The final publication is available at Springer via https://doi.org/10.1007/978-3-031-21488-2_3

Supporting the Individuation, Analysis and Gamification of Software Components for Acceptance Requirements Fulfilment

Recommended Citation:

F. Calabrese, L. Piras, and P. Giorgini, “Supporting the Individuation, Analysis and Gamification of Software Components for Acceptance Requirements Fulfilment”, in 15th International Conference on the Practice of Enterprise Modeling (PoEM). Springer, 2022.

Supporting the Individuation, Analysis and Gamification of Software Components for Acceptance Requirements Fulfilment

Federico Calabrese[1], Luca Piras[2] and Paolo Giorgini[1]

[1] Department of Information Engineering and Computer Science,
University of Trento, Trento, Italy
federico.calabrese@alumni.unitn.it
paolo.giorgini@unitn.it
[2] Department of Computer Science,
Middlesex University, London, United Kingdom
L.Piras@mdx.ac.uk

Abstract. In the last few years, Gamification has proved effective for motivating users in using software systems. Gamification Engineering has been proposed as a systematic way to gamify systems. Goal-Oriented Requirements Engineering (GORE) techniques have been emerging for supporting the gamification of a system from the initial stages. However, the gamification of a system itself proved not effective, unless requirements engineers design the gamification solution being driven by characterising users, the context, and considering factors and strategies coming from Social Sciences (e.g., Psychology, Sociology, Human Behaviour, Organisational Behaviour). GORE Gamification Engineering techniques have been enhanced for supporting such concepts, referred to as Acceptance Requirements, and the Agon Framework, with its systematic Acceptance Requirements Analysis Based on Gamification, has been proven effective in different EU Projects. However, according to engineers we interviewed in our projects, some GORE gamification activities remain difficult and require further support. In this paper, our contributions are: **(i)** individuating such activities and providing lessons learned, **(ii)** considering a crucial activity, i.e. individuation of software components to gamify and how to gamify them, and proposing a solution for this. Our solution is called *Supporting the individuation, analysis, and GAMification* (SiaGAM) algorithm. To evaluate SiaGAM, we considered the gamification results of 5 EU projects, compared them with the results of applying SiaGAM, and we found that SiaGAM is effective in supporting the engineer in individuating software functions to gamify.

Keywords: Requirements Engineering · Acceptance Requirements · Gamification · Goal Models · Goal Modeling Analysis · Software Engineering

1 Introduction

Gamification is becoming a crucial element to consider when designing software systems to engage the user and to stimulate them to use the system [1, 12, 18]. Gamification has been defined by Deterding et al. as "the use of game design elements in non-game contexts" [6]. Non-game contexts are environments whose purpose is not to have fun but to induce a certain behaviour in the user [7]. This behaviour allows to fulfill the goals of the environment. A potential way to encourage the user to embrace such behaviour is to use game design elements [6]. This means that it is possible to decorate system functions with game elements, thus making software more attractive, interesting, and engaging for the user [5, 25].

Accordingly, in the Software Engineering area, the important term and concept of "Gamification Engineering" has been proposed recently and, as far as we know, the first related definition comes from Piras et al. "Gamification Engineering is the Software Engineering of Gamification" [17, 22]. Gamification Engineering includes new languages, engines, models, frameworks, and tools [9, 17--19, 21--23, 25] for making the gamification of software systems more systematic and supported. In relation to Requirements Engineering, Goal-Oriented Requirements Engineering (GORE) techniques emerged for supporting the gamification of a system from the initial stages. However, the gamification of a system itself has proved not effective unless requirements engineers design the gamification solution being driven by user characterisation and considering factors and strategies coming from Social Sciences (e.g., Psychology, Sociology, Human Behaviour, Organisational Behaviour) [5, 7, 16--19, 25]. Consequently, GORE Gamification Engineering techniques have been enhanced to support such concepts, referred to as Acceptance Requirements. In parallel, the Agon Framework, with its systematic Acceptance Requirements Analysis Based on Gamification, proved effective in different EU Research Projects [11, 17--19, 23].

In this context, according to the perspective of requirements analysts, the current GORE gamification engineering techniques are considered useful [11, 17--19, 23]. However, requirements analysts also identified that some GORE gamification activities, although useful, remain difficult and require further support or improvement [18]. Accordingly, we derived the following Research Questions (RQs), which we address in this paper:

RQ1. What are the current gamification supporting phases that can be improved or automated to better support the analyst in the gamification of software systems?

RQ2. How can we support the analyst in the analysis and individuation of software functionalities to gamify in a guided/supported/automated way?

To answer **RQ1**, we identified within our projects [11, 17--19, 23] the activities that are still complex and difficult to carry on, and thus require support or improvement. We obtained such insights by observing analysts directly using GORE gamification engineering techniques, and by interviewing them via questionnaires.

RQ2 concerns the investigation on how to further support analysts regarding problematic activities found through **RQ1**. The most problematic activity, indicated by analysts as requiring more support [18], concerns the identification of the *subset* of software components to gamify. This is a crucial aspect because the gamification of the *entire* software system can be expensive and could not give the expected outcome [5, 25]. Thus, to minimise organisations' effort and costs related to gamification design activities, it is necessary to identify what *subset* of system functionalities it is better to gamify [5, 25]. In this paper, the solution we devised to address **RQ2** is an algorithm that we call: *Supporting the individuation, analysis, and GAMification* (SiaGAM) of software components. SiaGAM guides and supports the analyst in a semi-automated way by: **(i)** representing the software to be gamified as a goal model; **(ii)** characterising software functionalities in relation to different qualities; **(iii)** guiding during functionalities annotation, using the algorithm in a semi-automated way, to identify the set of functionalities to consider to gamify according to criteria identified.

To evaluate SiaGAM, we considered the gamification results obtained in 5 case studies from EU projects [11, 17--19, 23]. We focused on the subsets of functionalities that had to be gamified, and had been selected manually by the researchers. We then applied SiaGAM to the same goal models and obtained subsets of functionalities to consider to be gamified. Finally, we compared our results with the previous ones, and we found that SiaGAM is effective in supporting the analyst in identifying software functionalities to gamify, as it can derive the same set of functionalities. It is important to note that the results of the projects had been obtained manually: analysts highlighted this aspect as the one needing further support [18]. An essential advantage of SiaGAM is that it offers analogous results in a guiding, supporting, and semi-automated way.

The rest of the paper is organised as follows: section 2 addresses **RQ1**, discussing lessons learned identifying which gamification activities require more support and how to provide it. Section 2 also provides the basis and motivation for **RQ2**, explaining the context in which our algorithm and approach are applied. Section 3 addresses **RQ2** describing our algorithm and approach. Section 4 addresses **RQ2** by describing our case study and the evaluation of SiaGAM. Section 5 discusses related work, while section 6 concludes this work.

2 Motivation and Lessons Learned

In this section we address **RQ1**. Specifically, in 2.1 we outline the Agon Framework and its systematic Acceptance Requirements Analysis based on Gamification, focusing on the phases relevant to our RQs. In 2.2, we summarise research projects where Agon has been applied successfully, and address **RQ1** by discussing Lessons Learned (LL) we derived from such real-world experiences.

2.1 Agon Framework

Agon [17, 18, 21--23] is a framework for performing systematic Acceptance Requirements analysis on software systems based on modeling, analysis and strategies to fulfil such requirements using Gamification operationalisations. Agon allows to analyse and gamify any kind of software, as demonstrated in many domains [17, 18, 21--23]. For instance, it has been applied successfully to a real case study within the PACAS EU Project [17, 23] to analyse Acceptance Requirements and gamify the PACAS platform in the context of architectural change management for ATM (Air Traffic Management) systems.

Agon is composed of a multi-layer meta-model (Fig. 1), based on Goal Modeling, extending the NFR framework [4, 13, 15]. It has been designed with

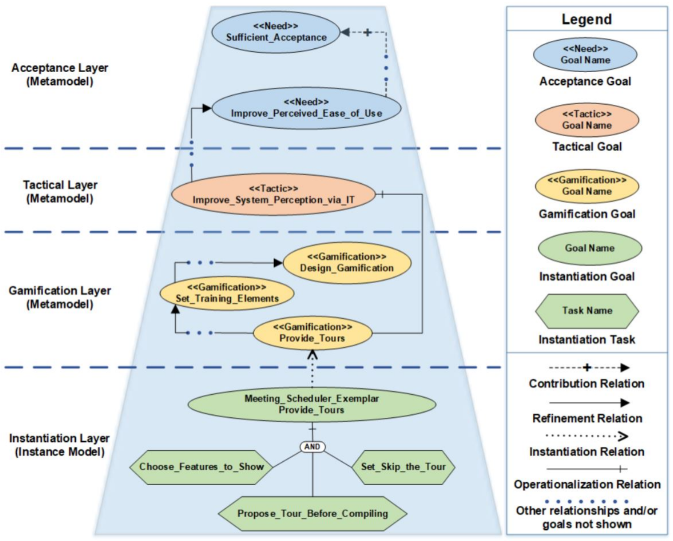
Acceptance Layer (Metamodel)
<<Need>> Sufficient_Acceptance
<<Need>> Improve_Perceived_Ease_of_Use
Tactical Layer (Metamodel)
<<Tactic>> Improve_System_Perception_via_IT
Gamification Layer (Metamodel)
<<Gamification>> Design_Gamification
<<Gamification>> Set_Training_Elements
<<Gamification>> Provide_Tours
Instantiation Layer (Instance Model)
Meeting_Scheduler_Exemplar Provide_Tours
AND
Choose_Features_to_Show
Set_Skip_the_Tour
Propose_Tour_Before_Compiling
Legend
<<Need>> Goal Name
Acceptance Goal
<<Tactic>> Goal Name
Tactical Goal
<<Gamification>> Goal Name
Gamification Goal
Goal Name
Instantiation Goal
Task Name
Instantiation Task
Contribution Relation
Refinement Relation
Instantiation Relation
Operationalization Relation
Other relationships and/or goals not shown

Fig. 1: Simplified example of Agon Multi-Layer Meta-Model [17, 23], descriptions of the example available at [17, 21, 22], and the Agon full models available at [20]

the next models [17, 23]: **(i)** Acceptance Meta-Model (AM) capturing user needs, which the analyst can consider to determine aspects able to stimulate the target user, to accept and use a software system; **(ii)** Tactical Meta-Model (TM), which allows the analyst to identify tactical goals, as refinements for needs selected in AM [21]; **(iii)** Gamification Meta-Model (GM), which provides the analyst with ready-to-use, game design solutions operationalising tactical goals and (in turn) needs selected in the higher-level models (AM and TM) [21]; **(iv)** Instance Model (IM), produced by the analyst via a goal modeling language. This instantiates GM by defining tasks, relations and lower-level gamification goals. The final purpose of IM is to decorate software functions by instantiating and configuring ready-to-use game-based elements (suggested by Agon and chosen also taking into account the specific software domain) to stimulate the target user [17, 18, 21, 23].

The next 2 subsections briefly illustrate the first 2 phases of the Agon process, to which we applied SiaGAM to further support the analyst addressing **RQ2**. The third subsection summarises the other phases of the Agon process, which are relevant for discussing our lessons learned (**RQ1**).

Phase 1: Base System Requirements Modelling (BSRM). This is the first activity analysts performs with the Agon process. The aim is to analyse the

software to gamify, for creating a Goal Model representing the system and its functionalities [17,18]. Those will be potential functionalities to gamify through analysis of acceptance requirements, tactics and game design strategies [17,18,23].
Phase 2: Acceptance Requirements Elicitation Analysis (AREA). In this phase the BSRM goal model is analysed for identifying the *subset* of functions to gamify. This is performed manually by the analyst and Agon just provides recommendations. Specifically, the identification should aim at selecting functions of the software that meet the following conditions: "***(a)*** *cannot be fulfilled automatically by IT procedures;* ***(b)*** *needs human contribution;* ***(c)*** *has to stimulate and engage the user to carry out the activity (e.g., the activity is boring, complex, repetitive, etc.);* ***(d)*** *contributes positively to the achievement of critical purposes of the system and depends on human contribution to be fulfilled.* " [17,18].
Next Phases of the Agon Process. The other phases relevant to our lessons learned (**RQ1**) are [17,18]: Phase 3, Context Characterization; Phase 4, Context-Based Analysis of Acceptance Requirements; Phase 5, Acceptance Requirements Refinement; Phase 6, Context-Based Operationalization via Gamification; Phase 7, Domain-Dependent Instantiation of Incentive Mechanisms.

2.2 Activities and Lessons Learned

The Agon Framework, with its systematic Acceptance Requirements Analysis Based on Gamification, proved effective in different EU Research Projects [11, 17--19, 23]. Specifically, according to the perspective of requirements analysts, the current GORE gamification engineering techniques, included in Agon and its process, have been recognised as useful [11,17--19,23]. The case studies we consider refer to the following 5 research projects[1] [11,17--19,23]: **(i)** in the context of the VisiOn EU project[1] [18], a complex platform for Public Administration (PA) has been delivered to support citizens in improving their awareness on privacy aspects of their personal data and to support them and organisations in the management of personal data. In this case study, a core tool of this platform, STS-Tool, has been gamified to engage analysts in complex modeling-related scenarios for the analysis of privacy and security requirements [2,17,18]; **(ii)** within the PACAS EU project[1] [17,23], a participatory system that supports collaborative decision-making and related to Air Traffic Management (ATM) has been delivered and gamified [17,23]. In this case, it was important to motivate heterogeneous professionals to collaborate to improve complex ATM procedures. The aim was to support the analysis and design of strategies and to improve the interest, engagement and pro-activeness of the professionals involved [17,23]; **(iii)** within the SUPERSEDE EU project[1] [17,19], the focus was the gamification of a system to motivate analysts to contribute pro-actively to *Collaborative Requirements Prioritization* activities [17,19]; **(iv)** the SUM project[1] [11,17] gamified a system providing a solution to encourage citizens to use *Sustainable urban Mobility* [11, 17]; **(v)** the MA4C project[1] [11, 17] gamified a system to promote *Mobility*

[1] VisiOn, PACAS, SUPERSEDE, SUM and MA4C Case Studies available at: `https://pirasluca.wordpress.com/home/acceptance/case-studies/`

Assistance for Children to progressively support the child in autonomous growth to avoid manifesting different problems as adults [11,17].

In total, we involved 32 participants as requirements analysts (5 junior in a preliminary phase, then 21 junior and 4 senior) within case studies, using semi-structured interviews, questionnaires and evaluation reports (full details available at [17]). Even though requirements analysts found these activities useful, they declared that some of them remained difficult and require further support or improvement [18]. To identify the activities needing further development, we observed the analysts while using GORE techniques and we interviewed them using questionnaires. This allowed us to address **RQ1**, and in the next subsections we will show the activities found and provide the lessons learned.

Analysis on Large Models and Concepts Interpretation. Among the difficulties, analysts found problematic to interpret AM, the Goal Model provided by Agon representing acceptance solutions (Fig. 1). Junior analysts have been the most affected, having less knowledge and experience related to goal models, and Social Sciences concepts, while senior analysts faced fewer difficulties [17,18]. In this context, a glossary[2] of the concepts, and explanations from us, helped with the interpretation. Similarly, further explanations were needed to mitigate analysis issues regarding the Agon models structure. Furthermore, when the analysis moved towards the Gamification Model (Fig. 1), which is an even larger model, they started experiencing difficulties again. In this case, the issue was not related to the interpretation of gamification concepts (also thanks to the glossary), but more related to the dimension of the model. Again, the problem was more evident in junior than in senior analysts [17,18].

In summary, analysts recognised the models and the overall systematic process as useful; however, the approach still needs improvements. To mitigate the interpretation problem, the glossary and explanations helped. For the size of goal models, analysts suggested abstraction layers on goal models, or concepts separated in different perspectives, and proposed at different times. Regarding abstractions layers, analysts would like graphical visualisations able to show simplified models, or alternative summarised graphical solutions. For the perspectives, we could consider categorising parts of the models, and devising strategies for proposing solutions and concepts at different stages, making users to focus on a separate aspect at a time. However, such strategies should also consider situations where separate parts are involved and contribute to the same solution.

Instantiation of Gamification Solutions. The last phase ‘‘Domain-Dependent Instantiation of Incentive Mechanisms’’ [17] requires a huge effort from the analyst. Indeed, since GM is already a large model, its instantiation (Fig. 1) can result in a even larger model, and it is up to the analyst to instantiate it. At the moment, this is a manual activity supported by a graphical editor. A potential solution for further supporting the analyst can be to enhance the editor with further functionalities suggesting typical instantiations of high-level gamification solutions, according to the domain, as low-level patterns. Another solution could be to enhance this phase with automatic code generation, or skeleton code, based

[2] Agon Glossary: `https://pirasluca.wordpress.com/home/acceptance/glossary/`

on the analyst's instantiation. In this way, the instantiation work could be merged with the development phase. Moreover, the process can be iteratively refined.
Individuation of Software Components to Gamify. This lesson learned is related to the first two phases, discussed above (BSRM and AREA phases), of the systematic Acceptance Requirements Analysis Based on Gamification of the Agon Framework. These are crucial phases where the analyst needs to select the *subset* of system functionalities to gamify. Agon provides high-level guidelines for this, but no further support. In the research projects considered [11,17--19,23], analysts have generally recognised Agon and its process as useful and have been able to properly identify the functionalities to gamify, designing effective gamification solutions. However, they highlighted the need for further guided and automated support concerning these crucial phases. Agon indicates specifically for this part that the analyst should identify the functions that need to be gamified through the analysis of BSRM. To support the identification, Agon provides guidelines to select software functions satisfying certain conditions (see "Phase 2", section 2.1). In this paper, to address **RQ2**, we focus on such critical phases (BSRM and AREA). Based on the feedback of the analysts and by following the high-level guidelines of Agon, we have designed the SiaGAM algorithm. SiaGAM guides the analyst (semi-automatically) in characterising system functionalities, identifying qualities, and applying annotations for identifying functions to gamify.

3 SiaGAM Algorithm and Process

This section addresses **RQ2** by describing the algorithm we implemented within Agon, to further support the analyst in relation to *Supporting the individuation, analysis, and GAMification* (SiaGAM) of software components. Agon is an Acceptance Requirements and Gamification framework for gamifying software systems by fulfilling User Acceptance. In this context, SiaGAM and our approach improve a crucial part of the Agon process and tool: SiaGAM provides semi-automatic support to BSRM and AREA phases of Agon (addressing the last lesson learned, discussed above). BSRM and AREA are early phases of the Agon process. These are crucial for building an effective gamification solution. Redesigning such phases as semi-automatic can provide further support to analysts, both reducing their effort and minimising errors occurrence. To make such phases semi-automatic (**RQ2**), we devised modifications of BSRM and a redesign of AREA to make it more goal modeling oriented. The idea was to change the process by guiding the analyst to characterise and annotate a goal model of the system (BSRM enhancement) with factors that SiaGAM can consider for providing solutions as a filtered goal model based on choices made by the analyst (redesign of AREA). The process can be reiterated to produce further refinements of the solution. We designed the SiaGAM process in 2 phases, i.e. *Annotation Model* and *Algorithm Execution*. These are illustrated in the following.
Annotation Model. This phase extends BSRM, and the outcome expected is an annotated goal model. The analyst designs the goal model by representing the system to gamify, and follows guidelines allowing to assign annotations to

concepts of the model. This step is fundamental since the analyst, who has a deep knowledge of the software, can find and annotate the most relevant system functions. Annotations are expressed using human language (making the assignment of annotations easier) and indicated/supported by SiaGAM (these are extensible). Thus, annotations support the analyst in the system functions characterisation. Below, we outline annotations and related guidelines to consider for using our approach by applying annotations properly.

- **B**: indicates `Boring`. This includes unattractive, tedious, monotonous or repetitive system functions. These are usually interesting from the gamification perspective because they could require users to be motivated to use them.
- **HC**: indicates `High Complexity`. These functions could demand high knowledge, skills, or require engaging or collaborating with other people, which could increase the difficulty. They could require also significant effort by the user. These functions are the most complex of the software and could require considerable stimuli to allow adequate user engagement.
- **LC**: indicates `Low Complexity`. These functions could require a limited amount of time to be completed, or indicate no need to collaborate with other people, or can be simple actions (e.g., clicking a confirmation button). This annotation identifies simple tasks that could require little attention, but that could still need to be gamified to motivate the user to complete them.
- **A**: indicates `Automatic`. This annotation is assigned to a system function that does not require any effort/intervention by the user.
- **M**: indicates `Manual`. This includes activities fully performed manually by the user (the goal model represents both "Socio" and "Technical" activities) or semi-automatically (i.e. where more than 50% is performed manually).

Accordingly, we specify an Annotations Set (AS) composed by {`B, HC, LC, A, M`}. Furthermore, we designed SiaGAM to be flexible enough to enable the analyst to extend AS, by adding new annotations, in case other characteristics of particular software domains/contexts need to be considered. Another concept we included in our approach is called `Gamification Impact Probability` (GIP). This represents the probability of a function to receive a positive impact if gamified, i.e. the higher the GIP, the more likely the related function will be considered for gamification. However, the GIP that SiaGAM considers is calculated on a path, which we call `GIPPath`. The range considered is: $0 \leq \mathit{GIPPath} \leq 1$. From our experience in gamifying software successfully in different EU projects [11,17--19,23], and according to the feedback we received by requirements analysts involved, we determined the default SiaGAM GIP values as: $\mathtt{B} = 0.5$; $\mathtt{HC} = 1$; $\mathtt{LC} = 0.25$; $\mathtt{A} = 0$; $\mathtt{M} = 0.8$. Such values, as well as AS, are configurable based on specific domain/context characteristics, needs, constraints.

Another annotation is `Current Engagement Probability` (CEP). This represents the current probability of user engagement recorded by a particular function. CEP is summed to GIPPath during its calculation and is expressed negatively to decrease GIPPath: decreasing GIPPath indicates decreasing the need of gamifying functions of that path. CEP is useful in case the organisation has already measured the user engagement per function. If this information is not available, CEP value will be 0 and will not affect the GIPPath calculation.

A significant aspect of SiaGAM is its flexibility. In fact, the analyst can specify if to consider the full AS, or subsets of it, via an expression, which is passed to SiaGAM. This can be reiterated with different subsets, by specifying different annotations until the analyst is satisfied with the analysis results. Hereafter, we call the subset chosen by the analyst as Analyst Annotation Set (AAS).

Fig. 2 shows a simplified example of SiaGAM applied, within the VisiOn EU Project, for the gamification of STS-Tool [3, 18].

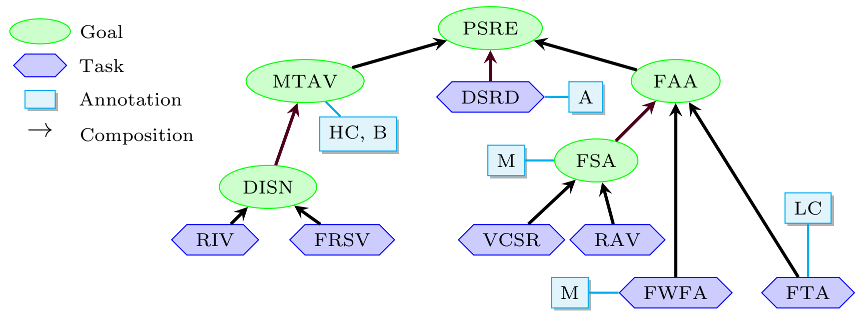

Fig. 2: Annotated Goal Model: applying SiaGAM for gamifying STS-Tool [3, 18]

The example shows how we improved the first phase of Agon process (BSRM): now the analyst is guided and supported semi-automatically in applying the annotations to the BSRM model during the definition of the Goal Model by following the guidelines and aspects of SiaGAM discussed above. To achieve this, an Annotation Goal Model (AGM) [3] is produced, which includes characteristics of both the original BSRM and AREA models. In this phase, analysts can use the default configuration of SiaGAM (determined from our experience in successfully gamifying software in different EU projects [11, 17--19, 23], and according to feedback received from analysts involved), reducing their effort, or configure it. For instance, Fig. 2 shows that the analyst decided to annotate `MTAV`, i.e. ''Model The Authorization View'' goal [3, 18], as High Complex and Boring (`HC` and `B` annotations respectively) to be fulfilled.

Algorithm Execution. This phase regards the support provided to the analyst by SiaGAM through the semi-automation of AREA. SiaGAM analyses AGM and supports the analyst in identifying the most interesting paths related to the functions where to apply gamification. Specifically, SiaGAM analyses AGM with an adapted pre-order visit, considering annotations of the functionalities for calculating GIPPath, and providing a final model with functionalities that, according to annotations and criteria specified, are considered candidates to be gamified. The next Figure shows the pseudo-code of the SiaGAM algorithm.

SiaGAM works on a Goal Model g, specified as `g=(C,R)`, where the set of concepts C represents functions of the system by using goals, sub-goals and tasks (Fig. 2). R is the set of relations used to connect the concepts (e.g., composition). SiaGAM takes in input a `GoalModel g` (the AGM model processed by the algorithm),

Algorithm 1: **SiaGAM**(GoalModel g, Expression aasExp, Float minGIPPath)

```
1  if (g.root isNull) then
2      return g;
3  else if (g.root.annotations isEmpty) then
4      while ((GoalModel child = nextChild(g)) != null) do
5          SiaGAM(child, aasExp, minGIPPath);
6      delete(g.root);
7  else if (evaluate(g.root, g.root.annotations, aasExp, minGIPPath)) then
8      printModel(g.root, g.root.annotations, g.root.GIPPath);
9  else
10     delete(g.root);
11 return g;
```

aasExp (i.e. AAS, the Analyst Annotation Set expression discussed before), and minGIPPath (the minimum GIPPath value a function should have to be selected as a candidate function to gamify). Thus, aasExp variable represents AAS, the satisfactory annotations the algorithm must verify. AAS can be specified by the analyst and defined through aasExp. In this way, the analyst can perform different simulations based on different expressions, with the possibility to perform an accurate, iterative analysis by considering distinct aspects each time. For instance, according to the example in Fig. 2, we can express the aim of knowing which are the candidate functions satisfying minGIPPath $\geq$ 0.8, and AAS = {M, HC, B} specified through aasExp. To achieve this, SiaGAM checks the root node (Fig. 2): if the result is null, the algorithm stops; otherwise, the annotations set of the current functionality (represented as a goal) must be checked to see if it is empty. If the value is True, i.e. current element is not annotated, the algorithm is recursively called for each of g children (from the left child to the latest to the right side). Once on a node with annotations, the evaluate function checks if there is a match among annotations of the current node and aasExp, it updates GIPPath of the node accordingly, and checks if minGIPPath is satisfied for the node. If evaluate returns True, the node is automatically kept, otherwise it is deleted. This process is executed in AGM, and, at the end, the process returns the filtered AGM with the functionalities to gamify according to criteria specified.

SiaGAM Example and Results. The example illustrated in Fig. 2, is an extract, simplified, example from the VisiOn EU Project [3,18], aiming at identifying candidate functions to gamify for the STS-Tool, a tool used by analysts for modeling privacy and security requirements via different views [3,18]. In the example we execute SiaGAM with g, representing the Goal Model of STS-Tool functions (Fig. 2), minGIPPath $\geq$ 0.8, and aasExp = {M, HC, B}. SiaGAM starts from the PSRE (i.e. performing "Privacy and Security Requirements Engineering" via the STS-Tool) root goal, and, not being annotated, SiaGAM recursively continues with the first, left, child: MTAV. MTAV is the "Model The Authorization View" goal, and it is annotated with High Complex (HC) and Boring (B). Thus, the evaluate function is called, it finds a match among such annotations and `aasExp` ones, and `minGIPPath` is satisfied, therefore MTAV is kept becoming candidate function to be gamified. Then, SiaGAM recursively visit the other children of `PSRE`. `DSRD`, being annotated as Automatic function, is deleted. Continuing the recursion, `FSA` and `FWFA`, become candidate functions due to their annotation (i.e. Manual), and `FTA` is discarded being annotated with Low Complexity (`LC`). With this process, based on the filtered AGM processed by SiaGAM, it is also possible to obtain a formula extracted by the AGM. For instance, the following one has been obtained after the application of SiaGAM for the gamification of STS-Tool (Fig. 2) [3,18] (VisiOn EU Project).

$$Acceptance\ [\{MTAV,\ FSA,\ FWFA\},\ Users] \geq 80\%$$

This formula represents an acceptance requirement, indicating that to achieve (at least 80%) acceptance of our system by users, we need to consider gamifying functions obtained using SiaGAM, i.e. $\{MTAV,\ FSA,\ FWFA\}$ (Fig. 2) [3,18].

4 Case Study and Evaluation

We evaluated SiaGAM by considering the successful results obtained from 5 case studies within EU projects [11,17--19,23]. Analysts had manually selected various subsets of functions to gamify. We started by reusing the original Goal Model of the system (from each case study [11,17--19,23]) as designed in the original BSRM phase. Models did not show decisions about what functions to gamify, which were made in the original AREA phase. We used such models as input for our enhanced BSRM phase (*Annotation Model*, section 3) and proceeded creating a new AGM. The aim was to compare the 2 resulting models, by checking if SiaGAM is able to support the analyst in finding a subset of functions to gamify similar to the one of the original case study. We used the default configuration of SiaGAM described in section 3. Our case study, including both original results and our results for each of them, is available at [3]. The results of our analysis, as shown in [3] and summarised in Table 1, are based on the comparison of the Original Solution (**OS**) and the new solution, called SiaGAM Solution (**SGS**).

The 5 case studies from EU Research Projects we considered[3] [11,17--19,23] are indicated below (more details in section 2.2) and in Table 1 with the results of our evaluation: **(i)** VisiOn Case Study on "*Gamification for Privacy Requirements Modelling*"[3] [18]; **(ii)** PACAS Case Study on "*Gamification for Participatory Architectural Change Management in ATM Systems*"[3] [17,23]; **(iii)** SUPERSEDE Case Study on "*Gamification for Collaborative Requirements Prioritization, the DMGame Case*"[3] [17,19]; **(iv)** SUM Case Study on "*Sustainable Urban Mobility (SUM)*"[3] [11,17]; **(v)** MA4C Case Study on "*Mobility Assistance for Children (MA4C)*"[3] [11,17]. The results of our analysis show that SiaGAM is an effective tool for supporting the analysts in this crucial phase: in fact, SiaGAM had been able to derive sets very close to those obtained manually (Table 1). As the

[3] VisiOn, PACAS, SUPERSEDE, SUM and MA4C original Case Studies available at: `https://pirasluca.wordpress.com/home/acceptance/case-studies/`

main issue found by the analysts was the manual aspect of the subset definition, it is evident that SiaGAM offers valuable help with its semi-automatic guide. According to Ghezzi [8], being our problem undecidable, we aimed at evaluating if SiaGAM is a sound and complete algorithm by calculating its precision and recall [8] in the cases considered. In fact, according to Ghezzi [8], our results "Oracles" stem from the results of the EU Projects considered, and we calculated false positives, false negatives and true positives in relation to the application of SiaGAM and its results. We defined such parameters as follows: **(i)** a *True Positive* (TP) is obtained when SiaGAM selects a task that was selected in the OS case study as well; **(ii)** a *False Positive* (FP) is obtained when SiaGAM selects a task that was not selected in the OS case study; **(iii)** a *False Negative* (FN) is obtained when SiaGAM discards a task that was selected in the OS case study. In Table 1 we show such values calculated per case study.

Case Studies	**OS Func.**	**SGS Func.**	**FP**	**FN**	**TP**	**Precision**	**Recall**
VisiOn[3] [18]	7	7	0	0	7	1	1
PACAS[3] [17,23]	5	6	1	0	5	0.83	1
SUPERSEDE[3] [17,19]	2	2	0	0	2	1	1
SUM[3] [11,17]	4	4	0	1	3	1	0.75
MA4C[3] [11,17]	3	3	0	0	3	1	1

Table 1: Case studies with functions identified ("OS Func.") considered as Oracles [8], functions identified with SiaGAM ("SGS Func.") and related False Positives (FP), False Negatives (FN), True Positives, Precision and Recall.

In Table 1 we indicate also precision and recall calculated as [8]: Precision $= \frac{TP}{(TP+FP)}$ and Recall $= \frac{TP}{(TP+FN)}$. If precision is equal to 1, the algorithm is sound; if recall is equal to 1, the algorithm is complete [8]. According to Table 1, most cases show precision and recall equal to 1. However, we can see that for the PACAS case study, the precision is slightly less than 1. This is because SiaGAM selects one functionality that was not considered in the OS. Conversely, the recall parameter shows a lower value only in the SUM case study: in this case, SiaGAM selects a function that was not considered in the OS.

Overall, the results of the application of SiaGAM show that, according to most of the cases considered (Table 1), the algorithm and our approach can support *semi-automatically* the identification of the same acceptance requirements and system functionalities to gamify, which had been identified by previous, successful, *manual* analyses[3] [11,17--19,23]. Finally, it is important to compare our approach, proposed in this paper, with previous approaches. The main differences are summarised as follow: **(i)** with our approach, the analyst is guided and supported during BSRM in analysing the goal model, representing the system and its functions, characterising them with relevant annotations (suggested and supported by SiaGAM), which can help identifying functions to gamify. Previously, in this phase there was only the "pure" creation of the goal model; **(ii)** with previous approaches, AREA offered only high-level guidelines for identifying functions to gamify and specifying acceptance requirements, which meant the effort was entirely manual. Now, during AREA, the execution of SiaGAM provides the analyst with semi-automatic, flexible and extensible support for identifying functions to gamify; **(iii)** SiaGAM generates the AGM goal model, within the new AREA phase, together with the generation of formal definition of acceptance requirements obtained, including the subset of functions to gamify.

5 Related Work

Available contributions regarding Goal Modeling are various. The work by Jureta et al. [10] proposes a Goal Argumentation Method (GAM) and is composed of three roles. It defines the guidelines of the argumentation and justifications of the choices, while through the reasoning of a goal model it is possible to detect problematic argumentation and check if the information is contained in both the argumentation itself and the model elements. The main difference with our work is that SiaGAM provides guidelines and supporting annotations, which are specific for characterising systems concerning the identification of components to gamify, with the specific purpose of making a system more engaging. Moreover, our approach is extensible and can be applied to different goal modeling contexts and domains. Furthermore, our approach allows the analyst to iteratively use our algorithm to consider different parts of the system to be gamified. The framework PRACTIONIST proposed by Morreale et al. [14] aims at supporting programmers for the definition of Belief-Desire-Intention (BDI) agents: it captures the purpose of the software from the system goals and creates a Goal Model; then, relations between goals are analysed to determine if goals can be achieved. SiaGAM is not exclusively agent-oriented, it is oriented towards software characterisation from the analyst perspective. User characterisation also plays an essential role for the identification of functions to gamify. Furthermore, SiaGAM analyses goals differently working with the definition of a probability value in the Annotation Model and a list of annotations characterising the software and its usage.

The RationalGRL framework by Van Zee et al. [24] is structured in 4 separate phases: an argumentation phase, which elicits requirements through the definition of arguments; a translation phase, where arguments are translated into a Goal Model; a goal modelling phase, in which stakeholders evaluate the model; a final update phase for translating the model into GRL models, for comparison with argumentation elements. Our approach differs under various aspects. Firstly, we do not generate the goal model from argumentation. Instead, we support the creation of a goal model of the system with a graphical editor, and support analysis and annotations of the model for functionalities characterisation. Furthermore, we also support the analyst with semi-automated analysis provided by SiaGAM.

6 Conclusion

Gamification has clearly proved an effective technique for motivating users to use software systems. Systematic Goal-Oriented Requirements Engineering (GORE) activities have been proposed for supporting engineers to gamify software systems, and have been recognised as effective and useful by requirements analysts in different EU Research Projects. However, analysts have also indicated difficulty in performing some of such activities, highlighting the need for further support.

In this paper we identified such activities, provided related lessons learned, and proposed a solution for further supporting analysts in relation to one of these activities: the identification of software components to gamify. This is a crucial aspect as the gamification of a full software system can be expensive and could give unexpected results. Therefore, to reduce efforts and costs for organisations, the exact identification of functionalities to be gamified is necessary. The solution we devised in this paper is the algorithm SiaGAM (*Supporting the individuation, analysis, and GAMification* of software components), which guides and supports the analyst semi-automatically concerning the next crucial aspects: **(i)** representation of the software to gamify as a Goal Model; **(ii)** characterisation of software functions with the support of representative and decisive qualities; **(iii)** analysis and suggestion of candidate system functions to gamify, based on decisive criteria identified; **(iv)** extensible and iterative approach in terms of qualities considered and criteria defined; **(v)** generation of Acceptance Requirements with functionalities selected to be gamified. To evaluate SiaGAM, we applied it to 5 case studies from EU projects, where requirements analysts had successfully identified system functions to gamify *manually*. Then we compared their results with the ones we obtained using SiaGAM. We found that SiaGAM is effective in supporting the analyst in *semi-automatically* identifying software functions to gamify, as the functions to gamify derived by SiaGAM were in most cases analogous to the ones selected *manually* by the analysts.

As future work, due to the flexibility and configurability of SiaGAM, we will consider using it to support the identification of software components for other GORE purposes. For instance, for the identification of set of functionalities that are candidates to be secured, or to be made GDPR compliant. We will also perform more advanced evaluation, by involving analysts, for considering also further aspects (e.g., ease of use, annotations set adequacy). Furthermore, thanks to the other contribution of this paper (activities and lessons learned), we will consider providing further support to analysts by improving the other GORE gamification activities which were found useful but requiring further support.

References

1. Bassanelli, S., Vasta, N., Bucchiarone, A., Marconi, A.: Gamification for Behavior Change: A Scientometric Review. Elsevier Acta Psychologica Journal (2022)
2. Calabrese, F.: Gamification with the Agon Framework: a Case Study on Privacy Requirements Modeling. Bachelor thesis, University of Trento, Italy (2018)
3. Calabrese, F., Piras, L., Giorgini, P.: Models and dataset related to Case Study and Evaluation, respectively available at: `https://data.mendeley.com/datasets/6s4c87c494/4` and `https://data.mendeley.com/datasets/rczj54927m/1`
4. Chung, L., Nixon, B., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Springer, New York (2012)
5. Deterding, S.: The Lens of Intrinsic Skill Atoms: A Method for Gameful Design. Human-Computer Interaction Journal pp. 294--335 (2015)

6. Deterding, S., Dixon, D., Khaled, R., Nacke, L.: From Game Design Elements to Gamefulness: Defining "Gamification". In: Int. MindTrek Conference. ACM (2011)
7. Fernández, D., Legaki, N., Hamari, J.: Avatar Identities and Climate Change Action in Video Games: Analysis of Mitigation and Adaptation Practices. In: CHI Conference on Human Factors in Computing Systems. pp. 1--18 (2022)
8. Ghezzi, C.: Being a Researcher. Springer, New York (2020)
9. Herzig, P., Ameling, M., Schill, A.: A Generic Platform for Enterprise Gamification. In: Europ. Conf. on Software Architecture. IEEE (2012)
10. Jureta, I.J., Faulkner, S., Schobbens, P.: Clear Justification of Modeling Decisions for Goal-Oriented Requirements Engineering. Requirements Engineering (2008)
11. Kazhamiakin, R., Marconi, A., Perillo, M., Pistore, M., Valetto, G., Piras, L., Avesani, F., Perri, N.: Using Gamification to Incentivize Sustainable Urban Mobility. In: 1st Intern. Smart Cities Conf. (ISC2). pp. 1--6. IEEE, New York (2015)
12. Koivisto, J., Hamari, J.: The Rise of Motivational Information Systems: A Review of Gamification Research. Int. Journal of Information Management (2019)
13. Li, F.L., Horkoff, J., Mylopoulos, J., Guizzardi, R., Guizzardi, G., Borgida, A., Liu, L.: Non-Functional Requirements as Qualities, with a Spice of Ontology. In: Int. Requirements Engineering Conference (RE). pp. 293--302. IEEE, New York (2014)
14. Morreale, V., Bonura, S., Francaviglia, G., Centineo, F., Cossentino, M., Gaglio, S.: Goal-Oriented Development of BDI Agents: the PRACTIONIST Approach. In: Int. Conference on Intelligent Agent Technology. pp. 66--72. IEEE, New York (2006)
15. Mylopoulos, J., Chung, L., Nixon, B.: Representing and Using Nonfunctional Requirements: A Process-Oriented Approach. Transac. on Software Engineer. (1992)
16. Peng, C., Xi, N., Hong, Z., Hamari, J.: Acceptance of Wearable Technology: A Meta-Analysis. In: Hawaii Int. Conference on System Sciences (HICSS). (2022)
17. Piras, L.: Agon: a Gamification-Based Framework for Acceptance Requirements. Ph.D. thesis, University of Trento, Italy (2018)
18. Piras, L., Calabrese, F., Giorgini, P.: Applying Acceptance Requirements to Requirements Modeling Tools via Gamification: A Case Study on Privacy and Security. In: Int. Conf. on Practice of Enterprise Modeling (PoEM). Springer (2020)
19. Piras, L., Dellagiacoma, D., Perini, A., Susi, A., Giorgini, P., Mylopoulos, J.: Design Thinking and Acceptance Requirements for Designing Gamified Software. In: Intern. Confer. on Research Challenges in Information Science (RCIS). IEEE (2019)
20. Piras, L., Giorgini, P., Mylopoulos, J.: Models, dataset, case studies, prototype and glossary of Agon (an Acceptance Requirements Framework), `https://pirasluca.wordpress.com/home/acceptance/` and `https://data.mendeley.com/datasets/56w858dr9j/1`
21. Piras, L., Giorgini, P., Mylopoulos, J.: Acceptance Requirements and their Gamification Solutions. In: Int. Requirements Engineering Conf. (RE). IEEE (2016)
22. Piras, L., Paja, E., Cuel, R., Ponte, D., Giorgini, P., Mylopoulos, J.: Gamification Solutions for Software Acceptance: A Comparative Study of Requirements Engineering and Organizational Behavior Techniques. In: IEEE International Conference on Research Challenges in Information Science (RCIS). pp. 255--265. IEEE (2017)
23. Piras, L., Paja, E., Giorgini, P., Mylopoulos, J.: Goal Models for Acceptance Requirements Analysis and Gamification Design. In: 36th International Conference on Conceptual Modeling (ER). pp. 223--230. Springer, New York (2017)
24. Van Zee, M., Marosin, D., Bex, F., Ghanavati, S.: RationalGRL: A Framework for Rationalizing Goal Models Using Argument Diagrams. In: International Conference on Conceptual Modeling (ER). pp. 553--560. Springer Cham, New York (2016)
25. Zichermann, G., Cunningham, C.: Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps. O'Reilly Media, Inc. (2011)