Trust: Promoting Interdependence, Knowledge Sharing & Collaboration in Scrum Teams

Trish O'Connell
School of Science & Computing
Galway-Mayo Institute of Technology
Galway, Ireland

Owen Molloy
Dept. of Information Technology
National University of Ireland
Galway, Ireland

Abstract— *Developing software using Scrum is predominantly a team-based activity. Consequently it is an intensely social endeavour. The Agile Manifesto strongly advocates using teamwork to develop software. Scrum is the most widely adopted Agile software development method currently being used both worldwide and in Ireland.*
The success of the Scrum team follows from the effective interdependence of the team members. Interdependent teams trust each other implicitly. However, little research has been conducted into the development and fostering of trust in co-located Scrum teams. Using a Constructivist Grounded theory study involving one team from a large multinational software development organisation in the West of Ireland we investigate how trust is developed and fostered in the team.

Keywords
Agile; Scrum; Trust; Interdependence; Constructivist Grounded Theory;

I. Introduction

Agile has come to dominate the software development industry landscape. A VersionOne report [1] on the 12th Annual State of Agile in 2018 reported that 56% of those organizations surveyed cited Scrum as their adopted methodology for developing software. Scrum is essentially a framework for developing software that uses a minimally prescriptive approach to development, in contrast to the more plan-driven Waterfall model and its derivatives.

In keeping with the Agile Manifesto [2], Scrum is a people-centered, team-based approach to software development. The Scrum team is crucial to the development process given that they are tasked with evolving the design from inception to completion, culminating in the delivery, at the end of each Sprint, of a potentially shippable increment of software. Prior to the popularity of Agile the development method was almost always a top-down, plan-driven process. By contrast the Scrum team is self-organizing which means that as a group of individuals the team members unite behind a common purpose or goal and, more importantly hold themselves accountable for progress towards the common goal. Being self-organizing, in a Scrum context, means setting and agreeing project deliverables to be achieved during the timeboxed development period known in Scrum as a Sprint. Sprints tend to last from two to four weeks depending on the organisation. Having a common goal is often cited as being important to achieving a successful Sprint, as the team members 'buy-in' to the development tasks.

Moe, Dingsøyr and Dybå [3] argue that "software development depends significantly on team performance." As with any successful team it is necessary to share knowledge and collaborate to achieve the goals of the team. The purpose of this article is to introduce trust as the essential glue that results in successful interdependence thus enhancing knowledge sharing and collaboration.

Whilst there has been some research on the topic of trust in distributed Agile teams,[4],[5],[6], to date little empirical research has been published on the development of trust in co-located teams. In this article a Constructivist Grounded theory study will be used to investigate the key building blocks of trust in a Scrum team.

Section II of this article presents the background to the study in terms of interdependence, knowledge-sharing and collaboration – all of which are adjudged to be crucial to a high performing Scrum team. Section III details the research that was undertaken. Sections IV and V present the data Analysis and results and finally Sections VI, VII and VIII present a discussion of the findings, the limitations of the study and conclusions.

II. BACKGROUND

Katzenbach and Smith define a team as "a small number of people with complementary skills who are committed to a common purpose, set of performance goals, and approach for which they hold themselves mutually accountable [7]."

According to the Scrum Guide [8] the recommended team size should be between three and nine. As the team works to achieve the goals of the Sprint it is evident that team members are working to a common purpose. The self-organizing nature of the Scrum team involves the team members embracing mutual accountability for the team's output at the end of the Sprint.

Morgan et al. by contrast refer to the nature of the interaction as the key definer of a team i.e. they refer to a team as "a distinguishable set of two or more individuals who interact *interdependently* and adaptively to achieve specified, shared, and valued objectives [9]."

In traditional software development organizations, most often using the Waterfall methodology to develop software, management has tended to be 'command and control' in that people are assigned to teams based on functionality. As Hoda describes "roles on traditional teams are based around functional tasks reflected by their organizational roles, such as programmers responsible for programming, testers responsible for testing, analysts responsible for requirements analysis, etc. [10]."

In a total departure from this approach the Scrum team is cross-functional and self-organizing. It is expected that team members bring their unique skills, experience and talents to help the team achieve its' objectives. Whilst Katzenbach and Smith cite the need for "problem-solving and decision-making skills, and interpersonal skills [7]," Faraj and Sproull refer to "technical expertise (knowledge about a specialized technical area), (2) design expertise (knowledge about software design principles and architecture), and (3) domain expertise (knowledge about the application domain area and client operations) [11]" as being of importance in helping the team to reach its' goals.

Possession of these complementary skills is important for the team members but these skills in isolation will not achieve the team's goals. It is imperative that team members are interdependent.

A. Interdependence

Interdependence has three specific connotations. The first is task interdependence. This may be described as "the degree to which group members must rely on one another to perform their tasks effectively [12]." This description concurs with Courtright et al. who postulate that a high degree of interdependence exists "when members simultaneously work on all aspects of the task, when they intentionally form groups with members possessing varied access to resources [13]." This type of interdependence underpins the entire Scrum philosophy. The team works together to move the development forwards. During the Daily Stand up team members are asked three specific questions viz. *What did you do yesterday? What will you do today? What is blocking your progress?* When team members admit an impediment to progress it is not uncommon for another team member to volunteer to help. Generally speaking some team members possess specific expertise and knowledge e.g. a developer may be knowledgeable about client-side or server-side and may volunteer to work with a more junior developer to help with task completion.

The second type of interdependence which is importance in a Scrum team is that of team interdependence. According to Colquitt, LePine and Wesson (2016) "team interdependence is the way in which the members of the team are linked to one another [14]." By its very nature Agile relies on strong interdependence between team members. In part, this is achieved by constant communication and joint problem-solving initiatives. Whilst regular face-to-face communication is strongly encouraged this is often enhanced by the Scrum team members sitting in close proximity to each other.

The third type of interdependence is that of outcome interdependence. This can be described as the degree to which the outcome "depends on the performance of others [15]." Outcome interdependence accounts for the very essence of the Sprint in which "everyone focuses on the work of the Sprint and the goals of the Scrum Team [8]."

As part of the interdependence in the team it is vitally important that the team members share knowledge. During Sprints the team must exchange task-related information, ideas on how to overcome obstacles to progress, feedback on testing/progress, etc.

B. Knowledge Sharing

There are two types of knowledge which are of significance in Scrum teams: explicit knowledge and tacit knowledge. Wyatt [16] describes how explicit knowledge deals with facts, relationships and rules that can be written down and shared without the need for explanation or discussion. In software development some activities, e.g. requirements gathering may be considered to be explicit.

Scrum encourages explicit knowledge sharing through a number of practices which Cervone [17] refers to as Scrum events. These include the Release Planning meeting, the Daily Stand-up and the Sprint Retrospective.

The first of these is the Release Planning meeting in which the project is discussed, often with the customer or major stakeholder, and the team agree, in broad detail, which features can be incorporated in the upcoming Sprints. The team agree an initial timeline and match features to iterations. The Scrum Product backlog is the list of all of the tasks that need to be done to realize the product. The team works in conjunction with the Product owner to prioritize items to be included in the Sprint. This exchange is a form of explicit knowledge sharing.

Following on from the Release Planning meeting and the development of the Product backlog, the next event at which explicit knowledge sharing occurs is the Daily Stand-up. This is generally a short meeting, held each working day, during which each team member gives a concise status update of their previous day's work. Again this is explicit knowledge sharing, since team members who have encountered barriers which prevent progress share their issues with their peers in the hope of getting feedback or advice, if required.

The Sprint Retrospective functions as the project postpartum. It is the meeting at which the team reflects on what it has achieved... what went well and what needs to be improved for the next Sprint. Without a frank and explicit exchange of views this would fail to generate improvements in future Sprints.

In addition to explicit knowledge, software development relies heavily on tacit knowledge. Tacit knowledge refers to the knowledge that someone possesses. It is implicit as opposed to explicit. It can be thought of as the experience and skill that a team member brings to the development process. Knowledge of design practices or systems knowledge, often garnered from prior experience, all count as tacit knowledge and a difficulty in the Scrum team can be in getting team members to share their knowledge with their colleagues. Part of the problem with tacit knowledge sharing can be summed up by Polanyi (1966) who affirmed "we can know more than we can tell [18]."

Regardless of the difficulties involved in sharing knowledge Levy and Hassan nevertheless contend that "software development work requires various forms of explicit as well as implicit knowledge [19]."
Knowledge sharing, however, whilst crucial to the success of a software development endeavour is not, in itself, sufficient. The Agile Manifesto [2] emphasizes "working together daily", i.e. collaborating, to move the development forwards.

C. Collaboration

In software development, collaboration refers to working jointly with cross-functional team members to develop solutions to customer requirements and using consensus to reach decisions. It may be seen as a combination of cooperation, coordination and communication [20] but it is more than these. Cooperation refers to working together for mutual benefit. Coordination relates to the management of people and/or the resources required to achieve a task. Communication is the sending and receiving of information. Collaboration, on the other hand, is more about synergy where the sum of all the contributions exceeds that of the team members acting as individual contributors. The importance of collaboration was highlighted by Nerur, Mahapatra and Mangalaraj who argued that "A cooperative social process characterized by communication and collaboration between a community of members who value and trust each other is critical for the success of agile methodologies [21]."

A team that collaborates successfully can organize and prioritize what needs to be done, when, and by whom. As issues arise, they can be discussed openly and resolution can be found by employing participatory approaches and consensus-based decision making. The team-focused, social-interaction element of Agile is key to achieving the goals of the Sprint because it is through this socialization that communication and ultimately collaboration can be enhanced.

In order to achieve a high degree of collaboration in Scrum we can again refer to Cervone's [17] Scrum events. The Daily Stand-up is the foremost venue for collaboration. With a view to helping each other, but more specifically, helping the team to achieve the Sprint goals, team members can discuss barriers to progress and even '*throw around*' potential solutions to these barriers.

In addition, the Sprint planning meeting is the ideal forum for identifying the common goals and objectives of the upcoming Sprint. By eliciting participation and buy-in on the common goals for the Scrum team, the team comes to internalize and identify with the shared vision for the upcoming Sprint.

Furthermore, the collaborative environment of the Scrum team is optimal for tacit knowledge sharing as working closely together facilitates knowledge transfer through daily interaction between team members. This collaborative style of team working moreover "offers opportunities for heightened creativity and enhanced quality [22]." With all of this emphasis on interdependence, knowledge sharing and collaboration for a Scrum team to be successful it is worth noting that many cite trust as being essential for these activities to happen. Mach, Dolan and Tzafrir state that "trust is an integral part of teamwork [23]." However, whilst many have cited trust as being essential to successful Agile teams, to date no-one has conducted research into what actually builds or leads to trust in co-located Scrum teams.

Consequently, in 2017, we commenced research into this domain. As trust is a predominantly sociological construct we decided to use Constructivist Grounded Theory to investigate how the trust is fostered and developed in a Scrum team in a large multinational technology company that provides software solutions to clients in the automotive industry.

III. THE RESEARCH

Grounded theory is an inductive (some might argue abductive), methodology that was commonly used in the social sciences domain to systematically gather and construct theory that is '*grounded'* in the collected data. What is today referred to as Classical Grounded Theory was introduced in the 1960s by Barney Glaser and Anselm Strauss [24]. Unlike previous qualitative research methods which sought to verify theory, grounded theory seeks to create theory, or at the very least explain a phenomenon, by continually returning to the data and engaging in constant comparison. Eventually, patterns emerge in the data which can be coded, and these codes then form categories which can lead to a theory, which ultimately seeks to explain or facilitate an understanding of how the social construct works.

Constructivist grounded theory is a branch of grounded theory that differs from classical grounded theory in that Glaser and Strauss believed that the nature of reality was objective so the researcher is merely present in an 'observer' capacity. Charmaz, by contrast, advocated that the researcher brings their own experiences, perceptions and bias to the table and consequently, together with the research subject, they co-create meaning from the data [25]. Thus, the researcher is an active participant in the process of constructing meaning from the data.

We decided to use a constructivist grounded theory approach for this research for the following reasons:

- No hypothesis has ever been formulated to account for trust in colocated Agile Scrum teams.
- Grounded theory is widely used in researching social constructs. Trust is a socially constructed phenomenon of great importance to interdependence, knowledge sharing and collaboration.
- Grounded theory seeks to offer an explanation about the research domain that is substantiated in the participants lived experience.
- Constructivist Grounded theory has been successful in previous software research studies.

In this study, which forms part of doctoral research on the topic of trust in Scrum teams, purposive sampling was used to contact software development companies that were known by the researcher to use Scrum as their development methodology. Purposive sampling can be described as a type of focused sampling and in this case an organisation that uses Scrum was approached and permission was sought to conduct the research. Having conducted a short pilot run to perfect the interview process, a single large multinational was initially selected. The company has a number of both distributed and onsite Scrum teams. Given the logistics of time zones and language barriers we were able to interview participants from one of the local onsite teams as shown in Table 1 below:

TABLE I TEAM COMPOSITION & ROLES

Participant #	Team A
P#1	Scrum Master
P#2	Product Owner
P#3	Developer
P#4	Developer
P#5	Developer
P#6	Developer

With the aim of gathering what Charmaz refers to as '*rich data'*[25] a series of in-depth interviews were conducted with all of the participants. Rich data refers to collecting data which fully addresses the complexities and depth of the topic under study. The interviews lasted between 30 and 50 minutes. Each interview was audio recorded and transcribed. In order to ensure that all of the nuances and subtleties were captured by the author, the transcribed interviews were subsequently returned to the participants for verification as shown in Figure 1.

Once background data and consent for audio recording from each participant was obtained , the face-to-face, one-on-one interviews focussed initially on knowledge sharing, interdependence and collaboration and then logically continued on to encompass how trust is established in the Scrum team. Following the interviews, the researcher, in accordance with the established constructivist grounded theory methodology, wrote memos detailing ideas and refinements that were based on what was said during the interviews with the intention that it would advance theoretical understanding of the trust construct. In this sense the interviews were unstructured but participants who were not naturally talkative on being given an opening were prompted in a semi-structured manner for their response to topics that had been of interest in a prior interview with a previous interviewee. In this way the knowledge base of questions of the researcher is enhanced and each interview becomes progressively more focused.

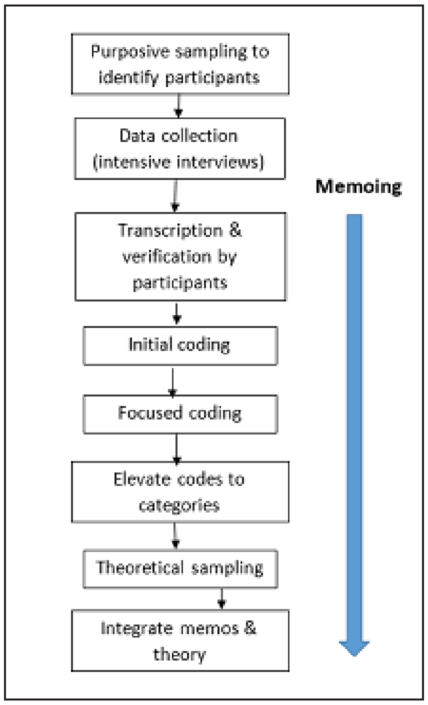
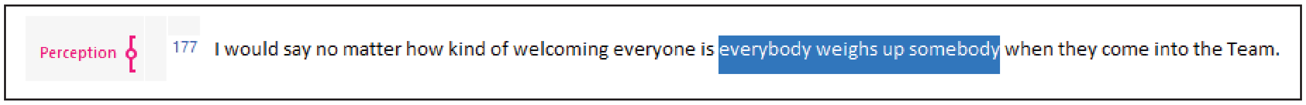

Purposive sampling to identify participants
Data collection (intensive interviews)
Transcription & verification by participants
Initial coding
Focused coding
Elevate codes to categories
Theoretical sampling
Integrate memos & theory
Memoing

Figure 1 The Constructivist Grounded Theory process



Perception 177 I would say no matter how kind of welcoming everyone is everybody weighs up somebody when they come into the Team.

Figure 2 Coding fragment from Interview with P#1

IV. DATA ANALYSIS

A key aspect of Constructivist Grounded Theory involves coding the interviews for meaning. Thus the transcribed interviews were uploaded into a qualitative analysis software package. MAXQDA was chosen for its intuitive easy to use interface. Transcribed interviews can be stored, analysed and coded in MAXQDA. Once participants' interviews were transcribed and validated as shown in Figure 1 the process of initial coding began.

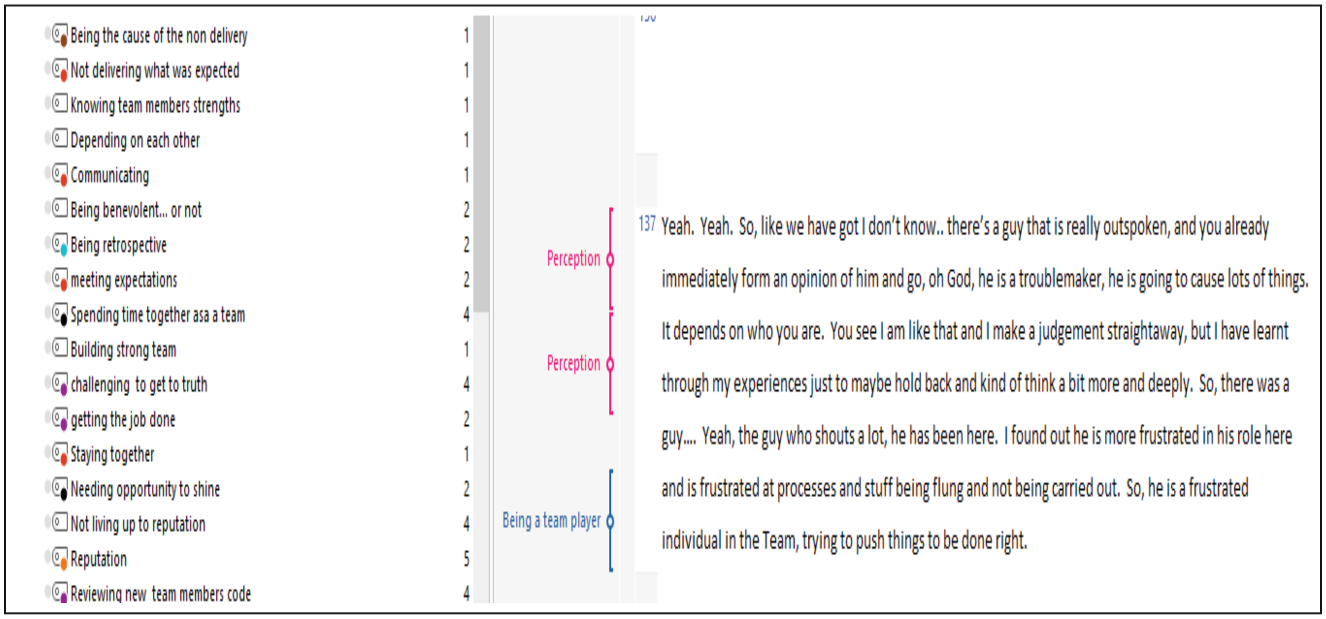
In this phase each line of the participants' transcription was analysed with a view to encapsulating the meaning in a code which essentially describes what the segment of text is about. Ideally the codes are gerunds which describe actions or labels which describe what happening e.g. "perception" [P#1] which is depicted in Figure 2. On the right hand side, highlighted, is the fragment of what was said by the participant. On the left, is the code, 'Perception' that was used to encapsulate what it was felt the participant meant.

Once the initial interview was coded, subsequent interviews were similarly analysed and compared to each other. Constant comparison is a key strategy used in grounded theory where each piece of elicited data is compared to other pieces of data by the researcher to identify and highlight similarities and differences in the participants' experiences.

MAXQDA was helpful in facilitating this process as codes assigned from earlier in the transcript were available to view in a portion of the window as shown at left in Figure 3.

The ability to view other codes helped with what Charmaz refers to as "focussed coding" [25] where codes are analysed to advance the theoretical direction of the study. Charmaz describes these codes as more conceptual than the initial coding since they attend to "how your initial codes account for the data [25]."

Finally, the focused codes were raised to conceptual categories which, when assimilated, addressed the initial question of how trust is nurtured and developed in Scrum teams. It is at this stage that the meaning is co-constructed with the participants. This happens due to the fact that the researcher is not just an observer. A researcher brings their own experience, and perhaps, more importantly, their own bias to the analysis. This is a central pillar of Constructivist Grounded Theory and it is where this method differs most from Classical Grounded Theory. Whereas Glaser & Strauss adopted a positivist approach, Charmaz, by contrast argued for an Interpretivist position. The researcher brings their own experience into the analysis to help make sense of the focused codes. An example of how the open codes build into conceptual categories is shown in Fig. 4. The participants' words are converted into codes by the researcher; these are then grouped into more focused codes



Being the cause of the non delivery	1
Not delivering what was expected	1
Knowing team members strengths	1
Depending on each other	1
Communicating	1
Being benevolent... or not	2
Being retrospective	2
meeting expectations	2
Spending time together asa a team	4
Building strong team	1
challenging to get to truth	4
getting the job done	2
Staying together	1
Needing opportunity to shine	2
Not living up to reputation	4
Reputation	5
Reviewing new team members code	4

Perception
Perception
Being a team player

137 Yeah. Yeah. So, like we have got I don't know.. there's a guy that is really outspoken, and you already immediately form an opinion of him and go, oh God, he is a troublemaker, he is going to cause lots of things. It depends on who you are. You see I am like that and I make a judgement straightaway, but I have learnt through my experiences just to maybe hold back and kind of think a bit more and deeply. So, there was a guy.... Yeah, the guy who shouts a lot, he has been here. I found out he is more frustrated in his role here and is frustrated at processes and stuff being flung and not being carried out. So, he is a frustrated individual in the Team, trying to push things to be done right.

Figure 3 Segment of MAXQDA window displaying coding and codes

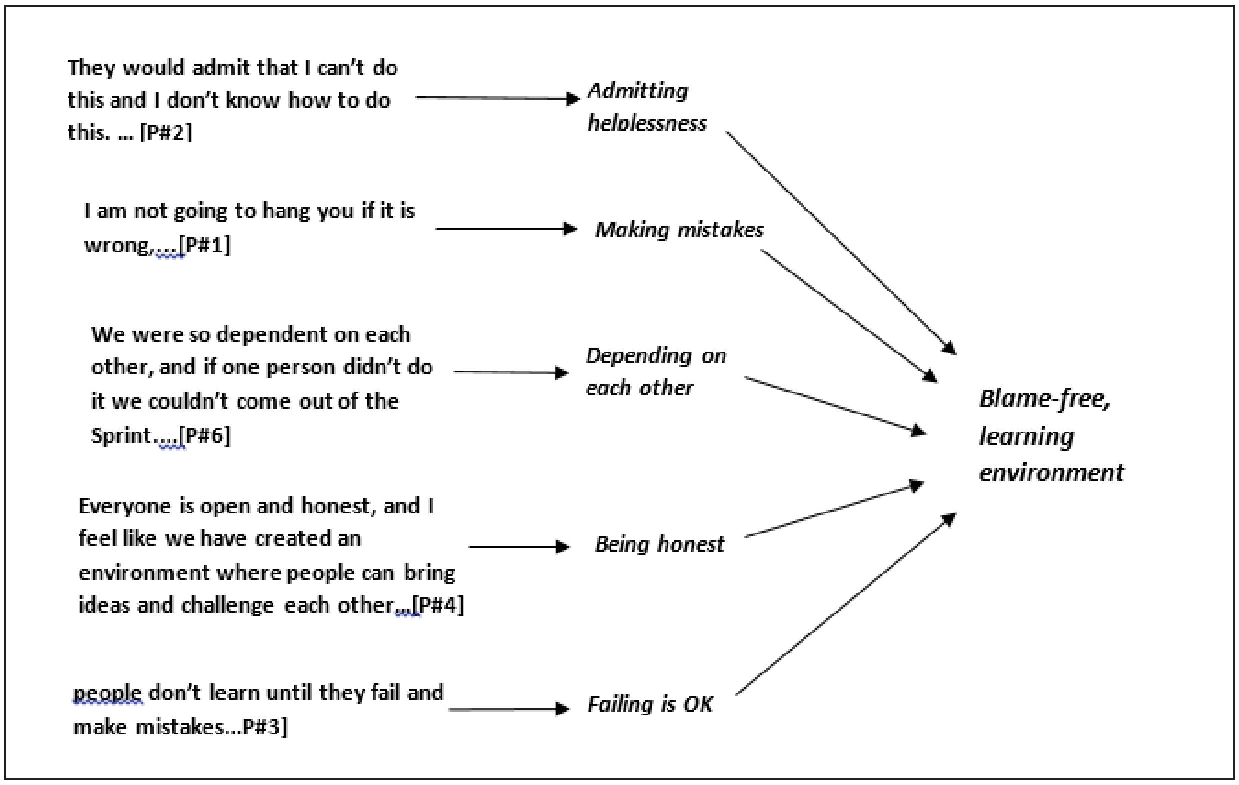

They would admit that I can't do this and I don't know how to do this. ... [P#2]
Admitting helplessness
I am not going to hang you if it is wrong,...[P#1]
Making mistakes
We were so dependent on each other, and if one person didn't do it we couldn't come out of the Sprint....[P#6]
Depending on each other
Everyone is open and honest, and I feel like we have created an environment where people can bring ideas and challenge each other...[P#4]
Being honest
people don't learn until they fail and make mistakes...P#3]
Failing is OK
Blame-free, learning environment

Figure 4 Emerging categories from codes

as shown in the middle of Figure 4 and these subsequently become the categories of the emerging theory as shown on the right of the figure.

V. Results

Although this research is still continuing it is nevertheless valid, in our opinion to present the key initial findings. For successful interdependence, knowledge sharing and collaboration to exist in Scrum teams, such as the one investigated in this study, various enablers must be in place to promote interpersonal trust, knowledge sharing and collaboration in the team. What follows attempts to take the participants words and interpret them through the lens of the researcher's own experience into categories which help to facilitate the growth of trust in the Scrum team. What results is a constructed theory grounded in the participants' experience.

A. Familiarity

In order to create a climate of interdependence it is necessary for the team to know one another quite well. Largely this is because *"We were so dependent on each other, and if one person didn't do it we couldn't come out of the Sprint."* P#6, Developer.

That this knowledge of team members happens as a function of time is indisputable since participants found that *"Trust evolves over time."* P#5, Developer, and "*I think that over time trust will build."* P#6, Developer.

The benefits of getting to know team members was explained as

"You need to get to know them, because I think it helps the Team dynamics." P#6, Developer.

Sometimes, the team appears to have bonded sufficiently well as *"they feel that they are meeting each other more than they are meeting their families."* P#3, Developer.

Ultimately *"With a good bond in the Team comes better trust. "* P#1, Scrum Master.

The benefit of being familiar is that team members "*know each other. They know that he is a good Developer."* P#1, Scrum Master.

Additionally *"it becomes clear where people's real strengths are."* P#4, Developer.

In addition to knowing each other well and perhaps as a way of getting to know each other better communication is important.

B. Communication

Communication is vital in a Scrum team since often it is necessary to resolve blockers that may have been mentioned in the Daily Stand Up. It helps if team members *"can talk to each other and get the things resolved."* P#1, Scrum Master.

Good communication rarely happens automatically. When discussing trust one respondent stated

"It has just got better. It has taken a long time, but I think it's because of regular communication and getting to know one another." P#2, Product Owner.

The success of the Sprint can be attributed to working towards a common goal

"Basically, how the objectives are communicated as to what we are trying to achieve in a Scrum." P#6, Developer

"Understanding the objectives and being clear about what we have to do." P#4, Developer.

Once communication and familiarity have settled in the Scrum team, team members can feel psychologically safe to be vulnerable to either ask for, or alternatively receive help from another team member.

C. Vulnerability to ask for /receive help

Due to the complexity of tasks required to be included in the Sprint *"you most likely will never know how to do everything. You will always need to go to someone."* P#4, Developer.

Consequently, it is not unusual for team members to admit "*that I can't do this and I don't know how to do that."* But generally, as a result of familiarity with one's team members, *"they would know somebody in the room that might be able to help them, might have more knowledge than them"* and *"You need to say if you need a hand."* P#5, Developer.

Team members need to feel safe enough in the team environment to be able to admit *"I need help with this because I don't know what the hell is going on here."* P#6, Developer.

Most often *"if you ask them they will help you. "* P#1, Scrum Master. Most likely this is because it is *"a core concept that everyone helps each other."* P#1, Scrum Master.

Again the time factor has an effect on this too *"they are trying to help a lot more and that has come over time."* P#2, Product Owner.

As described by the Scrum Master, *"They are able to identify the issues they have, and they are able to help each other whenever help is required between them."* P#1, Scrum Master.

This was reiterated as *"We are able to identify the issues we have, and we are able to help each other whenever help is required between us."* P#4, Developer.

The bottom line is that *"there is a willingness to help each other,"* P#2, Product Owner, in this team and that undoubtedly contributes to the team's interdependence and cohesiveness.

An alternative way of viewing this is that the interdependence in the team is sufficiently strong that *"the relationship that gets built makes them actually* ***want*** *to help."* P#4, Developer.

D. Openness

Part of trusting a team member relates to the team members' honesty and openness in disclosing pertinent information and not holding back on sharing knowledge.

As one respondent articulated *"If you are upfront with us, we work as a Team."* P#3, Developer. It should also be worth mentioning that *"The daily Scrum is completely transparent."* P#1, Scrum Master.

Added to this *"openness is huge…it is probably the core requirement of an Agile Team."* P#2, Product Owner.

Furthermore, this respondent claimed "*in our Team, at Team level I think that everyone is open and honest, and I feel like we have created an environment where people can bring ideas and challenge each other."* P#5, Developer.

E. Commitment to the task

With any team endeavour the key thing is to get commitment to the objective. If everybody on the team buys-in to the task it will most likely result in a successful outcome.

It is felt to be vital that the team has an *"understanding of what the goals and objectives of the Sprint are."* P#6, Developer.

This is in line with the ethos of keeping *"consistent with the objective that they have from the start of the Sprint."* P#4, Developer.

Additionally, in addition to commitment to the team's objective one team member felt it was important to know *"what we are delivering, who we are delivering to, and have visibility into that. It helps to….. I think it helps to build all that, you are not just in a silo, that you can see the bigger picture even though, okay, you may not have any input into it. But even what other Teams are delivering. What is it all going into…? I think that would be the thing and I think that helps with the whole trust and everything else around the Teams."* P#6, Developer.

F. Benevolence

It is most likely that this enabler of trust comes from within the team. As voiced by the Scrum master *"I want them all to be Team players*." P#1, Scrum Master.

Understanding the position vis à vis the larger picture was deemed to be important "*if we didn't have our basically all our eggs in the basket ready to go to exit the Scrum or to the Sprint we held up the core Team as well."* #6, Developer.

With a view to demonstrating a high degree of benevolence toward fellow team members, team members often engaged in an element of horse- trading to get the tasks accomplished, viz. *"I have got spare hours I will take this, you take that."* #5, Developer.

From the perspective of the Product Owner he noticed that team members *"want to do well because it helps the rest of the Team. ","* P#2, Product Owner.
To an extent this is because *"we do feel like we are responsible."* #4, Developer.

G. Integrity

Integrity is a key factor for establishing trust in a Scrum team *"integrity is crucial I think for a Team's success,"* P#4, Developer. In discussing integrity with one of the team members he commented that
"I think actually that is one good quality. We do have integrity." P#2, Product owner.
The most important aspect of integrity is that once it has been established the team has a vested interest in maintaining it.
"*That is when you start to find a really high level of integrity because you tell these guys, you are a high performing Team now, and they won't compromise that."* P#1, Scrum master.
This even cascades down to the developers themselves...
"It is about time spent together and it is about kind of taking smaller steps to get to the point that if we are high performing our integrity is higher." P#3, Developer.

H. Building Trust from Scratch

"*I think trust is earned. So, you are going to have to build up trust."* P#5, Developer.
"Once a new member starts in the Team we don't give him a trust as a Team from day one for sure." P#3, Developer.
New team members are scrutinised from the outset *"if he is a new member we give him more focus."* P#4, Developer.
A new team member's code is thoroughly checked by a more senior team member *"if the person is new you make more reviews for him."*P#3, Developer.
And "*You start to make more checkpoints for him."* P#5, Developer.
However, a new developer is not thrown in at the deep end. The philosophy in this organisation is to start small *"we just try to give him for example, one sprint or two, just to experience the Team and experience the Project without having any kind of dependency or commitment based on that new Engineer."* P#1, Scrum Master.
In fact, initially there is no danger of a new developer doing anything that might cause an issue since *"the first thing is to run the current system, the same like anyone else. That is the first thing we do. So, you have to execute and run the system. After you are able to execute and run the system, now you can start to change on that system."* P#4, Developer.

VI. Discussion

Although this study forms only one of what is intended to be several similar studies carried out as part of our research on trust in Scrum teams in various Irish software development organizations the findings are nevertheless considered to be significant in that they represent the findings from a successful multinational company based in the West of Ireland.
The literature refers to a stepwise calculative approach to building trust. Lewicki and Bunker describe, "achievement of trust at one level enables the development of trust at the next level [26]." This appears to be the approach taken in the company we worked with. New team members are firstly invited to run the current system and try to make sense of it. They are then given small tasks to run on this system and a more senior developer, who effectively acts as as mentor reviews and checks the new code. The new developer is allowed to make mistakes and asking for help is almost expected. As this is happening the new team member is becoming known to the team and familiarity, openness and integrity are established.

VII. Limitations

The key limitation is that the research has not yet concluded. Thus what is presented is a snapshot which pertains to a single co-located Scrum team in a single multinational. Consequently, at this stage the findings are in no way generalizable.

VIII. Conclusion and Future Work

In terms of future work the research is ongoing in other multinationals. It is hoped that from this work the body of knowledge regarding the development of trust in Scrum teams will be augmented.

References

[1] VersionOne, "VersionOne 12th Annual State of Agile Report," 2018.

[2] M. Fowler and J. Highsmith, "The Agile Manifesto," *Softw. Dev.*, vol. 9, no. 8, pp. 28–35, 2001.

[3] N. B. Moe, T. Dingsøyr, and T. Dybå, "A teamwork model for understanding an agile team: A case study of a Scrum project," *Inf. Softw. Technol.*, vol. 52, no. 5, pp. 480–491, May 2010.

[4] B. RAMESH, L. CAO, K. MOHAN, and P. XU, "CAN DISTRIBUTED SOFTWARE DEVELOPMENT BE AGILE?," *Commun. ACM.*, vol. 49, no. 10, pp. 41–46, 2006.

[5] S. Dorairaj, J. Noble, and P. Malik, "Understanding lack of trust in distributed agile teams: a grounded theory

study," in *16th International Conference on Evaluation & Assessment in Software Engineering (EASE 2012)*, 2012, pp. 81–90.

[6] E. Hasnain and T. Hall, "Investigating the Role of Trust in Agile Methods Using a Light Weight Systematic Literature Review," in *XP 2008: Agile Processes in Software Engineering and Extreme Programming*, 2008, pp. 204–207.

[7] J. R. Katzenbach and D. K. Smith, "The Discipline of Teams Harvard Business Review," 1991.

[8] K. Schwaber and J. Sutherland, *The Scrum Guide. The definitive guide to scrum: The rules of the game*. Scrum.org 268, 2013.

[9] B. Morgan, A. S. Glickman, and E. A. Woodward, "MEASUREMENT OF TEAM BEHAVIORS IN A NAVY TRAINING ENVIRONMENT," 1986.

[10] R. Hoda, "Self-Organizing Agile Teams: A Grounded Theory," 2011.

[11] L. Faraj, S. & Sproull, "Coordinating Expertise in Software Development Teams," *Manage. Sci.*, vol. 46, no. 12, pp. 1554–1568, 2000.

[12] R. Saavedra, C. P. Earley, and V. Dyne, L, "Complex Interdependence in Task-Performing Groups," *J. Appl. Psychol.*, vol. 78, no. 1, pp. 61–72, 1993.

[13] S. H. Courtright, G. R. Thurgood, G. L. Stewart, and A. J. Pierotti, "Structural Interdependence in Teams: An Integrative Framework and Meta-Analysis," *J. Appl. Psychol.*, no. May, 2015.

[14] J. Colquitt, J. A. LePine, and M. J. Wesson, *Organizational behavior : improving performance and commitment in the workplace*. 2016.

[15] R. Wageman, "Interdependence and Group Effectiveness.: GMIT Libraries Collections," *Adm. Sci. Ouarterly*, vol. 40, no. March, pp. 145–180, 1995.

[16] J. C. Wyatt, "0. Management of explicit and tacit knowledge," *J R Soc Med*, vol. 94, pp. 6–9, 2001.

[17] H. Frank Cervone, "'Understanding agile project management methods using Scrum', OCLC Systems & Services: International digital library perspectives," *OCLC Syst. Serv. Int. Digit. Libr. Perspect.*, vol. 27, no. 1, pp. 18–22, 2011.

[18] M. Polanyi, *The tacit dimension*. University of Chicago Press, 1966.

[19] M. Levy and O. Hazzan, "Knowledge management in practice: The case of agile software development," in *2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering*, 2009, pp. 60–65.

[20] H. Fuks, A. B. Raposo, M. A. Gerosa, and C. J. P. Lucena, "APPLYING THE 3C MODEL TO GROUPWARE DEVELOPMENT," *Int. J. Coop. Inf. Syst.*, vol. 14, no. 3, pp. 299–328, 2005.

[21] S. Nerur, R. Mahapatra, and G. Mangalaraj, "Challenges of Migrating to Agile Methodologies," *Commun. ACM May Commun. ACM*, vol. 48, no. 5, 2005.

[22] E. Sopensky, "The Skill and Art of Collaboration," *Tech. Commun.*, vol. 41, no. 4, pp. 709–713, 1994.

[23] M. Mach, S. Dolan, and S. Tzafrir, "The differential effect of team members' trust on team performance: The mediation role of team cohesion," *ournal Occup. Organ. Psychol.*, vol. 83, pp. 771–794, 2010.

[24] B. G. Glaser and A. L. Strauss, *The discovery of grounded theory : strategies for qualitative research*. 1967.

[25] K. Charmaz and Askews & Holts Library Services., *Constructing grounded theory*. 2014.

[26] R. J. Lewicki and B. B. Bunker, "Trust in relationships: A model of development and decline," in *The Jossey-Bass conflict resolution series. Conflict, cooperation, and justice: Essays inspired by the work of Morton Deutsch*, 1995.