A Systemic Perspective on Software Engineering

Mirko Farina
Innopolis University
Innopolis, Russia
m.farina@innopolis.ru

Giancarlo Succi
Università di Bologna
Bologna, Italia
g.succi@unibo.it

Ananga Thapaliya
Innopolis University
Innopolis, Russia
a.thapaliya@innopolis.ru

Abstract

Systemic thinking is an interdisciplinary field of research that attempts to comprehend complex human (and non-human) structures by explaining their mutual interconnections from a holistic standpoint. This paper investigates whether systemic thinking can help understand how software teams work and operate. To do so, we carried out exploratory and qualitative work that involved: i. a literature review, ii. the formulation of an online survey (which was administered to a relatively large number of IT professionals across three different countries: Nepal, Luxembourg, and Russia), and iii. a series of semi-structured interviews conducted with experienced programmers and managers from different software development industries based in the Innopolis Special Economic Zone. Our findings confirm that systemic thinking is of paramount importance in explaining software development processes, especially when large teams are involved.

CCS Concepts: • **Systemic Thinking** → *Psychological Systemic Thinking*; *Sociological Systemic Thinking*; • **Teams** → *Software Development Teams.*

Keywords: Systemic Thinking, Software Engineering, Software Development, Software Teams

ACM Reference Format:
Mirko Farina, Giancarlo Succi, and Ananga Thapaliya. 2023. A Systemic Perspective on Software Engineering. In *Proceedings of Submission to TSE.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

This study investigates the relevance, significance, and potential applicability of systemic thinking to software development processes, especially in the context of complex, multifactorial projects, where a large number of people are involved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Submission to TSE, September 2021,
© 2023 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

In this study, we consider two perspectives or approaches to systemic thinking; namely, psychological systemic thinking and sociological systemic thinking. For an introduction to systemic thinking see section 2. For an analysis of the major differences between these two approaches please refer to 3.

Our working hypothesis is that the joint combination of these two approaches can help explain how software teams operate and achieve their goals, as well as the goals set by the organizations, firms, and institutions for which they may work.

We conceptualize, test, and verify our hypothesis along the following dimensions:

1. (For the benefit of those already not familiar with this discipline) what is systemic thinking?
2. What traits of systemic thinking are present (often unnoticed) in the activities and dynamics of individuals, teams, and organizations in software engineering?

We perform a literature review to analyze whether there are already aspects of systemic thinking covered by existing research, then we run an empirical study organized in two steps: a survey and an online interview [27, 76]. We conclude our work by presenting the evidence we gathered to support our hypothesis.

Our work offers original insights into the discipline of software engineering as it describes an application of systemic thinking in the context of software development. In addition, it reports about the reliability and validity of the survey and interview we carried out, applied to distinct samples of the relevant population; justifies the relation to the literature review for further use; and discusses the best representative construct of systemic thinking traits. Although various systemic traits that fall under the general label of 'systems thinking' have become quite popular, little is known about their efficacy in enhancing organizational effectiveness or productivity. With few exceptions, the relationship between the use of systems thinking and organizational performance remains the province of anecdotal evidence. In our contribution, we demonstrate why it is crucial to comprehend these traits in order to lay a solid foundation for the development and application of system dynamics and systemic thinking tools.

The characteristics of systemic thinking helped us to see how interconnected and interdependent individuals, groups, and organizations are, as well as how human behavior is usually too complicated to be well-defined or comprehended

from just one of these perspectives. Our findings can aid in directing and prioritizing future research. We believe that the industry will be more inclined to incorporate new findings and collaborate with researchers to better cater to systemic qualities in software development if the research community listens even more to the demands and experience of software development teams. Ultimately, our result can help software teams in software engineering organizations to increase the likelihood of successfully implementing change initiatives that result in a changed organizational behavior by applying systemic thinking.

The paper is organized as follows. Section 2 provides a short introduction to systemic thinking for readers not familiar with it. Section 3 discusses two different approaches to systemic thinking (psychological and sociological) and their related traits in detail. Section 4 reviews the methodological protocol we adopted. Section 5 describes the qualitative and the quantitative analyses we performed in this study 6 summarises the results of our study and puts them in context, highlighting their relevance to the software engineering community. Section 7 shows how our results support our hypothesis. Section 8 focuses on shortcomings and potential issues threatening the validity of our study. Eventually, Section 9 summarises what we achieved and outlines possible future research directions.

2 A short introduction on systemic thinking

Systemic thinking is an idea that has been around for a long time, with many Eastern philosophies emphasizing the importance of understanding the connections and relationships between different parts of nature. Systemic thinking was first proposed, in the early '40s, by Ludwig von Bertalanffy [69], an Austrian biologist known -among other things- for his important contributions to the laws of thermodynamics. However, in the early'50s, systemic thinking became very influential also in cybernetics, where it was applied and further developed by world-renowned cyberneticist Ross Ashby [3]. At the beginning of the '70s, systemic thinking became pervasive among other disciplines and psychologists (such as George Bateson [5] as well as philosophers (such as Ervin Laszlo [41] started using its principles as a conceptual palette for novel insightful investigations. More recently, systemic thinking has been successfully applied to organizational development as well as to management theory [56, 57]. Various fields such as systemic dynamics, systemic science, and systemic engineering have their own interpretations of systemic thinking, but all of them focus on understanding how different parts of a system are connected and how decisions made in one part of the system can impact the entire system.

One can therefore say that systemic thinking is associated with holistic thinking and with the concept of complex structures, whether physical, virtual, or entirely numerical [58]. It is thus an approach to studying systems 'from a system of systems point of view' [25]p.2. This means that systemic thinking typically attempts to formulate research frameworks and paradigms in which 'physical, technological, biological, economic, environmental, social, cognitive, and metaphysical systems can be observed, studied, thought, modeled, simulated, and intervened' [25] p.2.

In other words, systemic thinking draws from different disciplines (ranging from engineering, computer science, and cognitive science, to management, philosophy, psychology, and even biology) and attempts to provide a discipline-agnostic approach for dealing with complex and sophisticated problems [43]. Such an approach typically allows its users to understand the structure and properties of a given system in terms of the relationships of interdependence occurring among its different components [49].

As noted above, we believe that the application of systemic thinking to the design [11, 22] and management of complex structures of various systems and their corresponding life cycles [30] can provide important insights for better understanding emerging approaches in software engineering systems. For this reason, in this paper, we use systemic thinking as a conceptual palette to inform, individuate, specify and even explain and characterize the relations between team members, where teams are understood as systems.

In truth, systemic thinking found its way into software engineering a long time ago with the seminal study of Weinberg (1971) [71]; however, much of the work that followed such study centered largely and squarely on technical issues, often ignoring research that envisaged a significant contribution of organizational, sociological, or psychological traits [52] in software development. Nevertheless, the advent of agile methodologies [7] and [73] forced many researchers in the IT industry to reflect on and appreciate the increasing importance of individuals, organizations, and of their collaborative work [16, 33]. Yet, a deep understanding of this phenomenon hasn't been readily forthcoming and some of the issues underlying it remained mostly unexplored [9, 72].

3 Extending systemic thinking to psychological and sociological construct

Systemic thinking in software engineering is an approach that views the software development process as a complex and dynamic system composed of multiple interconnected parts [53]. It emphasizes the interdependence and interconnection of individuals, teams, and organizations, and how they influence and are influenced by one another [32].

Psychological and sociological systemic thinking are two key constructs of systemic thinking in software engineering that provide insight into how individuals and organizations behave and interact within the software development process [29]. These two constructs are chosen out of many others because they have been widely researched and applied

in the field of software engineering. For example, psychological systemic thinking, which focuses on the cognitive, emotional, and behavioral aspects of individuals, has been applied to understand the impact of cognitive load on software development performance [26]. Similarly, sociological systemic thinking, which focuses on the social, organizational, and cultural traits, has been applied to understand the impact of organizational structure on the adaptability of a firm [59]. Together, these two constructs provide a comprehensive understanding of the software development process and how to improve the performance of individuals, teams, and organizations.

Through a review of existing research in the field, this section examines the key aspects of systemic thinking in software engineering, including the role that social networks, power, and politics, as well as organizational culture, may plan in shaping the behavior and performance of individuals and groups in computer science organizations [21, 31]. Additionally, it also describes some of the most relevant psychological aspects of systemic thinking (such as team cognition, emotions, and motivation).

3.1 Psychological Systemic Thinking

Psychological systemic thinking examines the interactions and interdependencies between individuals, teams, and firms in software engineering and computer science from a psychological perspective. It focuses on understanding how the cognitive, emotional, and behavioral processes of individuals and groups influence and are influenced by the broader social and organizational context in which they operate [21].

One of the key features of psychological systems thinking in software engineering is the emphasis on understanding the relationships and connections between different parts of the system. For example, research has shown that individual developers' cognitive processes are closely linked to the overall performance and quality of software development projects [48]. Similarly, research has shown that team dynamics and communication patterns can have a significant impact on the overall performance and quality of software development projects [75].

Another important feature of psychological systems thinking in software engineering is the emphasis it places on understanding how individuals and teams make decisions within the context of software development projects. Research has shown that individuals and teams often use heuristics and biases when making decisions [64], and that these heuristics and biases can have a significant impact on the overall performance and quality of software development projects [48].

Overall, psychological systems thinking in software engineering aims to understand how individuals and teams think and make decisions within the context of software development projects. The field is based on the idea that software development is a complex, dynamic, and adaptive system, and that understanding how individuals and teams think and make decisions within this system is crucial for improving the overall performance and quality of software development projects.

In light of the above, we can therefore say that psychological systemic thinking can help us understand some of the dynamics that characterize the performance of many software teams. Having established this crucial point, we can now turn to analyze the kind of contribution that sociological systemic thinking can give to software engineering.

3.2 Sociological Systemic Thinking

Sociological systemic thinking typically deals with aspects related to software development in the context of technological/culturally scaffolded artifacts and -primarily- with respect to communication [63]. Thus, sociological systemic thinking highlights the importance of understanding the dynamics underlying the production of cultural artifacts, especially in the context of social/collaborative institutions [51]. The conceptual origins of this theory can be traced back to the seminal works of philosophers (such as Herbert Spencer [50]) and social scientists (such as Émile Durkheim [1]). The field is based on the idea that software development is a socially-constructed activity, and that understanding how social traits and social structures shape the development and maintenance of software systems is crucial for improving the overall performance and quality of software development projects.

One of the key features of sociological systems thinking in software engineering is the emphasis on understanding the social traits that shape the development and maintenance of software systems. For example, research has shown that the organizational culture and structure of software development organizations can have a significant impact on the overall performance and quality of software development projects [19]. Similarly, research has shown that the social dynamics and communication patterns within software development teams can have a significant impact on the overall performance and quality of software development projects [75].

Another important feature of sociological systemic thinking in software engineering is the emphasis it places on understanding how social structures shape the development and maintenance of software systems. Research has also shown that the dominant norms, values, and practices of software development communities can have a significant impact on the overall performance and quality of software development projects [10].

Having discussed how psychological and sociological systemic thinking can influence the practice of software development and therefore be beneficial to software engineering teams, we next go on to present our research method, which -as the reader may recall- is devised to specifically test and eventually corroborate our working hypothesis; the idea that

systemic thinking can explain and help us comprehend how software teams operate and achieve their goals, as well as the goals set by the organizations, firms, and institutions for which they may work.

4 Methodology

To gain a deeper understanding of our working hypothesis (how psychological and sociological systemic thinking can help explain how software teams operate and achieve their goals as well as the goals set by software firms) the study employed an open questionnaire as the primary research tool. This method is considered appropriate for exploratory research. However, designing a questionnaire for a study on interaction and communication can be challenging, as the way questions are framed and presented can affect the responses. To minimize bias, the questionnaire we developed followed established guidelines and the best norms in the disciplines (as described, for instance in [37, 47]).

The survey was developed in two steps. We first established the overall goals of the research using a GQM. We then created a test version of the questions. Then, two rounds of the pilot interview were conducted to test the questions. The wording of the questions was adjusted based on feedback from the two pilot interviews, and a final version of the questionnaire was created. Careful attention was paid to best practices in the field, taking into account the issues that can arise with online interviews [6, 44].

This strategy of organizing a study based on a survey followed by an interview promises to offer deeper insights than a purely quantitative approach based on surveys [17, 39, 62]. In addition, we must note that such an approach has been already undertaken in other, high profile works both in computer science and in psychology [46, 54, 55].

Moreover, pre-validating questionnaires via pilot interviews is a recommended procedure by many researchers [15, 68]. This is because pilot studies can be used as a preparation of a full-scale study [23], being useful in spotting early flaws, or limitations at an early stage, thus allowing timely corrections that would not be feasible or possible at a later stage [38].

In this section, we thus detail the topics, the interview protocol, and the methods we used in our research.

4.1 The Survey

The survey we developed in this study was designed to provide answers for our working hypothesis; that psychological and sociological systemic thinking can help us understand how developers' teams work and operate.

The survey consisted of a series of questions (a combination of multiple choices and numerical answers), which were asked to test out whether the participants involved would agree or disagree with the idea that certain systemic traits may constitutively affect their performances. The survey subsequently attempted to pinpoint and determine (by using the Likert scale [2]), the respective importance of such traits. The survey lasted no more than 30 minutes (with an average actual duration of 25 minutes).

We used a Goal Question Metric (GQM) approach to develop the survey and to orient and define its research objectives/questions [4, 24, 66, 67]. First, we established our goal; that is, the information we wished to obtain. We note that this goal applied to both the survey and the interviews (more on which below).

Having defined our goal, we then focused on producing first drafts of our questions, which were then administered and tested in two independent rounds of pilot interviews; one for the survey and one for the interviews.

We refined and updated our initial questions based on the results of the pilot interviews. In particular, we used a mix of both positive and negative frames to avoid the emergence of biases [34], and to ensure that the wording of each question was as balanced as possible [18]. We subsequently organized and structured our questions, ensuring a logical flow and a clear progression from simple to complex ones as recommended in the literature [35, 45].

We then prepared the final version of the survey and the final version of the questionnaire, which we used for the semi-structured interviews (more on this in section 3.3 below), paying particular attention to avoid ambiguous and double-barreled questions [34].

The GQM we used is summarised below.

1. ***Goal***: Determining whether systemic thinking can explain the work of individuals and teams in software engineering.
2. ***Question***: How can the application of systemic thinking help us understand how software teams work and operate in their daily practice?
3. ***Metrics***: Use of open-ended and close-ended questions with the adoption of Likert scales, scores - ranging from No (0) to Yes (1), disagree or agree (1,2,3,4,5), Non-compliance or Compliance (1,2,3,4,5).

The full questionnaire can be found online at URL: https://github.com/donrast41/Systemic-Thinking-Questions. As mentioned, the questions contained in our questionnaire for the interview and the survey have been taken and modified from [40]. They have been also inspired, as noted above, by discussions with experts in the field of systemic thinking.

To make sure that the responses obtained from the questionnaires would help us answer our research questions the questionnaires were produced following the best practices in our discipline [8, 12]. Specifically, the questions we formulated were reviewed by two experts in the field: (a) one, who carried out similar research in the field of system thinking for

STEM education, and (b) another, who did similar research with similar questionnaires in the field of aerospace engineering. Insights gathered from these two experts helped us in shaping our questionnaires.

After discussing the questionnaires with the experts above mentioned we also checked whether the questionnaires were consistent and/or aligned with similar ones, that had been developed for similar purposes, within our discipline. We found they were, especially with the work of Van Solingen and Berghout (2001) [67].

The professionals participating in the survey were selected with the intent to bring a realistic and practical viewpoint on the performance indicators used in their working environment. They were given the opportunity to fill in the survey online for approximately one month. All of the developers gave their permission for the author to use and reveal the information they provided when undergoing the survey. The survey was answered by 50 software developers working for different tech companies in Russia, Nepal, and Luxembourg, as noted above. Despite the fact that it was not mandatory, most of the developers provided information about their experience in the field. Specifically, 12 developers said they have been working in the field for more than eleven years, 24 developers said their working experience ranged between five and nine years, and -finally- 14 developers confirmed that their working experience in the field was less than five years. In addition, additional data about their background was recovered. 17 of the respondents were students who recently were recently graduated, 16 were undergraduate students, 13 were master students, and 4 were Ph.D. students.

4.2 Interview

We started interviewing the participants in our experiment based on the modified questions we got from [40]. The questions were modified based on the pilot interviews. The interviews were useful because they allowed us to flesh out some of the details behind the survey (such as trends for process usage, decision-making, and assessment of the effect of systemic thinking in software teams).

Interviews were conducted:

- to elicit feedback on the definition of systemic thinking;
- to validate the emphasis placed on psychological and sociological systemic thinking as critical enablers of systems theory, and
- to provide feedback on the adequacy and relevance of the work we reviewed in section 3.

In addition to providing a clear measure for the research's direction, which allows our findings to be potentially replicated [20], we also wanted the interviews to be adaptive [61]. So, we decided to perform semi-structured interviews [61, 74].

Table 1. Descriptive summary of the participants that took part in the interviews

Type of firm	# of subjects	Roles & Experience
Software Development (E-commerce related)	2	Software developer - 11 Years Product Manger - 4 Years
Software Development (Self-driving cars)	2	Data Scientist - 6 Years Project Manager - 2 Years
Software Development (Blockchain related)	1	UI Engineer - 8 Years
Consultant	2	Project Manger - 8 Years Software Developer - 3 Years
Software Development (Human-robots related)	1	Software Developer - 13 Years
Consultant	2	Software Developer - 2 Years QA Engineer - 6 Years
Software Development (E-commerce related)	1	Software Developer - 5 Years

The first part of the interview gathered descriptive data to complement the results we obtained from the survey. The latter portion of the interview concentrated, specifically, on systemic thinking. Participants were asked to define systemic thinking. If their definition lacked any of the important components of the definition of systemic thinking, a brief discussion ensued to inform the interviewee about the standard definition of systemic thinking used in this research (please refer to Section 1). At that point, the interviewee was asked to rate his or her team's perceived usage of or reliance on systemic thinking.

4.3 Participants Interviewed

We interviewed a total of 11 expert Russian programmers and managers, of which four women and seven men, aged between 25 and 45 years. The subjects worked for seven tech companies all situated in Innopolis, a newly established technological city near Kazan, in the Volga Valley. Table 1 gives information about the people interviewed together with their roles and experiences.

5 Analysis

We used basic statistics using ordinal scales and coding procedures to analyze the responses we gathered from the questionnaires. We note here that our research is mostly observational. We also note that finding subjects willing to participate in a study like this is extremely difficult, and it is even more difficult because the emphasis of our work is on software managers and engineers involved in complex and large-scale projects.

The data analysis approach was based on a conceptual methodology designed by [13, 28, 70]. As this methodology suggests, first we proofread the transcribed interviews to get an overall sense of what they are about. This procedure was repetitive as it involved the examination of data and specific transcript components again and again. This procedure though enabled the acquisition of novel data which in turn influenced the overall content translation, and thus the interpretation of the specific components [42].

Table 2. Some of the most commonly occurring codes and phrases

Some of the most commonly occurring codes and phrases
Collaboration and communication within internal and external teams enable sociological systemic thinking.
Identifying the social and psychological components of the software development team enables systemic thinking.
Individuals in the software development team with different levels of understanding of systemic thinking promotes system thinking.
Team leaders involve their team members in decision making promotes system thinking.
Many systemic thinkers on a single software team demotes systemic thinking.
More restrictive size of the team or a bigger number of team doesn't concern with enabling or disabling systemic thinking.
The blame game between team members during any mistakes within the project is not a part of systemic thinking.
Team members who cannot cope with changes happening in the firm don't have the quality of systemic thinking.

After the proofreading, we looked through the interview transcripts for introductory codes. A code defines a part of the information (contextual or residual) that tends to be important to the research query [68]. To acquire such codes we went through the transcripts word by word. We structured details specific to each code by coding intriguing characteristics of the data in an organized way across the datasets. The details were structured to organize the codes into patterns and dissect the relations between them to increase the level of accuracy. This process aimed at individuating a common relevant conceptual pattern found across all the data [28]. Crucially, the pattern captures a significant aspect of the data relating to the research problem.

Finally, we revisited the data and allocated extracts that characterized the main pattern and divided it into categories and sub-categories as shown in 3. Below, the paper describes the brief results of the analysis.

Table 3. Identified categories and subcategories (systemic traits)

Categories	**Sub Categories** (systemic traits)
1) Individual	1.1) Focus of attention 1.2) Interconnection 1.3) Adaptability
2) Software Team	2.1) Communication 2.2) Project Complexity 2.3) Team Leader
3) Software Firm	3.1) Administrative Change 3.2) Relationship with Clients 3.3) Work Environment

6 Results

This section provides a detailed description of the results obtained from the survey and interview study on systemic thinking in software development teams and organizations.

6.1 Survey

According to the online survey, many of the psychological and sociological systemic thinking traits identified by researchers as being characteristic of systemic thinking in Section **??**, are not taken into account by software development firms and teams. These variables though can have a positive effect on the success of software development teams and can assist software companies in better understanding their software teams. It is therefore important for businesses to make sure they understand how these variables affect their developers and improve their performance.

8 of the 50 people in the survey referred to technical aspects of engineering a system: e.g., management, technical depth of teams, producing a product, or finding interfaces. The remaining people referred to communication, team interactions, and social leadership. This highlights the fact that systemic thinking is deeply ingrained in the daily activities of software engineers and therefore demonstrates the importance of psychological and sociological traits in developers' teams.

6.2 Interviews

The first step, as noted above, in analyzing the data we gathered through semi-structured interviews was reviewing the transcripts and performing text analysis. Commonly occurring phrases and ideas were tagged, organized, and combined when appropriate, resulting in a set of commonly cited codes as shown in Table 2. After a set of unique codes was identified (open coding) the codes were reorganized into categories (axial coding), grouping similar code constructs and providing greater explanatory power. Axial coding resulted in three main categories: individual, software team, and software firms as shown in 3. These categories were further developed into subcategories as informed by the critical constructs identified in the literature review discussed above.

The resulting codes are categorized in Figure 1 by the number of times they were mentioned in our experimentation. The most commonly cited codes are associated with the individual and team characteristics that are thought to facilitate systemic thinking. In this context, it is worth noting that many of the most commonly stated codes have to do with communication, teamwork, and social leadership. This balance underscores that systems thinking is a social phenomenon that is heavily impacted by group dynamics and individual personalities.

All in all, the subjects we interviewed acknowledged that psychological and sociological systemic traits should be considered in all software engineering practices and that an improved understanding of these systemic traits would benefit all the roles in any given team.

6.3 Summary

We found that different psychological and sociological traits identified in our literature reviews as characteristic of systemic thinking in Section **??**, are part of the usual operations of software development individuals, teams, and companies. These traits, identified in Section **??** with coding, trigger a

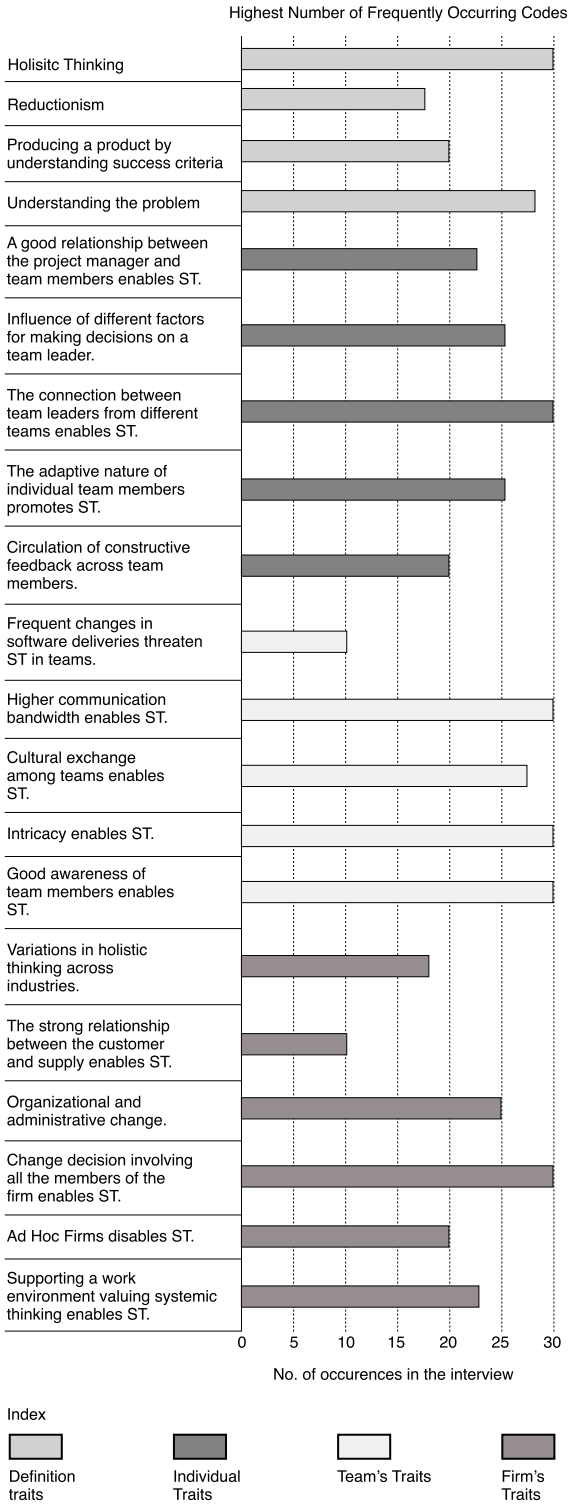

Figure 1. Systemic traits [codes] identified during empirical investigation

holistic approach to understanding the problem and achieving success in their work and the project.

Finally, most of the companies and team members were not aware that they were using systemic thinking in their daily projects and even mentioned that systemic thinking is not practiced in their company, but still, they mentioned the importance of psychological and sociological traits. (e.g. communication and collaboration), which shows that they are aware of its relevance at least at a very basic level.

In addition, three descriptive categories of use of modes of systemic thinking in software development emerged from the study of the interview transcripts, which can be roughly associated with individuals, teams, and corporations (Table 3). The categories are not mutually exclusive, and multiple viewpoints on the same occurrence may be included in them. The categories are described in further depth in the following sections, which are loosely organized based on the problem description, context, and identified patterns. These patterns are based on the codes we identified as listed in Figure 1.

7 Discussion

Our findings support the claim from section 1, that our work provides an original contribution to the discipline by describing the understanding and applications of systemic thinking in software development teams and organizations. The survey and interview used in the study have proven to be reliable and valid, and the results were obtained from distinct samples of the relevant population.

The findings were also related to the literature review from section 3, providing a strong foundation for further use and applications. Our results indicate that understanding and applying systemic thinking traits, such as the focus of attention, interconnection, adaptability, communication, project complexity, team leadership, administrative change, relationship with clients, and work environment, is essential for enhancing organizational effectiveness and productivity in software development.

Furthermore, our findings indicate that the relationship between the use of systems thinking and organizational performance remains the province of anecdotes rather than the identification and application of systemic traits [60]. The results of this study can help software development teams to increase the likelihood of successfully implementing change initiatives that result in a changed organizational behavior by applying systemic thinking [32]. Thus, the findings of this study support the claim that our work provides an original contribution to the discipline by describing the understanding and application of systemic thinking in software development teams and organizations.

In terms of individual traits, the findings suggest that focus of attention, interconnection, and adaptability are important for understanding how individual developers think and make decisions within the context of software development

projects. These findings align with the literature on psychological systems thinking, which emphasizes the importance of understanding how individual developers' cognitive processes are connected to the overall performance and quality of software development projects [48].

In terms of software team traits, the findings suggest that communication, project complexity, and team leadership are important for understanding how software development teams work together and make decisions. These findings align with the literature on sociological systems thinking, which emphasizes the importance of understanding how social dynamics and communication patterns of software development teams can impact the overall performance and quality of software development projects [75].

In terms of software firm traits, our findings suggest that administrative change, relationships with clients, and work environment are important for understanding how software development firms operate and make decisions. These findings align with the literature on sociological systems thinking, which emphasizes the importance of understanding how organizational culture and structure, as well as norms, values, and practices of software development communities, can impact the overall performance and quality of software development projects [10, 19]. The results explain the presence of variations in holistic cognitive expectations across industries and a consistent positive connection between systemic thinking and task execution. The interview transcripts captured this dimension through different statements such as "the team leader reviews and consider comments from the team while taking decisions," "team leaders first assess the project in general terms, then analyze it in small details" which then determines team members' actions.

Overall, our findings suggest that systemic thinking, both psychological and sociological, is essential to understand the traits that impact the performance and quality of software development projects. The themes and sub-themes found in our study align with the literature on systems thinking in software engineering and provide insight into the complex relationships between individuals, teams, and firms in software development projects.

8 Limitations and Threats to Validity

There are a series of potential limitations underlying our research. These can be summarised in two main categories.

Those concerning external validity and those concerning internal validity [65]. External validity is about whether the results obtained can be reliably generalized. Internal validity is achieved when a study eliminates extraneous variables as contributing to the phenomenon under observation.

As part of the former category (limitations affecting external validity), we can mention the relatively limited number of people tested in this study. Naturally, one may object that the quality of the quantitative data we gathered is partly affected by the limited sample size. Furthermore, since we used a qualitative research approach, generalizing our findings can be rather difficult. However, we could respond to this important worry by pointing out that we tested 66 [interview + survey] people overall in this study, which is not that little) and that we adopted two different types of testing techniques (online survey and interviews). Hence, we could respond that the data we gathered is fairly representative and reliable. In other words, while only a limited number of people were tested, the information gathered was diverse and representative.

As part of the latter category (limitations affecting internal validity), a skeptical reader may wonder about the potential presence of some bias in our research. Although it is virtually impossible to prevent biases, we adopted a process called observed triangulation to mitigate the occurrence of the experimenter-expectancy effect. For every interview, we held there was always an external person watching over the interviewer. This method was used to increase the credibility and validity of our research findings. In addition, we also carefully documented all interviews, transcribing the information, and employing a standardized data analysis process.

Certainly, other biases may apply to our research findings (such as the confirmation bias). Confirmation bias involves, as the name suggests, only seeking information to support the information that was sought. Indeed, we started this study by embracing systemic thinking and indeed we looked for instances of this theory in the practice of software engineers. However, to minimize this type of bias we thoroughly and carefully evaluated the statements and impressions received from participants in our study. So, we certainly attempted to mitigate its occurrence. Another bias potentially affecting our research is the so-called cultural bias. As noted above, all respondents were from just three countries (Nepal, Russia, and Luxembourg). However, here we note that the three countries we selected for our investigation are quite different culturally, and -in any case- in posing our questions we tried to be aware of specific cultural norms and assumptions.

Finally, our work awaits replication. To corroborate and extend our findings and preliminary conclusions, further analysis is required. This could involve clustered observational studies and/or further inferential statistics [14, 36, 77].

9 Conclusion

In this paper, we argued that systemic thinking can be taken as a framework to explain the operations of software teams. In particular, we claimed, that a series of traits (psychological and sociological) underlying and characterizing systemic thinking is crucially important in understanding how firms and software developers operate and achieve their goals as well as in comprehending and formulating potential remedy

strategies that can boost, improve or enhance their daily practices.

More specifically, we noticed that from a systemic point of view, it is crucially important to not understate the actions of the employee and to maintain an inclusive atmosphere, which is conducive to learning and collaboration. In addition, we saw that team leadership, flexibility, adaptability, and relationships with clients are crucially important traits for the development of successful software development teams. We also noted that systemic thinking may help better appreciate, hence rewarding, teamwork.

Yet, we acknowledged that this research faces some important limitations, mostly due to the fact that it is in its early stages. Nevertheless, we hope that the field - in welcoming our findings- will start implementing a systemic view more systematically and that, ultimately, many software development teams will embrace a systemic approach, because -as we showed in this paper- it can be beneficial to represent, express, coordinate, and understand how teams work and operate.

References

[1] Jeffrey C Alexander. 1990. *Durkheimian sociology: cultural studies*. Cambridge University Press.

[2] I Elaine Allen and Christopher A Seaman. 2007. Likert scales and data analyses. *Quality progress* 40, 7 (2007), 64–65.

[3] W Ross Ashby. 1961. *An introduction to cybernetics*. Chapman & Hall Ltd.

[4] Victor R Basili, Gianluigi Caldiera, and H Dieter Rombach. 1994. Goal, question metric paradigm. Encyclopedia of Software Engineering, vol. 1.

[5] Gregory Bateson. 1979. *Mind and nature: A necessary unity*. Vol. 255. Bantam Books New York.

[6] Paul C Beatty and Gordon B Willis. 2007. Research synthesis: The practice of cognitive interviewing. *Public opinion quarterly* 71, 2 (2007), 287–311.

[7] Kent Beck. 2000. *Extreme programming explained: embrace change*. addison-wesley professional.

[8] Peter M Bentler, Douglas N Jackson, and Samuel Messick. 1971. Identification of content and style: A two-dimensional interpretation of acquiescence. *Psychological Bulletin* 76, 3 (1971), 186.

[9] Georgine Beranek, Wolfgang Zuser, and Thomas Grechenig. 2005. Functional group roles in software engineering teams. In *Proceedings of the 2005 workshop on Human and social factors of software engineering*. 1–7.

[10] Finn Olav Bjørnson and Torgeir Dingsøyr. 2008. Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used. *Information and Software Technology* 50, 11 (2008), 1055–1068.

[11] John Boardman and Brian Sauser. 2008. *Systems thinking: Coping with 21st century problems*. CRC Press.

[12] Norman M Bradburn and Seymour Sudman. 2004. The current status of questionnaire research. *Measurement errors in surveys* (2004), 27–40.

[13] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101.

[14] Jeff Carver, John VanVoorhis, and Victor Basili. 2004. Understanding the impact of assumptions on experimental validity. In *Proceedings. 2004 International Symposium on Empirical Software Engineering, 2004. ISESE'04.* IEEE, 251–260.

[15] Milagros Castillo-Montoya. 2016. Preparing for interview research: The interview protocol refinement framework. *The qualitative report* 21, 5 (2016), 811–831.

[16] Alistair Cockburn. 2006. *Agile software development: the cooperative game*. Pearson Education.

[17] Thomas D Cook and Charles S Reichardt. 1979. *Qualitative and quantitative methods in evaluation research*. Vol. 1. Sage publications Beverly Hills, CA.

[18] John W Creswell and Cheryl N Poth. 2016. *Qualitative inquiry and research design: Choosing among five approaches*. Sage publications.

[19] Bill Curtis, Herb Krasner, and Neil Iscoe. 1988. A field study of the software design process for large systems. *Commun. ACM* 31, 11 (1988), 1268–1287.

[20] David De Vaus. 2001. *Research design in social research*. Sage.

[21] James K Doyle. 1997. The cognitive psychology of systems thinking. *System Dynamics Review: The Journal of the System Dynamics Society* 13, 3 (1997), 253–265.

[22] R Edson. 2008. Systems Thinking. Applied. A Primer. Arlington, VA: Analytic Services.

[23] Christian Erzberger, Udo Kelle, A Tashakkori, and C Teddlie. 2003. Handbook of mixed methods in social & behavioral research. *Handbook of Mixed Methods in Social and Behavioral Research* (2003), 457–90.

[24] Alfonso Fuggetta, Luigi Lavazza, Sandro Morasca, Stefano Cinti, Giandomenico Oldano, and Elena Orazi. 1998. Applying GQM in an industrial software factory. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 7, 4 (1998), 411–448.

[25] Luciano Gallón. 2020. Systemic Thinking. *Quality Education* (2020), 830–840.

[26] Lucian Gonçales, Kleinner Farias, Bruno da Silva, and Jonathan Fessler. 2019. Measuring the cognitive load of software developers: A systematic mapping study. In *2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*. IEEE, 42–52.

[27] J. Green and N. Thorogood. 2018. *Qualitative Methods for Health Research*. SAGE Publications.

[28] Christina Hellström. 2001. Affecting the future: chronic pain and perceived agency in a clinical setting. *Time & Society* 10, 1 (2001), 77–92.

[29] Patrick T Hester, Kevin MacG Adams, Patrick T Hester, and Kevin MacG Adams. 2014. *The why of systemic thinking*. Springer.

[30] Duane W Hybertson. 2016. *Model-oriented systems engineering science: a unifying framework for traditional and complex systems*. CRC Press.

[31] Michael C Jackson. 2010. Reflections on the development and contribution of critical systems thinking and practice. *Systems Research and Behavioral Science: The Official Journal of the International Federation for Systems Research* 27, 2 (2010), 133–139.

[32] Michael C Jackson. 2016. *Systems thinking: Creative holism for managers*. John Wiley & Sons, Inc.

[33] A James. 2002. Highsmith. Agile Software Development Ecosystems. (2002).

[34] Ng Chirk Jenn. 2006. Designing a questionnaire. *Malaysian family physician: the official journal of the Academy of Family Physicians of Malaysia* 1, 1 (2006), 32.

[35] Ng Chirk Jenn. 2006. Designing A Questionnaire. *Malaysian family physician : the official journal of the Academy of Family Physicians of Malaysia* 1, 1 (04 2006), 32–35.

[36] Natalia Juristo and Ana M Moreno. 2013. *Basics of software engineering experimentation*. Springer Science & Business Media.

[37] Barbara A Kitchenham, Shari Lawrence Pfleeger, Lesley M Pickard, Peter W Jones, David C. Hoaglin, Khaled El Emam, and Jarrett Rosenberg. 2002. Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on software engineering* 28, 8 (2002), 721–734.

[38] Steinar Kvale and S Brinkmann. 2007. Introduction to interview research. *Doing interviews* (2007), 2–11.

[39] M Lakshman, Leena Sinha, Moumita Biswas, Maryann Charles, and NK Arora. 2000. Quantitative vs qualitative research methods. *The Indian Journal of Pediatrics* 67, 5 (2000), 369–377.

[40] Caroline Marie Lamb. 2009. *Collaborative Systems Thinking: An exploration of the mechanisms enabling team systems thinking.* Ph.D. Dissertation. Massachusetts Institute of Technology, Department of Aeronautics and ....

[41] Ervin Lasklo. 1972. *Introduction to systems phylosophy: Toward a new paradigm of contemporary thought.* Gordon & Breach.

[42] Per Lenberg, Robert Feldt, and Lars-Göran Wallgren. 2014. Towards a behavioral software engineering. In *Proceedings of the 7th international workshop on cooperative and human aspects of software engineering*. 48–55.

[43] Hod Lipson. 2007. Principles of modularity, regularity, and hierarchy for scalable systems. *Journal of Biological Physics and Chemistry* 7, 4 (2007), 125.

[44] Peter Lynn and Olena Kaminska. 2012. Factors affecting measurement error in mobile phone interviews. In *Telephone Surveys in Europe: Research and Practice*. Springer, 211–228.

[45] Peter V. Marsden and James D. Wright (Eds.). 2010. *Handbook of Survey Research. Second Edition.* Emerald Group Publishing, Bingley, UK.

[46] Courtney A McKim. 2017. The value of mixed methods research: A mixed methods study. *Journal of Mixed Methods Research* 11, 2 (2017), 202–222.

[47] S McLafferty. 2016. Conducting questionnaire surveys. *Key methods in geography* 3 (2016), 129–142.

[48] Rahul Mohanani, Iflaah Salman, Burak Turhan, Pilar Rodríguez, and Paul Ralph. 2018. Cognitive biases in software engineering: a systematic mapping study. *IEEE Transactions on Software Engineering* 46, 12 (2018), 1318–1339.

[49] Howard T Odum. 1994. *Ecological and general systems: an introduction to systems ecology.* Univ. Press of Colorado.

[50] John Offer. 2010. *Herbert Spencer and social theory.* Springer.

[51] Talcott Parsons. 1977. *Social systems and the evolution of action theory.* Free Press.

[52] Dewayne E Perry, Nancy A. Staudenmayer, and Lawrence G Votta. 1994. People, organizations, and process improvement. *IEEE Software* 11, 4 (1994), 36–45.

[53] Magnus Ramage and Karen Shipp. 2009. *Systems thinkers.* Springer.

[54] Paula Reams and Darla Twale. 2008. The promise of mixed methods: Discovering conflicting realities in the data. *International Journal of Research & Method in Education* 31, 2 (2008), 133–142.

[55] Wendy L Richman, Sara Kiesler, Suzanne Weisband, and Fritz Drasgow. 1999. A meta-analytic study of social desirability distortion in computer-administered questionnaires, traditional questionnaires, and interviews. *Journal of applied psychology* 84, 5 (1999), 754.

[56] Edgar H Schein. 2015. Organizational psychology then and now: Some observations. *Annu. Rev. Organ. Psychol. Organ. Behav.* 2, 1 (2015), 1–19.

[57] Peter M Senge. 2006. *The fifth discipline: The art and practice of the learning organization.* Currency.

[58] Thomas B Sheridan. 2010. The system perspective on human factors in aviation. In *Human factors in aviation*. Elsevier, 23–63.

[59] Nicolaj Siggelkow. 2002. Evolution toward fit. *Administrative science quarterly* 47, 1 (2002), 125–159.

[60] Aelita Skaržauskienė. 2010. Managing complexity: systems thinking as a catalyst of the organization performance. *Measuring business excellence* 14, 4 (2010), 49–64.

[61] Jonathan A Smith. 2003. *Qualitative psychology: A practical guide to research methods.* Sage Publications, Inc.

[62] Olusegun A Sogunro. 2002. Selecting a quantitative or qualitative research methodology: An experience. *Educational Research Quarterly* 26, 1 (2002), 3.

[63] Rudolf Stichweh. 2011. Systems theory. *International Encyclopedia of Political Science. New York: Sage* (2011).

[64] Amos Tversky and Daniel Kahneman. 1974. Judgment under Uncertainty: Heuristics and Biases: Biases in judgments reveal some heuristics of thinking under uncertainty. *science* 185, 4157 (1974), 1124–1131.

[65] Ricardo Valerdi and Heidi L Davidz. 2009. Empirical research in systems engineering: challenges and opportunities of a new frontier. *Systems Engineering* 12, 2 (2009), 169–181.

[66] Frank Van Latum, Rini Van Solingen, Markku Oivo, Barbara Hoisl, Dieter Rombach, and Gunther Ruhe. 1998. Adopting GQM based measurement in an industrial environment. *IEEE software* 15, 1 (1998), 78–86.

[67] Rini Van Solingen and Egon Berghout. 2001. Integrating goal-oriented measurement in industrial software engineering: industrial experiences with and additions to the Goal/Question/Metric method (GQM). In *Proceedings Seventh International Software Metrics Symposium*. IEEE, 246–258.

[68] Edwin Van Teijlingen and Vanora Hundley. 2002. The importance of pilot studies. *Nursing Standard (through 2013)* 16, 40 (2002), 33.

[69] Ludwig Von Bertalanffy. 1993. *General system theory: Foundations, development, applications.* Technical Report. Georges Braziller, Inc.

[70] Lars Göran Wallgren and Jan Johansson Hanse. 2011. The motivation of information technology consultants: The struggle with social dimensions and identity. *Human Factors and Ergonomics in Manufacturing & Service Industries* 21, 6 (2011), 555–570.

[71] Gerald M Weinberg. 1971. *The psychology of computer programming.* Vol. 29. Van Nostrand Reinhold New York.

[72] Carol A Wellington, Thomas Briggs, and C Dudley Girard. 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*. 1–5.

[73] Andrew Whiteley, Julien Pollack, and Petr Matous. 2021. The Origins of Agile and Iterative Methods. *The Journal of Modern Project Management* 8, 3 (2021).

[74] Carla Willig. 2013. *Introducing qualitative research in psychology.* McGraw-hill education (UK).

[75] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering.* Springer Science & Business Media.

[76] Brent Wolff, John Knodel, and Werasit Sittitrai. 1993. Focus groups and surveys as complementary research methods: A case example. In *Successful focus groups*, Morgan, David L (Ed.). SAGE Publications, Thousand Oaks, CA.

[77] Marvin V Zelkowitz and Dolores Wallace. 1997. Experimental validation in software engineering. *Information and Software Technology* 39, 11 (1997), 735–743.