EAI Endorsed Transactions
on Serious Games
Research Article EAI.EU

Toward reusable game technologies: assessing the usability of the RAGE component-based architecture framework

Wim van der Vegt[1,*], Kiavash Bahreini[1], Enkhbold Nyamsuren[1] and Wim Westera[1]

[1] Open University of the Netherlands, Valkenburgerweg 177, 6419 AT Heerlen, The Netherlands

{wim.vandervegt;kiavash.bahreini,enkhbold.nyamsuren,wim.westera}@ou.nl

Abstract

This paper investigates the usability of the RAGE component-based software architecture (RCSA). This architecture was designed to support serious game development by enabling cross-platform reuse of game software components. While the architecture has been technically validated elsewhere, this paper studies the perceived usefulness and ease of use of the architecture in practice. An extensive questionnaire based on the Technology Acceptance Model (TAM) was administered to 23 software and game developers that have been creating RCSA-compliant game components or integrating these in actual serious games. The results show that developers are generally positive about the usability of the architecture and that the architecture helps them to do a better job in less time. It turns out that developers effectively use all communication modes that are offered by the architecture, most frequently those based on the component´s APIs and the bridge pattern. Some issues were reported, but could be easily addressed. Most developers reported that they have well understood the effectiveness of the architecture and indicated to keep using the architecture in future projects. The outcomes of this study show that the architecture opens up new opportunities to the cross-platform reuse of advanced game functionalities in serious game projects, to reduce production efforts and to advance the domain of serious games at large.

Keywords: serious games, software components, game development, reuse, cross-platform, portability, game engines.

Received on 05 December 2018, accepted on 28 May 2019, published on 11 July 2019

doi: 10.4108/

[1]Corresponding author. Email:wim.vandervegt@ou.nl

1. Introduction

Although the potential of games for teaching and training has been widely recognised, their uptake in schools and business has been quite limited [1, 2]. The serious game industry displays many features of an emerging, immature branch of business, being scattered over a large number of small independent studios, displaying weak interconnectedness, limited knowledge exchange, and absence of harmonising standards [3]. Notably, progress is hampered by the wide variety of programming languages, game development systems and delivery platforms that are being used, all of which go with specific technical constraints and incompatibilities that pose severe barriers to growth. Moreover, access to emerging media technologies that could be easily incorporated in serious game projects, such as novel adaptation algorithms, artificial intelligence kernels, or natural language processing methods, is limited, while the alternative of in-company development of such technologies is not feasible, either because of required investments or because of lacking know-how.

This paper presents the evaluation results of the RAGE component-based software architecture (RCSA), which was designed to accommodate the development and reuse of advanced software components offering pedagogically relevant functionalities for serious games [4,5]. The RCSA was developed by the RAGE project (rageproject.eu), which is a leading serious gaming research project funded by the Horizon 2020 Programme of the European Commission. RAGE focuses on the development of advanced software

3. Method

The study was carried out with two extensive questionnaires that were administered in January 2018 to 18 component developers and five game developers (component users), respectively, involved in the RAGE project. Both groups are users of the RCSA, be it from different perspectives: component developers need to accept the RCSA to build upon, while game developers need to accept RCSA based components and the integration methodology the RCSA provides.

3.1. Target groups

The pool of potential participants familiar with the architecture was necessarily restricted to individuals within the RAGE project. The 18 component developers in the RAGE project were employees at research institutes from different European countries. The five game developers were professionals from the four game studios that were part of the RAGE consortium. In both groups, the age distribution is bimodal, revealing two peaks, one typically under 25 years and one around 40 years, respectively.

3.2. RCSA Components

The study relies on participants´ operational experiences, either as a developer or as a user, with one or more of up to 30 initial software components developed by RAGE. The quality and nature of the components' pedagogical functionalities are expressly excluded from current evaluation, as these are reported in separate studies. Now, the focus is on the usability of the architecture in the practices of software development and game development. Usability issues might particularly surface for client-side RCSA components, as they are inherently bound to the abstraction layers, e.g. by using the bridge pattern.

Instructions and support to component developers and game developers were provided through manuals, workshops and component code reviews. Component developers were supported with downloadable Visual Studio project templates for both C# and TypeScript (a superset of JavaScript including static typing). Most of the (client-side) components are written in C# and benefit from portable assemblies that are used across Visual Studio, Xamarin as well as the Unity3D game development platform. C# based project templates have been made available, including a regular (.NET 3.5) project and a portable assembly counterpart using the same source code. Both projects preserve portability by using a common subset of the two .NET framework versions in order to compile. Also, code snippets for implementing various bridge interfaces were made available.

3.3 Games

To assess the functioning of components in real games with real end-users, the four game studios in RAGE created seven component-based serious games of which the majority was created using Unity3D. The games focus on various social and entrepreneurial skills and address diverse target groups including school and university students, sports volunteers, policemen and corporate candidates. Overall, over 1500 participants in total were involved in the game pilots. Details about the game pilots and their evaluations can be found in [27, 28].

3.4 Questionnaires

We opted for questionnaires rather than interviews to avoid 1) any influences of interviewers and 2) potential issues resulting from (spoken) language barriers, given the various nationalities involved. Because of the two different target groups, two separate questionnaires were developed, both with a similar setup and structure, but with slightly different questions in some sections. The questionnaires were based on the Technology Acceptance Model (TAM) [7, 8], which was designed to collect information on perceived usefulness and ease of use, both being indicators of technology acceptance and usability. TAM was preferred to USE (Usefulness, Satisfaction, and Ease of use) [9], TTF (Task-Technology Fit) [10] and SUS (System Usability Scale) [11]. The USE and SUS instruments were discarded as they are more focused on the (graphical) user interfaces and associated end-user experiences and are difficult to apply to software coding and architectures. Task-Technology Fit was discarded, because of the lack of a suitable profile and the efforts required to create a new profile and validate it. The TAM-based questionnaire uses six items for each scale; topics are briefly indicated in Table 1.

Table 1. Topics covered by the TAM-based questionnaire for RCSA usability.

	Perceived usefulness	Ease of use
1	Faster task accomplishment	Easy to learn
2	Enhanced job performance	Easy to control
3	Improved productivity	Clear and understandable
4	Enhanced effectiveness	Flexible to use
5	Makes jobs easier	Easy to become skilful
6	Usefulness in job	Easy to use

For the TAM questions we used 'RAGE architecture' as subject, except for the first question on perceived usefulness in the component developers' questionnaire where we expressly used 'the RAGE architecture when creating reusable components' specifying the task more explicitly.

The 7 point Likert scales used the following labels for the perceived usefulness questions: 'extremely unlikely', 2, 3, 4, 5, 6 and 'extremely likely', respectively.

Figure 1 shows that game to component communication through component's API is most abundantly used. Communication in the reversed direction, that is, the component using an interface from the Bridge in order to gain access to the game or operating system functionality (such as saving and loading data), is also frequently used. Using this same mechanism to gain access to web-services was less used. Mutual communications between components were not much used as most components work independently from each other. Publish/subscribe broadcasting was the least popular communication mode. In sum, most RCSA communication modes are being used in the components, most frequently the ones using the components' APIs and the bridge interfaces.

Reported issues and comments

A comment was made about the risks of using files with the textual data format. This may cause UTF encoding issues when loading XML files. The .NET framework works internally with UTF-16 encoded strings [16], and as such it defaults to UTF-16 encoded XML files. Forcing UTF-8 output as used by some web-services requires some additional coding [18]. Binary data is currently only supported in C# by base64 encoding it [17].

One component developer highly appreciated using the bridge for platform dependent functionality but expressed concerns about the obligation for game developers to implement interfaces for the bridge, because they are reluctant to implement code that is not strictly related to their games. Their proposed solution was to include a ready to use bridge class with the component. Although the concern is legitimate, the proposed solution of adding a bridge actually undermines platform independence. Pointing towards the available code snippets providing reference was inspired by one of the leading game platforms, Unity3D, not supporting modern async/await type of method invocations during RCSA design. Only recently Unity3D has started supporting a more up-to-date .NET framework [19]. The RCSA easily supports this new framework with its portable assembly counterpart. Preliminary research also indicated that .Net Core 2.0 and newer are easy to add using the same shared sources mechanism as used for creating the portable assemblies. Besides the current interface, which does not enforce async calls, leaves the actual sync/async choice to the game programmer.

During component creation, one-third of the component developers reported having requested (and received) some support for the architecture team. Most component developers indicated that they would use the RCSA in future projects.

Architecture usability

Figure 2 and figure 3 show the normalised mean scores from the component developers on six items of the perceived usefulness scale and ease of use scale, respectively.

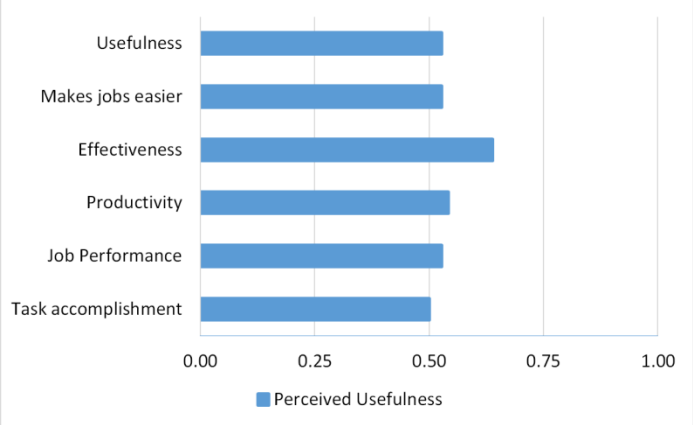

Fig. 2. Perceived usefulness according to component developers (normalized scores).

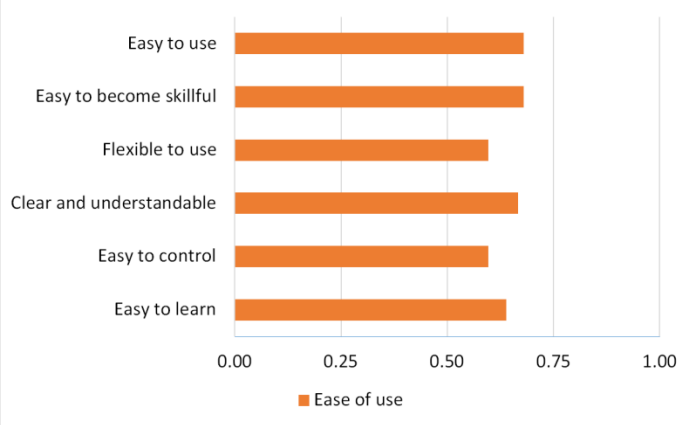

Fig. 3. Ease of use according to component developers (normalized scores).

Perceived usefulness has a mean score of 0.55 (standard error 0.05), whereas ease of use received a mean score of 0.64 (standard error 0.05), both representing values well above average. Actually, all separate items received scores above 0.50. Notably, component developers indicate that the RCSA makes tasks easier, helps to accomplish tasks more quickly and efficiently, and thereby improves job performance and productivity (perceived usefulness). Also, the RCSA is easy to understand, provides a flexible way to create components, which can be easily applied.

4.3. Results from game developers

Responses

most game developers qualify the RCSA-compliant components as useful and easy to integrate. Also, game developers used most of the communication patterns provided by the RCSA. Functionality for handling run-time and default settings to be compiled into the game was scarcely used, however. It seems that most game developers prefer to supply the settings through the game code. The tendency to stay in full control of their game application may pose a barrier to adoption of the RCSA. In [20] it was established that game studios are generally open and positive toward new technologies, but they are critical as such. They look for added value in terms of better games or commercial potential, but at the same time, they are afraid of complex and cumbersome implementation, which is understandable as their games should run smoothly without bugs or crashes. This exploitation requirement inevitably goes with some reluctance toward innovation: game developers first want to see the evidence before adopting something new. Some ambiguity was also shown by game developers raising concerns about software from academic origin, while at the same time they claimed to be confident with using third-party code.

Nevertheless, most game developers in the sample indicated that they would keep using the RCSA in future.

Although the respondents where RAGE project participants and this might have led to a bias in the TAM scores, the absence of high TAM scores and the presence of critical comments indicates that the respondents completed the questionnaire from a professional viewpoint and thus gives confidence the TAM scores are not biased.

Overall, this qualitative study has confirmed the practicability of the RCSA by tapping on the practical experiences of targeted component developers and game developers using the RCSA. The positive outcomes of this study open up new opportunities to flexibly incorporate advanced game functionalities in serious game projects, reduce production efforts and advance the domain of serious games at large. The outlook would be a flourishing market of advanced and affordable serious games that would contribute in purposeful ways to addressing societal problems in the fields of, e.g., media literacy, education and training, cultural heritage and social inclusion.

Future work will include monitoring acceptance by component and game developers outside RAGE and a closer investigation of the not RCSA architecture related questions on acceptance by game developers of foreign code (and especially code with an academic origin) but that might lower acceptation of components created according to the RCSA.

Acknowledgements.
This work has been partially funded by the EC H2020 project RAGE (Realising an Applied Gaming Eco-System); http://www.rageproject.eu/; Grant agreement No 644187.

References

[1] Carl Abt: Serious games. Viking Press, New York (1970).
[2] T.M. Connolly, E.A. Boyle, E. MacArthur, T. Hainey and J.M. Boyle: A systematic literature review of empirical evidence on computer games and serious games. In: Computers & Education 59 (2), 661–686. DOI: 10.1016/j.compedu.2012.03.004 (2013).
[3] Stewart, J., Bleumers, L., Van Looy, J., Mariën, I., All, A., Schurmans, D., Willaert, K., De Grove, F., Jacobs, A., and Misuraca, G.: The Potential of Digital Games for Empowerment and Social Inclusion of Groups at Risk of Social and Economic Exclusion: Evidence and Opportunity for Policy. Centeno, C. (Ed.), Joint Research Centre, European Commission. (2013).
[4] G.W. van der Vegt, W. Westera, E. Nyamsuren, A. Georgiev and I. Martinez Ortiz: RAGE architecture for reusable serious gaming technology components. In: International Journal of Computer Games Technology. Article ID 5680526. DOI: 10.1155/2016/5680526. (2016).
[5] W. van der Vegt, E. Nyamsuren and W. Westera: RAGE Reusable Game Software Components and Their Integration into Serious Game Engines. In: Proceedings of the 15th International Conference on Software Reuse (ICSR 2016). Springer International Publishing, Basel, 165-180 (2016).
[6] RedMonk: The RedMonk programming languages rankings: January 2015, http://redmonk.com/sogrady/2015/01/14/language-rankings-1-15/, last accessed 2018/05/15.
[7] Davis, F. D.: Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. In: MIS Quarterly 13(3): 319-340 (1989).
[8] Marangunić, N. and Granić A.: Technology acceptance model: a literature review from 1986 to 2013. Universal Access in the Information Society 14(1): 81-95 (2015).
[9] Lund, A. M.: Measuring Usability with the USE Questionnaire. STC Usability SIG Newsletter (2001).
[10] Furneaux, B.: Task-Technology Fit Theory: A Survey and Synopsis of the Literature. In: Information Systems Theory: Explaining and Predicting Our Digital Society, Vol. 1. Y. K. Dwivedi, M. R. Wade and S. L. Schneberger. New York, NY, Springer New York: 87-106. (2012).
[11] Bangor, A., et al.: Determining what individual SUS scores mean: adding an adjective rating scale. J. Usability Studies 4(3): 114-123 (2009).
[12] Siegmund, J., et al.: Measuring and modeling programming experience. In: Empirical Software Engineering 19(5): 1299-1334 (2014).
[13] W^{3}Techs: Usage statistics and market share of Java for websites, https://w3techs.com/technologies/details/pl-java/all/all/, last accessed 2018/05/15.
[14] Cronbach, L. J.: Coefficient alpha and the internal structure of tests. In: Psychometrika, 16, 297-334 (28,307 citations in Google Scholar as of 4/1/2016). (1951).
[15] Tavakol, M. and Dennick R.: Making sense of Cronbach's alpha. Int J Med Educ 2: 53-55. (2011).
[16] Microsoft: Character Encoding in .NET, https://docs.microsoft.com/en-us/dotnet/standard/base-types/character-encoding, (2017).
[17] Microsoft: Convert Methods, https://msdn.microsoft.com/en-us/library/system.convert_methods(v=vs.110).aspx, last accessed 2018/05/15.
[18] Lacovara, R.: How To Create XML in C# with UTF-8 Encoding, (2011),

http://rlacovara.blogspot.nl/2011/02/how-to-create-xml-in-c-with-utf-8.html, last accessed 2018/05/15.
[19] Unity: Unity Blog: Unity 2018.1, https://blogs.unity3d.com/2018/05/02/2018-1-is-now-available/, last accessed 2018/05/15.
[20] Saveski, G., Westera, W., Yuan, L., Hollins, P., Fernández Manjón, B., Moreno Ger, P. and Stefanov, K.: What serious game studios want from ICT research: identifying developers' needs. In: Games and Learning Alliance Conference 2015, Rome (2015).
[21] Nystrom, B.: Game Programming Patterns. Genever Benning (2014).
[22] NuGet Gallery, https://www.nuget.org/, last accessed 2018/09/18.
[23] Unity3D, https://unity3d.com/, last accessed 2018/10/22.
[24] MonoGame, http://www.monogame.net/, last accessed 2018/10/22.
[25] Cocos2D, http://www.cocos2d.org/, last accessed 2018/10/22.
[26] Xamarin, https://visualstudio.microsoft.com/xamarin/, last accessed 2018/10/22
[27] Bazzanella, B., Casagranda, M., Molinari, A., Humphreys, S., Sleightholme, G., Lepoivre, O., ... Kommeren, R. (2018). D5.4 – Pilots quality report round 2. RAGE project. https://research.ou.nl/en/publications/d54-pilots-quality-report-round-2, last accessed June 24, 2019.
[28] Steiner, C., Gaisbachgrabner, K., Nussbaumer, A., Mertens, J., Hemmje, M., Nadolski, R. J., ... Santos, P. A. (2018). D8.4 – Second RAGE Evaluation Report. RAGE project. https://research.ou.nl/en/publications/d84-second-rage-evaluation-report, last accessed June 24, 2019.