# Do We Need to Teach Testing Skills in Courses on Requirements Engineering and Modelling?

Gayane Sedrakyan
Dept. of Decision Sciences and Information Management
K.U. Leuven
Leuven, Belgium
gayane.sedrakyan@kuleuven.be

Monique Snoeck
Dept. of Decision Sciences and Information Management
K.U. Leuven
Leuven, Belgium
monique.snoeck@kuleuven.be

***Abstract*—It is commonly accepted that quality testing is the integral part of system engineering. Recent research highlights the need of shifting testing of a system to the earliest phases of engineering in order to reduce the number of errors resulting from miscommunicated and/or wrongly specified requirements. Information and Computer Science education might need to adapt to such needs. This paper explores the perspectives and benefits of testing-based teaching of requirements engineering.**

**Model Driven Engineering (MDE) is known to promote the early testing perspective through fast prototyping of a prospective system contributing in this way to semantic validation of requirements. Our previous research presents empirically validated positive results on the learning effectiveness of model-based requirements engineering in combination with adapted MDE-prototyping method within an educational context to test the requirements and to test the requirements testability. Despite these positive results, our observation of the prototype testing patterns of novice analysts suggest that combining this prototype-based learning with the teaching of testing skills, such combined approach can result in even better learning outcomes.**

***Index Terms*—Requirements, analysis, conceptual modelling quality, testing, validation, prototyping, feedback, technology-enhanced learning.**

## I. Introduction

### *A. Problem domain*

In the early project phases the functionality of the prospective system is not yet understood precisely enough for formalization, which makes the requirements elicitation not only a refinement, but also ***a learning process***. This process is complicated by at least two problems present in natural language: ambiguity and inaccuracy. Formalization of requirements through ***models*** enables quality control at a level that is impossible to reach with requirements articulated in natural language. While experienced requirements engineers manage to mentally picture the prospective system in their mind when transforming requirements into formal models, such ability to truly understand the consequences of modelling choices can only be achieved through extensive experience. However, the tacit knowledge expert have developed over time is difficult to transfer to junior analysts. While teaching such knowledge and skills to novice analysts is already a challenging task considering that system *analysis* is by nature an inexact skill, transferring the academic knowledge and skills to real world businesses is yet another concern as the classroom and real world situations are not identical [1]. In their early careers the error-prone problem-solving patterns of novices and their lack of capability to identify relevant triggers for requirements verification lead to incomplete, inaccurate, ambiguous, and/or incorrect specifications [2]. When detected later in the engineering process such requirements errors can be expensive and time-consuming to resolve [3]. This significant gap between the knowledge and skills of novices and experts triggers the question of how analysis skills can be trained to facilitate the fast progression of novice analysts into advanced levels of expertise.

### *B. Testing perspective contributes to improved knowledge*

Testing is known as an integral part of software engineering. Recent research highlights the need of shifting testing of a system to the earliest phases of engineering [4]. The term *early testing* is used to define a line in test research oriented to enhance the systematic implementation of test cases based on system requirements and business models [5]. Several approaches (such as the V-model [6] or the Business Driven Test Management [7]) focus on early testing of business requirements within the system development process. Testing of requirements includes the following perspectives: 1. requirements must be tested and validated, 2. Test cases must be defined early, 3. Requirements must be specified in a way to be testable [8]. Teaching testing knowledge and skills is however largely neglected from Requirements Engineering courses. While testing is refined into a more exact discipline using well-established standards, processes and document artefacts to integrate software and requirements [9], knowledge of requirements analysis is inexact by nature and is mostly reliant on experience. This suggests that teaching requirements engineering using a test-based approach may contribute to improved requirements engineering skills.

### *C. Prototyping supports testing-based learning*

Model Driven Engineering (MDE) [10] is known to promote early testing of software requirements through fast prototyping of a prospective system contributing in this way to the semantic validation of requirements (see *Fig. 1*).

The learning context of prototyping as a type of simulation (e.g. learning by experiencing [11], [12]) suggests that, when adapted to the educational context, MDE prototyping can support the testing-based teaching of requirements engineering skills. In this work we explore the effectiveness of testing-based teaching of requirements analysis and validation using conceptual modeling and MDE prototyping method. We posit that *testing-based teaching of conceptual modeling can contribute to improved skills of novice business analysts for analysis, verification and validation of requirements*. This then raises the question of "*how the testing perspective can be integrated in the educational context?*".

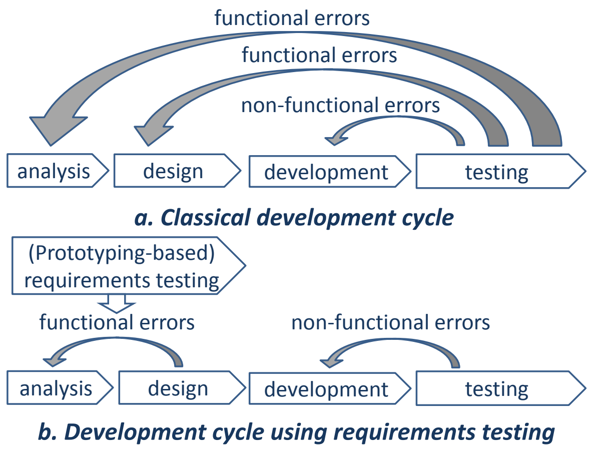


Fig. 1. Prototyping-based testing of requirements

## II. EDUCATIONAL CONTEXT AND CONCEPTS

The proposed method (adapted MDE environment) has been developed by the Management Informatics research group at the faculty of Business and Economics, University of K.U. Leuven. The approach has been subsequently tested and validated within the course "Architecture and Modeling of Information Systems"[1] over a 5-years period of teaching, with participation and constant feedback from 500 students overall. The course targets at master level students with heterogeneous backgrounds from the Management Information Systems program. The goal of the course is to familiarize the students with modern methods and techniques of Object-Oriented Analysis and Design for Enterprise Information Systems. Within the course the specific focus is on ***functional requirements***. We motivate this choice by several reasons. When propagated to the later stages of development, requirements errors incur high cost to repair. Empirical studies show that more than half the errors that occur during system development are requirements errors [3]. Furthermore requirements errors are the most common cause of failure of development projects [3]. The software development process involves the translation of information from one form to another (e.g. from customer needs to requirements to architecture to design to code). Because this process is human-based, mistakes are likely to occur during the ***translation steps*** [13]. Formalization of requirements through ***models*** enables quality control at a level that is impossible to reach with requirements articulated less formal in natural language. Formalization of requirements includes transformation of informally represented knowledge into a formal specification that is a good example of a (transformation step) affecting all three dimensions of requirements engineering: specification, representation, agreement [14]. Because of targeting a high level functional view on the prospective system, functional requirements can be formalized by means of highly abstract design representations – ***conceptual models***. As a sub-discipline of requirements engineering, conceptual modeling is described as the process of formally describing a problem domain for the purpose of understanding and communicating system requirements [15], thus making it easier to integrate business domain and ICT expertise in the system design process. In particular, conceptual models are an essential instrument to capture and formalize the domain assumption part of requirements [16]. Furthermore being a sub-discipline of requirements engineering (communicating requirements) and software engineering (providing a foundation for building information systems) [17] makes conceptual models the ***earliest formally testable artefact***. Conceptual modeling also supports the MDE approach, which, in addition to its testing potential, brings forward additional requirements towards models such as a sufficient level of preciseness and detail to provide executable specifications, contributing in this way to improved quality of design artefacts. Thus we focus on conceptualization of functional requirements as a basis of producing formally testable artefacts to facilitate the process of domain understanding and requirements elicitation.

[1] The course's page can be found on http://onderwijsaanbod.kuleuven.be/syllabi/e/D0I71AE.htm

## III. RELATED WORK

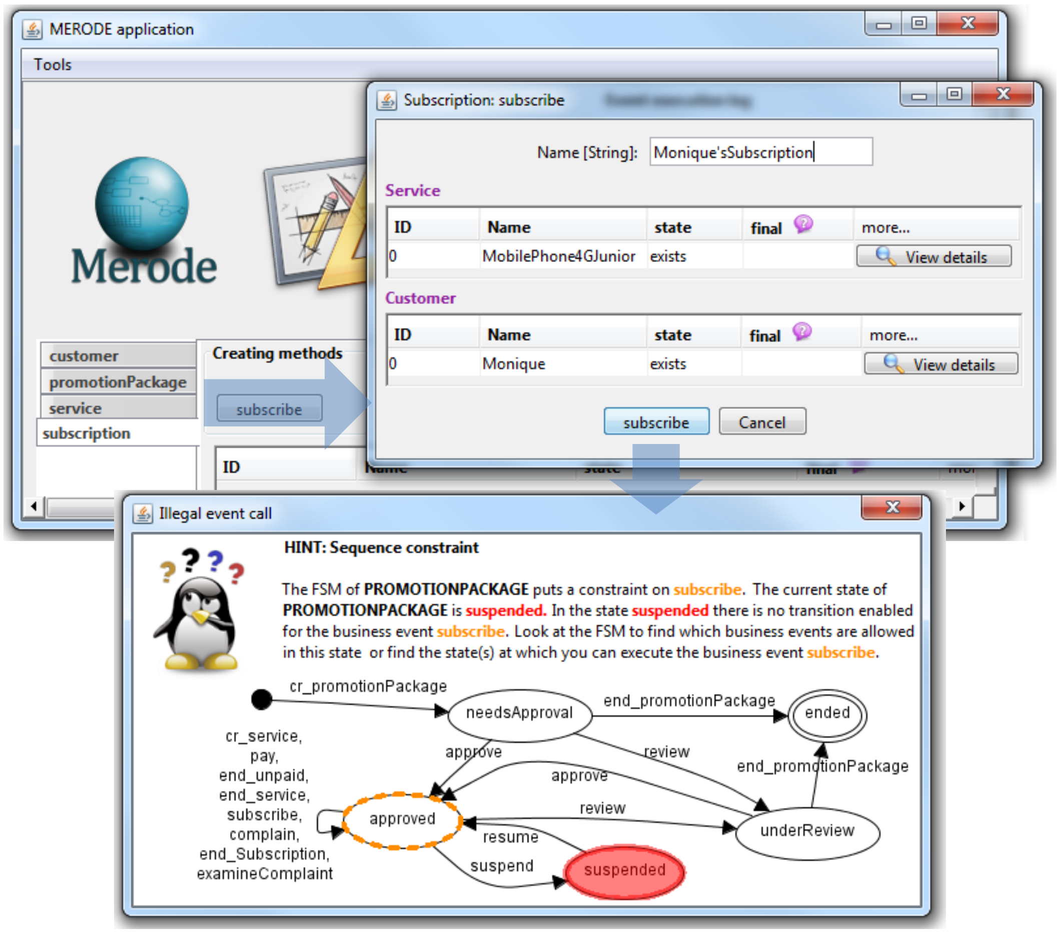
Despite the considerable amount of work devoted to simulation methodologies and prototyping in particular, to our knowledge no research publications have been written describing courses that use prototyping in the context of requirements engineering, nor empirically proven learning benefits have been reported for a certain tool. The reason is that the existing standards for simulation/prototyping technologies also introduce a number of shortcomings. Among major reasons are (1) being too complex and time consuming to achieve by novice analysts whose technical expertise is limited, (2) the difficulty of interpreting the simulation results. Among different types of simulation, the method of prototyping is capable of achieving the most concrete form of a prospective system. In our previous works we proposed a lightweight MDE-based prototyping method adapted to learning context. The effectiveness of a prototype in a learning context was enhanced by the use of *textual and graphical feedback* when and why the execution of a triggered business event is refused, thus making the links between a prototype and its model explicit [18], [19], [20]. The methodology used (rapid prototyping method enabled by executable conceptual models) is based on the concepts of MERODE [21]. A sample screen shot is shown in *Fig. 3*.

Fig. 2. Testing a prototype requires a skill

The prototyping method was also maximally adapted to novice analysts whose technical expertise is limited. The effects of feedback-enabled simulation on learning outcomes of novice learners were observed by means of empirical studies. Extensive experimental testing with participation of 114 students has demonstrated the positive effect of prototype-based simulation on requirements analysis and validation skills of junior modelers [19]. Despite the significant improvement of learning outcomes, we also observed several difficulties in students' testing cycles (see the following chapters).

## IV. Teaching Experiences with Feedback-Enabled Prototyping

Throughout the semester testing-based analysis and validation cycles are stimulated by a problem-based learning method. In parallel with theoretical sessions students are requested to participate in computer lab exercise sessions in which they are given analysis tasks such as validating a given conceptualized specifications (usually a conceptual model solution of their peers) against given business requirements. The proposed solutions usually contain erroneous models which students need to read, understand, validate against requirements and in case design errors are detected propose improvements. Validation cycles are supported by MDE-prototyping as described in this paper. During the semester students are also assigned a group project (a real-world case with approximately 5-15 pages requirements document). At the end of the semester the solution is scored, and then students are interrogated to determine the final score as a correction on the model score. In the cohort of January 2012, students were asked to demonstrate their solution by *manually* inspecting the model using a test case provided by the teacher. Less than half of the students in this cohort were able to identify mistakes in their solution, not even when manually simulating it through a mental execution with a given test scenario. In the cohort of January 2013, the same type of evaluation was performed, but this time students had to *execute* the given test scenario *using the prototype*. By means of the dynamic testing approach in this cohort, more than half of the students were not only able to see mistakes but were also able to correct them. Although this result is positive, we nevertheless observed student incapacities to develop their own adequate test scenarios [19]. To assess the effectiveness of the feedback-enabled simulation cycle on learning outcomes



Subscription: subscribe

Name [String]: Monique'sSubscription

Service

| ID | Name | state | final | more... |
|---|---|---|---|---|
| 0 | MobilePhone4GJunior | exists | | View details |

Customer

| ID | Name | state | final | more... |
|---|---|---|---|---|
| 0 | Monique | exists | | View details |

subscribe Cancel

**HINT: Sequence constraint**

The FSM of **PROMOTIONPACKAGE** puts a constraint on subscribe. The current state of **PROMOTIONPACKAGE** is **suspended.** In the state **suspended** there is no transition enabled for the business event subscribe. Look at the FSM to find which business events are allowed in this state or find the state(s) at which you can execute the business event subscribe.

Fig. 3. Validation through prototyping using feedback

of novice learners three studies were conducted in the context of two master-level courses from two different study programs spanning two academic years with participation of 104 students overall. During the experiments students were asked to assess whether or not the model reflected a particular requirement statement correctly by responding to a set of true/false questions (requirements rephrased into test questions), e.g. "in this model solution invoicing is required to buy a retail product (TRUE/FALSE?)". They were also asked to motivate their answers. For each correct answer 1 point was attributed, and 0 for each wrong answer. In total 8 questions had to be answered (min. score = 0; max. score = 8). The results were analyzed by comparing the test scores of students using the simulated model in the process of validating the proposed model solutions to the results of the tests in which they did manual inspection. The results of the statistical analysis showed significant improvement on students' capabilities to validate conceptual specifications for given requirements (relative advantage (positive correction) of approximately 2.33 points on 8 was observed; $\overline{X}_{without} = 3.1$, $\overline{X}_{with} = 5.43$, $p = 0.000$) [19]. The evaluation by students for the improved tool extended with feedbacks in 2013 resulted in average of 4,58 on perceived usefulness (for the prototyping tool) and 4,52 (for the incorporated feedbacks) on a five-point Likert scale.

### *A. Observations of testing patterns*

As stated above while the findings of the experiments showed a significant improvement in students' model-based validation capabilities when using feedback-enabled simulation, we still observed difficulties in testing by students. In this work we report on our findings on testing approaches of novice analysts by exploring the wrong answers by students. Motivations to the answers provided by students were qualitatively analyzed and the scenarios that occurred more frequently were generalized into patterns.

### *B. Testing patterns*

Major problems generalized from students motivations resulted in the following error patterns: (1) Omitted prototyping cycle; (2) Partial testing with a use of prototype characterized by incomplete testing scenarios. In their motivations for the answer when a simulation cycle was omitted, students referred to a modeling construct that according to them was already obvious with manual inspection (e.g. relationship is optional), failing to consider another constraint that resulted in a mandatory relationship (e.g. cardinality constraint was omitted). The following frequent patterns were found in the motivations where a partial test was performed:

- Pattern 1: Confirmative rather than explorative (approximately 20% of wrong answers)

  **Sample requirement** : "Each request can be processed by exactly one reviewer".
  **Testing approach** : The testing scenario is limited to confirmation scenario. While the requirement is tested for the positive case "can be viewed by a reviewer", testing the constraint "by not more than one" was omitted.

- Pattern 2: Insufficient examination of path dependencies to identify related instances through transitive paths of dependencies (approximately 50% of wrong answers)

  **Sample requirement** : "Ordering is not required for selling Retail Products to Walk-in Customers".
  **Testing approach** : The testing scenario is limited to the first level of dependency, e.g. the student's motivation refers to the need of creating an invoice line which only requires an instance of invoice, thus rejecting the dependency to order. Testing the next level dependency between invoice and order was omitted (i.e. the creation of invoice was not executed to discover the dependency).

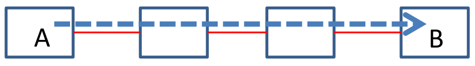


Fig. 4. Transitive path of dependencies

- Pattern 3: Insufficient examination of path dependencies to identify related instances through parallel paths of dependencies (approximately 30% of wrong answers)

  **Sample requirement** : "If a business customer A orders some products, then it is possible that business customer B pays the invoice for these products.
  **Testing approach** : Testing scenario is limited to one of the parallel paths, e.g. when a direct relationship between invoice and a customer was examined, the examination of a hidden relationship through order object linked both to invoice and customer objects was omitted.

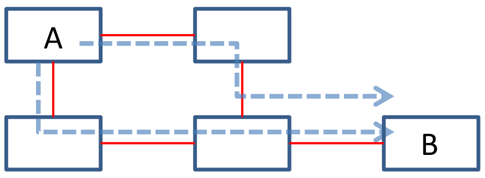


Fig. 5. Parallel paths of dependencies

## V. Proposed Solution: Borrowing Testing Arsenal

An example of testing an erroneous model is shown in *Fig. 3* by means of a model about (mobile phone) services which customers can subscribe to, and for which promotion packages are offered regularly. Testing the prototype reveals a semantic mismatch (design error): trying to subscribe to a service results in execution failure due to a sequence constraint violation (the state of the "promotionPackage" object to which the chosen service is associated is "suspended"). The scenario fails because of a behavioral constraint, but it actually reveals a wrong hidden dependency from "service" to "promotionPackage": it seems a service depends on the availability of a promotion, which is incorrect. The explanation can be extended with graphical visualization linking to the specific part of the model that causes the error.

While in the example above the testing results can be interpreted subjectively by students depending on their analytical skills, teaching a more systematic testing approach would benefit to improved skills for verification. To stimulate test-based requirements validation we propose borrowing the concept of acceptance test, the goal of which is to ensure the testability of requirements [6]. This requires teaching knowledge of how to write/reformulate requirements as tests with the use of testing artifacts such as Test Case (purpose, assumptions, preconditions, steps, expected outcome, actual outcome, post-conditions) and Test Scenario (process flows, i.e. sequence of executing test cases). Next, the concept of coverage testing can be including to ensure the completeness of execution (each requirement should be exercised at least once). To ensure better results peer expertise can be exploited by peer reviews of group projects in which one group of students would act as testers for another group. A simple example demonstrating an improved validation cycle for an erroneous model (see *Fig. 6*) with the use of a testing artefact is presented below.

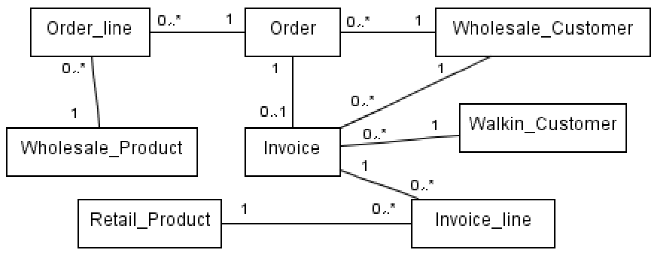


Fig. 6. Sample erroneous model

For the requirement statement "Ordering is not required for selling Retail Products to Walk-in Customers" a student would have to specify a test scenario (in the model solution of a student selling requires registering an invoice) … In random, blind verification of this requirement, a student's attempt to create an invoice line will reveal the need for an invoice first: a popup window of a prototype would suggest creating an instance of invoice (or choosing from existing instances) to be associated with a newly created instance of invoice line. This will lead to the conclusion that invoicing is required (but not ordering) and hence to the erroneous conclusion that the requirement is satisfied. A systematic approach to test plan development would stipulate defining a *complete* test scenario, including the creation of the invoice which would then reveal a dependency from invoice to the order object (a popup window of a prototype requiring a creation or choice of an existing order instance) to be associated with an instance of invoice, leading to the conclusion that the above requirement is not satisfied.

Furthermore, teaching regression testing knowledge can benefit to improved skills for integrating changes in requirements (identifying the test scenarios to be repeated because of a change). To stimulate such analytical skills assignments for integrating modifications in requirements can be used.

## VI. Conclusions

We compared the results of oral examination with and without testing scenarios provided by the teacher. Two conclusions were obtained from this comparison: 1. the results demonstrate that the testing by means of a working prototype improves model understanding compared to a paper exercise by 2.33 points on 8. The paper exercises limit the scope of understanding to a static view of a model, whereas dynamic testing fosters a more thorough understanding; 2. Validation cycles supported with test scenarios provided by the teacher resulted in better model understanding indicators than unassisted testing cycles. The results of experiments from our previous studies also confirmed the effectiveness of testing-based learning of analysis and validation of requirements over traditional methods of learning allowing a student to build a deeply understood knowledge that is developed from own practice. The observations of testing patterns of students also suggest that when combined with teaching high level testing knowledge and skills the method will result in even better learning outcomes. The results of this work contribute to innovative teaching practices by means of computer-enhanced learning [22] in the domain of requirements engineering thus promoting to better skill preparedness of novice analysts.

The work presented in this paper can be extended in several ways. One direction would be related to data collection by means of the logs of the prototyping tool that might provide new insights on testing approaches and patterns of novices. While our observations were limited to a single prototyping cycle within the context of oral exams and experiments, another possibility could be the investigation of testing patterns extended to longer periods of observations, e.g. prototyping logs of testing activities for group projects. Examination of testing patterns where a combination of structural and behavioral constraints are involved could be interesting as well. Based on the findings a tool support to enable automated assistance or generation of test scenarios can be investigated as well.

## References

[1] Damassa, D. A., & Sitko, T. (2010). Simulation Technologies in Higher Education: Uses, Trends, and Implications. *EDUCAUSE Center for Analysis and Research (ECAR), Research Bulletins*.

[2] Schenk, K. D., Vitalari, N. P., & Davis, K. S. (1998). Differences between Novice and Expert Systems Analysts: What Do We Know and What Do We Do? *Journal of Management Information Systems, 15*(1), 9-50.

[3] A. Enders, H.D. Rombach. (2003). A Handbook of Software and Systems Engineering: Empirical Observations, Laws and Theories Addison-Wesley, Reading, MA, USA

[4] Robertson, S. (2000). Requirements testing: Creating an effective feedback loop. *FEAST 2000*.

[5] Gutiérrez, J.J., Escalona, M.J., Mejías, M., Torres, J. Generation of test cases from functional requirements A survey, 4th Workshop on System Testing and Validation, Potsdam, Germany (2006)

[6] V-Model Lifecycle Process Model. http://v-modell.iabg.de/

[7] Roodenrijs, E., van der Aalst, L., Baarda, R., Visser, B., Vink, J., (2008) TMAP NEXT® - Business Driven Test Management, UTN, ISBN 9789072194930

[8] Pohl, K. (2010). Requirements Engineering: Fundamentals, Principles, and Techniques, Berlin, ISBN 978-3-642-12577-5

[9] ANSI/IEEE Std 829-1983, IEEE Standard for Software Test Documentation

[10] OMG. Model-Driven Architecture. http://www.omg.org/mda/

[11] Kluge, A. (2007). Experiential Learning Methods, Simulation Complexity and their Effects on Different Target Groups. *Journal of Educational Computing Research, 3*(36), 323-349.

[12] Barjis, J., Gupta, A., Sharda, R., Bouzdine-Chameeva, T., Lee, P. D., & Verbraeck, A. (2012). Innovative Teaching Using Simulation and Virtual Environments. *Interdisciplinary Journal of Information, Knowledge, and Management, 7*, 237-255.

[13] Walia, G., Carver, J. (2009). A systematic literature review to identify and classify software requirement errors, Information and Software Technology, Volume 51, Issue 7, pp.1087-1109, ISSN 0950-5849

[14] Pohl, K. (1994). The three dimensions of requirements engineering: A framework and its applications, Information Systems, Volume 19, Issue 3, pp. 243-258, ISSN 0306-4379, http://dx.doi.org/10.1016/0306-4379(94)90044-2

[15] Siau, K. (2004). Informational and computational equivalence in comparing information modeling methods. Journal of Database Management (JDM), 15(1), 73-86.

[16] Jureta I.J., Mylopoulos J., Faulkner S. (2008). Revisiting the Core Ontology and Problem in Requirements Engineering, IEEE International Requirements Engineering Conference, 71-80

[17] Moody D. L., Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions, Data & Knowledge Engineering, Volume 55, Issue 3, December 2005, pp. 243-276, ISSN 0169-023X.

[18] Sedrakyan, G., & Snoeck, M. (2013). A PIM-to-Code requirements engineering framework. In Proceedings of Modelsward 2013-1st International Conference on Model-driven Engineering and Software Development-Proceedings, 163-169.

[19] Sedrakyan, G., & Snoeck, M. (2013). Feedback-enabled MDA-prototyping effects on modeling knowledge. In Enterprise, Business-Process and Information Systems Modeling (pp. 411-425): Springer.

[20] Sedrakyan, G., Snoeck, M., Poelmans, S. (2014). "Assessing the effectiveness of feedback enabled simulation in teaching conceptual modeling", Computers & Education (accepted).

[21] Snoeck, M., Dedene, G., Verhelst, M., Depuydt, A.: Object-oriented enterprise modelling with MERODE, Leuvense Universitaire Pers, Leuven (1999)

[22] EuropeanCommission. (2013). Opening up education: Innovative teaching and learning for all through new technologies and open educational resources, http://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1389115469384 &uri=CELEX:52013DC0654