Collaborative Software for Coding Workshops

Jennifer Sparks

April 2018

1.0 Introduction

The number of women working in the technology industry has been steadily decreasing in the last 20 years (Petronzio, 2016). Fewer women are taking computer science degrees than ever before. This lack of gender diversity in the IT industry means that there is a lack of diversity in the software being produced (Burnett, et al., 2016).

The software design described in this project supports collaborative coding in a learning environment. The program will be used when coding tasks are being carried out, such as during workshops. It uses existing technologies that allow multiple users to edit the same file at the same time. This system has been developed using the coding techniques of pair and mob programming (Zuill, 2014). Mob programming is an extension of the Agile pair programming technique which is part of the Extreme Programming (XP) framework (Measey, 2015). In pair programming one person is the driver and the other the navigator, in mob programming there is one driver and everyone else is a navigator. The driver has control for a set amount of time and then the role is allocated to another member of the group. The driver is the only person who can update the code. As well as improving the quality of the code produced, these methods are also good for knowledge sharing and teaching (Stackify, 2017). There are no other applications currently available that combine collaborative computing and mob programming.

The design addresses the issues raised in the PWC report about women in tech (PWC, 2017). The survey suggests that girls don't choose STEM subjects because they are not regarded as creative subjects, teachers do not make STEM subjects appealing and STEM subjects are not as interesting as other subjects. Aspects that will make the software more appealing to girls and women have been incorporated into the design. It has been shown that '… collaborative pedagogical techniques … can increase the success and satisfaction of female students' (Berensen, et al., 2004).

There are many collaborative coding programs available but they are mostly text editors and work in the same way as Google Docs. Multiple people can edit one file at a time but the results of the code cannot be viewed. An exception to this is the collaboration feature in JSFiddle. The program uses TogetherJS (Mozilla, 2013), a JavaScript library developed for collaborative coding by Mozilla. JSFiddle also has simulated AJAX calls, so any changes to the code are displayed immediately. This functionality has been used in the design of this project.

2.0 Product Functionality

The software is designed to be run on several workstations in a teaching environment. The design will allow multiple users access to one computer program file at the same time. Only one of the users can edit the file while the others have read only access. This control is managed by using roles for the participants. In addition to the driver and navigator roles there is also a facilitator (instructor or teacher) role, someone who has overall control of the system.

To enable everyone to contribute in a session, the groups should ideally be less than six people. The driver will usually have control of coding for about 15 minutes, then control will be handed over to the next person. The team discusses options and contribute ideas either through speech or the chat window, which the driver then codes. Chat can be viewed by the whole group but there is no individual chat available. The navigators can view any part of the program and return to the same view as the driver's screen with a button click. The driver's view can also be brought up in a smaller window in the side panel.

The software can also be used for teaching in a classroom environment with all students having read only access to the program. The instructor could allow update permission to particular users if required. Both the workshop group and classroom options would also benefit from the driver screen view being projected for everyone to see.

The main window displays the code with tabs for the JavaScript, HTML and CSS (Appendix A). A collapsible side panel displays the chat window and the resulting webpage. The current driver and next driver are highlighted with the time remaining before the change of driver is displayed. The list of all the members of the group can be displayed. All of the side panel options can be collapsed individually.

3.0 Design Considerations

3.1 Collaborative Interaction

Each aspect of CSCW (Computer Supported Collaborative Work) is supported by the program. The main type of collaboration is co-located synchronous use in a classroom or workshop environment. Working asynchronously is possible but would need support from a version control system such as Github which allows forking. This functionality is not the main purpose of the software as it is already available in other applications.

Collaboration Type	**Software Situation**
Co-located / synchronous	Classroom / joint session
Co-located / asynchronous	Classroom / own work (copy of file)
Distributed / synchronous	Other location / joint session
Distributed / asynchronous	Other location / own work (copy of file)

Although the members of the group are usually co-located they are also represented in the system. A profile picture with a colour coded border will identify them to the rest of the group. The cursor of the driver editing the file is highlighted and the updates are displayed in real time on every group member's screen.

3.1.1 Supporting Awareness

Different types of awareness are supported by the system.

i) Situational awareness

Navigators can see where the driver is editing. The main screen can be used to scroll through the program to show where the editing is taking place in the file. This gives the users context so that they always know what is happening.

ii) Social awareness

Users can view who else is in the group. The chat window allows users to gauge other people's mood with implicit and explicit communication. With a co-located group there will also be face to face communication.

iii) Workspace awareness

The system provides information about who is currently driving the session, who the next driver will be and how long before the change of driver. A warning of the change will be given in advance.

iv) Team Awareness

The window displaying the website generated by the code shows how the team is doing. The whole team knows whether the program works or not.

v) Awareness of status and expertise

The users know who is driving the session at any one time. The facilitator has overall control of the system.

Users can use direct observation to see where the program is being edited and by who. They will use peripheral awareness to keep track of any chat going on and can choose whether to take part or not. The chat window can be closed to reduce the cognitive workload and increase focus on the task. Historical awareness will be maintained by having all changes recorded and rollback functionality.

3.1.2 Support the Creation of Online Communities

Although this software is primarily for teaching in a classroom environment it can also be used for instructing distributed groups. For this type of use some measures have been introduced to prevent malicious use of the system. The facilitator determines who can join the group, all members must register with their student id and all changes to the program are tracked.

3.1.3 Support Referencing

Users have reference points in the system to refer to other parts of the system. Here are some of those features:

- The driver is identified and any recent changes they make to the code are highlighted
- Any references in the chat can be linked back to the point in the code being discussed
- The code window is a source code editor and supports syntax highlighting, indicate code in brackets etc.

3.1.4 Other Collaborative Design Considerations

The coding work is split equally among the group, all members of the group contribute and each member takes a turn at being the driver. The collapsible sidebar and sections within it allow the users to reduce the interference of collaboration features if they need to concentrate on a particular section of the code.

3.2 Distributed Cognition for Teams

This design is a socio-technical system relying on social interaction and knowledge sharing in a technical environment. It is a way of solving complex tasks in a team, where people have different roles. Cognitive processes are distributed across the members of the group, the users are all contributing to the writing of the program. The distributed cognition for teams (DiCoT) framework has been used to inform the design of the system. The following principles were considered during the design process:

3.2.1 Physical Layout

i) Space and Cognition

The driver's screen can be projected to support awareness. If the group is co-located they can communicate with each other face to face. Artefacts to support creative thinking will also be available such as individuals' notepads and whiteboards for brainstorming.

ii) Perceptual principle

Users may have printed learning materials such as study notes, to refer to for tips on coding. Each member of the group will be able to communicate with others and will know which person is driving.

iii) Naturalness principle

The users are represented as icons and the computer program is represented as lines of code, not the compiled machine code. The study notes support cognition.

iv) Subtle bodily supports

Users can ask direct questions or attract the instructor's attention for help.

v) Situation awareness

Group members use the workstation screen, projected screen or discussions within the group to raise situation awareness.

vi) Horizon of observation

For co-located use each user will be able to see the projected screen and ideally each other.

vii) Arrangement of equipment

Ideally the workstations will be small enough for everyone in the room to see everyone else. Larger individual screens are better for coding but could inhibit the view and increase isolation of the users.

3.2.2 Information Flow

i) Information movement

Information moves through the system with conversations and interaction through the workstations, via chat and the code. Whiteboards and paper are also used to transfer information between the group.

ii) Information transformation

The driver is the only person who can update the program, they write down what the navigators say and their own ideas. It may not be exactly what the navigators intended.

iii) Hubs

The server holds the main program, but information is also held in the chat, on the students' notes, whiteboards and handouts.

iv) Buffering

The information flow in the system may be held up when the coding process is put on hold. This may happen if the driver needs clarification or the group needs to be brought up to speed. A delay may occur for the instructor to explain more complex techniques.

v) Communication bandwidth

All coding is performed by the driver, their speed of typing will be a limiting factor.

vi) Informal communication

This can occur with chat between neighbouring members of the group during the session or outside the session. The information can then be fed into the session later.

vii) Behavioural trigger factors

Replying to chat is a response to a trigger.

3.2.3 Artefacts

i) Mediating artefacts

The window showing the website generated by the code is a mediating artefact. It represents the state of the system.

ii) Scaffolding

Support is provided by enabling the user to scroll through the code on the main screen and display the driver's screen in a separate window.

iii) Goal representation

Writing a successful program is the goal of the student. As the program is being written the resultant website is displayed in another window. This demonstrates the degree to which the goal has been achieved.

iv) Coordination of resources

The program is written with input from the navigators and the driver. The server holds the file and runs the code to display the resultant website. This is all done with coordination of different resources.

3.3 Other Design Considerations

This system is made up of several windows, to reduce navigation excise the windows can be minimised. The design is simple, only relevant information is displayed, this supports the locus of attention. The data entry process is easy as it uses a software editing tool (Notepad++), this enables autofill and awareness of brackets etc.

3.4 Designing Gender Neutral Software

The collaborative aspect of this software will encourage women to use it. GenderMag (Burnett, et al., 2016) is an analytical way of evaluating software's gender inclusivity. The method identifies five facets of gender differences, when designing the system these issues were considered.

Gender Facet	**System considerations for female users**
Motivation for using software	The software will produce a final product, it has a purpose, it is a social application
Style of processing information	The information available is comprehensive so decisions can be made more easily
Computer self-efficacy	The system is easy to use and intuitive which reduces self-confidence issues
Risk aversion	Rollback is available and visible to the users, this increases risk taking
Willingness to Tinker	Users feel comfortable using the software and can experiment playfully

4.0 Feedback

For the report I have incorporated the comments made in the proposal feedback (Appendix B). I have identified what the software will do in the introduction. The main models I used for the design are collaborative computing and distributed cognition. The software will use a combination of existing applications such as TogetherJS and Notepad++. The information workflows have been identified in the section on Information Flow (3.2.2). The features supported are found in the Product Functionality (2.0) section. The report itself has been given a better structure than the proposal.

5.0 Conclusion

The research for this design has helped me to understand the role of cognition in collaborative design. The support of awareness has made me think about aspects in the design which I wouldn't have previously considered. The principles of distributed cognition have been useful in designing the system and how the users interact with the software and the environment.

If this new software can encourage girls into coding and increase diversity in the technology industry I feel that it has been a very worthwhile project.

6.0 References

Berensen, S. B., Slaten, K. M., Williams, L. & Ho, C.-W., 2004. Voices of women in a software engineering course: reflections on collaboration. *Journal on Educational Resources in Computing (JERIC) - Special Issue on Gender-Balancing Computing,* 4(1).

Burnett, M., Peters, A., Hill, C. & Elarief, N., 2016. *Finding Gender-Inclusiveness Software Issues with GenderMag: A Field Investigation.* San Jose, CHI 2016.

Measey, P., 2015. *Agile Foundations Principles, Practices and Frameworks.* 1st ed. Swindon: BCS Teh Chartered Institute for IT.

Mozilla, 2013. *TogetherJS.* [Online]
Available at: https://togetherjs.com/
[Accessed 14 April 2018].

Petronzio, M., 2016. *8 ways you can empower girls to learn coding.* [Online]
Available at: https://mashable.com/2016/01/27/girls-coding-how-to-help/#tExeJteoGSq6

PWC, 2017. *Women in Tech.* [Online]
Available at: https://www.pwc.co.uk/who-we-are/women-in-technology/time-to-close-the-gender-gap.html

Stackify, 2017. *What is Pair Programming? Advantages, Challenges, Tutorials & More.* [Online]
Available at: https://stackify.com/pair-programming-advantages/

Zuill, W., 2014. *Getting Started with Mob Programming.* [Online]
Available at: https://www.agileconnection.com/article/getting-started-mob-programming

7.0 Appendices

Appendix A – Design Sketch