UI Programming in Scrum

Lakshmi Sridhar Movva
Certified Scrum Master and Industry Consultant in Information Technology.

Satya Prasad Ravi, PhD
Associate Professor, Dept.of Computer Science & Engg., Acharya Nagarjuna University, Guntur, Andhra Pradesh, India

B.Reddaiah
Assistant Professor, Dept of computer Applications, Yogi Vemana University, Kadapa, Andhra Pradesh, India

ABSTRACT
Agile development methods are key to the future of flexible software systems. Scrum is one of the vanguards of the new way to manage software development when business conditions are changing. However; the tricky part in agile software development is that there is no manual which tells you exactly how to do it. Scrum teams have to experiment and continuously adapt the process until it suits the specific situation and to overcome the challenges. The aim of this research paper is to address the quality challenges and issues in Scrum implementation and proposing solutions to overcome or minimize the issues. The common issues in Scrum implementation are Scope Creep, Requirement changes which is inherent, the biggest challenge of inadequate time to prepare test plans, minimal requirements documentation to prepare the test cases, highly compressed test execution cycles, minimal time for regression. In addition [10] communication and building trust is another issue if the team is working in a distributed environment. These factors may result in having an adverse impact on the quality of the product delivered. To overcome this to an extent the concept of the "principle of factor sparsity" or [13] pareto principle can be applied which states that, for many events, roughly 80% of the effects come from 20% of the causes. This was originally described by Vilfredo Pareto and later formalized by Joseph Juran.The principle is just a rule of thumb, but an important one. Whether the percentages are really 80/20 or 70/30 or 90/10, the reality is that most things are caused by a few underlying factors. The same applies for Scrum and 80% of the quality issues would a raise from the tasks that are performed related to 20% of the stories .So; [12]concentration on the vital few stories or tasks to arrest the critical issues and having a technique to double check the code quality delivered by agile scrum teams would provide better results. UI programming is one such new technique proposed. For an increase in quality and decrease in the defects generation; the call for a UI programming can be taken by the product owner while creation of product backlog or can be taken by the team along with product owner during the sprint planning meeting when actually the tasks are defined. The tasks that are candidates for UI programming would base on the story/tasks criticality, complexity, likeliness of being source for defects bugs, the need for high quality. At times the story and all its tasks defined can be candidates for UI programming or specific tasks within the story can be candidates for UI programming. In nutshell, the UI programming can be used in any projects which are implementing scrum and are having considerable quality issues in the implementations.

Keywords
Agile methodologies, Scrum, Quality, Defects, UI programming, You and I programming, UI pair, Pair programming, UI pair pace,UI task,UI switching,UI shuffling, Distributed environment

1. INTRODUCTION
Software organizations constantly need to react to market dynamics, new customer requirements and technological innovations (Beck 2000; Lycett et al. 2003) [7]. The degree of market dynamics and needs has increased over the past decades creating a number of fast moving software organizations (Börjesson and Mathiassen 2004). To cater to these dynamics agile methods have gained popularity in the recent years.The continuous experiments and surveys on agile methods promise faster development thus improving the quality of the working software delivered by the agile teams and increase of satisfaction within the teams, customers and business units . [6]Many organizations regard agile methods as a way of addressing key problems in software development; namely, the software takes too long to develop, costs too much and have quality issues upon delivery (Holström et al. 2006). [10]If the Team can't produce Quality Code, then it's going to have a tough time being effective. And by Quality Code, it means code that is extensible, modifiable, maintainable, and can be quickly integrated and verified with extensive unit, functional, and performance regression tests. This paper gives us an additional feature UI programming which would be helpful to enforce the quality of the software delivered by the agile scrum teams. With an understanding of how pair programming in XP is used, the UIprogramming technique is developed and scrum teams can use this technique with ease.However,neither every team member in the scrum team, nor all the stories/sprint tasks would require a UI programming to be done all the time. This should be judiciously exercised as the task defined as a UI task would require nearly double the effort to perform than normal. The UI programming and further details are depicted as in the sections that follows.

2. UI PROGRAMMING
'You' and 'I' programming, in short UI programming is an agile software development technique of having two professionals working together on a task on a single workstation. One of them would be looking on, offering suggestions, validate the correctness of the task as it is done, while the other actually executes the task.

2.1 UI Programming Roles
There are two **UI roles** that are played in UI programming

2.2 UI Handler
Programmer who is actually executing the task/story .UI Handler can focus all of his or her attention on the tactical

aspects of completing the current task, using the observer as a safety net and guide.

2.3 UI Observer

is the one who reviews the task being done. The UI observer considers the strategic direction of the work, coming up with ideas for improvements and likely future problems to address. The UI Observer would be looking over shoulder when the UI Handler does the task

2.4 UI Pairing

The pairing up of UI Observer and UI Handler is UI Pairing and the task they do is UI Programming. Of the two in the pair one should be more experienced than the other such that there is always a scope for sharing of knowledge to the less experienced in pair. If the UI Observer is a senior member of the two, he has a scope to offer suggestions and best practices enabling the junior to with additional knowledge and if UI observer is a junior member he can watch the senior performing the task and learn from it. However the two roles played by team members carry equal signicificance despite one being more experienced.

2.5 UI Task

Not all the tasks in the product backlog can be candidates for UI programming. Any task which is critical, very complex should have quality adherence at any cost and which needs to be with zero defects or bugs even if the effort put on the task is high, can be treated as a candidate for UI task.

During defining the product backlog the product owner can define any task as an UI task or during the Sprint planning meeting the sprint team can discuss and define the tasks which need to be classified as an UI task.

Also, tasks like production code deployment which requires huge checklist of activities to be done can be candidates for UI tasks. For e.g. the deployment activities in data warehousing projects would demand high concentration and accuracy as the code deployment is done into production and are always candidates for inducing bugs, defects if the deployment is not done properly. So its better that such tasks is classified as UI tasks.

2.6 UI Walk

Once the UI task is done from the list of UI task or after considerable time spent to work on UI task, the UI pair should walk away from the workstation to a common location where they can have some very informal talk on the things done so far. The location can be a coffee area, a seating area or a lobby where the UI pair can have a chat for very short duration. This would help in the pair grooming if the pair is formed newly and enable the pair to discuss and think better on the progress so far. The ideas that pop up during that short break can be discussed and implemented once back at the workstation.

2.7 UI Switching

After considerable amount of time spent as a UI Handler or UI Observer, the UI pair should switch their roles.

2.8 UI Shuffling:

The UI pairing should be dismantled frequently and new set of UI pairs should be formed to ensure that the knowledge is shared across the team and also ensures that the same enthusiasm to pair up with a new member is prevailing across the team during the sprint

2.9 UI Programming in Distributed Environment

Provided with additional software tools and hardware, the UI Programming might suit to an extent for the distributed scrum teams as well if proper infrastructure and communication is established.

[10]A web conference with screen sharing and webcam facilities can be used to mimic the UI programming even in distributed environment effectively. The screen can be shared between users and the normal flow of UI programming, UI Pairing, UI Role playing,UI switching,UI shuffling etc can be performed with little or no hindrance.

2.10 UI Requirements

UI programming requires very large Monitors to respect personal space of the professionals and to effectively work. A couple of such UI desk arrangements can be made available for the team such that they can work in pairs whenever required

3. UI PROGRAMMING TERMS:

3.1 Terms Explained.

3.1.1 Story Type

UI:'You and I' programming enabled Sprint

N: Normal sprint

3.1.2 Complexity:

Complexity of the story

H –High

M-Medium,

L-Low

3.1.3 Effort allotted:

Planned effort estimate for the user story

3.1.4 Effort consumed:

Actual Effort consumed for execution of user story

3.1.5 No of Bugs Produced:

No of bugs, defects produced. Can be code issues, user interface issues, data quality issues etc

3.1.6 Effort to fix Bugs:

Effort consumed to fix the bugs/defects.

3.1.7 Effort (including bug fixing):

Total effort to complete the programming for user story including bug fixing.

3.1.8 Tot:

Total

3.1.9 Total spent for a sprint of effort estimate 100:

Actual effort spent (including bug fixing) for a sprint with planned effort estimate of 100 units. This is used to compare the sprints as having a sprint planning with exactly same effort estimates is not possible.

3.1.10 Effort saved:

Effort saved by using UI programming

3.1.11 Pace of UI pair:

The pace of pair is the pace at which the professional in UI pair executes 1 unit of work. In normal scenario the pace would be 1 for 1 unit of work.However; adopting the UI programming would bring down the pace to lower than 1 as two people would be working on the same computer and on the same task. Here Pace of pair 0.72 means each one in pair is accomplishing 72% of the task only

3.1.12 Defects Dropped by %:

Defects dropped by the usage of UI programming in specific stories/tasks

3.1.13 Base lining Pace of UI pair:

The pace of pair can be base lined upon observations on the UI programming executions in atleast two sprints.

4. ADOPTION

This technique can be used with other techniques described in the topic 'Distributed Agile Development: Practices for building trust in team through Effective communication'. To understand and analyze the impact of the UI programming a scrum team is selected and two pairs of identical sprints in terms of effort, technical complexity, functional complexity are chosen .It was ensured that the changes in the environmental factors such team expertise, working environment, team structure, technologies etc are kept to minimal such that it doesn't have any adverse impact of the accuracy of the UI programming being adopted. The total four sprints in scope are chosen in the following way.

At the start of a new Sprint –Mercury (Normal Sprint)-The complexity, type of stories, tasks were noted. The Planned, execution effort, Defects raised are noted leaving a scope for getting in the details for the effort to fix those defects.

The next sprint-Venus(with UI programming) is planned which is similar to first one and with almost identical stories in terms of effort,complexity.This has the stories identified for the UI programming by the product owner considering various factors. The details as in sprint Mercury are noted for the sprint Venus also. Similarly Sprint Earth is a normal sprint planned and the Sprint Mars is planned as sprint with some UI stories.Sprints Mercury and Venus would form a pair and sprints Earth, Mars which are identical in nature are a pair to get an insight of normal vs. UI working.

Table -1 and Table-2 below gives the insight of the UI working

Table1 –Sprint pair -1

Story Type	Sprint Name	Complexity	Effort Allotted	Effort Consumed	No of bugs/Defect produced	Effort to fix	Total Effort(including Bug fixing)
N	Mercury	H	7	6	2	4	10
N	Mercury	H	23	22	5	11	33
N	Mercury	H	20	18	3	16	34
N	Mercury	L	8	9	0	0	9
N	Mercury	L	7	7	1	2	9
N	Mercury	L	8	10	4	2	12
N	Mercury	M	10	10	1	3	13
N	Mercury	M	8	6	1	3	9
N	Mercury	M	9	10	3	2	12
Tot			**100**	**98**	**20**	**43**	**141**
UI	Venus	H	20	31	1	2	33
UI	Venus	H	14	20	1	2	22

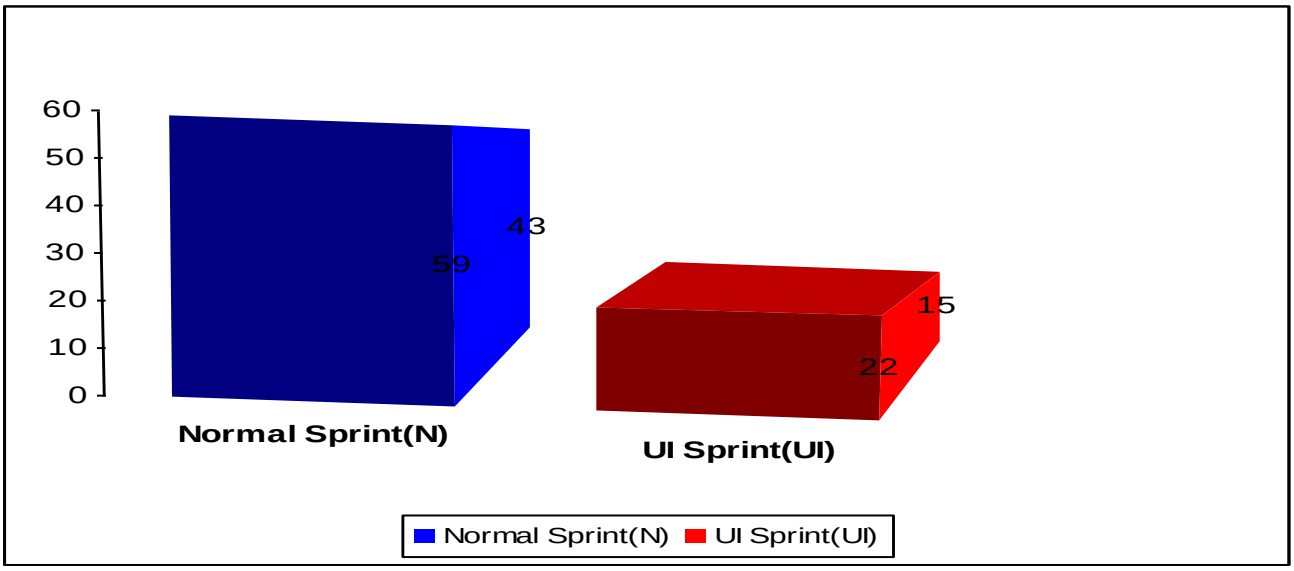
Story Type	Sprint Name	Complexity	Effort Alloted	Effort Consumed	No of bugs/Defect produced	Effort to fix	Total Effort(including Bug fixing)
N	Earth	H	17	18	4	9	27
N	Earth	H	26	22	5	15	37
N	Earth	L	22	24	1	6	30
N	Earth	L	18	16	3	4	20
N	Earth	L	6	8	2	4	12
N	Earth	L	11	16	4	4	20
N	Earth	M	12	14	2	6	20
N	Earth	M	17	17	2	5	22
N	Earth	M	19	16	4	6	22
Tot			**148**	**151**	**27**	**59**	**210**
UI	Mars	H	15	23	1	2	25
UI	Mars	H	29	44	1	3	47

UI	Venus	H	16	22	2	2	24
N	Venus	L	12	12	1	3	15
N	Venus	L	8	8	2	4	12
N	Venus	L	12	12	0	0	12
N	Venus	L	6	6	0	0	6
N	Venus	L	7	7	1	1	8
N	Venus	M	12	12	1	1	13
N	Venus	M	12	12	0	0	12
Tot			**119**	**142**	**9**	**15**	**157**

Total spent for a sprint of effort estimate 100	
Normal	144
UI	132
Effort saved	12
Pace of UI pair	1.46
Defects Dropped by %	55

N	Mars	L	6	6	1	2	8
N	Mars	L	19	22	1	2	24
N	Mars	L	11	11	1	2	13
N	Mars	L	19	19	2	4	23
N	Mars	L	12	12	1	3	15
N	Mars	L	6	6	1	1	7
N	Mars	M	6	6	2	2	8
N	Mars	M	6	6	1	1	7
Tot			**129**	**155**	**12**	**22**	**177**

Total spent for a sprint of effort estimate 100	
Normal	139
UI	137
Effort saved	2
Pace of UI pair	1.52
Defects Dropped by %	56



Graph 1: UI sprint vs. Normal Sprint

4.1 UI Programming Benefits

The UI programming would help doing built in, automatic, on the fly QA peer review and testing at a time. Below are the benefits observed during the implementation of this new feature.

1)It produces a better thought out code and a better control of the task performed

2)The Errors are caught earlier with the UI programming

3)The UI programming enables better knowledge sharing across the scrum team.

Figure 2 shows the knowledge accessibility for each one in the UI pair i.e UI observer can learn the coding skills, knowledge out of experience etc from UI Handler and vise versa. The scope for knowledge gain would be vast as the pairs get switched over time.

4)The UI programming enables the Code transferability i.e Two developers understanding the code

5)The fixing of the failing test cases would be done well in advance i.e Defects produced are very less. Fig 1 shows the defects produced by UI pair reduces drastically when compared to the defects produced by programmers individually

6)Having two heads are better than one and would help catching the mistakes early

7)UI programming helps to give more quality code, test ready code as an output

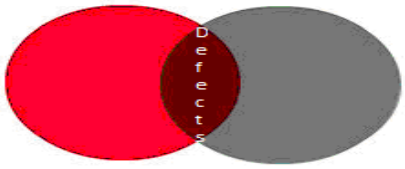


Figure-1 : Defects in UIprogramming

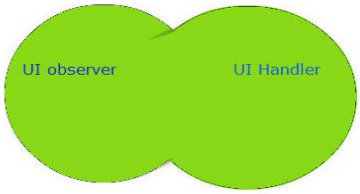


Fig2:UI Sprint-Knowledge sharing

5. CONCLUSION:

The study of the current paper starts with findings on the issues which are common in agile methodologies especially scrum implementation. Within this paper the quality of the product delivered by the scrum teams is the main area of problem identified for research and analysis, and a new technique called 'You and I programming ' or 'UI programming' in short has been proposed to minimize the producing of defects by the teams. The UI programming technique is implemented on two sprints and results are compared against the normal sprints of same genere. The results shows that the UI programming implemented in the scrum improvises the quality of the code delivered to a greater extent, though there is not much significant gain of the effort spent as compared to the normal sprint. The results shows that the defects dropped by almost 55%(Graph 1) .However; there is a steep increase in the factors like combined ownership of the code, learning and sharing of knowledge ,quality at first go with fewer defects,decrease in the effort of communication and co-ordination related to bug fixing and increased customer satisfaction through the implementation of the proposed UI programming techniques in scrum team.The UI programming can be used by the scrum teams which focuses on better quality of the product produced along with the above factors.

6. REFERENCES

[1] A practical Guide to Distributed Scrum: Elizabeth Woodward,Steffan Surdek and Matthew Ganis.

[2] Coaching Agile Teams-Lyssa Adkins

[3] Agile Product Management with Scrum –Roman Pichler.

[4] Experience Report: Distributed agile: project management in a global environment Seiyoung Lee & Hwan-Seung Yong

[5] Agile Alliance, Principles behind the Agile Manifesto, "http://agilemanifesto.org/principles.html."

[6] High smith J, Cockburn A. Agile software development: The business of innovation

[7] Lycett M, Macredie RD, Patel C, Paul RJ. Migrating agile methods to standardized development practice

[8] A paper by Dr. Satya Prasad Ravi, B. Reddaiah, Lakshmi Sridhar Movva-named "Distributed Agile Development: Practices for building trust in team through Effective communication "

[9] Dr. Satya Prasad Ravi, B. Reddaiah, Lakshmi Sridhar Movva " Framework to mitigate dynamic and static risks with respect to agile" ESTIJ, Vol.2, No.1, 2012: PP.63-68,2012.

[10] Dr. Satya Prasad Ravi, B. Reddaiah, Lakshmi Sridhar Movva, Rajasekhar Kilaparthi "A Critical review and empirical study on success of risk management activity with respect to scrum" ESTIJ, Vol.2, No.3, June 2012: PP.467-473, 2012.

[11] Software Engineering-A practioners guide –Roger S Pressman

AUTHOR'S PROFILE

Lakshmi Sridhar Movva is a Research Scholar in Acharya Nagarjuna University doing his Ph.d in Agile Methodologies under the esteemed guidance of Dr R.Satya Prasad..He has rich experience in IT industry and working as an Industry consultant. He is a certified Scrum Master and has been practicing Scrum in the projects he executed. He has completed M.Sc in computer Science from Acharya Nagarjuna University and M.Tech in Information Technology from Punjabi University.

Dr. R. Satya Prasad received Ph.D. degree in Computer Science in the faculty of Engineering in 2007 from Acharya Nagarjuna University, Andhra Pradesh. He received gold medal from Acharya Nagarjuna University for his out standing performance in Masters Degree. He is currently working as Associate Professor and H.O.D, in the Department of Computer Science & Engineering, Acharya Nagarjuna University. His current research is focused on Software Engineering. He has published several papers in National & International Journals

B. Reddaiah is a Research Scholar in Acharya Nagarjuna University doing his Ph.d under the esteemed guidance of Dr R.Satya Prasad .He is working as an Assistant Professor, Department of computer Applications, Yogi Vemana University, KADAPA, Andhra Pradesh. He has completed MCA from Sri Venkatesawra and ME from Satyabhama University.