Free Hugs — Praising Developers For Their Actions

Roberto Minelli, Andrea Mocci, and Michele Lanza
REVEAL @ Faculty of Informatics — University of Lugano, Switzerland

***Abstract*—Developing software is a complex, intrinsically intellectual, and therefore ephemeral activity, also due to the intangible nature of the end product, the source code. There is a thin red line between a productive development session, where a developer actually *does* something useful and productive, and a session where the developer essentially produces "fried air", pieces of code whose quality and usefulness are doubtful at best. We believe that well-thought mechanisms of gamification built on fine-grained interaction information mined from the IDE can crystallize and reward good coding behavior.**

We present our preliminary experience with the design and implementation of a micro-gamification layer built into an object-oriented IDE, which at the end of each development session not only helps the developer to understand what he actually produced, but also praises him in case the development session was productive. Building on this, we envision an environment where the IDE reflects on the deeds of the developers and by providing a historical view also helps to track and reward long-term growth in terms of development skills, not dissimilar from the mechanics of role-playing games.

I. Introduction

What is a game? According to McGonigal [1] games share four defining traits: a *goal*, *rules*, a *feedback system*, and *voluntary participation*. The goal gives a sense of purpose. The rules unleash creativity and foster strategic thinking. The feedback system provides motivation. The voluntary participation makes the experience safe and pleasurable. Suits sums it up with "playing a game is the voluntary attempt to overcome unnecessary obstacles" [2]. McGonigal provides several examples of contexts, ranging from house holding chores to physical exercise, where the performance of subjects has been boosted through gamification [1]. While this may seem remote from software engineering, Werbach and Hunter provide an illuminating example closer to our discipline: Microsoft's testing team in charge of the multi-language aspect of Windows 7 invented the Language Quality Game, recruiting thousands of participants who reviewed over half a million dialog boxes, logging 6,700 bug reports, resulting in hundreds of fixes [3]. Another example is StackOverflow, a Q&A website where asking and answering technical questions is rewarded with points and badges. There is evidence that gamification is in part responsible for StackOverflow's success [4].

We envision the use of gamification in the context of how developers use the main vehicle for programming, namely the integrated development environment (IDE). Modern IDEs have become powerful tool suites that allow one to construct, understand, and modify software systems. This is quite a step away from the (in our opinion outdated) notion, promoted by Weinberg several decades ago, that programming is a "kind of writing" (of source code) [5].

We believe this is a fundamentally flawed perception, and among others is also responsible for the wrong, and fashioned, assumption that productivity can be measured in terms of lines of code [6], [7]. Programming is more, beautifully put by Brooks: *"The programmer, like the poet, works only slightly removed from thought-stuff. He builds his castles in the air, from air, creating by exertion of the imagination."* [7].

Although this sounds all very poetic and romantic, in the end developers often end up as unsung heroes, *i.e.,* there is no mechanism (other than venal aspects) that rewards them for a good job. What can be done about it?

The ICSE NIER track welcomes *"Bold visions of new directions which may not yet be supported by solid results but rather by a strong and well motivated scientific intuition."*. As opposed to the common belief that gamification is a recent, under-explored and thus not very scientific domain, behind it there is in fact a strong scientific intuition, rooted in the realm of psychology, and more specifically behaviorism, an approach that combines elements of philosophy, methodology, and theory [8]. In fact, behaviorism, whose main tenet is "if you do this you'll get that", is the antithesis of successful gamification, because simple rewarding mechanisms like token programs have been shown to be bound to fail [9] in the long run. Put simply, our goal is not to assign points to development actions. Such a simplistic approach is destined to fail. Rather, we propose a comprehensive approach where the ultimate goal is the creation of an *alter ego* of a developer, which we believe is the key to enable what McGonigal [1] identified as the 4 key aspects of successful gamification: (1) Satisfying work (after all, programming is creative), (2) the experience/hope of being successful, (3) a social connection, and (4) a deeper meaning. We argue that the creation of such an alter ego is the key to provide both short-term and long-term gratification to developers.

We are currently devising and implementing an approach which leverages fine-grained interaction data mined from an IDE using our tool DFLOW [10]. DFLOW models and silently harvests any low-level action performed by developers and thus offers a complete and precise summary of what is being done. Our vision is composed of the following steps:

- ***Session Digest.*** The session digest is a short-term form of gratification, similar to the one present in fitness apps, offered to developers for their last development session. It summarizes the last session from various perspectives, *e.g.,* how was time used, how much was achieved from a coding point of view, which program entities were involved, etc. It also enables to dig into the fine-grained recorded data and acquire a deeper understanding.

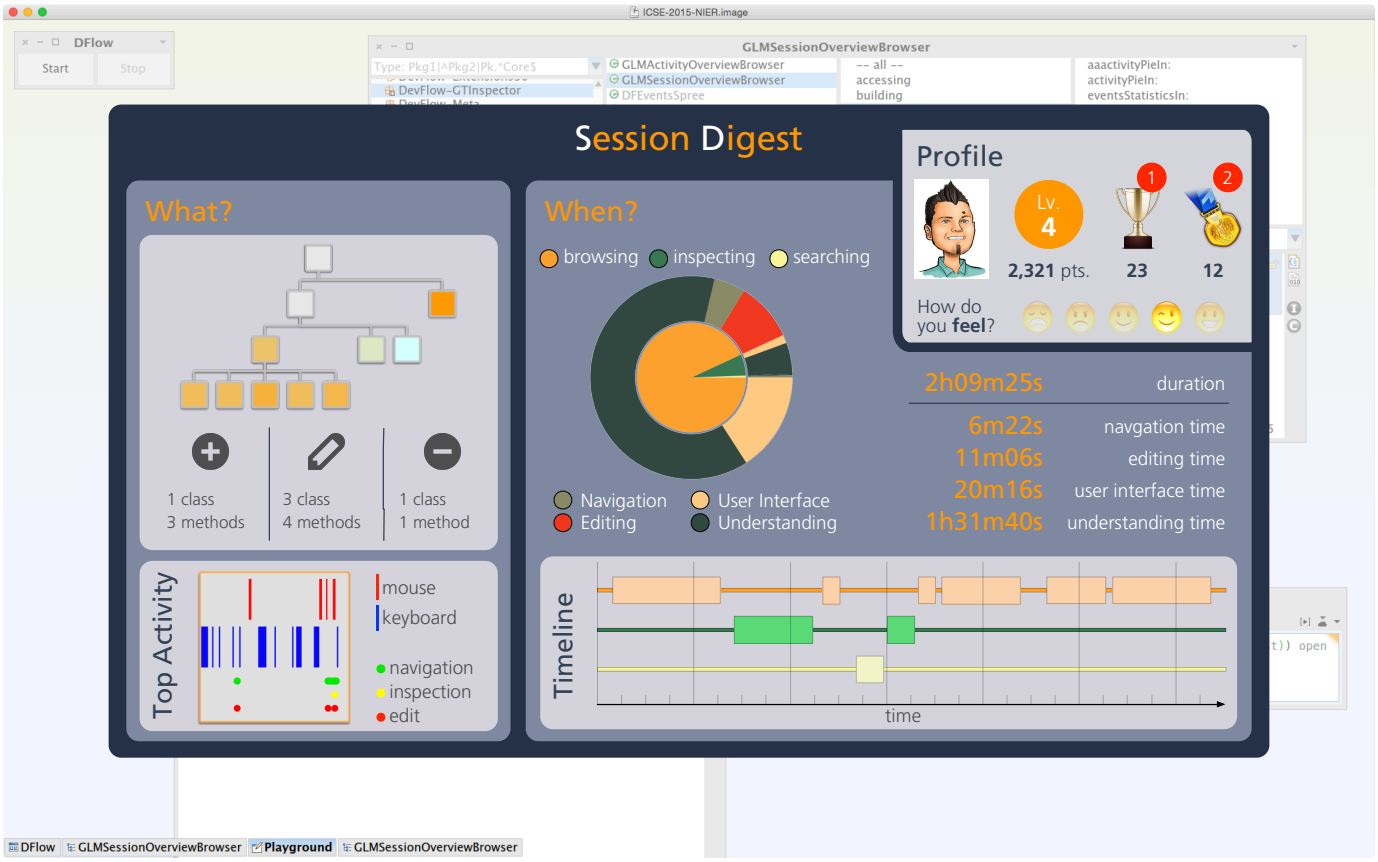

Fig. 1. Session Digest: How have you spent your time and what did you do?

- ***Alter Ego.*** A developer is like a character in a role-playing game: She moves her first steps, evolves, acquires new skills, and unlocks new achievements. Developers are thus assigned an avatar that they can evolve, providing them short- and long-term satisfactions to turn software development into a more engaging activity.
- ***Development Empire.*** The last, and most ambitious goal is provide developers with long-term gratification mechanisms. We envision a comprehensive gamification layer on top of the IDE: the *Development Empire*. It is not all about assigning points to them, but a ramified system that rewards complex actions and best practices (adherence to design patterns and design heuristics [11], test-driven development, etc.) of a developer with badges, achievements, and trophies of different types. The history and the evolution of the alter ego of a developer is a key factor. When this mechanism is in place, all the alter egos will originate a new community, where people can observe, challenge, and interact with other developers.

Our vision builds on our previous work on interaction mining and visualization [10] and fine-grained evolutionary information collection [12]. We believe our idea calls for novel research directions encompassing both software engineering and psychology. In Section II we detail our technical contribution, the session digest, a short-term gratification system.

II. Session Digest: Free Hugs for Developers

The *"session digest"* is a form of short-term gratification for developers, that can potentially augment their level of engagement. We shape and frame the rewards in a digest as visual summary of the development session. The digest is composed of three main parts: (a) an overview of the development session; (b) a selection of fine grained information to highlight the most important actions; and (c) a glimpse on the "profile" of the developer. The general overview, *e.g.,* in terms of how the developer spent her time, serves as an entry point for the digest. Once the developer gets the global picture, she can use the remaining part of the digest to retrospectively analyze *how well she did* in the current session. The last part, the developer profile, summarizes the developer's avatar status.

In Practice: Figure 1 depicts a sample session digest prototype on top of the standard IDE window. The left part of the digest serves as a general overview on how the developer spent her time. The overview is achieved by a sunburst visualization accompanied by a set of time metrics. Figure 2 shows the sunburst in details. Its central part distinguishes the time devoted to the three different high-level activities, namely: browsing (orange), inspecting (green), and search (yellow). For each of the three types of activities, the visualization shows four time components: navigation (green), editing (red), understanding (dark green), and user interface (pale orange).

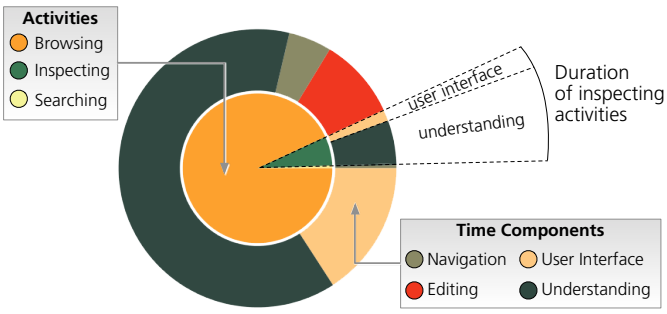

Fig. 2. Activities and time components in a sunburst visualization

For completeness, the visualization provides the same information as text. We quantify these time components using interaction histories mined with DFLOW [13].

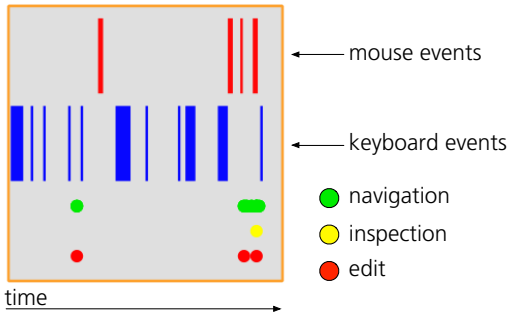

Fig. 3. Activity View: Decomposing an High-Level Development Activity

After an overview, a developer has the possibility to dig into her last development session by means of the central part of the digest. In the example of Figure 1, the central part shows an interactive tree visualization that portrays all the entities that she has interacted with in her last session. The tree has 4 different levels, (1) subsystems, (2) packages, (3) classes, and (4) methods. Each entity have a color to represent the intensity of the interactions: gray for entities with no interactions (*i.e.,* inserted only as a transitive closure to complete the tree) and a color scale from light blue (*i.e.,* few interactions) to orange (*i.e.,* lot of interactions) for the other elements. Below this view there is a table that shows how many entities were respectively added, modified, or removed. In Figure 1, the bottom right corner presents the *activity view*. This view, detailed in Figure 3, uses a custom layout to decompose a single high-level development activity. It depicts mouse, keyboard, and *meta* events. The first two kinds of events have a duration, proportional to the width of the rectangles representing them. *Meta* events have no duration since they represent IDE actions such as *saving a method*. Their color represents their type, according to the impact they have on source code. Navigation events (green) do not modify source code, inspect events (yellow) are a deeper form of navigation (*i.e.,* in a debugger), while editing events (red) modify source code. In the session digest, for example, we can show the *hardest activity*, as depicted in Figure 1.

The last component of the digest is the developer's profile, depicted in the top right corner. It shows a profile picture of the developer, her level, points, trophies, badges, and lets her provide a "sentiment feedback" about the last development session on a *smiley scale*. This enables analyses on what characterizes, for example, a frustrating session. Figure 1 shows a prototypical profile of the first author of this paper. He is at level 4, with 2,321 points. In the last session he achieved 1 new trophy and received 2 new badges. Overall, he owns 23 trophies and 12 badges. This goes towards an application of gamification in software development [14]. Gamification, if carefully engineered [1], can increase the motivation and engagement of developers. For an open source community, such as the one behind the PHARO IDE[1], our target development environment, this can provide several benefits. Having more motivated people will most likely increase both the quality and the quantity of the contributions to the community. At the same time, this will originate a novel developer community that, at first sight, is similar to Open HUB[2], the open source network formerly known as Ohloh. But our vision goes beyond *mere* version control system data. We imagine a rich developer profile that includes interactions with different sources of information including, but not limited to, IDE interactions. For example, the system will integrate bug tracking systems, questions & answers services, mailing list participation, etc. Points, trophies, and badges will cross the boundaries of semantically different domains, delineating a comprehensive profile for developers.

A. *How this is done:* DFLOW – *The IDE Interaction Profiler*

The digest presents fine-grained interaction data mined from an IDE. While interacting with IDEs developers generate a huge amount of interaction data of different granularities. Examples include code-specific events, like adding a new class or a method, or editing a method implementation, and user interfaces (UI) interactions, like resizing or moving windows, etc. Current IDEs, however, neglect or, in general, do not fully exploit this information [15]. To enable retrospective analysis of a development session, as part as our previous work we implemented DFLOW, an interaction profiler [10]. DFLOW is an extension for the PHARO IDE that non-intrusively records all IDE interactions. Collected data includes high level events, such as adding or modifying an entity, as well as low level events, such as mouse movements, keystrokes, and UI interactions, *e.g.,* opening, resizing, or closing a window.

When enabled, DFLOW automatically records interaction data. When the tool detects the end of a session (*i.e.,* the user attempts to close the IDE or she is inactive for a given time), the session digest comes into play. The IDE provides the developer with a session digest, as shown in Figure 1. Moreover, since we collect long-term data, the developer can then also compare the just finished session with previous sessions, which allows her to obtain a view on her productivity over longer time spans, beyond the one of a single session.

[1]See pharo.org
[2]See https://www.openhub.net

III. Future Work

This section highlights possible extensions of the digest (III-A) and provides details on our evaluation plans (III-B).

A. Extending the Session Digest

Regardless from the fact that we can present IDE interaction data from different perspectives, the session digest can also be extended in multiple, orthogonal, directions.

Bug-tracking systems. Our research group is also working on how the process of submitting bug reports and patches can be ameliorated [16]. Our goal is to integrate the process of submitting a bug report directly within the IDE. Once bug-tracking information is available in the IDE, statistics about how a developer behaves in reporting and fixing bugs can be integrated inside the session digest. For example, we can keep track on *how active* a contributor is in the bug-fixing process, *i.e.,* how much she contributes, how many bugs she fixes, or how many issues she submits to the bug tracker system.

Coding style and guidelines. Our group is also working on code style and quality metrics. The session digest can include evolutionary visualizations on how quality and style of a software system evolved over time. This information gives a tangibile feedback, and intrinsically a reward, to developers investing efforts in ameliorating the design and implementation of their systems.

Questions & answers services (Q&A). Our group is also working on integrating Q&A services inside the IDE. We envision IDEs that harness the potential of the crowd knowledge, for example by letting a developer read, answer, and post new questions on these platforms directly from the IDE [17]. At that point, our digest can include such information and reward developers that better exploit and contribute to the crowd knowledge base.

Discussion. These ideas introduce just a few of the many sources of data that the digest can present to a developer. In turn, this sets the ground for our more ambitious vision of a gamification layer built into an object-oriented IDE.

B. Evaluation Plan

Our target IDE is the Pharo IDE, a open-source object-oriented language and environment. Behind Pharo there is a an open-minded and reachable community, with whom we are in touch. Our plan is to strongly rely on Pharo developers and to value their feedback as the most important parameter.

Our aim to evaluate to what extent developers benefit from what we propose, in terms of engagement, self satisfaction, and improved productivity. The first step is to release the gamification system to the Pharo community and conduct a continuous qualitative evaluation. Interviews and questionnaires will help us to evaluate the design of the gamification system and refine it to meet developers' feedback, expectations, and criticism. This would enable a positive feedback loop with developers, eliciting latent issues that cannot be foreseen in advance. When we reach a stable release, we plan to conduct a comparative evaluation between developers participating in the Development Empire and developers using the standard Pharo release. We want to assess different parameters such as satisfaction, engagement, productivity, and quality of the code being produced.

Our conjecture is that developers involved in the Development Empire will likely achieve better results, in terms of code quality, productivity, and community engagement with respect to the others. At the same time they will "feel better" due to the rewarding mechanism offered by our system.

IV. Conclusions

We presented our vision and initial design and implementation of a micro-gamification layer on top of an object-oriented IDE. Our long term goal is a system that rewards long-term growth in terms of development skills.

Synergies between gamification and software engineering are a very novel phenomenon. In this work we first explained the scientific intuitions, which are rooted into in the realm of psychology, a field which is orthogonal to software engineering. We strongly believe that such an approach might have a positive impact, particularly—but not only—on small and open-minded communities, such as the one we are targeting.

References

[1] J. McGonigal, *Reality is Broken*. Penguin, 2011.

[2] B. Suits, *The Grasshopper: Games, Life and Utopia*. Broadview Press, 2005.

[3] K. Werbach and D. Hunter, *For the Win*. Wharton Digital Press, 2012.

[4] B. Vasilescu, V. Filkov, and A. Serebrenik, "Stackoverflow and github: Associations between software development and crowdsourced knowledge," in *Proceedings of SocialCom 2013 (International Conference on Social Computing)*, Sept 2013, pp. 188–195.

[5] G. M. Weinberg, *The Psychology of Computer Programming*. New York, NY, USA: John Wiley & Sons, Inc., 1985.

[6] T. C. Jones, "Measuring programming quality and productivity," *IBM Systems Journal*, vol. 17, no. 1, pp. 39–63, 1978.

[7] F. Brooks, *The Mythical Man-Month*, 2nd ed. Addison-Wesley, 1995.

[8] B. Skinner, *Reflections on Behaviorism and Society*. Prentice Hall, 1978.

[9] A. Kohn, *Punished by Rewards*. Houghton Mifflin, 1993.

[10] R. Minelli, A. Mocci, M. Lanza, and L. Baracchi, "Visualizing developer interactions," in *Proceedings of VISSOFT 2014 (2nd IEEE Working Conference on Software Visualization)*, 2014, pp. 147–156.

[11] A. Riel, *Object-Oriented Design Heuristics*. Addison-Wesley, 1996.

[12] R. Robbes and M. Lanza, "A change-based approach to software evolution," *Electronic Notes in Theoretical Computer Science (ENTCS)*, vol. 166, pp. 93–109, Jan. 2007.

[13] R. Minelli, A. Mocci, M. Lanza, and T. Kobayashi, "Quantifying program comprehension with interaction data," in *Proceedings of QSIC 2014 (14th International Conference on Quality Software)*, 2014, p. to be published.

[14] E. Mastrodicasa, "Ludus opus proficit - a gamification framework for software engineering," Master's thesis, University of Lugano, 2014.

[15] G. C. Murphy, M. Kersten, and L. Findlater, "How are java software developers using the eclipse IDE?" *IEEE Software*, vol. 23, no. 4, pp. 76–83, 2006.

[16] T. dal Sasso and M. Lanza, "in*bug: Visual analytics of bug repositories," in *Proceedings of CSMR-WCRE 2014 (1st Joint Meeeting of the European Conference on Software Maintenance and Reengineering and the Working Conference on Reverse Engineering)*, 2014, pp. 415–419.

[17] L. Ponzanelli, A. Bacchelli, and M. Lanza, "Seahawk: Stack overflow in the ide," in *Proceedings of ICSE 2013 (35th International Conference on Software Engineering, Tool Demo Track)*. IEEE CS Press, 2013, pp. 1295–1298.