Olympiads in Informatics, 2017, Vol. 11, 93–107
© 2017 *IOI, Vilnius University*
DOI: 10.15388/ioi.2017.08

On a Metodology for Creating School Curricula in Computing*

Krassimir MANEV[1], Neli MANEVA[2]
[1]*Department of Informatics, New Bulgarian University,*
21, Montevideo str., 1618 Sofia, Bulgaria
[2]*Bulgarian Academie of Science, Institute of Mathematics and Informatics*
Acad. G. Bonchev Str., bl. 8, 1113 Sofia, Bulgaria
e-mail: kmanev@nbu.bg, neman@math.bas.bg

Abstract. It is obvious that the scientific and practical human activity *domain* having computers as main objects – called further Computing – became crucial for development and well-being of the Humanity. That is why education in Computing has to find its adequate place in the curricula of the school all over the world. The paper proposes a methodology for development of school curricula in Computing. The main feature of this methodology is that it is extracted from the guidance for creating university curricula in Computing of the most respected professional associations in the domain – ACM and the CS section of IEEE. As a sample that illustrates application of the methodology, a part of a specific Curriculum is created.

Keywords: computing, education in Computing, curriculum, curricula development, methodology.

1. Introduction

As defined in The Glossary of Education Reform (Glossary, 2017): “The term ***curriculum*** refers to the lessons and academic content taught in a school or in a specific course or program“. Very frequently the word is used to denote just the list of courses offered by a school, but such understanding of the term is more appropriate for some external use – to demonstrate briefly to the community or some institutions the essence of the proposed education. Here we will consider the notion in its depth. As the above mentioned Glossary underlines: “… curriculum typically refers to the knowledge and skills students are expected to learn …” which includes:

- The ***learning standards*** or ***learning objectives*** the students are expected to meet.
- The ***units*** and ***lessons*** that teachers teach.
- The ***assignments*** and ***projects*** given to students.

* This work was a result from the discussions during the IOI Workshop dedicated to teaching of Informatics in secondary schools, held in Bitola, Macedonia, April 2015.

- The ***learning resources*** – books, materials, videos, presentations and readings, used in a course.
- The ***assessment resources*** – tests, tasks, projects and other materials used to evaluate student's learning.

The authors of Glossary go beyond the above mentioned items and append that as inevitable parts of the modern curriculum have to be considered also:

- ***Curriculum philosophy***, i.e. the educational philosophy of the institution which developed it.
- ***Curriculum packages*** that have been developed by an outside organization as an example of usage of best practices in the domain.
- ***Curriculum standardization*** dedicated to increase teaching quality with greater curricular consistency throughout schools, regions and states.

In this paper we would like to propose an idea for methodology for development of curricula for the schools in the very large domain of the science and the technologies, which is dedicated to computers, their programming and usage called below *Computing*. These ideas raised during the IOI Workshop, held in Bitola, Macedonia, in April 2015 (called briefly *the Workshop*). Its main goal was to resume the current state of the teaching Computing in schools of the presented countries and to propose some ideas for development of an International School Curriculum in Computing (ISCC) on the base of the experience of the IOI community.

In Chapter 2 we briefly describe the status quo of teaching Computing in the secondary schools on the base of the presentations of the participating in the Workshop countries. We resume the results of some well-known efforts for creating an ISCC for schools over the world and define the goal of the paper – to transform the methodology of ACM-IEEE for creating university curricula in Computing into an agile methodology for creating curricula for the primary and secondary schools. In Chapter 3 we present the methodology of ACM-IEEE for development of university curricula. The transformed methodology is given in Chapter 4. Chapter 5 contains a sample of applying the methodology for development of a part of a specific curriculum. In Chapter 6 there are conclusion and perspectives for future work on the methodology.

2. Teaching Computing in Secondary Schools

The outcomes of the Workshop were presented in (Ackovska *et al.*, 2015), so let us briefly resume the status quo of education in Computing, as shown in the presentations of the participating 12 countries: Belgium, Bolivia, Brazil, Bulgaria, Croatia, Estonia, Hungary, Lithuania, Macedonia, New Zealand, Serbia and Slovenia. The dominating conclusion is that no one of the participating countries has officially approved national curriculum for teaching Informatics and only few of the countries have some curriculum for teaching Information Technologies. Compulsory courses in programming are given in some special (math and science oriented) schools. Because all participating in the Workshop countries are IOI members more serious teaching of Informatics (Programming, Data Structures and Algorithms) is proposed for pupils that take part in Olympiads in Informatics and other programming contests.

Having as one of the objectives of the Workshop the discussion of the possibilities for creation an international curriculum for primary and high school for teaching the basics of Computing, some existing attempts for elaboration of such curricula as well as guidelines for elaboration of such curricula were considered.

It is important to mention here the efforts of the colleagues from ACM. They started long years ago to discuss teaching of Computing (ACM Curriculum Committee, 1968). But since 2001 their activity become persistent together with colleagues from CS section of IEEE. And the goal is not just to create a curriculum but a **methodology for elaboration university level curricula** in all area of the domain that is linked to creating, programming and using computers, called here ***Computing*** (Computing Curricula, 2001; Computer Science Curricula, 2013; Information Technology, 2008). This significant work is in the base of this paper.

Working on this paper we considered also some available official and stable curricula for primary and secondary school – recommendations of Computing at School Working Group for United Kindom, endorsed by British Computer Society, Microsoft, Google and Intellect - UK trade body for the hi-tech sector, (Computer Science, 2012), the Australian curriculum guidance (Information and communication technology, date of publishing is unavailable), the USA recommendation K-12 (A Model Curriculum, 2003), the recent K-12 CS framework () and the Russian Secondary school Curriculum in Informatics (Kiryukhin and Tsvetkova, 2016). We do not use them in this paper but if the proposed here methodology is adopted then some ideas from the mentioned curricula will be very helpful. In this work we also rely on our more than 15 years experience as authors and co-authors of textbooks for the universities, as well as for Bulgarian primary and secondary school Curriculum in Computing, which is too dynamic, to be referred here.

After presenting the national reports of the participants in the Workshop a natural question raised: *Is it possible to have universal and unique ISCC?* The opinion of the most of the participants was that **it seems impossible**. Some arguments for this are listed below:

- The general *educational structure* of different countries presented in the Workshop is very different! We expect that if a larger number of countries decide to apply such ISCC then the differences will be even more drastic.
- The Countries have various history of introducing education in Computing and so – different level of experience in teaching these disciplines, different traditions, different quantity and quality of teachers, different computing resources, etc.
- The specific structure of the economy of the countries supposes different *models* of school education in Computing. For example, some countries with developed software industries will need a model which is different from the model of countries in which there is no such industry and there no intentions to develop it.

That is why it seems more realistic that **not a single Curriculum but many different Curricula** have to be created even in one specific country.

The question is: are we able to predict what exactly will be necessary for education in Computing in each country, depending of its specific needs and to predefine some fixed number of Curricula, developing them in depth. The answer again is – rather not! That is why it seems more appropriate for our goals a **set of *curriculum elements*** to be developed as well as ***guidance*** for using these elements in order to create many specific curricula.

Good example for such kind of activity are the mentioned above efforts of the professional organizations ACM and IEEE-CS to elaborate and maintain during the years a system of curriculum elements and guidance for creating specific curricula in Computing for the university stage of education. That is why these efforts are resumed in the next chapter as well as a proposal how the methodology of ACM-IEEE curricula guidance group could be tuned for our goals.

3. ACM-IEEE Computing Curricula Guidance

ACM-IEEE Computing Curricula Guidance is not a single methodical act. It has about 20 years history. It is a persistent work of the ACM-IEEE Computing Curricula Guidance work group, which started with a single recommendations updated on a regular base through the year that recently led to specific recommendation for the different fields of the Computing domain. That is why this important work could be a model for creating and future maintenance of corresponding guidance for development of the ISCC.

3.1. *Principles*

The ACM-IEEE curriculum guidance work groups are based on some principles, that seem to be applicable to our goals:

- Curriculum guidance should not only identify the **fundamental knowledge** and **skills** of the domain but should provide a **methodology** for creating specific courses, defining their content and the appropriate order of teaching them.
- The required body of knowledge must be **as small as possible**.
- Curriculum guidance must strive to be **international** in scope, **broadly based** and must include **professional practice** as an integral component.
- The rapid evolution of Computing requires an **ongoing review** of the corresponding curricula recommendations through the years.
- Curriculum guidance must be **sensitive to progress** in technology, pedagogy and lifelong learning.

3.2. *Structure*

The ACM-IEEE Computing Curricula Guidance has the following structure:

- *Computing* is a broad, scientific and practical human activity *domain* including a corpus of *teaching elements* – both theoretical (knowledge) and practical (skills), having computers as main objects – their creation as well as their programming and usage.

- *Body of knowledge* of the domain describes it's fundamental concepts, theories and notions. *Core of knowledge* specifies the body of knowledge elements, for which there exists a broad consensus that they are essential for the education in the domain. Body of knowledge is hierarchically structured in *fields*, *areas*, *units* and *topics*.
- *Learning objectives* are composed of two parts – a set of requirements that all students should be able to meet and set of requirements to promote individual assessment of each student achievements.
- *Curriculum models* present different approaches for organizing the educational process among which the creator of a specific Curriculum could choose. ACM-IEEE guidance considers three level of models – *introductory*, *intermediate*, and *advanced*.
- The part *Course descriptions* contains detailed description of the *courses* – both compulsory and elective. Course is the main structural object of each curriculum. One discipline could be covered by one or more courses and one course could be dedicated to one or more disciplines.

Some other parts of the guidance structure could be also considered on the next stages of development of guidance for ISCC.

3.3. *Body of Knowledge*

Body of knowledge of the Computing domain is hierarchically organized in four levels. Level 1 contains the *fields* of the domain; on Level 2 the fields are divided into *areas*; on Level 3 each area is divided to *units;* on Level 4 each unit is composed from *individual topics*.

ACM-IEEE group considered 5 **fields**: *Computer Engineering*, *Computer Science* (CS), *Information Systems*, *Software Engineering* and *Information Technologies* (IT).

For example, the field CS is composed of the following **areas**: Discrete Structures, Programming Fundamentals, Algorithms and Complexity, Architecture and Organization, Operating Systems, Net-Centric Computing, Programming Languages, Human-Computer Interaction, Graphics and Visual Computing, Intelligent Systems, Information Management, Social and Professional Issues, Software Engineering, Numerical Methods. The field of IT includes the **areas**: Information Technology Fundamentals, Human Computer Interaction, Information Assurance and Security, Information Management, Networking, Programming Fundamentals, Systems Administration and Maintenance, Social and Professional Issues, Web Systems and Technologies, etc.

As a first example we would like to mention the unit Discrete structures, which is the mathematical foundation of the domain. It includes the following units:

- Sets, relations, and functions (6).
- Basic logic (10).
- Proof techniques (12).
- Basics of counting (5).

- Graphs and trees (4).
- Discrete probability (6).

The underlying of a unit means that it is included in the Core of the corresponding area. In this case all items in both units are in the Core of the area. The numbers in brackets are the predicted in the ACM-IEEE guidance class hours.

As a second example let us to consider the **units** that ACM-IEEE guidance includes in the area Programming Fundamentals, common for the fields Computer Science and Information Technologies:

- Fundamental programming constructs (9/10).
- Algorithms and problem-solving (6/6).
- Fundamental data structures (14/10).
- Recursion (5/0).
- Object-Oriented Programming (0/9).
- Event-driven programming (4/3).

The numbers in brackets, separated by slash in this case are the predicted in the ACM-IEEE guidance class hours for the area when included in CS or IT field, respectively. As it could be seen the very small differences in the two areas are in the different number of class hours of the same units and that the unit Recursion is included in one of the area instead the unit OOP in the other. In this case all items in both units are in the Core of the area too, but it is not the case in all areas.

For each unit the guidance of ACM-IEEE contains description of the individual **topics** included in the unit and the corresponding learning objectives. As example, these topics for the unit Sets, relations and functions of the area Discrete structures are:

- Sets (Venn diagrams; Union, intersection, complement, Cartesian product, Power set; Cardinality of finite sets).
- Relations (Reflexivity, symmetry, transitivity; Relations of equivalence and Partial orders).
- Functions (Surjections, injections, bijection; Inverses; Composition).

3.4. *Learning Objectives*

Learning objectives are less formal but important part of the guidance in the structure of created curricula. In narrative form they not only show the vision of the creator of the curricula for achievement of the students but are the media for unifying the topics of the unit in something complete. Learning objectives of the topic Sets, relations and functions of the unit Discrete structures could be:

- Explain with examples the basic terminology of sets, relations, and functions.
- Perform the operations associated with sets, relations, and functions.
- Relate practical examples to the appropriate set, or relation or function model, and interpret the associated operations and terminology in the context.
- Demonstrate basic counting principles, including uses of diagonalization and the Pigeonhole principle.

3.5. *Models*

The different *models* in ACM-IEEE curriculum guidance are strongly connected to the university degree of education. A model is some leading approach of organizing of the teaching material, i.e. splitting the core units in courses, ordering the courses in time etc. The model includes different quantitative parameters of the specific university – how many semesters covers the curriculum, how many weeks per semester, how many class hours per week, the ratio of classes for lectures toward classes for practice, the ratio of classes for compulsory courses toward elective courses, etc.

But the model includes some other parameters, too. For example, on the introductory level, different models could be identified depending of the preferred paradigm of the introductory programming course: Imperative language first, Object-oriented language first or Functional language first. The chapter of ACM-IEEE Curricula Guidance that discusses models contains some other organizing concepts and could be used for some helpful ideas.

3.6. *Courses Creation*

When all curriculum elements are defined and the model fixed, the last step is creation of the set of courses. For this purpose a *decision table* is created. Its rows are labeled with the units (or topics if the curriculum has to be more detailed) and columns labeled with the courses chosen by the requirements of the model. The next step is to decide which core element (unit or topic) in which course will be included, with how many class hours, and how the elements will be ordered inside the course. The final step is to choose distribution of identified courses in semesters. A part of such decision table of covering some of the elements by some courses for the model Imperative language first from ACM-IEEE Curricula Guidaence is shown on Fig. 1.

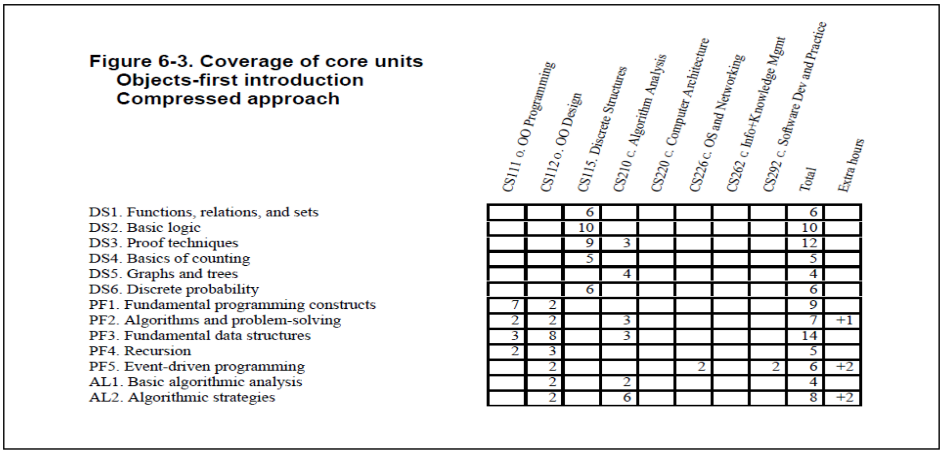

Figure 6-3. Coverage of core units
Objects-first introduction
Compressed approach

	CS111 o. OO Programming	CS112 o. OO Design	CS115. Discrete Structures	CS210 c. Algorithm Analysis	CS220 c. Computer Architecture	CS226 c. OS and Networking	CS262 c. Info+Knowledge Mgmt	CS292 c. Software Dev and Practice	Total	Extra hours
DS1. Functions, relations, and sets			6						6	
DS2. Basic logic			10						10	
DS3. Proof techniques			9	3					12	
DS4. Basics of counting			5						5	
DS5. Graphs and trees				4					4	
DS6. Discrete probability			6						6	
PF1. Fundamental programming constructs	7	2							9	
PF2. Algorithms and problem-solving	2	2		3					7	+1
PF3. Fundamental data structures	3	8		3					14	
PF4. Recursion	2	3							5	
PF5. Event-driven programming		2				2		2	6	+2
AL1. Basic algorithmic analysis		2		2					4	
AL2. Algorithmic strategies		2		6					8	+2

Figure 1. Decision table for creating courses (Computing Curricula, 2001).

4. Implementing the ACM-IEEE Methodology

4.1. *Body of Knowledge*

Creating a methodology for development of different curricula, our efforts will be concentrated especially on fields Computer Science and Information Technologies. We consider teaching Computing Systems, Information System and Software Engineering not appropriate for the school students because these fields require maturity of higher level and some specific knowledge – electronic engineering, business management and software business management, which are more appropriate for universities. But some knowledge elements from these fields could find place in our guidance, too.

We propose to use the body of knowledge from the version of ACM-IEEE guidance from 2001 and, after a broad discussion/questioning in IOI community and outside, to extract the areas and the units that are appropriate for the school. It will be necessary for sure to append some units or/and topics that are considered nowadays not appropriate or not necessary for the universities, but are important for school students. As the presentation of the countries that participated in the Workshop shows, there is some corpus of knowledge in the area IT (let us call it *computing literacy*) that is traditionally studied in schools, but not as university program, and have to find place in ISCC. The units of core of knowledge have to be identified as a result of the discussion/questioning mentioned above. Further we can choose an appropriate quantity of class hours for each unit.

A similar process of deleting/appending has to be done for the individual topics of each included in the body unit (core or elective). In parallel, the corresponding educational objectives have to be edited with the respect of the age of the student. It is quite possible that in our recommendations we will have to define few alternative versions of the learning objectives connected with each individual topic, unit and even area, depending on the chosen educational model.

4.2. *Educational Model*

Identifying the necessary educational model in our case will be the most important and difficult task. As mentioned above the model is the element of each curriculum that has to introduce a flexibility in the process of creating curricula. This will be our instrument to cover as many as possible concepts of organization of education on different age levels of the school, as well as of the different kind of schools.

Let us start with the levels of education which exist in the educational system of each country (and could be different). In most countries there are three levels in secondary school – *primary, intermediate* and *high*. In Bulgarian system, for example, two models exist for the level of education:

- 1–4 grades as primary, 5–8 grades as intermediate, 9–12 grades as high.
- 1–4 grades as primary, 5–7, grades as intermediate, 8–12 grades as high.

The trend is the first of them to be eliminated soon or later. There is also trend to separate the high level to two sublevels 8–10 grades and 11–12 grades in order to give a possibility for some specialization in the sublevel 11–12 of the students. Similar subdivision is predicted in (Computer Science, 2012).

In addition we have to consider also existence of some special schools – mathematical, language, art, sport, professional, etc. That is why it is necessary to define very carefully the core units of the body of knowledge which to be common for the national educational system in order to give to each student, graduated in secondary school the possibility to continue her/his education in an university program of Computing does not matter in what kind of school is graduated. As well as to classify the elective units of the body in such a way that the curricula for the specialized schools to include the most appropriate for the peculiarities of the school elective units.

In Bulgarian system, for example, the courses are classified in 3 categories – *compulsory*, *compulsory-elective* (which means that the students have to take *m* among the proposed set of *n* such courses) and *free-elective* (which means that students are not obliged at all but could take as many such courses as they would like). In last category the corresponding school could include any course which is appropriate to the profile of the school (professional school of machine engineering, for example, could include learning of one CAD/CAM system, which is inappropriate for other schools).

Is it possible to define different kind of models also on the base of preferred main fields – *CS-oriented* (most appropriate for mathematical and engineering schools), *IT-oriented* (more appropriated for language, art and sport schools), *CS&IT-oriented* (more appropriated for regular schools), etc.

The most important difference between university models (we mean the *classic universities* but not the so called *liberal art*, where the educational model is more close to the models of secondary schools) and the school models is that in the university programs in the domain of Computing in each moment students take few courses from the domain. In secondary school model we will have in each moment only one or maximum two courses in the domain with 1-4 class hours per week – asking for more class hours for Computing nowadays seems not realistic. So the model has to include some „class hours per week“ *scheme*, which defines the grade, number of courses, class hours per week and distribution of the class hours between the two courses (or between IT and CS if the model includes only one course).

4.3. *Creating Specific Courses*

When the body of knoledge and the model are fixed this stage will be relatively easy. Because we supposed that the curricula will have usually 1 course per school year or maximum 2. That is why we propose the corresponding decision table to be composed of rows labeled with grades and columns labeled with units. If necessary the columns will be distributed between the courses when the model predicts two.

5. A Sample for Implementing the Methodology

In this Chapter we apply the proposed methodology for creating part of a possible School Computing Curriculum. In the sample we use also our experience of long years writing textbooks in *Informatics* (this is the traditional name of the course corresponding to CS) and IT for the Bulgarian schools, obeyng to set of so called *learning programs* (separate program for each grade‘s course), composition of which is not really curriculum regarding used in this paper notion. The principles and requirements of these learning programs could not be neglected, because they are mandatory for all schools, and to the authors of textbooks, respectively. That is why applying the proposed methodology we have to keep in mind the corresponding learning programs, too.

Traditionally for the Bulgarian schools (and, probably, the same is in many other countries) the teaching elements in the Learning programs in Computing are from two main areas from ACM-IEEE Curricula Guidance – Computer science and Information Technologies. That is why we will create our sample based on this two areas too.

5.1. *Model*

Our model will be dedicated for the regular schools. That is why it will be of type CS-IT oriented with 4 levels – primary level (1-4 degrees), intermediate level (4-7 degrees) and two high levels – basic high level (8-10 degrees) and advanced high level (11-12 degrees). The school year in the primary level will be 32 weeks long, 34 weeks long in the intermediate level and 36 week in the high levels.

There will be a single compulsory course per year named *Information Technologies* in the primary and intermediate level, single course per year called *Informatics and Information Technologies* for the basic high level and two courses for the advanced high level – *Informatics* and *Information Technologies*.

The number of class hours will be: 1 class hour per week for the primary level, 2 class hours per week for the intermediate level, 3 class hours per week for the basic high level and 4 class hours per week for the advanced high level – 3 for the course Informatics and 1 for the course Information technologies. The 108 class hours in the basic high level will be divided to 36 hours for Information Technologies and 72 hours for Informatics. The 144 class hours in the advanced level will be divided to 36 hours for Information Technologies and 108 hours for Informatics. I.e. our model predict 540 class hours in Information Technologies and 432 class hours in Informatics.

In addition to the single compulsory course a single compulsory-elective course named *Application of Information Technologies* will be specified for the intermediate level. A single compulsory-elective course named *Application of Informatics and Information Technologies* will be specified for the high level also. Both compulsory-elective courses will be with 2 class hours per week. The distribution of classes between Applications of Informatics and Applications of Information Technologies will be assigned to the administration of the schools.

5.2. *Body of Knowledge and Learning Objectives*

It is obvious that creating a complete Computing curricula is not possible in the volume of a journal paper. We will demonstrate here the application of the methodology only to some parts of the curriculum for the described above model. As mentioned above the body of knowledge for the created curriculum will be chosen mainly from the fields Computer Science and Information Technology, but for illustration we will develop only the Computer Science part of the body. We will use as a start point (Computing Curricula, 2001) having in mind that the last version is less appropriate for schools because it contains some new conceptions, which are most appropriate for the university programs.

According (Computing Curricula, 2001) the areas of the field Computer Science are: Discrete Structures, Programming Fundamentals, Algorithms and Complexity, Architecture and Organization, Operating Systems, Net-Centric Computing, Programming Languages, Human-Computer Interaction, Graphics and Visual Computing, Intelligent Systems, Information Management, Social and Professional Issues, Software Engineering, and Computational Science. The first task is to decide which areas are appropriate for the school and to distribute the 432 hours predicted by the model among the chosen areas.

For this purpose we consider the table from Fig. 5-1, page 17 in (Computing Curricula, 2001). This is the place where as broad as possible consensus is necessary in order to decide which are the areas which are appropriate for the school education in Computing and which are the units in each area that will be included in the Curriculum.

Modeling the work of the experts we proceeded in following way: we deleted from the table the units that we consider inappropriate for the school and then deleted the areas for which all units were deleted. Doing this we selected only the core units and distributed predicted by the model 432 hours among these units. The results of this stage are presented in the leftmost column of Table 1.

Table 1

Decision table for the Curriculum

	8	9	10	11	12
DS. Discrete Structures (46 hours)					
DS1. Functions, relations, and sets (10)	10				
DS3. Proof techniques (10)		10			
DS4. Basics of counting (12)				12	
DS5. Graphs and trees (14)		4	10		
PF. Programming Fundamentals (88 hours)					
PF1. Fundamental programming constructs (14)	10	4			
PF2. Algorithms and problem-solving (14)	6	8			
PF3. Fundamental data structures (30)	10	10	10		
PF4. Recursion (12)				6	6
PF5. Event-driven programming (18)	6	6	6		

To be continued

Continuation of Table 1

	8	9	10	11	12
AL. Algorithms and Complexity (24 hours)					
AL1. Basic algorithmic analysis (4)		2	2		
AL2. Algorithmic strategies (6)		2	4		
AL3. Fundamental computing algorithms (12)			12		
AL6. The complexity classes P and NP (2)			2		
AR. Architecture and Organization (29 hours)					
AR1. Digital logic and digital systems (6)				6	
AR2. Machine level representation of data (2)	2				
AR3. Assembly level machine organization (9)				9	
AR9. Architecture for networks (12)				12	
OS. Operating Systems (18 hours)					
OS1. Overview of operating systems (2)	2				
OS2. Operating system principles (2)		2			
OS6. Device management (4)		4			
OS8. File systems (4)	2	2			
OS12. Scripting (6)			4	2	
NC. Net-Centric Computing (33 hours)					
NC1. Introduction to net-centric computing (3)				3	
NC2. Communication and networking (9)				9	
NC4. The web as client-server computing (6)				6	
NC5. Building web applications (6)				6	
NC9. Wireless and mobile computing (9)				9	
PL. Programming Languages (42 core hours)					
PL1. Overview of programming languages (2)				2	
PL2. Virtual machines (2)				2	
PL3. Introduction to language translation (2)				2	
PL4. Declarations and types (12)					12
PL5. Abstraction mechanisms (6)					6
PL6. Object-oriented programming (18)					18
HC. Human-Computer Interaction (44 hours)					
HC1. Foundations of human-computer interaction (12)	4				
HC2. Building a simple graphical user interface (12)	4	8			
HC5. Graphical user-interface design (10)			10		
HC6. Graphical user-interface programming (10)				10	
GV. Graphics and Visual Computing (20 hours)					
GV1. Fundamental techniques in graphics (8)	6	2			
GV2. Graphic systems (12)	4	4	4		
IS. Intelligent Systems (not included)					
IM. Information Management (50 hours)					
IM1. Information models and systems (3)					3
IM2. Database systems (3)					3
IM4. Relational databases (4)					3
IM5. Database query languages (10)					9
IM6. Relational database design (30)					30

To be continued

	8	9	10	11	12
SP. Social and Professional Issues (15 core hours)					
SP1. History of computing (2)	2				
SP2. Social context of computing (3)					3
SP4. Professional and ethical responsibilities (3)				3	
SP5. Risks and liabilities of computer-based systems (2)		2			
SP6. Intellectual property (3)					3
SP7. Privacy and civil liberties (2)			2		
SE. Software Engineering (33 core hours)					
SE1. Software design (10)		2	2	3	3
SE2. Using APIs (7)			4	3	
SE3. Software tools and environments (2)	2				
SE6. Software validation (8)	2			3	3
SE8. Software project management (3)					3
SE9. Component-based computing (3)					3
CN. Computational Science (not included)					
	72	72	72	108	108

5.3. *Course Creating*

The last stage of the Curriculum development is the construction of courses. Here the different developers could take into account their national traditions and concepts for the organization of the knowledge body. But the concept of arranging the material in its internal logic and with increasing the level of difficulty is inevitable.

In our decision table (rightmost 5 columns of Table 1) only the core items of the body of knowledge are included. Similar work has to be done for the elective items when the model is extended with the corresponding compulsory-elective and free-elective courses.

Additional information that is missing in our decision table and has to be appended, is the recommendation how the predicted hour classes to be distributed between lectures (theoretic classes) and exercises (practice classes). It is clear that the most of exercises will be in the topics dedicated to programming but the creator of the Curriculum has to fix carefully the ratio. Without some efforts to build the necessary practical skills the education in Computing is nonsense.

6. Conclusion

In this paper a methodology for creating school curricula in Computing was presented. The main feature of this methodology is that it is extracted from the methodology for creating university curricula in Computing of the most respected professional associations in the domain – ACM and the CS section of IEEE. There is no doubt that teaching

Computing has not the place it deserved in schools over the world but this could not continue infinitely. Computing is a fundamentally important for the development of the humanity, as well as the languages, mathematics and philosophy. So the Computing community and the IOI community as a part of Computing community is obliged to answer the questions: what part of knowledge/skills of the domain to be taught/practices in the schools, when and how.

It is obvious that a single journal paper could not contain all aspects of such complex activity as curricula development. For implementing the proposed here ideas a large amount of work has to be done. On the first place a broad discussion is necessary for selecting the items of the body of knowledge (both core and elective) that have to be included. The expertize of two professionals is not enough at all.

On the second place, the last level of the body of knowledge hierarchy – the topics – has to be included in consideration. Only in such way the precise distribution of the class hours and precise ratio lectures/exercises could be estimated. The proposed in the sample distribution is rather subjective and so not precise.

So, the effective and helpful guidance for development of school curricula in Computing could be created only as a result of mentioned above broad discussion. We believe that IOI community could play and have to play significant role in this process. As IOI community includes high quality proffessionals – researchers, school teachers, and university professors from different arreas in the domain, who posses huge experience in preparing programmers. We believe that introducing adequate curricula in the schools will rise the level of education, will increase the number of students that compete and, finally, will prepare better the students with interest in the domain for the university level education and the future professional carrier, which is the main goal of the school olympiads in Informatics.

References

Ackovska, N, Németh, Á.E., Stankov, E., Jovanov, M. (2015). Creating an International Informatics Curriculum for Primary and High School Education (Report of the IOI Workshop). *Olympiads in Informatics*, 9, 205–212.

ACM Curriculum Committee on Computer Science (1968). Curriculum 68: Recommendations for Academic Programs in Computer Science. *Communications of ACM,* 11(3), 151–197.

A Model Curriculum for K–12 Computer Science: Final Report of the ACM K–12 Task Force Curriculum Committee, Computer Science Teachers Association (2003). Downloaded at June 2017: `http://marvin.cs.uidaho.edu/Teaching/K12-CS/acmK12CurriculumFinal2003.pdf`

Glossary (2017). *The glossary of Education Reform*. Downloaded at June 2017: `http://edglossary.org/curriculum/`

Computer Science: A Curriculum for Schools (2012). Downloaded at June 2017: `http://www.computingatschool.org.uk/data/uploads/ComputingCurric.pdf`

Computer Science Curricula 2013 (2013). Downloaded at June, 2017: `https://www.acm.org/education/CS2013-final-report.pdf`

Computing Curricula (2001). Computer Science (2001). Downloaded at June, 2017: `http://www.acm.org/education/curric_vols/cc2001.pdf`

Information and communication technology capability, (date of publishing unavailable), downloaded at June 2017: `http://www.australiancurriculum.edu.au/generalcapabilities/information-and-communication-technology-capability/introduction/introduction`

Information Technology (2008). Downloaded at June, 2017: `https://www.acm.org/education/curricula/IT2008%20Curriculum.pdf`

K-12 CS framework (date of publishing unavailable), downloaded at June 2017: `https://k12cs.org`

Kiryukhin, V.M., Tsvetkova, M.S. (2016). Informatics at Russian Secondary School. *Olympiads in Informatics*, 10 (special issue), 135–24.

Kr. Manev is a professor of Discrete mathematics and Algorithms in New Bulgarian University, PhD in Computer Science. He has published about 75 scientific papers and more than 30 textbooks in the fields of Informatics and Information Technologies. Member of Bulgarian National Committee for Olympiads in Informatics since 1982 and President of NC from 1998 to 2002; member of the organizing team of IOI'1989 and IOI'1990; Chairmen of IOI'2009; Leader/ Deputy Leader of Bulgarian team for IOI in 1989, 1998, 1999, 2000, 2005 and 2014. From 2001 to 2003 and from 2011 to 2013 he was elected member of IC of IOI, since 2005 to 2010 represented in IC the Host country of IOI'2009. Now he is a President of IOI for the period 2014–2017.

N. Maneva is a professor, Ph.D. in Computer Science, from Institute of Mathematics and Informatics, Bulgarian Academy of Sciences. Her major fields of scientific research are Software Engineering, Software Quality Assurance, Model-driven Software Development and Formal Methods in Software Engineering. She has published about 70 scientific papers and more than 30 textbooks in the field of Informatics and Information technology. She was a member of Bulgarian National Committee for Olympiads in Informatics from 1982 till 1993 and a Secretary of the Scientific Committee of IOI'1989..