Innovative Educational Technologies, Tools and Methods for E-learning
Scientific Editor Eugenia Smyrnova-Trybulska
"E-learning", 12, Katowice–Cieszyn 2020, pp. 77–87
DOI: 10.34916/el.2020.12.07

CONNECTION BETWEEN ONTOUML AND KNOWLEDGE REPRESENTATION MODEL OF STUDENTS' ACTIVITIES

David Buchtela[1], **Dana Vynikarová**[2]
[1]Centre of Business Informatics
Faculty of Information Technology, Czech Technical University in Prague
Thakurova 9, 160 00 Prague 6, the Czech Republic
[2]Faculty of Economics and Management,
Czech University of Life Sciences Prague
Kamycka 129, 165 00 Prague 6, the Czech Republic
[1]david.buchtela@fit.cvut.cz, [2]vynikarova@pef.czu.cz
ORCID [1]0000-0002-3564-9198, [2]0000-0001-8955-6002

Abstract: *In every focused system, e.g. the Learning Management System (LMS) Moodle, it is possible to select relevant entities (students, teachers, study resources, assessment, test and other activities) and their relations (associations). A conceptual model in OntoUML is suitable for the entities representation. It is possible to feel a knowledge decision process as a non-determinist finite automaton where entity state transitions are inspected. A way of entity state transition (needed data and conditions) is represented by guideline (procedural) knowledge representation model (like as GLIKREM). This paper aims to describe the possibilities of the conceptual model of the focused system designed in OntoUML and the Guideline Knowledge Representation Model (GLIKREM) for the Knowledge Representation Model of Students' Activities (KRMSA) based on knowledge and models of students' activities in a Moodle system. This article describes a link between OntoUML as a conceptual model and GLIKREM as a procedural knowledge model base with the aid of the main components of both models.*

Keywords: knowledge representation model, conceptual model, students' activities, decision process, OntoUML, Moodle.

INTRODUCTION

At present, the distance form of education is becoming more and more important. More and more courses are being taught with the support of electronic systems (e.g. LMS

Moodle) to provide students with new forms of electronic study materials, but also to provide feedback in the form of attendance records, records of studied materials and evaluation of individual projects and tests.

As the number of electronic courses grows, so it increases a suitable model requirement for representing the student¨s way through such a course. As the courses change, so does the role of the teacher toward the course guide and consultant (Koķe, 2003). A suitable model for the student's way through the course is a valuable tool for the teacher where the implementation of the model in the current LMS can automate many activities.

One of the usable models is GLIKREM. This model is originally intended to represent medical guidelines (Peleška, 2005) but is suitable for representing any procedural knowledge. The second suitable model is OntoUML, which is an extension of UML2 towards ontologies. OntoUML is primarily intended for (ontologically) a more accurate description of the investigated domain, especially data types and relationships between them. However, the resulting model in OntoUML is relatively easily convertible to an implementation model (Rybola, 2016) and can thus be the cornerstone of applications in the background of the primary LMS or applications of subject domain ontologies (Stoyanova-Doycheva, 2019). Knowledge Representation Model of Students' Activities described in this article uses elements of both mentioned models.

1. METHODS

1.1. OntoUML

OntoUML is an example of a conceptual modelling language which has been designed to comply with the ontological distinctions and axiomatic theories put forth by a theoretically well-grounded Foundational Ontology (Guizzardi, 2005). The conceptual model in OntoUML can then be transformed into various implementation models, typically relational or object (Rybola, 2017).

Types (classes) in OntoUML are based on the UML2 concept of classes as a description of common properties shared by certain entities (instances of the class). Attributes represent more or less intrinsic properties shared by instances of the class. Class instances already contain specific attribute values.

Types can be in a specialisation or association relationship with each other. Specialisation defines the taxonomic structure of types in which all attributes of a class are inherited through a chain of specialisation. Classes sharing a common supertype can be grouped in a so-called generalisation set. Two meta-attributes can be used a more robust semantics to a generalisation set, namely, the complete and disjoint meta-attributes.

- ***Complete*** – If a generaliz = sation set is complete, the subtype exhausts the instances of the common direct supertype. There is no instance of the supertype, which is not an instance of one of the subtypes participating in the generalisation.
- ***Disjoint*** – If a generalisation set is disjoint if all the subtypes participating in the generalisation are mutually exclusive. The intersection between any of these subtypes is always empty.

Associations are generally n-ary relations (but mostly binary) those bind entities together. Associations in OntoUML have the same meaning and notation as UML2.

The association is specified by an (optional) role name and (mandatory) multiplicities on both sides of the association. An arrow indicates the reading direction supplements the association. Specific multiplicities are determined by conceptual analysis of the problem domain.

Identity. Identity plays a crucial role in OntoUML. The principle of identity tells how to identify an object (instance of some type) during its entire existence, regardless of its (arbitrary) changes. It is, therefore, identity in terms of our perception, i.e. how we can clearly distinguish two objects from each other. This concept is fundamentally different from the usual identity in the object concept (object ID). Based on the (ontological) identity, we then define two main categories of types:

- ***Sortal type*** – Sortal type provides categorisation and has its own (ontological) identity.
- ***Non-sortal type*** – Non-sortal types have not an (ontological) identity and therefore represent abstract concepts in terms of our perception. We use them to categorize sortal types according to various properties and relationships. In terms of implementation, of course, each sortal type needs an identity, e.g. object ID.

For further categorisation of object types, it is necessary to explain the basic concepts of the so-called modal logic as an extension of predicate logic. From predicate logic, modal logic uses existential (there is at least one element such that…) and universal quantifiers (for all elements…).

Also, modal logic introduces the concept of the world. To put it simply, the world in modal logic represents a specific configuration of reality in time or space. Two more quantifiers are being introduced for worlds:

- Quantifier □ – In all worlds…
- Quantifier ◊ – In some world (at least in one)…

Modal logic makes it possible to distinguish between classifications and invariant relations and those, which may change depending on the context. The categorisation of types also follows from modal logic:

- ***Rigid type*** – Type T is rigid for each instance of x just when x is necessarily (in all worlds) an instance of type T. If x is an instance of type T in some world, then x must be an instance of type T in every possible world.
- ***Anti-rigid type*** – Type T is anti-rigid for every instance of x just when it is possible (in some world) that x does not have to be an instance of type T. If x is an instance of type T in some world, then there may be some other world where x is not an instance of type T.

The categories of all types in OntoUML are based on the (ontological) identity principle and modal logic and are listed in Table 1.

1.1.1. Sortal types

The <kind> and <subkind> types simply correspond to classes and subclasses according to the UML2 concept. There is a subtle difference between <kind> (type) and <subkind> (subtype) that UML2 does not distinguish, but OntoUML does. The <kind> type has and provides an ontological identity, while the <subkind> type does not have its own identity; it takes it from the <kind> type. In OntoUML, each sortal type can have only one (ontological) identity, so type <kind> cannot be a subtype of

another type <kind>. In the resulting model (see Figure 5), the type is <kind> Person (where the identity can be, for example, a personal number or login) or Course (the identity is the course code).
The <role> type is subject to a so-called relational dependency. The type T is relationally dependent on the type P via relation R just when for each instance x of the type T there exists an instance y of the type P such that x and y are related via R. All instances of a <role> type are of the same <kind> type, e.g., all Students (<role>) are Person (<kind>). A <role> type cannot be a supertype of any rigid type. In the resulting model (see Figure 5), the type <role> is a Student belonging to the type (<kind>) Person, who studies some type (<kind>) Course, and his role is thus valid in this "world" (course). Similarly, the role Teacher has tied to the "world" of the Course type that this teacher teaches.
The <phase> type is defined as an anti-rigid specialisation of the <kind> or <subkind> type such that the specialisation condition is intrinsic. Phases are always defined in a so-called phase partition. The phase partitions are always disjoint and complete generalisation sets. In the resulting model (see Figure 5), the type <phase> is the passed or non-passed of some of the course activities (types <kind> Chapter, Project, Test or Exercise). Which phase (type <phase>) the respective type <kind> enters depends on the fulfilment of some internal condition (e.g. obtaining a specified minimum rating). Simply put, the <phase> type can be thought of as the state of the corresponding <kind> type.

Table 1

Categories of Types in OntoUML

Category of Type	Supplies an identity	Has an identity	Rigidity	Dependence
SORTAL				
<kind>	+	+	+	–
<subkind>	–	+	+	–
<role>	–	+	– (anti)	+
<phase>	–	+	– (anti)	–
NON-SORTAL				
<category>	–	–	+	–
<roleMixin>	–	–	– (anti)	+
<mixin>	–	–	– (semi)	–

S o u r c e: Own work based on Guizzardi, 2005.

1.1.2. Non-sortal types

The <category> type is a rigid non-sortal representing the necessary (in the modal sense) property instances of different types. Categories form the uppermost layer of object types of ontologies and are therefore always defined as abstract classes, like all non-sortal types. That means that all <category> types must have some sort of sortal subtype (usually <kind>). The <kind> type can be a subtype of multiple <category> types, and the <category> type can be a supertype of multiple <kind> types. In this case,

all <kind> types in the category must be disjoint (form a disjoint generalisation set). Otherwise, instances of their intersection would inherit multiple (ontological) identities. The <roleMixin> type is an anti-rigid and relationally dependent non-sortal representing (in the modal sense) the properties of instances of different types. The <roleMixin> types are used to categorise the instance of the <role> types. However, the <roleMixin> type cannot be a <category> supertype, because no anti-rigid non-sortal can be a supertype of a rigid one.
The <mixin> type is a semi-rigid non-sortal representing properties that are (in a modal sense) necessary for some instances but possible for some other instances. Each <mixin> type must be a rigid type supertype (typically <kind>) and an anti-rigid type supertype (typically <phase>). Since all <kind> types are disjoint, all subtypes of <mixin> type form a disjoint generalisation set.

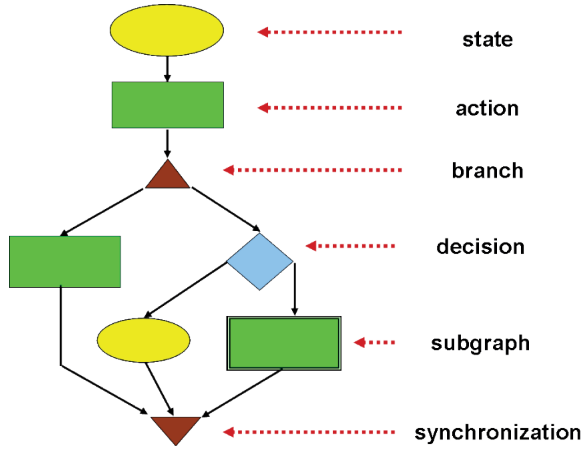
1.2. GLIKREM

The Guidelines Knowledge Representation Model (GLIKREM) is based on a Guidelines Interchange Format model which was published in a GLIF3.5 specification (Boxwala et al., 2004). GLIKREM contains some changes and extensions to the definition and implementation of the original GLIF model, which allow more accurate modelling of procedural knowledge as a graphical model which describes a process structure of the decision algorithm (Buchtela et al., 2010). GLIKREM also includes a parameter model, which serves as an interface between the graphical model and real data.

1.2.1. Main parts (steps) of the graph

The GLIKREM created in a construction stage is an oriented graph (see Figure 1) which is composed of five main parts (steps):

- ***Action steps*** specify actions that are to be performed. It can be a study of some resource, submission of the completed task, performance of the test etc. Action step also may name sub-guidelines (subgraph), which provide detail for the action.



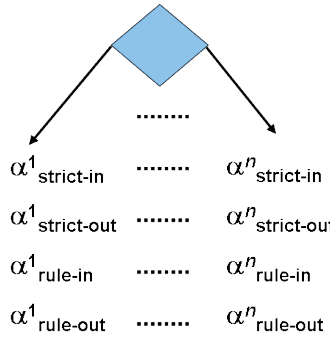
F i g u r e 1. Main parts (steps) of GLIKREM
S o u r c e: Own work.

- ***Decision steps*** are used for conditional branching. This step is used when branching is determined by evaluation of defined logical criteria based on data items. If the decision cannot be made automatically, the user can select himself the follow-up part of the graph.
- ***Branch and synchronisation steps*** enable concurrence in the model. Guideline steps that follow the branch step can be performed concurrently. Branches with root in a branch step eventually converge in a synchronisation step. In this step, all branches are synchronized after evaluation of the synchronizing condition.
- ***State steps*** characterise surveyed object states after the execution of the previous steps or at the beginning of the model.

1.2.2. Decision criteria

Each decision step specifies four criteria of condition for each decision option (see Figure 2). The subsequent flow of the model is automatically or manually chosen based on the evaluation of these criteria:

- ***Strict-in*** – if a *strict-in* is true, the control flows to the guideline step that is specified by that decision option's destination.
- ***Strict-out*** – if a *strict-out* is true, the decision option's destination is forbidden. The *strict-out* can be but don't have to be opposite of *strict-in*.
- ***Rule-in*** – if a *rule-in* is true, it is only recommended to flow to the guideline step that is specified by that decision option's destination. The user should select himself one of the next steps with positive *rule-in*.
- ***Rule-out*** – if a *rule-out* is true, the decision option's destination is not recommended, but it is not forbidden. The user shouldn't select one of the next steps with positive *rule-out*.



F i g u r e 2. Decision criteria in a decision step

S o u r c e: Own work.

The *strict-out* criterion is evaluated at first. If the *strict-out* criterion (of some option) is evaluated as true the rest of the criteria (of this option) is not assessed. This option is forbidden. In the opposite case, the *strict-in* criterion is evaluated. If both of *strict-in* and *strict-out* criteria are false, the *rule-in* and *rule-out* criteria are evaluated. The ranking of *rule-ins* and *rule-outs* (of all option's criteria) is left to the users who may use their clinical judgement or develop their ranking schemes.

1.2.3. Criteria evaluation

When evaluating the criteria (*strict-in*, *strict-out*, *rule-in*, *rule-out*), it often happens that input parameter values are not known. Therefore, the criteria are evaluated in three-value logic. The logic formulas contain variables from a model of parameters (Veselý, 2006) and logic or relational operands.

If these rules are thoroughly applied, the user can insert missing data when it is necessary. If a *strict-in* criterion of some option (destination) is true, the evaluation of other option criteria is not needed. The amount of essential data is dependent on the order of single option evaluations. Therefore, it is necessary to set an order of assessment, i.e. to set a priority of decision options. A specialist chooses the priority of each option.

2. RESULTS AND DISCUSSION

The students' way through an electronic course (e.g. realized in LMS Moodle) can be described using the Knowledge Representation Model of Students' Activity (KRMSA). The KRMSA design consists of both a graphical model (GLIKREM) of the student's progress and its transformation into the OntoUML model. A sample KRMSA design is made for a model situation of a hypothetic course in LMS Moodle.

2.1. Model situation

The subject of the research is a full-time student, the aim of which is to obtain credit from a hypothetic course for one semester. To complete the course (achieve the goal), the student must meet the following conditions (C1 – C4):

- C1 – To study ten chapters with theory related to the subject. There are several control questions at the end of each chapter that the student must answer correctly (he can answer in several attempts).
- C2 – To adequate attend at exercises. It is necessary to get at least 70% participation in exercises.
- C3 – To submit an individual project. The student submits the project itself on a given topic. The student must obtain a grade of at least 70%. It is possible to make one repair in a separate project.
- C4 – To pass a credit test. The student achieves a control test of the discussed issues. Successful completion of the test means obtaining at least 70% of the possible evaluation. The test can be passed in a maximum of two attempts.

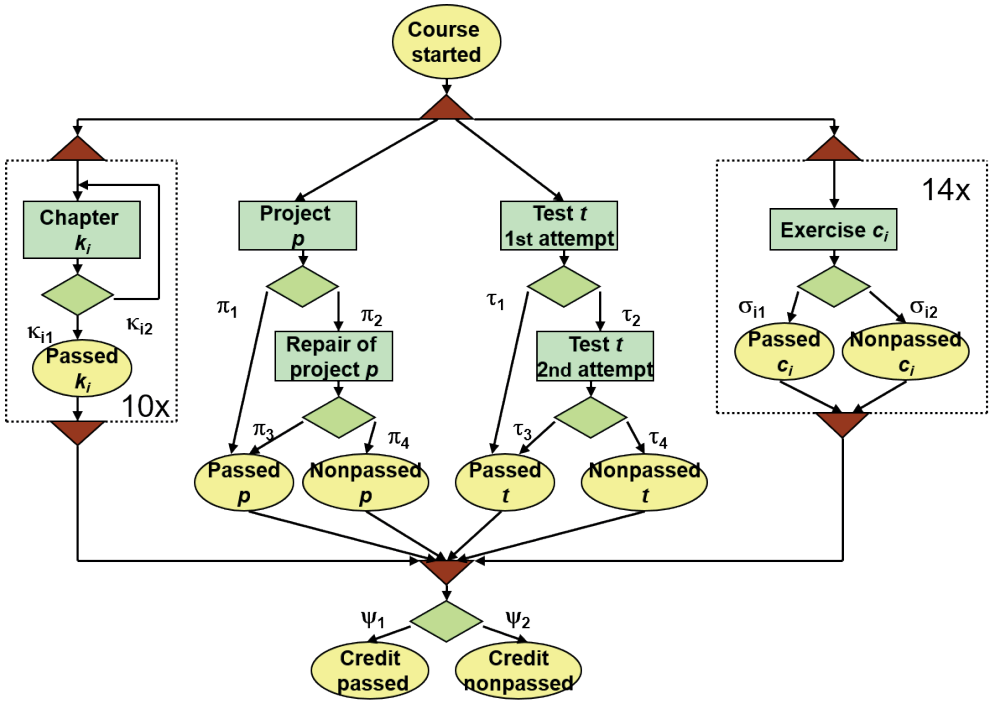
The course of study, i.e. attendance and results of partial conditions, is recorded in the system for the support of electronic education (Moodle). In the same system, the student has the necessary study materials (theoretical chapters) and feedback tools (project submission and electronic test).

2.2. GLIKREM of model situation

The model situation will be shown in GLIKREM as four parallel branches, one branch for each condition C1 – C4 (see Figure 3). The first branch represents the study of ten chapters (k_i), the second branch of completing 14 exercises (c_i), the third branch of achieving a separate project (p) and the last branch of passing a credit test (t). The dot-

ted lines indicate the repetition of the same structure, i.e. ten times the study of the chapter and 14 times the completion of the exercise.

- C1 – Study of a chapter (k_i). The student repeats the study of the chapter k_i until he answers the control questions correctly, i.e. until the strict-in criterion of the branch κ_{i1} is true.
- C2 – Completion of exercises (c_i). The student either participates in each exercise (true strict-in criterion σ_{i1}) or does not attend (true strict-in criterion σ_{i2}). The teacher records attendance (non-attendance).
- C3 – Completion of the project (p). The student completes the project if the strict-in edge criterion π_1 or π_3 is met. Both strict-in criteria are defined by the condition evaluation (p) ≥ 0.7. The strict-in criteria of branches π_2 and π_4 are a negation of the strict-in criteria of edges π_1 and π_3.
- C4 – Passing the test t. The student successfully passes the credit test if the strict-in edge criterion τ_1 or τ_3 is met. Both strict-in criteria are defined by the condition evaluation (t) ≥ 0.7. The strict-in criteria of branches τ_2 and τ_4 are a negation of the strict-in criteria of edges τ_1 and τ_3.



F i g u r e 3. GLIKREM of model situation

S o u r c e: Own work.

Credit. The student gets the credit if the strict-in criterion of the edge ψ_1 is met, i.e., the student completes the study of all chapters (k_i), completes at least 70% of exercises, or completes the project p and successfully passes the test t. The strict-in cri-

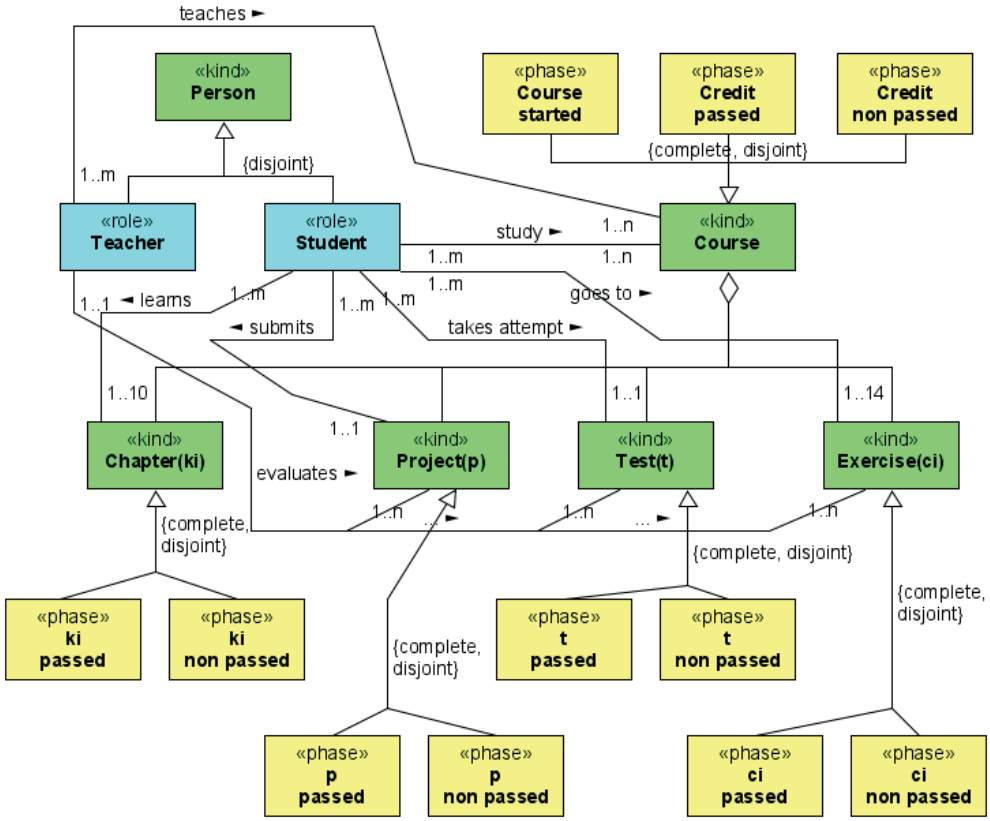
terion of an edge ψ_2 is defined as a negation of the strict-in criterion of an edge ψ_1. All strict-out criteria are, in all cases, a negation of the strict-in criteria.
The rule-in and rule-out criteria are not used at all for this model situation. However, these criteria could be used, for example, as a recommendation for the student to re-study the study materials (Chapters) in case he fails to obtain the necessary test evaluation on the first attempt (rule-in criterion of edge τ_2). Studying the chapters again is not obligatory for the student (strict-in), but only recommended (rule-in).

2.3. OntoUML conceptual diagram of the model situation

When transforming GLIKREM to a model in OntoUML notation, it is necessary to find all relevant types (OntoUML) based on the analysis of individual steps and parameters in GLIKREM. In this process, the following rules can be generalized:

- Action steps are usually converted to <kind> (or <subkind>) types or relationships between <role> types.
- The state steps are converted almost exclusively to the <phase> types of the corresponding <kind> types.
- Decision steps, including decision criteria, will be applied only during the implementation of the OntUML model, i.e., during its transformation into a specific implementation model. The same applies to the branch steps and synchronisation steps.

The result of the GLIKREM transformation of the model situation into OntoUML is shown in Figure 4. For clarity, the model in OntoUML is not complete, i.e. it does not contain all the elements (types), attributes and association between types that would be necessary for its implementation.
When transforming a model situation with GLIKREM into OntoUML, it is necessary first to define the basic types <kind>, which are the Person and Course types. Two types <role> Teacher and Student are set for the Person supertype, and both roles have a relational dependence on the Course type.
For the Course type, three <phase> types are defined, which represent the state steps of GLIKREM Course started, credit passed, and Credit non passed. The Course type consists of the <kind> Chapter, Project, Test, and Exercise types corresponding to the action steps of GLIKREM of the same name. For each activity, two types <phase> Passed and Non passed are defined, representing the respective state steps in the GLIKREM model situation. The first and second test attempts are implemented as an Attempt attribute of type <kind> Test.
All decision conditions (or criteria) and the actual process of transition between phases (types <phase>) are implemented in the corresponding methods of the respective taps <kind>. The mentioned elements (steps) of GLIKREM, i.e. decision steps and synchronisation steps including decision criteria, are not transferred to OntoUML at the conceptual level of the model, but only subsequently in the phase of transfer to a specific implementation model.



F i g u r e 4. The model situation in OntoUML notation

S o u r c e: Own work.

CONCLUSION

This article proposes a Knowledge Representation Model of Students' Activities (KRMSA), which describes a model situation of a student passing an electronic course and obtaining credit from this course. The resulting KRMSA model situation is based on the Guideline Knowledge Representation Model (GLIKREM) and its subsequent transformation into a conceptual model in OntoUML notation.

From the conceptual model in OntoUML, it is then possible to derive a specific implementation model usable, for example, for automatic student evaluation in a given course or as the basis of decision support systems in the background of LMS Moodle. The resulting model is based on the described model situation but is easily adaptable to any other situation according to the requirements for completing the course.

ACKNOWLEDGEMENTS

Centre of Business Informatics supported this work at the Faculty of Information Technology, CTU in Prague.

REFERENCES

Boxwala, A. A., Peleg, M., Tu, S. W., Ogunyemi, O., Zeng, Q., & Wang, D. (2004). GLIF3: A Representation Format for Sharable Computer-Interpretable Clinical Practice Guidelines, *Journal of the Biomedical Informatics*, *37*(3), pp. 147–161.

Buchtela, D., Veselý, A., & Vynikarová, D. (2010). Guideline Knowledge Representation Model (GLIKREM). *Knowledge Management and Modern Information Technologies*, pp. 26–41. Prague: Alfa publishing.

Guizzardi, G. (2005). *Ontological foundations for structural conceptual models*. Enschede: Telematica Instituut / CTIT.

Koķe, T. (2003). Continuous Education: the Main Tasks and Their Implementation. *Nepārtrauktās iz glītības sociāli pedagoģiskie aspekti.* Rīga: SIA "Izglītībassoļi", pp. 4–16.

Peleška, J., Buchtela, D., Anger, Z., Šebesta, K., Tomečková, M., Veselý, A., Zvára, K., & Zvárová, J. (2005). Formalisation of Medical Guidelines. *European Journal for Biomedical Informatics*, pp. 133–141.

Rybola, Z., Pergl, R. (2016). Towards OntoUML for Software Engineering: Transformation of Rigid Sortal Types into Relational Databases. *Proceedings of the 2016 Federated Conference on Computer Science and Information Systems*, Vol. 8 (pp. 1581–1591). M. Ganzha, L. Maciaszek, M. Paprzycki (Eds).

Rybola, Z., Pergl, R. (2017). Towards OntoUML for Software Engineering: Transformation of Kinds and Subkinds into Relational Databases. *Computer Science and Information Systems*, *14*(3), pp. 913–937.

Stoyanova-Doycheva, A., Glushkova, T., & Ivanova, V. (2019). Application of subject domain ontologies in e-learning. *E-learning and STEM Education Scientific.* Editor E. Smyrnova-Trybulska. "E-learning", *11*, Katowice–Cieszyn, pp. 93–107.

Veselý, A., Zvárová, J., Peleška, J., Buchtela, D., & Anger, Z. (2006). Medical Guidelines Presentation and Comparing with Electronic Health Record. *International Journal of Medical Informatics*, *75*(3–4), pp. 240–245.