# About One Approach to Building Systems for Testing Physical Knowledge

Michail Lvov [0000-002-0876-9928], Sergey Kuzmenkov [0000-0002-5257-9523] and Hennadiy Kravtsov [0000-0003-3680-2286]

Kherson State University, 27 Universitetska st., Kherson, 73000 Ukraine lvov@ksu.ks.ua, ksg@ksu.ks.ua, kgm@ksu.ks.ua

**Abstract.** The paper presents an approach to building a system for testing procedural physical knowledge, i.e. knowledge of basic physical laws and the ability to use them.

This approach consists of constructing mathematical models for each academic module in a physics course. The main constructive objects are test templates, which are mathematical models of test tasks, based on physical models of systems, processes and phenomena. The template of the class of physical tests for checking knowledge of the physical laws and abilities of transformations of a physical system is represented by a set of geometric drawings, diagrams, graphs of functional dependencies, a system of formulas for transforming physical values, templates of scenarios for changing the states of a physical system and a response template.

Each such template can be used both in generating similarity algorithms for specific multiple tests, and in algorithms for automatically checking the correctness of answers. The proposed method allows describing a relatively simple class of specific tests. An important feature of the system is the ability to automatically check not only the final answer, but also the parameters of the intermediate states of the physical system. The implementation of a procedural physical knowledge testing system can be performed by creating software interactive multimedia objects using the methods of computer mathematics and algebraic programming technology.

**Keywords:** physics test, mathematical models of test, test template, interactive multimedia object, algebraic programming technology.

# 1 Introduction

The teaching process of the exact sciences in secondary school, especially physics, includes both the lecture part of the lesson and the active teaching forms: practical works, double-acting works, independent and test papers, etc. Thus, it is necessary to control not only the declarative knowledge, but also procedural knowledge, that is, knowledge of methods for solving physical problems. General methods of constructing of computer mathematics systems for educational purposes, one of the subsystems

of which is the testing environment, are described in [1-10]. We adapted these methods to build a procedural physical knowledge testing system.

Physical tests, which will be discussed, are designed to control procedural knowledge of a physics course using the example of a physics course in the 7th-9th grades of the secondary school in Ukraine. Technologies for the control of procedural knowledge have not yet been studied and developed. Thus, the problem of research is relevant.

In [11] the general approach to the description of subject areas in mathematics and other exact sciences is described. In [12], a methodology was proposed for constructing systems for testing procedural mathematical knowledge and its refinement for constructing of a system for algebraic knowledge testing. In [13], a methodology for constructing systems for procedural mathematical knowledge testing and its refinement for building a system for geometric knowledge testing was proposed.

In this paper, within the framework of this general approach, the features of the construction of physical knowledge testing systems are considered. It is assumed that a testing system containing procedural physical tests will be implemented as a module *Physical Knowledge Testing Environment*, which can be used both in computer-aided educational mathematics systems (CAEMS) in physics and for other purposes.

The problem of the present work can be formulated as a study of the functional requirements, mathematical models and algorithms for constructing of a system for testing procedural physical knowledge in CAEMS.

Separately, it can be highlighted the task of designing and creating software environment testing of physical knowledge and skills. In this case, the actual need is to develop a software module for physical knowledge testing and skills in distance learning systems (DLS) [14].

The widespread use of distance learning in educational institutions in secondary and higher education was increased demands on the quality of distance learning. One of the main tasks is to improve the quality of distance courses, especially training modules of a practical orientation. There are testing systems, simulators, laboratory and practical works. The problem is that the international standards do not clearly spell out the specifications for the structure and implementation of such training modules. At the same time, an essential requirement for testing systems is the requirement of compliance with existing international standards of distance learning, which contributes to the integration of learning information resources. The problem of standardization of methods and technologies for developing a system of physical knowledge and skills testing is debatable and requires discussion of a wide range of specialists. In this paper, we consider methods for solving the problem of modeling and implementing of a module for physical knowledge and skills testing in distance courses based on *adaptive (laboratory) and template tests technology*, which is consistent with international standards IMS and SCORM [15, 16].

The relevance of solving the problem of designing of a module for physical knowledge and skills testing in distance courses is determined by the fact that the use of test types based on choice currently allows only knowledge control [17], but does not control skills, which is essential for the natural sciences.

# 2 Examples of formal models of physical processes

The subject area (ontology) is represented by structural-logical schemes (SLS). These ontologies are represented by a three-level hierarchy:

"Discipline" - "Educational Module" - "Model of Phenomenon, Process".

Our approach is the system of test tasks should be unified within the framework of the discipline "Physics 7-9" [18]. However, its development is carried out in stages PHYSICS 7 – PHYSICS 8 – PHYSICS 9 – ....

 $r_{111}$  Sics  $r_{-111}$  Sics  $s_{-111}$  Sics  $s_{-111}$  Sics  $s_{-111}$ 

Before proposing a general definition of the pharma model of the physical process, we consider several examples.

#### 2.1 Example 1. Ohm's law for subcircuit.

From the program Physics 7-9: Ohm's law for subcircuit. Series and parallel connection of conductors.

The testing system should verify the application of the following physical laws:

- 1. The current strength *I* in the circuit is determined by the Ohm's law formula: I = U/R
- 2. With a series connection of resistors, their total resistance is equal to the sum of their resistances:  $R = R_1 + R_2 + R_3 + ...$
- 3. With parallel connection of resistors, their total conductivity (inverse of resistance) is equal to the sum of their conductivities:  $1/R = 1/R_1 + 1/R_2 + ...$

Test task. The given scheme (Fig. 1)

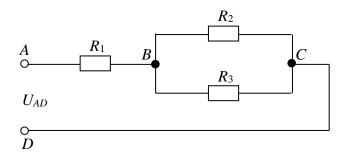


Fig. 1. Test task on the topic of "Ohm's law for subcircuit".

Given: 
$$R_1 = 100 \Omega, R_2 = 20 \Omega, R_3 = 60 \Omega,$$
  
 $U_{AD} = 220 V$   
Find:  $I_{BC}, R_{AD}$ 

The formal circuitry model should use

- model of resistor (resistance),

- model of the element (primitive) of the electrical circuit (EC),
- circuit model as a connection of elements.

**Models** of resistance, current strength, voltage are the models of physical quantities. A model of a physical quantity is a five (name, unit of measure, range of values, step of change, value).

The model of the EC element (electrical circuit) is a data set:

- designation of connection point 1,
- designation of connection point 2;
- resistance model (resistor);
- current strength model;
- voltage model.

The EC model is defined as follows: for circuit elements  $\sigma_1, \sigma_2, ..., \sigma_k$  the scheme  $\sum$  is a formula for the variables  $\sigma_1, \sigma_2, ..., \sigma_k$  in the signature  $\langle +, \parallel \rangle$ , where "+" is the sign of the serial connection operation, " $\parallel$ " is the sign of the parallel connection operation with consistent notations of beginnings and ends. For example, the formula of the circuit in Figure 1 is  $\sum = \sigma_1 + (\sigma_2 \parallel \sigma_3)$ . Here, the primitives are the following EC elements:

$$\sigma_1 = (A_1, B_1, R_1, I_1, U_1), \sigma_2 = (A_2, B_2, R_2, I_2, U_2), \sigma_3 = (A_3, B_3, R_3, I_3, U_3),$$
(1)

In the circuit model it is natural to distinguish subcircuits. In general, the relation "circuit – subcircuit" is determined through one (any) of the circuit construction operations: if  $F(x_1, ..., x_n)$  is a circuit construction operation, and  $\sum_1, ..., \sum_n$  is a circuits, then  $\sum = F(\sum_1, ..., \sum_n)$  is a constructed circuit and  $\sum, \sum_j$  are in a "circuit-subcircuit" relationship. In this example, three primitive subcircuits are defined:

$$\sigma_1 = <\_R_1\_>, \sigma_2 = <\_R_2\_>, \sigma_3 = <\_R_3\_>,$$
(2)

as well as constructions

$$\sum_{1} = < \sigma_{2 \parallel} \sigma_{3} >, \sum_{1} = < \sigma_{1 +} \sum_{1} >$$
(3)

For each subcircuit, as well as for primitive subcircuits, variables R, I, U(1) are defined, and also meta variables – the points of connection of subcircuits into circuits. In (2), (3) these points are marked with an underscore. So any scheme  $\sum = (A_{\Sigma}, B_{\Sigma}, R_{\Sigma}, I_{\Sigma}, U_{\Sigma})$ . Connection points of subcircuits are denoted by letters – the values of metavariables. The data that is included in the sections *Given* and *Find* in Test task, is displayed using these symbols. The notation can be simplified, but we do not give a simplification mechanism (see Fig. 1).

**Test generation method:** The scheme construction mechanism is called as *the test design template*.

A set of subcircuits (for example, (2) and (3)) is called as *instance of a test template*.

The test task generation method is carried out in three stages (Fig. 2):

1. The test task generation begins with the use of a construction template to obtain a specific instance of the test template.

2. The construction of a set of test template instances can be done pre-bye (see the *Difficulty of the test task*)

3. The test task is generated from the test template instance by selecting the variable sections *Given* and *Find*, as well as randomly generating the values of variables of the section *Given*.

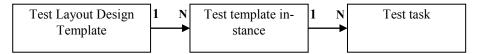


Fig. 2. Stages of the test task formation (on the topic of Ohm's Law for the subcircuit).

The test task generation is now carried out by selecting the test template instance schema, randomly selecting variables and their values for the *Given* section, as well as randomly selecting the names of other variables for the *Find* section.

**Test's Complexity.** The complexity of the model and the test, including the level of computational complexity, is determined by the number of characters of operations in the formula of the test template instance. For the scheme of this level of complexity, the complexity of the actual test task is also determined based on the number of quantities in the *Find* section.

The presented example should be considered rather complicated, since the formula contains two signs of operations.

To limit the complexity, it is proposed to limit the maximum complexity of the circuit. In the example, this maximum can be schemes with three operation signs.

For each difficulty level, it is easy to write out all the schemes (test template instances) of a given difficulty level and a test exercise for this model can now be defined as a set of test tasks with a given distribution of difficulty levels.

### 2.2 Example 2. Mixing liquids' model.

The elementary process of mixing liquids can be described as follows: two vessels contain liquids, the physical characteristics of which are given by the heat capacities c, the temperatures t of the masses m. It is required to determine the physical characteristics of their mixture contained in the third vessel (Fig. 3).

As in the first example, the formal model of a physical quantity is a five (name, unit, range of values, step of change, value).

In the definition of a fluid mixing model, formal models of such physical quantities are used: heat capacity, temperature, mass, heat.

Formal fluid model: heat capacity c, temperature t.

*The formal model of a vessel with a liquid* is an element of the scheme: a model of a liquid, mass *m*.

The formal model of mixing liquids uses formal models of vessels with liquids  $(c_1, t_1, m_1), (c_2, t_2, m_2), (c_3, t_3, m_3).$ 

The testing system should verify the application of the following physical laws:

$$m = m_1 + m_2,$$
 (4)

$$Q = Q_1 + Q_2$$
 (heat balance equation), (5)

where

$$Q_1 = c_1 m_1(t_1 - t), Q_2 = c_2 m_2(t_2 - t)$$
(6)

$$c = \frac{c_1 m_1 + c_2 m_2}{m_1 + m_2} \tag{7}$$

$$t = \frac{c_1 m_1 t_1 + c_2 m_2 t_2}{c_1 m_1 + c_2 m_2} \tag{8}$$

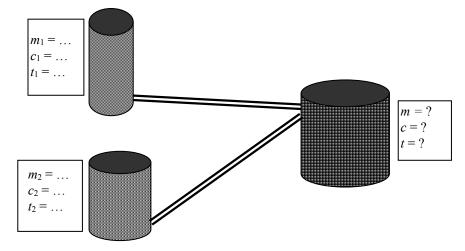


Fig. 3. Test task on the topic of "Mixing liquids".

### Formal model

Physical quantities: Mass = (Real, kg, [1; 100], 1), HCap = (Real, J / kg  $\cdot$  ° C, [0.2; 20], 0.2), Tem = (Real, ° C, [10; 100], 1), Element of the model L = (Mass, HCap, Tem), Schematic construction operation:  $L = L_1 + L_2$ , or  $L = Mixt(L_1, L_2)$  **Examples of test tasks:** Example 1 Given:  $L_1(m_1, c_1, t_1), L_2(m_2, c_2, t_2).$  $L_1 = ((2 \text{ kg}), (4.19 \cdot 103 \text{ J / kg ° C}), (15 ° \text{ C})) - \text{H2O};$   $L_2 = ((1 \text{ kg}), (2.39 \cdot 103 \text{ J} / \text{kg} \circ \text{C}), (5 \circ \text{C})) - \text{C2H5OH};$  *Find*  L = (m, t).Example 2 *Given*:  $L_1, L;$ *Find*  $L_2$ .

**Method of test task generating.** As in example 1, the installation of mixing liquids can be considered as a sequential mixing scheme. If the operation of mixing two liquids is denoted by a symbol "+", i.e. the scheme of example 2 is given by the formula  $L = L_1 + L_2$ , schemes are possible  $L = (L_1 + L_2) + L_3$ ,  $L = (L_1 + L_2) + (L_3 + L_4)$ .

In this case, the test task is generated:

- selection of a test construction template;

- random selection of the type of template from the list of liquids;

- random test generation by generating data from the *Given* section and the names of the *Find* section.

**Test complexity.** As in example 1, the complexity of the test task is determined by the complexity of the scheme - the design template, i.e. the amount of mixed liquids. You can mix 2, 3 and even 4 substances according to the following schemes:

(1, 2), (1, 2, 3), ((1, 2), 3), (((1, 2), 3), 4), ((1, 2), (3, 4))

The complexity of the construction template makes sense to specify a discrete probability distribution. For example:

Scheme	(1, 2)	(1, 2, 3)	((1, 2), 3)	(((1, 2), 3), 4)	((1, 2), (3, 4))
Probability	0.5	0.2	0.2	0.05	0.05

Table 1. Test task template.

In addition, for this particular example, the complexity of the test task is influenced by the fact whether two identical or two different liquids mix with different heat capacities. As in example 1, the complexity of the test task is determined by the complexity of the construction template, the amount of data in the *Find* section, taking into account the number of different mixable liquids.

# **3** Formal model of physical knowledge testing system

### 3.1 The course of the discipline "Physics"

is represented by a hierarchy of dependence of the subject (training) modules (structural-logical scheme of the discipline).

3.1.1 In the training module, a list of the intrinsic physical quantities used in it is defined (Fig. 4). Each physical quantity is defined by a five: *(name, basic physical dimension, type of value, range of values, step of change)* 

Training module "Name" Physical value

#### **Physical value**

Fig. 4. Model of the training module.

3.1.2 Relationship dependence determines the sequence of study of relevant topics of the training course (Fig. 5).

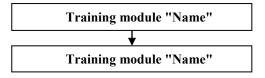


Fig. 5. Model of hierarchical relations of dependencies of training modules.

3.1.3 In addition to the attributes described in 4.1.1, the training module is represented by a set of formal models of physical phenomena and processes (Fig. 6).

#### **Training module**

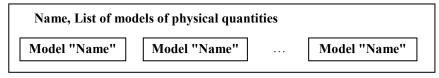


Fig. 6. Model of a subject module as a set of models of phenomena, processes.

### 3.2 The formal model of a physical phenomenon, a process

The formal model of a physical phenomenon, a process is a scheme (formula) in the signature of the operations of constructing schemes, a list of types of elements of the scheme, a set of interpreters of operations for constructing a scheme (Fig. 7).

# *Physical value* is defined as the five attributes:

(Name, type of value, physical dimensionality, range of values, step of change)

The element of the scheme (primitive scheme) is a set of physical quantities. Thus, if  $\sigma$  is an element of the, and  $a_1, ..., a_l$  are physical quantities, then  $\sigma = (a_1, ..., a_l)$ . The schema element has its own type:  $Type(\sigma) = (type(a_1), ..., type(a_l))$ . If  $x_1, ..., x_l$  – variables are physical values, then a structured variable  $\sigma = (x_1, ..., x_l)$  is a scheme element.

Scheme construction operations. Each scheme is composed of scheme elements and other schemes using construction operations. The set of names and arity of construction operations is called the schema signature:  $OP_{\Sigma} = \langle f_1^{(k_1)}, \dots, f_m^{(k_m)} \rangle$ .

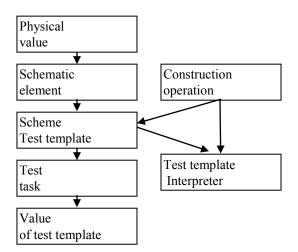


Fig. 7. The system of categories for the formal model of a physical phenomenon, process.

The construction operations are, generally speaking, multi-sorted. If  $y = f_i(x_1, ..., x_{k_i}) \in OP_{\Sigma}$ , then the types of variables  $y, x_1, ..., x_{k_i}$  belong  $TP_{\Sigma}$  and, generally speaking, are different. The types of operations for constructing schemes are determined by the types of variables y.

*Test schemes (templates)* are constructed from the elements of schemes and other schemes using construction operations:  $\Sigma_{f_i} = f_i(\sigma_1, ..., \sigma_{k_i}) \in$ . Thus, the test template  $\Sigma$  is the formula in the signature  $OP_{\Sigma}$ . All the arguments of the schema  $\Sigma$  are the variables — the schema elements. Thus, the template (formula scheme) has the form  $\Sigma(x_1,...,x_l)$ .

The complexity of the scheme is the number of symbols of functions in the formula  $\Sigma$ . Since the value of any function for a given variable requires computation, the concept of the computational complexity of a scheme is closely related to the concept of the complexity of a scheme.

The test task is obtained as a schema specification with the values of some subset of the set  $(x_1,...,x_l)$ . The test task has the form  $\Sigma(a_1,...,a_k,x_{k+1},...,x_l)$ . Values  $a_1,...,a_k$ form *Given* section of the test task. In the simple case, *Find* section contains the names of one or more variables from the set  $(x_{k+1},...,x_l)$ .

The test template interpreter is a software function that calculates the value of a scheme given the values of its arguments. Let  $\sigma(x_1,...,x_l)$  be an any element of the scheme. Variables  $x_1,...,x_l$  are related by relationships  $\Phi(x_1,...,x_l)$  – physical laws whose knowledge is being tested. Therefore, all values  $(x_{k+1},...,x_l)$  of the *Find* set variables can be calculated by values  $a_1,...,a_k$ . Thus, for any type scheme  $f(\Sigma_1,...,\Sigma_l)$ , its value  $y = Value(f(\Sigma_1,...,\Sigma_l))$  can be calculated from bottom to top:

 $a_1 = Value(\Sigma_1), \dots, a_l = Value(\Sigma_l), y = Int f(a_1, \dots, a_l),$ 

where *Intf* is the interpreter of signature operation *f*.

Thus, the interpreter of the scheme calculates not only the values of the variables of the set  $(x_{k+1}, \ldots, x_l)$ , but also all the physical values of any subscheme of the test template.

Thus, the number of test case variables includes not only variables  $x_1, ..., x_l$ , but also all intermediate variables of a form  $y = f(x_1, ..., x_l)$  for any subschem of a type  $f(\Sigma_1, ..., \Sigma_l)$ .

# 4 Methods for generating and validating test items

#### 4.1 Test template selection

In principle, there are two approaches to solving the problem of choosing a test pattern. First, for each model of the physical process, the phenomena are created manually by the table of templates (see tab.2), which are stored in the database. With this approach, together with the table of templates, it makes sense to keep a table of the distribution of probabilities for choosing this template. Secondly, it is possible to implement an algorithm for automatically generating circuits of a given complexity. In this case, first with a given probability, the level of complexity is selected, and then a circuit of this complexity is generated. Each of these approaches has its own advantages and disadvantages. In the first case, you need to spend a lot of time filling the database with template tables. In the second case, time is spent on the implementation of an algorithm for automatically generating a circuit of a given complexity.

Another technological challenge is to implement a graphic image of the physical process, the phenomenon. The test window contains the fields Picture, *Given*, *Find*. The Picture field displays the scheme in which variables are marked. Fields *Given* and *Find* are filled in at the stage of test task generation. The uniqueness of the model leads to the fact that each drawing scheme is programmed separately.

#### 4.2 Test generation

Let be the test pattern template. Consider a set of schemes of a given circuit containing all elements and all composite schemes  $\Sigma = \langle \Sigma_1 = \sigma_1, ..., \Sigma_k = \sigma_k, \Sigma_{k+1} = \sigma_l \rangle$ . Each element of this set depends on the variables - physical values:  $\Sigma_1(x_{11},...,x_{1n}),...,$  $\Sigma_l(x_{l1},...,x_{ln})$ . As already mentioned, for each subscheme  $\Sigma_j(x_{j1},...,x_{jn})$  the following property takes place: if the values of any *m* variables from the set  $(x_{j1},...,x_{jn})$  are known, the values of the other n - m variables can be calculated. Therefore, the process of calculating the data values of a particular test task is carried out from the bottom up to the structure of the circuit.

Our approach to growing the practical knowledge testing system is as follows:

- 1. On the basis of algorithmic analysis, define a specific subject area as a set of models of physical quantities, models of elements of schemes and schemes as formulas in a given signature.
- 2. Develop a unified general model of this class (test pattern design), as well as test pattern templates as an implementation of the test pattern design

- 3. Implement the algorithms for generating conditions and responses to this model pattern, thereby determining the algorithm for generating a test task and verifying it
- Develop CASE-technology for describing subclasses of test items based on a single common model for users of the computer-aided mathematics system for educational purposes.
- 5. Develop common mechanisms for storing and invoking algorithms for generating specific test items.
- 6. Develop methods for generating and validating test items.

# 5 Test template selection

IMS QTI standard specifications allow you to develop a model and create software for testing physical knowledge and skills in the DLS. For this purpose, specifications of Adaptive Items and Item Templates [16] can be used.

#### 5.1 Adaptive test items

The software module for creating and executing test tasks for testing physical knowledge and skills can be based on the technology of developing adaptive tests. The IMS QTI version 2.2 specification provides support for adaptive tests in distance learning systems. In accordance with this standard, adaptive test questions may contain software object modules that provide interactive user interaction with them. For such questions, we introduce into consideration a new type of test question — an object type, and tests containing object-type questions will be called adaptive (laboratory) tests. Below is the specification and schema of the attributes of the object type question of the adaptive (laboratory) test (Fig. 8) [19].

Each type of question separately has its own characteristics in the specification. This is due to differences in the parameters of these types.

The response of the test is processed in the Response Processing module. Evaluation of the answer in the module can occur in two ways: a differentiated assessment of the entire question and the accumulation of evaluation of the response options.

According to the IMS QTI Specification, when working out questions of the adaptive test, there is a feedback with the test person, determining the response adjustment at each stage and thus forming the variability of the response. In this question there may be additional parameters that are not specified by the standard. An example of the implementation of such an object question is a controlled interactive Flash animation in which a specific task is programmed [19]:

- module initialization with some input parameters,
- interactive game situation in which the tested person participates,
- output as a result of the action of the test.

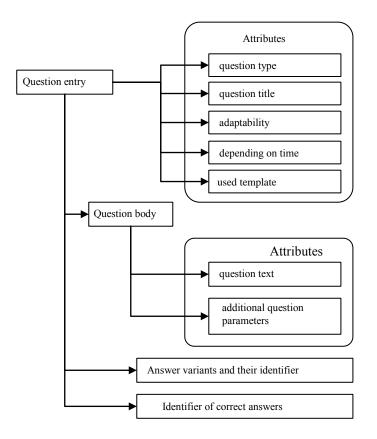


Fig. 8. The question attributes scheme of the object type of the adaptive test and its specification.

The result of answering a question of a laboratory (adaptive) type can be determined in the test passing object, taking into account the value of the maximum mark for correct passing the test, and used in automatic (software) assessment. Alternatively, the score can be determined (changed) by the tutor during the check.

Due to the lack of a separate specification of the IMS standard for laboratory (adaptive) tests, we consider the method of their simulation, which consists of the following. Two object modules are developed that are interconnected by a special format data transfer interface. The first module (the test passing object) is designed to pass a laboratory (adaptive) type test, and the second (answer display object) is intended to reflect the test passing results during the test. The scheme of interaction between the objects of the testing module and the database is shown in Fig. 9.

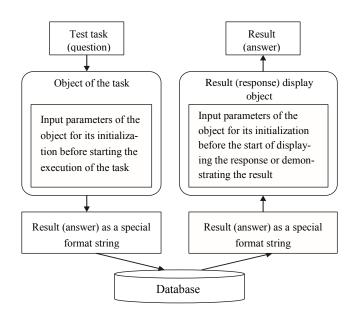


Fig. 9. The scheme of interaction between the objects of the test module and database.

Object modules can be Java applets, COM objects (for example, Adobe Flash) or HTML 5 program objects, which upon initialization receive the values of input parameters (attributes) and are set up in a working state. For an object passing the test, such parameters may be the value of the maximum mark for correct passing the test, the value of the time limit for passing the test, other values of the object initialization parameters; for the response display object, the values of the input parameters that ensure the initial state of the object, corresponding to the final state of the test object.

Thus, the combination of two object modules forms a closed system for passing, checking and evaluating tests of the adaptive (laboratory) type. The proposed method of creating adaptive (laboratory) type tests meets the specifications of the IMS standard and allows packaging for portability in other distance learning systems.

### 5.2 Template type test items

At developing of the testing module, we will understand the testing process as a sequence of calls of operators with a certain behavior in the context of a learning model.

The process of automatic test assembly in the DLS testing module is a workflow in which certain logic is implemented with the possibility of automation. It is important to begin by defining an abstract model that contains all the necessary data describing the purpose of training and explaining the correlation between objects of specific instances of this model in order to automate the compilation of test questions as a workflow.

Suppose Item Templates is an instance of a test task generated by this template with the automatic test generation process described above.

Automatic generation of Item Templates will help to significantly simplify the preparation of practical tasks, reducing the time for their creation and ensuring the unique nature of each generated task. It will also provide an opportunity to automate the evaluation of test results without the negative impact of the human factor.

We define the test as a set of learning tasks set and test settings:  $ST = \langle S(S_1, ..., S_L), T(T_1, ..., T_N) \rangle$ , where  $S(S_1, ..., S_L) - a$  set of test settings, defined by a user (a test compiler), and  $T(T_1, ..., T_N) - a$  set of learning tasks. To build the abstract model of the learning task T we use its signature of following view:

#### $T = \langle M, P, Q \rangle$

where M – a set of the task input parameters, and is described as  $M(X_1, ..., X_n)$ , where  $X_1, ..., X_n$  – the task parameters; P – a set of the task conditions, described as  $P(P_1, ..., P_k)$ , where  $P_1, ..., P_k$  – task conditions; Q – the task result model, described as  $Q(Q_1, ..., Q_m)$ , where  $Q_1, ..., Q_m$  – sets of metadata, defining the results of the task. We introduce the following software modules to describe the Item Templates object: LrnTaskTemplate (abbreviated from Learning Task Template) for T; Lrn-TaskModel (abbreviated from Learning Task Model) for M; LrnTaskValidator (abbreviated from Learning Task Validator) for P; LrnTaskModel for Q.

The abstract generator model is a generalized model of math tests. The scheme of interrelations of the objects described above is shown in Figure 10.

According to this scheme, ISmartTestModule is a library of learning task generator templates for compiling a test task. The template library of the instructional task generator module is a program class (LrnTaskTemplate), which contains all the functions necessary for building learning task instances, validating and solving them using a named model. The functions of the task generator template use the common (TestSettings) and local (LrnTaskTemplateSettings) user-defined settings (the test compiler) as input parameters. These settings can provide the template with parameters for generating and checking specified learning tasks in the form of: instructions for using preferred question types [17], the number of generated answers, fine-tuning the module with the inclusion / exclusion of certain templates, input parameter fields, etc. Number of such settings may differ depending on the module and its templates. It is expected that the test compiler will adjust the parameters when adding Item Templates to the system. The learning task model (LrnTaskModel) is a program class for storing the input parameters of the learning task. Depending on the goals, the model of the training task may contain extended attributes necessary to describe the parameters of the task. The number of parameters is not limited by the system. The next element of the learning task generator is the LrnTaskValidator class. The purpose of this program class is to check the compliance of the model instance with the task settings. The solver (LrnTaskSolver) is the last element of the template. The solver's task is to solve the specified task (to achieve results) using the LrnTaskModel parameters with the conditions specified in the LrnTaskValidator. The results of the solver are formal expressions in the signature of a specific subject field, which can be defined in the heir to the LrnTaskResponseModel class. In particular, there can be atomic expressions - numbers, strings, dates.

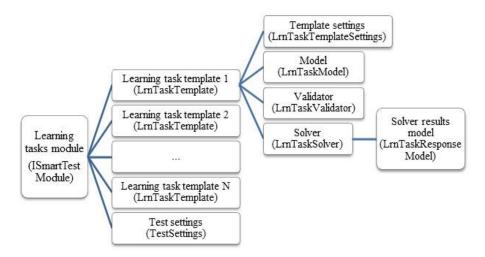


Fig. 10. Scheme of interrelations of elements of model Item Objects.

# 6 Conclusions and Outlooks

The approach to building a system for practical physical knowledge testing described in this paper allows you to:

- 1. Quickly and correctly draw up a system of tests for practical knowledge testing on a given topic, in accordance with a given level of computational complexity of this system.
- 2. Automate the procedure for generating a sufficiently large number of specific test tasks based on a single template.
- 3. Solve the problem of automatically checking the correctness of the answer and the progress of the test solution.
- 4. Solve the test task step by step, applying symbolic expressions and checking at each step the correctness of their use.
- 5. According to the proposed model, develop software for physical knowledge and skills testing.

The proposed approach to the process of physical knowledge and skills testing can be used to build a model and develop software for a testing module in a distance learning system in accordance with the requirements of international standards IMS and SCORM.

### References

- 1. Lvov M.S. The basic principles of the construction of pedagogical software support practical training. Control Systems and Computers, № 6, P. 70-75 (2006)
- Lvov M.S. Designing a logical inference as a step-by-step solution of problems in mathematical systems for educational purposes. Control Systems and Computers, № 1, P. 25-32 (2008)
- Lvov M.S. The concept of information support for the educational process and its implementation in pedagogical software environments. Control Systems and Computers, № 2, P. 52-57 (2009)
- 4. Lvov M.S. The concept, architecture and functionality of a flexible distributed program environment for educational purposes for secondary schools. Workplace of the Methodist. Control Systems and Computers, № 6, P. 71-78 (2009)
- Lvov M.S. Distributed software environments for educational purposes. Subsystem of educational process management. Control Systems and Computers, № 1, P. 66 - 71 (2010)
- Lvov M.S. Synthesis of interpreters of algebraic operations in extensions of multi-sorted algebras. Bulletin of Kharkiv National University (Series "Mathematical Modeling - Information Technologies - Automated Control Systems"). No. 847, P. 221-238 (2009)
- Lvov M.S. Mathematical Models and Methods of Supporting the Settlement of Educational Problems in Analytical Geometry. Artificial Intelligence. No. 1, P. 86-92 (2010)
- 8. Lvov M.S.Trigonometric computations in computer mathematics systems for educational purposes. Artificial Intelligence, № 4, P. 417 423 (2011)
- 9. Lvov M.S. On one approach to building a system for testing geometric knowledge. Journal of Information Technologies in Education. Kherson, № 18, P. 42 50 (2014)
- Lvov M.S., Shishko L.S., Chernenko I.E., MS On the organization of control of knowledge as a feedback in systems of computer mathematics of educational assignment. Journal of Information Technologies in Education. Kherson. No. 16, P. 29 - 42 (2013)
- Lvov M.S. Mathematical models of subject areas in computer mathematics systems for educational purposes. Bulletin Harkiv national university. (Series "Mathematical modeling. Information technologies. Automated control systems"). № 987. P. 46 – 60 (2011)
- 12. Lvov M.S. Mathematical tests in computer math systems for educational purposes. Control Systems and Computers, №6. P. 60 67 (2011)
- Lvov M.S. Intellectual properties of computer mathematics systems for educational purposes and methods for their implementation. Artificial Intelligence, № 2, P. 45 52 (2011)
- Peter Brusilovsky and Philip Miller. Web-based Testing for Distance Education. In: Web-Net 99 World Conference on the WWW and Internet Proceedings (Honolulu, Hawaii, October 24-30, 1999), 7p.
- 15. Available at: http://www.contrib.andrew.cmu.edu/~plb/papers/WebNet99.html
- H. Kravtsov, D. Kravtsov. Knowledge Control Model of Distance Learning System on IMS Standard. Innovative Techniques in Instruction Technology, E-learning, Eassessment, and Education. Springer. P. 195 – 198 (2008)
- 17. IMS Question & Test Interoperability (QTI) Specification. Version 2.2. Available at: http://www.imsglobal.org/question/index.html#version2.2
- 18. GRE(R) Physics Test Practice Book. Educational Test Service (2017). Available at: https://www.ets.org/s/gre/pdf/practice\_book\_physics.pdf
- Physics, grades 7-9. Educational program for general educational institutions. Available at: https://mon.gov.ua/ua/osvita/zagalna-serednya-osvita/navchalni-programi Kravtsov H., Kravtsov D. Adaptive and object tests in the model of knowledge control according to the IMS standard. Control Systems and Computers, № 1, P. 42 - 48 (2008)