Modeling training content for software engineers in parallel computing

Modeling training content for software engineers in parallel computing

Y O Sitsylitsyn[1], V V Osadchyi[2,4], V S Kruglyk[1] and O H Kuzminska[3,4]

[1] Bogdan Khmelnitsky Melitopol State Pedagogical University, 59 Naukovoho mistechka Str., Zaporizhzhia, 69000, Ukraine
[2] Borys Grinchenko Kyiv University, 18/2 Bulvarno-Kudriavska Str., Kyiv, 04053, Ukraine
[3] National University of Life and Environmental Sciences of Ukraine, 15 Heroyiv Oborony Str., Kyiv, 03041, Ukraine
[4] Academy of Cognitive and Natural Sciences, 54 Gagarin Ave., Kryvyi Rih, 50086, Ukraine

E-mail: yuriy@mdpu.org.ua, v.osadchyi@kubg.edu.ua, krugvs@gmail.com, o.kuzminska@nubip.edu.ua

Abstract. This study proposes a robust framework for the training of software engineers specializing in parallel computing. We first curated essential content for parallel computing education based on international standards and evolving recommendations from Computing Curricula. We then systematically structured the content and designed a well-defined learning pathway for aspiring software engineers. Concurrently, we conducted a comprehensive assessment of the current state of training for parallel computing in Ukrainian higher education institutions. We analyzed bachelor's programs in Information Technologies and scrutinized individual course syllabi to identify valuable insights. By merging our findings with the review of educational programs, we formulated a comprehensive model for training in parallel computing. We also examined the pivotal role of the course "Parallel and Distributed Computing" in the developed curriculum and identified essential tools and methodologies for developing parallel and distributed programs. Our research contributes to the advancement of parallel computing education and provides a valuable reference point for curriculum designers and educators.

1. Introduction

One of the methods of increasing the competitiveness of IT enterprises is to reduce the cost of software development and increase its efficiency. It should be noted that the development of programs that use only one processor core, relying on classic sequential algorithms, does not provide the necessary increase in productivity compared to those that use parallel multi-core solutions in their code.

The current level of development of supercomputer technologies based on parallel computing, the spread of multicore processors determines the relevance of studying parallel programming. Computational parallelism operates in various specific forms depending on the programming phase, the complexity of the parallel fragments, and the nature of the connections between them. Parallel programming includes all the features of classic serial programming.

In the process of studying parallel programming technology, a parallel style of thinking is formed, which implies the presence of abilities: to preliminary imaginary "parallelization" of the assigned task – its analysis with the aim of selecting subtasks that can be performed in parallel; to

"parallelization" of data flows – selection of data flows that will be exchanged between subtasks performed in parallel; keeping in memory the actions of all subtasks in a certain period of time to properly manage their joint work.

With a sufficiently high level of development of the parallel style of thinking, a software engineer can foresee specific problems that arise during the operation of parallel algorithms, which do not appear every time and significantly complicate the debugging of programs [1]. Only the achievement of a certain level of development of a parallel thinking style will allow a software engineer to effectively implement parallel computing. Achieving success in mastering parallel programming hinges on a fundamental shift in cognitive processes, necessitating the cultivation of a parallel thinking style. This transformation, in turn, will facilitate the development of training for software engineers specializing in parallel and distributed computing.

In the pursuit of this objective, it is imperative to establish a structured curriculum tailored to the needs of aspiring software engineers entering the realm of parallel computing. This curriculum will serve as the foundational cornerstone upon which to construct a comprehensive model for the training of software engineers specializing in parallel computing. This model will not only prescribe the optimal sequence for their coursework but also outline the pivotal role and seamless integration of the "Parallel and Distributed Computing" course within the broader structural and logical framework of the bachelor's program in computer sciences.

Vakaliuk [2], Seidametova et al [3], Osadchyi [4], Varava et al. [5], Striuk and Semerikov [6] carry out fundamental research in the field of training of software engineers in HEIs. Scientists also highlight certain aspects of the mentioned problems, in particular: the issue of the quality of training of programmers [7]; requirements for professional qualities of software engineers [8]; organization of software engineering education in universities worldwide [9, 10]. Diaz et al [11], Sitsylitsyn [12] and other scientists are engaged in the review and selection of parallel programming tools. Marowka [13], Wilkinson et al [14], Capel et al [15] are explored teaching parallel programming.

Giacaman and Sinnen have delved into the nuances inherent in the conventional pedagogy of programming within higher education institutions, with a specific focus on preparing software engineers [16]. Furthermore, Vasconcelos et al. have conducted an insightful analysis of the temporal aspects involved in incorporating the knowledge grid related to "Parallel and Distributed Computing" [17].

Objective of the article: This article aims to formulate a training content model for software engineers in parallel computing.

2. Theoretical foundation of the study

To formulate a curriculum model for the training of aspiring software engineers with a specialization in parallel computing, the initial step is to establish the foundational content necessary for their education in this domain. To achieve this, we will reference established international standards, particularly the guidelines pertaining to computer science education [18–21].

The issue of parallel computing is primarily related to one of the constituent parts of computing – Computer Science [19,20]. Aspects of parallelism were first mentioned in the edition of Computing Curricula from 2001, but both in 2001 [18] and in 2008 [19] parallelism was not separated into a separate field of knowledge – the issue of parallel computing was included as separate sections in several different parts of the computer science study recommendations. And only in 2013, the first edition of the recommendations for the study of computer sciences was developed, where a separate branch of knowledge "Parallel and Distributed Computing" was created [20]. In the 2020 recommendations, there were significant updates made to the content within the knowledge domain of "Parallel and Distributed Computing".

To establish the content for the training of software engineers specializing in parallel and distributed computing, it is prudent to draw upon the following primary sources:

1. Recommendations outlining the composition of the subject area of parallel and distributed computing (referred to as Table of Contents 1), prepared as part of a collaborative effort between ACM and IEEE-CS within their curriculum development project [21].
2. Recommendations pertaining to the structure of the subject area of parallel and distributed computing (referred to as Content 2), developed within a project supported by the National Science Foundation (NSF) in the United States and in conjunction with the IEEE Technical Committee on Parallel Processing (TCPP) [22].

The fundamental approach to delineating the content for training in parallel and distributed computing adheres to the following principles:

1. The collection of knowledge and skills essential for effective professional practice is determined by distinct knowledge domains, each representing integral facets of the corresponding field of expertise.
2. These knowledge domains are further subdivided into smaller units known as sections, which serve as discrete thematic modules within the field.
3. Each section, in turn, is comprised of a collection of topics, representing the lower tier within this hierarchical structure within the respective field of expertise. Each topic is accompanied by an indication of its mandatory or optional status, along with the recommended amount of study time necessary for its comprehension.

It's essential to clarify that the structure of domains, sections, and topics is indicative of the essential knowledge required for proficiency in the relevant field of expertise, rather than an exhaustive list of educational courses. The curricula and corresponding training modules are subsequently developed based on this foundational content.

In our endeavor to shape the curriculum for the training of software engineers specializing in parallel computing, we will draw upon the aforementioned documents, denoted as Content 1 [21] and Content 2 [22], to define the subject domain of parallel computing. Additionally, we will incorporate insights from the seminal work in parallel computing, "Structured Parallel Programming: Patterns for Efficient Computation" by McCool [23], which we will refer to as Content 3.

At the highest level of this foundational content, we identify five primary knowledge domains, collectively spanning the entire spectrum of parallel computing topics:

1. Mathematical foundations for parallel computing.
2. Parallel computing systems (fundamentals of computing).
3. Parallel programming technologies.
4. Parallel algorithms for problem solving.
5. Parallel computing for large-scale tasks and specialized domains.

We present the curriculum content as a structured list of thematic sections (table 1). The right-hand columns of the table indicate the presence (+) or absence (–) of relevant Content topics in alternative developments, i.e., Content 1 [21] and Content 2 [22], respectively. The mark "+/–" means that the topic is presented to a large extent, the mark "–/+" says that the topic is revealed partially.

Let's examine the elements of the training content in parallel computing as outlined in Content 1 (ACM 2020 Recommendations) [21], Content 2 (NSF / IEEE-TCPP Project Recommendations) [22] and Content 3 (McCool et al) [23].

In content 3 [23], scientists highlight the following methods and technologies for developing parallel programs:

Table 1. Summary of training in parallel computing.

No	Area of knowledge, section	Content1	Content 2
1	**Mathematical foundations for parallel computing**		
1.1	Graphs of program models	–	–
1.2	The concept of unlimited parallelis	–	–
1.3	Thin information structure of programs	–	–
1.4	Equivalent program transformations	–	–
1.5	Calculation models for computer systems	+	+
1.6	Mathematical models of parallel computing	+/–	+/–
2	**Parallel computing systems (fundamentals of computing)**		
2.1	Basics of machine computer	–	+
2.2	Basics of building computer system	–/+	+/–
2.3	Parallel computing systems	–/+	+/–
2.4	Multiprocessor computer systems	+/–	+/–
2.5	Multiprocessor computing systems with shared memory	–/+	+
2.6	Multiprocessor computing systems with distributed memory	+	+
2.7	Graphics processors	–/+	–
2.8	Computing systems of transpetaflop and ex-scale characteristics	–	–
2.9	Distributed computing systems	+	–/+
2.10	Challenges of supercomputers and data center	–	–
3	**Parallel programming technologies (fundamentals of software engineering)**		
3.1	General principles of parallel programming	–/+	–/+
3.2	Basics of parallel programming	–/+	+/–
3.3	Methods and technologies for developing parallel programs	–	+/–
3.4	Parallel problem-oriented libraries and software packages	–	–
3.5	Tools for parallel development of programs	–	–
3.6	Methods of increasing the efficiency of parallel programs	–	–
4	**Parallel algorithms for problem solving**		
4.1	General principles of parallel algorithm development	–/+	+/–
4.2	Educational algorithms of parallel programming	–	+
4.3	Parallel algorithms of matrix calculation	–	+/–
4.4	Parallel algorithms for sorting and searching data	–/+	+
4.5	Algorithms for parallel processing of graphs	–	+
4.6	Parallel algorithms for solving partial differential equations.	-	–
4.7	Parallel algorithms for solving optimization problems	–	–
4.8	Parallel Monte Carlo algorithms	–	–
4.9	Parallel algorithms for other classes of computationally intensive problems	–	–

1. Traditional programming languages and compilers that can parallelize. Vectorization of programs.
2. Software libraries for developing parallel programs: Intel TBB (Thread Building Blocks), Linda, Microsoft TPL (Task Parallel Library), MPI, PVM, Shmem.
3. Superlanguage tools for organizing parallelism: DVM, Cray Fortran, OpenMP, Cilk, HPF.

4. Parallel extensions of traditional programming languages: CAF, UPC.
5. Parallel programming languages: Occam, SISAL, NORMA.
6. Parallel programming languages for distributed shared memory systems in the PGAS model: X10, Chapel.
7. Parallel programming languages for graphic processors: CUDA, OpenCL.
8. Functional parallel programming languages: Parlog, Parallel Haskell, Erlang, T-System.
9. Tools and technologies to support metacomputing and distributed computing: Globus, UNICORE, gLite, X-Com, BOINC, MapReduce.
10. Programming technologies of FPGA computers.
11. Automation of parallelization and optimization of programs.
12. Elements of circuit engineering, languages for describing electronic circuits, VHDL.

List of sections of the field of knowledge "Parallel and Distributed Computing" based on research materials [23] and [21] are given in the table 2.

The performed comparison (table 1, table 2) shows that the recommendations of 2020 [21] almost completely overlap with options from the other two considered approaches. It can be noted that the sections related to distributed systems, cloud computing and formal models and semantics are more thoroughly disclosed in the recommendations [21].

Table 2. Sections of the field of knowledge "Parallel and Distributed Computing" (Content 1).

Section	Area of knowledge	Content 3	Content 1
A	parallelism fundamentals	+	+
B	parallel decomposition	+	+
C	communication and coordination	+	+
D	parallel algorithms, analysis, and programming	+	+
E	parallel architectures	+	+
F	parallel performance	+	+
G	distributed systems	–/+	–/+
H	cloud computing	–/+	–/+
I	formal models and semantics	+/–	–/+

The delineation of the subject domain of parallel and distributed computing was also undertaken as part of a project funded by the National Science Foundation (NSF) in the United States and facilitated by the IEEE Technical Committee on Parallel Processing (TCPP) [22]. The initial version of these recommendations was formulated in 2010, with the most recent working edition being published in 2020.

According to these recommendations, four areas of knowledge have been identified in the field of parallel and distributed computing:

1. Architecture.
2. Programming.
3. Algorithms.
4. Additional sections.

As you can see, the selected areas of knowledge largely repeat the content structure of training proposed in Contents 3. Along with this, two areas of knowledge ("Mathematical foundations" and "Parallel computing for large-scale tasks and specialized domains") are available in Contents 3, absent from the NSF / IEEE-TCPP recommendations. On the other hand, the area "Additional sections" of the considered recommendations as such is absent and distributed in other areas of knowledge of Content 3.

In the recommendations of the NSF / IEEE-TCPP project, structural elements of the second level – sections – are not explicitly highlighted. In fact, the composition of the areas of knowledge is determined immediately in the topics, which complicates the use of such a definition of the subject area. In some cases, the proposed topics are combined into groups.

Table 3. Sections of the field of knowledge "Parallel and Distributed Computing" (Content 2).

No	Area of knowledge, section	Content3	Content 2
1	**Architecture**		
1.1	Data and control parallelism	+	–/+
1.2	Shared and distributed memory	+	–/+
1.3	Memory hierarchy	+	–/+
1.4	Performance indicators	+	–/+
1.5	Floating point representation	+	–
2	**Programming**		
2.1	Parallel software paradigms	+	–/+
2.2	Problems of semantics and correctnes	+	+
2.3	Performance issues	+/–	–/+
3	**Algorithms**		
3.1	Models and complexity estimates	+	+/–
3.2	Algorithmic paradigms	+	–/+
3.3	Algorithmic problem	+	–/+
4	**Additional topics (no topic groups)**	–/+	–/+

By analyzing the key insights from the seminal works cited above, we can develop a comprehensive curriculum for software engineers who specialize in parallel computing, including a recommended sequence for their coursework (figure 1).

Based on our analysis of the training curriculum for software engineers in parallel computing, we identified the following goals for the course "Parallel and Distributed Computing":

1. Study of parallel computing systems, their programming methods, principles, and phases of software development using MPI and OpenMP technologies; formation of abilities to apply parallel programming technologies, as well as the main functions of these libraries.
2. Formation of parallel algorithm compilation skills for solving professional tasks, in particular: dividing a task into subtasks, identifying, and analyzing information dependencies between subtasks, information interaction between subtasks within MPI and OpenMP technologies.
3. Formation of a method of algorithmic actions, in which the well-thought-out process of compiling an algorithm naturally fits into the stages of development of a parallel algorithm, that is, the formation of a parallel style of thinking.

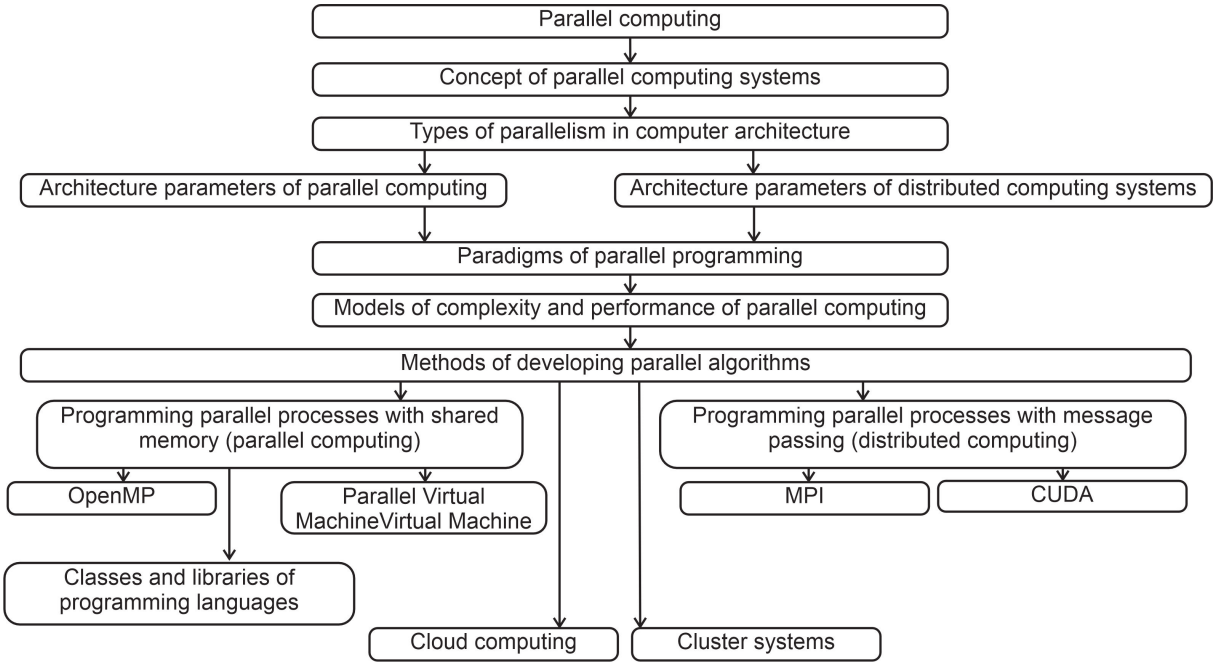

Figure 1. The trajectory of learning parallel computing.

4. Gaining experience in programming parallel computing systems, applying the principles of parallel algorithms, organizing information interaction between individual subtasks, applying parallelism to planning one's activities.

In addition to examining the curriculum for prospective software engineers in parallel computing, a thorough evaluation was conducted to assess the current state of such training in Ukrainian HEIs.

The instruction of the "Parallel and Distributed Computing" course is primarily aimed at instilling specific professional competencies in students pursuing higher education, as defined by the standards set forth in Ukraine for bachelor's programs in Computer Science [24]. Our analysis encompassed various HEIs in Ukraine, specifically within the domain of Information Technologies, and included a detailed review of individual course outlines [25–28].

Due to variations in course nomenclature across different institutions, our evaluation considered course outlines under diverse names such as "Parallel and Distributed Computing", "Technologies of Distributed Systems and Parallel Computing", and "Parallel Programming".

Our exhaustive review leads us to a compelling conclusion: parallel computing holds an indispensable place within the mandatory curriculum for software engineers pursuing bachelor's degrees in Information Technologies across HEIs in Ukraine.

By systematically applying the stages of system modeling to the course "Parallel and Distributed Computing", we have developed a comprehensive model, as shown in figure 2.

The model is structured as a cylinder with layers that represent different levels of semantic value. The foundational layer provides supporting knowledge, including essential concepts and their interconnections, which are essential for understanding the theoretical content.

The subsequent layers of the "concept cylinder" are arranged hierarchically. The first-level concepts are formed on top of the foundational layer. The third-level blocks help students learn MPI and OpenMP programming concepts.

The fourth-level layer of concepts and connections deepens understanding of previously learned material by helping students develop more abstract ideas, such as:

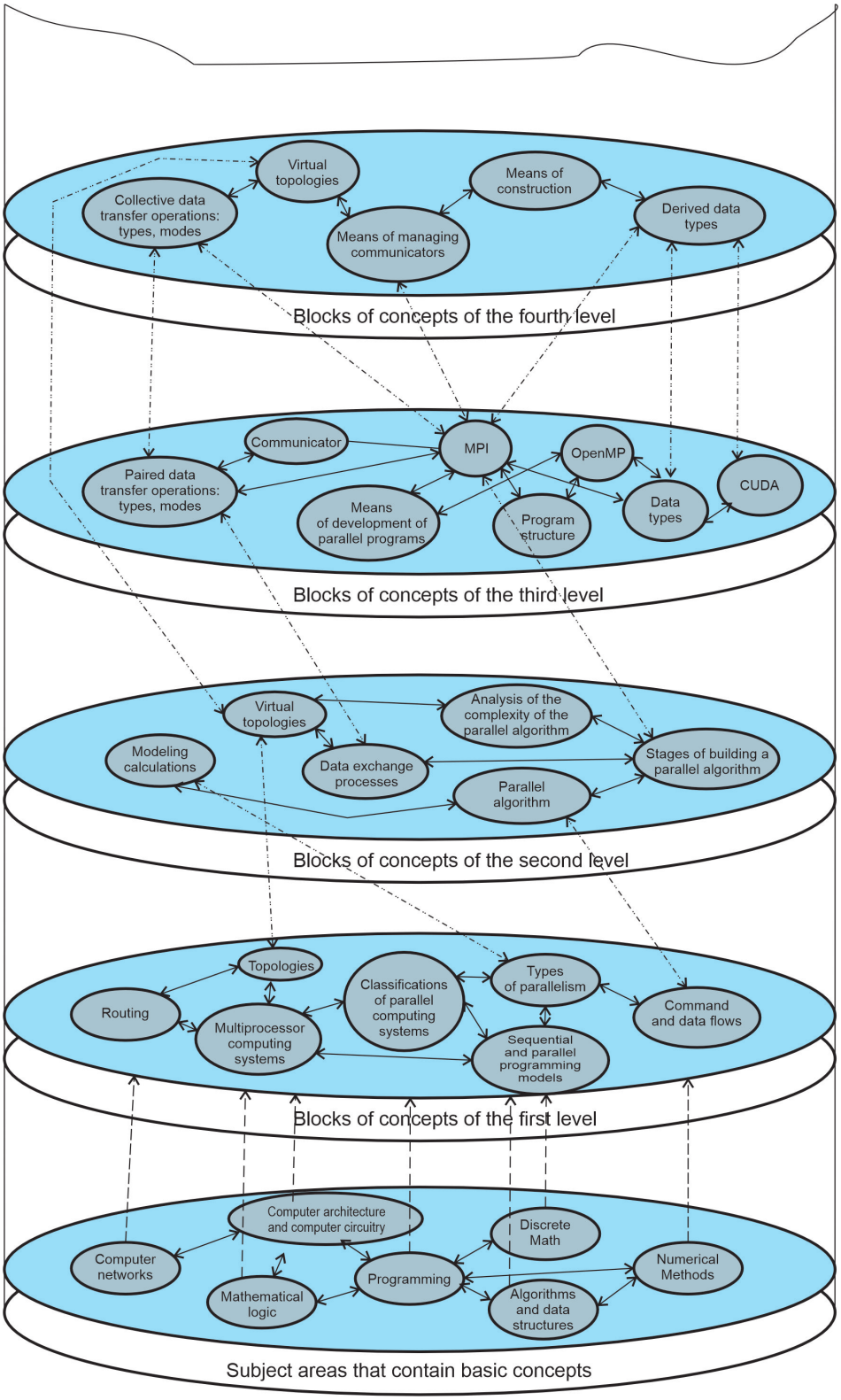

Figure 2. Training content model for software engineers in parallel computing.

- raw data types and their projection;
- controlling communicators and utilizing virtual topologies (linked to level 2).

This level of knowledge is applied to solve more complex problems in linear algebra, numerical methods, and graph theory.

It's important to note that the "concept cylinder" doesn't have an upper limit, signifying that the study of parallel computing shouldn't be confined to a single course. Relationships within the model aren't limited to adjacent levels, hence there's a direct connection between graph theory and parallel computing modeling, and algorithm theory with the analysis of parallel algorithm complexity.

Let's consider the elements of the model in more detail. Blocks of concepts of the first level are a set of basic concepts necessary for mastering the concepts of the following levels. They reflect the main blocks of the level, each of which can be represented by a separate logic-semantic scheme.

Blocks of concepts of the second level serve as a support for building the content part of the next section of the curriculum. These blocks also contain concepts, some of which were included in lectures and laboratory works, and others served as material for projects.

The blocks of concepts of the third level contain the material of the next chapter, within which the study of MPI and OpenMP technologies begins. These blocks of concepts are almost completely included in the classroom part of the course. When technical possibilities appear, the content of this level can be replaced by another programming technology, provided that its concepts are developed accordingly.

The fourth level of the model is the basis for processing the content of the next section of the module. The educational material of this level of the model includes only the basic concepts: creation of communicators, description and construction of data types intended for the user, etc. On the basis of blocks of this level, simple projects aimed at software implementation of mathematical models can be proposed.

By implementing the training content model for parallel computing software engineers into the curriculum, we can better understand the role of the course "Parallel and Distributed Computing" and the tools and methods used to develop parallel and distributed programs.

The model reveals that studying parallel computing necessitates gaining knowledge and skills in areas such as algorithmization, object-oriented programming, operating systems, computer circuitry, and computer architecture. These foundational knowledge and skills underpin the first and second level concepts of the model, enabling effective use of existing parallel libraries situated at the third level.

It's worth noting that the Modeling and Control of Students' Learning Pathways, particularly in terms of acquiring key competencies, should be conducted in a Cloud Service [29–31].

By analyzing the model of training content for software engineers in parallel computing, we can draw the following conclusions:

1. It is possible to choose a programming language for working with parallel libraries only after students have mastered the basics of algorithmization and the basics of object-oriented programming, that is, not before the third training.
2. The development of parallel computing programs will be effective only after students have mastered the basics of computer architecture, that is, not before the second year of study.
3. The development of distributed computing programs will be effective only after the applicants study the course of computer networks, that is, not before the third training.

Given the above, it can be concluded that the optimal time for teaching the courses of parallel and distributed computing would be no earlier than the third year of study for effective comprehension.

3. Conclusions

We conducted an analysis to construct a training content model for software engineers in parallel computing. This comprehensive analysis drew upon the multi-year paradigms of global computer education set forth by ACM, as well as insights gleaned from fundamental works authored by international experts in the field. The result of this endeavor was the formulation of a curated list of topics intended to shape the education of software engineers in parallel computing, complete with a recommended sequence to chart their learning journey.

Our investigation included an assessment of the current state of training for prospective software engineers in parallel computing within higher education institutions (HEIs) in Ukraine. This evaluation played a pivotal role in shaping our model. The model underscores that, prior to delving into the intricacies of parallel libraries (positioned as the third level in our model), students need to build a solid foundation in the first and second level concepts. This necessitates proficiency in algorithmization, object-oriented programming, an understanding of operating systems, familiarity with computer circuitry, and a grasp of computer architecture. As a result, a comprehensive study of the "Parallel and Distributed Computing" course is best undertaken in the third year of study. However, it is advisable to introduce certain elements of parallel programming during programming courses in the first and second years.

The formulation of this model provides a well-grounded rationale for positioning the "Parallel and Distributed Computing" course in the third year of the bachelor's program in computer science, within the broader structural and logical framework. The course curriculum, carefully designed to align with the third and fourth levels of the content model for training of software engineers in parallel computing, reflects this placement.

In the future, our focus will shift toward the development of topics that can be seamlessly integrated into programming language courses during the first and second years, thereby equipping students with the foundational knowledge necessary to excel in the "Parallel and Distributed Computing" course.

ORCID iDs

Y O Sitsylitsyn https://orcid.org/0000-0002-3888-5575
V V Osadchyi https://orcid.org/0000-0001-5659-4774
V S Kruglyk https://orcid.org/0000-0002-5196-7241
O G Kuzminska https://orcid.org/0000-0002-8849-9648

References

[1] Striuk A M and Semerikov S O 2022 *Journal of Physics: Conference Series* **2288**(1) 012012 ISSN 17426588 URL https://doi.org/10.1088/1742-6596/2288/1/012012
[2] Vakaliuk T 2021 *Educational Technology Quarterly* **2021**(2) 257–273 URL https://doi.org/10.55056/etq.17
[3] Seidametova Z, Abduramanov Z and Seydametov G 2022 *Educational Technology Quarterly* **2022**(1) 20–34 URL https://doi.org/10.55056/etq.5
[4] Osadchyi V 2017 *Ukrainian Journal of Educational Studies and Information Technology* **5**(4) 89–99 URL https://doi.org/10.32919/uesit.2017.04.08
[5] Varava I P, Bohinska A P, Vakaliuk T A and Mintii I S 2021 *Journal of Physics: Conference Series* **1946**(1) 012012 ISSN 17426588 URL https://doi.org/10.1088/1742-6596/1946/1/012012
[6] Striuk A M and Semerikov S O 2019 *CEUR Workshop Proceedings* **2546** 35–57 ISSN 16130073
[7] Striuk A M, Semerikov S O, Shalatska H M and Holiver V P 2022 *CEUR Workshop Proceedings* **3077** 3–11 ISSN 16130073
[8] Ko Y, Burgstaller B and Scholz B 2013 *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* SIGCSE '13 (New York, NY, USA: Association for Computing Machinery) p 415–420 ISBN 9781450318686 URL https://doi.org/10.1145/2445196.2445320
[9] Vakaliuk T A, Kontsedailo V V, Antoniuk D S, Korotun O V, Mintii I S and Pikilnyak A V 2019 *Proceedings of the 2nd International Workshop on Augmented Reality in Education, Kryvyi Rih, Ukraine, March 22,*

2019 (*CEUR Workshop Proceedings* vol 2547) ed Kiv A E and Shyshkina M P (CEUR-WS.org) pp 66–80 URL https://ceur-ws.org/Vol-2547/paper05.pdf
[10] Zahorodko P V, Modlo Y O, Kalinichenko O O, Selivanova T V and Semerikov S O 2020 *CEUR Workshop Proceedings* **2832** 94–103 ISSN 16130073
[11] Diaz J, Muñoz-Caro C and Niño A 2012 *IEEE Transactions on Parallel and Distributed Systems* **23** 1369–1386 URL https://doi.org/10.1109/TPDS.2011.308
[12] Sitsylitsyn Y 2020 *SHS Web of Conferences* **75** 04017 URL https://doi.org/10.1051/shsconf/20207504017
[13] Marowka A 2008 *IEEE Distributed Systems Online* **9** 1–1 URL https://doi.org/10.1109/MDSO.2008.24
[14] Wilkinson B, Villalobos J and Ferner C 2013 *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* SIGCSE '13 (New York, NY, USA: Association for Computing Machinery) p 409–414 ISBN 9781450318686 URL https://doi.org/10.1145/2445196.2445319
[15] Capel M I, Tomeu A J and Salguero A G 2017 *Journal of Parallel and Distributed Computing* **105** 42–52 ISSN 0743-7315 Keeping up with Technology: Teaching Parallel, Distributed and High-Performance Computing URL https://doi.org/10.1016/j.jpdc.2017.01.010
[16] Giacaman N and Sinnen O 2018 *Journal of Parallel and Distributed Computing* **118** 247–263 ISSN 0743-7315 URL https://doi.org/10.1016/j.jpdc.2018.02.028
[17] Vasconcelos L B A, Soares F A L, Penna P H M M, Machado M V, Góes L F W, Martins C A P S and Freitas H C 2019 *2019 IEEE Frontiers in Education Conference (FIE)* pp 1–8 URL https://doi.org/10.1109/FIE43999.2019.9028566
[18] The Joint Task Force on Computing Curricula 2001 *J. Educ. Resour. Comput.* **1** 1–es ISSN 1531-4278 URL https://doi.org/10.1145/384274.384275
[19] Cassel L, Clements A, Davies G, Guzdial M, McCauley R, McGettrick A, Sloan B, Snyder L, Tymann P and Weide B W 2008 Computer Science Curriculum 2008: An Interim Revision of CS 2001 Tech. rep. New York, NY, USA URL https://doi.org/10.1145/2593246
[20] Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM) and IEEE Computer Society 2013 *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science* (New York, NY, USA: Association for Computing Machinery) ISBN 9781450323093
[21] Clear A and Parrish A 2020 Computing curricula 2020: Paradigms for global computing education Paradigms for global computing education Association for Computing Machinery URL https://dl.acm.org/citation.cfm?id=3467967
[22] Prasad S K, Estrada T, Ghafoor S, Gupta A, Kant K, Stunkel C, Sussman A, Vaidyanathan R, Weems C, Agrawal K, Barnas M, Brown D W, Bryant R, Bunde D P, Busch C, Deb D, Freudenthal E, Jaja J, Parashar M, Phillips C, Robey B, Rosenberg A, Saule E and Shen C 2020 NSF/IEEE-TCPP Curriculum Initiative on Parallel and Distributed Computing - Core Topics for Undergraduates, Version II-beta Tech. rep. URL https://tcpp.cs.gsu.edu/curriculum/
[23] McCool M, Robison A D and Reinders J 2012 *Structured Parallel Programming: Patterns for Efficient Computation* (Morgan Kaufmann)
[24] 2019 Standart vyshchoyi osvity Ukrayiny pershoho (bakalavrs'koho) rivnya stupenya "bakalavr" za haluzzyu znan' 12 "Informatsiyni tekhnolohiyi" spetsial'nistyu 122 "Komp'yuterni nauky" URL https://mon.gov.ua/storage/app/media/vishcha-osvita/zatverdzeni%20standarty/2019/07/12/122-kompyut.nauk.bakalavr-1.pdf
[25] Dobrovol's'kyy H A 2020 Robocha prohrama navchal'noyi dystsypliny "Paralel'ni ta rozpodileni obchyslennya" URL https://moodle.znu.edu.ua/mod/resource/view.php?id=152709
[26] Khylenko V V 2020 Robocha prohrama navchal'noyi dystsypliny "Tekhnolohiyi rozpodilenykh system ta paralel'nykh obchyslen" URL https://nubip.edu.ua/sites/default/files/u286/kn_rp_tehnologiyi_rozpodilnih_sistem_ta_paralelnih_obchislen_rp_2021.pdf
[27] Oksiyuk O H 2020 Robocha prohrama navchal'noyi dystsypliny "Suchasni tekhnolohiyi bahatoprotsesornykh obchyslen"
[28] 2020 Robocha prohrama "Tekhnolohiyi rozpodilenykh system ta paralel'nykh obchyslen"' dlya studentiv za spetsial'nistyu 122 "Komp'yuterni nauky" osvitn'oyu prohramoyu "Komp'yuteryzatsiya obrobky informatsiyi ta upravlinnya"
[29] Pavlenko V, Prokhorov A, Kuzminska O and Mazorchuk M 2017 *CEUR Workshop Proceedings* **1844** 257–264 URL https://ceur-ws.org/Vol-1844/10000257.pdf
[30] Popel M, Shokalyuk S V and Shyshkina M 2017 *CEUR Workshop Proceedings* **1844** 327–339 URL https://ceur-ws.org/Vol-1844/10000327.pdf
[31] Vlasenko K, Chumak O, Bobyliev D, Lovianova I and Sitak I 2020 *CEUR Workshop Proceedings* **2740** 278–291 URL https://ceur-ws.org/Vol-2740/20200278.pdf