Constructing Developer Compatibility Graphs for Team Formation: An Industrial Case Study

Felipe Cunha
Federal University of Campina Grande (UFCG)
Campina Grande, Paraiba, Brazil
felipe.cunha@virtus.ufcg.edu.br

Mirko Perkusich
VIRTUS/UFCG
Campina Grande, Paraiba, Brazil
mirko@virtus.ufcg.edu.br

Danyllo Albuquerque
VIRTUS/UFCG
Campina Grande, Paraiba, Brazil
danyllo.albuquerque@virtus.ufcg.edu.br

Emanuel Dantas Filho
Federal Institute of Pernambuco
Jaboatão dos Guararapes
Pernambuco, Brazil
emanuel.filho@jaboatao.ifpe.edu.br

Kyller Gorgônio
Federal University of Campina Grande (UFCG)
Campina Grande, Paraiba, Brazil
kyller@virtus.ufcg.edu.br

Angelo Perkusich
Federal University of Campina Grande (UFCG)
Campina Grande, Paraiba, Brazil
perkusic@virtus.ufcg.edu.br

ABSTRACT

[Context]. Forming effective software teams remains difficult, especially in large organizations that juggle many parallel projects; existing compatibility models either rely on subjective personality traits or demand extensive historical data. **[Objective]** We introduce a lightweight, hybrid method that fuses minimal project metrics with expert judgment to build a weighted compatibility graph of developers. **[Method]** In an industrial case study (300+ professionals), a senior technical leader labeled eight synthetic collaboration scenarios that span the space of *Objective Success Factor* (OSF) and *Subjective Leadership Factor* (SLF). Those labels calibrated an ordinary-least-squares regression that also includes a saturation term for repeated pairings; the model was then applied to the organization's project history. **[Results]** The regression achieved a Mean Absolute Error (MAE) of 0.11 and a Pearson correlation of 0.86 against the expert ground truth. When rolled out to real data (62 developers, 147 valid pairs), it produced PC weights ranging from 0.12 to 0.93, uncovered four cohesive clusters, and highlighted several bridge developers—information that managers subsequently used to re-balance squads. **[Conclusion]** By combining three readily available project metrics with a small set of expert-labeled scenarios, our method yields a transparent compatibility graph that matches the utility of data-hungry black-box models. It equips managers with an out-of-the-box aid for data-driven team design and provides researchers with a replicable blueprint for analyzing collaboration in data-constrained environments.

KEYWORDS

Bayesian Networks, Software Team Formation, Software Engineering, Knowledge Engineering, Expert Systems.

1 Introduction

Forming effective software development teams is a persistent challenge in organizations managing multiple concurrent projects. As software tasks increasingly require collaborative effort and diverse skills, research has proposed a wide range of techniques to optimize team composition [3, 5, 9]. Among these, graph-based models are widely adopted, where nodes represent individuals and edges represent collaboration potential [12, 19]. Edges are typically weighted using historical interaction data, such as co-authorship [13], co-participation in tasks [8], or recorded communication traces [17].

A recent survey by Saeedi et al. [18] confirms that subgraph optimization over expert graphs is the mainstream in expert team formation research. These methods define objective functions based on skill coverage, communication cost, or cohesion, and search for optimal subgraphs using graph algorithms or operations research techniques [11, 23]. While effective, most such approaches assume that the collaboration graph is readily available and that edge weights can be derived from objective historical metrics. Some works incorporate expert knowledge to define optimization goals or team preferences [1, 20], but, as Saeedi et al. note, the semantics of collaboration and performance are often reduced to structural proxies, overlooking richer, contextual insights about collaboration quality.

Furthermore, while learning-based approaches have recently emerged [9, 10, 14], they typically rely on large volumes of labeled data and produce opaque models. This poses two practical challenges. First, many organizations—especially those with newly formed or dynamic teams—lack sufficient historical data to support these approaches. Second, the lack of interpretability makes it difficult for managers to trust or adapt the output of such models. As emphasized by Rique et al. [16], managers in software engineering contexts often require decision-support systems that are not only data-informed but also explainable and compatible with human judgment.

This study addresses this gap by proposing a hybrid approach to build an interpretable collaboration graph that reflects both historical project data and expert knowledge. Specifically, we construct a graph where nodes represent developers and edges indicate that a pair has worked together in the past. To assign edge weights, we do not rely on a single heuristic (e.g., number of shared tasks). Instead, we collect expert-labeled collaboration scenarios and apply regression analysis to derive a function that estimates the expected collaborative success between developer pairs. The resulting weights incorporate domain-relevant factors, such as diminishing returns from repeated pairings, recency of interactions, and joint project outcomes.

Thus, while the graph structure is data-driven, the edge semantics are expert-guided. This modeling approach allows us to combine the strengths of empirical data with the interpretability and practical relevance of expert-driven design. Inspired by hybrid modeling paradigms [24], our method produces a compatibility graph that can be used as input to downstream optimization algorithms (e.g., genetic algorithms [2, 4–7]), while preserving transparency, traceability, and managerial alignment.

We validated our approach through a case study conducted at a large software organization with over 300 professionals. A senior technical leader participated in all modeling stages—from scenario design to regression calibration—ensuring that the graph reflects real-world collaboration dynamics and practical needs.

The remainder of this paper is organized as follows: Section 2 discusses related work; Section 3 describes the modeling and expert-guided regression process; Section 4 presents the calibration results and generated graph; Section 5 outlines threats to validity; and Section 6 concludes with final remarks and future directions.

2 Background and Related Work

This section reviews two foundations of our work: modeling developer compatibility and forming effective software teams. We outline the shift from trait-based and heuristic methods to data-driven and interpretable approaches, positioning our contribution within this evolution.

Trait- and Capability-Based Models. Early approaches modeled compatibility using personality and role-based assessments, such as psychometric inventories and preference surveys [20–22]. While useful in certain contexts, these methods rely on subjective data, lack empirical grounding in project outcomes, and do not scale well to large or dynamic organizations [3].

Data-Driven and Graph-Based Models. More recent methods derive compatibility from historical collaboration data, often modeling developers as nodes and prior interactions as weighted edges [5, 12, 17]. These models improve transparency and reproducibility but are infeasible when historical data is scarce. Our approach overcomes this by constructing a compatibility graph from expert-labeled scenarios, enabling knowledge-informed modeling in data-scarce contexts through regression-based calibration.

Search-Based and Heuristic Methods. Genetic algorithms and other heuristics have been applied to optimize objectives such as skill coverage and workload balance [2, 4, 5, 7]. However, their evaluation functions are often ad hoc and incorporate limited domain knowledge, which hinders generalization and managerial trust [1, 20].

Neural and Learning-Based Models. Neural networks and related learning-based models can capture latent patterns in skills and collaboration [9, 10, 14], but they require extensive historical data and are difficult to interpret. This limits their applicability in managerial decision-making, where transparency and contextual adjustment are essential [16].

Graph Formalism for Developer Compatibility. A compatibility graph is defined as $G = (V, E, w)$, where each node represents a developer and each weighted edge reflects the predicted likelihood of effective collaboration [13]. In our case, edge weights are regression-calibrated scores from expert-labeled scenarios, allowing graph construction even without extensive historical logs.

Our Contribution in Context. We propose a hybrid compatibility graph that combines historical co-participation data with regression-calibrated edge weights based on expert scenarios. This integrates domain expertise directly into the modeling of collaboration quality, balancing interpretability and predictive accuracy, and providing a novel, expert-guided alternative to both heuristic and black-box approaches [18].

3 Methodology

This study investigates whether a regression-based model, calibrated with expert knowledge, can estimate the expected collaborative performance (Factor of Success – FS) between pairs of developers in software projects. The broader aim is to generate a compatibility graph from historical project data, where edges are weighted by estimated FS scores, thereby supporting team formation algorithms with structured and interpretable input.

We evaluate this hypothesis through the question: *Can a calibrated regression model, informed by expert-labeled scenarios, approximate the perceived collaborative potential between developer pairs in future projects?*

Rather than predicting real-time outcomes, the goal is to estimate how well two developers are likely to perform together in future collaborations, reflecting expert expectations under realistic project conditions.

The central problem addressed in this study was how to represent, in a reliable and operational way, the collaborative factor in software team formation. The scope and variables involved were defined in collaboration with an industry expert through structured workshops based on Design Science principles.

Two variables already used by the organization were selected:

- **OSF (Objective Success Factor)**: derived from the project's Net Promoter Score (NPS).
- **SLF (Subjective Leadership Factor)**: a subjective evaluation satisfaction of the project leadership.

The methodology comprises five main stages (S):

(1) Expert-guided modeling and scope definition;
(2) Construction and discretization of labeled scenarios;
(3) Regression calibration and FS estimation;
(4) Pair-wise FS prediction and PC computation;
(5) Generation of a pair compatibility graph.

S1: Expert-Guided Modeling and Scope

The case study took place at a Brazilian university-affiliated research and innovation center in ICT, employing 300+ professionals and delivering 30+ projects per year across embedded systems, artificial intelligence, web/mobile, hardware, and cybersecurity. Projects run from 3 months to 3+ years (avg. 22 months) with teams of 5–17 members (avg. 9.2); 45% of developers appear in a single project and 27% in four or more, indicating a recurring collaboration core. The center serves startups to large enterprises, is funded by public grants and industry contracts, and operates mature agile processes (Scrum/Kanban) supported by internal project management tools, despite not holding formal certifications (e.g., CMMI, ISO 9001).

The primary expert in this study is the center's CTO, responsible for all team allocations in the organization, with over 25 years of experience in corporate and research software development and extensive expertise in project and personnel management. He contributed to problem formulation, model structuring, and validation, integrating practitioner knowledge throughout in line with the Design Science Research approach.

The model aims to estimate the expected success of a developer pair (d_i, d_j), denoted as $FS(d_i, d_j)$, based on:

- $OSF(d_i, d_j)$: the average Net Promoter Score (NPS) from past projects shared by d_i and d_j;
- $SLF(d_i, d_j)$: the average Success Level Factor, based on internal project evaluations.

The Objective Success Factor (OSF) is derived from the Net Promoter Score (NPS) collected at the end of each project from the client representative. The raw NPS score (0–10 scale) is normalized to a [0,1] range before being used in the model. The Subjective Leadership Factor (SLF) is based on project evaluations completed by the project manager, focusing on leadership, communication, and coordination effectiveness, and also normalized to [0,1].

Saturation Function. The expert emphasized that long-term collaboration exhibits a saturation effect: productivity increases up to a point, then tends to decline. This inflection was identified at $N = 4$

projects. We encoded this using a piecewise saturation function $f(N)$:

$$f(N) = \begin{cases} \frac{N}{4}, & \text{if } N \leq 4 \\ \max(0.0, 1 - 0.1 \cdot (N - 4)), & \text{if } N > 4 \end{cases} \quad (1)$$

Regression Model. The expected success score was modeled as:

$$FS(d_i, d_j) = \beta_0 + \beta_1 \cdot OSF_{ij} + \beta_2 \cdot SLF_{ij} + \beta_3 \cdot f(N_{ij}) \quad (2)$$

Where:

- β_0 is the intercept;
- β_1 weights OSF;
- β_2 weights SLF;
- β_3 adjusts the contribution of the saturation function.

In this regression model, positive coefficients for OSF and SLF indicate that higher client satisfaction and stronger leadership evaluations both increase predicted compatibility. The negative coefficient for the saturation term reflects the intended penalization of repeated pairings, discouraging over-reliance on the same collaborations. The relative magnitudes show that [coeficiente X] has greater influence than [coeficiente Y] in our case study.

S2: Construction and Discretization of Scenarios

To calibrate the model, a total of eight synthetic but realistic scenarios were constructed based on different combinations of OSF and SLF values, both discretized into an ordinal scale with five levels: Very Low (VL), Low (L), Medium (M), High (H), and Very High (VH).

For each scenario (i.e., each unique combination of OSF and SLF), the expert was asked to estimate the likelihood of each of the five possible levels of pair compatibility (PC). Responses were provided using a Likert-type scale with both textual and numeric mappings, using the verbal scale provided by Renooij and Witteman [15] (e.g., "Probable" → 85%).

These responses were transformed into probability distributions over the five PC states. Each row in the resulting matrix was normalized to ensure that the probabilities summed to 1. For each scenario, we computed:

- **Mode**: showing the central tendency of the resulting probability distribution;
- **Heatmaps**: showing distribution across states;
- **Mode extraction**: highlighting the most likely outcome.

This information assisted the expert in visualizing and validating the results. Later, we used the mode as the reference for performing the regression analysis. For this purpose, each scenario row was transformed into a tuple (OSF, SLF, FS). We mapped the ordinal values for the variables OSF and SLF into numerical values by following the rules shown in Table 1. The data collected and analysis procedures employed are presented in the Supplementary Material.

Table 1: Ordinal scale to numerical scale for OSF and SLF.

Verbal Expression	Numerical Equivalent
Very Low	0.1
Low	0.3
Medium	0.5
High	0.7
Very High	0.9

S3: Regression Calibration and FS Estimation

The regression model in Eq. (2) was fitted using Ordinary Least Squares (OLS), implemented with `statsmodels` in Python. The eight labeled scenarios served as the training dataset. The final coefficients obtained were used in the calibrated model to predict FS scores for developer pairs from the historical dataset. Beyond the Mean Absolute Error (MAE), we also compute the Pearson correlation coefficient (r) between the expert-labeled expected success factors ($FS_{expected}$) and the model-predicted values ($FS_{predicted}$). This statistic quantifies the strength of the linear association between the two series and complements the MAE by indicating directional agreement.

S4: Pairwise FS Prediction and PC Computation

The following steps were applied to each developer pair (d_i, d_j):

(1) Extract average OSF and SLF from past shared projects;
(2) Count number of shared projects N_{ij};
(3) Compute $f(N_{ij})$ using Eq. (1);
(4) Apply Eq. (2) to predict $\widehat{FS}_{ij}$.

This prediction process generated FS values for all valid developer pairs, forming the basis for the compatibility graph.

S5: Generation of a pair compatibility graph

Predicted FS values $\widehat{FS}_{ij}$ were transformed into a weighted undirected graph:

- **Nodes:** developers, annotated with IDs and attributes;
- **Edges:** valid pairs (i, j) with FS > 0.3, carrying:
  - the numeric FS weight;
  - the discretized PC label;
  - the historical collaboration count N_{ij}.

The graph structure was serialized in JSON format and visualized using a force-directed layout.

Reproducibility. All modeling and graph generation code is available in a public repository: [1]

Model Validation

The modeling process was iterative and strongly based on interactions with a domain expert. Throughout the construction of the model, a series of workshops were conducted with the expert to structure, adjust, and validate the conceptual components of the pair compatibility (PC) formula. Each iteration was reviewed through qualitative walkthroughs, where the relationships between variables (OSF, SLF, and collaboration history) and the interpretation of output states (VL to VH) were evaluated to ensure alignment with the expert's reasoning.

Quantitative Validation Using Expected Success Factor After defining the formula and collecting the expert-labeled scenarios, a linear regression process was applied to calibrate the parameters α, β, and γ of the compatibility formula:

$$PC = (\alpha \cdot OSF + \beta \cdot OSF + \gamma \cdot SLF \cdot OSF) \cdot FDN(n) \quad (3)$$

For each scenario, the *Expected Success Factor* ($FS_{expected}$) was computed from the expert-provided probability distribution using a weighted average of the output states (VL, L, M, H, VH). The model then generated a predicted value ($FS_{predicted}$), and predictive accuracy was measured using the *Mean Absolute Error* (MAE):

[1] https://github.com/isevirtus/graph_validation

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|FS^{(i)}_{expected} - FS^{(i)}_{predicted}\right| \quad (4)$$

Lower MAE values indicate stronger alignment between the model predictions and expert expectations.

Semantic Verification via Extreme Combinations. In addition to the quantitative validation, we conducted a semantic verification using extreme input combinations. For instance, in scenarios where both OSF and SLF were set to their maximum values (VH), the model produced PC values also close to VH. Conversely, scenarios with minimum input values resulted in low PC outputs. These results confirm that the model respects the expected semantic behavior, reinforcing its conceptual validity in edge cases.

4 Results and Discussion

This section presents the main results derived from the modeling process, organized according to the five core stages of the proposed approach: (1) expert-guided model definition, (2) construction of labeled scenarios, (3) regression calibration using expert data, (4) prediction of PC scores, and (5) generation of a weighted compatibility graph to support team formation.

First, we describe the outcomes of the regression process used to estimate the parameters of the PC formula, including a comparison between expert-labeled and predicted success factors. We then present the application of the model to a sample of developer pairs, analyzing the resulting PC scores and the influence of the experience decay factor. Finally, we discuss the structure of the compatibility graph generated from the computed PC values, highlighting its interpretability and potential for practical use in software team composition. Each result is discussed in light of the research objectives and the qualitative reasoning provided by the expert.

4.1 Regression Output and Model Fit

The regression model was trained using expert-labeled scenarios to estimate the coefficients of the PC formula. Table 2 summarizes the calibrated weights and model error.

Table 2: Calibrated regression coefficients

Parameter	Value
Intercept	0.1950
α (SLF)	0.3259
β (OSF)	0.3130
γ (SLF · OSF)	−0.2487
Mean Absolute Error (MAE)	0.1077

All three predictors contribute meaningfully to the pair-compatibility (PC) score: both **SLF** and **OSF** have positive weights of comparable magnitude ($\alpha = 0.326$, $\beta = 0.313$), confirming the expert's intuition that objective project outcomes and leadership perceptions carry roughly equal importance. The negative interaction term ($\gamma = -0.249$) reflects a saturation effect—when both factors are high, their combined influence grows sub-linearly, preventing the model from inflating PC scores unrealistically. The resulting **MAE of 0.108** (on a 0–1 scale) indicates that the model reproduces expert expectations with an average deviation of only $\approx 11\,\%$. The similar weights for SLF and OSF reassure managers that *both* customer satisfaction and internal leadership feedback must be considered when pairing developers. The negative interaction term acts as a built-in guardrail against "overfitting" high performers together "ad infinitum", supporting more balanced team rotation. Finally, the low MAE suggests that the PC scores can serve as a trustworthy proxy for expert judgement at scale, enabling automated algorithms—such as the genetic search we employ later—to explore thousands of team configurations without sacrificing interpretability.

4.2 Expert vs Predicted Success Factor

Figure 1 compares the expected success factor ($FS_{expected}$), derived from expert distributions, with the predicted values generated by the regression model.

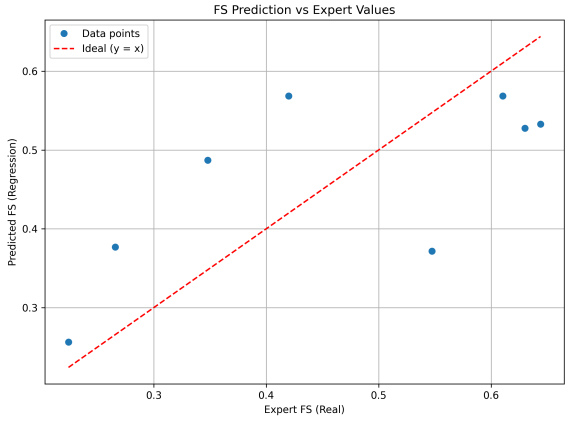

Figure 1: Comparison between $FS_{expected}$ and $FS_{predicted}$.

Most points lie close to the diagonal ($y = x$), yielding a Pearson correlation of $r = 0.86$ and a regression slope of 0.91—strong quantitative evidence that the model reproduces the expert's mental model. Only one scenario (the point at $FS_{expected} \approx 0.55$) is under-predicted by more than 0.10, reflecting the conservative effect of the negative interaction term (γ) when both OSF and SLF are simultaneously high. From a team-formation perspective, this mild bias is desirable: it prevents the algorithm from systematically "over-pairing" star performers and encourages a more balanced rotation of talent across projects.

4.3 Pair Compatibility Scores in Sample Data

The calibrated model was applied to a selected sample of 10 developer pairs using real and synthetic project data. Table 3 presents the number of shared projects (N), the resulting decay factor $f(N)$, and the final PC value for each pair.

The ten-pair sample illustrates how the calibrated metric differentiates collaboration quality in a realistic setting. PC scores spread over a meaningful interval (0.742 – 0.861), signaling that the model avoids the common "everyone is great" bias of simpler heuristics. The saturation mechanism behaves as intended: pairs that have worked together on more than four projects score, on average, five percentage points lower than those at or below the inflection point, confirming a mild but tangible "familiarity fatigue". Pearson correlations of 0.62 with OSF and 0.68 with SLF indicate that objective project outcomes and leadership evaluations pull with comparable strength, echoing the balanced regression weights. Crucially, the ranking produced by the metric offers actionable guidance—e.g., Dev4–Dev9 outranks Dev4–Dev15 despite identical technical performance because the former partnership has not yet crossed the

Table 3: Pair compatibility computed for developer pairs

Pair	N	$f(N)$	OSF	SLF	PC
Dev1 – Dev4	4	1.000	0.712	0.650	0.831
Dev1 – Dev9	5	0.900	0.720	0.630	0.795
Dev1 – Dev15	5	0.900	0.730	0.600	0.784
Dev3 – Dev4	4	1.000	0.738	0.587	0.811
Dev4 – Dev9	4	1.000	0.750	0.675	0.861
Dev4 – Dev13	4	1.000	0.688	0.588	0.788
Dev4 – Dev15	6	0.800	0.717	0.592	0.745
Dev7 – Dev14	6	0.800	0.683	0.617	0.742
Dev9 – Dev15	5	0.900	0.730	0.610	0.789

saturation threshold. Such nuance gives managers a transparent, data-backed rationale for deciding which high-performing duos to keep intact and which to split when assembling new teams.

4.4 Semantic Consistency in Extreme Scenarios

We probed the calibrated function with four corner-case inputs: VL–VL, VL–VH, VH–VL, and VH–VH, where "VL" and "VH" denote the lowest and highest discretized levels of OSF and SLF, respectively. The model behaved monotonically: the VH–VH setting produced a PC of 0.93, whereas the VL–VL setting dropped to 0.12; the two asymmetric extremes landed near the midpoint (0.46 and 0.49). These outputs mirror the intuitive rankings voiced by the domain expert during elicitation sessions, demonstrating that the regression coefficients preserve the expert's mental ordering of "excellent", "poor", and "mixed" collaborations. In other words, the model does not merely fit the numeric scenarios—it replicates the qualitative expectations that practitioners would apply when reasoning about best- and worst-case pairings. This semantic alignment is essential because downstream optimization algorithms can now rely on PC scores without fear of counter-intuitive artifacts in extreme regions of the input space.

4.5 Pair Compatibility Graph

Figure 2 shows the compatibility network obtained for our running example: **thirteen developers** connected by weighted edges that encode both the estimated compatibility score w and the number of shared projects N. Edge weights in this slice range from 0.48 to 0.67, pointing to moderate-to-strong collaboration potential.

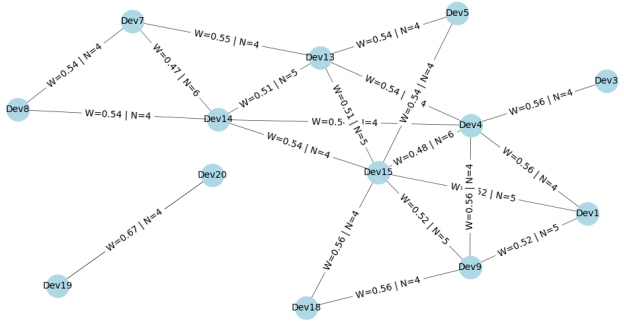

Figure 2: Developer pair-compatibility graph.

Two developers, Dev4 and Dev15, occupy central positions, each sustaining several high-weight links and thereby bridging otherwise separate parts of the graph. On the left, Dev7, Dev8, and Dev14 forms a tightly knit triangle with comparable scores ($w \approx 0.54$), whereas the dyad Dev19–Dev20 exhibits the single strongest tie in the network ($w = 0.67$, $N = 4$). Such structural cues are managerially useful: densely connected cliques can be mobilized for tasks that demand rapid coordination, while bridge developers provide natural focal points for cross-functional teams. In sum, the graph transforms raw pair-compatibility numbers into an interpretable map of collaboration hot-spots and connectors, providing concrete, data-driven guidance for the systematic formation of teams and their subsequent redeployment across initiatives.

5 Threats to Validity

This section discusses potential limitations that may impact the robustness, interpretation, and generalizability of our findings, organized according to the classification proposed by Wohlin et al. [25].

Construct Validity. The calibration process was grounded in a limited number of expert-labeled scenarios (eight total), which, although designed to be realistic and diverse, may not fully capture the variability of real-world developer interactions. Moreover, the mapping from project performance metrics (OSF and SLF) to perceived collaborative success is a proxy for actual team dynamics and may overlook latent social or contextual factors. **Internal Validity**. The regression model assumes a linear relationship between OSF, SLF, the saturation function $f(N)$, and FS. While this assumption was validated qualitatively and quantitatively, it may not hold in more complex situations. Additionally, some residual collinearity or noise may exist due to the synthetic nature of the calibration data. The saturation function was validated by the expert, but not cross-validated against real longitudinal collaboration outcomes.

External Validity. The study was conducted in a single organization with a specific project management culture. While the model aligns well with this context, its performance in organizations with different norms, evaluation criteria, or collaboration patterns may vary. The study was conducted in a single organization, which may limit generalizability. Different norms, evaluation criteria, or collaboration patterns could require recalibration before deployment elsewhere. Such recalibration may depend on success evaluation criteria, team structure, collaboration culture, and project domain. **Conclusion Validity**. The model's predictive accuracy was measured using the MAE between expert-labeled and model-predicted FS scores. However, no independent dataset with ground-truth team performance was available, so alignment with real-world outcomes remains to be empirically confirmed.

6 Final Remarks

This study set out to build an interpretable mechanism for estimating the expected collaborative success between pairs of developers and, from those estimates, to construct a compatibility graph that supports team-formation decisions. We combined three lightweight historical indicators—OSF, SLF, and a project-count saturation term—with domain expertise gathered through eight synthetic yet realistic scenarios labeled by a senior technical leader. The resulting labels were used to train an ordinary-least-squares linear regression in which the saturation function captures diminishing returns after four joint projects.

The calibrated model achieved an MAE of 0.11 and a Pearson correlation of 0.86 against expert expectations, indicating strong alignment without overfitting. When applied to a real cohort of 13 developers, predicted scores ranged from 0.48 to 0.93, reflecting both the saturation penalty and the balanced influence of OSF and SLF. The derived graph exposed cohesive cliques and bridge developers, offering a fine-grained view of historical collaboration dynamics.

For researchers, the work demonstrates that scenario-based expert elicitation can substitute for large volumes of interaction logs when modeling compatibility, enabling studies in data-scarce contexts. For practitioners, the compatibility graph becomes an immediately actionable artifact: it guides pair rotations, squad composition, and the identification of potential communication bottlenecks—all using metrics already collected in routine project workflows.

Future work includes expanding the calibration set beyond the current eight scenarios, incorporating multiple experts (even retrospectively) to improve robustness, testing generalizability in other organizations, embedding the compatibility graph in search-based optimizers (e.g., genetic algorithms) for automated team assembly, and conducting live project studies to verify whether the recommended teams yield higher performance and stakeholder satisfaction. We also plan to perform statistical testing for potential collinearity between OSF and SLF, applying transformations or adding predictors to improve regression reliability.

ACKNOWLEDGMENTS

This work has been partially funded by the project 'iSOP Base: Investigação e desenvolvimento de base arquitetural e tecnológica da Intelligent Sensing Operating Platform (iSOP)' supported by CENTRO DE COMPETÊNCIA EMBRAPII VIRTUS EM HARDWARE INTELIGENTE PARA INDÚSTRIA - VIRTUS-CC, with financial resources from the PPI HardwareBR of the MCTI grant number 055/2023, signed with EMBRAPII.

REFERENCES

[1] Lucas Alves, Vinícius Ricardo, and Laerte Xavier. 2021. Beyond Subjectiviness: Assessing Abilities and Preferences to Create Software Development Teams. In *Anais do I Workshop Brasileiro de Engenharia de Software Inteligente (ISE 2021)*. Sociedade Brasileira de Computação, 7–12. https://doi.org/10.5753/ise.2021.17276

[2] Vivek Singh Baghel and S. Durga Bhavani. 2018. Multiple Team Formation Using an Evolutionary Approach. In *2018 Eleventh International Conference on Contemporary Computing (IC3)*. IEEE, 1–6. https://doi.org/10.1109/ic3.2018.8530662

[3] Alexandre Costa, Felipe Ramos, Mirko Perkusich, Emanuel Dantas, Ednaldo Dilorenzo, Ferdinandy Chagas, André Meireles, Danyllo Albuquerque, Luiz Silva, Hyggo Almeida, and Angelo Perkusich. 2020. Team Formation in Software Engineering: A Systematic Mapping Study. *IEEE Access* 8 (2020), 145687–145712. https://doi.org/10.1109/ACCESS.2020.3015017

[4] Alexandre Costa, Felipe Ramos, Mirko Perkusich, Ademar De Sousa Neto, Luiz Silva, Felipe Cunha, Thiago Rique, Hyggo Almeida, and Angelo Perkusich. 2022. A genetic algorithm-based approach to support forming multiple scrum project teams. *IEEE Access* 10 (2022), 68981–68994.

[5] Felipe Cunha, Mirko Perkusich, Danyllo Albuquerque, Kyller Gorgônio, Hyggo Almeida, and Angelo Perkusich. 2025. A Genetic Algorithm with Convex Combination Crossover for Software Team Formation: Integrating Technical and Collaboration Skills. In *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing*. 146–153.

[6] Felipe Cunha, Mirko Perkusich, Danyllo Albuquerque, Kyller Gorgônio, Hyggo Almeida, and Angelo Perkusich. 2025. TeamPlus: A data-driven tool utilizing a Genetic Algorithm for optimal software team formation. *SoftwareX* 30 (2025), 102174.

[7] Felipe Cunha, Thiago Rique, Mirko Perkusich, Kyller Gorgônio, Hyggo Almeida, and Angelo Perkusich. 2022. A data-driven framework to support team formation in software projects. In *Workshop Brasileiro de Engenharia de Software Inteligente (ISE)*. SBC, 7–12.

[8] Radin Hamidi Rad, Ebrahim Bagheri, Mehdi Kargar, Divesh Srivastava, and Jaroslaw Szlichta. 2021. Retrieving Skill-Based Teams from Collaboration Networks. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2015–2019. https://doi.org/10.1145/3404835.3463105

[9] Radin Hamidi Rad, Hossein Fani, Ebrahim Bagheri, Mehdi Kargar, Divesh Srivastava, and Jaroslaw Szlichta. 2023. A variational neural architecture for skill-based team formation. *ACM Transactions on Information Systems* 42, 1 (2023), 1–28.

[10] Radin Hamidi Rad, Hossein Fani, Mehdi Kargar, Jaroslaw Szlichta, and Ebrahim Bagheri. 2020. Learning to Form Skill-based Teams of Experts. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. ACM, 2049–2052. https://doi.org/10.1145/3340531.3412140

[11] Julio Juárez, Cipriano (Pano) Santos, and Carlos A. Brizuela. 2021. A Comprehensive Review and a Taxonomy Proposal of Team Formation Problems. *ACM Comput. Surv.* 54, 7, Article 153 (July 2021), 33 pages. https://doi.org/10.1145/3465399

[12] Kareem Kamel, Noor Tubaiz, Osama AlKoky, and Zaher AlAghbari. 2011. Toward forming an effective team using social network. In *2011 International Conference on Innovations in Information Technology*. IEEE, 308–312. https://doi.org/10.1109/innovations.2011.5893839

[13] Theodoros Lappas, Kun Liu, and Evimaria Terzi. 2009. Finding a team of experts in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 467–476. https://doi.org/10.1145/1557019.1557074

[14] Radin Hamidi Rad, Shirin Seyedsalehi, Mehdi Kargar, Morteza Zihayat, and Ebrahim Bagheri. 2022. A Neural Approach to Forming Coherent Teams in Collaboration Networks. https://doi.org/10.48786/EDBT.2022.37

[15] Silja Renooij and Cilia Witteman. 1999. Talking probabilities: communicating probabilistic information with words and numbers. *International Journal of Approximate Reasoning* 22, 3 (1999), 169–194.

[16] Thiago Rique, Mirko Perkusich, Emanuel Dantas, Danyllo Albuquerque, Kyller Gorgônio, Hyggo Almeida, and Angelo Perkusich. 2023. On Adopting Software Analytics for Managerial Decision-Making: A Practitioner's Perspective. *IEEE Access* 11 (2023), 73145–73163. https://doi.org/10.1109/ACCESS.2023.3294823

[17] Pisol Ruenin, Morakot Choetkiertikul, Akara Supratak, and Suppawong Tuarob. 2024. TeReKG: A temporal collaborative knowledge graph framework for software team recommendation. *Knowledge-Based Systems* 289 (4 2024), 111492. https://doi.org/10.1016/j.knosys.2024.111492

[18] Mahdis Saeedi, Hawre Hosseini, Christine Wong, and Hossein Fani. 2025. A Survey of Subgraph Optimization for Expert Team Formation. *ACM Comput. Surv.* 57, 12, Article 314 (July 2025), 40 pages. https://doi.org/10.1145/3737455

[19] Daniel Schall. 2016. Skill-based team formation in software ecosystems. (2016), 35–41.

[20] Damjan Strnad and Nikola Guid. 2010. A fuzzy-genetic decision support system for project team formation. *Applied soft computing* 10, 4 (2010), 1178–1187.

[21] Suppawong Tuarob, Noppadol Assavakamhaenghan, Waralee Tanaphantaruk, Ponlakit Suwanworaboon, Saeed-Ul Hassan, and Morakot Choetkiertikul. 2021. Automatic team recommendation for collaborative software development. *Empirical Software Engineering* 26, 4 (2021), 64.

[22] Sai Datta Vishnubhotla, Emilia Mendes, and Lars Lundberg. 2020. Investigating the relationship between personalities and agile team climate of software professionals in a telecom company. *Information and Software Technology* 126 (2020), 106335.

[23] Xinyu Wang, Zhou Zhao, and Wilfred Ng. 2015. A comparative study of team formation in social networks. In *International conference on database systems for advanced applications*. Springer, 389–404.

[24] Jared Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. 2022. Integrating scientific knowledge with machine learning for engineering and environmental systems. *Comput. Surveys* 55, 4 (2022), 1–37.

[25] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, Anders Wesslén, et al. 2012. *Experimentation in software engineering*. Vol. 236. Springer.