# Bug Report Networks: Varieties, Strategies, and Impacts in a F/OSS Development Community

Robert J. Sandusky Les Gasser Gabriel Ripoche
*Graduate School of Library and Information Science*
*University of Illinois at Urbana-Champaign*
*{sandusky,gasser,gripoche}@uiuc.edu*

## Abstract

*Our empirical research has shown that a predominant structural feature of defect tracking repositories is the evolving "bug report network" (BRN). Community members create BRNs by progressively asserting various formal and informal relationships between bug reports (BRs). In one F/OSS bug repository under study, participants assert two formal relationships (duplications and dependencies) and various informal relationships (like "see also" references).*

*BRNs can be interpreted as (1) information ordering strategies that support collocation of related BRs, decreasing cognitive and organizational effort; (2) sense-making strategies wherein BRNs provide more refined representations of software and work-organization issues; (3) social ordering strategies that rearrange collective relationships among community members. This paper presents findings from an investigation of the nature, extent, and impact of BRNs in one large F/OSS development community. We investigate whether and how specific classes of BRNs influence problem management within the community, and identify several new research questions.*

## 1. Introduction

We are conducting empirical investigations into how F/OSS development communities manage software problems. The goal of our research is to develop models of how software problems are managed by large, distributed software development organizations. We aim to identify factors, such as *information*, *activity*, and *process*, which help explain better or worse software problem management (SWPM) performance, with the goal of both understanding such distributed collective practices and improving software production. The early stages of our work include qualitative analysis of the information used and activities performed by members of this community. We use this qualitative analysis to identify concepts, phenomena, and relationships between them as revealed through the examination of the bug reports created and managed by this community. The factors we identify can then be related to each other, hypotheses can be created, and the hypotheses can subsequently be tested in order to isolate the factors that affect SWPM performance. In addition, the seeds created from this human-based mining and analysis can be "computationally amplified," forming the basis of broader automated extraction of process models from very large corpuses of problem data [1].

The negative financial and social impacts of low quality software have been well documented [2, 3]. Previous research on software quality has focused on the development of metrics [4] and defect prediction models [5]. Other research has identified relationships between organizational structure, processes, and quality [6, 7, 8]. The SWPM process itself has been studied less frequently [9]. Our research approach, while grounded in empirical data, acknowledges the contributions of research on process and organizational issues.

Figure 1 shows the main elements of the bug report repository used by one community we are studying. The repository itself is a relational database system and a set of associated scripts that interact with the database and provide a Web-based user interface. The repository contains more than 235,000 records, referred to as bug reports (BRs). Each BR consists of (1) a number of fixed, vocabulary-controlled fields (e.g., status, resolution, severity), (2) several short-length text fields (e.g., keywords, summary), (3) attachments (e.g., screenshots, code patches), (4) a sequential series of text comments that are time-stamped and show the identity of the submitter, and (5) optional indications of relationships between BRs (e.g., duplication, dependency, and informal citations).

One of the most notable structural features of this community's bug report repository is the *bug report network* (BRN). A bug report network is created when members of the software community assert *duplication*, *dependency*, or *reference* relationships among bug reports. Duplication and dependency are both formal, symmetrical types of relationships with an explicit and codified representation in the bug reports. Community

members frequently create informal relationships, like "see also" references, by referring to other bug reports when they are adding text comments to existing bug reports. Sixty-five percent of the bug reports in this repository are associated with other bug reports using one of these three types of relationships.

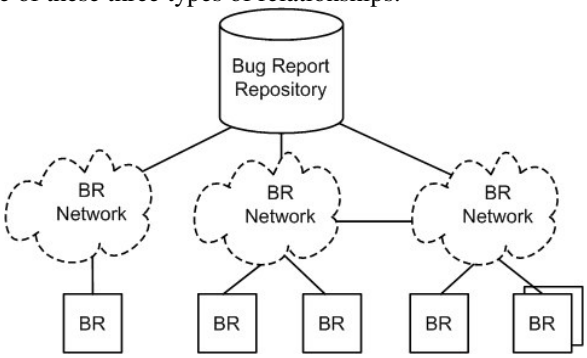


**Figure 1. Bug report repository elements**

Bug reports are first-class database objects, but bug report networks are not. Figure 1 shows three typical BRN patterns. The leftmost BRN represents the 35% of the bug reports that have zero formal or informal relationships to other bug reports: each of these bug reports forms a trivial BRN. The middle BRN shows two bug reports associated by a dependency or informal relationship. The rightmost BRN shows a more complex set of relationships. The doubled bug report on the right represents a bug report with a duplicate relationship to the second bug report behind it. The duplicated bug report is also associated by either a dependency or informal relationship with the other bug report in the network. Note that it is also possible for BRNs to be connected to each other, as indicated by the line connecting the middle and rightmost networks.

We use the following definitions, based upon the definitions stated by the community in their SWPM documentation, to identify the formal relationships between bug reports:

- **Duplicate**: A bug report is marked as a *duplicate* if the problem represented by the bug report is believed to be already represented by another bug report. A duplicate relationship is a formal, symmetrical relationship between two bug reports.
- **Dependency**: A bug report is marked as a *blocker* of another bug report if resolution of the software problem it represents blocks development and/or testing work on the problem represented by the other bug report. A bug report is marked as *dependent on* another bug report if the problem it represents can't be fixed until the problem represented by the other bug report is fixed. Bug reports that are dependent on each other have a formal, symmetrical "blocks" / "depends on" relationship.

Community members also frequently assert informal references between bug reports. While it is possible to automatically extract instances of informal relationships from the repository, the nature and purpose of these references vary considerably and are most reliably understood by reading the bug reports and understanding the contexts in which the citations are made. Here are some examples of these informal references:

```
- This looks related to #X
- See comments on X -- same applies here
  I think.
- My fix for X kinda helps fixing this
  too.
- Should bug X be added to this?
```

## 2. Method

A random sample of 385 BRs was systematically drawn from a population of more than 182,000 bug reports opened over a five year period. The bug report is the primary unit of analysis in this study. The sample size was determined using an approach reported by Powell [10] (p.75). A conservative sample size was suitable here because we did not have complete information about the variability of all characteristics of the bug reports at the time the sample was drawn.

### 2.1. Qualitative analysis

Each bug report in the sample was treated as a text and was read and analyzed using a content-analytic approach [11]. Concepts, phenomena, and relationships between phenomena were identified and refined as they emerged from the bug reports during data analysis using grounded theory [12]. References to other BRs were noted (their location within the BR, reference type, BR serial number) as each BR in the sample was analyzed.

### 2.2. Automatic processing

Another characterization of bug report networks was attempted using automatic extraction of relationships in a snapshot of over 130,000 bug reports originating from the same bug report repository. Two types of relationships were considered:

- *Formal* relationships identified by specific fields ("blocks" and "depends on") or computer generated output inserted as comments in the bug report's discussion.
- *Informal* relationships in the form of references made by participants in their comments (e.g.: "`See bug #X`", "`Looks like bug Y`", etc.), which were mined using regular expressions.

The processing yielded relationship matrices from which BRNs could be identified. However, the automated processing was less accurate than content analysis. When we compared the automatic and manual processes, we found that the automatic process completely identified all the "informally" connected BRs 40% of the time. Also, our current extraction approach does not allow for the distinction of the various types of relationships that are being established. Results from the qualitative analysis are being used to improve the regular expressions used to automatically identify the informal relationships.

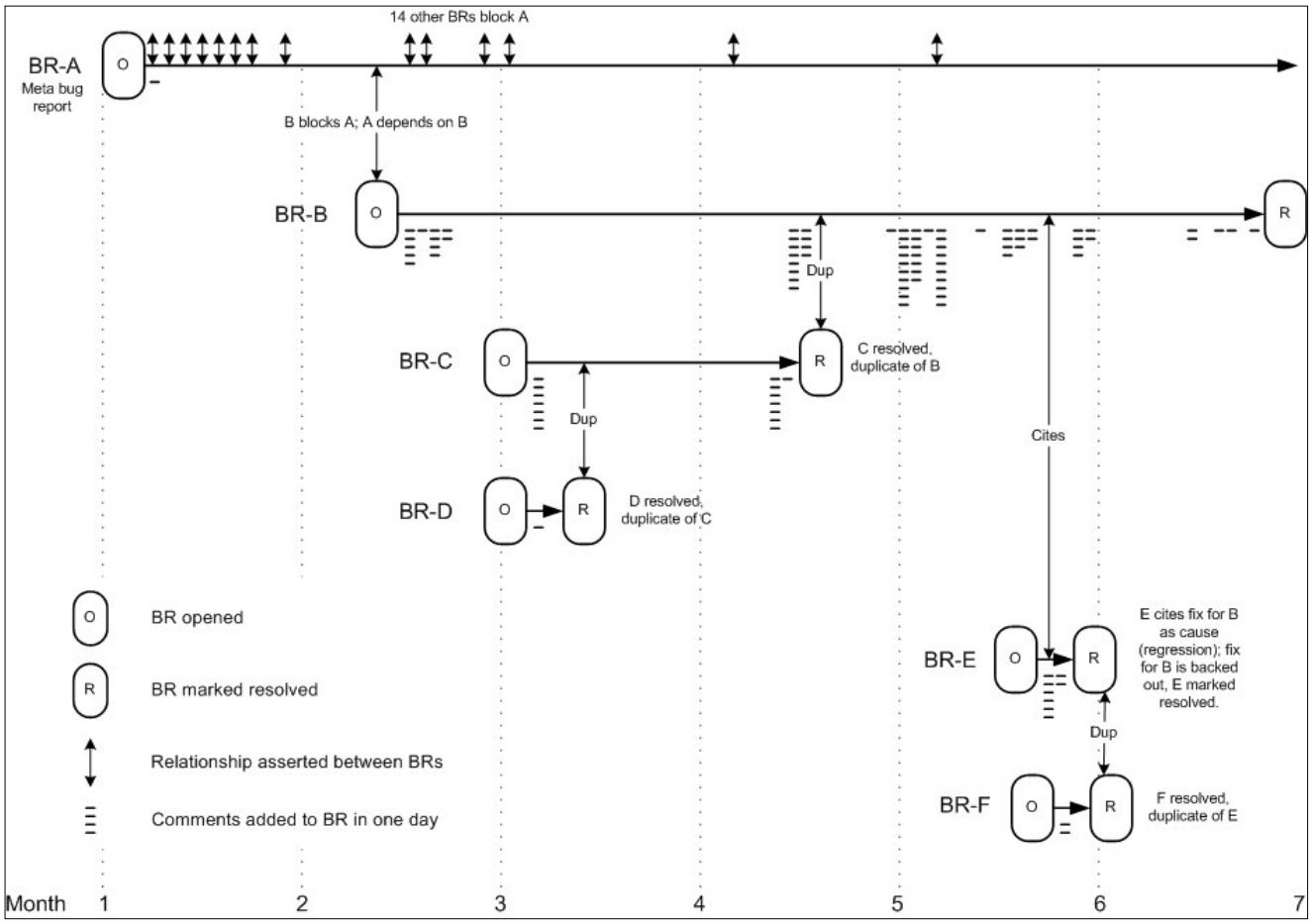


**Figure 2. Bug report network**

## 3. Anatomy of a bug report network

Figure 2 represents the bug report network associated with one "critical" severity bug report drawn from the bug report repository under study. This bug report network, consisting of six bug reports, illustrates a number of different relationships that often occur in this repository. The x-axis represents time over a 6-month period; the relationship of the objects in the diagram to the timescale is approximate. The columns of dashes below each bug report's lifeline represent the count of comments added to a bug report on a single day and are shown to provide a sense of the level of activity associated with each bug report throughout the bug report's life.

Bug report "B" (BR-B in the diagram) is the central report in this network. BR-B was opened with a "critical" severity level because it represented a bug that caused the software system to crash. As soon as it was opened, it was associated as "blocking" the resolution of BR-A. BR-A already existed, and was defined as a *meta* bug used to collocate a group of 15 (including BR-B) bug reports representing bugs that caused crashes in this part of the overall system. (Note that it would be possible to look at BR-A as the central BR in a different BRN: this is an example of how BRNs can be connected to each other as shown in Figure 1. See discussion of meta bug report networks below.)

The chain of duplicate relations between BR-B, BR-C, and BR-D is of interest. BR-C was opened several weeks after BR-B, during a six-week period when BR-B was not

very active (no comments added). BR-D was opened later the same day BR-C was opened. BR-D was quickly identified as representing the same bug as BR-C and marked "resolved/duplicate." BR-C was not identified as representing the same phenomena as BR-B until about six weeks after BR-C was opened.

The relationships between BR-B, BR-E, and BF-F are also of interest. The level of activity on BR-B was high during month 5. At one point, a patch for the bug represented by BR-B was introduced. This change caused the bug represented by BR-E (a type of bug and bug report identified as a "regression") to occur. BR-F was opened a couple of hours after BR-E was opened, and was immediately recognized as a duplicate of BR-E. The bad patch associated with BR-B was quickly backed out to resolve the problem associated with BR-E. BR-E was then marked "resolved/fixed."

# 4. Varieties, strategies, and impacts

The kinds of relationships found between bug reports, their frequency of occurrence, BRN strategies, implications of the construction and use of BRNs, and future work are discussed in this section.

## 4.1. Varieties

Almost two-thirds (65%) of the 385 bug reports in the sample have either a formal or informal relationship with at least one other bug report. Table 1 shows the frequency with which different types of relationships occur within the sample of 385 bug reports.

**Table 1. Frequency of relations in sample**

| | |
|---|---|
| **Duplicates** | |
| BRs with one or more duplicate BRs | 10% |
| BRs resolved as a duplicate of another BR | 33% |
| **Dependencies** | |
| BRs "blocking" one or more BR | 12% |
| BRs "dependent on" one or more BR | 7% |
| **Informal** | |
| BRs with "informal" relation to one or more BR | 33% |

Community members sometimes create bug reports that, instead of representing problems (bugs), anchor a collection of bug reports having common characteristics (e.g., all the high priority bug reports that should be fixed prior to the next software release). Community members refer to these anchor bug reports as "meta" or "tracking" bug reports. In the BRN illustrated in Figure 2, BR-A is a "meta" bug report used to create a network of bug reports representing system crashes of a similar type. Creation of a meta bug report and its associated BRN represents a specific social and information management adaptation made by community members to increase the utility of the bug report repository.

## 4.2. Strategies

Constructing BRNs is an information structuring strategy. Individual bug reports, first-class database objects, are composed over time into a new form of information, the BRN. Creating a BRN collocates a group of bug reports that would otherwise remain scattered and disassociated from each other. A BRN thus adds virtual structure to the bug report repository. Collocating information by adding or imposing structure is a complexity management / complexity reduction technique.

Asserting that a bug report is a duplicate of another bug report, for example, shrinks the set of bug reports that must be worked on. Shrinking the set of bug reports to work on reduces the complexity of the field of work. However, identification of duplicates is costly because members of the community must identify duplicates manually. There is also danger of mis-identification: bug reports that are not true duplicates (false positives) will be ignored because their status is "resolved." It's also clear, because of late-marked duplication and duplication time inversion, that "undiscovered" duplicate bug reports exist and multiple groups of people may be duplicating effort by working on two bug reports that represent the same issue (see, for example, the time period between the opening of BR-C and its resolution as a duplicate of BR-B in Figure 2.)

We also suspect that there are patterns of BRNs, for example patterns in the kinds of links that appear, and patterns in the types of links that are sanctioned and even crystallized into standard categories and supporting tools, such as dependencies and duplicates. There may also be patterns in how such networks are formed.

## 4.3. Impacts

As BRN construction orders information, it also orders social relations. BRs are specifications/codifications of social relationships, such as roles (reporter, assigned-to, cc: list member) and dynamic and patterned interactions (e.g. dialogues, question-response-elaboration sequences; negotiation; coordination of work, etc.). This means that as information is ordered through BRN creation/extension/modification, social relations are also being ordered. The impacts of this kind of social reordering might vary. In some cases, time-to-resolution may be improved by bringing more resources to bear upon a problem. In other cases, performance might deteriorate if, for example, the cost of coordinating the activities of more people slows progress toward resolution.

When a bug report is marked "resolved/duplicate" this means the bug *report* is resolved but it does not mean that the underlying bug itself has been resolved. "Resolved/duplicate" means that the resolver(s) believe

this is a duplicate report of a phenomenon that already has an effective representation elsewhere in the repository. It doesn't even mean that that the resolved bug report can now be ignored, since we have seen instances of late-identification of duplicates (e.g., BR-C in Figure 2) in which accumulated knowledge and dialogue may still be relevant to the resolution of the other bug reports in the BRN. Thus the semantics of the "resolved" keyword are clearly complex.

### 4.4. Future Work

Our work on understanding and identifying bug report networks has just begun. Many challenges remain, including:

- Identifying the situations in which BRNs are helpful (or unhelpful) in managing software problems; understanding the extent to which complex BRNs are taken into account by community members during problem resolution.
- Determining if BRNs are present in all bug report repositories; how the capabilities of different repositories and the conventions developed by the different communities influence the use of BRNs.
- Quantifying the range of complexity of BRNs in this and other bug report repositories; identifying the most useful metrics for measuring the size and complexity of BRNs (for example, a BRN can be thought of as a graph, with each bug report as a vertex in the graph).
- Developing useful representational forms (e.g., Figure 2) for BRNs that can contribute to our understanding and increase the utility of BRNs as a tool for SWPM.
- Determining how the inclusion of a BR in a BRN affects the community's SWPM performance (e.g., testing for a correlation between BRN membership and time to resolution).

Automatic extraction and representation of BRNs will be an important part of addressing the research questions raised here. The practical application of results of this research to software engineering practice also depends upon the development of effective and scalable automatic extraction and representation techniques. Challenges related to automatic extraction and representation include:

- Improving techniques for automatically extracting and representing BRNs from a bug report repository.
- Develop computational tools to discover and formalize the latent, undiscovered relationships between bug reports.

## 5. Conclusion

The analysis performed so far demonstrates that bug report networks are common in the bug report repository studied here: 65% of the bug reports sampled are part of a BRN. Members of this community commonly use the formal, symmetrical relationships of duplication and dependency as well as a wide variety of informal relationships. BRNs are a common and powerful means for structuring information and activity. BRNs, however, have not yet been the subject of concerted research by the software engineering community. The continuation of this stream of research will result in a more complete understanding of the contribution BRNs make to effective software problem management.

## 6. References


[1] Gasser, L., & Ripoche, G. (2003). Distributed collective practices and F/OSS problem management: perspectives and methods. *CITE'03,*Troyes, France, December 2003.

[2] NIST. (2002). *The economic impacts of inadequate infrastructure for software testing: final report.* May 2002. Planning report 02-3. Gaithersburg, MD: NIST.

[3] Leveson, N. & Turner, C.S. (1993). An investigation of the Therac-25 accidents. *IEEE Computer*, 26(7), 18-41.

[4] Osterweil, L. (1996). Strategic directions in software quality. *ACM Computing Surveys*, 28(4), 738-750.

[5] Fenton, N. E., & Neil, M. (1999). A critique of software defect prediction models. *IEEE Transactions on Software Engineering*, 25(5), 675-689.

[6] Conway, M.E. (1968). How do committees invent? *Datamation*, 14(4), 28-31.

[7] Parnas, D. L. (1972). On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12), 1053-1058.

[8] Herbsleb, J., Zubrow, D., Goldenson, D., Hayes, W., & Paulk, M. (1997). Software quality and the capability maturity model. *Communications of the ACM*, 40(6), 30-40.

[9] Crowston, K. (1997). A coordination theory approach to organizational process design. *Organization Science*, 8(2), 157-175.

[10] Powell, R.R. (1991). *Basic research methods for librarians*. (2nd ed.). Norwood, NJ: Ablex.

[11] Weber, R. P. (1990). *Basic content analysis*. (2nd ed.). Newbury Park, CA: Sage.

[12] Strauss, A., & Corbin, J. (1990). *Basics of qualitative research: grounded theory procedures and techniques*. Newbury Park, CA: Sage.