

Identification patterns of Community Smells

This chapter presents one fundamental contribution of this master thesis, which is the specification of *identification patterns of five Community Smells* considered in the scope of this study, namely: Organisational Silo Effect, Missing Links, Black-cloud Effect, Prima-donnas Effect and Radio Silence.

We considered the set of Community Smells identified within an industrial case study, conducted by Tamburri et al. [6], as a starting point and then we proposed identification patterns, capable of capturing possible social related problematic situations, for five different Community Smells. Proposed *identification patterns can be operationalised in order to provide an automated identification, quantification and recovery of Community Smells*, within a software development community, by means of tool-supported processes or software-repository mining techniques.

Every identification pattern of a Community Smell proposed within this chapter is founded on the concept of *Developer Social Networks*, in order to retrieve information about communication and collaboration relationships within a software development community. Therefore, the identification process of every Community Smells is enabled by the availability of the communication and collaboration Developer Social Networks of a development community.

While the identification patterns of four Community Smells base their discovery logic solely on the state of a development community in a particular point in time, one identification pattern founds its detection capabilities on the additional consideration of historical information, i.e., it considers more than one temporal window.

Community Smells, based on the exploitation of identification patterns presented in this chapter, it is necessary to have access to the VCS and mailing list archives of a software development community, or to any other typology of development artefact capable of providing information related to organisational structure and technical dependencies of a software project.

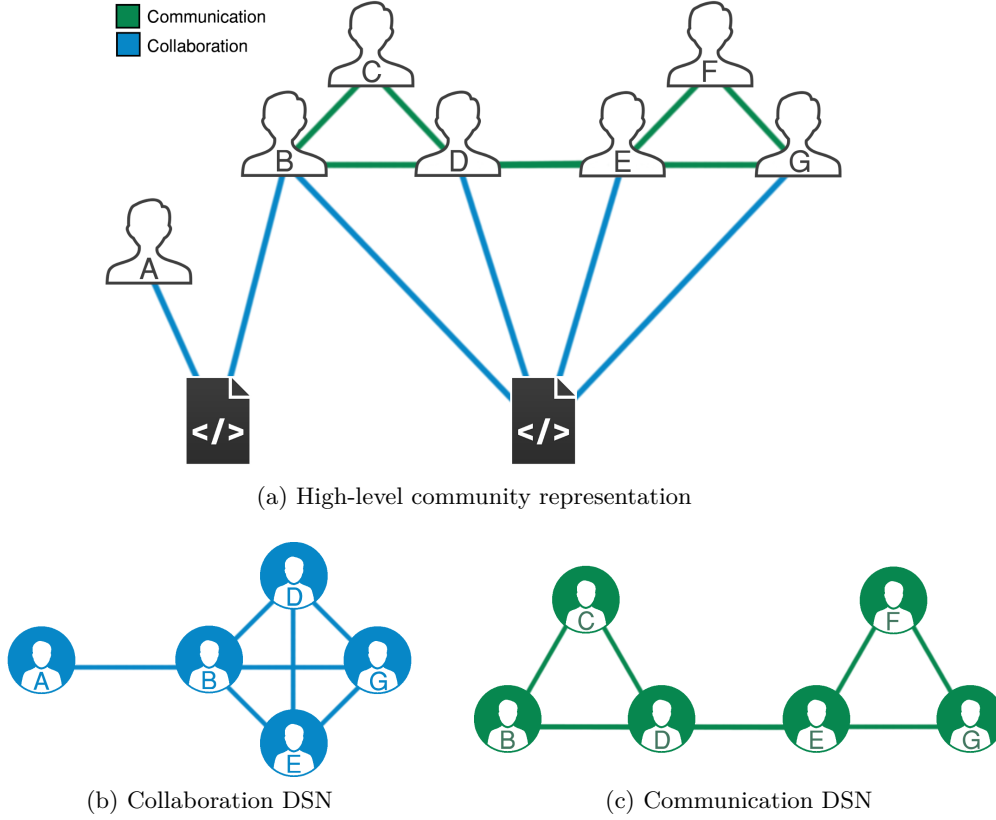


Figure 4.1: Example of software development community

An explicative example of a software development community is represented in Figure 4.1a, where it is possible to notice the presence of eight community members, identified by different letters, who are connected through green links if they communicated within the mailing list and community members are connected through blue links to software components on which they worked (represented by a file icon). Given such high-level representation of a software development community, it is possible to generate the communication DSN, representing the social and organisational structure of a project, and the collaboration DSN, representing technical relationships between developers working on similar portions of a project's source code. While the communication DSN of the proposed example, represented in Figure 4.1c, can be easily generated considering exclusively communication links of the high-level representation of the development community, the generation of the collaboration DSN, represented in Figure 4.1b, requires additional reasonings because

two developers are linked if and only if they worked on a similar portion of source code. Thus, since the file icon in the high-level representation of the development community identifies a similar portion of source code, all developers linked to the same source code portion have to be connected in the collaboration DSN.

Within the proposed example it is possible to consider the communication DSN constituted by two different sub-communities: developers identified by the letters B, C and D and developers identified by the letters E, F and G. It is important to highlight that a Developer Social Network is constituted only by *active* community members, with respect to the considered development aspect. Therefore, it is possible to notice that, within the previously exposed example, the collaboration DSN do not contain community members identified by the letters C and F because they are not working on any source code portion and the communication DSN do not contain the community member identified by the letter A because he or she is not communicating with anyone within the development community.

The following sections address all the Community Smells considered in the scope of this master thesis. For each Community Smell it is provided an identification pattern and a synthetic example explaining the behaviour of the identification pattern. Furthermore, each section presents a detailed explanation of the execution of the considered identification pattern within the development community presented in Figure 4.1a and a visual representation of identified occurrences of Community Smells within such example.

4.1 Organisational Silo Effect and Missing Links

Tamburri et al. [6] defined Organisational Silo Effect Community Smell as the occurrence of a too high level of decoupling between the organisational structure and development activities. Considering its definition and the literature based on Conway’s law, it was possible to assimilate this Community Smell to an *imperfection in the mirroring relationship between the organisational structure and the technical structure*, constituted by collaborations of software developers.

An Organisational Silo is characterised by an independentist sub-community that is loosely dependent to other development partners and that duplicates or wastes resources over the development life cycle, due to its isolation from the rest of the software development sub-communities.

This Community Smell is associated to the main side effect of decreasing communication activities between community members belonging to different sub-communities; thus, it negatively impacts on mutual awareness and it implies a degradation of the socio-technical congruence. Another side effect that can characterise this Community Smell is the rise of a “tunnel vision” within the software development community, since members belonging to the Organisational Silo tend to limit their coopera-

tion and communication activities with members who do not belong to their sub-community. Moreover, community members belonging to an Organisational Silo can exhibit egoistic and superior behaviours, giving rise to autonomous architectural decisions taken without the necessary authority or knowledge. A possible mitigation to this Community Smell can be the implementation of a “social wiki”, that combines documentation, developer profiles and artefacts [6].

While we were pursuing our goal of identifying a possible automatic identification pattern of this Community Smell, mining commonly available software development artefacts, we focused of the most important Organisational Silo Effect side effects: decrease of communications within the community and generation of a “tunnel vision”. Therefore, we proposed two different identification patterns in order to provide the ability of identifying both typologies of side effects:

1. Organisational Silo Effect

The identification pattern of Organisational Silo Effect Community Smell is based on the detection of *community members who collaborate with other members but who do not communicate* within the analysed communication channel, i.e., mailing list. An explanatory example is represented in Figure 4.2, where the occurrence of this Community Smell is due to the developer identified by the number 1 because he or she does not have outgoing communication edges even if there is a collaboration with the developer identified by the number 2.

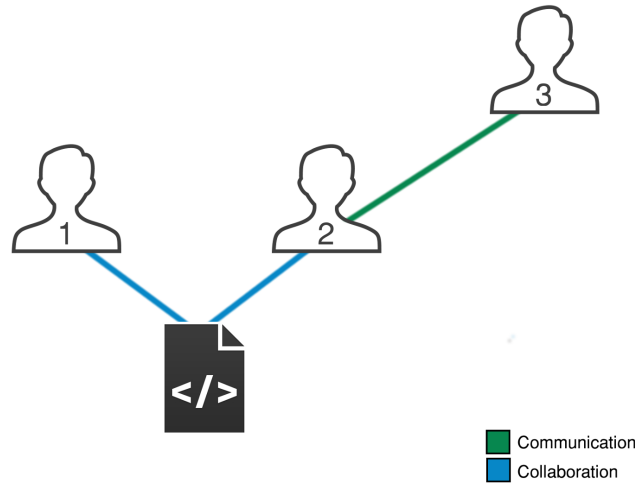


Figure 4.2: Identification pattern of Organisational Silo Effect

This identification pattern is capable of identifying the presence of actual sub-communities present within the software development community that generate a “tunnel vision” because they do not communicate with community members external to their sub-community. Pushing to its extreme the concept

of sub-community, thus considering two developers at a time, an occurrence of *Organisational Silo Effect* is detected whenever one of them shows *unco-operative behaviours, not participating in the mailing list of the development community*.

In order to operationalise the proposed identification pattern, the collaboration and communication Developer Social Networks, generated respectively mining the VCS and mailing list archives of a development community, should be considered at the same time and each collaboration present between two different community members contributing to adjacent parts of the source code should be considered as an atomic cooperation unit to identify possible Organisational Silo. Since we considered the absence of one developer belonging to an atomic cooperation unit as the trigger of the occurrence of Organisational Silo Effect Community Smell, it is necessary to consider every software development cooperation present within the collaboration Developer Social Network and check if the two community members are present or not in the communication Developer Social Network. We quantified the number of occurrences of Organisational Silo Effect as the number of single collaborations between different developers in which at least one of them do not participate in the communication channel. Therefore, we focused on collaborations and not on developers, in order to capture the repercussions of the absence of developers in communication activities. This choice was made because the absence of a highly active and highly connected contributor is surely more relevant than the absence of an occasional contributor.

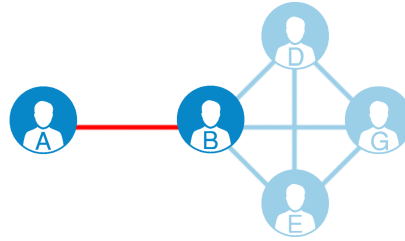


Figure 4.3: Example of occurrences of Organisational Silo Effect

Considering the example of development community proposed in Figure 4.1a, in order to execute the proposed identification pattern of this Community Smell it is possible to compare directly the collaboration DSN (Figure 4.1b) with its communication counterpart (Figure 4.1c) and verify that the developer identified by the letter A is present in the collaboration DSN but he or she is not present in the communication Developer Social Network. Since we focused on collaboration level and not on software developer level, we have to consider every collaboration link between two different community members of the collaboration DSN. The absence of the developer identified by the letter A

is counted only one time since he or she is characterised by a unique outgoing link in the collaboration DSN. Therefore, within the proposed example there is just one occurrence of Organisational Silo Effect Community Smell and a graphical representation is presented in Figure 4.3, where the only occurrence of this Community Smell is highlighted in red and all the developers belonging to the collaboration DSN who are not involved in the insurgence of this Community Smell are faded out.

The operationalisation of this identification pattern should provide a correct quantification of Organisational Silo Effect in cases where both the collaborating developers do not participate within a project's communication channel. Moreover, since we focused on collaborations if in the proposed example, the collaboration Developer Social Network was characterised by an additional developer who was isolated from other developers without having any identified collaboration and he or she was not present in the communication DSN, this additional community member would not have any repercussion on the Organisational Silo Effect detection.

2. Missing Links

The identification pattern of Missing Links Community Smell is based on the detection of *development collaborations between two community members that do not have communication counterparts*. An explanatory example is represented in Figure 4.4, where the occurrence of Missing Links Community Smell is due to the fact that developers identified by the number 1 and 2 are collaborating on a similar portion of source code, but they are not directly connected through a communication link.

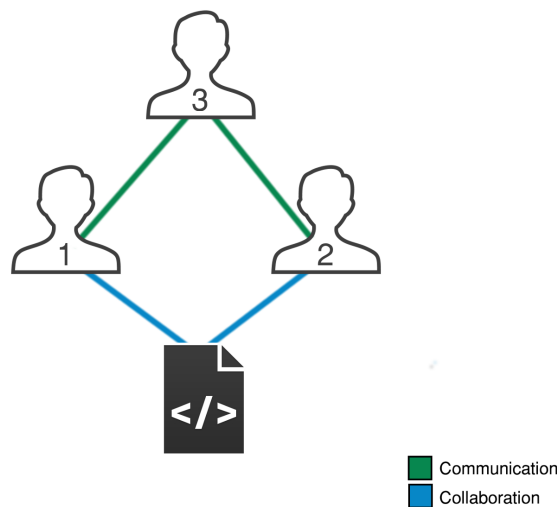


Figure 4.4: Identification pattern of Missing Links

This identification pattern has the purpose of detecting side effects as the decreasing of communication activities, which implies a negative influence on mutual awareness, and the inherent socio-technical congruence degradation caused by the presence of uncooperative Organisational Silo. Thus, while the Organisational Silo Effect identification pattern is focused on the detection of the “tunnel vision” side effect, this identification pattern is supposed to capture all the other typologies of associated side effects. Pushing to its extreme the concept of sub-community, thus considering two developers at a time, an insurgence of Missing Links is detected whenever a couple of co-committing (collaborating) developers exhibit uncooperative behaviours, not communicating in the mailing list of the development community.

Considering the given definition of this identification pattern, it is obvious that Missing Links is a more general case and actually incorporates in itself the identification pattern of Organisational Silo Effect but we considered necessary to define and consider both typologies of identification patterns because they provide different detail levels of information and they characterise and analyse different aspects associated to Community Smells.

In order to operationalise the proposed identification pattern, the collaboration and communication Developer Social Networks, generated respectively mining the VCS and mailing list archives of a development community, should be considered at the same time and each collaboration present between two different community members contributing to adjacent parts of the source code should be considered as an atomic cooperation unit. Since we considered the absence of a communication counterpart of a collaboration as the trigger of the occurrence of Missing Link Community Smell, it is necessary to consider every software development cooperation present within the collaboration Developer Social Network and check if the two community members are connected in the communication Developer Social Network.

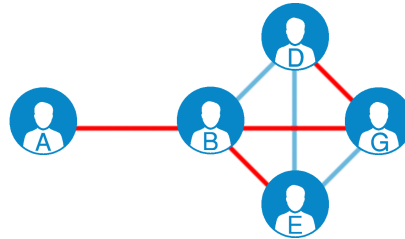


Figure 4.5: Example of occurrences of Missing Links

Considering the example of development community proposed in Figure 4.1a, in order to execute the proposed identification pattern of this Community Smell it is possible to compare directly the collaboration DSN (Figure 4.1b)

with its communication counterpart (Figure 4.1c) and verify that four links within the collaboration DSN do not have corresponding links between the same community members in the collaboration DSN. Detected occurrences of Missing Links are identified in Figure 4.5 with red links and are constituted by the collaborations between developers identified by the following couple of letters: A-B, B-E, B-G and G-D. Using this example it is possible to verify that Organisational Silo Effect is a subset of Missing Links, since the missing communication link associated to the collaboration between developers identified by the letters A and B is actually an Organisational Silo Effect.

In conclusion, since the main purpose of Missing Links identification pattern is to identify and count the number of unsatisfied communication needs due technical collaborations, thus liaising it to socio-technical literature related to Conway’s law, *it is possible to consider Missing Links as the opposite measurement of socio-technical congruence*. While socio-technical congruence is supposed to quantify the accordance of technical and organisational structures, Missing Links quantifies the discordance of such structures and counts the number of the technical relationship that are not mirrored within the organisational structure.

4.2 Black-cloud Effect

The identification pattern of Black-cloud Effect Community Smell is based on the detection of *isolated sub-communities that, in different and subsequent time periods, do not communicate with the exception of one communication link*. An explanatory example is represented in Figure 4.6, where the occurrence of this Community Smell is due to the fact that developers identified by the number 3 and 4 are the only two community members who communicate between the two sub-communities, over time.

Black-cloud Effect Community Smell was initially defined by Tamburri et al. [6] as a social pattern, within a software development community, characterised by the occurrence of two concurrent circumstances:

1. inability of community members to cover knowledge and experience gaps between two different software products developed within the same software community;
2. lack of periodic and official opportunities to share and exchange knowledge between all the community members.

Whenever these two different circumstances occur together, they can generate a “black cloud” within the development community, constituted by confusing and

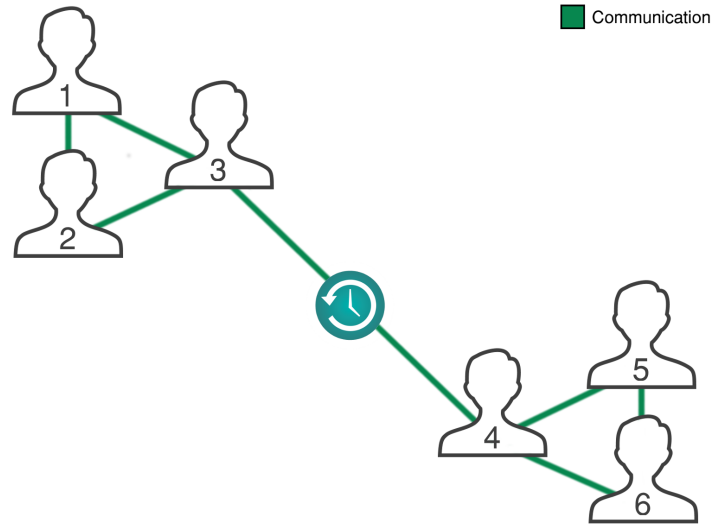


Figure 4.6: Identification pattern of Black-cloud Effect

unnecessary communications that generate a communication overload, which obfuscate the project’s reality and its global and shared vision. The occurrence of this Community Smell can be generated or worsened by the following socio-technical triggers:

- absence of official protocols dedicated to knowledge sharing;
- lack of people with the objective of linking the work of their team with the one of other teams (boundary spanners);
- presence of inefficient communication filtering protocols.

We speculated that the lack of official knowledge sharing opportunities, which constitute one of the two characterising factor of this Community Smell, implies the analysis of formal communication activities, and thus the development mailing list archives of a software development community. On the other side, the inability to cover knowledge and experience gaps between two different software products developed within the same software community suggested a temporal analysis rather than exclusively consideration a snapshot of a development community in time. We used these considerations as a starting point and we defined a methodology to identify possible Black-cloud Effect present within a software development community mining communication relationships between community members, extracted from the communication Developer Social Network, and considering them along different time windows in order to achieve a temporal analysis.

We defined as a “potential” Black-cloud Effect implies that one of the two sub-community is isolated from the rest of the communication DSN external to its boundaries with the exception of a unique communication link from one of its

constituting members toward another community member belonging to a different sub-community. The members who constituted the unique communication link acts as a unique knowledge broker (*boundary spanner*) toward the other sub-community. A “potential” Black-cloud Effect can be considered as an actual Black-cloud Effect Community Smell if and only if the same exact ”potential” Black-cloud Effect is perpetrated in the following time window of analysis.

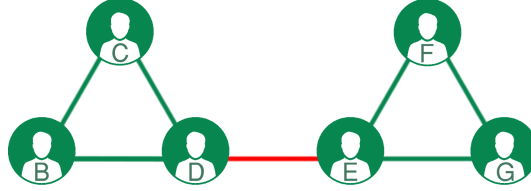


Figure 4.7: Example of occurrences of Black-cloud Effect

After the generation of the project’s communication Developer Social Network it is necessary to classify all community members into clusters, with respect to their communication habits, in order to be able to identify different sub-communities present within the software development communication community. Considering the example of development community proposed in Figure 4.1a, the two sub-communities present within the development community can be easily visualised considering the density of communication links. Analysing the communication DSN (Figure 4.1c), it is evident that the communication link between community members identified by the letters D and E constitutes a unique connection between the two sub-communities and thus community members identified by the letters D and E act as unique knowledge brokers toward the other sub-community. The detected occurrence is represented in Figure 4.7 with a red link, that represents a “potential” Black-cloud Effect and it can be considered as an effective Black-cloud Effect if and only if the same “potential” Black-cloud Effect will be iterated again in the next analysed time window.

4.3 Prima-donnas Effect

The identification pattern of Prima-donnas Effect Community Smell is based on the *detection of isolated sub-communities that cooperate on similar parts of the source code but do not communicate with the exception of one communication link*. An explanatory example is represented in Figure 4.8, where the occurrence of this Community Smell is due to the fact that developers identified by the number 2, 3, 4 and 6 are collaborating on the same portion of source code but the community members identified by the numbers 3 and 4 are the only two community members

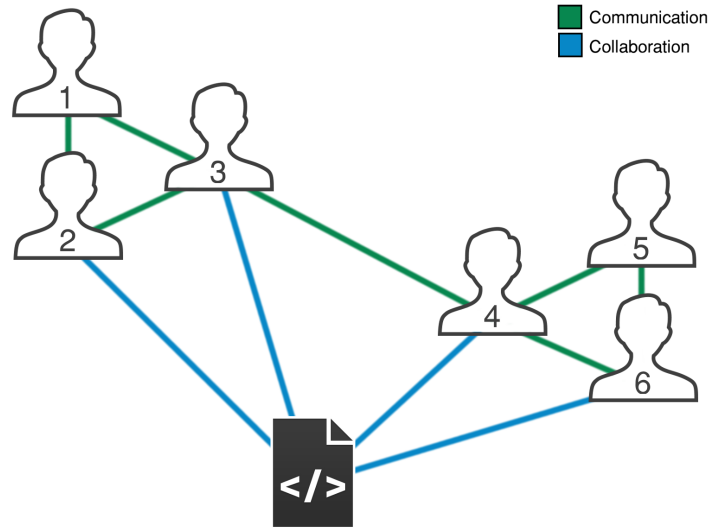


Figure 4.8: Identification pattern of Prima-donnas Effect

who communicate between the two sub-communities.

Prima-donnas Effect Community Smell was initially defined by Tamburri et al. [6] and it is characterised by the presence of a sub-community of developers which is unreceptive with respect to different influences coming from outside its boundaries, unwilling to communicate with external community members, unwilling to welcome support from other members and resistant to imposed organisational or processes changes.

In order to operationalise the proposed identification pattern, the collaboration and communication Developer Social Networks, generated respectively mining the VCS and mailing list archives of a development community, should be considered at the same time and it is also necessary to classify all community members into clusters, with respect to their communication habits, in order to be able to identify different sub-communities present within the software development communication community. The proposed Prima-donnas Effect Community Smell can be subdivided into two identification steps:

1. **Step one:** identification of sub-communities that communicate with each other exclusively through a unique communication link, thus it is the same concept the concept of “*potential*” *Black-cloud Effect Community Smell*;
2. **Step two:** Once potential problematic sub-communities with isolationist behaviours are identified (“potential” Black-clouds) in the previous step, it is necessary to consider the collaboration Developer Social Network and compute the collaboration level of developers belonging to every couple of identified sub-communities. If computed collaboration level is over a given threshold, then a Prima-donnas Effect Community Smell is detected. The threshold used

to decide if two isolationist sub-communities are collaborating, thus if they should be considered as Prima-donnas, is not universally definable because it may depend on the software development context, characteristics and research preferences.

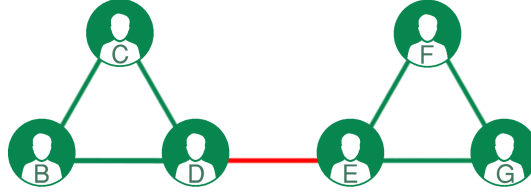


Figure 4.9: Example of occurrences of Prima-donnas Effect

Considering the example of development community proposed in Figure 4.1a, in order to execute the proposed identification pattern of this Community Smell it is necessary consider every sub-community present within the communication DSN (Figure 4.1c) and it is evident that the communication link between community members identified by the letters D and E constitutes a unique connection between the two sub-communities and thus community members identified by the letters D and E act as unique knowledge brokers toward the other sub-community. Then, it is necessary to consider the collaboration relationships between members of every couple of identified isolationist sub-communities within the collaboration Developer Social Network (Figure 4.1b) and it is evident that community members belonging to the two sub-communities are highly connected (full mesh network), so a Prima-donnas Effect Community Smell is detected. The detected occurrence is represented in Figure 4.9 with a red link, that represents the unique link connecting the two Prima-donnas.

4.4 Radio Silence

The identification pattern of Radio Silence Community Smell is based on the *detection of unique knowledge and information brokers toward different sub-communities*. An explanatory example is represented in Figure 4.10, where the occurrence of this Community Smell is due to the fact that the community member identified by the number 3 is the only member of its sub-community who communicates with the other sub-community.

Radio Silence Community Smell was initially defined by Tamburri et al. [6] and it was identified by the occurrence of the following negative characteristics within a software development community: proposed changes require an extraordinary quantity of time to be implemented, time wasting, hidden information, highly

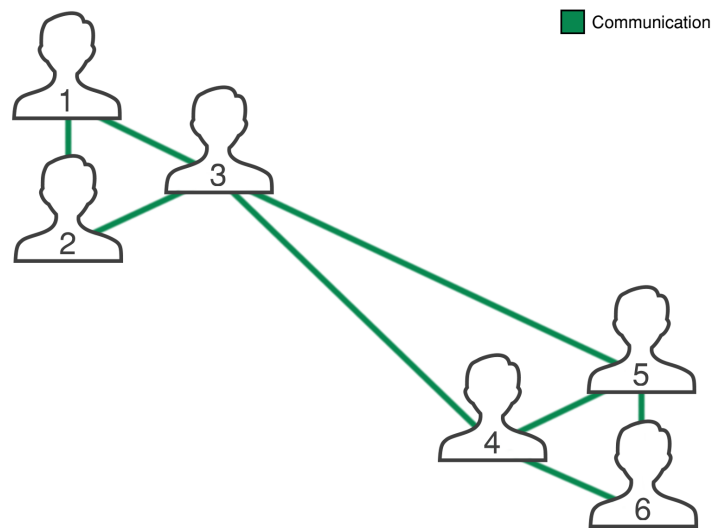


Figure 4.10: Identification pattern of Radio Silence

formal organisational structure, complex organisational structure, and highly regular procedures. The fundamental side effect generated by this Community Smell is a massive delay that characterises decision making processes within a software development community, due to people unavailability or to the communication overload created to cope with further generated information needs.

The presence of a unique boundary spanner toward another sub-community, i.e., a unique knowledge and information broker, implies that every information and knowledge exchange between the two sub-communities needs to be processed by that unique community member. This demanding task requires an additional amount of time from that single community member, generating delays. Moreover, the absence of a unique boundary spanner can stall every formal interaction between two sub-communities.

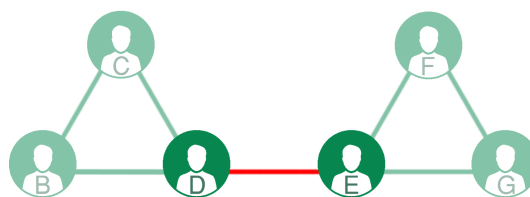


Figure 4.11: Example of occurrences of Radio Silence

Considering the example of development community proposed in Figure 4.1a, since the detection of Radio Silence Community Smell is defined through the concept of unique boundary spanner, it is necessary to identify all community members who are the only members of their sub-community that communicate with another sub-

community. Therefore, we have to consider every inter-communication between different sub-communities.

The decisional process to detect Radio Silence Community Smell using the proposed identification pattern within the proposed example, considering one sub-community at a time, is the following:

- **Sub-community B-C-D:** It communicates with sub-community E-F-G exclusively through community member identified by the letter D, then this is identified as a Radio Silence occurrence and the community member identified by letter D is the knowledge and information broker toward the other sub-community;
- **Sub-community E-F-G:** It communicates with sub-community A-B-C exclusively through community member identified by the letter E, then this is identified as a Radio Silence occurrence and the community member identified by letter E is the knowledge and information broker toward the other sub-community.

The two detected occurrences are represented in Figure 4.11, where knowledge and information brokers are highlighted while all the other community members are faded out. The quantification of the Radio Silence Community Smell that we proposed is focused on the number of different developers who act as boundary spanner toward other sub-communities and not on the number of unique communication links.