CAPSTONE S19-074:

ALTERNATIVE APPLICATIONS FOR AGILE METHODOLOGIES

RESEARCH PROPOSAL

BEN KELLY – 1254451

CONTENTS

INTRODUCTION

This research project looks at the handling of user-centred and non-functional requirements with the current applications of agile methodologies. The research and development outlined in this proposal looks at the current limitations of existing agile applications, and asks how we can develop an alternative workflow model to better suit user driven projects.

The topic premise was developed from my experiences with agile teams in small to medium sized development teams focused on user facing product design in both the workforce and with university programs. This project is co-ordinated through the UTS engineering capstone program and is planned to be developed with UTS programs such as the Software Development Studio and Games Studio, and potentially with local development companies for the research and development of new tools and practices.

PROBLEM DEFINITION

Agile development is focused primarily on the mapping of functional requirements, which is essential for providing deliverables and releases, but have trouble dealing with non-functional requirements (NFRs). Attempts like NORMAP (Farid 2012) have been proposed to deal with these, especially the more measurable NFRs such as security and performance. However, these processes are still challenged with less quantitative NFRs, such as those that come from the lens of user-centred design (UCD) practices boil down to transferring empathetic requirements into deliverables.

In many usability focused software projects, such as websites, games or applications, many functional requirements are an attempt at mapping user-centred requirements (UCRs) into more concrete deliverables. As this field has both developed and become more prevalent since the creation of agile methodologies, attempts to change the shape of development teams, and the workloads within have been made, but come against some of the core principals of agile. This can result in a situation where the UI and UX members of a development team work completely independently, with their own methodologies. This creates a similar channel to what developers have to the client, where they must convey and potentially lose detail between the design and implementation of the interface.

While implementing a completely new framework of methodology would be too much of a risk for any company, developing a gradual shift in ideology of how we develop UCRs is achievable, especially with a suite of tools that caters towards it.

CONTEXT

SOCIAL CONTEXT

A HISTORY OF AGILE DEVELOPMENT

While iterative design principals have been around since the creation of software development itself, the tenants of agile were formalised in by representatives from many existing development methodologies (such as SCRUM and extreme programming) in the Agile Manifesto (Beck et al., 2001). Since then, these twelve principles have been a cornerstone to the development of new practices, such as continuous integration, test-driven development, and has spread into other tech fields such as game design (Keith 2010).

Parallel to this, User-Centred Design was developing as an offshoot of Human-Centred Design, with a stronger focus on software use (Norman 1986). While these two field do intersect, there has been challenges in merging them completely, primarily due to Agile's focus on continuous delivering above all.

Supervisor: Julia Prior
Author: Ben Kelly

BASELINE ASSUMPTIONS

As rates of computer literacy have improved over time, certain baseline assumptions about how software should work for users have been established. Standardised design patterns such as Google's Material Design (Google 2014) exist, but mainly serve to map these concepts onto a set of rules for developers to follow, rather than a complete understanding of 'why' these patterns are commonplace.

ECONOMIC CONTEXT

GROWTH OF START-UP CULTURE

Start-ups and small companies are the primary users of agile methodologies. Often with a start-up product or service, requirements change much faster than in more established fields, and rapid prototyping is essential for gaining interest. Many start-ups also have an initial user focused premise, as it's much easier to bootstrap through crowdfunding and user interest than by directly approaching an investor. Because of this, start-ups would be the target of a user-centred requirements-based development model.

SOFTWARE AS A SERVICE (SAAS)

Many software tools now are not sold as licenced one-off purchases, but instead act as a continuous service, often with a subscription model. Regular patch updates are far more common now than previously, where agile was primarily used pre-release. This allows for consistent minute changes to the software, and continuous experience and feature improvements, given it doesn't break the user's workflow.

TECHNICAL CONTEXT

EXISTING TOOLS

Many tools have been developed in the aid of software development, and shape how a team will often work. A few that primarily affect how developers handle feature development are:

- Version control, such as Git, is used to keep many developers' changes tracked and consistent, so there is a definitive log of work, and nobody's code gets overwritten.
- Automated testing frameworks are applied to code to test functional requirements, simulating user input to ensure the desired functionality. This makes making and deploying changes a faster and safer process. Recently, test-driven development has become popular, allowing for the acceptance criteria to be written into the tests before the feature is implemented.
- Issue tracking software, like Jira, Zenkit, and Trello, mimic the traditional use of a Kanban, and often have options for applying sprints and other Scrum options. These are the biggest influencer for a company's agile workflow.

LITERATURE REVIEW

In the literature review of user-centred agile software development (UCASD) by Manuel Brhel et al. (2015), there have been plenty of previous research into UCASD, the majority of the papers have been from a theoretical/ conceptual level, and many were excluded from their review due to a lack of aggregatable codes of process. From this, the remaining were synthesised into a branching code, with the high-level being process, practices, people/social, and technology. It was noted that the people/social dimension is hampered by the single function teams (i.e. only UX designers on one team, developers on the other), and discussed a cross-functional alternative if there was an allowance for a process matchup or technological bridge.

One such approach to technology is supporting UCD through IDE plugins, to bridge the gap of the "lack of standard ways to involve users in agile teamwork" (Humayoun et al. 2017). Other technological approaches, such as implementation into issue tracking or version management tools have also been proposed.

Salah, D. et al.'s review (2014) also focuses on the de-prioritisation of non-functional requirements (referred to as UCD activities), however has more of a focus on optimising the work-dynamics of developers and UCD practitioners as two separate entities, as opposed to the hybrid practices mentioned in the review above.

These reviews showed me that the limiting factor of these methods often is the communication and tools ready. This makes sense as many of the agile practises align with continuous research and evaluation, two of the core feedback channels for UCD. The only aspect of practices this review doesn't have an answer for is "the prioritization of non-functional usability requirements in comparison to functional requirements".

CHALLENGES, UNCERTAINTIES & RISKS

SOCIAL CHALLENGES

COMMUNICATION AND DOCUMENTATION

The core to this problem is that not everyone sees the world the same way as the developer(s); Not the client, product owner, or users. For any software or documentation to be understood, the audience not only needs an understanding of intent, but also "requires a commonality of experience" described in the Inuendo Studio's video on semiotics in experience design. (Danskin 2016, 8:50).

This concept applies on two fronts, communication within the development team, and visualising how the end user will experience your software. There is a whole field of study and industry for user experience but is limited in how these ideas can be communicated through documentation. Jeff Patton sees documentation fundamentally limited in this sense, "Good documents are like vacation photos", in that someone else would see them and say "that's cute", but you see them and remember the experience. (Patton, J., 2014). This asks the question of whether our current systems of documentation are enough, or if there is something else developers need to use to develop for experience?

ETHICAL ISSUES

MANIPULATION OF THE USER

Much of the software users interact with day to day has been meticulously designed not just to be useable, but also to keep the user hooked. While the functionality is what gets them signed up in the first place, "the values and goals of the designers are implicitly encoded in the interface and the documentation but can conflict with the values of the user. This is when both the intentional and unintentional manipulation with the user starts" (Brejcha 2014). The results of this kind of design can be creating a system that compels the user to perform an action without their cognisant knowledge. B. F. Skinners theory of operant conditioning (McLeod 2007) shows how an interaction can be reinforced through repetition and reward.

This problem space revolves around better developing quantitative workflows for user experience, so ensuring the metrics used don't conflict with the user's actual needs will be essential in creating a process that benefits all, and not just increasing user retention through tricks.

TECHNICAL CHALLENGES

TECHNICAL DEBT

Technical debt is the principal that if non-perfect code for a feature is ever delivered, due to time constraints or a skill gap on the part of the developer, the development team is essentially 'buying problems' from future user stories, accruing debt that extends the development of future . If not dealt with correctly, "mess builds, the productivity of the team continues to decrease, asymptotically approaching zero" (Martin 2009).

As with other NFRs, the integrity, readability, and modularity of the code itself is often ill handled. A factor in this is the lack of documentation or visibility with the minutia of code. Features have user-stories, architecture has documentation, and defects are logged, but the it's rare to see more than a code comment for tracking technical debt (Kruchten et al. 2012).

User stories are often built so that functionality will always be added to the code-base, and any issues with the feature will be refactored as you go. This comes with a few assumptions that cause the build-up of technical debt, that the estimates of user story deliver factors in the fixing of this debt, and that fixing said debt will ever be a high enough priority to fix until it's too late.

Issues with technical debt are far more prevalent in UCRs, as unlike direct features, they far more commonly require tweaks and alterations to the existing functionality and code-base. While frontend can be developed modularly, so much of UX design relies on the cohesiveness of the experience, meaning even small changes have much larger impacts if not developed with expansion in mind.

SOFTWARE DRIVING WORKFLOW

Issue tracking tools like Jira, Microsoft Teams, and Trello are attempts to map the described workflow for agile, specifically Kanban and Scrum methodologies. Because of this, how they work is seen as the industry standard. As with all software, they have limitations to their features, and as such can't account for every possible workflow and team structure. While all these tools have their differences, they all share common ideas that mirror the core principals of agile, yet rarely provide options for the more fringe processes.

This has led to a feedback loop where a developments team of what agile is supposed to look like is influenced by the software they are using, which drives the updates of those tools. Because of this, without a significant alteration to setup, any new practices will have a hard time getting a foothold with these tools, and potentially entirely new platforms might need to be created.

PROBLEM ANALYSIS SUMMARY

Software development has expanded its focus into UCD, with software developers in the user-facing development fields using agile principles to solve a problem they aren't written for. Something else must be done to accurately map NFRs and UCRs for the development workload to sufficiently create the ideal hybrid UCASD described in previous studies. This project aims to explore the effectiveness of existing solutions and provide alternatives that fit within the challenges of developing accurate semiotics for experience documentation, ethical and sustainable design practices, and development tool limitations.

PROJECT SCOPE

RESEARCH AND INTERVIEWS

The main drive of this project will be collecting and analysing various implementations of agile methodologies in user facing projects. The participants will be sourced from UTS studio projects, such as the Software Development Studio, Games Studio, and the Innovation Hub. This will be done via interviews with the developers, as well as a summary of their project's user stories. The focuses will be on the tools, documentation, and language used to capture user stories and manage workload.

PROTOTYPING AND FEEDBACK

Parallel to this research, I will be prototyping an issue tracking tool with features and workflow designed reflect the ideals of a hybrid user-centred agile software development. This will be done through plugins and modifications to pre-existing open source tools to cut down on development time. A follow-up survey will be given to the participants of the original interviews giving feedback on this prototype.

OUTCOMES

The result of this project will be a report of the findings, summarising them into a list of fundamental tenants discovered. Alongside this will be the codebase and documentation of the prototype with explanation of how the workflow aims to fulfil those tenants.

LIMITATIONS & EXCLUSIONS

- The focus will not be on the work of these projects, only on the methodologies.
- The project is not designed to be fully operational, but a representative proof of concept.
- No development covered by a non-disclosure agreement will be directly discussed.
- Participants are not expected to use the prototype outside of a trial.

METHODOLOGY

The project centres on qualitative interview data driving the requirements for a user centred design process. The research will be broken into three phases, initial interviews and requirements gathering, iterative prototype development, and follow up trials for the project's conclusion.

INTERVIEW PROCESS

A request is planned to be sent out to relevant UTS studio coordinators, who will recommend certain projects fitting the criteria of a user centred or iterative design. Information sheets will be sent out to the team leads of these projects, with willing participants signing a consent form before interviews take place.

The interview will follow a series of pre-written questions, the primary focus of which will be exploring how the team manages UCRs and iterative development with their current implementation of agile methodologies, such as:

- How the team would go about redesigning the UI/UX of a feature, including the documentation
- What the team uses for acceptance criteria for a user story with user centred requirements
- The limitations of their current issue tracker with how they would like to document their work
- How the team deals with technical debt or other internal NFRs. Is this different from how they would like to?

INTERPRETATION OF REQUIREMENTS

Based off these responses, requirements will be processed by grouping similar responses into categorical limitations and needs. This data will be cross referenced against other reports of other hybrid methodologies and synthesised into a workflow methodology. The design and backlog of the prototype issue tracker will be created to fulfil the workflows requirements other issue tracking software cannot achieve.

DEVELOPMENT AND TRIAL PROCESS

After a backlog of tasks is created, decisions will be made about the implementation of the prototype, such as if it should be a plugin/ modification of an existing issue tracking software, a workflow guide with a paper prototype, or a completely new project. The project will be continuously scoped to fit the development criteria and timeline, with a focus on proof of concept over completed features. The project itself will be utilising the methodologies gathered from the initial interviews, with the backlog of user stories will be written in an iterative approach. The conclusions will summarise the development process as a first-hand insight of these approaches.

Trials of the software will be given to the willing projects, along with a survey asking if/how this tool would alter their agile workflow, as well as what aspects of the tool should be prioritised or changed. A screen recording will be taken during the usage of the tool and broken down into usage metrics as a form of feedback.

VALIDITY OF CONCLUSIONS

Most of the feedback for the prototype will be qualitative, by both collecting a post-trial survey and recording the usage patterns of the prototype. The validity of the research goals will be determined by the improvement in workflow between the initial interviews and the follow up. The trial size is limited, and as such the results would not be definitive, but an indication of the effectiveness of the prototype and its underlying user centred agile methodologies.

PROJECT TIMELINE

Note: I am planning on postponing Engineering Capstone (41030) until the spring session of 2020, due to my 2nd work placement. Because of this, there is be a six-month break in the milestones and tasks, although I will be searching for prospective projects to interview in the time.

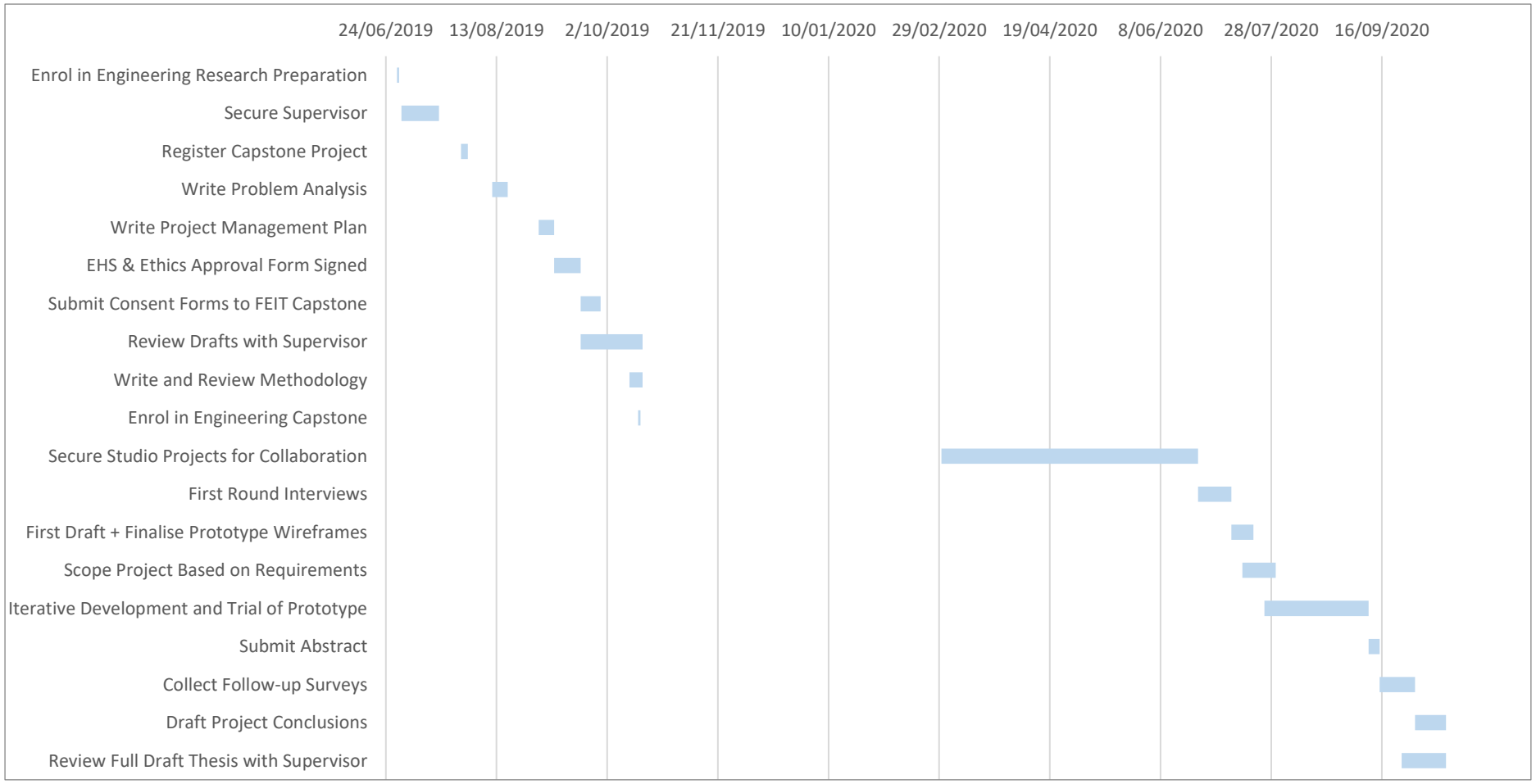
MILESTONES

ID	Milestone	Planned Finish	Related Tasks
M1	Capstone Topic Confirmed	3th July 2019	T1-T3
M2	Project Approved by FEIT Capstone	29th September 2019	T6-T7
M3	Finalised Research Proposal Submitted	20th October 2019	T4-T5, T8-T9
M4	Requirements Gathering Completed	Late June 2020	T10-T12
M5	Finish Prototype Development	Early September 2020	T13-T15
M6	Present for Capstone Showcase	Mid October 2020	T16
M7	Submit Final Thesis	Late October 2020	T17-T19

TASKS

ID	Description	Planned Start	Planned Finish
T1	Enrol in Engineering Research Preparation	30th June 2019	30th June 2019
T2	Secure Supervisor	1st July 2019	18th July 2019
T3	Register Capstone Project	31st July 2019	31st July 2019
T4	Write Problem Analysis	11th August 2019	18th August 2019
T5	Write Project Management Plan	1th September 2019	8th September 2019
T6	EHS & Ethics Approval Form Signed	8th September 2019	20th September 2019
T7	Submit Consent Forms to FEIT Capstone	20th September 2019	29th September 2019
T8	Review Drafts with Supervisor	20th September 2019	20th October 2019
T9	Write and Review Methodology	12th October 2019	18th October 2019
T10	Enrol in Engineering Capstone	16th October 2019	16th October 2019
T11	Secure Studio Projects for Collaboration	Early March 2020	Late June 2020
T12	First Round Interviews	Early July 2020	Mid July 2020
T13	First Draft + Finalise Prototype Wireframes	Early July 2020	Mid July 2020
T14	Scope Project Based on Requirements	Mid July 2020	Late July 2020
T15	Iterative Development and Trial of Prototype	Mid July 2020	Early September 2020
T16	Submit Abstract	Mid September 2020	Mid September 2020
T17	Collect Follow-up Surveys	Mid September 2020	Early October 2020
T18	Draft Project Conclusions	Early October 2020	Mid October 2020
T19	Review Full Draft Thesis with Supervisor	Late September 2020	Mid October 2020

GANNT CHART


24/06/2019
13/08/2019
2/10/2019
21/11/2019
10/01/2020
29/02/2020
19/04/2020
8/06/2020
28/07/2020
16/09/2020
Enrol in Engineering Research Preparation
Secure Supervisor
Register Capstone Project
Write Problem Analysis
Write Project Management Plan
EHS & Ethics Approval Form Signed
Submit Consent Forms to FEIT Capstone
Review Drafts with Supervisor
Write and Review Methodology
Enrol in Engineering Capstone
Secure Studio Projects for Collaboration
First Round Interviews
First Draft + Finalise Prototype Wireframes
Scope Project Based on Requirements
Iterative Development and Trial of Prototype
Submit Abstract
Collect Follow-up Surveys
Draft Project Conclusions
Review Full Draft Thesis with Supervisor

PROJECT UNCERTAINTIES

RISK MATRIX

The risk matrix scored the risk factor by multiplying the severity by the likelihood.

	Negligible	Minor	Moderate	Significant	Severe
Very Likely	5	10	15	20	25
Likely	4	8	12	16	20
Possible	3	6	9	12	15
Unlikely	2	4	6	8	10
Very Unlikely	1	2	3	4	5

RISK CLASSIFICATION

ID	Description	Mitigation/ Treatment	Likelihood	Severity	Risk
R1	Limitations with issue tracking tools restricting development	Prior research into the most modifiable tools.	Likely	Moderate	12
R2	Not enough projects sourced for accurate interviews	Allow plenty of time to find projects. Source from open development studios that would be more willing	Unlikely	Severe	10
R3	Scope Creep with prototype, from unachievable interview requests	Prioritisation of the tenants into functional/ non-functional	Possible	Minor	6
R4	Delaying partnered projects with prototype problems/ stunted methodology	Ensuring the testing is within a limited window and won't affect the overall performance	V/ Unlikely	Moderate	3
R5	Loss of research and code via data corruption or deletion.	Backing up the software to a repository and the documentation to a cloud hosted platform (i.e. google drive)	Unlikely	Significant	8

R6	Not having accurate testing data if the prototype is misused	Descriptive and easy to follow documentation alongside the prototype, as well as a comparison to their existing method for reference	Unlikely	Minor	4
R7	Project Delays due to rescheduling of interviews or troubles with development.	Allow a window for interviews, and backups in case any fall through.	Possible	Minor	6
R8	Participant unwillingness to share project information	Allowances to sign an NDA, and base level limitations on what we report.	Unlikely	Negligible	2
R9	Misconstruing research results and coming up with inaccurate conclusions or an invalid thesis	Round 2 prototyping testing will reflect the outcome of the initial results. Cross referencing answers against existing research will solidify an accurate stance	V/ Unlikely	Significant	4

COMMUNICATION PLAN

Subject	Channel	Purpose	Ideal Time(s)
Julia Prior (Supervisor)	In Person/ Email	Review of my drafts. Help with ideation. Potential access to SDS projects for interviews.	We meet weekly through the Software Development Studio Wednesday.
Project Managers	In Person	Research Interviews. Trial of prototype.	Once during the initial interview phase, and then again during the follow up survey.
Scott Mckeon (FEIT Innovation Hub), Jaime Garcia (Game Studio) UTS Start-ups	Email	Management of UTS project spaces, may help me find connections to projects willing to participate.	Will contact at the beginning of the autumn semester 2020, as that is often when new projects start.

PROGRESS STATEMENT

As of the submission of this proposal (20th of October 2019), milestones 1 and 3 have been achieved with all tasks complete. All forms for M2 have been sent out and are awaiting approval from FEIT capstone. Discussions for interviewing relevant projects have been made for the software development studio, but will be postponed until the commencement of the research next year.

REFERENCES

Farid, W.M., 2012, December. The Normap methodology: Lightweight engineering of non-functional requirements for agile processes. In 2012 19th Asia-Pacific Software Engineering Conference (Vol. 1, pp. 322-325). IEEE.

Patton, J., 2014. User story mapping: discover the whole story, build the right product. " O'Reilly Media, Inc.".

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R. and Kern, J., 2001. Manifesto for agile software development.

Norman, D. A., 1986. User-Centered System Design: New Perspectives on Human-Computer Interaction.

Keith, C., 2010. Agile Game Development with Scrum. Pearson Education.

Google, 2014, Material Design, viewed 18/08/2019 <https://material.io/design/>

Brhel, M., Meth, H., Maedche, A. and Werder, K., 2015. Exploring principles of user-centered agile software development: A literature review. Information and software technology, 61, pp.163-181.

Humayoun, S.R., Catarci, T., Kimani, S. and Dubinsky, Y., 2017. Managing User-Centered Design in Agile Projects.

Salah, D., Paige, R.F. and Cairns, P., 2014, May. A systematic literature review for agile development processes and user centred design integration. In Proceedings of the 18th international conference on evaluation and assessment in software engineering (p. 5). ACM.

Martin, R.C., 2009. Clean code: a handbook of agile software craftsmanship. Pearson Education.

Kruchten, P., Nord, R.L. and Ozkaya, I., 2012. Technical debt: From metaphor to theory and practice. Ieee software, 29(6), pp.18-21.

McLeod, S. A., 2007. Skinner - Operant Conditioning. <http://www.simplypsychology.org/operant-conditioning.html>

Danskin, I., 2016, The Artist is Absent: Davey Wreden and The Beginner's Guide, Inuendo Studios, viewed 18/08/2019 <https://youtu.be/4N6y6LEwsKc?t=530>

Brejcha, J., 2014, Ideologies in HCI: A Semiotic Perspective. In International Conference of Design, User Experience, and Usability (pp. 45-54). Springer, Cham.