Systemic Theory for Software Teams: A Perspective

Sergey Masyagin, Giancarlo Succi and Ananga Thapaliya
Innopolis University, Innopolis, Russia

Keywords: Systemic Theory, Psychological Systemic Theory, Sociological Systemic Theory.

Abstract: Complex problems involve a concerted effort by the software team and can absorb vital resources, but our understanding of how the software team forms and succeeds has been minimal. It is not possible to explain the relationships between team achievement and scale, concentration, and especially team expertise by confounding elements, such as age group, additional participation from other individuals who are not in the team, or by team structures. This generates a need to understand software teams using systemic theory. This position paper presents the efforts we have undertaken to study the impact of systemic factors on software development teams and how systemic theory can be used to understand software teams. Our approach looks at the effect of psychological and sociological systemic variables on software teams to identify a way to represent software teams as systems.

1 INTRODUCTION

Systemic theory is an interdisciplinary field of science that is associated with the concept of complex structures, whether physical, virtual, or entirely numerical (Sheridan, 2010). The application of systemic theory to the design and management of complex structures of various systems and their corresponding life cycles can provide an important focus for emerging approaches in software engineering systems.

A team is described as a position of authority, the results of aggregate work, the inspiration for open-minded debate, and critical thinking (Katzenbach and Smith, 2008). This also refers to software development teams where all developers collaborate to build an application, provide assistance, or perform other related tasks, and, indeed, the team's coordination and execution play a central role in the completion of tasks in a project (Mahdieh, 2015). A team has an distinct personality and ability to accomplish the mission, like an individual. Therefore, software development teams are also an imperative part of a software firm. These are the ones that define the company's success and lead the company towards its target. Inside the tech industry, the competitive landscape makes it more necessary for the organization to encourage its software engineers ts software engineers through various media such as promotions and bonuses (Zahra and Bogner, 2000). For this cycle to work it is crucial to understand the software teams, for which systemic theory comes into play.

In the field of software development, it was thought that recognizing systemic factors associated with software engineering was important (Weinberg, 1971). In any case, much of the research and analysis was concentrated largely on technical or process-related factors in subsequent years, while analysis that considered the organizational, sociological, or psychological variables of the systemic theory was uncommon (Perry et al., 1994). Even though the use and focus on agile methodologies have again shown the significance of individuals, organizations, and their network and collaboration over the last few years (James, 2002), (Cockburn, 2006) these issues can still not be considered in the conventional field of software development. A few computer scientists have the need to see how developers' human factors form the collective actions of software teams because the behavior of the team has a major effect on the progress effort and nature of the items obtained (Beranek et al., 2005), (Wellington et al., 2005). The current circumstances have opened up an entirely new line of research in software development, which is designed to understand these human components and their effects on the outcomes of the undertaking.

This paper considers two specific applications of systemic theory, namely, psychological systemic theory and sociological systemic theory. The goal is to assess their potential contribution to understanding the operations of software teams. The specific hypothesis is that these two applications of systemic theories and additional strategies from multiple systemic

Masyagin, S., Succi, G. and Thapaliya, A.
Systemic Theory for Software Teams: A Perspective.
DOI: 10.5220/0010476803560361
In *Proceedings of the 16th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2021)*, pages 356-361
ISBN: 978-989-758-508-1

approaches can explain how software teams operate and achieve their goals and the goals of their organizations.

2 SYSTEMIC THEORIES AND FACTORS

Software teams may succeed or fail for reasons that have nothing to do with the technology or the project that is being built, but because of external systemic factors. The systemic theory of psychology and sociology is an essential external factor that affects software teams and should not be ignored. The production of software has been designed to be a scholar and is recognized by cognitive and systemic processes of psychology and sociology (Feldt et al., 2010), (Khan et al., 2011), (Lenberg et al., 2014). In companies, software design takes place in our minds first, then in objects. We are people, and thus, as we experience the world through them, our behavior depends on psychological and sociological systemic variables. Software production in companies is, thus, fundamentally driven by such variables.

2.1 Psychological Systemic Theory

In the field of psychology, where it is called a psychological systemic theory, the systemic theory has been applied. To build a structure that works for all people, it is important to think of the interests, needs, wants, and practices of every person within it. At the stage where difficulties occur, these are due to breakdowns in basic interactions as opposed to one person's inadequacy. Based on our position (psychological factor affects the software teams and is necessary to understand them) and related works, we have divided the psychological factors into:

2.1.1 Effort Measurement

This is one of the important psychological variables that can help to understand the software team. The effort can be viewed as the work hours spent on completing their assigned tasks by the software team. The hours may be technical (design, debug, coding, and testing) or other forms of events, such as team meetings, seminars, and consultation. Therefore, several scholars (Canedo and Santos, 2019) clarify that in a method of understanding the software team, it is important to characterize what efforts would be considered. All development efforts should be included in the total hours reported and, thus, it is important to characterize which activities would be correlated with product improvement, both in terms of technical and non-technical efforts.

2.1.2 Team Size

Another important quality of a work-group is its size. It needs to be neither too big nor too small to be convincing. There is a balance between increased team competence and reduced participation and fulfillment of team members as they grow. A very small community may not have the variety of skills it requires to operate well. The optimal size depends somewhat upon the enthusiasm of the team. A data sharing or decision-making community may have to be larger than one for critical thinking. There is a trend in various associations to remember delegates for all councils from each potential team in the belief that this increases investment and profitability. There is also the view (Pendharkar and Rodger, 2007) that placing a delegate of a given team for each conceivably relevant division helps smooth the project's data stream and progress. In reality, coordination is usually decreased in bigger groups. As the size of the team develops, people feel less involved with the cycle, the distance will generally increase, and the dedication to the project will generally decrease.

Unfortunately, this rule cannot be regarded as a universal rule for the size of teams. The division of large teams into sub-teams can not help a team achieve its goals. For a small team to deal with the challenge and to welcome relevant participants to go to specific departments, one way to keep large numbers of people informed about a project is for a small team to handle the task. Then again, a smaller team may arrange to offer a larger group of colleagues information courses. Thus, for the reasons behind achieving group goals, it is best to deliberately and cautiously manage the process of rebuilding large teams into more modest groups.

Considering these aspects discussed above, psychological systemic theory helps to manage and understand the software teams and their size. Systemic theory suggests that the number of individuals who make up a group's membership and the interactions they have with each other has an exceptional effect on the sustainability of the group (Adams, 2012). Individuals should all have the option to apply their expertise and ability to the goals of the community to use the assets. If there is a chance to select our own group, our objectives and the strategies for achieving our goals should be defined. This will let us know the capabilities, such as the attributes of team members and their abilities. The least-sized team rule states that optimal team size is the one that consolidates a wide variety of viewpoints and evaluations but includes as few individuals as possible (Hare, 1992). The relation between team works and the necessary abilities to recognize differences and start recognizing tasks and planning training and so on must be assessed as consistently as possible.

Figure 1 shows the systemic way to manage and understand teams. This is a helpful method of weighing up the combination of 'responsibility' and 'individuals' capacities of a group.

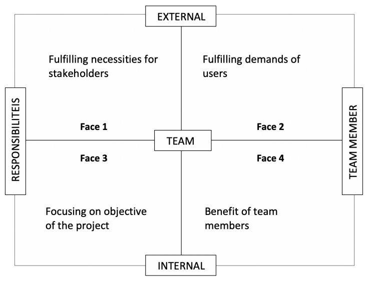
EXTERNAL
Fulfilling necessities for stakeholders
Fulfilling demands of users
RESPONSIBILITEIS
Face 1
Face 2
TEAM
Face 3
Face 4
TEAM MEMBER
Focusing on objective of the project
Benefit of team members
INTERNAL

Figure 1: Systemic Way of representing team [modified from (Lewis and Lawton, 1992)].

Face 1 shows that climate adjustment and the use of authoritative assets viably fulfill the criteria of the support of the community, and Face 2 clarifies that the concerns of clients or customers, whether within or outside the organization, are properly handled by individuals outside the group. Face 3 clarifies that it is necessary to use structures and approaches to complete objective arranged undertakings, and Face 4 clarifies operating in a way that makes people feel part of a group.

2.2 Sociological Systemic Theory

In sociological systemic theory, there are aspects related to software development, especially concerning the Internet, primarily with respect to the communication. Technology has had an immense impact on society and how individuals convey and share culture. The dynamics of social institutions and prevailing cultural configurations are emphasized by this theory (Parsons, 1977). The sociological structural theory is linked either to the internal surroundings of other social institutions or to external non-social settings. Furthermore, they vary in the way they relate to time: they are either put towards realizations in the future or compliance is involved in the present (Stichweh, 2011). Based on our position (sociological factor affects the software teams and is necessary to understand them) and related works, we have divided the sociological factors into:

2.2.1 Team Cohesion and Collaboration

The way of understanding team members is affected by cohesion between members of the software team. Many findings explain that the effect of teamwork and effectiveness among team members is positive. It is important to appoint colleagues in a way that encourages their cohesion (Kang et al., 2011). The productivity of the software development team (Bhardwaj and Rana, 2016) is affected emphatically by teamwork. The program manager should strive to maintain a workplace that promotes teamwork so that it is possible to achieve better productivity levels (Kang et al., 2011). Collaboration between the various positions associated with software engineering should be promoted in a software team, which provides a systematic way of understanding the software team (Clincy, 2003).

2.2.2 Communication

Another systemic way to understand software teams depend on communication activity. Therefore, communication activity is a factor in the feasible elements of social structures. Regardless of what assets are accessible inside a system, without communication operations, those assets will remain lethargic and no benefits will be provided to individuals. Communication's importance is expressed in the imaginary meanings of small meetings and networks that convey the significance of the association (Bonner, 1959; Hare, 1994; Freeman, 1992; Stogdill, 1959; Locke et al., 1981). Without some form of contact operation, effect, social support, collaboration, or information sharing will not occur.

Communication flexibility affects the way a software team works and how we can perceive them. Quick collaboration impacts the productivity of software teams (Yilmaz et al., 2016), (Nunamaker and Chen, 1989), (Wagner and Ruhe, 2018). Under cooperative conditions, ease of communication is essential for increasing the productivity of development(Lima et al., 2015). The effectiveness of communication is not limited to human facets. Components defined by the software company with the communication system and the workplace significantly affect all communication facilities and various variables previously explained such as cohesion and collaboration.

3 SOFTWARE TEAMS AS SYSTEM

The systemic theory approach to understanding software teams is discussed in this section. Considering all the systemic variables we discussed above, here we will represent teams as systems. This approach will motivate teams to understand the implications in which they operate. Team measures are considered in the approach, divided into three sections: problems as inputs, transformational processes, and solutions as outcomes. These highlights the numerous challenges and roles that a team and leaders need to deal with or supervise throughout a team's life.

3.1 Systemic Approach to Understand Software Teams

The systemic theory teaches us to think of teams as part of an organization or the big established network. Given the systemic theory, team leaders will discover the means to help the teamwork efficiently. A systemic theory asks: within and outside the team, what should be regulated, observed, or affected? At the same time, leaders need to understand the team, beginning to end, with regard to their company phases and cycles. This helps leaders to place a particular team-related problem in the environment to better understand it. The role of a leader is to acquire, schedule, and screen all of these different cycles. This seemingly complicated and cumbersome job is simpler to comprehend and handle when divided into its components. Figure 2 shows the systemic manner to represent and understand software teams.

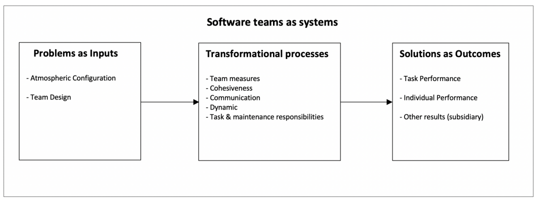

Figure 2: Systemic Approach to understand Software Teams [modified from (Schermerhorn et al., 1998)].

There are different models that also propose that it is possible to accept teamwork as a three-stage arrangement (Ingram et al., 1997), (Schermerhorn et al., 1998). Teams are seen as systems that take properties, such as time, personnel, expertise, issues (inputs), and change them into outputs, such as work, solutions, and completion, through transformative processes, for example, decision-making and various behavioral traits (Ingram et al., 1997). Consequently, team success is the overall talents, experience, and behaviors of team members (Gribas et al., 2017).

Table 1: Input related questions.

Q1)	What aims will the project satisfy?
Q2)	What assets will be accommodated in the software project? What others may be required? Where might they be able to come from?
Q3)	By what means will people dealing with the project be remunerated?
Q4)	What may they learn? What aptitudes would they be able to want to develop?
Q5)	What number of individuals will be expected to perform this project/task?
Q6)	What specialized aptitudes are required?
Q7)	What training and workshop openings are accessible?
Q8)	What roles should be fulfilled?
Q9)	Who may work in groups?

3.2 Application of Systemic Framework

Problems as Inputs: Inputs are variables that the management regulates and affects. This may be the team's direct administrator or the outcome of senior administration planning and process. Practically speaking, this means that the manner a team is assembled and will work is affected by the characteristics, mission, and philosophy of the association and its strategies and methods.

Inputs include 'atmospheric configuration', the setting in which the team operates, and 'team design', how the team is assembled, who is selected to operate in it, and why. The board will also influence how a team can operate by ensuring at the beginning that the team approach is in accordance with the association's vision and organizational direction and that it utilizes the favored work practices of the association; for example, physical or remote working.

At this point, two fundamental elements to consider are the communication atmosphere and team arrangement. Communication arrangement refers to the open guidelines for a work setting, which typically highlights how eager or unwilling people are to discuss problems or complaints and speak uninhibitedly. Team design refers to the roles played by teammates. Some teams have a clear order (the leader agrees on all decisions), while other groups disseminate choices to all participants (i.e., majority rule). The communication atmosphere and members' agreement determine how inputs should be used.

Transformational Processes: Transformational processes refer to the activities and tasks that help to transform inputs into outputs. They may have the greatest influence on effective teamwork as they in-

Table 2: Process related questions.

Q1)	What would you be able to do to fabricate a sense of belonging among the colleagues?
Q2)	By what method will the teams communicate?
Q3)	Do any guidelines need setting up?
Q4)	What strategies for decision-making are there?
Q5)	Will there be a group leader? In what manner will the individual be selected?
Q6)	What tasks should be done to finish the product?
Q7)	What upkeep practices does the team need to show to finish the work and to profit and get from the experience?
Q8)	Who will confirm that the various tasks are performed?
Q9)	Are there any structures and frameworks to review at the end?

clude team processes such as developing and maintaining cohesiveness and communication. They also involve task activities that get the work done and maintenance activities which support the development and smooth functioning of the team.

Common transformational processes include:

- Team Measures: A feeling of solidarity is made by expressing concrete objectives that are perceived and acknowledged by the individuals.
- Cohesiveness: This includes facilitating feelings of having a position, teamwork, accountability, and dedication to the team.
- Communication: This includes being transparent, specific, accessible, and legitimate.
- Dynamic: This involves ensuring that plans are developed, that everyone is clear about management, and that a climate of confidence is established.
- Task and Maintenance Responsibilities: These involve activities to guarantee that the task is efficiently completed, such as organizing, accepting techniques, and controls, for example. They often integrate practices that reduce risks to the process, such as inspecting and surveying inner systems and resolving disputes.

Solution as Outcomes: It refers to those results that meet hierarchical or individual goals or other specified criteria. Different partners, including the association itself and team members, and a number of other shareholders, may examine the success of outputs. Team outputs include the display of team activities and the results of a single team member.

Outputs can be analyzed in terms of task execution, how each team member performs and other (subsidiary) results.

- Task Performance: This may be determined by different criteria, such as the nature of traditional results or targets. A commodity (in this case a newsletter) and the time needed to accomplish the task are the requirements for this scenario.
- Performance of each Team Member: This incorporates individual fulfillment and self-awareness.
- Other Results: These refer to adaptable skills that can contribute to other teamwork in later tasks. They have knowledge of viable collaboration and work-related skills.

Table 3: Output related questions.

Q1)	Has the group finished the task it was given?
Q2)	Was the project completed in a given budget and time?
Q3)	What has the group gained from this experience?
Q4)	Should the group currently be separated or would it be able to go on to another project?
Q5)	What have the group members gained from the experience?
Q6)	Have individuals created adaptable cooperation and different abilities?
Q7)	Where can these abilities be utilized in the association?

4 CONCLUSION

We assume that the basis for the interpretation of multidisciplinary systems in systemic theory. When seeing multidisciplinary systems and their related problems, experts will benefit from the use of systemic theory as a focal point. Systemic theory and the associated systemic variables are important, enabling systems experts to understand software teams with ideas.

It is important to understate the actions of the employee and to maintain a proper atmosphere and contact within the teams. This even goes for software engineers in software firms. The knowledge of the software developer's needs and prerequisites assumes a significant part of having optimum performance from them to ultimately obtain a profit. Teamwork plays an important role in software projects and it depends on the human characteristics of team members. Systematic knowledge of the team helps to recognize elevated teams in firms and helps to reward and appreciate teamwork.

This said, there's not much to conclude, this early in our work. We hope that a systemic look in software teams would provide us with bits of information on the best way to represent, express, coordinate, and order software development processes, in strategies to exploit the strength of the team.

REFERENCES

Adams, K. M. (2012). Systems theory: a formal construct for understanding systems. *International Journal of System of Systems Engineering*, 3(3-4):209–224.

Beranek, G., Zuser, W., and Grechenig, T. (2005). Functional group roles in software engineering teams. In *Proceedings of the 2005 workshop on Human and social factors of software engineering*, pages 1–7.

Bhardwaj, M. and Rana, A. (2016). Key software metrics and its impact on each other for software development projects. *ACM SIGSOFT Software Engineering Notes*, 41(1):1–4.

Bonner, H. (1959). *Group dynamics; principles and applications*. New York: Ronald Press Company.

Canedo, E. D. and Santos, G. A. (2019). Factors affecting software development productivity: An empirical study. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, pages 307–316.

Clincy, V. A. (2003). Software development productivity and cycle time reduction. *Journal of Computing Sciences in Colleges*, 19(2):278.

Cockburn, A. (2006). *Agile software development: the cooperative game*. Pearson Education.

Feldt, R., Angelis, L., Torkar, R., and Samuelsson, M. (2010). Links between the personalities, views and attitudes of software engineers. *Information and Software Technology*, 52(6):611–624.

Freeman, L. C. (1992). The sociological concept of" group": An empirical test of two models. *American journal of sociology*, 98(1):152–166.

Gribas, J., Gershberg, Z., DiSanza, J. R., and Legge, N. J. (2017). in the study of communication. *Narrative, Identity, and Academic Community in Higher Education*.

Hare, A. P. (1992). *Groups, teams, and social interaction: Theories and applications*. Praeger Publishers.

Hare, A. P. (1994). Types of roles in small groups: A bit of history and a current perspective. *Small Group Research*, 25(3):433–448.

Ingram, H., Teare, R., Scheuing, E., and Armistead, C. (1997). A systems model of effective teamwork. *The TQM Magazine*.

James, A. (2002). Highsmith. agile software development ecosystems.

Kang, D., Jung, J., and Bae, D.-H. (2011). Constraint-based human resource allocation in software projects. *Software: Practice and Experience*, 41(5):551–577.

Katzenbach, J. R. and Smith, D. K. (2008). *The discipline of teams*. Harvard Business Press.

Khan, I. A., Brinkman, W.-P., and Hierons, R. M. (2011). Do moods affect programmers' debug performance? *Cognition, Technology & Work*, 13(4):245–258.

Lenberg, P., Feldt, R., and Wallgren, L.-G. (2014). Towards a behavioral software engineering. In *Proceedings of the 7th international workshop on cooperative and human aspects of software engineering*, pages 48–55.

Lewis, R. and Lawton, J. (1992). The four functions of organizations—where does the individual fit in? *Strategic Change*, 1(3):147–152.

Lima, A. M., Reis, R. Q., and Reis, C. A. L. (2015). Empirical evidence of factors influencing project context in distributed software projects. In *2015 IEEE/ACM 2nd International Workshop on Context for Software Development*, pages 6–7. IEEE.

Locke, E. A., Shaw, K. N., Saari, L. M., and Latham, G. P. (1981). Goal setting and task performance: 1969–1980. *Psychological bulletin*, 90(1):125.

Mahdieh, O. (2015). Interaction between communication and organizational conflict and its relationship with performance. *Interaction*, 1(2):6–12.

Nunamaker, J. F. and Chen, M. (1989). Software productivity: a framework of study and an approach to reusable components. In *Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences. Volume II: Software Track*, volume 2, pages 959–960. IEEE Computer Society.

Parsons, T. (1977). *Social systems and the evolution of action theory*. Free Press.

Pendharkar, P. C. and Rodger, J. A. (2007). An empirical study of the impact of team size on software development effort. *Information Technology and Management*, 8(4):253–262.

Perry, D. E., Staudenmayer, N. A., and Votta, L. G. (1994). People, organizations, and process improvement. *IEEE Software*, 11(4):36–45.

Schermerhorn, J. R., Hunt, J. G., and Osborn, R. N. (1998). *Basic organizational behavior*. J. Wiley.

Sheridan, T. B. (2010). The system perspective on human factors in aviation. In *Human factors in aviation*, pages 23–63. Elsevier.

Stichweh, R. (2011). Systems theory. *International Encyclopedia of Political Science. New York: Sage*.

Stogdill, R. M. (1959). Individual behavior and group achievement: A theory; the experimental evidence.

Wagner, S. and Ruhe, M. (2018). A systematic review of productivity factors in software development. *arXiv preprint arXiv:1801.06475*.

Weinberg, G. M. (1971). *The psychology of computer programming*, volume 29. Van Nostrand Reinhold New York.

Wellington, C. A., Briggs, T., and Girard, C. D. (2005). Examining team cohesion as an effect of software engineering methodology. In *Proceedings of the 2005 workshop on Human and social factors of software engineering*, pages 1–5.

Yilmaz, M., O'Connor, R. V., and Clarke, P. (2016). Effective social productivity measurements during software development—an empirical study. *International Journal of Software Engineering and Knowledge Engineering*, 26(03):457–490.

Zahra, S. A. and Bogner, W. C. (2000). Technology strategy and software new ventures' performance: Exploring the moderating effect of the competitive environment. *Journal of business venturing*, 15(2):135–173.