

# Software Problem Management as Information Management in a F/OSS Development Community

Robert J. Sandusky

Graduate School of Library and Information Science

University of Illinois at Urbana-Champaign

sandusky@uiuc.edu

**Abstract** – Bug reports created by a large, successful open source software development community show that software problem management (SWPM) is first of all information management and secondarily a problem solving activity. Solving software problems occurs only after a bug report, a first-class information object, has been created and “triaged” by community members. One 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; and (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 describe how specific classes of BRNs influence problem management within the community, and identify several new research questions.

## I. INTRODUCTION

We are conducting an empirical investigation 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. This work has focused on 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 corpora 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 275,000 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., screen shots, 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 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.

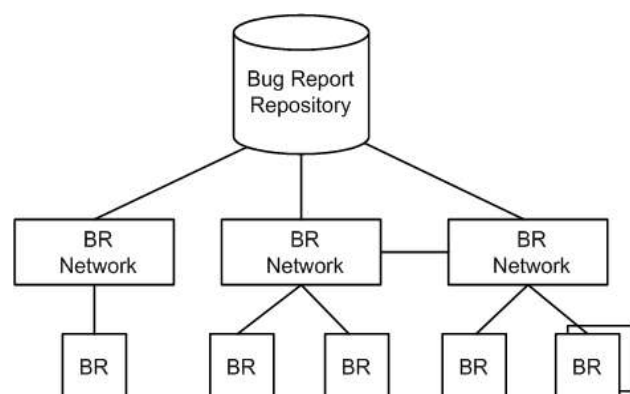


Figure 1. Bug Report Repository Elements

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?

## II. 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.

### A. 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.

### B. 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 BRs connected exclusively by informal relationships about 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.

## III. 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*, or *tracking*, bug report used to collocate a group of 15 bug reports (including BR-B) representing software problems 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.)

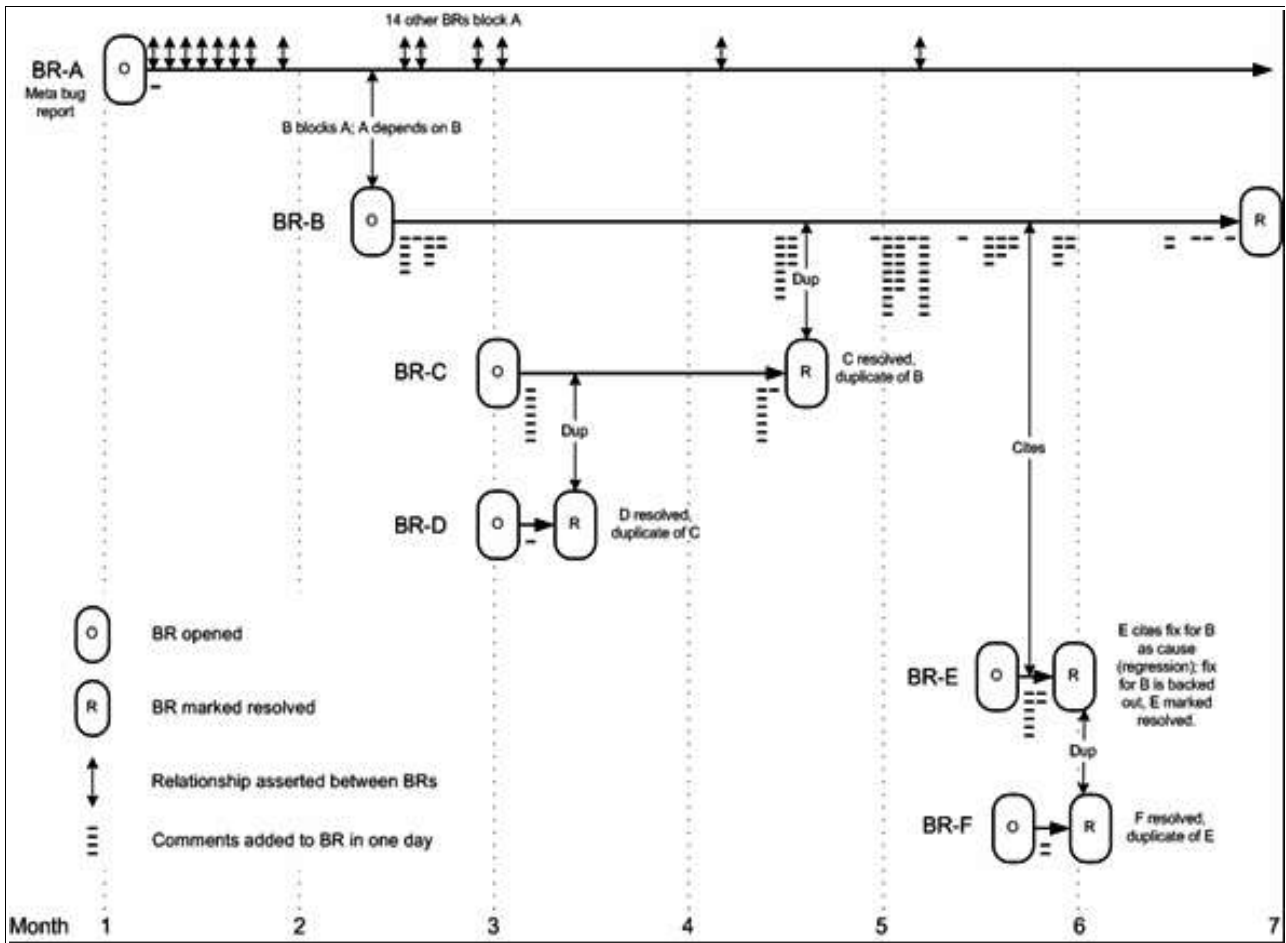


Figure 2. Bug Report Network

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/fixes.”

IV. FINDINGS

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.

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.

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%

Meta bug reports are an informal adaptation of the bug report repository made by the community in order to increase the utility of the bug report repository. The community defines a meta bug report as “A placeholder bug [report] for tracking the progress of other bug [report] s. Meta bug [report]s are made dependent on other bug

[report]s so that interested parties can be kept up-to-date with status via one bug [report], without having to receive all the mails related to all the bug [report]s related to the development of a particular area” [13].<sup>1</sup> By convention, users mark bug reports of this type by entering the string “meta” in the keyword field of the bug report.

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. Thus a BRN of duplicates collocates all of the bug reports that have been determined by the community to represent the same problematic phenomenon. BR-D in Figure 2 is marked as a duplicate of BR-C, creating a two-bug report network. About six weeks later, BR-C is finally recognized as a duplicate of BR-B, and thus BR-C and BR-D are both identified as duplicates of BR-B. BR-A, the meta bug report, collocates fifteen bug reports by establishing formal dependency relationships between the meta bug report and the fifteen separate bug reports that represent actual software problems.

Marking bug reports as duplicates of existing bug reports is a critical means for the community to control the number of bug reports requiring attention by software developers. On a typical day, more than 200 new bug reports may be created in this community’s bug report repository. Much of this identification work occurs during the triage phase early in a bug report’s life cycle. This was apparent from the results of a small investigation performed on the 105 bug reports created during the first 18 hours of one day and summarized in Table 2. Out of those 105 new bug reports, about 27% (28) had been removed from the workload in the first 16 hours of the day through application of five different strategies: (1) mark resolved as duplicate, (2) mark resolved as invalid, (3) mark resolved as won’t fix, or (4) mark resolved as works for me; or, (5) create a patch that fixes the reported problem. Only 3 of these bug reports had been “fixed” by changing the software; 24% of these 105 bug reports were effectively, if not completely or permanently, removed from the field of work almost immediately, without the effort of designing, coding, testing, and deploying a code change.

Shrinking the set of bug reports to work on reduces the size and complexity of the field of work. However, identification of duplicates is costly because members of the community must identify duplicates manually with a search interface as their only tool. 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 long time period between the opening of BR-C and its resolution as a duplicate of BR-B in Figure 2.)

**Table 2. Disposition of bug reports during triage**

BR Disposition	Count	Percentage of BRs	Cumulative Percentage
Marked “works for me”	1	1%	1%
Marked “won't fix”	1	1%	2%
Marked “fixed”	3	3%	5%
Marked “invalid”	6	6%	11%
Marked “duplicate”	17	16%	27%
Remaining to be worked on	77	73%	100%
<b>Total</b>	105		

BRN construction also re-orders social relations among participating community members. BRs are specifications / codifications of social relationships, such as roles (reporter, assigned-to, quality assurance contact, cc: list member) and dynamic and patterned interactions (e.g. dialogues, question-response-elaboration sequences; negotiation; coordination of work, etc.). Thus, 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. In regard to the BRN shown in Figure 2, there are five distinct reporters represented. In terms of assignees, five distinct individuals were assigned, and in three of the bug reports, two different actors were assignees at various points in time. There are three distinct QA contacts associated with these six bug reports. Some consistency in the identity of assignees and QA contacts is to be expected because these values correlate with the settings of the product and component fixed fields and because these six bug reports share significant characteristics (which is implied because of they are linked into a bug report network). In total, there are a total of 31 distinct individuals, based upon distinct e-mail addresses, associated as reporters, assignees, QA contacts, cc: list members, or commentators on these six bug reports.

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

<sup>1</sup>Within the community the term “bug” is often used to refer to both the phenomenon that is a software problem and the bug report, the information object that represents that phenomenon and the community's response. It is important to clearly distinguish the phenomenon from its representation in a bug report during analysis of the distributed work performed by this community.

BRN. Bug reports marked duplicate may contain descriptive information or have associated attachments that remain useful for resolving the underlying problem. Nor can we say that they are permanently removed because any bug report marked resolved may be re-opened. Thus the semantics of the “resolved” keyword are clearly more complex than they might first appear.

## V. CONCLUSION

The preceding discussion has reviewed some of the findings from our empirical investigation into how one F/OSS development community manages software problems, demonstrating the critical importance of information management in the software problem management (SWPM) process. This project views SWPM as a complex socio-technical enterprise, and takes a particularly close look at the information-related behavior of individuals and small groups in the context of individual bug reports and bug report networks (BRNs) in order to identify and understand the information practices used in this distributed work setting. The analysis of BRNs illustrates how activity (linking information objects to each other) results in the creation of new, complex forms of information (BRNs), affects the organization of work (e.g., through the assertion of dependency relationships between bug reports), and modifies the social relationships between community members (e.g., by making one person’s planned work dependent upon the work of another).

The creation of BRNs is one frequently occurring way the community responds to the information management challenges presented as they execute their SWPM process, as about two-thirds of all bug reports are included in a BRN. The creation and evolution of BRNs represents an information ordering strategy as bug reports are collocated as duplicates or dependents. Accurate identification of duplicate bug reports is a critical method by which the community reduces the number of bug reports requiring attention by software developers, a scarce and valuable resource. The establishment of dependency relationships configures future work in a particular way: problem resolution activities, often by different developers, are configured in a sequence to ensure that the pre-requisite problems are resolved first. Meta bug reports are a special case of the application of dependency relations: this practice is a strategy for reducing the cognitive effort of individuals who wish to monitor progress on a set of bug reports that, while not sequentially dependent upon each other, have some other factors in common that are of importance to the community. Dependency relationships, whether a meta bug report relationship or not, arrange bug reports in ways that provide a more accurate mapping of the bug reports to the requirements of solving problems and the community’s conventions regarding the of organization of work.

Activities related to the creation and evolution of BRNs also has a direct impact on the organization of work within the community by sequencing the implementation of problem resolutions into correct sequences and by bringing various community members into association through the establishment of dependency relationships between bug reports.

A number of practical and research issues remain to be addressed in regard to the way the bug report repository

supports the community’s information management and problem resolution requirements. In terms of the practical issues, the bug report repository provides limited support for the management of BRNs as information objects. First, duplicates and dependencies must be identified and marked manually with only the support of the repository’s search interface. Automatic processes for the identification of potential duplicate and dependent bug reports would make the community’s SWPM process more efficient. Second, the current repository implementation provides simple visualization support for only dependency relationships; duplicate and informal relationships are not represented at all in the visualizations. Improved visualization tools could provide easier-to-understand summaries of the often complex relationships between bug reports. Third, while bug reports are first class information objects and therefore have associated direct operations (like setting the bug report to a resolved status) and data elements, BRNs cannot be managed or tracked in analogous ways. It is not possible to mark a BRN as “resolved,” nor is it possible to search for BRNs in any way. Support for BRNs as first class objects with related behavior can widen the community’s repertoire of techniques for managing software problems.

Several important research questions still remain to be addressed. First, because of the limitations of the research method employed here, we would like to use other techniques, such as interviewing or participant/observer methods, to understand the situations in which the information represented by BRNs is helpful or unhelpful in managing software problems. We would also like to understand the extent to which complex BRNs are taken into account by community members during problem resolution. Second, comparative studies should also be performed to determine whether the information management practices identified here are employed by other F/OSS communities or in traditional, closed software development communities. Finally, we would like to determine how the inclusion of a BR in a BRN affects the community’s SWPM performance (e.g., testing for correlation between the membership of a bug report in a BRN and time to resolution for individual bug reports).

The analysis performed so far demonstrates that BRNs 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.

## VI. ACKNOWLEDGMENTS

I thank Les Gasser and Gabriel Ripoché for their ideas, comments, and other contributions to the work presented here. This material is based upon work supported by the National Science Foundation ITR (Digital Society and Technologies) Program under Grant No. 0205346. Any opinions, findings, and conclusions or recommendations

expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

## VII. 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.
- [13] Mozilla.org Bugzilla Keyword Descriptions List. Available at: <http://www.mozilla.org/describekeywords.cgi>; accessed [2005, January 13].