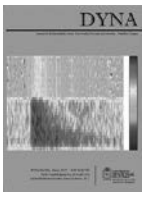
UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE MEDELLÍN
FACULTAD DE MINAS



DYNA
http://dyna.medellin.unal.edu.co/

UAREI: A model for formal description and visual representation /software gamification

Darius Ašeriškis, Tomas Blažauskas & Robertas Damaševičius

Software Engineering Department, Kaunas University of Technology, Kaunas, Lithuania. darius.aseriskis@ktu.lt, tomas.blazauskas@ktu.lt, robertas.damasevicius@ktu.lt

Received: November7th, 2015. Received in revised form: October 12th, 2016. Accepted: December 17th , 2016.

Abstract
The paper presents the UAREI (User-Action-Rule-Entities-Interface) model for formal specification of software gamification, and the UAREI visual modelling language for graphical representation of game mechanics. A case of study in gamification of the Trogon project management system is presented. The proposed model and visual language is compared against the Machinations gamification framework using visual complexity metrics, game simulation and qualitative comparison.

Keywords: gamification; modelling; abstraction; formal model.

UAREI: Un modelo para la descripción formal y la representación visual de la gamificación de software

Resumen
El artículo presenta el modelo UAREI (Usuario-Acción-Regla-Entidades-Interface) para la especificación formal de gamificacion software y el UAREI visual lenguaje de modelado para la representación gráfica de la mecánica del juego. Un estudio de caso en gamificacion del sistema de gestión de proyectos Trogon se presenta. El modelo y visual lenguaje propuesto se compara con las maquinaciones marco gamificacion utilizando métricas de complejidad visuales, juego de simulación y comparación cualitativa.

Palabras clave: gamificacion; modelado; abstracción; modelo formal.

1. Introduction

Gamification have been defined as a process which shapes the world (achieves goals/objectives) by influencing the actions, behaviours, characteristics and state of entities within the world through the use of games strategies and enabling technologies [1]. The concept is relatively new, but it has gained considerable interest in the software development and user interface design community over the last few years. The roots of gamification are in game design, with some elements from psychology, so there are still little academic research how to design and develop software systems with and for gamification.

According to Gartner Inc. [2], the widespread interest that gamification has been attracting recently lies in its potential to strengthen engagement, change user behaviours and support innovation. Game theory based models are being widely adopted now in different contexts and used as a driver for solving problems in a wide variety of domains, including disaster management [3], education [4,5], e-learning [6], workplace improvement [7], marketing [8], healthcare management [1,9], IT service management [10], social policy [11], sports and fitness [12], tourism business [13], customer engagement, social missions, fostering creativity, employee and management training, etc.

The underlying concept of gamification is motivation. Gamification is driven primarily by the external motivation, i.e., the users strive to compete against other playing users and to get recognized by the game community [12]. As motivation tends to decay over time, it however must be

How to cite: Ašeriškis, D., Blažauskas, T. and Damaševičius, R., UAREI: A model for formal description and visual representation /software gamification, DYNA 84(200), pp. 326-334, 2017.

supported by the increasing complexity and evolving dynamics of game mechanics [14]. Meaningful gamification (otherwise known as "serious game") is the use of game design elements to help users find meaning in a non-game context. Rather than just using game mechanics to give points or badges to users as external rewards, meaningful gamification focuses on the playing process (aka game mechanics) itself in order to engage the players to do meaningful tasks in a real world.

The modelling of gamification is important for design of systems based on the principles of serious game in order to quantify and validate the impact of gamification and to get a better understanding why and how gamification works. Existing evaluations of gamification usually focus on using user questionnaires and other methods of qualitative evaluation. There is still lack of high-level formal or abstract modelling methods and tools to aid the design and development of gamification in serious systems.

This paper aims to introduce tools which would allow to build a bridge between formal modelling of gamification and quantitative simulation of games, analysis and evaluation of game rules and processes.

The structure of the remaining parts of the paper is as follows. The overview of gamification models and gamification modelling languages is presented in Section 2. Similar formal approaches to game design are considered in Section 3. Formal description of the proposed UAREI (User-Action-Rule-Entities-Interface) model is given in Section 4. The visual notation used for modelling is described in Section 5. A case of study is presented in Section 6. The evaluation is given in Section 7. Finally, the conclusions are presented in Section 8.

2. Gamification models and modelling languages

In game research there is a strong separation between design methodologies and usability evaluation tools, which are rarely employed in the early stages of the design process. Although the game developers use many often heuristically designed tools to assist the design, there is still very little existing methods employed to connect design practices with gamification and game design [32]. Currently game and gamification development is strongly related to the qualifications and skills of game designers. This limitation drives the need to better and faster game building. Recently several new tools were developed or adapted to help game designers to model, build and analyses games.

Unified Modelling Language (UML) is a *de-facto* standard modelling language used in multiple domains. Tenzer [15] argues that UML modelling tools could be also used to build games and proposes a framework for building games using UML. The advantage of UML is that it is well known in the software engineering community. SysML is a general-purpose modeling language for systems engineering applications. That supports specification, analysis, design and verification of a broad range of systems. SysML has been used for building a training game [16].

The most notable examples of domain-specific game description languages are GaML [17,18] and ATTAC-L [19]. GaML is a formalized language for specifying and automatically generating gamification solutions. This allows to free the IT expert from the generation of gamification solutions. ATTAC-L is a domain specific language which allows the user to specify the game scenario in XML and to build a game using a code generator.

Another approach to gamification modelling is based on using formal (or mathematical) models [20]. Kim and Lee [21] model the effectiveness of gamification effectiveness using a mathematical model based on a sigmoidal equation. They argue what gamification effectiveness can be represented using curiosity, challenge, fantasy and control factors. Bista *et al.* [22] have proposed the first formal gamification model. Chan *et al.* [23] offer a similar approach for social game modelling, which also allows for verification of the built model. Oliveira *et al.* [24] model games using Petri nets. The disadvantage of this approach is the lack of domain specificity which is preventing its adoption by game designers.

The third category of gamification modelling approaches is visual languages for fast prototyping in gamification domain. Most known examples are Sketch-It-Up [25], Ludocore [26], and Machinations [27]. Sketch-It-Up is a tool for creating sketches of possible games. Ludocore is a logical "game engine", which employs formal logic used by automated reasoning tools in AI domain to enable automated design and prototyping of game systems and providing fast feedback to the designer. Machinations is a conceptual framework and diagram tool that focusses on structural qualities of game mechanics. Machinations graphical diagrams are an abstraction of Petri nets for modelling and simulating games and game-like systems on a varying level of abstraction. Recently, Micro-Machinations [28] were proposed for reusing Machinations models in software development.

3. Formal models of game design and gamification

Games are a kind of systems and the design of games is the creation of models for games [33]. In computer science, games can be considered as a kind of information systems consisting modelled using of objects (or entities, concepts), attributes (properties), their relationships and the environment (or context) [37]. A similar approach has been adopted by ontology engineering [34] for building ontologies, i.e., formal representations of concepts within a domain and the relationships between those concepts.

Formally, games can be modelled as abstract control systems [35] consisting of a set of states and a definition of the evolution of the state of game under different actions of a player. The game can be represented by a set of states, for which transition functions define when to move from one state to another. Following this approach, gamification can be described as the product of two games, where a gamified system is considered as one game with its one rules and mechanics, and the gamification layer is considered as another game.

Another game modelling framework presented in [36] incorporates structural, temporal and boundary frameworks (subsystems). The structural subsystem consists of Game Elements, Game Time, Players, Interface and the Facilitator, the arbitrating entity between the players and the game system, which takes care of setting up the game, synchronises the game state and maintains the game time. The temporal

subsystem represents the flow and causality of the game by defining the actions that are provided and the actions that can be taken at the particular states in the game. The boundary subsystem defines the constraints in the game that limit the activities performed in a game by establishing social contracts between the players which have to be satisfied while playing through a set of limitations.

In [38], another kind of formal model (Petri Nets and Hypergraphs) are investigated and methods and tools for the integration of formal modelling into the game design and production process are proposed.

These efforts in formal game modelling are however directed at game design rather than gamification of the existing systems as considered in this paper. In the following Section, the elements of the proposed UAREI model are presented.

4. Description of gamified systems as UAREI model

The gamified systems can be described as a tuple $G = \{U, A, R, E, I\}$, here: U – users, which are interacting with the system; A – actions, which trigger system behaviour; R – rules, which encapsulate logic in the system; E – data entities; and I – interfaces which define data format.

The users are defined as a tuple $U = \{L_U, S_U\}$, here: L_U – a set of all outgoing links to other elements in the model; and S_U – a selection function which defines how a user is selected from a collection in a simulation mode.

Actions are a collection $A = \{A_1, A_2, \dots, A_i, \dots, A_n\}$, here A_i is a single action, n the total number of actions. A single action is defined as $A_i = \{L_A, S_A\}$, here:L_A – a set of all outgoing links to other elements in the model, andS_A – a selection function, which defines the way an action related data entity is selected from a collection.

Rules are a collection $R = \{R_1, R_2, \dots, R_i, \dots, R_n\}$, here R_i is a single rule, n the total number of rules. A single rule is defined as $R_i = \{L_R, r_i(C, M)\}$, here: L_R – a set of all outgoing links to other elements in the model, and$r_i(C, M)$ is a rule function defined as:

$$r_i(C, M) = \begin{cases} NULL - if\ no\ value\ is\ computed \\ y - if\ value\ is\ computed\ by\ rule \end{cases}$$

here: C – context of current execution path; M – a system model; y is a computed result value, and NULL is returned if rule doesn't apply.

Rules are used to control context flow in the system. If a rule execution evaluates to an empty result the current execution path is continued. We can define the "else" path by using inversion "$!R_i$". No data will be stored in storage and no other rules will execute if the previous rule failed or returned empty value, but system flow will continue giving feedback to the user node. Rules can update the context in anyway needed for the application.

Entity collection is a collection of all data entities in the system $E = \{E_1, E_2, \dots, E_i, \dots, E_n\}$, here E_i is a single storage entity and n is the total number of storage entities. A single entity is defined as $E_i = \{D, O, L_E\}$, here: D – entity scheme definition, O – data objects, and L_E – a set of all outgoing links to other elements in the model.

Interface is a collection $I = \{I_1, I_2, \dots, I_i, \dots, I_n\}$, here I_i is a single interface and n is the total number of interfaces. A single interface is defined as $I_i = \{L_I, Q\}$, here:L_I –a set of all outgoing links to other elements in the model, Q – data query, on which the data for the interface is selected.

Table 1.
Graphical notation of UAREI modelling language

Type	Grapheme	Description
User node		Visualizes system user group. Normally a single action is triggered from this node.
Action node		Visualizes an action. Action triggers its outgoing connections. Normally actions are connected to rules and other actions
Rule node		Visualizes a rule node. Rule encloses all logic of a model. Rule triggers other rules, entities and interfaces.
Entity node		Visualizes a data entity. On triggering the node stores the data received with the current context.
Interface node		Visualizes user interfaces. Triggers user nodes finishing the feedback loop.
Connection		Visualizes relationships in the model. The direction of arrow points from the outgoing node to the incoming node.
User node		Visualizes system user group. Normally a single action is triggered from this node.

Source: The authors

5. Graphical notation of UAREI model

The UAREI model is visualized as a directed graph consisting of nodes (vertices) and links (edges) between nodes as follows: $G = \{L, N\}$, here: N is a set all nodes $N = \{N_1, N_2, \dots, N_i, \dots, N_m\} = U \cup A \cup R \cup E \cup I$; L is a set of links between nodes $L = L_U \cup L_A \cup L_R \cup L_E \cup L_I$, and L_U, L_A, L_R, L_E, L_I are collections of corresponding types of nodes $L_X = \{L_{X_1}, L_{X_2}, \dots, L_{X_i}, \dots, L_{X_{n_X}}\}$, L_i is the list of links, $L_i = (N_{out}; N_{in})$, here $N_{in}, N_{out} \in N$, L_{N_i} – are links which start N_i node.

In Table 1 we present the list of graphical symbols (graphemes) used in the UAREI model diagrams.

6. A case of study in modelling gamification in Trogon PMS

For the illustration of gamification modelling, we have selected the Trogon Project Management System (PMS) already discussed in our previous work [29,30]. Here we demonstrate how gamification rules can be described and modelled using the proposed UAREI model as well as depicted graphically using the proposed graphical notation. The gamification solution for Trogon PMS is defined as follows:

A software company employee receives random stream of tasks is coming from the project manager. There are two main types of tasks – normal tasks and tasks with badges. There are nine distinct types badges rewarded based on the tickets specificity. Everything translates to points, a certain amount of points is awarded per task done. Based on the number of badges of the same type a bonus is awarded. For

every task completed with a badge a user gets 20% bonus. When five and more of the same type badges are collected for those tasks the user is awarded with an additional 20% bonus. There is a quality element to the tasks done, if the task fails to pass Quality Assurance, a badge can be removed.

The Trogon PMS gamification is defined using the UAREI model as follows:

$$G_{TROGON} = \{$$

$$\{U_{employee}\},$$
$$\{A_{finishtask}\},$$
$$\{R_{recievepoints}, R_{recievebadge}\},$$
$$\{E_{user}, E_{badges}, E_{tasks}, E_{points}\},$$
$$\{I_{leaderboard}\}$$
$$\}$$

here:

$$U_{employee} = \{\{L_{finishtask}\}, S_{random}\}$$
$$A_{finishtask} = \{\{L_{recievepoints}, L_{recievebadge}\}, S_{random}\}$$
$$R_{recievepoints} = \{\{L_{points}\}, r_{recievepoints}(C, M) = 5\}$$

$$R_{recievebadge} = \{\{L_{points}\}, r_{recievebadge}(C, M)$$

$$= \begin{cases} \sum_{i}^{E_{points_{B_i}}} \left(E_{points_i} \cdot 1.2\right) + r_{recievepoints}(C, M) \cdot 1.4, if E_{task_i} \xrightarrow{badge} B_i andcount(E_{points_{B_i}}) = 5 \\ r_{recievepoints}(C, M) \cdot 1.4, if E_{task_i} \xrightarrow{badge} B_i andcount(E_{points_{B_i}}) > 5 \\ r_{recievepoints}(C, M) \cdot 1.2, if E_{task_i} \xrightarrow{badge} B_i andcount(E_{points_{B_i}}) < 5 \\ 0, if E_{task_i} \xrightarrow{badge} \emptyset \end{cases}$$

$$E_{user} = \{S_{user}, \{D_{John}\}, \{L_{Users}\}\}$$
$$E_{Badges} = \{S_{Badges}, \{D_1, \dots, D_9\}, \{L_{Tasks}\}\}$$
$$E_{Tasks} = \{S_{Tasks}, \{D_{B_1}, \dots, D_{B_9}, D_1, \dots D_4\}, \{L_{finishtask}\}\}$$
$$E_{Points} = \{S_{Points}, \{\emptyset\}, \{L_{leaderboard}\}\}$$
$$I_{leaderboard} = \{\{L_{users}\}, Q_{leaderboard}\}$$

$$\mathrm{R_{recievebadge}} = \{\{\mathrm{L_{points}}\}, \mathrm{r_{recievebadge}(C, M)} =$$

$$\begin{cases} \sum_{\mathrm{i}}^{\mathrm{E_{points_{B_i}}}} \left(\mathrm{E_{points_i}} \cdot 1.2\right) + \mathrm{r_{recievepoints}(C, M)} \cdot 1.4, \mathrm{if E_{task_i}} \xrightarrow{\mathrm{badge}} \mathrm{B_i andcount(E_{points_{B_i}})} = 5 \\ \mathrm{r_{recievepoints}(C, M)} \cdot 1.4, \mathrm{if E_{task_i}} \xrightarrow{\mathrm{badge}} \mathrm{B_i andcount(E_{points_{B_i}})} > 5 \\ \mathrm{r_{recievepoints}(C, M)} \cdot 1.2, \mathrm{if E_{task_i}} \xrightarrow{\mathrm{badge}} \mathrm{B_i andcount(E_{points_{B_i}})} < 5 \\ 0, \mathrm{if E_{task_i}} \xrightarrow{\mathrm{badge}} \emptyset \end{cases}$$

}here S_{user}, S_{Badges}, S_{Tasks}, S_{Points}define data schema.

In order to be made executable, a formal model has been converted into a JSON notation. This is done by writing down a JSON structure, which is composed of two parts (model nodes and model name). Every model node follows main format of name, type and links. Rules are generated by interpreting a meta-language represented as a JSON structure. The language has 1 to 1 translatable language constructions like conditions, iterations, logical operations, mathematical operations and other. Next to this the meta-language has code structural constructs. The language can be extended with necessary element to support required features.

The model of gamification of Trogon PMS using the UAREI modelling language is given in Fig. 1. The model contains:

- Entities: E_{user} – all system employee, E_{Badges} – types of badges, E_{Tasks} – the tasks which can be completed by employees, E_{Points} – points gained by the users.
- Users ($U_{employee}$) node which is a starting point for interaction with the system.
- System has only a single action ($A_{finish\ task}$) which is triggered by system users when a task is completed.

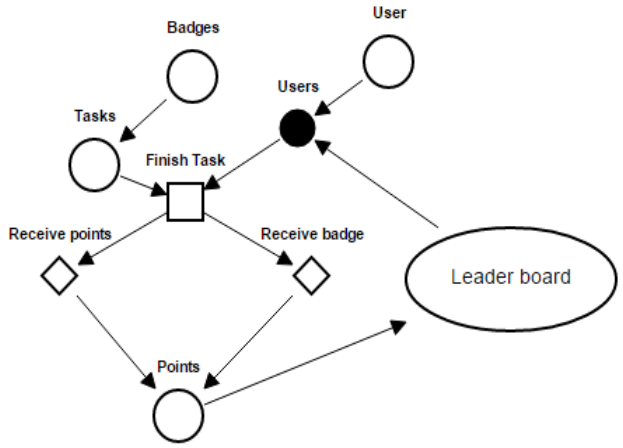


Figure 1. Visual model of Trogon PMS gamification.
Source: created by the authors

- System has two main rules: the Points rule ($R_{recieve\ points}$) describes normal behavior how user

receives the points for a completed task, and the Badge rule ($R_{recieve\ badge}$) describes how user gets points for finished tasks which have badges associated with them.

- For comparison, the UML diagram which represents the same logical flow is given (see Fig. 2) as well as the same model described using the Machinations visual notation (Fig. 3).
- User feedback loop is finished by leader board interface ($I_{leaderboard}$), which gives relevant feedback to the user.

As the UAREI model is described using the elements of the graph theory, we use the graph metrics to evaluate its visual complexity: number of nodes *N*, number of links *E*, and McCabe Cyclomatic Complexity defined as

$$M=E-N+2\cdot P,$$

here *P* is the number of independent paths in a graph.

The complexity of the UAREI and Machinations gamification models of Trogon PMS is summarized in Table 2. The comparison results show that the UAREI model is significantly less complex than its Machinations counterpart.

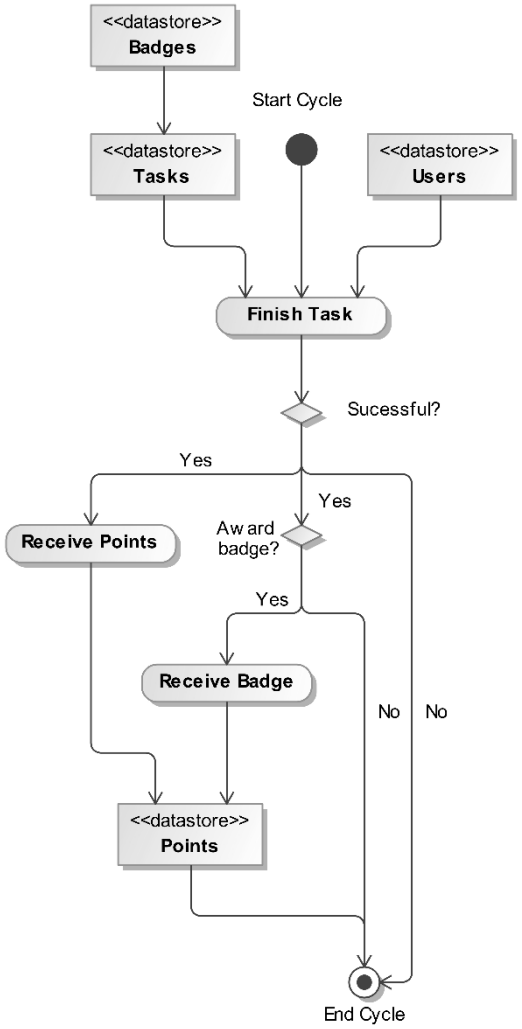


Figure 2. Gamification model of Trogon PMS specified using UML activity diagram.
Source: created by the authors

Table 2.
Visual complexity of Trogon PMS models.

Complexity metric	UAREI model	Machinations model	UML activity model
Number of nodes	9	90	11
Number of links	10	153	13
McCabe Cyclomatic complexity	3	65	4

Source: created by the authors

The computational simulation results of the proposed gamification model are presented in Fig. 4. The UML activity model is not illustrated, because UML has no simulation engine.

We assume that the system has two players ('Blue' and 'Red') with exactly the same behaviour competing at the same time. Fig. 4 shows the data recorded during such simulation.

There are two distinct parts of the simulation:

- Tie zone – from the start models are behaving similarly and both players have a similar number of points.
- Winner zone – one of the players starts winning and the other player needs time to close the gap.
- Both players can become the winner because:
- At the core of these models is a binomial distribution of a fare coin, so any player can win based on luck, while no player specific attributes are taken into consideration.
- The winner is only determined, because we stop the model at a certain time limit. In case the model goes to infinity we would end up in a tie state.
- There is some difference in the simulation data, because of different simulation execution and model specifics. In case of Machinations, a tick is executed every time the resource passes from node to node, and in the UAREI model a data record in point entity triggers a data point in the graph.

7. Evaluation

For comparative evaluation, we use the Machinations visual language [27]. As comparison criteria we use the most important problems / attributes in gamification modelling.

The game rules are supported in both UAREI and Machinations. The main difference is that Machinations only allow to build a logical structure to imitate the "rule" concept. UAREI natively supports the rule concept. Rule in the model holds the logic inside it and not disclosing its logic in model visualization. This is a main difference between these two modelling tools. The biggest problem in Machinations is that model complexity grows exponentially if one tries to model real world systems. In UAREI, most of the game logic is encapsulated in rules which decreases model complexity.

Both modelling frameworks support user-based modelling. However, in Machinations every user behaviour model has a separate copy of the model. UAREI natively supports multiple users working with the same model in parallel. Machinations currently support logical attributes which describe user behaviour. UARSEI currently does not have such modelling capacities.

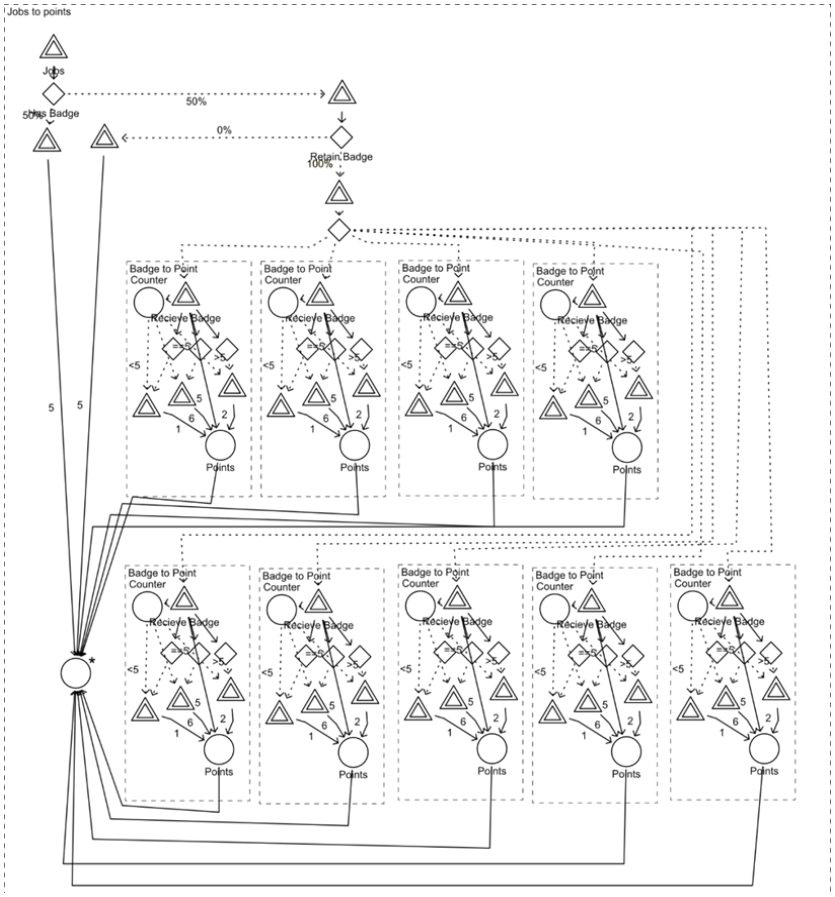


Figure 3. Gamification model of Trogon PMS specified using Machinations.
Source: created by the authors

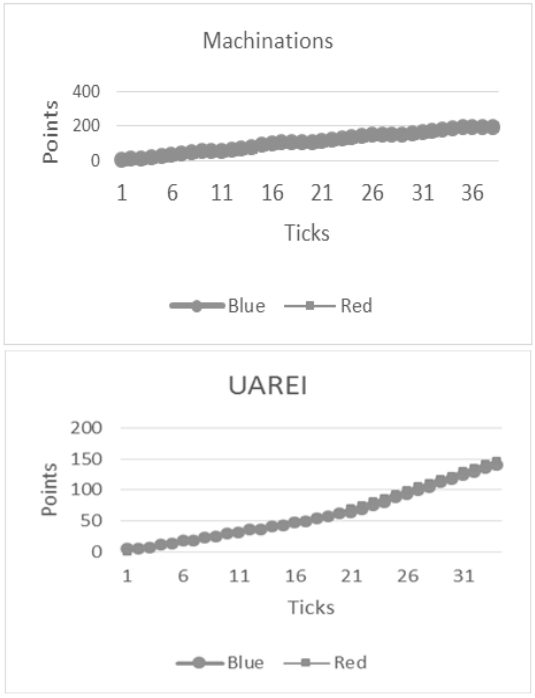


Figure 4. Simulation of game results using Machinations and UAREI.
Source: created by the authors

Machinations is based on the economic functions and the resource concept. UAREI natively focuses on real data entities which carry more information. UAREI has the "context" concept which is carried through the model execution flow. In general, the context concept of UAREI is similar to Machinations resource concept.

UAREI supports real world data entities and that allows mapping into software domain. UAREI separates actual data from the actual model. Normally in software engineering this is a common way to ensure data-program separation, the same concept is encapsulated into UAREI. Machinations does not have a concept of data.

Machinations does not have any model transformation capabilities and it never was designed for this goal. On the other hand, UAREI is designed for transformation into executable code. The rule logic is written in a meta-language which is processed into executable Javascript code. Other model parts are executed using a simulator.

Both UAREI and Machinations have minimal analysis tools which allow to view model data. In Machinations one is able to view "pool" changes over time. In UAREI one is able to see interface data change over time.

UAREI has a native feedback loop in the system. The modelling framework is designed to ensure feedback to model users. In Machinations it is up to designer to setup such loop to model user behaviour during simulation.

Table 3.
Graphical notation of UAREI modelling language

Property	UAREI	Machinations	UML
Game rules	Native support	Logical support	Native and Logical support
Visual model complexity	Medium	High	Medium
User based simulation	Able to simulate any number of users	Every simulation is a copy of the model	No simulation
Real data support	Able to use real data entities	The only data used are resources	Able to define real entities
Data-Model separation	Data is separated from the model, so it is possible to use any dataset	Data is directly encapsulated in the model	Data is not part of the model
Model transformation	Future work	Model has no functionality to generate executable code	Possible to convert to code
Feedback loop	Has a native support feed back loops	It is possible to simulate a feedback loops directly into model	No feedback loops
Model reusability	Does not support yet	Importing is the only functions which allows incorporating other models.	Full support
Abstraction level	Higher	Designer-dependant	Designer-dependant

Source: created by the authors

Table 4.
Cognitive dimensions of UAREI and Machinations.

Property	UAREI	Machinations	UML
Abstrac-tion gradient	Model itself has single level abstraction, but level of details needed to specify is chosen by user. Rules and interfaces encapsulate logic.	User chooses the level of abstraction. The more details are represented the more complex model is build. One can build reusable parts of the model.	User customizes the abstraction level by choosing which modelling tools to incorporate.
Closeness of mapping	Straightforward model. Problems appear while transcribing form formal to JSON model.	One needs to learn how to build complex logic. It works very well if you exchange parts of the logic with simplifications. Also one needs to understand the four economic function paradigm.	Straightforward modelling language which allows different levels of abstraction.
Consisten-cy	The whole language is built on top of 6 elements. After learning these constructs you can build any system. Hardest part are query and rule logic function writing, which need to be learned separately.	The language itself is quite wide. It consists of 15 different elements and a lot of settings. The hardest part is implementing out complex logic, because model lacks of programmable logic nodes.	UML activity diagram language used in this case of study are composed of over 20 different types of elements. Which allows to build many concepts into the model.
Diffuse-ness	Six graphic elements make up the language.	17 constructs allow to build almost anything one needs for game modeling.	Over 20 elements and multiple types of connections.
Error-proneness	Errors originated from rule and query specification.	We didn't find error possibilities in small models. Problems would arise with big and complex models.	Low error-proneness. Model supports aggregation difficulty can be divided.
Hard mental operations	Writing in JSON notations at some point would build to hard structures to follow easily.	If a model has many asynchronous operations or high number of nodes it can be hard to follow.	Easy language with real natural meaning. Tracing the model requires hard mental operations.
Hidden dependen-cies	Dependencies are clearly visible because you see all incoming and outgoing connections.	Dependencies are clearly visible, but can be harder to understand due to specified logic on connections	Dependencies are clearly visible.
Premature commit-ment	No premature commitment	No premature commitment	Need to be committed to UML to optimize benefits.
Progressi-ve evaluation	At any point the model can be executed if is in valid form.	At any point the model can be evaluated.	Model has no automated evaluation.
Role expressive-ness	The system dependencies are clearly visible.	The system dependencies are clearly visible, but can be hard to interpret.	System dependencies can be hard to deduct.
Secondary notation	Allows only label notation.	Allows label, colour notations.	Allows labels and comments.
Viscosity	Any change is not harder to do as initially.	Can be harder to restructure complex rules.	Changes might be harder to introduce, depends on complexity.
Visibility	It is possible to view a model until fits on the screen. Problems occur when the model is too big to fit on the screen. JSON notation of a complex rule can be hard to follow.	Until the model is simple enough there are no problems. Problems arise with large models which don't fit in the screen and after some point zooming out doesn't help.	Complexity is decreased by decomposition into smaller parts. In large systems it can get quite hard to follow whole system model.

Source: created by the authors

8. Conclusions

In this paper we have presented the description of the UAREI modelling framework. We have demonstrated a case of study in modelling the Trogon PMS gamified application using UAREI. The same gamified application was modelled using the Machinations framework and UML activity diagrams. All modelling frameworks are good tools for

modelling gamification of software systems.

All analysed models were used to compare their visual complexity. We run a sample simulation of two players using the system under UAREI and Machinations. The comparison disclosed the benefits and weakness of the modelling frameworks in question as follows.

The advantages of the UAREI model are a high level of abstraction, native support for feedback, model transformation to executable code, explicit separation of data and code. The disadvantage of the UAREI model is that currently it still does not support reusability.

Future work will focus on improving properties of the UAREI model in supporting model transformation, analysis and reusability.

References

[1] Wortley, D., Gamification and geospatial health management, Proc. of 7th IGRSM International Remote Sensing & GIS Conference and Exhibition IOP Publishing. IOP Conf. Series: Earth and Environmental Science, 20, 2014. DOI: 10.1088/1755-1315/20/1/012039

[2] Gartner Group, Gartner Says By 2015, More Than 50 Percent of Organizations That Manage Innovation Processes Will Gamify Those Processes. 2011.

[3] Vasquez, O.C., Sepulveda, J.M., Alfaro, M.D. and Osorio-Valenzuela, L., Disaster response project scheduling problem: A resolution method based on a game-theoretical model, International Journal of Computers Communications & Control, 8(2), pp. 334-345, 2013. DOI: 10.15837/ijccc.2013.2.313

[4] Caponetto, I., Earp, J. and Ott, M., Gamification and education: A literature review, Proc. of the 8th European conference on games ECGBL 2014, pp. 50-57, 2014.

[5] Bothsa, A., Herselman, M. and Ford, M., Gamification beyond badges, IST-Africa Conference Proceedings, IEEE, pp. 1-10. 2014.

[6] Gené, O.B., Martínez, M. and Blanco, F., Gamification in MOOC: challenges, opportunities and proposals for advancing MOOC model, Proc. of the 2nd Int. Conference on Technological Ecosystems for Enhancing Multiculturality, 2014, pp. 215-220. DOI: 10.1145/2669711.2669902

[7] Sammut, R., Seychell, D. and Attard, N., Gamification of project management within a corporate environment: An exploratory study. Proc. of 6th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES), pp. 1-2. IEEE, 2014. DOI: 10.1109/vs-games.2014.7012158

[8] Freudmann, E.A. and Bakamitsos, Y., The role of gamification in non-profit marketing: An information processing account, Procedia - Social and Behavioral Sciences, 148, pp. 567-572, 2014. DOI: 10.1016/j.sbspro.2014.07.081

[9] Wilson, A.S. and McDonagh, J.E., A gamification model to encourage positive healthcare behaviours in young people with long term conditions, EAI Endorsed Trans. Serious Games 2: e3, 2014. DOI: 10.4108/sg.1.2.e3

[10] da Conceicao, F.S., da Silva, A.P., de Oliveira-Filho, A.Q. and Silva-Filho, R.C., Toward a gamification model to improve IT service management quality on service desk, Proc. of 9th International Conference on the Quality of Information and Communications Technology (QUATIC), IEEE, pp. 255-260. 2014. DOI: 10.1109/quatic.2014.41

[11] Hall, M., Kimbrough, S.O., Haas, C., Weinhardt, C. and Caton, S., Towards the gamification of well-being measures, IEEE 8th International Conference on E-Science (e-Science), 2012, pp. 1-8.

[12] Larsson, R.S., Motivations in sports and fitness gamification: A study to understand what motivates the users of sports and fitness gamification services, MSc Thesis, Umea Universitet, Umeå, Suecia, 2013.

[13] Wells, S., Kotkanen, H., Schlafli, M., Gabrielli, S., Masthoff, J., Jylhä, A. and Forbes, P., Towards an applied gamification model for tracking, Managing, & Encouraging Sustainable Travel Behaviours. ICST Trans. Ambient Systems 4: e2, 2014. DOI: 10.4108/amsys.1.4.e2

[14] Bauckhage, C., Kersting, K., Sifa, R., Thurau, C., Drachen, A. and Canossa, A., How players lose interest in playing a game: An empirical study based on distributions of total playing times. IEEE Conference on Computational Intelligence and Games (CIG), 2012, pp. 139-146. DOI: 10.1109/CIG.2012.6374148

[15] Tenzer, J., Improving UML design tools by formal games, Proc. of 26th Int. Conference on Software Engineering (ICSE 2004), IEEE, pp. 2004, 75-77. DOI: 10.1109/icse.2004.1317428

[16] Hetherinton, D., SysML requirements for training game design, Proc. of IEEE 17th International Conference on Intelligent Transportation Systems (ITSC), 2014, pp.162-167. DOI: 10.1109/ITSC.2014.6957684

[17] Herzig, P., Jugel, K., Momm, C., Ameling, M. and Schill, A., GaML-A modeling language for gamification, Proc. of IEEE/ACM 6th Int. Conference on Utility and Cloud Computing, 2013, pp. 494-499. DOI: 10.1109/ucc.2013.96

[18] Matallaoui, A., Herzig, P. and Zarnekow, R., Model-Driven serious game development integration of the gamification modeling language GaML with unity, Proc. of 48th Hawaii International Conference on System Sciences (HICSS), IEEE, 2015, pp. 643-651. DOI: 10.1109/hicss.2015.84

[19] Janssens, O., Samyny, K. and Hoecke, S., Educational virtual game scenario generation for serious games, Proc. of IEEE 3rd Int. Conference on Serious Games and Applications for Health (SeGAH), IEEE, 2014, pp. 1-8. DOI: 10.1109/segah.2014.7067106

[20] Nummenmaa, T., Berki, E. and Mikkonen, T., Exploring games as formal models, In 4th South-East European Workshop on Formal Methods (SEEFM), IEEE, 2009, pp. 60-65. DOI: 10.1109/seefm.2009.15

[21] Kim, J.T. and Lee, W.-H., Dynamical model and simulations for gamification of learning, International Journal of Multimedia and Ubiquitous Engineering, 8(4), pp. 179-190, 2013.

[22] Bista, S.K., Nepal, S., Colineau, N. and Paris, C., Using gamification in an online community, CollaborateCom 2012, 2012, pp. 611-618.

[23] Chan, K.T., King, I. and Yuen, M.-C., Mathematical modeling of social games, International Conference on Computational Science and Engineering, CSE'09, IEEE, 4, 2009, pp. 1205-1210. DOI: 10.1109/cse.2009.166

[24] de Oliveira, G.W., Julia, S. and Soares-Passos, L.M., Game modeling using WorkFlow nets, in Proc. of IEEE International Conference on Systems, Man and Cybernetics (SMC), IEEE, 2011, pp. 838-843. DOI: 10.1109/icsmc.2011.6083757

[25] Agustin, M., Chuang, G., Delgado, A., Ortega, A. Seaver, J. and Buchanan, J.W., Game sketching, Proc. of the 2nd int. conference on digital interactive media in entertainment and arts, 2007, pp. 36-43.

[26] Smith, A.M., Nelson, M.J. and Mateas, M., Ludocore: A logical game engine for modeling videogames, in: IEEE Symposium on, Computational Intelligence and Games (CIG), 2010, pp. 91-98. DOI: 10.1109/itw.2010.5593368

[27] Dormans, J., Machinations: elemental feedback patterns for game design, in: Saur, J. and Loper, M. (eds.), GAME-ON-NA 2009: 5th Int. North American Conference on Intelligent Games and Simulation, pp. 2009, 33-40.

[28] van Rozen, R. and Dormans, J., Adapting game mechanics with micro-machinations, Proc. of the 9th Int. Conference on the Foundations of Digital Games, 2014.

[29] Ašeriškis, D. and Damaševičius, R., Gamification patterns for gamification applications, Procedia Computer Science: 39, pp. 83-90, 2014. DOI: 10.1016/j.procs.2014.11.013

[30] Ašeriškis, D. and Damaševičius, R., Gamification of a Project management system, Proc. of Int. Conference on Advances in Computer-Human Interactions ACHI2014, 2014, pp. 200-207.

[31] Green, T.R.G. and Petre. M., Usability analysis of visual programming environments: A 'cognitive dimensions' framework, Journal of Visual Languages & Computing, 7(2), pp. 131-174, 1996. DOI: 10.1006/jvlc.1996.0009

[32] Rao, V., Heuristic evaluation of persuasive game systems in a behavior change support systems perspective: Elements for discussion. Second International Workshop on Behavior Change Support Systems – BCSS, 2014, pp. 21-25.

[33] Grünvogel, S.M., Formal models and game design, Game Studies 5(1), pp. 1-9, 2005.
[34] Devedzić, V., Understanding ontological engineering, Communications of the ACM, 45(4), pp. 136-144, 2002. DOI: 10.1145/505248.506002
[35] Tabuada, P., Pappas, G.J. and Lima, P., Compositional abstractions of hybrid systems. Discrete Event Dynamic Systems, 14(2), pp. 203-238, 2004. DOI: 10.1023/B:DISC.0000018571.14789.24
[36] Narayanasamy, V., Wong, K.W., Rai, S. and Chiou, A., Complex game design modeling, Cultural Computing, Springer, 333, pp. 65-74, 2010. DOI: 10.1007/978-3-642-15214-6_7
[37] Salen, K. and Zimmerman, E., Rules of play - game design fundamentals, The MIT Press, Cambridge, MA, USA, 2004.
[38] Vega-Zazueta, L., Modélisation et analyse spatiale et temporelle des jeux vidéo basées sur les réseaux de Pétri, PhD Thesis, Paris, France, 2004.

D. Ašeriškis, is MSc. degree in Software Systems Engineering in 2013 from Kaunas University of Technology. His research interests include: gamification related simulation, modeling and development.
ORCID: 0000-0002-6309-8784

T. Blažauskas, holds the PhD degree from Kaunas University of Technology. He is author of many scientific articles, participated in numerous scientific projects related mostly with software engineering, e-learning. His main research fields are smart user interfaces, service oriented architecture, e-learning amd e-security.
ORCID: 0000-0003-2858-328X

R. Damaševicius, received PhD. degree in Informatics Engineering in 2005 from Kaunas University of Technology, Lithuania. He currently works as a Professor at Faculty of Informatics, Software Engineering Department, KTU. He is an author / co-author of over 120 refereed scientific articles (80 in Thomson Reuters Web of Science). His research interests include software engineering, artificial intelligence methods, brain-computer interface.
ORCID: 0000-0001-9990-1084