Participatory Design to Lower the Threshold for Authoring Intelligent Support

First Author, Second Author, Third Author, and Fourth Author

No Institute Given

Abstract. One of the fundamental aims of authoring tools is to provide teachers who have low level of technical expertise with opportunities to adapt and appropriate content and pedagogical strategies of intelligent systems. However, there are still many challenges that affect teachers′ effective engagement with such authoring systems and there is a need for systems that have lower thresholds in terms of the users′ technical expertise. Here, we demonstrate that reducing the entry barrier for authoring tools can potentially be achieved through co-design activities of such systems with non-programmers and carefully observing novices. Following an iterative participatory co-design cycle with teachers who have little or no programming expertise, we reflect on their proposed enhancements. Our investigations focus on an authoring tool that has been designed primarily for Exploratory Learning Objects but we conclude the paper by providing transferable lessons for other authoring tools, particularly the strong preference for visual interfaces and high-level pedagogical predicates for authoring analysis and feedback rules.

Keywords: Intelligent systems · authoring tools · participatory design

1 Introduction

The aspirational goal behind the development of authoring tools for many years has been to enable users with low technical expertise to create and modify content, including ideally their adaptive features according to their own pedagogical strategies [3, 2, 5]. However, the usability of such tools and particularly the time required to invest in learning them, are factors that affect teachers′ adoption and engagement in the design process of authoring [4]. It is important to understand that teachers have different expertise, needs and motivations and authoring tools should aim to meet those. In this paper, we present our approach to better appropriate authoring tools for teachers through participatory co-design activities.

Our case study is an AuthoringTool[1] that has been specifically designed for authoring automated support for Exploratory Learning Objects (ELOs). Inspired by the example-tracing approach [1], AuthoringTool encourages the author to develop feedback by executing the activity like a student. This provides the author with data in a log window that represent the various states of the

[1] The name AuthoringTool is used throughout the paper to adhere to the blind review

student throughout the learning activity. Based on this evidence, the author can then set up rules for the generation of formative and summative feedback (see Fig **??** for more details). See also our previous work (ref blinded).

This paper reflects on a participatory design study aiming to inform further development of the AuthoringTool towards lowering the entry threshold for teachers who have little or no programming expertise.

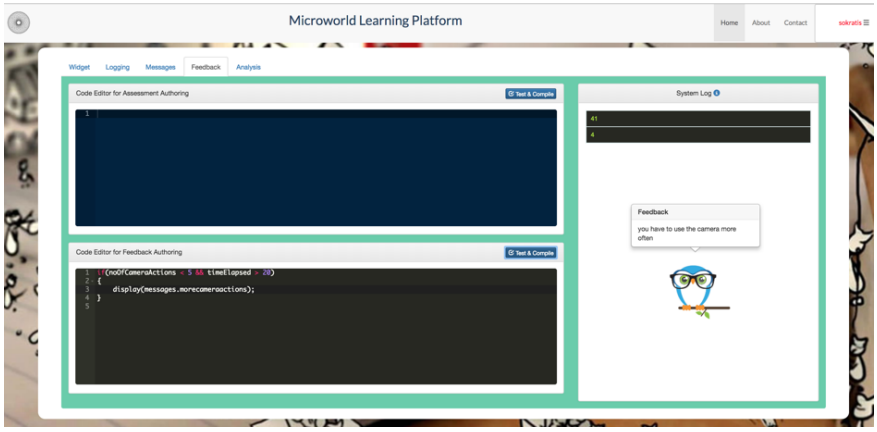

Fig. 1. Part of the interface of the AuthoringTool. After configuring the 'logging' in the corresponding tab, authors can start doing the activity like a student. That will immediately start generating data in the log and can used to author rules as in this example. The owl is the chosen feedback agent and here it is being tested.

2 Participatory design of authoring UI

For the purposes of this study participation of teachers in the design process was of paramount importance since the main aim is to lower the entry barrier. This is a clear case for participatory design, a well-accepted method for attempting to solve a complicated design problem with the active involvement of people from different backgrounds and different expertise (Bdker and Kyng, 2018).

Based on a non-random sampling strategy and a design-thinking approach, we carefully selected 6 newly qualified teachers who were studying Educational Technology masters and had a range of expertise in using technology in preschool and primary education settings, but no programming background (non-programmers group). We also selected 3 more experienced computing teachers with enough programming background to teach computing but not necessarily professional programmers to develop applications. They are all skilled in basic JavaScript (novice group). The six non-programmers were divided into two focus groups that were facilitated by one of the authors (EP) going through ideation,

sketching brainstorming, and thinking aloud around the interaction with a prototype. In parallel, the novice AuthoringTool users were supported to develop 15 different learning scenarios in the AuthoringTool. We recorded the support that one of the authors (SK) had to provide. The objective was to see how authoring is used and identify difficulties commonalities and patterns in their solutions that can provide the basis of a higher-level more expressive languages. These solutions were then analysed and we managed to reduce the code into a very small set of functions that seem to be a common requirement in all the activities.

3 Key findings and discussion

Due to space limitations, we focus on two themes that emerged from the brainstorming phase.

The influence of block coding One of the participants of the first group (familiar with block coding user interfaces) spontaneously proposed to introduce blocks of code with pre-defined "variables already written on, so we can drag and drop the blocks and connect them". Building on this idea, another participant drew a sketch with custom select lists "from where you can click on to see all the variables and choose one". Ensuing a conversation and brainstorming, the group sketched their final idea that involved use four custom select lists as shown in Figure 3. They named the first list 'condition', and from this list, they could pick the words 'if', 'then' and 'others. The second one was named 'situation', and when clicked all the previously set variables would appear on the list so they are able to pick the one that they need. The participants named the third list 'action', and with this list, they set what the variable should do, e.g. 'display', 'do', 'play sound'. The rest of the discussion was pragmatic and involved including a list that the participants named 'type' to choose from the list of resources that should be displayed and other aspects such as a delete button. The key contribution here was the idea that the interaction would result with the constructed rule clearly visibly below that would be added in the list of rules.

The second focus group, led by a one member of the group who volunteered to sketch their thoughts, steered towards an idea of 'board' with a standard structure where the words 'if', 'then', 'else', 'or' were written i.e. a custom select list for the various conditional statements (called 'conditions'). This idea became the centrepiece around which two other boards were proposed ('actions' and 'reactions') as well as a 'construction' area for dragging and dropping the various choices to make the feedback rules.

3.1 Situation and actions as predicates

Analysing the brainstorming of the non-programmer groups, a dominant theme was their concern on how to trigger feedback — something that they came

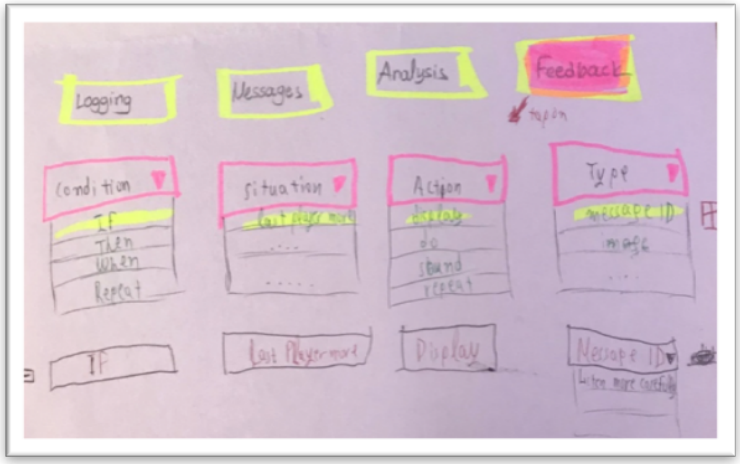

Fig. 2. Low fidelity paper prototype proposed by the non-programmers group

up with unprompted. Without any prior knowledge in authoring rules or functional programming, they referred to 'situations' or 'actions' in a similar way to predicates in programming. Generalising even from a simple task they had, the participants referred to "all previously set variables to appear on the list so they are able to pick the one that they need".

Analysing the support we had to provide to the novice users, the majority also revolved around the type of actions seen. Some of the high level constructs used in analysis and feedback involved: 1) Number of actions since a particular trigger point (e.g. the start of the activity, or a particular event such as enabling the camera view), 2) The number of actions of a particular type (e.g. the number of times a button was clicked), 3) A list of actions of a particular type and references mostly to the first and last of those. Furthermore, other high-level constructs were the time elapsed from the beginning of the activity, a pre-defined expected duration of the activity and a way to refer to the potential feedback messages and their types directly in a simplified way.

4 Conclusions

In this paper, we described our co-design approach to potentially lower the threshold for intelligent support authoring. We have incorporated the high-level constructs that emerged here in the design of the AuthoringTool to enable authoring declarative statements. Although, the results are from a small sample studied here, we think that the designs proposed here can reduce significant the amount of initial knowledge required from an author, reduce the complexity, increase readability, testability and maintainability of the code generated, allow the analysis to be communicated easier between authors in collaborative

projects, and allow the focus to be directed to things that matter most like rules and feedback. The work described here also paves the way for a new user interface that take advantage of the prevalence of block coding among teachers. All of these potential improvements can eventually lead to wider spread adoption of authoring tools among teachers who have low technical expertise.

References

1. Aleven, V., McLaren, B.M., Sewall, J., van Velsen, M., Popescu, O., Demi, S., Ringenberg, M., Koedinger, K.R.: Example-Tracing Tutors: Intelligent Tutor Development for Non-programmers. International Journal of Artificial Intelligence in Education **26**(1), 224–269 (Mar 2016). https://doi.org/10.1007/s40593-015-0088-2, https://doi.org/10.1007/s40593-015-0088-2
2. Da, F., Durdu, L., Gerdan, S.: Evaluation of Educational Authoring Tools for Teachers Stressing of Perceived Usability Features. Procedia - Social and Behavioral Sciences **116**, 888 – 901 (2014). https://doi.org/https://doi.org/10.1016/j.sbspro.2014.01.316, http://www.sciencedirect.com/science/article/pii/S1877042814003334
3. Gaffney, C., Wade, V.P., Dagger, D.: Authoring and Delivering Personalised Simulations an Innovative Approach to Adaptive eLearning for Soft Skills (2010), http://www.tara.tcd.ie/handle/2262/67212
4. Karoui, A., Marfisi-Schottman, I., George, S.: Mobile Learning Game Authoring Tools: Assessment, Synthesis and Proposals. In: Bottino, R., Jeuring, J., Veltkamp, R.C. (eds.) Games and Learning Alliance. pp. 281–291. Springer International Publishing, Cham (2016)
5. Murray, T.: Coordinating the complexity of tools, tasks, and users: On theory-based approaches to authoring tool usability. I. J. Artificial Intelligence in Education **26**(1), 37–71 (2016). https://doi.org/10.1007/s40593-015-0076-6, https://doi.org/10.1007/s40593-015-0076-6