A Constructivist Grounded Theory of Trust in Agile Scrum Teams

Trish O'Connell
School of Science & Computing
Galway-Mayo Institute of Technology
Galway, Ireland
trish.oconnell@gmit.ie

Owen Molloy
Dept. of Information Technology
National University of Ireland
Galway, Ireland
owen.molloy@nuigalway.ie

***Abstract*— As Scrum is predominantly a team-based activity, it is consequently an intensely social endeavour. In order to deliver on the mutually agreed goals of the Sprint, Scrum teams need to collaborate and share knowledge effectively. Many authors have cited trust as being crucial to fostering collaboration and knowledge sharing. However, to date there has been no published research into this crucial social construct in the context of agile software development teams. This paper revisits the conceptual model of trust presented at SOFTENG 2018 in light of the findings from a preliminary Constructivist Grounded Theory study conducted on two Scrum teams in a major multinational software development company in the West of Ireland.**

Keywords-Agile; Scrum; Team; Trust; Collaboration; Knowledge-sharing; Constructivist Grounded Theory;

I. INTRODUCTION

Agile software development is a "task oriented, social activity [1]." This is particularly true of Scrum, the ubiquitous software development framework most closely associated with Agile. The Agile Manifesto [2] advocates "business people and developers must work together daily throughout the project... build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done." In Scrum, this is accomplished by the Scrum team. The Scrum team is vital to achieving the goals of the software development initiative.

As Moe, Dingsøyr and Dybå posit "Software development depends significantly on team performance, as does any process that involves human interaction [3]." In part, this is because, according to Schwaber, "When people work by themselves, they can achieve great things. When people work with others, they often achieve synergy, where the joint effort far exceeds the sum of the individual efforts [4]." In order to fully achieve synergy team members need to share knowledge within the team and collaborate to achieve the goals of the Scrum Sprint, which is the timeboxed period used to develop a software increment. Dorairaj, Noble and Malik cite trust as "one of the key factors [5]" in successful Agile projects. Largely this is because, "trust has been found to be a critical factor facilitating collaboration [6]." Tschannen-Moran elucidates, "collaboration and trust are reciprocal processes; they depend upon and foster one another [7]." With regard to knowledge sharing Ghobadi posits "the unique and inherent characteristics of software development signify the importance of effective knowledge sharing, referring to the exchange of task-related information, ideas, know-hows, and feedback regarding software products and processes [8]." Butler refers to research conducted by Zand when he states that "trust leads to the disclosure of information [9]." Furthermore Zand describes how "persons who trust one another will provide relevant, comprehensive, accurate, and timely information, and thereby contribute realistic data for problem-solving efforts [10]." The study by Fields and Holste acknowledged the role of trust in a "willingness to share and use knowledge [11] ."

Whilst trust has often been included in the academic discourse it has predominantly been through a sociological, psychological, economic or organizational lens. Consequently, the findings have been somewhat incongruous when applied to a software development context. McKnight and Chervany referred to the lack of consensus about trust as causing "conceptual confusion [12]." In this study a preliminary constructivist grounded theory will be presented which seeks to understand the construct of trust and how it develops in the Agile Scrum software development team engaged in the development of software products.

Section II of this paper presents the background to the study in terms of knowledge sharing, collaboration and trust. Section III outlines the research that was conducted including a breakdown of the study methodology and how it was implemented. Section IV presents the results of the research and leads into Section V, where the findings are presented. Section VI presents a discussion of the results and leads into Section VII which examines the limitations of the research. Finally, Section VIII concludes and outlines the plans for future work.

II. BACKGROUND

According to the co-creators of the Agile Manifesto, Scrum teamwork is characterised by "intense collaboration [2]." Tabaka specifically refers to the concept of collaboration in a software development context, citing as useful, the sharing of "ideas, information, decisions and

solutions [13].” The relevance of collaboration to Agile software development was highlighted by Nerur, Mahapatra and Mangalaraj who expounded “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 [14].” It is important for the team to work cooperatively to share information where cooperation, according to Collier involves the “ smooth transfer of work in progress, work products, and information from one member to another [15].” Collaboration, by contrast, “elevates groups beyond cooperation, adding an essential ingredient for emergent, innovative, and creative thinking [15].”

Ghobadi and Mathiassen posit, “Software development is a collaborative process where success depends on effective knowledge sharing [16].” Thus it may be argued that knowledge sharing plays a vital part in enhancing the success of the Sprint. As Ryan and O’Connor assert “knowledge sharing is a key process in developing software products [17].”

In order to facilitate collaboration and knowledge sharing in a Scrum team there is one key ingredient, which binds the team together, trust. Chen, Lin and Yen state unequivocally that “trust leads to better inter-organizational collaboration and knowledge sharing [18].”

A. *Knowledge Sharing*

Cummings (2004) argues that knowledge sharing within a group “includes the implicit coordination of expertise … and information about who knows what in the group [19].”

There are two types of knowledge which are of vital significance in Scrum teams, explicit knowledge and tacit or implicit knowledge. The distinction between these two distinct types of knowledge was highlighted by Nonaka in 1994 when he wrote about the “joint creation of knowledge [20]” in organizations. Wyatt defines explicit knowledge as consisting of “of facts, rules, relationships and policies that can be faithfully codified in paper or electronic form and shared without need for discussion [21].” Nonaka concurs, describing that explicit knowledge “refers to knowledge that is transmittable in formal, systematic language [20].”

By contrast, Nonaka maintains that tacit knowledge is “a personal quality, which makes it hard to formalize and communicate. Tacit knowledge is deeply rooted in action, commitment, and involvement in a specific context [20].” Chau et al. posit that tacit knowledge includes “system knowledge, coding convention, design practices, and tool usage tricks [22] .” The authors argue that “developers tend not to document this knowledge and it is usually not explicitly taught through formal training [22].”

From a Scrum team perspective the sharing of both types of knowledge is crucial to Scrum team performance since as Levy and Hazzan claim “software development work requires various forms of explicit as well as implicit knowledge [23].”

At the outset, in a Scrum team, the Sprint Planning meeting is the forum where the product backlog is discussed and negotiated. In general the information is shared explicitly among the team members. However, when the Sprint commences it is not unusual for developers to discover obstacles which are shared with the team at the Daily Scrum. The sharing of information at the Daily Scrum is mostly explicit in nature given that the team uses the meeting as the vehicle to describe what progress has been made during the previous day; what progress is expected to be made in the current day and what, if any, blockers are impeding progress and causing an impasse in the development.

However, knowledge sharing, whilst vital is not sufficient on its own for successful software development. Judy and Krummins-Beens describe how the Agile Manifesto emphasizes “collaboration among team members and project sponsors [24].”

B. *Collaboration*

Tabaka establishes the importance of collaboration by stating that “when teams declare a collaborative imperative in their work, it is their pledge to employ consensus-based decision approaches through participatory decision-making. They apply high-bandwidth information gathering coupled with well-formed and well-articulated priorities [13].” The Agile Manifesto argues strongly for collaboration advocating “customer collaboration over contract negotiation [2]” and “developers must work together daily throughout the project [2].” Fowler and Highsmith contend that “only through ongoing collaboration can a development team hope to understand and deliver what the client wants [25].” Tabaka summarises that collaboration in Agile has become “an integral component of what would be considered a responsive, adaptive software development approach [13].” Chau and Maurer concur, positing software development is “a collaborative process that needs to bring together domain expertise with technological skills and process knowledge [26].”

That Scrum software development is a collaborative endeavour is undeniable. The Agile Manifesto advocates strongly for “face-to-face conversation [2].” Paulk argues that this can best be achieved by having the team members in close proximity to each other, stating “Agile teams are expected to be co-located [27].” In Scrum, Deemer *et al.* advocate “An excellent practice is for the team to be collocated [28].” In addition to the informal opportunities offered by collocation of having team members discuss ideas and solutions to possible problems the Scrum framework has a number of events which facilitate

knowledge sharing and problem solving by providing opportunities for collaboration.

McHugh, Conboy and Lang cite "sprint/ iteration planning, daily stand-up, and sprint/iteration retrospective [29]" as three key practices which require the "collective participation of all team members with a focus on people, communication, interaction, and teamwork [29]."

The Sprint planning meeting is the timeboxed meeting which happens prior to each iteration. It is at this meeting that the planning for the upcoming Sprint "is created by the collaborative work of the entire Scrum Team [30]." It is at this meeting that the Scrum team negotiates a shared understanding of the work to be completed for the upcoming Sprint.

The development team uses the Daily Scrum or Daily Stand-Up as it is often called "to inspect progress toward the Sprint Goal and to inspect how progress is trending toward completing the work in the Sprint Backlog. The Daily Scrum optimizes the probability that the Development Team will meet the Sprint Goal [30]." The Daily Scrum is the optimum vehicle for collaboration since it considers what was accomplished in the previous day of the Sprint. It explores what will be achieved in the coming day and it seeks to clarify what, if any, impediments exist to progress. It is at this point that collaboration comes into its own. If a team member has a particular obstacle which is causing an impasse to progress, the team can come together to brainstorm possible solutions in what Levy and Hazzan refer to as a "*collaborative workspace* – a space which supports and facilitates communication [23]."

In similar vein, the Sprint Retrospective is the forum to "encourage the Team to revise, within the Scrum process framework, its development process to make it more effective and enjoyable for the next Sprint [31]."

C. Trust in the Academic Discourse

Mach, Dolan and Tzafrir argue, "trust is an integral part of teamwork because team tasks require a high level of interdependence between members [32]." Furthermore, Sandy Staples and Webster postulate, "team members must rely on each other and share required knowledge with others. If sharing does not happen within the team, it is unlikely to meet its objectives [33]."

As previously stated, the literature on trust examines it from a number of different perspectives. Consequently the result is highly fragmented and unintegrated when it comes to applying it to the Scrum team context. Nevertheless, it is possible to take some generalities from the extant research into trust.

Sociologists have often seen trust as a type of mystical 'faith' that one person has in another. Simmel in 1950 wrote "confidence is intermediate between knowledge and ignorance about a man. The person who knows completely need not trust [34]." Giddens would appear to concur advocating "There would be no need to trust anyone whose activities were continually visible and whose thought processes were transparent [35]." It is likely that from a psychological viewpoint these sociologists were referring to trusters having a "propensity to trust" or "trusting disposition" as referred to by Rotter [36] and McKnight and Chervany [12].

The notion of expectancy or expectation is often synonymous with trust in the literature. Largely this notion emanates from the realm of social psychology. Deutsch offers as a definition of trust that "Person I trusts Person II to do something and I perceives that the behavior he expects of Person II is perceived by II to have relevance to I [37]." Barber concurs with Deutsch acknowledging that trust is "a dimension of all social relationships [38]." In his seminal volume The *Logic and Limits of Trust* he describes how actors in social relationships have "expectations" of each other [38]. Given that expectation has the connotation of often being reciprocated, Deutsch associated trust with a "reciprocal, cooperative relationship between people who make the decision to trust [1]." Additionally Deutsch introduced the notion of 'competence' being involved in the fulfilment of expectations. One can only meet someone's expectations, if one has the competence to so do.

Gabarro added to Deutsch's notion of competence and included "openness about task problems [39]." Openness, "freely sharing ideas and information [40]," and integrity, "honesty and truthfulness [40]," were also cited by Butler and Cantrell who listed these as conditions of trust.

Assuming that we trust people that we know better than those we do not know Luhmann contended that familiarity should also be seen as the "prerequisite for trust [41]."

Whilst initially Mayer [42] and Mishra [6] perceived trust as a willingness to accept vulnerability, Mayer described this willingness as largely cognitive. That this cognitive based trust should eventually develop affective or emotional overtones was postulated by McAllister who described "affect based trust [43]." This view was advocated also by Lewicki and Bunker who described initial trust as being "calculus based [44]." To clarify, calculus based trust is arrived at in a stepwise process with each trusting endeavour being used as the basis for the next level. In this sense it is described as "tactical climbing [44]." From this cognitive position Lewicki and Bunker then describe "knowledge based" trust as relying on "information rather than deterrence. The better we know the other individual, the more accurately we can predict what he or she will do [44]." The authors also described "identification based" trust in which a "collective identity develops [44]."

Thus, it would seem that the academic discourse presents trust as initially cognitive in that one makes a judgement call on whether to trust, and if expectations are fulfilled, this calculative trust can develop into an emotional

connection with the person being trusted into a deeper bond of genuine affect where both the trustee and the truster have "fully internalized the other's preferences [45]."

D. The Scrum Team

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 [46]." According to Schwaber and Sutherland, the co-creators of the Scrum framework, Scrum teams should be "small enough to remain nimble and large enough to complete significant work within a Sprint [47]." Three to nine team members is regarded as being optimal. In terms of complementary skills Scrum team members, viewed as an entity, tend to possess "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) [48]."

In addition Katzenbach and Smith advocate the need for "problem-solving and decision-making skills, and interpersonal skills [46]."

The common purpose element of the team definition contributed by Katzenbach and Smith is unquestionable. By its very design Scrum teams collaborate to achieve Sprint goals. As Moe, Dingsøyr and Dybå explain "In a software team, the members are jointly responsible for the end product and must develop shared mental models by negotiating shared understandings about both the teamwork and the task. Project goals, system requirements, project plans, project risks, individual responsibilities, and project status must be visible and understood by all parties involved [49]."

Similarly, mutual accountability in Scrum is *de rigeur* given the requirement to account for progress at the Daily Scrum meeting. Cervone explains "the purpose of the daily Scrum is to both track the progress of the team as well as allow team members to make commitments to each other and the Scrum Master so that work can proceed in the most expedient and unimpeded manner." McHugh,Conboy and Lang concur that the Daily Scrum meeting "provides transparency and visibility on the day-to-day progression of tasks [29]."

E. Trust in the Scrum Team

Having examined trust in isolation in the academic discourse and furthermore having introduced the Scrum team as the vehicle for collaboration and knowledge sharing and collaboration it is somewhat surprising that no published studies appear to have "examined trust in an agile context [29]." Consequently, what follows is an attempt to synthesize the extant literature with a view to applying it to a Scrum team. Figure 1 represents a first stage conceptual model of trust in a Scrum team.

1) Perception

Whilst perception does not really appear in the literature on trust the authors contend that in any team scenario, perception may well play a role. An individual who is new to a team will most likely be subject to a degree of initial judgement. Based on how they are initially perceived the calculus based trust will enhance or detract from their position.

2) Reputation

Some authors [29] contend that reputation is involved in the trust construct. Undoubtedly, a team member's reputation for delivering on their commitments plays a part in whether or not they can be trusted to deliver on their next commitment. This too must surely play a part in the decision to trust or calculus based trust scenario.

3) Integrity

As a team member becomes enmeshed in the Scrum team their integrity and credibility is often tested by other team members. Insofar as a team member does what s/he says s/he will do, integrity is strengthened in the calculative decision to trust.

4) Competence

As shown in Figure 1 the first four conditions for trust, as described above function to enhance the positive reinforcing feedback loop that is calculus based trust. In other words, as a team member demonstrates integrity or competence, for example, the trust in them grows. This allows the relationship to transition to knowledge based trust in which familiarity and openness themselves function as positive feedback loops as described below.

5) Familiarity

As the team members spend time together they come to know each other better; a good rapport is established and the relationships within the team can move past the calculus-based, cognitive decision to trust to a more affect-based knowledge of the other. It is at this stage that the team has really bonded. As Santos et al. describe "Agile values and principles foster changes in team members' attitudes and strengthen their relationships [50]."

6) Openness

Largely as a consequence of an increase in familiarity the team members should become more open with each other. As Zand described in Section I, "persons who trust one another will provide relevant, comprehensive, accurate, and timely information, and thereby contribute realistic data for problem-solving efforts [10]." This happens directly as a result of the openness in the team.

Once this reinforcing loop has begun it is argued that the team members come to identify with each other's goals and the goals for the Sprint itself. At this stage the calculus based trust has been sidelined in favor of affective bonds within the team.

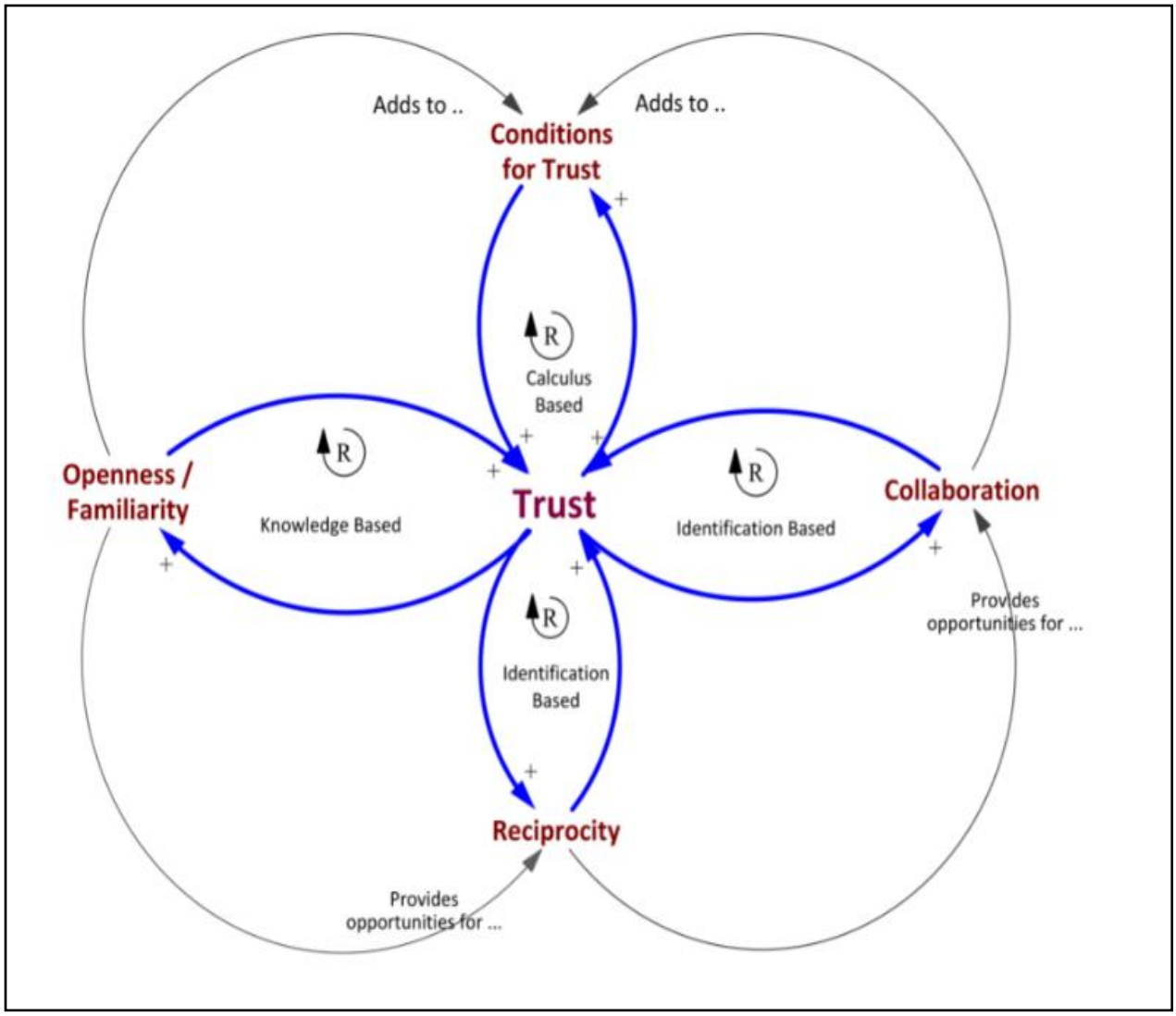

Figure 1. Conceptual Model of trust in a Scrum Team

7) Reciprocity

Once familiarity has become embedded in the team DeVries et al. describe "a cycle of reciprocity, in which team members are more likely to exchange (i.e., both donate and collect) knowledge with each other [51]" becomes the norm.

This again would appear to be a reinforcing loop since as the team bonds the emotional ties become stronger and team members are more likely, and willing, to help each other.

It should be noted, however, that this is a conceptual model of trust within the Scrum team. With a view to understanding the construct of trust in the Scrum team and how it develops in the Agile Scrum software development team engaged in the development of software products the research study that was undertaken will now be described.

III. THE RESEARCH

As this research involves the construct of trust, which does not well suit quantitative analysis it would be more usual for social science topics such as trust to fall into the realm of qualitative data. Johnson and Onwuegbuzie outline the strengths of qualitative research as "The data are based on the participants' own categories of meaning [52]."

Since this research is quite unequivocally involved in the perceptions and feelings of the Agile team members who will be interviewed the ontological perspective of this study must be subjective. In terms of the epistemology that underpins this research the constructivist epistemology (also referred to in the literature as Interpretivist) asserts, "social phenomena and their meanings are continually being accomplished by social actors. It implies that social phenomena are not only produced through social interaction

but are in a constant state of revision [53].” According to Vanson, the interpretivist approach “suggests that facts are based on perception rather than objective truth. With this approach, the conclusions are derived from the interpretations of the participants rather than the abstract theories of the researcher [54].”

Thus, it is intended to use a constructivist grounded theory approach to this research with the intent of gathering the views, perspectives and feelings of the members of a purposive sample of Agile software development teams from a selection of different industries with a view to generating a theory of how trust is developed and serves to enhance collaboration in Agile teams.

It is hoped that using this inductive approach this study will in some way contribute to understanding the construct of trust enhancing team performance in Agile software development teams.

Grounded theory was initially conceived by Glaser and Strauss as a polemic against the logico-deductive method of generating a theory whereby new knowledge (theory) follows from old knowledge through the application of research hypotheses and sound arguments that verify these new theories. Glaser and Strauss, by contrast, argued against data collection being influenced by pre-conceived hypotheses. Rather, “systematic data collection and analysis should lead into theory [55].” However, whilst Glaser and Strauss adopted an ontologically positivist approach Charmaz [56], by contrast advocates an interpretivist approach to the process acknowledging “subjectivity and the researcher’s involvement in the construction and interpretation of data [56].” Since trust is socially constructed the study will adopt the Constructivist Grounded theory (CGT) method as described by Charmaz.

In CGT the researcher must obtain “rich data [56]” from interviews with participants. Rich data refers to collecting data which fully addresses the complexities and depth of the topic under study. The data is then analysed, initially using what is termed “initial coding [56]” where each sentence is fractured and analysed for meaning. Through the process of constant comparison the open codes eventually build into focused codes which are basically at a higher level of abstraction. Eventually the theory emerges from the data as codes are elevated to categories.

With a view to ensuring that the codes fully describe the emerging theory a process known as theoretical sampling is used to elaborate and refine the newly-constructed categories. This is achieved by continuing to sample until no new categories emerge.

In order to conduct this study purposive sampling was used to contact software development companies that use Scrum as their development methodology. Purposive sampling [57] is a type of focused sampling and in this case an organization known to use Scrum was approached and permission was sought to conduct the research. The company has a number of onsite Scrum teams and given the logistical issues, viz team availability, participant workload etc. we were able to interview participants from two of the local onsite teams as shown in Table I.

TABLE I. TEAM COMPOSITION AND ROLES

Participant #	Team A	Participant #	Team B
P#1	Scrum Master	P#6	Scrum Master
P#2	Product Owner	P#7	Product Owner
P#3	Developer	P#8	Developer
P#4	Developer	P#9	Developer
P#5	Developer	P#10	Developer
		P#11	Developer

In-depth interviews were conducted with all of the participants with a view to eliciting what Charmaz refers to as *‘rich data’* [56]. The interviews lasted from 30 minutes to 50 minutes. Each interview was audio recorded and transcribed. With a view to ensuring that all of the nuances and subtleties were captured by the author, as illustrated in Figure 2, the transcribed interviews were subsequently returned to the participants for verification.

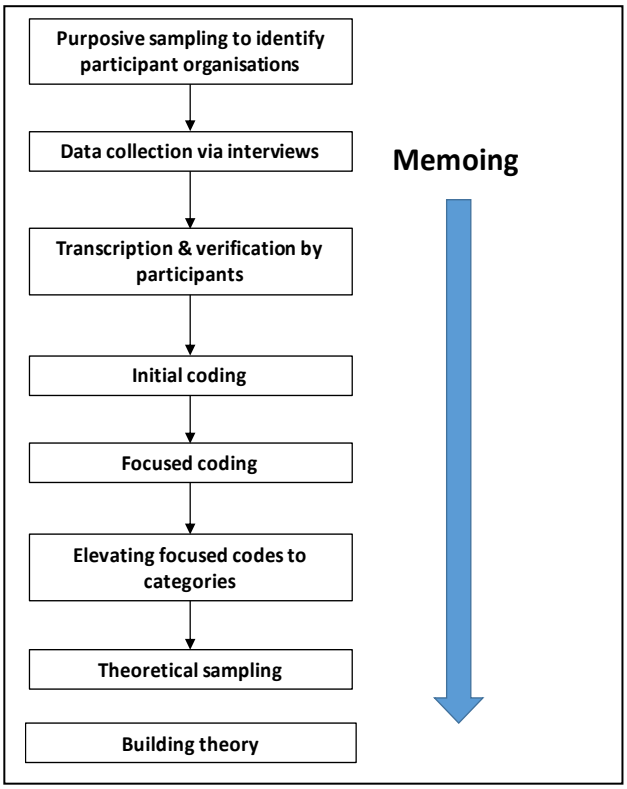

Figure 2. The Constructivist Grounded Theory process

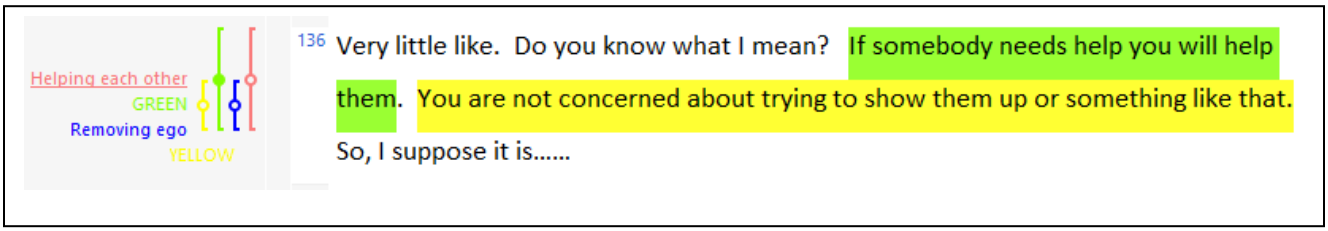

Figure 3. Coding fragment from Interview with P#9

Subsequent to gathering background data and obtaining consent for audio recording from each participant, the interviews focused initially on collaboration and then moved on to describing how trust is established in the Scrum teams. Following the interviews, the researcher, in accordance with the established constructivist grounded theory methodology, transcribed the interviews. Additionally, memos detailing ideas and refinements that, it was hoped, would advance theoretical understanding were written. It is, perhaps, noteworthy that interviews were largely unstructured but participants who were not naturally loquacious 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.

IV. Results

In keeping with the tenets of Constructivist Grounded Theory 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 the process of initial coding began. This is where 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 e.g., "removing ego" [P#9] which is depicted in Figure 3. On the right hand side, highlighted in yellow, is the fragment of what was said by the participant. On the left, in blue font, is the code that was used to encapsulate what it was felt the participant meant. Similarly the code 'helping each other' on the left hand side is associated with the fragment of the interview on the right hand side where the participant commented that "if somebody needs help you will help them."

Once the interview was coded, subsequent interviews were analysed in a similar manner 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 previous transcripts were available to view in a portion of the window as shown in Figure 4. On the right of Figure 4 is an interview displaying codes.

This facilitated what Charmaz refers to as "focused coding" [56] where codes are analysed to advance the theoretical direction of the study. Charmaz describes these codes as more conceptual than the initial coding.

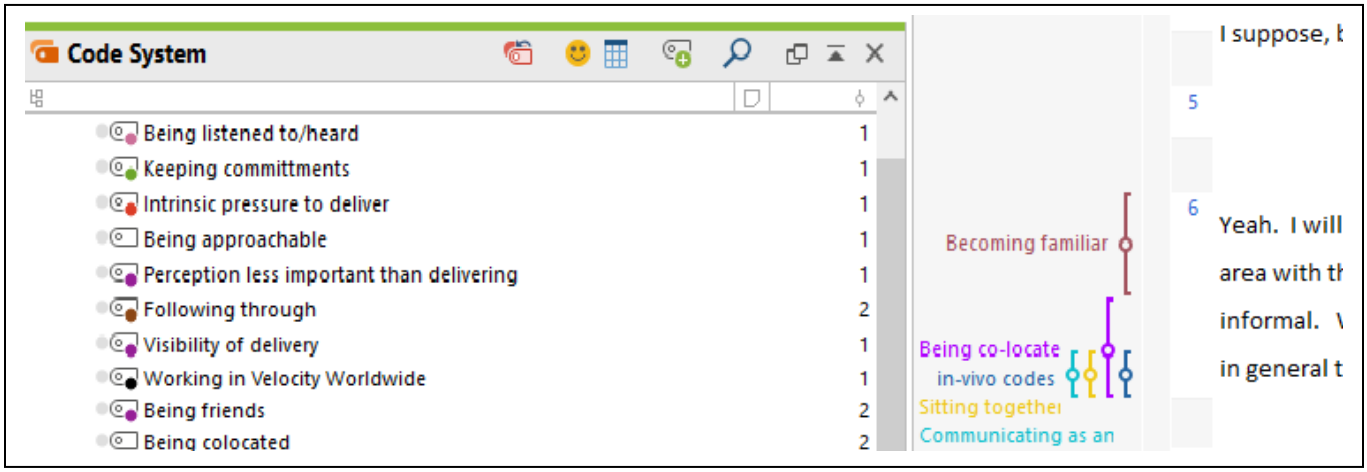

Figure 4. MAXQDA Window depicting codes and coding

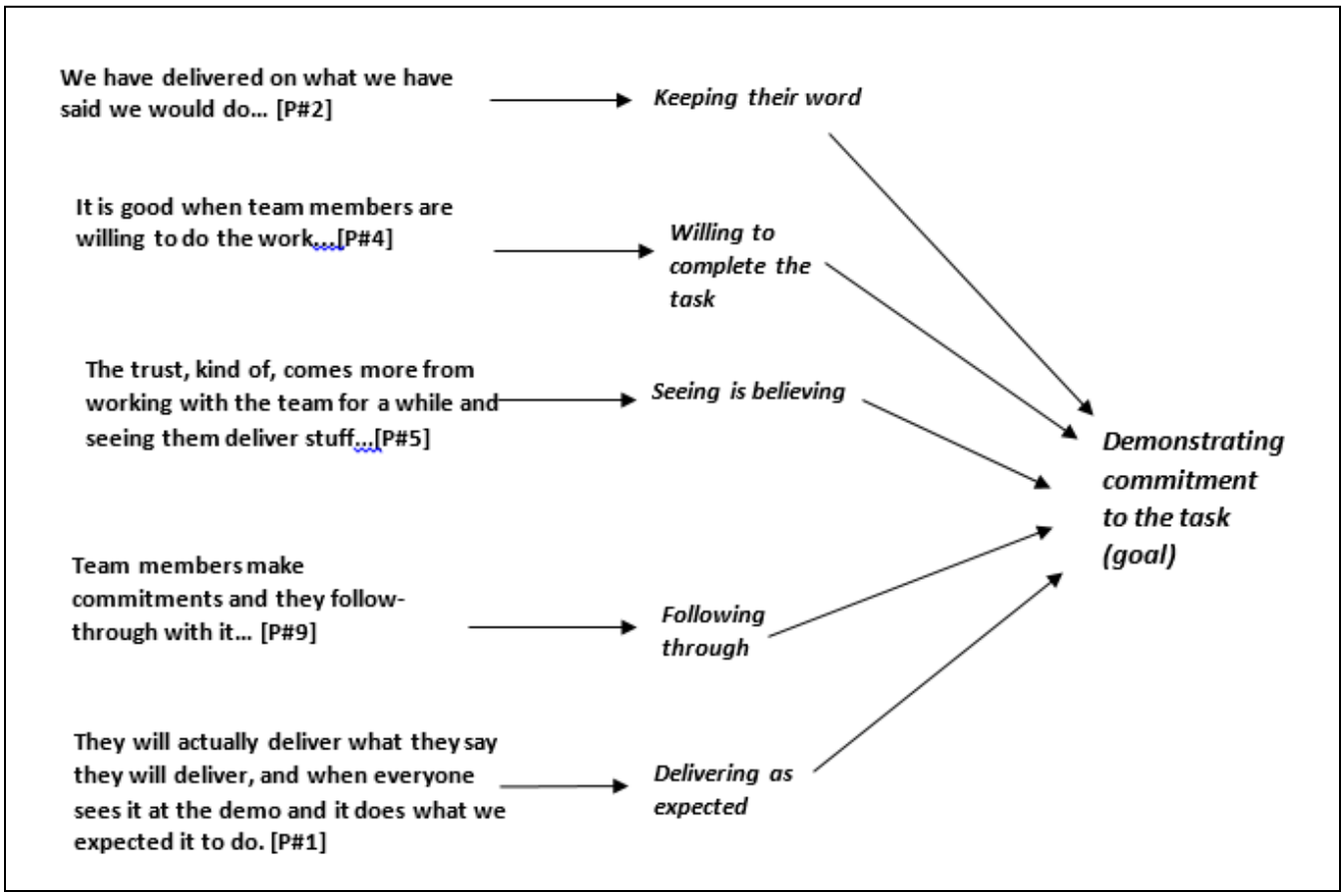

Figure 5. Emerging categories from codes

Finally, the focused codes were raised to conceptual categories which, when integrated, addressed the question of how trust is fostered and developed in Scrum teams. It is at this stage that we had to co-construct the meaning with the participants. Central to Constructivist Grounded Theory the researcher brings their own experience into the analysis to help make sense of the focused codes. Figure 5 depicts an example of how the codes emerge as categories.

V. FINDINGS

Whilst this research is still ongoing it is nevertheless believed useful to present the initial key findings.

For successful knowledge sharing and collaboration to exist in Scrum teams, such as the ones investigated in this study, various factors must be in place to promote inter-team trust.

A. Commitment to the Sprint goal

During the Sprint planning meeting the team reaches a consensus about the Product backlog. This dictates the plan of what will be achieved in the upcoming Sprint, who will undertake it and how long it is estimated to take. That the teams are working towards a common goal for their team Sprints is evidenced by:

"*We are all focused in around a common goal, and a common vision of what we are doing, and the guys buy into that."* P#1, Scrum master.

Commitment to a common goal is of significant importance as Badke-Schaub *et al.* state "Team performance can benefit from shared mental models in situations with a high need of information exchange in the team [56]."

B. Having integrity

As a team member being honest and transparent with team mates is a key requirement to foster trust. It is important that:

*"they do what **they** say, not what **you** say"* P#3, Developer.

Another participant highlighted the importance of:

"If they say they are going to do something, they do it or they put their hand up and say that didn't get done today we'll get it tomorrow." P#7, Developer.

"You could have saved us a lot of hassle and it would have made for a lot simpler conversations if you had just gone ahead and done what we had advised you to do in the first place." P#1, Scrum Master

C. Openness and Honesty

Openness and honesty are a crucial component of the Scrum teams' ability to move the project forward as evidenced by:

"I have never heard anybody come in and lie at the stand-up, you know, to say, oh, yeah, I am doing great, when they are not." P#5, Product Owner,

The participants agreed that delivering on what has been committed at the Daily Stand up is the final arbiter of success:
"There are a lot of people who talk but it doesn't prove anything really until it is actually done." P#8, Developer.

D. Familiarity with team members

There was unanimity that having friendly relations with one's team mates improved trust and thus enhanced knowledge sharing and collaboration:

"The more familiar you are with people and the friendlier you get with them, the easier it is to work with them and have those informal conversations." P#9, Developer.

"You would be sitting together at lunchtimes and you would be chatting about this and that and the rest of it." P#7, Product Owner.

"I know them all fairly well on a personal level outside of work. We wouldn't meet too much outside of work like, but talk at breaks, etc." P#4, Developer.

In fact, some members of the team seemed to know each other so well that:

*"You know the name of the kids, you know, more or less when the birthday of the kids are, and that, kind of thing."*P#2, Developer.

And from another team member:
"I would see Joe in here every day. His wife bought my car." P#3, Developer.

To sum up:
"Outside of work I would say that team members would know each other socially, in either their kids going to Clubs or sporting things." P#1, Scrum Master.

E. Seeking and Accepting Help

It was somewhat surprising that one of the key findings from the study was the importance of team members being able to both ask for and accept help from their peers.

"it is okay to not be able to do something straight away like you can ask your Team-mates and you will eventually get it done" P#8, Developer.

"When the team is working well everyone is prepared to say I do not know how to do this today but give me a day and I'll find out who is really good at it and they will help me." P#7, Developer.

"If you see someone has any problem or any concern, or they, even any, kind of, questions that you have, we are really not scared to ask them." P#2, Developer.

Perhaps the rationale for this came from the developer who commented
"Definitely, on our Team, people are motivated by delivering a good product and delivering what they say they will deliver. That is our primary motivation and we are willing to help each other. There is no selfishness in it." P#9, Developer.
This lack of selfishness and team spirit was encapsulated by the Scrum master who stated
"We are not an individual, we are a group here, we have to fight this battle" and
"We are all in this or none of us are in it." P#1, Scrum Master.

F. Competence

It was expected that competence would feature prominently in the interviews but surprisingly this did not appear to be the case. Rather there was an acceptance that
"I think that with time enough everybody can be competent" P#2, Product Owner.
There appeared to be a recognition that
"You have to accept that everyone has different levels of abilities." P#4, Developer and

*"Some people within the team could have competence in certain areas and would have lower competence in other area*s." P#1, Scrum master.

VI. DISCUSSION

It woud appear from the research that the conceptual model as illustrated in Figure 1 came very close to accounting for the empirical findings. However, the use of CGT was not intended to validate the model. The strength of CGT in this study lies in exploring the participants' view of trust within their Scrum teams. CGT builds the theory out of the 'rich data' collected from the participants' own experience.

Whilst integrity, openness and familiarity featured strongly throughout the interviews it would appear that perception and reputation do not appear to matter unduly. It might be argued that the category of seeking and accepting help are components of reciprocity but there is a degree of limitation in this as reciprocity has a two way connotation whilst asking for and accepting help tends to only benefit the team member who has requested the help.

Competence really did not seem to be as crucial to building trust as had been expected. The emphasis appeared to be more on the willingness to learn.

There was unanimity throughtout all of the interviews in the team unifying behind the Sprint goal and this resonates with the shared mental model as mentioned in Section II D.

The literature refers to a stepwise calculative approach to building trust. Lewicki and Bunker describe how "achievement of trust at one level enables the development of trust at the next level [44]." This appears to be the approach taken in the company we worked with. New team members are firstly invited to pair with a more experienced developer and tasks are worked on jointly. After a period of time (which largely depends on the new team member) an individual task is assigned and the experienced developer steps back but is still available to mentor on an as needs basis. As this happens the new team member is becoming known to the team, familiarity and integrity are established. Once the new team member has become embedded in the team s/he begins to fully identify with the team, knowledge sharing and collaboration are enhanced and the teams' goals for the Sprint are met.

VII. LIMITATIONS

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

VIII. CONCLUSION AND FUTURE WORK

Although this study forms the first of what is intended to be part of 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 large successful software development multinational company based in the West of Ireland.

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 co-located Scrum teams will be enlarged.

REFERENCES

[1] T. O'Connell and O. Molloy, "The Antecedents and Feedback Loops Contributing to Trust in Agile Scrum Teams," in *SOFTENG 2018*, 2018, pp. 16–23.

[2] J. Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R. and Kern, "Manifesto for agile software development," 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] K. Schwaber, *Agile Project Management with Scrum*. 2004.

[5] S. Dorairaj, J. Noble, and P. Malik, "Understanding the Importance of Trust in Distributed Agile Projects: A Practical Perspective," in Understanding the Importance of Trust in Distributed Agile Projects: A Practical Perspective in: Sillitti A., Martin A., Wang X., Whitworth E. (eds) Agile Processes in Software Engineering and Extreme Programming. XP 2010. Lecture Notes in Business Information Processing, vol. 48, pp. 172–177. Springer, Berlin, Heidelberg.

[6] A. K. Mishra, "Organizational Responses to Crisis: The Centrality of Trust," in*: Kramer, R, Tyler,T. (eds)Trust in Organizations: Frontiers in Theory and Research*, 1996, pp. 261–287.

[7] M. Tschannen-Moran, "Collaboration and the need for trust," *ournal Educ. Adm.*, vol. 39, no. 4, pp. 308–331, 2001.

[8] S. Ghobadi, "What drives knowledge sharing in software development teams: A literature review and classification framework," *Inf. Manag.*, vol. 52, pp. 82–97, 2015.

[9] J. K. Butler, "Toward Understanding and Measuring Conditions of Trust: Evolution of a Condit Toward Understanding and Measuring Conditions of Trust: Evolution of a Conditions of Trust Inventory," *J. Manage.*, vol. 17, no. 3, pp. 643–663, 1991.

[10] D. E. Zand, "Trust and Managerial Problem Solving," *Adm. Sci. Q.*, pp. 229–239, 1972.

[11] D. Fields and J. S. Holste, "Trust and tacit knowledge sharing and use," *Artic. J. Knowl. Manag.*, 2010.

[12] D. H. McKnight and N. L. Chervany, "The meanings of trust.," 1996.

[13] J. Tabaka, *Collaboration Explained: Facilitation Skills for Software Project Leaders*, Pearson, 2006.

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

[15] K. Collier, *Agile analytics: A value-driven approach to business intelligence and data warehousing*, Addison-Wesley, 2012.

[16] S. Ghobadi and K. Mathiassen, "Perceived Barriers to Effective Knowledge Sharing in Agile Software Teams," *Information Syst.*, vol. 26, no. 2, pp. 95–125, 2016.

[17] S. Ryan and R. V. O'Connor, "Acquiring and Sharing tacit knowledge in software development teams: An empirical study," *Inf. Softw. Technol.*, vol. 55, no. 9, pp. 1614–1624, 2013.

[18] Y.-H. Chen, T.-P. Lin, and D. C. Yen, "How to facilitate inter-organizational knowledge sharing: The impact of trust," *Information Management*, vol 51, no 5, 2014.

[19] J. N. Cummings, "Work Groups, Structural Diversity, and Knowledge Sharing in a Global Organization," *Manage. Sci.*, vol. 50, no. 3, pp. 352–364, 2004.

[20] I. Nonaka, "A Dynamic Theory of Organizational Knowledge Creation," *Organ.. Sci.*, vol. 5, no. 1, pp. 14–37, 1994.

[21] J. C. Wyatt, "Management of explicit and tacit knowledge," *Journal of the Royal Society of Medicine*, vol 94, no. 1, pp. 6-9, 2001.

[22] T. Chau, F. Maurer, and G. Melnik, "Knowledge sharing: Agile methods vs. Tayloristic methods," in *Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE*, 2003, vol. 2003–Jan., pp. 302–307.

[23] 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*, pp. 60–65, 2009.

[24] K. H. Judy and I. Krumins-Beens, "Great Scrums Need Great Product Owners: Unbounded Collaboration and Collective Product Ownership," in *41st Hawaii International Conference on System Sciences*, pp. 1-10, 2008.

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

[26] T. Chau and F. Maurer, "Knowledge Sharing in Agile Software Teams," in *Logic vs. Approximation* 2004, pp. 173–183. Springer, Berlin, Heidelberg.

[27] M. C. Paulk, "Agile methodologies and process discipline," *Crosstalk*, 2002.

[28] P. Deemer, N. K. V. Hazarati, and G. Benefield, *The Distributed Scrum Guide*. 2013.

[29] O. McHugh, K. Conboy, and M. Lang, "Agile Practices: The Impact on Trust in Software Project Teams," *IEEE Softw.*, vol. 29, no. 3, pp. 71–76, 2012.

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

[31] A. Sofia, C. Marçal, B. Celso, C. De Freitas, F. S. Furtado Soares, and A. D. Belchior, "Mapping CMMI Project Management Process Areas to SCRUM Practices," in *31st Software Engineering Worklshop*, 2007, pp. 13–22.

[32] 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.

[33] D. Sandy Staples and J. Webster, "UKISJ Information Systems," *Authors; J. Compil. ©*, vol. 18, pp. 617–640, 2008.

[34] G. Simmel, *The sociology of Georg Simmel*. Free Press of Glencoe, 1950.

[35] A. Giddens, *The Consequences of Modernity Anthony Giddens*. 1990.

[36] J. B. Rotter, "A new scale for the measurement of interpersonal trust," *J. Pers.*, vol. 35, no. 4, pp. 651–665, Dec. 1967.

[37] M. Deutsch, "Conditions Affecting Cooperation Final Technical |Report for the Office of Naval Research Contract NONR- 285," 1957.

[38] B. Barber, *The Logic and Limits of Trust*. 1983.

[39] J.J. Gabarro, "The development of trust, influence and expectations," *Interpers. Behav. Commun. Underst. relationships*, pp. 290–303, 1978.

[40] J. K. Butler and R. S. Cantrell, "Communication Factors and Trust: An Exploratory Study," *Psychol. Rep.*, vol. 74, no. 1, pp. 33–34, Feb. 1994.

[41] N. Luhmann, "Familiarity, Confidence, Trust: Problems and Alternatives," in *Trust: Making and Breaking Cooperative Relations*, 2000, pp. 94–107.

[42] R. C. Mayer, J. H. Davis, F. D. Schoorman, and F. David Schoorman, "An Integrative Model of Organizational Trust," *Acad. Manag. Rev.*, vol. 20, no. 3, pp. 709–734, 1995.

[43] D. J. McAllister, "Affect-and Cognition-Based Trust Formations for Interpersonal Cooperation in

Organizations," *Artic. Acad. Manag. J.*, vol. 38, no. 1, pp. 24–59, 1995.

[44] 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.

[45] D. . Shapiro, B. H. Sheppard, and L. Cheraskin, "Business on a Handshake," *Negot. J.*, vol. 8, no. 4, pp. 365–377, 1992.

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

[47] K. Schwaber and J. Sutherland, "The Scrum Guide™ The Definitive Guide to Scrum: The Rules of the Game," 2017.

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

[49] 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.

[50] J. L. Guedes dos Santos *et al.*, "Methodological perspectives in the use of grounded theory in nursing and health research," *Esc. Anna Nery - Rev. Enferm.*, vol. 20, no. 3, 2016.

[51] R. E. De Vries, B. Van Den Hooff, and J. A. De Ridder, "Explaining Knowledge Sharing," *Communic. Res.*, vol. 33, no. 2, pp. 115–135, 2010.

[52] R. Johnson, A. Onwuegbuzie, and L. Turner, "Toward a Definition of Mixed Methods Research," *J. Mix. Methods Res.*, vol. 1, pp. 112–133, 2007.

[53] A. Bryman and E. Bell, *Business research methods*. 2015.

[54] S. Vanson, "What on earth are Ontology and Epistemology? - The Performance Solution," *What on earth are Ontology and Epistemology?*, 2014. [Online]. Available: https://theperformancesolution.com/earth-ontology-epistemology/. [Accessed: 14-Feb-2019].

[55] D. Ezzy, *Qualitative Analysis*. Routledge, 2013.

[56] K. Charmaz, *Constructing grounded theory*. 2014.

[57] P. Nardi, *Doing survey research : a guide to quantitative methods*. Pearson/Allyn & Bacon, 2003.

[58] P. Badke-Schaub, A. Neumann, K. Lauche, and S. Mohammed, "Mental models in design teams: a valid approach to performance in design collaboration?," *CoDesign*, vol. 3, no. 1, pp. 5–20, Mar. 2007.