The conception, requirements and structure of the integrated computational environment[*]

V. P.Il'in

Institute of Computational Mathematics and Mathematical Geophysics, SBRAS,
Novosibirsk State University

Abstract

The general conception, main requirements and functional architecture of the integrated computational environment (ICE) for the high-performance mathematical modeling of a wide class of the multi-physics processes and phenomena on the modern and future postpetaflops supercomputers are considered. The new generation problems to be solved are described by the multi-dimensional direct and inverse statements for the systems of nonlinear differential and/or integral equations, as well as by variational and discrete inequalities. The objective of the ICE is to support all the main technological stages of large-scale computational experiments and to provide a permanent and extendable mathematical innovation structure for wide groups of the users from various fields , based on the advanced software and on integration of the external products. The technical requirements and architecture solutions of the project proposed are discussed.

Keywords: mathematical modeling, integrated computational environment, high performance, interdisciplinary direct and inverse problems, numerical methods, program technologies.

1 Introduction

In the epoch of fantastic growth of the postpetaflops supercomputers , the mathematical modeling of a wide class of multi-physics processes and phenomena has become the third way of the knowledge mining and deep learning, together with theoretical and natural experimental investigations. The huge computational and big data resources, the achievements of theoretical, applied and numerical mathematics, as well as the success in artificial intelligence and information technologies are a challenging chance for innovations in industry, nature, economy, social and human applications. The extremal machine experiments acquire a great importance both for basic research and practical issues.

As was pointed out in [1], in the near future the main problem for the computer science community will consist in the development of large volumes of the new generation software for upcoming supercomputers with millions and hundred millions of nodes, cores, special accelerators, and other units. The great opportunities for simulation are opened based on Data Centers, Grid Technologies and Cloud Computing. In such a situation the unique possibility to overcome the actual world crisis in programming is based on creating a modern software paradigm and on organizing a wide cooperation of various development groups, which should unify the academic expertise, software manufacturing and potential users. It is important to understand that an advanced algorithm can be proposed, investigated and tested efficiently by the operating mathematicians, but the robust implementation and code optimization for the method should be done by a professional programmer under the production requirements. And creative interaction between such specialists is the best way to a fast scientific innovation to industry.

[*]The work is supported by the Russian Scientific Foundation, grant N 14-11-00485 P., and by Russian Foundation of Basic Reseach, grant N 16-29-15122 ofi-m

Historically, the scientific simulation software has been evolved in the three main frameworks, that are either free accessible or commercial: applied program packages (ANSYS [2] or FEniCS [3], for example) , algorithm libraries (see NETLIB [4] and MKL INTEL [5]) and special tools for particular operations (grid generation– NETGEN [6], visualization– PARAVIEW [7], etc.) . The other trend in the past decades was based on creating an integrated computational environment (ICE) , from the system standpoint, to support the general issues of mathematical modeling. We can mention such projects as OpenFOAM [8], DUNE (Distibuted Unified Numerical Environment [9]), MATLAB [10], INMOST [11], and Basic System of Modeling (BSM, [12]). Such a kind of software products is considered as instrumental media for automatic construction of the new computational methods and technologies which present the background for a flexible creation of the configurations on the principle of popular intellectual constructor LEGO, for particalur applications. The objective of such intellectual system is to provide such alternative properties as the high efficiency, performance, robustness and the universality of the resulting codes.

In general, the statements of mathematical modeling problems, interdisciplinary direct and inverse ones, are described by the nonlinear systems of differential, integral, and/or discrete equations, or by equivalent, in a sense , variational relations in the corresponding functional spaces. In the applications with real data, the unknown solutions are defined in the computation domains with complicated geometries of multi-scale boundaries and contrast material properties in different domains.

The numerical experiments in various applications are based on a large number of algorithms and software technologies, but the fact is, an enormous set of numerical procedures can be classified and divided into separate stages: geometrical and functional modeling that is responsible for the automatic construction of the models for tasks to be solved, generation of adapted grids, approximation of the original continuous problems by the corresponding discrete relations, solving the obtained algebraic tasks, using methods of optimization for solving the inverse problems, post-processing and visualization of the results obtained, control analysis and decision making, etc. Each of these steps has a rich extendable intellectual functionality and is presented by the library of algorithms. Such subsystems act reciprocally via coordinated data structures, which also provides the interaction with external software products.

These components form the mathematical kernel that can be efficiently used for the automatic construction of algorithms in the framework of applied program configurations for particular practical fields, which are oriented to a wide class of the end users involved in various problems. Let us make one more remark. The numerical problems can be considered based on a hierarchical set of models: coarse, middle and fine, for example. And computer experiments should be made succesively, with increasing the accuracy of approximations, to provide the adequacy of the obtained simulation results. Correspondingly, the computational tasks to be solved can be classified in the three groups: small, middle, and large problems. The conventional division can be chosen as follows: the run times of the first, the second and the third groups of tasks present similar values (seconds, minutes, hours, etc.) on the gigaflops, teraflops and petaflops computers, respectively. Of coarse, the strategies and tactics of the parallelization are different in these situations, and the main topic of our interest is solving the large (and very large) problems.

The considered Integrated Computational Environment (ICE) should provide a high productivity of applied programmers from different groups of developers and conceptually represent a community project. In general, the resulting codes present an open source, but the implemented algorithms and technologies can be used for creating the confidential applications or for the special computing services.

The conception of a global environment for the computer simulation presents a tremendous software project. The described modeling problems can be examined in various coordinates. From the productive branches point of view, the processes and phenomena to be simulated, can be referred to energetics, machinery, biology, geophysics, chemistry, ecology, etc. From another standpoint, the same problems can be considered by means of the applied statements from hydro-gasdynamics, elasticity, electromagnetism, thermo-mass-transfer, and/or in multi-physics formulations. The favorable circumstances are that all these multi-variant particular cases can be presented by a finite set of abstract mathematical relations. So, the mathematical modeling involves in a powerful chain various branches of the fundamental sciences and technologies: theoretical, applied, and numerical mathematics, informatics and programming, intensive data computing, artificial intelligence, and different application fields. But the mathematical modeling is impossible without high-technological software, and the mission of the ICE concludes in unification the people of new mass professionals on the general industrial platform.

The contribution of this paper can be presented as follows: motivation of creating the Integrated Computational Environment (ICE) for a high performance scientific software for supercomputer modeling; description of the architecture of the ICE functional kernel for the instrumental support of the main stages for solving a wide class of the interdisciplinary direct and inverse problems; technical requirements for constructing the ICE and technological principles of implementing big programs; expected results of realizing the ICE for valuable increasing productivity of the development and for the effective wide distribution of the created applied products.

The paper is organized as follows. Section 2 presents the formal statements of the problems to be solved. Section 3 includes a description of technological stages of mathematical modeling, the corresponding data structures, the general description of the numerical algorithms, the issues of the scalable parallelism, and the implementation features of the functional kernel for a proposed scientific software. In Section 4, the general technical requirements and some architecture solutions for the integrated computational environment are presented. In conclusion, we discuss several methodological aspects and the conception of the ICE, and , also, the future activity dealing with the proposed project in terms of the fundamental problems of the mathematical modeling.

2 Statements of problems

In principle, the general mathematical statement of the numerical simulation is described by the interdisciplinary, or multi-physics, direct or inverse problem. In the abstract form, a direct task can be presented by the initial boundary value problem (IBVP) for the operator equation:

$$L\vec{u} = \vec{f}(\vec{x},\ t),\ \vec{x} \in \bar{\Omega},\ 0 < t \leqslant T < \infty, \tag{1}$$

where the unknown solution $\vec{u}$ is the vector function which is defined in the space-time computation domain $(\vec{x},\ t) \in \bar{\Omega} \times [0,\ T]$ and satisfies the boundary and initial conditions

$$l\vec{u} = \vec{g}(\vec{x},\ t),\ \vec{x} \in \Gamma = \Gamma_D \bigcup \Gamma_N,\ \vec{u}(\vec{x},\ 0) = \vec{u}^0(\vec{x}). \tag{2}$$

Here the operator l is responsible for the conditions on the boundary Γ. For example, L in (1) can be presented by the differential operator

$$L = A\frac{\partial}{\partial t} + \nabla B \nabla + C\nabla + D,\ \bar{\Omega} = \Omega \bigcup \Gamma \tag{3}$$

with the matrix coefficients $A,\ B,\ C,\ D$ whose entries can depend on independent space-time variables $\vec{x}$,t and, in nonlinear cases, on components of the solution $\vec{u}(\vec{x},\ t)$ to be sought for. The examples of the boundary conditions on different parts of the boundaries $\Gamma_D,\ \Gamma_N$ can be described as follows:

$$u = g_D,\ x \in \Gamma_D;\ D_N u + A_N \nabla_n u = g_N,\ x \in \Gamma_N, \tag{4}$$

where D_N and A_N are , in general, some matrices . The computation domain Ω can be presented as a union of the subdomains Ω_j with the corresponding interior and external boundaries Γ_i^j, Γ_j^e:

$$\bar{\Omega} = \bigcup \bar{\Omega}_j,\ \Gamma = \Gamma^e \bigcup \Gamma^i,\ \Gamma^i = \bigcup \Gamma^i_{j,\ k} = \bigcup (\bar{\Omega}_j \bigcap \bar{\Omega}_k). \tag{5}$$

It is important to remark that the coefficients of original equations can have sufficiently contrast values in different subdomains. Moreover, different equations can be solved in different subdomains. It is usually supposed that the input data in the direct problem (1)-(5) provide the existence and uniqueness of the solutions to be sought for some functional spaces. However sometimes we need to make a computational experiment for a given problem even without theoretical knowledge about its properties. The descriptions of various mathematical statements for differential and integral equations, as well as the review of the modern literature, can be found in the recent books [13]-[14].

In real cases the ultimate goal of the research consists in solving not direct but inverse problems, which means, for example, the identification of the model parameters, the optimization of some processes, etc. The universal optimization approach to solving the inverse problems is formulated as minimization of the objective functional

$$\Phi_0(\vec{u}(\vec{x},\ t,\ \vec{p}_{opt})) = \underset{\vec{p}}{min} \Phi_0(\vec{u}(\vec{x},\ t,\ \vec{p})), \tag{6}$$

which depends on the solution $\vec{u}$ and on some vector parameter $\vec{p}$ which is included in the input data of the direct problem. The constrained optimization is carried out under the linear conditions

$$p_k^{min} \leqslant p_k \leqslant p_k^{max},\ k = 1, ..., m_1, \tag{7}$$

and/or under the functional inequalities

$$\Phi_l \vec{u}(\vec{x},\ t,\ \vec{p})) \leqslant \delta_l,\ l = 1, ..., m_2. \tag{8}$$

Formally, the direct problem can be considered as the state equation and can be written down as follows:

$$L\vec{u}(\vec{p}) = \vec{f},\ \vec{p} = \{p_k\} \in \mathcal{R}^m,\ m = m_1 + m_2. \tag{9}$$

There are two main kinds of the optimization problems. The first one consists in the local minimization. This means that we look for a single minimum of the objective function in the vicinity of the initial guess $\vec{p}^0 = (p_1^0,\ ...,\ p_m^0)$. The second problem is more complicated and presents the global minimization, i. e. the search for all extremal points of $\Phi_0(\vec{p})$.

The numerical solution and high-performance implementation with scalable parallelism on the modern multi-processor computational systems (MPS) of the above mathematical problems present a tremendous set of the complicated computational schemes, (see the discussions in [15]-[16], for example) . The efficient scalability of the parallel algorithms is attained in a weak or in a strong sense. The first point of view means that the computing run time is approximately the same if the degree of freedom (d.o.f.) of the mathematical problem to be solved and the number of computational hardware units simultaneously increase . The second case corresponds to the situation, when the run time for a big problem with a fixed d.o.f. decreases in proportion to enlarging the volume of computational equipment. Different strategies and tactics of parallelization are attained by the mapping of algorithmic structures onto computer architectures. Here, the quantitative characteristics are estimated by the speedup S_p and the parallelization efficiency E_p:

$$S_p = T_1/T_p,\ E_p = S_p/P,\ T_p = T_p^a + T_p^c, \tag{10}$$

where T_p is the computer time needed for a given task (or the algorithm) on p processors. The description carefully analyzes a real model of computations at the heterogeneous cluster systems with distributed and hierarchical shared memory presenting a sufficiently complicated problem and will not be the topic of our consideration. We just mention that the conventional parallel technologies are based on creating the MPI (Message Passing Interface) processes on the multi-thread computations, on the vectorization of the operations by AVX special tools, as well as on the intensive computing at the fast graphical accelerators (GPGPU or Intel Phi, for example). It is important to consider the coefficient

$$Q_p = N_p^a/(N_p^a + N_p^c), \tag{11}$$

where N_p^a and N_p^c are the number of arithmetic operations and the volume of communications, respectively, because the interprocessor data transfers not only decelerate the computational process, but are essentially energy consuming. So, the problem of incresing the coefficient Q_p in (11) is an unexpected mathematical consequence of the engineer requirements.

3 Technological stages of mathematical modeling

In total, the large-scale computational experiments can be divided into the following main technological stages. In the framework of the ICE, all computational steps are implemented by the corresponding autonomous subsystems which are connected with each other via specified data structures. In general, a set of such subsystems forms the functional kernel of the integrated operating environment.

The first stage of a computing scenario consists in the geometrical and functional modeling. From the mathematical and technological standpoints, this means the automatic construction and modifications of the original problem. On the one hand, we need to describe the computation domain which includes different types of three dimensional (3-D), 2-D, 1-D and 0-D geometrical objects: the computation domain $\bar{\Omega} = \Omega \bigcup \Gamma$, the subdomains $\bar{\Omega}_j = \Omega_j \bigcup \Gamma_j$, the surface boundary segments (the faces Γ_j), the edges E_p (intersections of the surfaces)and the vertices, or points, P_q. In the dynamic and shape optimization problems , various operations can be defined on these objects: shifts, rotations, scaling, as well as topological and theoretical-set transformations. In the recent decades, many modern mathematical approaches have been developed : analytical metrics, differential calculus, isogeometric analysis (see [17], [18], for example).

The second part of the mathematical model includes the description of functional objects, i. e. the equations to be solved, their coefficients, initial values, boundary conditions, objective functionals, required accuracy, and so on. Of course, these data should be connected with geometric information. The end results of the first stage of the computer simulation consist in the geometric and functional data structures GDS and FDS, respectively, which map the whole input information onto a set of integer and real arrays. We will call the considered subsystem of the ICE as VORONOI.

It is important to remark that there is a large world market of the computer-aided design products (CAD, CAE, CAM, PLM) , which include the huge intellectual solutions in the geometric and visualization problems. The modern trend consists in the convergence, or integration, of CAD-systems and scientific codes for the mathematical modeling. Of course, the subsystem VORONOI should include convertors from the GDS and the FDS to the conventional information formats of the external computer design products, as well as to the popular visualization tools (PARAVIEW, for example).

Based on geometric and functional data structures we can realize the grid generation, which presents an important and resources-consuming stage of the numerical solution for the multi-dimensional problems. In the world software market, there are many available (free of charge) and expensive commercial codes, but the problem of constructing optimal or even "good" 3-D grids in the complicated computation domain is far enough from its final solution. Moreover, it is not easy to define the concept of an optimal or a good mesh, and there are many different quantitative characteristics of the grid quality.

Formally, the grid data structures are similar to the GDS and include the following objects: the grid computaion domain $\bar{\Omega}^h = \Omega^h \bigcup \Gamma^h$ with the boundary Γ^h, the grid subdomains $\bar{\Omega}^h_k = \Omega^h_k \bigcup \Gamma^h_k$ with the corresponding faces Γ^h_k, the grid edges E^h_p and the nodes P^h_q. The conventional approach to the discretization of the computation domain consists in constructing an adaptive grid. This means that the vertices P_q, the edges E_p and the surfaces Γ_k of the computation domain Ω should coincide with the corresponding grid nodes P^h_q, the grid edges E^h_p and the faces Γ^h_k. All these objects , in contrast to micro-objects, we call macro-objects: finite elements, or volumes, $\bar{T}^h_r = T^h_r \bigcup F^h_r$, with the element faces F^h_r, the mesh ribs R^h_s (intersection of the neighbouring faces F^h_r), and the mesh-points Q^h_l.

There are many kinds of the grids with different types of finite elements with various distributions of the meshsteps h, local refinement and multi-grid approaches included. Also, there are a lot of algorithms for the mesh generation, which are based on the frontal principles, on comformal or quasi-conformal transformations, on the differential geometry and various metrics. Here we consider the quasi-structured grids. This means that the grid structures can be different in different grid subdomains. For example, a grid can be non-structured, which means that the neighbouring mesh-points for every node can be defined by the enumeration only. In general, the quasi-structured grid consists of the grid subdomains which can have different types of the finite volumes, and grids in different subdomains can be constructured by the different algorithms.

In a sense, the grids considered present a two-level hyper-graph, at the macro- and micro-, or mesh, levels. In the technological sense, the final result of the grid generator should be the mesh data structure (MDS) with full mapping of the input geometric and functional data onto the micro (mesh) level. In particular, all inter-connections between grid objects should be strictly defined. Also, the affiliation of each finite volume T^h_j, grid face Γ^h_k into the corresponding subdomains Ω_k and the boundary surface segment Γ_p must be given.

The principles of constructing the library DELAUNAY are described in [19]. In fact, it presents the integrated instrumental media for the considered class of problems, based on original algorithms, as well as on re-using the external codes (there are popular free available mesh generators NETGEN, GMESH, TETGEN, for example) . In the world "grid developer community", there are several popular grid formats, and the subsystem DELAUNAY should include the corresponding data convertors with the MDS. At this stage one of important operations includes the decomposition of the grid computation domain into grid subdomains, when the number of mesh points is too large, from 10^8 to 10^{10}, for example. In this case, it is natural to implement such procedures in parallel, and form the corresponding MDS for subdomains, distributed among different processors and MPI-processes.

There are many numerical approaches to construct the qualitive or optimal grids, but, in general, these mathematical questions are open yet. Also , we do not consider here resource-consuming problems of generating the dynamic meshes which are changed during the computational process.

The next stage of the mathematical modeling presents the approximation of the original IBVP, based on the MDS, the GDS and the FDS. In this case, the most popular approaches are finite difference, finite volume, finite element, and discontinuty Galerkin methods (FDM, FVM, FEM, and DGM, see [14], [20], [21] , for example) . The advanced theoretical and applied mathematical results have profound foundations and technologies for constructing and justification of high order accuracy numerical schemes for complicated IBVPs with real data. The implementation of such algorithms on the non-structured grids is not simple, and the tools for automatic construction of the scheme are very useful for such problems, (see [3] , for example) . It is important to remark that a very powerful approach here is based on the element technology with computing the local matrices and assembling the global matrix, which provide the "natural" parallelization and easy programming of the algorithms.

The concept of the integrated operating environment for the methods of approximation of the multi-dimensional IBVP is presented in the library CHEBYSHEV [22] based on original algorithms and re-using the external software. The end result of this subsystem consists in the algebraic data structure (ADS) which presents the original problem to be solved at the discrete level. To provide the necessary accuracy, the obtained systems of linear algebraic equations (SLAEs) should have very large dimensions (10^8 and more) and sparse matrices. To save such systems in the memory, the conventional compressed formats are used, Compressed Sparse Row (CSR), for example. Of course, for the large d.o.f., the distributed versions of the CSR are used, i.e. the matrix is divided into block rows, and each one is placed in the corresponding MPI-process.

The most resource-consuming stage of mathematical modeling is a numerical solution of large sparse SLAEs, because the volume of arithmetic operations grows nonlinearly, when the number of unknowns increases. Fortunately, the computational algebra is one of the most progressive parts of numerical mathematics, both in algorithmic and in technological senses, (see [23]-[26] and the literature, cited therein). In particular, there are many applied software packages and libraries with algebraic solvers which are free accessible. The main approaches to solve large sparse SLAEs are based on the preconditioned iterative methods in the Krylov subspaces. The scalable parallelism is provided in the framework of the two-level iterative domain decomposition methods (DDM) by means of hybrid programming with using MPI tools for the distributed memory of the heterogeneous cluster MPS, multi-thread computing of the shared memory of the multi-core CPUs, vectorization of operations by means of the AVX system, as well as fast computation on the graphic accelerators (GPGPU or Intel Phi).

The grid computation domain is decomposed into subdomains with parametrized overlapping and different interface conditions on the interior boundaries. Algebraically, the external iterative process presents the multi-preconditioned generalized minimal residual (GMRES) or a semi-conjugate residual (SCR) algorithm, based on the parallel block Schwartz-Jacobi method, coarse grid correction, deflation and/or augmentation procedures, and an advanced low-rank approximation of the original matrix. At each external iteration, solving the auxiliary SLAEs in subdomains is implemented synchronously by means of direct or iterative algorithms, with various preconditioning matrices.

The described parallel methods are realized in the framework of the library KRYLOV [25] which presents the integrated algebraic environment, based on the original algorithms and efficient re-using the external products. In particular, the robust matrix-vector operations and other algebraic tools from the library MKL Intel are applied in KRYLOV in a productive manner.

If the original continuous problem is nonlinear, then after its discretisation we will have the system of nonlinear algebraic equations (SNLAEs). In these cases, the quasi-linearization process is applied, based on Newtonian type of iterations, and at each of such steps the linear equations are solved.

The optimization methods for solving inverse problems are presented in the library KANTOROVICH. This broad class of algorithms includes solving the tasks of linear, integer and nonlinear programming, constrained or non-constrained, local or global minimization of functionals. In the recent decades, such a scientific direction has been developed dramatically fast. The advanced approaches include the conjugate equation approaches, modified Lagrangians and regularization, interior point methods, successive quadratic programming, trust region algorithms, kriging technologies, and surrogate optimization (see [27]-[28] and the literature therein) . The last mentioned approach corresponds to the situation, when the run time for computing one value of the objective function is too expensive and requires several hours or more. In such a case, the design, or planning, of numerical experiments is very important, as well as using special methods for approximation of the investigated functionals by means of the radial basic functions (RBF, see [28] , for example).

So, in general, we have multi-step computational process, and on each stage it is necessary to repeat geometric and functional modeling, grid generation, approximation, and so on. When the numerical solution of the problem has been obtained, we have to understand and interpret the digital results which are presented usually by the values of the scalar or/and vector functions defined on the multi-dimensional non-structured grid or by the coefficients of the expansion of these solutions into the series of some basis functions. If we simulate some physical 3-D fields, for example, it is interesting to see the iso-surfaces, force lines, gradients, some extremal points and other characteristics, dynamic behavior included. Such postprocessing and vizualization approaches are very computation-consuming and may constitute the main run time of computer experiments. The usual way to overcome such issues consists in using the graphic accelerators.

We consider mainly the intensive computing stages of mathematical modeling, and such important questions as the general control of large-scale numerical experiments and decision making systems on the simulation results present the special topic for further research.

4 Technical requirements for integrated environment

The unification of the program implementations of the above -described mathematical stages of modeling presents a huge software complex, which consists of the functional kernel of the basic system of modeling [12] . From the system standpoint, it is a method-oriented set of tools for solving a wide class of mathematical problems. In order to provide the resulting success of the project in question, the BSM should be organized as an open source program product with a long life-cycle and professional maintenance, with active participation of different groups of developers and end users. In accord with such a conception, the integrated computational environment should satisfy the following evident requirements.

- The flexible extendability of the content of the models and of the problems to be solved in the framework of the ICE, as well as a manifold of applicable advanced numerical methods and program technologies without limitations on the d.o.f. and on the number of computer nodes, cores and other units. The matter of fact is that theoretical, applied and numerical mathematics, as well as computational and informational technologies are permanently fast developed, and the new generation software should be currently modernized in order to provide the scientific innovations for practical applications.

- Adaptation to the evolution of computer architectures and platforms. Component object technologies (COM, see [29] , for example) to provide the automatic concordance of the internal and external interfaces. Of course, such a feature of the applied software is obvious for a wide-spread distribution of the advanced computational technologies, and needs the automatic mapping of algorithm structures onto hardware equipment.

- Compatibility of the flexible and expandible data structures with the conventional formats to provide the efficient re-using the external products. Unification of the interfaces and possibilities for their convertation is a powerful device for the integration of intellectual software properties.

- High performance of the software developed, scalable parallelism based on the hybrid programming tools and code optimization on the heterogeneous multi-processor supercomputers with distributed and hierarchical shared memory on the cluster nodes and many-core CPUs (Central Processor Unit), respectively , as well as using vectorization of operations by means of the AVX system and graphic accelerators (GPGPU or Intel Phi, for example) .A special attention should be given to constructing the computational schemes with minimal data communications.

- The polyglot interaction and consistency of various program components, as well as opening the working contacts for different groups of developers. The similar multi-language requirements are necessary to create friendly interfaces for the end users of different professional applications. As it was pointed out in [30], the general problem consists in a software language engineering for creating domain-specific languages. Figuratively speaking, the road-map of artificial intelligence for the computer science community is to move from paleo-informatics to neo-informatics.

In order to satisfy such strict and diverse requirements, the integrated program environment should have a valuable system support. Such intelligent components constitute the infrastructure whose goal is to provide the maintenance, information security, collective exploitation and further development of the ICE. The corresponding instruments should include the following main procedures:

- automatic verification and validation of the codes, as well as testing and comparative analysis of the efficiency of algorithms on representative sets of the model and real life examples;
- construction of the particular program configurations for a specific application by assembling the functional modules from the ICE, multi-version control included;
- preparing the documentation on the project components (manuals, user guides, special descriptions, etc.) , different examples for demonstrations included;
- data structures control and transformations, manufacturing the problem-oriented languages and compilers, or convertors, providing the friendly interfaces for developers and users;in particular, such a problem includes the analytical transformations of the analytical expressions, as it is done in the popular systems MAPLE and MATHEMATICS; here we should recall the formula by Niclaus Wirt: "Program=algorithm + data structure".
- a permanently extended knowledge database, which includes the information about mathematical statements, supported by the ICE, employing computational methods and technologies, as well as industrial or natural problems and other applications , which can be solved; ontology principles and cognitive instruments can help the users to recognize the statement and pecularities of the problem, to choose the corresponding algorithm and/or available code for computational experiments.

5 Conclusion

In the recent years, the widespread discussions on the digital economy and on the post-industrial society require the understanding of the role of mathematical and computer sciences in the product and social life management. Mathematical modeling is becoming a significant device in the business and human evolution. And the high-performance numerical simulation is impossible without integrated program environment which should be a permanent background for the collective development of the advanced algorithmic approaches, for support, maintenance, and promotion of the unique software support to provide the wide practical innovation.

An important consequence of the global mathematical modeling consists in the appearance of new mass professions: developers, distributers and modellers. The latter present the new generation of the end users, from theoretical physicists to designers of building or aircraft industries, whose main working instrument becomes a power computer with intellectual interface.

Creating an integrated computational environment presents an unprecedent large project, based on the unification of the various proffesions. In a sense, this is the way from individual or small group program productions to the industrial conception of the unified community activity in the scientific software business. It is well known that the expenses of the supercomputer hardware and software

are comparable and big enough. So, the mathematical modeling is becoming the subject of the digital economy, and organizing solutions should take into acount this side of scientific innovations.

References

1. IESP:URL: www.exascale.org/iesp
2. ANSYS.URL: www.ansys.com
3. FEniCS.URL: http://fenicsproject.org
4. Netlib.URL: http://netlib.org
5. Intel R Mathematical Kernel Library.URL:http://software.intel.com/en-us/intel-mkl
6. Schoberl J.Netgen-An advancing front 2D/3D-mesh generator based on abstract rules.//Computing and Vizualization in Science. -Vol.1, N.1,41-52,(1997).
7. PARAVIEW: URL: www.paraview.org
8. OpenFOAM: URL:http://www.openfoam.com
9. DUNE.URL: http://www.dune-project.org
10. MATLAB: URL: https://www.mathworks.com/products/matlab.html
11. INMOST. A toolkit for distributed mathematical modelling.URL: www.inmost.org
12. Il'in V.P., Gladkih V.S. Basic system of modelling (BSM): the conception, architecture and methodology (in Russian) . Proceed.Internat. Confer. "Modern problems of mathematical modelling, image processing and parallel computing" (MPMMIPPC-2017), DSTU Publ., Rostov–Don, 151-158, (2017).
13. Brugnano L., Iavernano F. Line Integral Methods for Conservative Problems. CRC Press., Taylor Francis Group, N.Y.,(2015).
14. Il'in V.P. Mathematical modelling, Part I: Continuous and discrete models (in Russian). Novosibirsk, SBRAS Publ.,(2017).
15. Il'in V.P. Fundamental Issues of Mathematical Modeling.–Herald of the Russian Academy of Sciences, V. 86, N 2, 118-126, (2016).
16. Il'in V.P. On the parallel strategies in mathematical modeling. Lecture notes, CCIS, Springer, vol.753, 69-81, (2017)
17. Delfour M., Zolesio J.-P. Shape and Geometries. Metrics, Analysis, Differential Calculus,and Optimization. Philadelphia: SIAM Publ., (2011).
18. Cottrell J., Hughes T.,Bazilevs Y. Isogeometric Analysis. Towards Integration of CAD and FEA. Singapore: John Wiley Sons, (2009).
19. Il'in V.P. DELAUNAY: technological environment for grid generation (in Russian) , Sib.J.Ind.Math., vol.16,83-97,(2013).
20. Brenner S.C., Scott L.R. The Mathematical Theory of Finite Element Methods. New York: Springer-Verlag, (2008).
21. Riviere B. Discontinuous Galerkin Methods for Solving Elliptic and Parabolic Equations. Theory and Implementation. SIAM, Philadelphia, (2008).
22. Butyugin D.S.,Il'in V.P. CHEBYSHEV: the principles of automatical constructions of algorithms for grid approximations of initial-boundary value problems (in Russian). Proceed. Internat. Confer.PCT-2014, Chelyabinsk, SUSU Publ. , 42-50, (2014).
23. Saad Y. Iterative Methods for Sparse Linear Systems.-N.Y.: PWS Publ.,(2002).
24. Dolean V., Jolivet P., Nataf F. An Introduction to Domain Decomposition Methods:algorithms, theory and parallel implementation. Philadelphia: SIAM, (2015).
25. Butyugin D.S., Gurieva Y.L., Il'in V.P., Perevozkin D.V., Petukhov A.V. Functionality and algebraic solvers technologies in Krylov library (in Russian).-Vestnik YuUrGU. Series "Computational mathematics and informatics", v.2,N 3, 92-105,(2013).
26. Il'in V.P. Problems of parallel solution of large systems of linear algebraic equations –J. of Mathem.

Sci., v. 216, N 6, 795-804, (2016).
27. Il'in V.P. On the numerical solution of the direct and inverse electromagnetic problems in geoprospecting (in Russian), Sib.J.,Num.Math.,vol.6,N 4, 381-394, (2003).
28. Forrester A., Sobester A., Keane A. Engineering Design via Surrogate Modeling. A Practical Guide, JohnWiley and Sons, New York, (2008).
29. Maloney J. Distributed COM Application Development Using Visual C++. N.Y.: Prentice Hall,(1999).
30. Kleppe A. Software Language Enginneering: Creating Domain-Specific Language Using Metamodels. New York: Addison-Wesley, (2008).