Powwow: A tool for collaborative software jam sessions

Paul Muntean, Damir Ismailović, Sebastian Paetzold and Bernd Bruegge
Department of Computer Science
Technische Universitaet Muenchen
Muenchen, Germany
{ paul.muntean | damir.ismailovic | sebastian.paetzold | bernd.bruegge }@cs.tum.edu

***Abstract*—The increasing time, complexity and cost of today video game development projects demand for new software tools capable to support the development of fast runnable video game prototypes. This paper presents our conceptual solution for building runnable video game prototypes. For editing video game levels we added editing capabilities in a mobile software tool. We present a mobile software tool which supports editing video game levels. Furthermore, the tool supports testing of previously designed levels with the help of design recognition mechanisms that facilitate loading and simulating of games. The tool supports distributed, collaborative design sessions as it is based on a client server software architecture. Game designers collaborate by sharing their designs on a file server from where the files can be pulled by other users for further editing. This tool is used for software jam sessions in order to support local collaborative game level design. In addition to the tool we present a tool evaluation in this paper. We introduce software jam sessions in order to be able to support local collaborative game level design. In the last part of the paper, we evaluated our tool. The goal is to find out if the working efficiency increases when developing video game levels from scratch with our tool.**

Keywords-Collaborative tool; map based video game; fast prototyping; software jam sessions

I. INTRODUCTION

The usage of the paper and pencil to capture requirements in the video game industry is causing delays and engineering overhead during the development process [1]. Thus, shifting deadlines and increasing costs negatively influence the software development process.

When conceiving new game ideas, game designers should not stick to traditional methods of capturing requirements and then develop costly prototypes. We foresee that game designers should have an easy to use tool that they can carry easily around and that helps to test ideas in a matter of minutes. Such kind of tool can link the requirements elicitation phase to the implementation of fast runnable video games. Users with no programming experience should be able to use the tool right away.

When developing mobile applications and addressing user experience there is a great need for creating many user interfaces [11]. Every video frame of a game can be abstracted to a single user interface. We think that a software tool should have game editing features and an integrated simulator that help to build fast runnable games at a fair cost of time and effort.

Nowadays, it does not exist a specialized process model for mobile applications but it can be observed that large development projects have moved away from a process-intensive approach toward a more agile approach, with the Scrum approach and other agile techniques, e.g., test driven development, finding widespread acceptance [11].

With the emergence of new mobile devices and the begining of the post-PC era [12] new posibilities arrise that can help to speed up the development of runnable video game prototypes. During a project requirements meeting with the customer a software tool should facilitate not only requirements elicitation support but also the development of a runnable game prototypes. With such a tool we want to close the gap between requirements elicitation and the implementation by being able to develop fast runnable game prototypes that help to get fast feedback from the customer. In this regard, our solution is thought to be focused on a specialized tool deployed on a mobile device accompanied with the corresponding development technique.

We have identified three main aspects regarding our conceptual solution:

Editing: The user should be able to place different objects and elements on a grid map and immediate feedback should be provided by the user interface.

We think that a grid is needed to help the user place the game elements on predefined places on the grid. This represents in our opinion the framework for other game genres because a plane is common to: FPS, Soccer, car races games, etc.

Collaborative design: Firstly it should be possible to work collaboratively in near real time and in an asynchron manner where designers collaborate and share ideas over a file server. We call this approach distributed collaborative design.

Secondly it should be possible for designers to work collaboratively on the same prototype in real time. The users should get instant feedback from others. By tailoring tasks and profiting from shared team knowledge. We think that the process of collaborative video game editing can be performed more efficiently. We call this local collaborative design.

Thirdly a development technique should be defined that contains a set of rules and guidelines which do not constrain the participants but rather help them to profit from the setting type in which they are working. This technique should help to achieve real time collaborative work. The second and third

 ISBN: 978-1-61208-287-5

aspects remain currently work in progress and we will present only the achievements that our research has produced until the moment of writing this paper.

Design recognition: The collaboratively designed levels can be interpreted in different ways. We want to be able to load them into the integrated game simulator and recognize all previously designed game elements consistently. The goal is to test the game playability.

The paper is structured as follows. The section 1 contains the problems related to development of video games on mobile devices and the description of our solution used to achieve fast collaborative video game development. Related work is presented in section 2. In section 3 we analyze the development method and tools used in the Battlefield Wars case study and in section 4 we present our developed tool. Section 5 contains work in progress about software jam sessions. In section 6 we evaluate the Powwow tool. Section 7 contains our conclusions and future work.

II. RELATED WORK

At the moment of writing this paper there were only a few apps in the App Store that support fast video game prototype development.

The *Codea* app, [2] offers the possibility to modify code of the already deployed video game app. This approach provides deep level control on how the app behaves during user interaction with it. The *PGC* app, [3] is based on predefined prototypes of video games. Haladjian et al. [5] present a quick prototyping tool that is based on code generation that can be used by users with no programming experience to develop physics based game levels. The *Battle Map 2*, [4] app is designed for building map based games levels. This approach is interesting since it wants to be a replacement for pencil and pen when conceiving new game ideas.

These tools lack collaborative work capabilities and demand programming experience. Furthermore, they restrict the user by providing a fixed number of game templates.

Game designers that have no programming experience should have the possibility to use a tool right away. Limiting game designers with predefined game templates constrains game creativity in our opinion. In order to develop complex games, collaboration mechanisms should be incorporated also in an app.

The overall video game development is not suited for typical software life cycle methods, such as the waterfall model [6]. The stages of development are done in a serial manner linking the project phases rigidly together maintaining a high project risk during the whole project. Thus, requirements updates are difficult to be performed. To close the gap between user model and design model [9] specialized tools that support informal communication are needed [7]. We believe that a video game development technique supported by tools can speed up the development process of video games.

The design at run time concept described in [8] is used in the context of reconfigurable ubiquitous software systems. The design at run time concept can be extended for developing video games using an iterative developing technique.

In our approach we have assumed that the designers have no programming experience so we choose not to expose code-editing features as in the previously mentioned example.

III. CASE STUDY: BATTLEFIELD WARS PROJECT

The *Battlefield Wars* project had the goal of producing a framework that allows light interaction between users and map-based games and supporting development of multiuser map-based video games.

The goal of our *Battlefield Wars* case study was to find out how real video game projects are developed by an experienced team of video game developers from the point of view of software tools used and development methods. The first research question was, *RQ1: Do developers use specialized tools for developing map based video games? If yes which ones?* The second research question is the following, *RQ2: Which development technique or process model do developers use? Is this adapted to the special needs of video game projects?*

We have observed the team of developers during their four weeks development work. We were for two days per week with the developers and wrote down every detail regarding tools and development methods used. Firstly we have focused on how project tasks were addressed and solved. Secondly we were concerned with inter-team communication during the project and how this has influenced the project outcome.

The results of the case study revealed that at the beginning of the project until the end the developers have worked independently and without using specialized tools.

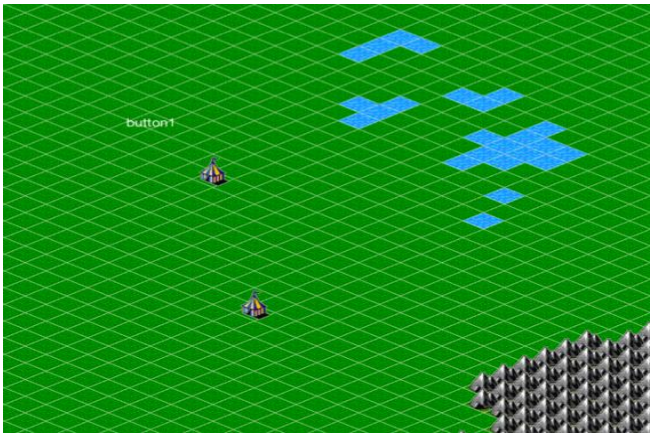

Figure 1. Battlefield Wars game

Co-located Collaborative work was not possible since the developers did not use any specialized tools that support collaboration. The development team has used the waterfall development model being forced to stick to a sequential development style. The developers reported that it was

difficult to add new requirements to the product backlog and that creativity was "damped" because of the incapacity of team members to efficiently communicate and test their ideas. The developers agreed to use in the future agile development methods and they suggested that they need an iterative and adaptive development technique tailored to their needs.

IV. THE POWWOW TOOL

The Powwow tool prototype that we have developed is based on our conceptual solution. It is available for iOS and can be deployed on the iPad.

The editing features are available in hidden pop up menus that appear by tapping on the buttons placed on the two tab bars located in the upper and lower part of the screen. The main aspects of the tool are highlighted with numbers (1-7). Not to clutter the UI we have chosen to have two fixed tool bars in the upper and lower part of the screen.

The editing, persisting, sharing and simulation features are available by tapping on the buttons present on the two toolbars. The number (1) indicates an initial map where every tile of the map represents a second freely editable map. In Fig. 2 we have a red dot, near number (1), representing a house. Tapping the red house tile the user opens a second "endless map view" where all the previously saved map elements can be further on edited. The user has the possibility to zoom in and out when editing so that the tiles do not appear too small as in Fig 2. After pressing one of the buttons located above the number (2) the user has the possibility to select different layers of the map, to save, delete, position, undo/redo and to center the map on the screen.

After the saving process is done all previously added elements are saved in a TMX meta-format file which can be easily parsed and shared with other designers.

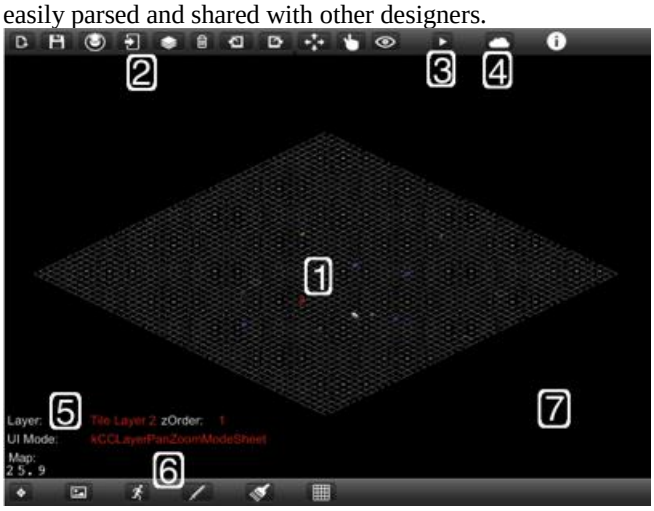

Figure 2. Powwow tool prototype

During the saving process a second file format is saved representing the same game level. This file contains all game elements, which are objects, in a serialized form. We used this second format because of performance reasons, mainly because it can be loaded and saved faster then the TMX file format, which has to be parsed. We also use the second object files for presentation reasons on the first map view indicated with number (1).

After successfully loading the game level we observe that all game elements from the TMX file are present on the level. The level can be further on edited on other iPads that run the Powwow app or with the help of the desktop program Tiled [10]. Successful design recognition consists in the TMX file parsing and the game objects instantiation.

Number (3) indicates the play button which triggers a sub view when pressed, where we can select a previously saved game level and play on it in order to find out if the game logic fits our needs and decide if further editing is necessary.

The Powwow users indicated in Fig. 3 can push all locally available game levels on the distributed server and pull all the remote available levels on their iPad. The user also has the possibility to erase every locally and remote available file. The users can be located in different locations and can collaborate by sharing these files. The files can be further on edited and pushed on the distributed file server. Every user has the possibility to play on the level that he is currently editing. The only restriction is that the level has to be previously stored on the iPad.

Number (4) represents a button with a cloud. When pressing this button, a sub view appears which asks the user if he wants to connect the Powwow tool with a distributed file server, which offers file services. After accepting this option another sub view appears offering the options of pushing, pulling, local deletion, remote deletion and disconnecting from the distributed server. At this stage of development we have added all our options for collaboration in this submenu.

The *File Server*, Fig. 3, files can be synchronized with local file folders distributed on desktop PCs. This offers the possibility to edit the prototype files on the PC by using the program called Tiled [10]. In order to be able to collaborate locally in real time without having to use a distributed file server we want to add real time capabilities to the Powwow tool. This issue will be addressed in section 5.

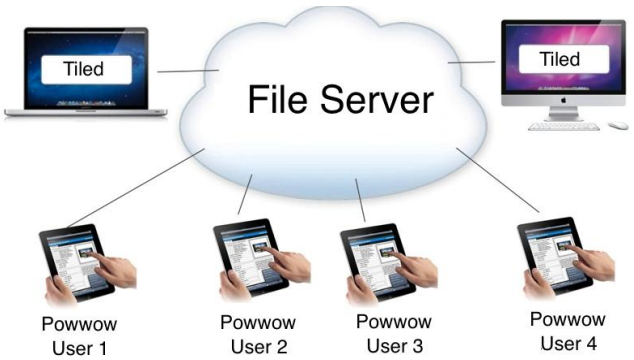

Figure 3. Distributed collaboration infrastructure

Number (5) labels indicate the current editing layer; the current selected drawing mode and the number of FPS (Frames per Second) available are also indicated. These labels can by hidden if needed.

 ISBN: 978-1-61208-287-5

Number (6) indicates several buttons that contain pop up menus with objects that can be added in order to edit our game level. The assets are at this stage of development restricted to only several types of objects. We also have a brush with different brush sizes that can be selected during editing.

Number (7) indicates a black area where the user cannot add tiles. The initial map can be dragged around on the black area by performing sweep gestures. This is handy when positioning and zooming the initial map and the second "endless map".

Design recognition is achieved by parsing the TMX file and then instantiating game objects. These are used afterwards to populate the game level.

V. COLLABORATIVE SOFTWARE JAM SESSIONS

This section represents work in progress and addresses the local collaborative design aspect presented in the introduction. We want to address the collaborative design not only from the tool point of view but also from the process development technique perspective. Our collaborative software jam sessions concept aims at porting the concept of musical jam sessions to quick games prototyping. The whole concept relies on the idea of jamming together, Fig. 4, in a group when developing a video game level.

The video game level artifacts represent the components that compose the game level. We want to design software instruments capable to build these artifacts. In order to design these instruments we need to identify the type of relations between instruments and artifacts.

The Fig. 5 contains the *JAMInstrument* class with which we modeled a software tool capable to produce different types of game artifacts represented by the *Artefact* class. These artifacts represent game level logic and level design assets. The produced artifacts compose our video game level.

Figure 4. Local collaborative jam session concept

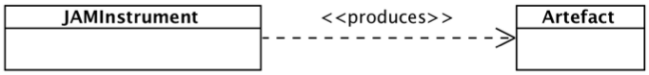

Figure 5. Jam session meta model

The Fig. 8 represents a high level view on the jam session technique. It illustrates our software jam session concept that we think it is superior in this context to the waterfall development process. Collaborative work increases communication and encourages knowledge sharing between participants that have different backgrounds. The jam session technique relates to the agile methods because it is an iterative and incremental activity that supposes that team members can organize themselves. As we perform research on this topic we think that this diagram will suffer further modifications.

The four swim lanes represent our main project stakeholders: user/player, customer, designer and tester. Looking at the Fig. 8 from left to right we have in the beginning of a jam session the requirements elicitation activity.

The first activity is the kick-off meeting where one user story is selected from the backlog and the working strategy is discussed with all stakeholders. After this phase the software jam session starts and we observe here work done in parallel and collaboratively.

In the second activity game designer designs game assets, the developer develops game logic code, the user plays/tests the current runnable prototype. Every stakeholder has the possibility to review the game level prototype at any instant in time. After the first iteration we have a wrap-up discussion where the participants analyze how the tasks were accomplished and further on feedback goes in the continuous development activity.

After several iterations we have the review product activity. At the end of each iteration we have a potential shippable product increment. Again feedback goes directly to each participant and to the continuous development activity. This helps to reduce the risk of ill-defined requirements and helps to update the product backlog. It provides a mechanism for collaborative knowledge sharing that helps the participants to improve themselves for the next jam sessions. In the end of the process the result is a potential shippable product.

We think that the software jam session technique can help to improve collaborative work by allowing 7+/-2 participants to design together in the same location a video game. We believe that software jam sessions will encourage knowledge sharing between participants and enforce creativity.

Currently, we are capable to create collaboratively a playable video game level in a matter of minutes with the help of the Powwow tool. Without having real time capabilities built in the tool yet.

We think that collaborative design with the support of the jam sessions technique can speed up the development of complex game levels where workload has to be tailored between participants.

VI. PRELIMINARY EVALUATION

We conducted a quasi-experimental study where we measured the time needed for developing a video game prototype with the Powwow tool.

First we describe how we tested and then show the results together with our framework current limitations. In Fig. 6 the *X* axis represent the five users and the *Y* axis the time measured in minutes. The blue, red and green color represents the three runs each user made. In Fig. 7 the *X* axis represents questions and the *Y* axis points.

 ISBN: 978-1-61208-287-5

We wanted to find out if the working efficiency is increasing when using the Powwow tool instead of writing code and if the tool is usable for developing runnable video game levels.

The quasi-experimental study was performed with 5 testers. The testers had never used the Powwow tool previously. We used shadowing to observe the testers during the experiment. The description of the task was provided to the testers at the beginning of the experiment. The time needed to complete the task was measured.

The study contained two parts. First the tester was introduced to the Powwow tool, which took around 5 minutes, the task was given to the tester, after finishing reading the task the time keeping was started, the tester finished the task, the time was stopped, the results were evaluated. The total time for each user was around 20 minutes.

The testers got on a sheet of paper the following task. *Please design a prototype having: one player, one enemy, one friend, one live item, one house, one tree and a five by five squared plane. Save the prototype. Simulate the prototype. Share the prototype onto the distributed file server.*

After each tester finished the task, they got a questionnaire, Table I., with five qualitative questions. All the questions had to be answered by checking a checkbox associated to each question. The possible answers were presented on a scale: 1 point (unsatisfactory), 2 points (satisfactory), 3 points (fair), 4 points (good), 5 points (very good).

The results show that one Powwow tester needs around three minutes to complete the task and the other 4 testers need between 5 and 10 minutes. Afterwards four users wanted to try the tool again.

TABLE I. THE SAMPLE QUESTIONAIRE

Questions	1	2	3	4	5
Q1:How do you find the usability?					
Q2:Are the tools implemented usefull?					
Q3:Did you had dificulties during prototype design?					
Q4:Are the pictures used for the buttons appropiate?					
Q5: Would you recommand the tool to a friend?					

The colors in the Fig. 6 represent the runs for each tester. After the second run almost all users improved their times. The time values presented in Fig. 6, of 0 min, 3.30 min, 50 sec, 2.20 min and 1.34 min represent the time difference between the first run and the third run for each of the testers.

Also we can observe that a learning curve appears for each of the testers. This means that the time needed to accomplish the same task reduces after the first attempt. Fig. 7 indicates that only 7 answers from a total of 25 answers are under the 2.5 average values. This means only 28% of the answers have obtained under 2.5 points, the maximum value being 5. Thus, 72% of the answers lie between 3 and 5 points in the Fig. 7.

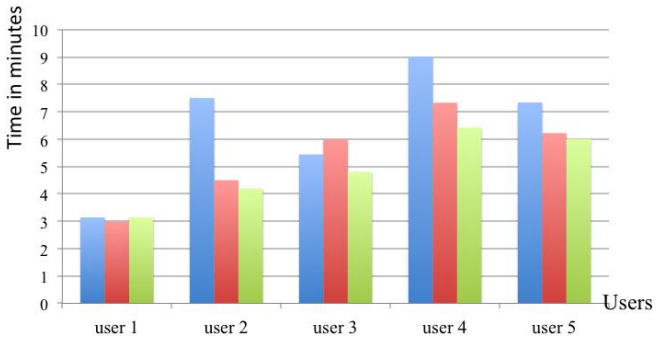

Figure 6. Time need for each tester

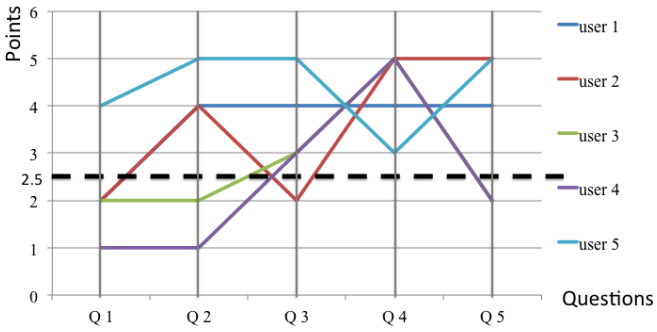

Figure 7. The results of the questionaire

The Powwow tool is not capable to support real time jam sessions yet. This issue has to be addressed in the future in order to achieve real time feedback during collaborative design of game levels. We did not test our collaborative jam session concept presented in section 5 because this represents work in progress. We plan to do this experiment also in the future when the Powwow tool is capable to support real time collaborative work.

VII. CONCLUSION AND FUTURE WORK

In this paper we described Powwow, a software tool that represents an alternative for developing fast runnable video game prototypes. Powwow can be used in the requirements elicitation phase and during the software jam session that we introduced in this paper too. The tool enforces communication and knowledge sharing through the interactive development work style.

We have made a case for rapid game prototyping as it can help to close the gap between the design model and user model. This does not mean that we not believe in the standard approach of firstly gather requirements and then develop incremental prototypes. We think that our tool is an alternative of gathering requirements and building fast runnable video game prototypes right away from the first

 ISBN: 978-1-61208-287-5

meeting with the customer.

By binding all the editing stages presented in section 4 together and implement in further releases of Powwow mechanisms that support real or near real time collaboration. We think that the tool should perform all the synchronization operations with the local file folder and the distributed file server independently.

In our case study we found out that developers need specialized apps for developing and testing RPG (role playing games) games. They would like to have a tool where they can right away test new game ideas without having to write necessarily code.

We also have introduced collaborative software jam sessions as an alternative development technique to take advantage of creative and ambitious game designers.

Finally, we have presented a preliminary study where we used our tool in order to develop runnable game prototypes. Until now we are able to work collaboratively with the Powwow tool and can develop a video game level in matter of minutes. We think that this result will motivate us to focus in the future on the development of real time capabilities in order to perform local collaborative jam sessions too.

ACKNOWLEDGMENT

Special thanks go to Damir Ismailović, the leader of the DRG (Dance Research Group). We would like to also thank the members of the DRG who provided key insights during the research phase of this paper.

REFERENCES

[1] D. Wuest, N. Seyff, M. Glinz, "Flexible, lightweight requirements modeling with flexisketch" Fourth International Conference on Mobile Computing, Applications and Services (MobiCASE), Springer-Verlag, Berlin, Heidelberg, October 2012, pp. 225-244, doi: 10.1007/978-3-642-36632-1_13.

[2] Codea app, http://twolivesleft.com/Codea/, retrieved: March, 2013.

[3] PGC app, https://itunes.apple.com/us/app/prototype-game-crafter-1.0/id419847669?mt=8, retrieved: March, 2013.

[4] Battle Map 2 app, https://itunes.apple.com/us/app/battle-map-2/id384800918?mt=8, retrieved: March, 2013

[5] J. Haladjian, D. Ismailović, B. Köhler and B. Bruegge, "A quick prototyping framework for adaptive serious games with 2D physics on mobile touch devices" IADIS International Conference Mobile Learning, March 2012, pp. 197-204, ISBN (Book): 978-972-8939-66-3.

[6] B. Bates, Game Design 2th edition. ISBN-10: 1592004938. Course Technology PTR, 2004, pp. 225.

[7] B. Bruegge, S. Krusche and M. Wagner, "Teaching tornado from communication models to releases" (EduSymp12), Proceedings of the 8th edition of the Educators' Symposium, Innsbruck, Austria, October 2012, pp. 5-12, doi: 10.1145/2425936.2425938.

[8] A. MacWilliams A Decentralized Adaptive Architecture for Ubiquitous Augmented Reality Systems Dissertation, Technische Universität München, June 2005, pp. 121.

[9] D. A. Norman and S. W. Draper, User centered system design; new perspectives on human-computer interaction. L. Erlbaum Associates Inc., 1986.

[10] Powwow tool, https://www.youtube.com/watch?v=njS2caLWFRI, retrieved: March, 2013.

[11] Tiled program, http://www.mapeditor.org/ , retrieved: March, 2013.

[12] A. I. Wasserman, "Software Engineering Issues for Mobile Application Development. " Proceeding FoSER '10 Proceedings of the FSE/SDP workshop on Future of software engineering research, 2010, pp. 397-400, doi: 10.1145/1882362.1882443.

[13] E. M. Maximilien and P. Campos, "Facts, trends and challenges in modern software, International Journal of Agile and Extreme Software Development, Vol. 1, No. 1, July 2012, pp. 1-5, doi: 10.1504/IJAESD.2012.048305.

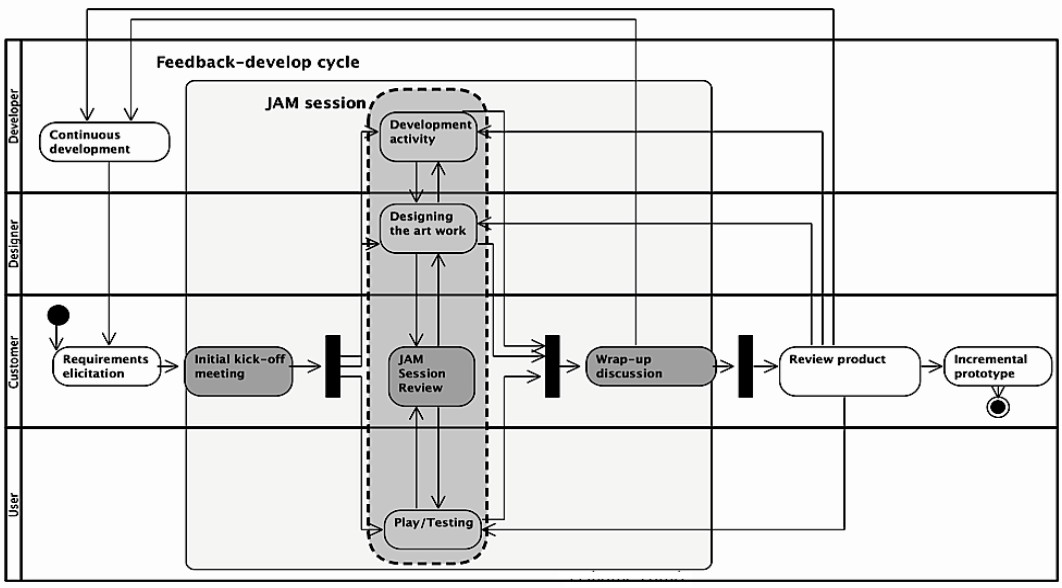

Figure 8. Collaborative software jam sessions