Work in Progress: Towards a Generic Platform for Implementing Gamified Learning Arrangements in Engineering Education

A. Bartel[1], G. Hagel[1] and C. Wolff[2]
[1]Kempten University of Applied Sciences, Germany
{alexander.bartel, georg.hagel}@hs-kempten.de
[2]University of Regensburg, Germany
christian.wolff@ur.de

***Abstract* — This contribution discusses problems of existing gamified learning platforms as a result of an analysis and reveals a research gap that is addressed using a domain-specific modeling (DSM) approach. Foundations of the DSM approach are described and our vision to use it for improving the creation, adaptability, and extensibility of platforms as well as the exchange of promising gamified learning concepts. Early results are presented and steps ahead are shown.**

Keywords — Gamification; Domain-specific Modeling; Domain-specific Modeling Language; E-Learning

I. INTRODUCTION

In the last years, the use of gamified techniques for educational purposes has gained more and more attention. While gamification is a concept which is not necessarily connected to digital platforms alone, however, this representation is a promising way to reach a large number of learners. That is one of the reasons, why the number of gamified learning platforms has increased, each platform providing more or less unique features to distinguish it from existing solutions. To clarify the term *gamified learning*, we rely on the definition of gamification in learning contexts that is "[…] described as a concept which integrates game elements and processes into learning activities in order to increase learning motivation and thereby changes the participant's attitude and behavior" [1].

A. Motivation and research gap

What is known so far, is that all gamified learning platforms take over elements of gameful design in order to reflect gamified concepts (e.g. [2, 3]). Although these platforms provide a variety of game design elements, researchers claim that profound gamified learning concepts require more sophisticated functionality. There is a "[…] necessity of a new type of educational software that can support intelligent mentoring of gamified flipped learning formal classes or informal groups of learners" [4, p. 6].

Currently, and to our best knowledge, there is little research on a broader view on gamified learning platforms that not only considers the implementation and educational contexts, but takes into account the *creation*, the degree of *adaptability* or *extensibility* and the *exchange* of promising gamified learning concepts as well, so that requirements like intelligent monitoring of learners can be implemented easily.

In order to gain insights on the characteristics mentioned above, an analysis of 11 gamified learning platforms was conducted. Only those platforms were included that fulfilled the following criteria:

- *Domain*: Computer science education
- *Focus group*: Advanced secondary school and university education
- *Cost model*: Significant learning parts are free of charge
- *Language*: German or English

Other criteria, for example the number of registered users of each platform, were not considered as a selection criterion, since the insights gained should not rely on the popularity of a platform. Thus, the list of platforms given in Table I was produced.

Table I. Gamified learning platforms included in the analysis

Platform name	URL[a]
Alison	http://alison.com/
Classgame	https://classgame.de
Codeacademy	http://www.codecademy.com/
Code School	https://www.codeschool.com/
Coursera	http://www.coursera.org
Iversity	https://iversity.org
Khan Academy	https://de.khanacademy.org/
Open Learning Initiative	http://oli.cmu.edu
Lagunita	https://class.stanford.edu
Udacity	https://www.udacity.com/
Udemy	https://www.udemy.com/

[a] All accessed in May 2016

All platforms in the result set are either exclusive providers of gamified learning arrangements in the field of computer sciences or subordinate areas.

This work is part of the EVELIN project which is funded by the German Ministry of Education and Research under the grant no. 01PL12022C.

B. Relevant findings

A quantification of the characteristics discussed below would not contribute to a meaningful representation of our findings. This is why we concentrate on summarizing the results on a more abstract level using a qualitative description.

a) Creation

The creation of such platforms is a compromise between partially antagonistic requirements (e.g. [3]): For example, demand-driven features for a certain purpose and the adaptability to reuse a platform are conflicting criteria, since they pose the question of specialization versus generalization. Hence, if lecturers want to use existing platforms, they need to modify the existing platform for a certain context first. If a platform is not flexible enough for this kind of task, because it was built to fulfil a special purpose, this intention can generate a great amount of effort [5, 6].

b) Portability and Extensability

In most cases, modifications cannot be done by lecturers, who are domain experts, though, but require more sophisticated knowledge, particularly technical knowledge [7]. Consequently, there is a gap between *domain experts*, whose focus is on the implementation of their ideas and concepts for gamified learning and *technical experts*, who actually undertake the implementation of changes or modifications required by domain experts' gamified learning concepts. The old idea of *authoring system*s which was to overcome this type of gap, obviously doesn't take hold for current gamified learning environments yet. From a software engineering perspective, this gap can cause various unwanted side effects, like an ambiguity in aims, due to the personal mental model regarding a certain domain – a common characteristic of human beings (e.g. [8]).

c) Exchange

An other case is given, when platforms cannot reflect existing gamified learning concepts which worked in other learning settings. The exchange of promising concepts is hindered, since the underlying technology is too tightly coupled to its contents. At the same time, a standardized format to describe gamified learning arrangements (typically described in natural language, as proposed in [9]) is not existent to our best knowledge. This circumstance can lead to compromises in applying gamified learning concepts, with the platform being the limiting factor. Therefore the potential of the gamification concept is significantly reduced and could result in providing lopsided motivational experiences – a fact that is often criticized in literature (e.g. [10]).

Furthermore, even if a gamified learning concept is completely transferable into a platform, there is no guarantee that the learning concept and its implementation would actually fulfil the lecturers' expectations, for example in terms of student motivation and learning outcomes. These effects can occur when learning platforms are seen as an isolated and rigid means for teaching and learning purposes, without any relation to in-class learning activities.

Summing up, the analysis revealed some general issues of existing platforms which creates a great potential for improvement.

II. Vision and Implementation Approach

To address these issues, we describe our vision and the recent state of an integrated approach that is based on the concept of *domain-specific modeling* (DSM) [7]. It will allow domain experts without the need of any technical background knowledge to transfer gamified learning concepts into a working platform by using a *domain-specific modeling language* (DSML) having a domain-based and therefor well-established notation. Once a gamified concept is expressed through the DSML, it can automatically be transferred into source code. The generated code can serve as the core of the dynamic behavior in a gamified learning platform that is able to control (static) elements of the platform, like the model and view parts (talking in terms of an MVC-architecture).

A. A brief overview on domain-specific modeling

The term DSM was originally coined by Tolvanen and Kelly who described it as an instance of the model-driven software design (MDSD) approach [11]. The use of DSM implies focusing on a specific (problem-) domain, extracting relevant domain concepts and domain rules and raising the level of abstraction in a way that both, structure as well as behavior can automatically be transferred into executable source code (also called solution domain) [7, 12, 13, 14]. Figure 1 illustrates this relation.

Both, DSM and MDSD, use one or more models in order to describe the concepts and rules and to automatically generate code, however, having the crucial difference that in MDSD "[…] model transformations normally mean that during each step developers extend the automatically produced models with further details" [7, p. 57]. This necessity is completely dispensed in the DSM approach.

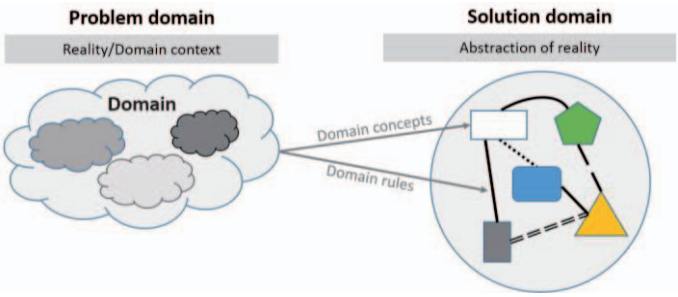

Fig. 1 Problem domain and solution domain

DSM builds upon principles of domain-driven design (DDD) [15], like focusing on the problem domain and describing the domain concepts and domain rules in model abstractions, and extends it with guidance for practicability [7, 13, 16]. This is the reason why Kelly and Tolvanen describe DSM through its components which are "[…] the *DSML*, *models*, *generator* and the *domain framework* [7, S. 68]. These components are briefly addressed in the following.

B. Elements of domain-specific modeling

1) Domain-specific modeling language

The DSML is the language which is applied in DSM and carries typical languages characteristics, like well-defined syntax and semantics. Kelly and Tolvanen suggest to further distinguish DSMLs with respect to their *concrete* and *abstract*

syntax [7]. While the concrete syntax corresponds the notation, hence, the visual means to express the relevant parts of a particular domain, the abstract syntax provides the underlying (data-) structures and formal grammatical rules the concrete syntax may express in a valid way [7, 13]. Thus, the concrete syntax is a realization of the abstract syntax of a DSML, where more than one concrete syntax can exist for realizing a single abstract syntax (e.g. [17, 18]).

2) Models

As previously suggested, abstractions of domain concepts and domain rules are expressed through models. Since the abstract syntax of a DSML covers these abstractions and thereby operates on a meta level, the common term for these kind of models is *metamodels* [19]. According to [19] and [20] a metamodel is defined as a model of a modeling language which formally defines the modeling language on a more abstract level than the modeling language itself does. A metamodel should reflect relevant domain concepts, hide unnecessary details of a domain and guide through the use of concise terms which are derived from a problem domain [8, 20].

3) Generator

To transform concrete models (instances of metamodels) into source code a generator is used which is just as specific for a certain DSML as the DSML is for the domain. In the context of DSM, the source code that a generator generates is complete and does not need any further additions or rewritings [7]. For generating, a generator "[…] accesses models, extracts information from them, and transforms it into output specific syntax" [7, p. 80]. These process steps heavily depend on how the metamodel, the domain concepts, and the rules of a DSML are built as well as on what is required by the domain framework (i.e., the environment the generated code is made for) [7, p. 80].

4) Domain Framework

As a last part of DSM, the *domain framework* "[…] provides the interface between the generated code and the underlying *target environment*" [7, p. 86]. A domain framework is not needed in every case. But when it is needed because the generated code is not working standalone and in isolation, it typically uses frameworks (e.g. J2EE) or utilities of the target environment (e.g. access library to database) to reduce complexity in the generator and in the generated code.

III. Early Results

Besides the findings of the analysis presented above, we have looked for similarities between the gamified learning platforms. This served as the foundation for abstracting domain concepts. Therefore, a domain analysis scheme (see figure 3) was used which separates between game design elements (green) and so-called containers (blue). The latter are components in the sense of software components, e.g. a forum.

We were able to identify several domain concepts, like different types of assessment (e.g. Single Choice tasks, Multiple Choice tasks etc.). For each domain concept a detailed description was created in order to identify

- the internal structure of the concept (properties),
- which functionalities the concept provides,
- how the concept relates to other domain concepts and
- in which context (purpose) the concept is applied.

Afterwards we use that information to describe the domain concepts as abstractions in metamodels. For this we use UML-class diagrams and subsequently transfer the metamodels to *Ecore*, an XML-based description format of the EMF framework [21]. Figure 2 shows an excerpt of the XML-based *Ecore* representation of a Single Choice task using EMF scheme. The *eClassifiers* are comparable to classes and *eStructuralFeatures* to attributes in an object-oriented sense.

```
<eClassifiers xsi:type="ecore:EClass" name="SingleChoice" eSuperTypes="#//Task">
  <eStructuralFeatures xsi:type="ecore:EReference" name="correctAnswer"
    unique="false" lowerBound="1" eType="#//SCAnswer"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="answers"
    unique="false" lowerBound="2" upperBound="-1" eType="#//SCAnswer"/>
</eClassifiers>
```

Fig. 2 Ecore representation of Single Choice task

For the development of the concrete syntax, we follow the recommendation of [7], who suggest to survey potential users of a platform about their perceptions how domain concepts should be graphically represented to prevent the "not-invented-here" phenomenon [7].

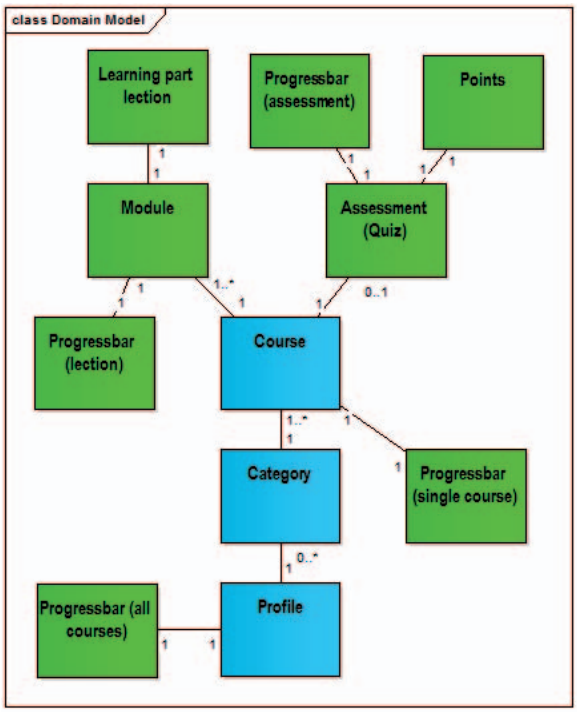

Fig. 3 Domain analysis model for Alison

Potential users (n=12) of the DSML were asked in a standardized survey format for their ideas to represent abstracted domain concepts. Figure 4 shows an example of suggestions from participants (1, 2) for the concept Single Choice task and the final notation which was created for that concept (3).

For actually allowing modeling with the DSML, *Eclipse Sirius* (https://eclipse.org/sirius/) was used. *Sirius* allows the user manipulation of instances according to the *Ecore*

metamodel by specifying a graphical modeling workbench which is based on *Eclipse* and uses the concrete syntax that was briefly presented above.

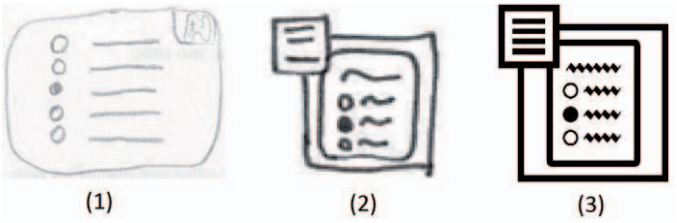

Fig. 4 Graphical notations suggestions for a Single Choice Task

The generator and the domain framework are built in parallel to avoid inconsistencies. For the generator we decided to work with *Acceleo* (www.acceleo.org), a template-based model to text language generator, which is able to process models that are based on EMF. The domain framework provides several facade patterns using interfaces that allow a defined communication between the generated controllers and the models as well as views of the platform.

IV. Steps Ahead

All steps in the development of the DSML and its tools follow a highly iterative and incremental process with intensive end-user involvement [12]. Consequently, we aim at completing our work by iterating through the work that is related to the components of the DSM that were addressed before. After the completion of DSM components, we plan to practically employ this tool chain in our software engineering courses and evaluate its effectiveness and applicability from both, the lecturer and student perspective.

V. Summary

In this contribution the vision and early results for the creation of a DSM-based tool chain are presented that allow lecturers to model gamified learning arrangements and automatically generate a working gamified learning platform out of it, which can be used to support students' learning progress. A research gap is revealed and an analysis of existing gamified learning platforms is demonstrated in order to investigate their characteristics. For the vision of improving gamified learning platforms in terms of their creation, the degree of adaptability or extensibility and the exchange of promising gamified concepts, DSM is taken as a basis. The DSM concept is briefly introduced and some early results as well as steps ahead are shown. We believe this contribution to be of interest for the research community, since it is an enabling technology for the field of engineering education and comparable systems are currently not existent.

REFERENCES

[1] A. Bartel and G. Hagel, "Gamified Just-in-Time Teaching - A Conceptual Approach Based on Best Practices," in Proceedings of ECSEE 2016 - European Conference Software Engineering Education, Aachen: Shaker, 2016, pp. 1–17.

[2] K. Berkling and C. Thomas, "Looking for Usage Patterns in e-Learning Platforms: A Step Towards Adaptive Environments," in Proceedings of the 6th International Conference on Computer Supported Education, Volume 1, Lisssabon, Portugal: SciTePress, 2014, pp. 144–152.

[3] P. Herzig, M. Ameling, B. Wolf, and A. Schill, "Implementing Gamification: Requirements and Gamification Platforms," in Gamification in Education and Business, T. Reiners and L. C. Wood, Eds., Cham: Springer International Publishing, 2015, pp. 431–450.

[4] D. Dicheva and C. Dichev, "An Active Learning Model Employing Flipped Learning and Gamification Strategies," in Proceedings of first Int. Workshop on Intelligent Mentoring Systems, 2016, pp. 1–6.

[5] G. D. Rey, E-Learning: Theorien, Gestaltungsempfehlungen und Forschung, 1st ed. Bern: Huber, 2009.

[6] P. Herzig, M. Ameling, and A. Schill, "A Generic Platform for Enterprise Gamification," in Joint Working IEEE/IFIP Conference on Software Architecture (WICSA) and European Conference on Software Architecture (ECSA), 2012, pp. 219–223.

[7] S. Kelly and J.-P. Tolvanen, Domain-specific modeling: Enabling full code generation. Hoboken, N.J.: John Wiley & Sons, 2008.

[8] C. Rupp, Requirements-Engineering und -Management: Professionelle, iterative Anforderungsanalyse für die Praxis, 5th ed. München: Hanser, 2009.

[9] A. Bartel and G. Hagel, "Gamifying the Learning of Design Patterns in Software Engineering Education," in Global Engineering Education Conference (EDUCON), 2016 IEEE, Los Alamitos: IEEE, 2016, pp. 74–79.

[10] K. Werbach, "(Re) Defining Gamification: A Process Approach," in vol. 8462, Persuasive Technology, A. Spagnolli, L. Chittaro, and L. Gamberini, Eds., Cham: Springer, 2014, pp. 266–272.

[11] J.-P. Tolvanen and S. Kelly, "Model-Driven Development Challenges and Solutions - Experiences with Domain-Specific Modeling in Industry," in Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development: SCITEPRESS - Science and Technology Publications, 2016, pp. 711–719.

[12] J.-P. Tolvanen, "Domain-Specific Modeling for Full Code Generation," Methods & Tools - Practical knowledge for the software developer, tester and project manager, vol. 13, no. 3, pp. 14–23, 2005.

[13] A. G. Kleppe, Software language engineering: Creating domain-specific languages using metamodels. Upper Saddle River, NJ: Addison-Wesley, 2009.

[14] S. Millett and N. Tune, Patterns, Principles, and Practices of Domain-Driven Design. Indianapolis, IN: John Wiley & Sons, 2015.

[15] E. Evans, Domain-driven design: Tackling complexity in the heart of software. Boston: Addison-Wesley, 2004.

[16] B. Combemale et al., Engineering Modeling Languages: Turning Domain Knowledge into Tools. Boca Raton: Taylor & Francis, CRC Press, 2016.

[17] T. Stahl and M. Völter, Model-driven software development: Technology, engineering, management. Chichester, England, Hoboken, NJ: John Wiley, 2006.

[18] M. Brambilla, J. Cabot, and M. Wimmer, Model-Driven Software Engineering in Practice. San Rafael: Morgan & Claypool Publishers, 2012.

[19] González Pérez, César A and B. Henderson-Sellers, Metamodelling for Software Engineering. Chichester, UK, Hoboken, NJ: John Wiley, 2008.

[20] T. Clark, A. Evans, P. Sammut, and J. Willans, Applied Metamodelling: A Foundation for Language Driven Development: CETEVA, 2004.

[21] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, EMF: Eclipse Modeling Framework, 2nd ed. Upper Saddle River, NJ: Addison-Wesley, 2009.