Modeling and Analyzing Information Flow in Development Teams as a Pipe System

Jil Klünder*, Oliver Karras*, Nils Prenner*, Kurt Schneider*
**Leibniz University Hannover,*
Software Engineering Group,
Hannover, Germany,
Email: {jil.kluender, oliver.karras, nils.prenner, kurt.schneider}@inf.uni-hannover.de

***Abstract*—Teamwork is essential for developing valuable software. Working in a team requires an appropriate information exchange among team members in order to avoid loss of information. In order to analyze and improve information flows, it is recommended to observe the information exchange in a team. We propose an approach for modeling the information flow of teams as a pipe system. Different pipe diameters represent the amount of information passing through the pipe. In order to show the applicability of our approach, we conducted a case study in a globally distributed software engineering company. The study consists of the elicitation of information flows inside the company and the automated analysis by our approach. We are able to visualize the information flows, find critical paths such as bottlenecks, and improve information flow structures. This enables project leaders to customize the communication structure to the needs of their team and to prevent loss of information.**

***Index Terms*—Communication behavior, information sharing, development teams, developer network, social network analysis**

I. Introduction

Communication and information exchange are two essential parts of daily work in software development teams [1]. An inadequate amount of communication is one of the main obstacles hampering successful collaboration [2], [3]. There exist different types of information such as customer needs or details of story cards that need to be shared within the team [4]. This intense communication takes place via different communication channels, such as face-to-face, e-mails, chat messages or documents [4], [5]. Developers and stakeholders are often not aware of the ongoing communication behavior and information flow [6], [7]. They frequently face the problem of instantaneously accessing the right information due to the distribution to different information stores. Information flow is impeded if the developers do not know where to find the required information. In turn, inadequate information sharing complicates teamwork. Analyzing information flow helps a team to become aware of the communication and information sharing [8]–[10]. Detecting critical issues helps to improve the teamwork and to prevent problems like lost information, missing functionality, and a dissatisfied customer. There are different approaches to analyze the information flow in projects [11], [12]. These approaches consider the existence or absence of an information flow. However, they take neither the amount nor the required time to transport information from one person to another into account. Information can flow directly between two persons, but it may also flow via different persons before reaching the required team member.

In this paper, we present the metaphor of a pipe system to consider the wide range of ways to transport information – and to facilitate the analysis of the information flow. The pipe's diameter resembles the amount of information transmitted per time, e.g., via chat, phone, or face-to-face. Water in a pipe system can take different ways from a source to a target, but it is limited by the smallest pipe on the way. The same holds for information: The maximum amount of information flowing between two persons is limited by the minimum amount of information that can be transported on each part on the way from the sender to the receiver of information.

We aim at visualizing a developer network with different kinds of pipe diameters representing the amount of information flow. The pipe system visualizes information flow within the network and helps to detect critical paths and bottlenecks. This facilitates the developers' understanding of the importance of adequate information sharing. Furthermore, central nodes can be easily recognized in the network. These central nodes are of particular importance for information sharing since they bundle a lot of information [13].

Despite the theoretical orientation of our paper, we apply our approach in an industrial case study with a globally distributed software developing company.

The remainder of this paper is structured as followed: In Sec. II, we present related work. In Sec. III, we depict the groundwork of graph theory used in our approach and the information flow analysis with FLOW. In Sec. IV, we introduce our approach for an automated analysis of FLOW diagrams based on pipe systems. Sec. V presents our case study. The results and limitations of our approach are discussed in Sec. VI. We conclude and present future work in Sec. VII.

II. Related Work

Our work focuses on quantifying information flow analysis. There are already existing approaches in this area. Durugbo et al. [14] give an overview of existing approaches for modeling information flow. The authors review literature and distinguish between approaches for diagrammatically (i.e. visually) and mathematically (i.e. analytically) modeling information flow. According to the literature review of Durugbo et al. [14], most visualizations are done using graphs or networks. Besides graph theory and social network analysis, probability theory, vector analysis, Markov models and interaction matrices are also applied to analyze the information flows [14]. The choice of visualization and analysis approaches often depends on the level of application: It is possible to analyze information flows on a macro, meso, and micro level [15]. Our concepts go along with Durugbo et al.'s [14] findings which are not tailored for software development teams.

Smith [16] provides an overview of the foundations of quantitative information flow. He mainly considers the flow of secure information. His approach is based on the concept of vulnerability, which is closely related to Bayes risk measuring uncertainty [16]. The author focuses on sensitive information as a security issue. His approach is not applicable to our information flow analysis which is based on social interactions.

Lowe [17] presents an approach of quantifying information flow by considering the capacity of a covert channel in a security system. His approach is based on the *Communicating Sequential Processes algebra* describing interactions between various kinds of communicating processes. Lowe [17] considers an information flow between two users, *High* and *Low*, using a covert channel. He wants to quantify the amount of information flowing from *High* to *Low*. Lowe's [17] approach is more technical due to the use in security and not based on human factors.

Kiesling et al. [13] combine the FLOW method for information flow analysis with social network analysis. They apply various centrality measures which are well established in sociology and psychology. These centrality measures detect central persons who are very important for information sharing. The authors consider degree centrality counting the number of incoming and outgoing edges, closeness centrality, betweenness and flow betweenness centrality. The different measures indicate which persons are central from different points of view or which are well situated in the network to share and receive information very fast and easily. This approach also helps to quantify information flow analysis since certain centrality measures underline the qualitative results. Compared to our approach, Kiesling et al. [13] only consider whether there is an edge between two nodes, i.e. an information exchange takes place. In contrast, we consider the amount of information which flows over an edge.

III. Background

Our approach of modeling information flow is based on graph theory and extends the FLOW method. In the following, we present the necessary basics to understand our approach.

A. Graph Theory

We use the Ford-Fulkerson-Method to realize our approach of considering a developer network as pipe system [18]. We consider a finite, directed graph $G = (V, E)$ with a set of nodes V and a set of edges $E \subseteq V \times V$. This graph grasped as a network is called *flow network* [18]. Let $c\colon V \times V \to \mathbb{R}_{\geq 0}$ be a non-negative function assigning a *capacity* $c(u, v)$ to each edge $(u, v) \in E$. This function is called *capacity function*. For $(u, v) \notin E$, we assume that there is no information flow, i.e. $c(u, v) = 0$. A *flow* in G is defined to be a real-valued function $f\colon V \times V \to \mathbb{R}$ satisfying

1) *Capacity constraint:* $\forall u, v \in V$ let $f(u, v) \leq c(u, v)$, i.e. the flow is always less or equal to the capacity
2) *Skew symmetry:* $\forall u, v \in V$ let $f(u, v) = -f(v, u)$, i.e. going in the opposite direction leads to a negative flow
3) *Flow conservation:* $\forall u \in V \setminus \{s, t\}$, let $\sum_{vu \in V} f(u, v) = 0$, i.e. there is theoretically no loss of information in a node; the amount of incoming information is equal to the amount of outgoing information

Given a flow and a capacity function, we can define the *residual capacity*, which is given by the difference between the actual flow between to nodes and the capacity of the edge connecting them. Formally, the residual capacity is defined to be

$$c_f(u, v) = \begin{cases} c(u, v) - f(u, v) & \text{if } (u, v) \in E \\ f(v, u) & \text{if } (v, u) \in E \\ 0 & \text{otherwise} \end{cases}$$

Then, the *residual network* $G_f = (V, E_f)$ induced by a flow f is given by the edges in $E_f = \{(u, v) \in V \times V\colon c_f(u, v) > 0\}$. Thus, the residual network consists of all nodes of the initial network and the edges are given by those with a not yet maximal flow. An *augmenting path* p is a simple path from s to t in the residual network G_f. One may show that each edge on such a path in the residual network admits some additional positive flow from u to v without violating the capacity constraint on the edge [18]. The *residual capacity* of an augmenting path p is defined to be the maximum amount by which the flow on each edge in p can be increased without violating the capacity criteria, i.e. $c_f(p) = \min\{c_f(u, v)\colon (u, v) \text{ is on } p\}$.

We aim at calculating the maximum information flow $f(s, t)$ between a source $s \in V$ and a target $t \in V$. Therefore, we apply the Ford-Fulkerson-Method. Algorithm 1 visualizes the proceeding.

Algorithm 1 FORD-FULKERSON-Method [18, p. 724]

```
1:  for each edge $(u, v) \in G.E$ do
2:      $f(u, v) = 0$
3:  end for
4:  while there exists a path $p$ from $s$ to $t$ in the residual
    network $G_f$ do
5:      $c_f(p) = \min\{c_f(u, v) : (u, v) \in p\}$
6:      for each edge $(u, v) \in p$ do
7:          if $(u, v) \in E$ then
8:              $f(u, v) = f(u, v) + c_f(p)$
9:          else
10:             $f(v, u) = f(v, u) - c_f(p)$
11:         end if
12:     end for
13: end while
```

In the beginning, the flow of each edge within the network is initialized with zero. Then, we search for a path p from the source s to the target t in G_f in the residual network G_f induced by the current flow value. We calculate the minimum of all residual capacities, $c_f(p)$ on p, and increase the flow of each edge in E by this $c_f(p)$. If an edge is not contained in the initial network, we decrease the contrary edge (v, u) by the same value. This proceeding is well-defined and leads to the requested results [18, p. 714 ff.].

The Edmond-Karp-Algorithm implements the Ford-Fulkerson-Method with polynomial runtime. This algorithm searches for the augmenting path (line 4 in Alg. 1) by using breadth-first-search to find the shortest augmenting path from s to t. This is helpful since the distance of the shortest path from s to any other node $v \neq t$ in the residual network increases monotonically with each flow augmentation [18, p. 727]. Using this implementation decreases the runtime, since the total number of flow augmentations is in $\mathcal{O}(VE)$ [18, p. 729]. The total runtime is $\mathcal{O}(VE^2)$.

B. *Information flow analysis with FLOW*

FLOW is an established method to analyze and improve the communication in software projects [12]. It is a systematic analysis of information flows in software development projects, but it is also applicable for other kinds of projects. FLOW helps to detect lacks or anomalies in order to avoid a loss of information by providing a structured proceeding for evaluating, visualizing, analyzing and improving information flow in teams [12], [19].

FLOW distinguishes between two types of information flows and stores: *solid* and *fluid* [19]. Solid information is long-term and repeatably accessible and can be understood by third parties with domain knowledge. An information flow is defined to be *fluid*, whenever one of these three criteria is not met [12]. *Solid* information is usually captured in written documents, source code or other long-living stores [12]. Examples for *fluid* information are undocumented meetings, informal face-to-face communication and implicit knowledge [10]. Emails may be either fluid or solid, depending on the further use and storing of the email [10].

Analysts conduct interviews with important members of the project to elicit the communication behavior. The group of interviewed persons often consists of the team leader, one or two developers and other important stakeholder or contributor to the team's work. Each interview starts with a short introduction of the interviewee to give an overview of the main tasks. For each task, the interviewee names the involved persons and the required information (input), the working products (output), the supporting artifacts such as templates, standards or tools [13].

The results of the FLOW interviews are visualized in a FLOW diagram. Figure 1 presents the main information stores (faces for fluid stores and documents for solid stores). The rectangle represents an activity summarizing parts of the diagram that are too fine-grained for the visualization or that are not further defined during the interviews. Note that the FLOW diagram is a directed network since information mainly flows from one person to another, even if the interaction (mostly communication) is usually bidirectional. If an information flow is bidirectional, i.e. information flows from A to B as well as from B to A, this is visualized by an arrow pointing in both directions.

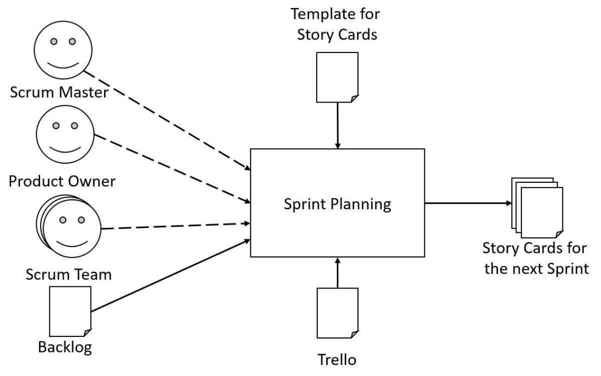


Figure 1: Exemplary FLOW diagram [20]

Figure 1 visualizes the activity *Sprint Planning* with incoming information from the *Scrum Master*, the *Product Owner*, the *Scrum Team* and the *Backlog*, which is usually documented on a task board or in a ticket system and hence solid. The result of the task are *Story Cards for the next Sprint*. The *Template for Story Cards* controls the task and *Trello*[1] supports the activity.

FLOW diagrams mostly comprise various patterns. During the analysis, these patterns need to be found in order to detect weaknesses. For example, a long chain of fluid information stores (also referred to as *Chinese-whisper-pattern*) [21] should be avoided since the information may change during the transfer due to misunderstandings. Another example for a pattern is the *Competence Spider* [22]. This is a person that receives and shares a lot of information. An absence of such a person endangers the information flow.

Afterwards, the analyst presents his findings and discusses possible improvements with the project team. He further suggests how the team can use its resources more appropriately.

IV. Quantifying Information Flow

In order to support the FLOW analyst during the analysis as well as to allow project leaders and team members to analyze the FLOW diagram, we want to support the analysis with quantitative measures and visualizations. In order to implement our metaphor of a pipe system, we have to make some assumptions which we present in this section.

A. *Weighting Information Flow*

In a first step, we define the diameter of the pipes which are given by weights for the edges. The weights represent the amount of information flowing between two nodes [23]. A small weight defines a small flow capacity whereas a large weight defines a large flow capacity.

Considering a FLOW diagram, we have to distinguish between four cases without activities and four cases with

[1]https://trello.com/

activities: Each combination of fluid and solid stores (solid $\rightarrow$ solid, solid $\rightarrow$ fluid, fluid $\rightarrow$ solid, fluid $\rightarrow$ fluid) and combinations with an activity (fluid $\rightarrow$ activity, solid $\rightarrow$ activity, activity $\rightarrow$ fluid, activity $\rightarrow$ solid). We define the amount of information flow between two information stores according to their states as represented in Table I. We define the edge weights representing the amount of information flow between two nodes to range between 0 and 1, where 0 represents no information exchange.

Table I: Edge weights representing the amount of information flow between two nodes in developer networks

Source	Receiver	Weight
Fluid	Fluid	0.7
Fluid	Solid	0.5
Solid	Fluid	0.4
Solid	Solid	0.2

The weights in Table I require some assumptions:

- *fluid $\rightarrow$ fluid:* Transporting information directly and personally enables a good flow of information because the receiver is able to directly ask questions in order to understand what the source tells him.
- *fluid $\rightarrow$ solid:* Information sharing from a fluid store to a solid one like writing something down, is worse than information sharing between persons, since the source cannot write everything down or may forget implicit knowledge. Hence, the receiver has less information than the source.
- *solid $\rightarrow$ fluid:* Reading is one example for transforming information from a solid store to a fluid one. The amount of information received by reading the document strongly depends on the receiver's knowledge. Missing or not understandable information may not be retrieved by reading the document twice. The only way of receiving missing specific information is talking to the source of the document.
- *solid $\rightarrow$ solid:* Information sharing from a solid store to another solid store is given by copying documents or changing the file format. However, since there is no human involved in the process, we assume a very low "collaboration" between these documents.

These assumptions imply the decreasing weights in Table I. In the case of information flows with involved activities, we have to make different assumptions. We know either the source or the receiver, but one of them is an activity, which is, first of all, a black box. Hence, we do not know the granularity of the information flow, i.e. if it is solid or fluid. Sometimes, knowledge about the activity helps to decide whether an information flow is solid or fluid, but in many cases, this decision is not possible.

We consider information flows within activities as the worst case and choose the minimum edge weight which is given by the information flow between two solid stores, i.e. 0.2, in case of doubts. We assume that information flows during each activity, since according to Watzlawick et al. "one cannot *not* communicate" [24]. However, it is also possible to reduce the activities as described in the following.

B. FLOW Diagrams as Pipe Systems

We aim at applying the Edmond-Karp-Algorithm of the Ford-Fulkerson-Method for calculating the maximum flow within a network to the FLOW diagram. In order to apply this algorithm, we first have to adjust the FLOW diagram, since the algorithm only considers nodes and edges, but no activities as in FLOW. As presented by Kiesling et al. [13], there are mainly three possibilities of transforming a FLOW diagram into a network:

1) Connect all incoming and outgoing stores of the activity.
2) Represent the activity with a separate node and connect all incoming and outgoing nodes with this one by preserving the direction of flow.
3) Decide for each incoming node whether it should be directly connected to an outgoing one or to a separately defined one. This case requires deeper insights into the activity.

There are some prerequisites deciding which case is most suitable.

- Case 1 implies that there is an information flow from each incoming to each outgoing node involved in the activity. For example, in case of a meeting, this might be correct. However, activities often represent a further communication network where the information does not necessarily flow from each incoming to each outgoing node.
- Case 2 is the most neutral way of integrating an activity within a network. After having analyzed the network, it might be possible to re-transfer the network into a FLOW diagram and adapt the results for analyzing the diagram. In case of doubts, we recommend using this possibility.
- Case 3 is the most exact way of dissolving an activity. This case requires deep insights into the activity that cannot always be achieved.

Depending on the choice of the case, there are different interpretations of the resulting network. In order to connect both approaches, i.e. the FLOW analysis and the algorithmic extension, it is required not to change the core statements of the network. Hence, the transformation method needs to be carefully selected for each activity.

After this step, we receive a FLOW network only consisting of information stores and information flows, i.e. nodes and edges. We are now able to apply the algorithm to the network.

C. Quantitative Measures

Based on the pipe system and the use of the Edmond-Karp-Algorithm, we calculate the maximum flow between two nodes. Comparing the results between different nodes and considering the paths of the maximum flow helps to detect central persons, i.e. persons who are very important for the information flow. A person can be central from different viewpoints. Social network analysis is a wide field providing

Table II: Overview of centrality measures and their importance for information flow analysis [13]

Centrality Measure	Influenced by	Persons with a high degree...
Degree Centr.	incoming resp. outgoing edges	receive resp. share a lot of information
Closeness Centr.	the average distance to each other node	obtain novel information early
Betweenness Centr.	the location on the shortest paths in the network	need to share many urgent information
Flow Betweenness Centr.	the location on all paths in the network	coordinate the information flow
Eigenvector Centr.	the number of neighbors who are also central	can share important information in a very short time with the whole network

different measures for centrality [23]. Kiesling et al. [13] apply commonly used centrality measures to the nodes of a FLOW diagram. Table II summarizes some centrality measures and explains their relevance for information flow analysis. All of these centrality measures are local ones, i.e. they can be calculated for a single node.

These measures help to detect critical issues in the network [13]. Some of them are obvious in the FLOW diagram, e.g. the degree centrality. One only needs to count the incoming and outgoing edges of a node and the nodes that have a lot of them are central in the sense of degree centrality. The persons represented by the nodes seem to have a certain importance for the information flow process. However, other measures such as the closeness centrality are difficult to identify in a network without having calculated the measures. We provide a visualization for the closeness centrality which measures the average distance of a node to all other nodes in the network in order to facilitate understanding of persons who are central in the sense of closeness centrality. We call this visualization *network expansion* since we highlight all nodes that can be reached by a certain node after 1, 2, 3, or more steps. The more nodes a node can reach within a few steps, the higher the closeness centrality. It remains future work to provide visualizations for the other centrality measures that are not easy to determine.

D. Analyzing the resulting network

We implemented a prototype depicting the transformed FLOW diagram and then calculating the maximum information flow from a source to a target. Figure 4 represents a screenshot of the tool. All fluid information stores are visualized as circles and all solid stores are rectangles. Currently, activities are represented as squares but have the same properties as the other nodes. The distinction between the information stores only supports the intuitive comprehension of the network; the algorithm does not differ between the type of information because this information is already contained in the pipe diameter. In Figure 4, the purple node in the upper left corner is the source and the orange node in the lower right-hand corner is the target, i.e. the receiver of the node. The blue path visualizes the best ways for information sharing according to the Edmond-Karp-Algorithm.

The tool also includes the visualization of the *network expansion* (see Figure 3) and the calculation of the centrality measures in Table II. To facilitate the interpretation of the centrality measures, our tool is able to highlight the extreme values, i.e. those differing from the other ones.

V. Case Study in Industry

We present a preliminary study with a globally distributed software engineering company [22] to demonstrate the applicability of our approach.

A. FLOW Analysis

According to the proceeding presented in Sec. III-B, we have interviewed the director of the company. In this interview, we gained profound knowledge about the information sharing behavior. The FLOW diagram after the first interview is rather small and clear. In order to present the proceeding of our quantitative analysis, it is sufficient and better to use a rather small, but clear FLOW diagram.

In the first interview, we collected data about the overall corporate structure, the hierarchies and the communication behavior within and across the teams. We talked to the director who has a good overview of all processes. The whole interview took about two hours. It started with some demographics about the director, his experiences and his background. Afterwards, he started describing the overall process. We took notes on the main activities and asked him about the results, incoming information, supporting tools and persons, and controlling elements such as templates. In the end, we summarized the results of the interview, i.e. the information on the main activities during the process to avoid misunderstandings.

We visualized the findings based on the interviews as a FLOW diagram. The result of the interview is presented in Figure 2 [22]. The developers work in Spain, while the customer and the consultants are in Germany. To coordinate the communication across borders, each team has a team leader who exchanges information with the other team leader. A project starts with a workshop (see (1) in Figure 2). The director, the customer and both team leader participate in the workshop. It helps to clarify basic conditions. The workshop results in a requirements catalog (2) documenting all requirements. Based on this requirements catalog, the consultants write story cards for the developers in Spain and create a concept. A click-dummy (3) basically visualizes the idea for the final product. The developers implement the story cards and regularly exchange information with the consultants (4) in Germany, who are in contact with the customer. In the end, the customer receives the remaining software product to test it and to express change requests. This step is not visualized in Figure 2.

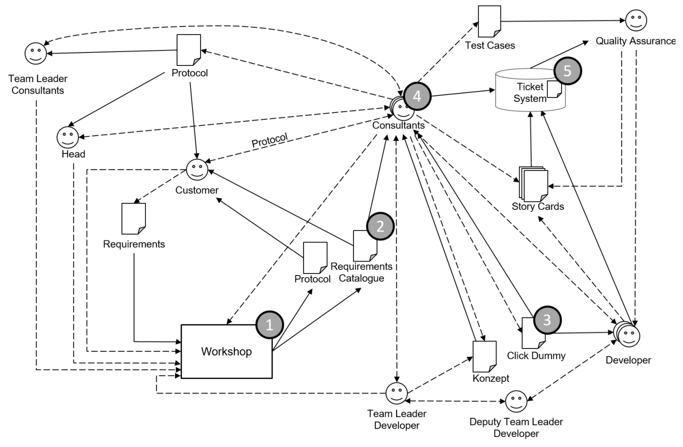


Figure 2: FLOW diagram after the first interview [22]

B. *Tool-Supported Analysis*

We consider exemplary the information flow from the team lead (purple node) to the developers (orange node) in Figure 4. The workshop, the requirements catalog and the click-dummy lay on the path of the maximum flow from the team lead to the developers. The most important node are the consultants because they lay on each of the paths. If this node is missing, information would flow worse or even not at all from the team leader of the consultants to the developers. The centrality measures of this node (closeness: 0.08, betweenness: 0.84, in-degree: 7, out-degree: 12) also support this fact. All of them are highlighted, i.e. they differ remarkably from the measures of all other nodes. The workshop (betweenness: 0.60, in-degree: 6, out-degree: 2) and the developers (betweenness: 0.49, in-degree: 4, out-degree: 4) are also very important.

Considering the information expansion of the consultants (central node) and the quality assurance (decentral node) in Figure 3 illustrates the benefit of this measure. Figure 3a visualizes the information expansion of the consultants that only need 2 steps to cover the whole network. Figure 3b presents the information expansion of the quality assurance which is rather decentral. This node needs 4 steps to cover the whole network. The main reason for this finding is that the quality assurance only talks directly to the developer which can also be found in the network.

VI. Discussion

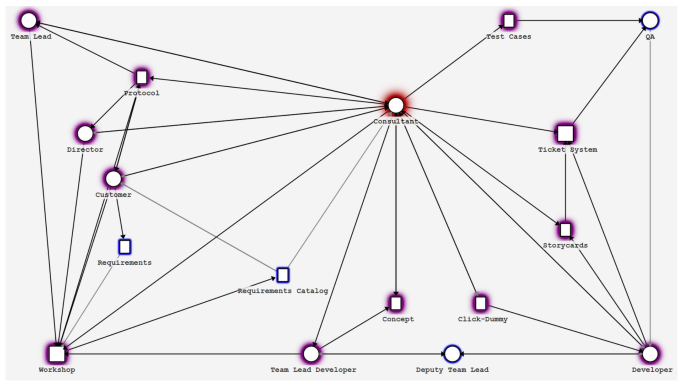
In the following, we reflect on the limitations of our approach before interpreting and discussing our findings from the example above.

A. *Critical Appraisal*

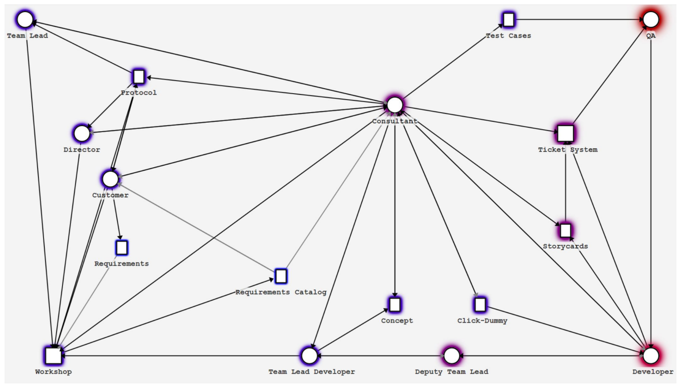
Due to the basically conceptional approach in this paper, the presented idea has two deficiencies, which are not unimportant for the interpretation and the reliability of the results after the application of our approach: (1) the choice of the algorithm and (2) the weightings of the information flow.

(1) The choice of the algorithm: There exist many different algorithms for calculating the maximum flow in a network. Hence, there might be a better algorithm for our approach. We decided to use the Ford-Fulkerson-method which is widely distributed in graph theory and which is even recommended by Cormen [18] for considering information flows in a network.

(2) The weightings of the information flows: The proposed weightings in Table I underly some assumptions presented in subsection IV-A. For calculations, we had to postulate concrete values. Although the assigned values seem arbitrary, they are chosen carefully based on the assumptions presented in subsection IV-A. But slightly different values with a comparable scaling do not lead to completely different results. The range from 0 to 1 is based on the calculation of FLOW distance, which defines the amount of information flow in smaller teams and also ranges from 0 to 1 [4]. Nonetheless, the values represent the current state of our research and may need to be adjusted later due to new findings. However, the basic idea remains valid also with slight adjustments. Nonetheless, we do not consider human factors influencing the information flow such as a node's absorption of information. The ability of a person to absorb and internally handle information can increase or decrease the amount of information to be shared

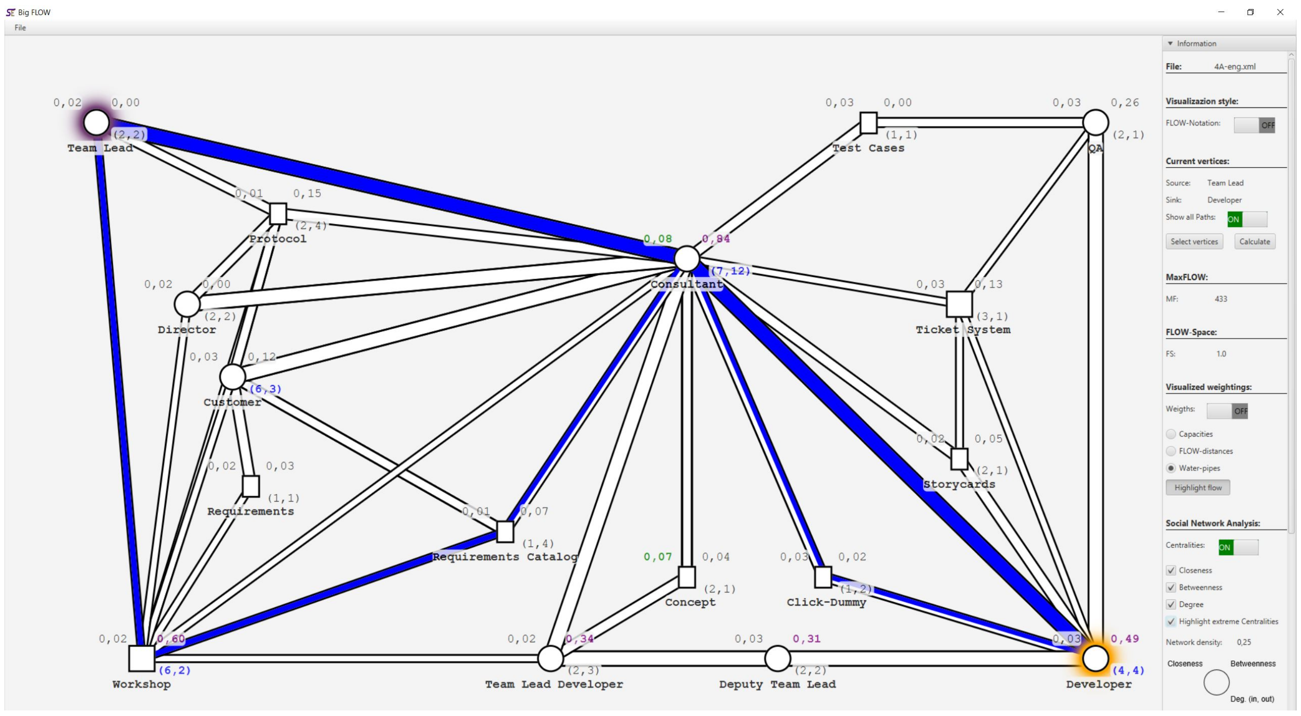


(a) Information expansion of a central node



(b) Information expansion of a decentral node

Figure 3: Stepwise information expansion. The colour of the starting node is red, turning towards blue with each additional step. Furthermore, the diameter of the coloured circle around the node decreases.



File: 4A-eng.xml

Visualizazion style:
FLOW-Notation: OFF

Current vertices:
Source: Team Lead
Sink: Developer
Show all Paths: ON

MaxFLOW:
MF: 433

FLOW-Space:
FS: 1.0

Visualized weightings:
Weigths: OFF
Capacities
FLOW-distances
[✓ Water-pipes]

Social Network Analysis:
Centralities: ON
✓ Closeness
✓ Betweenness
✓ Degree
✓ Highlight extreme Centralities
Network density: 0,25

Figure 4: Screenshot of our prototype calculating the maximum information flow between the purple (source) node and the orange (receiver) node as well as common centrality measures

with other persons. Furthermore, we do neither consider the loss of information nor misunderstood information that are falsely shared. We will consider these aspects, in particular ”personalized” weightings, in further research.

B. *Interpretation*

Information flow in software development teams or companies is a very complex behavior which requires a manual analysis. Some structures or patterns in the FLOW diagram are too interwoven and subliminal to be detected by a simple algorithm.

We want to support the information flow analysis and increase the developers’ awareness of the complexity of information sharing by presenting an approach of metaphorically considering a developer network as a pipe system. Hence, we quantify information flow analysis to apply the algorithm. This helps to increase the trustworthiness of the findings due to underlying data. Furthermore, it decreases the possibility of missing interpretations and forgotten obvious things. It helps to make the subjective analysis more objective.

Our case study reflects the advantages of the interplay of both qualitative and quantitative information flow analysis. The qualitative method finds complicated structures and patterns such as the “Chinese-whisper-pattern”. The quantitative analysis detects the same issues and – after adaptions and extensions – it may help to increase the developers’ awareness for the current state of information flow by visualizing and simulating the results. It currently detects ways of information flow that are too long to be suitable or that are duplicated. However, the analysis needs to be sharpened and extended to gain real benefits from the quantifications.

In the given example, we figured out the importance of the consultants within the process. A FLOW analyst recognizes the consultants as so-called ”competence spider” because they are involved in many information flows. As evident from the FLOW diagram, there are many incoming and outgoing edges from and to the consultants. Our pipe system supports the relevance of the consultants. The underlying data confirm that the consultants are at least mandatory for a fast information flow from the team leader to the developers. Without the consultants managing and coordinating the information sharing, an information flow from the team leader to the developers cannot be guaranteed. Furthermore, the centrality measures and the information expansion underline the importance of the consultants objectively. This objectivity may also help to increase the awareness of the involved persons. A qualitative analysis is always subjective, but the calculated measures are objective and the results easier to comprehend. However, there is some analysis required to support this assumption. The qualitative and the quantitative method supplement each other. Nonetheless, the quantitative method needs to be refined and extended to increase the advantages and to evaluate the practicability.

VII. Conclusion and Future Work

Information exchange is important in software development teams. Furthermore, information needs to be shared between the customer, the team, project leader and many other different involved persons.

One strategy to analyze information flow is the FLOW method which aims at observing, examining, and improving information flow in teams and companies. Until now, the FLOW method is a basically subjective method and the results depend on the analyst's experience and knowledge.

In this paper, we presented the conceptional idea of metaphorically considering the developer network as a pipe system. We defined the amount of information flow between two nodes in the network based on the states of information flow provided by FLOW. Information flow can then be analyzed based on established algorithms in graph theory. In this approach, we use the Ford-Fulkerson-method which is widely used to calculate the maximum flow in a flow network.

We applied our approach to a software development company which results after the first interview have already been analyzed corresponding to the FLOW method [22]. We were able to detect some findings which are rather obvious (e.g. the important role of the consultants) and have also been retrieved in the qualitative analysis. However, we can objectively support these findings by presenting centrality measures.

In future, we will evaluate the influence of our approach on the developers, i.e. if our approach increases the awareness for the relevance of information sharing. Furthermore, we plan to graphically simulate information flow in the network and include further metrics to measure the quality of information flow. Additionally, our automated approach with pipe systems is still based on a manual elicitation phase of the information flows. This phase consists of interviews with project members and drawing a flow diagram. This is highly time-consuming and requires an experienced analyst. In order to enable software companies to conduct a flow analysis on their own without any expertise, we have to automate the elicitation phase of the flow analysis as well. At the moment, we deal with this problem in future and ongoing research.

Acknowledgment

This work was supported by the German Research Foundation (DFG) under grant number *263807701* (project TeamDynamics, 2018-2020).

References

[1] B. Al-Ani and H. K. Edwards, "A comparative empirical study of communication in distributed and collocated development teams," in *Proceedings of the 3rd IEEE International Conference on Global Software Engineering*. IEEE, 2008, pp. 35–44.

[2] T. Wolf, A. Schröter, D. Damian, and T. Nguyen, "Predicting Build Failures Using Social Network Analysis on Developer Communication," in *Proceedings of the 31st International Conference on Software Engineering*. IEEE Computer Society, 2009, pp. 1–11.

[3] J. D. Herbsleb, H. Klein, G. M. Olson, H. Brunner, J. S. Olson, and J. Harding, "Object-oriented analysis and design in software project teams," *Human–Computer Interaction*, vol. 10, no. 2-3, pp. 249–292, 1995.

[4] J. Klünder, K. Schneider, F. Kortum, J. Straube, L. Handke, and S. Kauffeld, "Communication in Teams - An Expression of Social Conflicts," in *Proceedings of the 6th International Conference on Human-Centered Software Engineering and 8th International Conference on Human Error, Safety, and System Development*. Springer International Publishing, 2016, pp. 111–129.

[5] T. Niinimäki, A. Piri, and C. Lassenius, "Factors affecting audio and text-based communication media choice in global software development projects," in *Proceedings of the 4th IEEE International Conference on Global Software Engineering*. IEEE, 2009, pp. 153–162.

[6] D. Damian, S. Marczak, and I. Kwan, "Collaboration patterns and the impact of distance on awareness in requirements-centred social networks," in *Proceedings of the 15th IEEE International Requirements Engineering Conference*. IEEE, 2007, pp. 59–68.

[7] J. D. Herbsleb, "Global Software Engineering: The Future of Socio-Technical Coordination," in *Future of Software Engineering*. IEEE Computer Society, 2007, pp. 188–198.

[8] E. Bjarnason, K. Wnuk, and B. Regnell, "Requirements are slipping through the gaps a case study on causes & effects of communication gaps in large-scale software development," in *Proceedings of the 19th IEEE International Requirements Engineering Conference*. IEEE, 2011, pp. 37–46.

[9] B. Bruegge, A. H. Dutoit, and T. Wolf, "Sysiphus: Enabling informal collaboration in global software development," in *Proceedings of the International Conference on Global Software Engineering*. IEEE, 2006, pp. 139–148.

[10] K. Stapel and K. Schneider, "Managing Knowledge on Communication and Information Flow in Global Software Projects," *Expert Systems*, vol. 31, no. 3, pp. 234–252, 2014.

[11] B. S. Caldwell and N. C. Everhart, "Information flow and development of coordination in distributed supervisory control teams," *International Journal of Human-Computer Interaction*, vol. 10, no. 1, pp. 51–70, 1998.

[12] K. Schneider, K. Stapel, and E. Knauss, "Beyond documents: visualizing informal communication," in *Proceedings of the 3rd International Workshop on Requirements Engineering Visualization*. IEEE, 2008, pp. 31–40.

[13] S. Kiesling, J. Klünder, D. Fischer, K. Schneider, and K. Fischbach, "Applying social network analysis and centrality measures to improve information flow analysis," in *Proceedings of the 17th International Conference on Product-Focused Software Process Improvement*. Springer International Publishing, 2016, pp. 379–386.

[14] C. Durugbo, A. Tiwari, and J. R. Alcock, "Modelling information flow for organisations: A review of approaches and future challenges," *International Journal of Information Management*, vol. 33, no. 3, pp. 597–610, 2013.

[15] M. Benson-Rea and S. Rawlinson, "Highly skilled and business migrants: Information processes and settlement outcomes," *International Migration*, vol. 41, no. 2, pp. 59–79, 2003.

[16] G. Smith, *On the Foundations of Quantitative Information Flow*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 288–302.

[17] G. Lowe, "Quantifying information flow," in *Proceedings of the 15th IEEE Computer Security Foundations Workshop*. IEEE, 2002, pp. 18–31.

[18] T. Cormen, *Introduction to Algorithms*. MIT press, 2009.

[19] K. Stapel, E. Knauss, and K. Schneider, "Using FLOW to improve communication of requirements in globally distributed software projects," in *Collaboration and Intercultural Issues on Requirements: Communication, Understanding and Softskills*. IEEE, 2009, pp. 5–14.

[20] J. Klünder, C. Unger-Windeler, F. Kortum, and K. Schneider, "Team meetings and their relevance for the software development process over time," in *Proceedings of Euromicro Conference on Software Engineering and Advanced Applications*, 2017.

[21] K. Schneider and O. Liskin, "Exploring flow distance in project communication," in *Proceedings of the 8th International Workshop on Cooperative and Human Aspects of Software Engineering*. IEEE Press, 2015, pp. 117–118.

[22] J. Klünder and K. Schneider, "Information Flow in Distributed Software Projects – A Case Study (orig.: Informationsfluss in verteilten Software-projekten - Eine Einzelfallstudie)," in *PERSONALquarterly, 69(2)*, 2017, pp. 10–15.

[23] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*. Cambridge university press, 1994, vol. 8.

[24] P. Watzlawick, J. B. Bavelas, and D. D. Jackson, *Pragmatics of human communication: A study of interactional patterns, pathologies and paradoxes*. WW Norton & Company, 1967.