (hardware) interface*. Such an interface is essentially a *microprogramming language* that makes it possible to implement on its basis more complex procedures for processing stored constructions, the set of which essentially determines the list of commands of such a *microprogramming language*. The *sc-memory* itself is passive in this regard and simply executes commands initiated from outside by some subjects.

Despite all the advantages of *graph databases* in comparison with *relational databases* [12], [13], *new generation computer software systems*, due to their properties, [14] should operate not simply with *data*, but *knowledge*. To understand the meaning of knowledge, it is necessary to represent this knowledge in an understandable form for any kind of *cybernetic system* [14]. Speaking about the unification of the representation of all *types of knowledge*, it is considered important to use *graph databases* not just as a means for storing *structured data*, but for storing *semantically coherent* and *related* knowledge among themselves. Therefore, *sc-memory* is based on a graph representation of data and knowledge.

III. Principles underlying the Software platform of ostis-systems

The specification of such a complex program object as the *ostis-platform* must be represented in some formal *knowledge representation language*, in this case, in the *SC-code*, the texts of which it stores and processes. The language that should describe the *Software implementation of the ostis-platform* should be a *sublanguage** of the *SC-code*, i.e. it should inherit all the properties of the *Syntax and Denotational semantics of the SC-code* [15]. This representation of *software computer systems* specifications gives certainly strong advantages over other possible representations of specifications [16]:

- The language whose texts the system stores and processes and the language that specifies how the system represents the texts of the first language in its own memory are subsets of the same language. This simplifies not only the understanding of the developer who develops a complex *software computer system*, due to the fact that the form of representation of the language processed by this system and the