Computer Supported Cooperative Work (2006) 15:149–183
DOI 10.1007/s10606-006-9017-0

Experiences in Automating the Analysis of Linguistic Interactions for the Study of Distributed Collectives

GABRIEL RIPOCHE[1,2] & JEAN-PAUL SANSONNET[1]
[1]*LIMSI-CNRS, BP 133, 91403, Orsay Cedex, France (E-mail: gripoche@limsi.fr);* [2]*Graduate School of Library and Information Science, University of Illinois at Urbana–Champaign, 501 E. Daniel Street, Champaign, IL, 61820, USA*

Abstract. An important issue faced by research on distributed collective practices is the amount and nature of the data available for study. While persistent mediated interaction offers unprecedented opportunities for research, the wealth and richness of available data pose issues on their own, calling for new methods of investigation. In such a context, automated tools can offer coverage, both within and across collectives. In this paper, we investigate the potential contributions of semantic analyses of linguistic interactions for the study of collective processes and practices. In other words, we are interested in discovering how linguistic interaction is related to collective action, as well as in exploring how computational tools can make use of these relationships for the study of distributed collectives.

Key words: computational tools, distributed software problem management, interaction modeling, large-scale automated analyses, speech acts

1. Introduction

An important issue faced by research on Distributed Collective Practices (DCP) is the amount and nature of the data available for study. Because participants are not physically collocated, communication within distributed collectives is usually limited to various forms of electronic media. In many cases, these interactions leave a persistent trace in the form of archives, web sites, and other forms of repositories, which can be collected for research purposes. These data provide a great opportunity to empirically study the structures, organizations and practices of distributed collectives. First, because electronic media represent virtually the only channels of interaction within a collective, coordination and other social processes necessarily occur over these channels. Second, because archives are recorded and available, virtually all of the interactions taking place within a collective are accessible for study, providing a rich view of a collective's activity.

While persistent mediated interaction offers unprecedented opportunities for research, the wealth and richness of available data pose issues on their

own, calling for new methods of investigation. Methods relying on human analyses simply do not scale to the size and number of emerging repositories, and automated tools that can handle large amounts of data are increasingly needed to assist researchers in their analyses. The main advantage of automated tools is that they can offer coverage, both within and across collectives. Automation can make it possible to exploit the entirety of the available data and provide an exhaustive view in a given analysis. Analyses can also be reproduced more quickly on different cases, allowing for comparative studies on a larger scale.

Furthermore, a large part of the interactions taking place within distributed collectives occurs through natural language. Language enables the subtle, situated articulation work necessary to handle coordination and keep a collective running (Gerson and Star, 1986; Suchman, 1996; Schmidt and Simone, 2000). As such, it represents an important source of information about collective practices and activity. However, language is also a form of communication that is notoriously difficult to handle in a systematic fashion. To be of use, automated tools should therefore be able to process and deal with the semantics of interactions in natural language.

We believe such automated tools can provide a helpful contribution at three levels. First, from a research perspective, automated analyses can contribute to the building of a theory of collective activity and distributed practices by guiding and completing finer-grained, more qualitative approaches. Also, from a systems designer's perspective, making collective processes more explicit and obtaining models of how they are articulated can allow for the design of computer-supported infrastructures that could both better address the general issues encountered by distributed collectives (by embedding a theory of collective practices into design), as well as the specific needs of different types of collectives (by adapting support to particular practices in specific collectives). Finally, in the same way that researchers have to handle large amounts of data when analyzing distributed collectives, the collectives themselves face a problem of information overload, where members cannot keep up with the rate at which new information is created. Automated tools could also be made available to collectives as an additional way to cope with data by offering alternative means to effectively access information gathered throughout the life of the collective.

Obviously, such an approach is only possible if automated tools are able to process linguistic information to acceptable levels of precision, and if we are able to identify useful phenomena out of the generated models. In this paper, we investigate some of the potential contributions of semantic analyses of linguistic interaction for the study of collective processes and practices. In other words, we are interested in discovering how linguistic interaction is related to collective action, as well as in exploring how computational tools can make use of these relationships.

The rest of the paper will be structured in the following way: Section 2 presents our object of study and its interest for research on distributed collective practices. Section 3 introduces our current model of interaction. Section 4 reports on analyses we have carried out to illustrate our method. Section 5 details our method for automating interaction modeling and provides an evaluation of the current status of our tools. Section 6 discusses what we have learned from our research, listing issues we have encountered so far and how we envision solving them.

2. Object of study: FOSS and Bugzilla

2.1. FOSS as a prototype of DCP

Free/Open-Source Software (FOSS) has gathered significant interest over the last few years, both for of its interesting new paradigm (Raymond, 2001) and for its potential for study (Gasser et al., 2004; Howison et al., 2005). On many aspects, FOSS communities constitute prototypical distributed collectives.

First, FOSS participants are scattered around the world and communicate virtually entirely through computer-mediated channels such as email, mailing lists, forums, wikis, code repositories, and other electronic communication media. These interactions are carefully archived and preserved, and for the largest part made freely available to anyone on the Internet, providing enormous amounts of empirical data on structures and practices of FOSS communities (Yamauchi et al., 2000).

Also, FOSS communities are focused around complex tasks, which both motivate and sustain the communities, and require elaborate interactions between participants. Building software is far from being trivial and requires complex coordination among project members (Crowston, 1997; De Souza et al., 2003), especially in distributed contexts (Herbsleb and Grinter, 1999). Despite this complexity, FOSS does not seem to rely on many formal coordination mechanisms, such as work assignment or planning and scheduling (Scacchi, 2002), and organization emerges from the collective's evolution as much as it is planned.

Finally, there is not one form of FOSS community, but many different ones, with different structures and dynamics (Crowston and Howison, 2005; Gacek and Arief, 2004). This provides an opportunity for carrying out comparative studies across communities and for observing variations in practices within the common context of FOSS development.

Taken together, these characteristics make FOSS a good candidate for studies of distributed collective practices, providing both an appropriate setting and large amounts of data to study. The rest of this paper will focus on a specific collective we have been studying in our research: Bugzilla.

2.2. BUGZILLA

Our initial study focuses on the Mozilla community, and more specifically on its collective dedicated to reporting and resolving bugs found in software developed by the community. The Mozilla project, which aims at developing a robust, standard-compliant Internet software suite, has seen a constant increase in popularity since its creation in 1998. Their flagship product, Firefox, recently reached 100 million downloads, and Mozilla-based browsers are approaching the 10% market share (WebSideStory, 2005).

The project counts hundreds of developers distributed across the world, and tens of thousands of participants (users, developers, testers, etc.) who have contributed to one of the 300,000 bug reports currently in Bugzilla, the community's problem-management repository (Mozilla Organization, 2005).

Software problem management plays a central role in every software development effort (Cartensen et al., 1995) and FOSS is no exception (Sandusky, 2005). The bug reporting system (including both the tools and protocols developed by the community) often constitutes the main coordination mechanism within the software problem management process and as such represents a "microcosm of coordination problems" (Crowston, 1997, p. 173; Schmidt and Simone, 1996). Bug reports are key elements in a software maintenance process, keeping track of and making visible information about bugs and activities that surround them. However, a bug report does more than hold facts or information; it also has interpretive, management, and expressive functions, and provides an artifact and a corresponding protocol that supports the software maintenance process.

Bug reports (Figure 1) in Bugzilla contain (Bugzilla Team, 2005a):

(a) Descriptive information giving a synthetic view of the problem, including: Short description, classification of the issue (component, module),

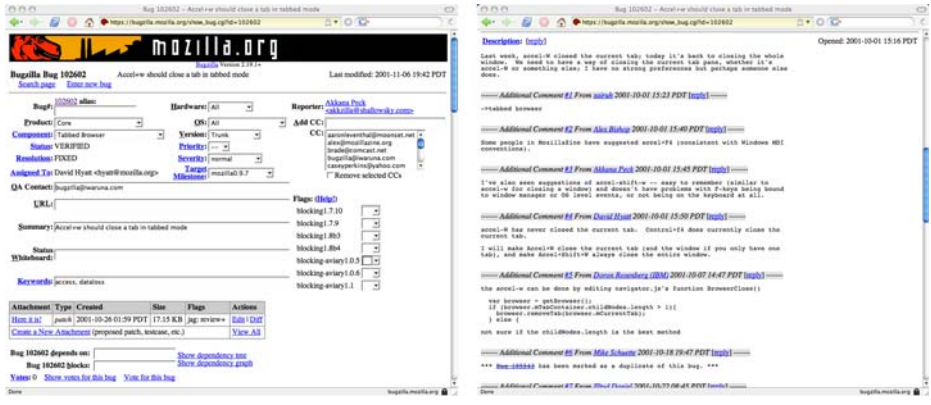

Figure 1. Screenshots of a bug report, showing the formally encoded information (left) and the free-form comments (right).

current bug report status (status, resolution, priority), relations to other bugs in the repository, etc. In a large proportion, these different elements take the form of coded entries selected from controlled vocabularies, which can evolve as the practices and needs of the maintenance process evolve. Some entries are entered as free text, but usually with constraints on the length and (collectively defined) acceptable form of input.

(b) A record of the identity of the people involved in the different aspects of the bug resolution process. Some roles are explicitly identified as part of the description of a bug report (reporter, developer in charge of the bug, QA contact), while others are less clearly tracked.

(c) A record (called "activity log" in Bugzilla) of all modifications made to the bug report, identifying who made the change, when it happened, and what the change was about.

(d) Attached documents such as patches, screen shots, stack traces, documents used as references, test cases.

(e) Comments in free text made by participants (i.e. anyone with a Bugzilla account), providing a weakly structured way for participants to contribute additional information. Typical uses of these comments include descriptions of the problem, instructions to reproduce, opinions, hypotheses, conjectures, requests, reports of actions taken, etc. Bug reports contain an average of ten comments ordered chronologically. Comments vary greatly in length, ranging from one-word instructions to multi-paragraph contributions, with an average of a few sentences.

At any given time, a very large number of reports (tens of thousands) are open, with thousands of them being simultaneously handled. This requires high amounts of coordination between the members of the collective to manage resources, avoid conflicting fixes, and reduce the amount of duplicate effort (Gasser and Ripoche, 2003).

The Bugzilla environment is in this way representative of large-scale, task-oriented distributed collectives, and provides important data about collectively enacted processes related to problem solving activities. From the perspective of developing analysis tools, we are interested in the two following questions:

– What does interaction look like in bug reports? (How can we characterize interaction?)
– Does interaction occurring in the comments map in some way to the bug resolution process(es)? (How can we learn about practices by studying interaction?)

It is these questions we attempt to address in our analyses, using the model we detail in the following section.

3. Model

3.1. TOWARD A MODEL OF INTERACTION

Schmidt and Simone (2000) discuss the inherent interlacing of formally defined work flows and *ad hoc* alignment in every collaborative work situation: "In cooperative work, *ad hoc* alignment and improvisation on the basis of mutual awareness is inexorably interlaced with the execution of predefined procedures and similar formal constructs, and vice versa" (p. 2). Bug reports explicitly support this duality, providing both formal constructs in the form of formally encoded information, and a shared space supporting *ad hoc* alignment through free form comments in natural language. Our model aims at exploiting both formally encoded (items *a–c* in Section 2.2) and linguistically expressed (item *e*) information to obtain representations that can help us make sense of collective practices occurring in Bugzilla.

Formally encoded information materializes the predefined process of bug resolution and enables the collective to keep track of the progress they make in this process. But this information captures more than just the evolution of a process. We think of it as information provided by the collective about various aspects of the bug resolution process, about the bug itself, and also about the structure and organization of the collective. For instance, marking a bug "assigned" and filling the corresponding "assigned to" field tells us something about the state of a given bug, in that there now is someone in charge of the problem. In addition, it also tells us something about the role of a specific individual in a particular resolution process, and to a certain extent, about her role within the collective in general. It thus provides information about how the collective is organizing around a set of bugs. Therefore, we see formally encoded information as clues given by the collective about its state, progress, priorities, etc.

While formally encoded information gives us precious information about the structure of the collective and the different stages a bug report goes through, it does not provide a complete picture of how these structures and processes are articulated, because coordination also occurs through informal, situated alignments that cannot be captured in a formal, static structure. Bugzilla comments therefore play a critical role in letting participants articulate around the processes embedded in, and enabled by, the formally encoded information. As such, the comments constitute an important source of information about collective practices.

Therefore, both "sides" of the coordination process complement each other and should not be interpreted without the support of the other (Gerson and Star, 1986). These two sides will always be present in every context of interaction, albeit not in the same proportion or the same form. For instance,

forums will provide much less formal structure, but message headers do contribute to shape, constrain, and guide the interaction. At the other extreme (staying in software development), environments such as version control systems provide a very codified structure and leave little room for informal articulation; but they still rely on free form short comments associated to code submissions to help developers make sense of particular actions and how they fit in the process.

Our model aims at exploiting this duality by using both formally encoded information and information extracted from linguistic interactions to identify patterns associated with specific collective practices, processes and structures (Gasser and Ripoche, 2003; Ripoche and Gasser, 2003). We illustrate our general model in Figure 2.

In order to use both sides of the interaction, the first step consists in extracting useable information from collected raw data. Because data is not generated for research purposes but as a byproduct of the collective's functioning, some formatting is often required, and extracting information from FOSS repositories can be quite complex depending on the source of the data (Howison et al., 2005). However, in the case of Bugzilla, we had direct access to the repository database, which made extraction relatively straightforward. In this paper, we consider data to be properly extracted and focus on the following steps.

Modeling semantic information from linguistic interactions will be treated in the rest of this section and in Section 5, and examples of analyses relying on both formal and linguistic data will be provided in Section 4.

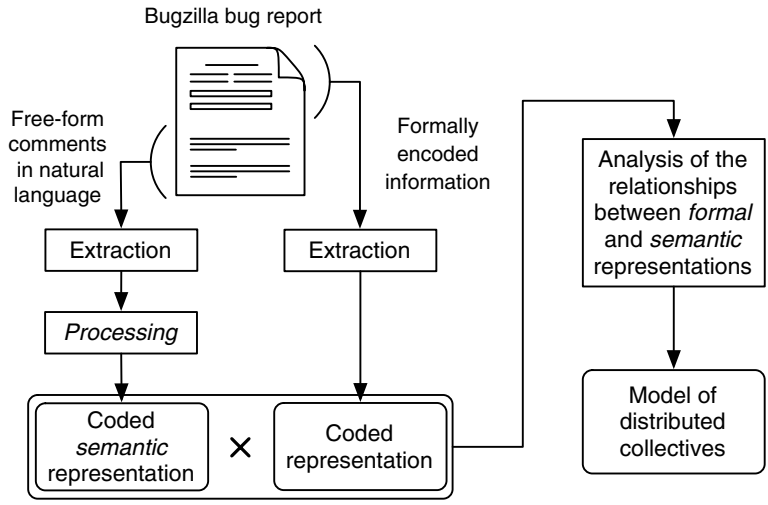

Figure 2. Our general approach.

3.2. FOUNDATIONS

Put simply, our objective is to capture activity by processing linguistic interactions. We adopt Winograd, Flores and colleagues' view of "language as the primary dimension of human cooperative activity" (Winograd, 1987, p. 4; Flores et al., 1988). The language/action perspective relies on speech act theory (Searle, 1975) for the modeling of interaction, on the basis that it provides an interesting foundation for understanding language as a reflection of what people *do*. Speech act theory emerged from the observation that not all utterances are statements aimed at asserting the truth or falsity of a fact, but that language can also be used to make promises, to request something, to give one's opinion, etc. – in short, to perform actions. The language/action perspective observes that speech acts are not isolated events but participate in larger patterns of conversation that structure human organization.

The language/action perspective and theory of speech acts have been at the center of a heated debate in the field of CSCW (Suchman, 1994; Winograd, 1994; Bannon, 1995), from which two threads of critiques need to be distinguished and addressed in the context of our study.

The first thread concerns the suitability (or lack thereof) of a theory based on speech acts as an account of human conduct. Critiques point out the arbitrariness of "universal", *a priori* taxonomies and their inability to deal with the situatedness of human interaction. While we acknowledge these limitations, we believe speech acts still constitute a useful device for the analysis of interaction from a computational perspective, providing a semantic, "action oriented" representation that lends itself well to modeling.

The second thread of critiques discusses the impact of explicit, externally defined categories on organizational life and control structures, focusing on *the Coordinator*, a tool based on the language/action perspective. These critiques have to be framed in a perspective of design, as it is the embedding of such categories into support tools that becomes problematic. However, not adopting a design perspective (our current objective is to model and analyze processes of interaction, not to design them), we do not find issues of control and discipline to be relevant to our problematic: Speech acts – or any other features we may extract from interaction – are not used as a communicative device within the observed collective but as a *post hoc* analytic device.

To summarize, we view speech acts as a useful device to abstract interaction and extract some of its semantics in order to put in evidence patterns of interaction. While necessarily incomplete (like any representation), speech acts give one perspective on some of the things people are doing through their interactions, as well as on how coordination and collective activity occurs through linguistic interaction.

More recently, many variants of speech act theory have been proposed. In the literature, speech acts have been alternatively referred to under different terms, such as ''communicative acts'' or ''dialogue acts'', both for practical reasons (e.g. to avoid being understood as a speech-only phenomenon), and to address issues with Searle's original definition. To clarify our use of the concept, we adopt Bunt's definition of dialogue acts – along with the terminology – as the starting point for our model:

> A dialogue act is a unit in the semantic description of communicative behaviour produced by a sender and directed at an addressee, specifying how the behaviour is intended to influence the context through understanding of the behaviour. (Bunt, 2005, p. 2)

However, because we are not concerned with dialogue *per se* but instead with interactions within loosely defined groups, we believe it is important to specify that the addressee is more of a role than a precise, physical interlocutor within a dialogue. In Bugzilla, it is perfectly possible to address ''the collective'' in its entirety, or rather, any potential participant who might feel concerned by the problem at hand.

Dialogue acts therefore provide a model of interaction as the articulation of joint action, through the representation of ''the way in which the utterance is meant to change the information state of an interpreting system upon understanding the utterance'' (Bunt, 2005, p. 2).

3.3. DIALOGUE ACT TAXONOMY

Our taxonomy is inspired by Searle's initial taxonomy (1975) and by recent work proposed in the field of dialogue modeling (e.g., Allen and Core, 1997; Jurafsky et al., 1998). The latter taxonomies provide finer grained categories and rely on empirical analyses of relatively large corpora. The selection of our taxonomy is guided by several criteria that attempt to balance issues of genericity with specific needs in the types of analyses we want to conduct.

While we believe it is necessary to take into consideration previous work in the domain of interaction modeling, it is also important to recognize the divergences between existing studies in conventional dialogue and the type of interaction we are attempting to model. Previous work focused on short, synchronous, oral, two-person dialogues with a specific and relatively simple task, such as itinerary planning (Core and Allen, 1997) or casual telephone conversation (Jurafsky et al., 1998). In the case of Bugzilla, interaction is asynchronous, written, involves multiple people, and handles much more complex and sometimes ill-defined problems. As such, some of the categories adapted to conventional dialogues may have a different meaning, or be irrelevant in the context of Bugzilla interactions. In addition, the Bugzilla

interaction process shows some specificities that are important to capture in a taxonomy, because they may help characterize important aspects of the processes and practices of the Bugzilla collective.

We are making incremental progress in our analyses of interaction, and are still at the stage of identifying useful constructs for out study. In this context, using an overly precise taxonomy (Switchboard DAMSL contains 42 tags (Jurafsky et al., 1998)) may not be the best way to approach our problem. Instead, adopting a relatively simple taxonomy – which at first contains coarse categories but can be further detailed as needed – seems like a more appropriate exploratory approach. This led us to keep the original high-level taxonomy proposed by Searle as our foundation, which we progressively detailed as we were identifying interesting phenomena.

This method presents the main advantage that we do not have to go through an annotation process that is more complex than it needs to be. Annotation reliability tends to decrease as the taxonomy becomes more refined, and by reducing the complexity of the taxonomy to the minimum needed, we avoid reliability issues to the best of our ability. The taxonomy is likely to change as our needs evolve, and it has in fact already done so a few times as we were making progress in our understanding of Bugzilla interactions.

Our taxonomy (Table 1, detailed in Appendix A) thus attempts to strike a balance between the existing models proposed by research in interaction modeling, the characteristics of the type of interactions studied, and the specific needs of our investigations.

At our stage of analysis, this taxonomy presents a good tradeoff between coverage and complexity, and has allowed us to design experiments to empirically investigate interactions in Bugzilla and their relationships to different aspects of the Bugzilla resolution process.

Table 1. Bugzilla dialogue act taxonomy (detailed in Appendix A)

Assertives	**Commissives**	**Expressives**
Description (`dsc`)	Commitment (`com`)	Acknowledgement (`ack`)
Prescription (`pre`)	Offer (`off`)	Agreement (`agr`)
Statement (`sta`)	**Directives**	Apology (`apo`)
Performatives	Direction (`dir`)	Disagreement (`dis`)
Action (`act`)	Instruction (`ins`)	Exclamation (`xcl`)
Others	Question (`que`)	Expression (`xpr`)
Introduction (`int`)	Request (`req`)	Greeting (`grt`)
Label (`lbl`)	Requirement (`rqr`)	Opinion (`opi`)
Unknown (`ukn`)	Suggestion (`sug`)	Smiley (`smi`)
	Summon (`sum`)	Thank (`thx`)

4. Experiments

In this section, we report some early results of an ongoing analysis of patterns of interaction in Bugzilla, using the model we just described. While it would be too early to draw conclusions at the level of the corpus based on these results, we hope that our initial observations can suggest trends to be expected from studying interaction in distributed collectives and orient further research in this area.

4.1. METHOD

Our analyses are based on a systematic sample of 252 bug reports drawn from a March 2002 snapshot of the Bugzilla repository containing over 128,000 reports. Each bug report was extracted from the Bugzilla database and manually annotated with dialogue act information (further details on the annotation process will be given in Section 5), leading to a total of 7626 dialogue acts. For each bug report, the following information is available for analysis:

- *General descriptive data*: Number of participants, number of comments, date the bug was opened and date it was closed.
- *Trajectory data*: Number of status changes, final resolution, and status trajectory.[1]
- *Interaction data*: Dialogue acts indexed by sentence, comment, and report number, and providing a semantic representation of the content of the comments of a bug report.

4.2. OVERVIEW OF INTERACTION IN BUGZILLA

Before any attempt at analyzing interactions and their relationships to collective processes, it is useful take a look at the general characteristics of Bugzilla interactions, in order to get a broad understanding of the object we are studying: Are interactions in Bugzilla similar to other forms of dialogue such as the ones studied in dialogue modeling, or are there significant differences? Furthermore, in case such differences do exist, what can they tell us about the type of interactions taking place in the collective?

Figure 3 provides an overview of the distribution of dialogue acts in the practice of managing bugs, based on the complete sample of 252 reports. Interaction is largely dominated by assertives (49.1%), followed by directives (15.6%), performatives (14.0%), expressives (9.3%), and commissives (1.9%). Other types constitute 10.1% of the sample, with only 2.1% of unknown utterances. There is a wide gap between a few dominant categories and the remaining ones: Nearly two-thirds (61.8%) of the dialogue acts fall into one of the three main categories (`statement`, 25.6%; `description`,

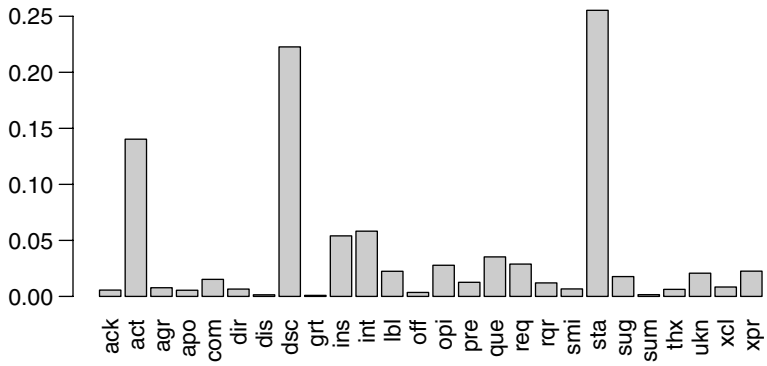

Figure 3. Distribution of dialogue acts in the sample ($N_r = 252$; $N_{da} = 7626$).

22.3%; `action`, 14.0%), with none of the other categories representing more than 6% of the total.

A distinction can be made between dialogue acts directly contributing to a dialogue and those that have more of a "stand alone" status. By this distinction, we mean that some dialogue acts can reliably indicate a dialogue between participants. For example, dialogue acts such as `question`, `request` or `suggestion` expect an answer from a participant, and as such initiate a more direct interaction. In the same way, dialogue acts such as `acknowledgement`, `agreement` or `disagreement` respond directly to a previous intervention. In opposition, `description` or `statement` do not necessarily initiate or react to a direct interaction, and multiple people can just "stack up" information comment after comment without specifically engaging in a dialogue. Looking at the proportion of direct dialogue acts can help us characterize the structure of interaction in Bugzilla.

We find that only 14.4% of the dialogue acts in the sample are direct dialogue acts,[2] which can be compared to the proportion of 38.1% of such acts in dialogues annotated in the Switchboard project (Jurafsky et al., 1998).[3] The comparison has to be carefully interpreted since, as we mentioned earlier, there are differences between annotation schemes and between the two types of interaction. Also, our estimation necessarily underestimates the true proportion of direct dialogue acts, since acts such as `statement` or `action` can occur as an answer in the context of a dialogue. However, the large difference between the two types of interaction does give us a first approximation of the sort of interactions going on in Bugzilla. This tendency is consistent with informal observations we made while annotating the reports, in that it suggests that interactions in Bugzilla are much less direct than in conventional dialogue. Instead information is accumulated more in the style of a "black board process" (Hayes-Roth and Hayes-Roth, 1979), in which each participant contributes without neces-

sarily articulating directly with the previous posters. This of course does not mean that there is no articulation at the level of the process, but that it happens mostly without being made explicit through traditional dialogue conventions.

4.3. INTERACTION AND RESOLUTION PROCESS

According to our model, we are attempting to identify relationships between formally encoded data (which we consider as information provided by the collective about its own state, structure and activity) and patterns of linguistic interaction, in order to better describe the processes and practices enacted by the collective.

One the most prominent indicator of the bug resolution process is the status of the bug. Bugzilla, like most development efforts (Cartensen et al., 1995), has a normative resolution process, which is explicitly described in the Bugzilla documentation and shown in Figure 4 (Bugzilla Team, 2005b). A bug report keeps track of the progress made in the resolution process of a bug through the status field (which indicates the state the process is currently in), and through the activity log (which tracks changes in status throughout the life of the report). Each status corresponds to a different stage in the resolution process, as identified by the collective. Typically, a report will be created as UNCONFIRMED or NEW, depending on the privileges of the reporter of the bug. After the bug report has been triaged, it is ASSIGNED to a developer, who becomes responsible for its resolution. Once the bug is solved, the bug report is marked as being RESOLVED, and then as VERIFIED once the QA team confirms that the bug is indeed fixed. Finally, the report is CLOSED when no more action is required. Eventually, a bug can be REOPENED if problems reappear after the bug was identified as being resolved.

The sequence of status changes of a given bug, together with the collective processes and articulation enacted at each step of the process can be thought as the trajectory of the resolution process for that bug (Strauss, 1988), and provides a useful structure to analyze and compare reports. For instance, looking at trajectories, one can ask what a ''normal'' trajectory is and what sorts of deviations can occur. In other words, do all bug trajectories correspond to the process explicitly defined by the Bugzilla community, or are there exceptions (and how common are the exceptions)? Within our model, we can ask the following questions to better characterize the bug resolution process and eventual variations of it:

– Do linguistic interactions differ at various stages of the resolution process?
– If so, what can the differences tell us about the process(es) of fixing bugs?

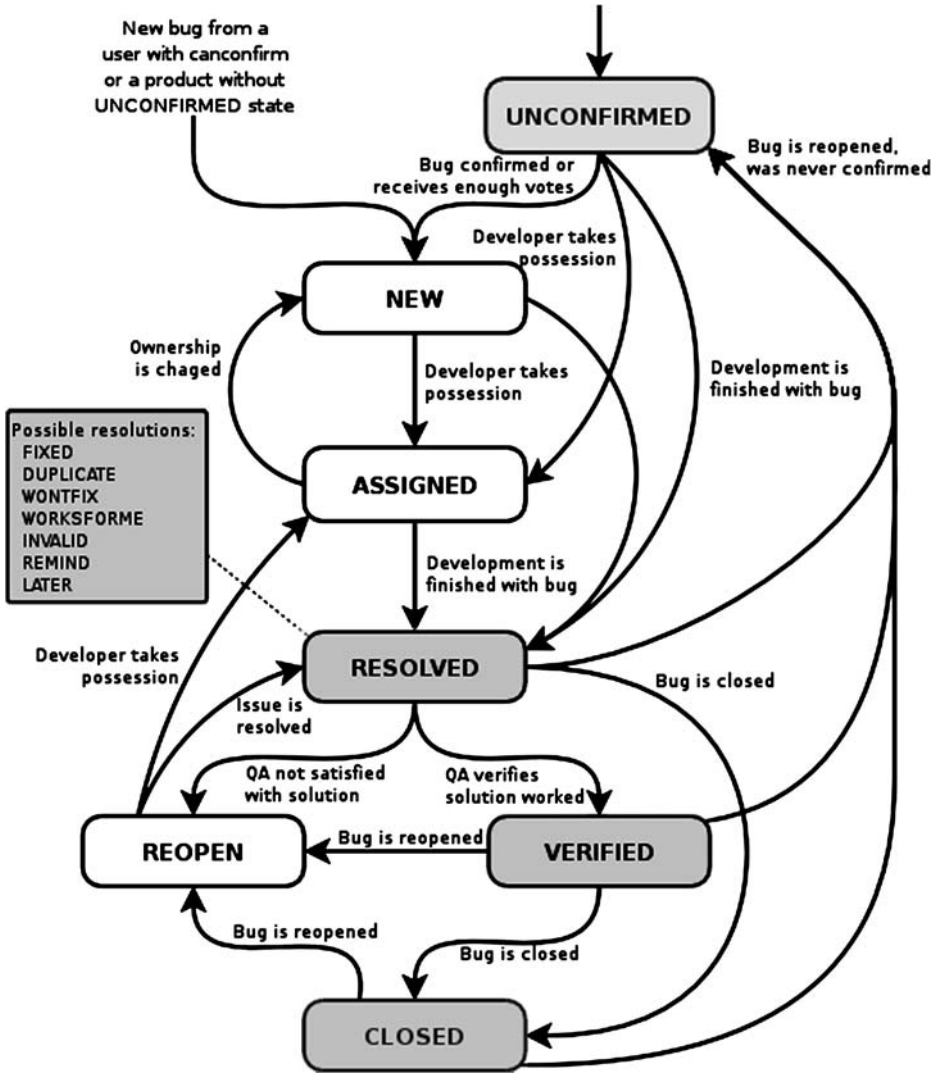

Figure 4. The Bugzilla bug resolution process (Bugzilla Team, 2005b).

To investigate these questions, each comment was associated with the status of its bug report at the time the comment was posted, by matching the timestamps available both in the activity log and in the comment list. All the comments were then grouped by status in order to analyze eventual variations in the distribution of dialogue acts across different stages of the life cycle of a bug.

Figure 5 displays the distribution of each type of dialogue act for each of the six statuses a bug report can go through during its life.[4] The graphs show large variations in dialogue act profiles at different stages of the resolution process. Most of them are shown to be significant through an analysis of

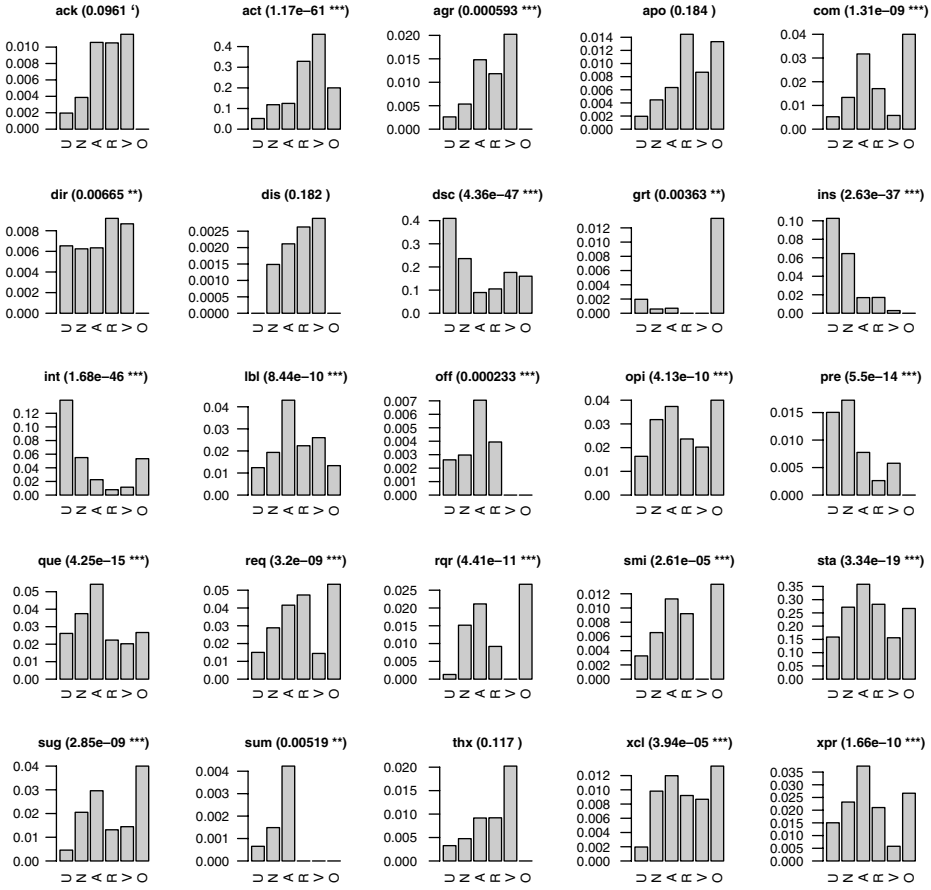

Figure 5. Dialogue act proportion by status (Kruskal–Wallis: $N_r = 252$; $N_{da} = 7494$; $df = 5$; p: *** < 0.001, ** < 0.01, * < 0.05, ‘ < 0.1).

variance (Kruskal–Wallis[5]) performed for each category of dialogue acts and reported in Table 2 (significance levels are also given in Figure 5 in the title for each dialogue act category).

Looking at the variations of the distribution of dialogue acts, four stages can be identified, with four distinct profiles of interaction:

– *Description*. This stage is characterized by high levels of `description`, `instruction`, `prescription` and `direction`. In this phase – corresponding roughly to the UNCONFIRMED and NEW statuses – a new bug has been reported and the collective is trying to get a precise picture of the problem by providing elements such as descriptions of the problem, descriptions of the environment in which the problem occurs, instructions on how to reproduce the problem, and prescriptive descriptions of what the ‘‘normal’’ behavior should be.

Table 2. Analyses of variance for each dialogue act category (Kruskal–Wallis: $N_r = 255$; $N_{da} = 7494$; $df = 5$; p: *** < 0.001, ** < 0.01, * < 0.05, ' < 0.1)

D. act	N reports	N d. acts	H	p
ack	32	43	9.3449	0.09607 '
act	234	1078	295.0309	$< 2.2e^{-16}$ ***
agr	41	59	21.7145	0.0005932 ***
apo	35	42	7.529	0.1842
com	67	116	50.1176	$1.311e^{-09}$ ***
dir	41	50	16.0681	0.006652 ***
dis	9	11	7.5669	0.1818
dsc	228	1701	227.1521	$< 2.2e^{-16}$ ***
grt	6	7	17.5062	0.003633 **
ins	115	412	181.4445	$< 2.2e^{-16}$ ***
int	138	444	224.4199	$< 2.2e^{-16}$ ***
lbl	86	172	51.0526	$8.437e^{-10}$ ***
off	20	27	23.8378	0.0002333 ***
opi	83	213	52.565	$4.131e^{-10}$ ***
pre	63	96	71.2994	$5.497e^{-14}$ ***
que	107	269	76.6306	$4.248e^{-15}$ ***
req	98	224	48.2193	$3.204e^{-09}$ ***
rqr	46	92	57.2863	$4.414e^{-11}$ ***
smi	35	51	28.7358	$2.613e^{-05}$ ***
sta	227	1953	96.2009	$< 2.2e^{-16}$ ***
sug	47	136	48.4663	$2.853e^{-09}$ ***
sum	9	12	16.6628	0.005186 **
thx	35	48	8.8131	0.1168
xcl	41	64	27.8254	$3.937e^{-05}$ ***
xpr	75	174	54.4913	$1.661e^{-10}$ ***

The results show that for most categories, bug status has an effect on dialogue act profiles.

– *Investigation and organization*. Once the collective has a "good idea" of what the problem is, discussion starts around what could and should be done, who should do it, etc. Commissives (`offer`, `commit`), directives (`question`, `request`, `summon`, `requirement`), "opinionated" expressives (`expression` and `opinion`), along with non-descriptive assertives (`statement`) increase significantly. People are also more often referred to by name (`label`) in this phase, showing a need for more direct organization of the interaction. This phase corresponds to the NEW and ASSIGNED statuses (ASSIGNED is often skipped, in which case bug reports keep a NEW status until the resolution phase).
– *Resolution*. At this stage, the problem is identified, a solution has been discussed, people have been assigned to handle the bug; it is time to take action and fix the problem. Performatives (`action`) clearly dominate this phase, representing 32.9% of acts in RESOLVED and 46.0% in VERIFIED. Each of the two statuses has a slightly different profile however: the earlier

period (RESOLVED) has higher levels of commissives and directives, likely to correspond to "late" organization. The VERIFIED stage shows much less variation, with `action`, `statement` and `description` representing 79.2% of the dialogue acts present in this stage (vs. 71.6% for RESOLVED). Contrary to the early phases in which descriptions show problematic behaviors, descriptions in these stages confirm the "fixed" behavior.[6]

– *Breakdown*. In this stage, a bug report is marked REOPENED, indicating that a problem that was thought to be solved turns out not to be. At this point, the collective has to reopen the investigation, evaluate what is wrong, look for more information, reassess the problem, and reorganize to solve it. This stage can be thought as a combination of the other stages in the "normal" life of a bug report (investigation, organization, and attempt to take corrective action). The profile of interaction shows a surge in directives (`request`, `requirement`, `suggestion`), along with an increase in commitments and slightly above average levels of performatives. Expressives with social function also increase (`expression`, `greeting`, `smiley`; `apology` increases as well but is not significant), indicating that there is a need to handle the breakdown at a social level as well as at a technical one.

This analysis demonstrates that profiles of interaction vary depending on the stage of resolution of a bug report. This gives us an alternative characterization of the resolution process that is not based on arbitrary steps in a process, but rather on different forms of interaction among the participants during the resolution process.

Using this relationship, we can envision detecting mismatches between the status of a report and its profile of interaction, which may point to misalignments or breakdowns in the resolution process. Furthermore, the mixed profile of the REOPENED status indicates that it might be useful to further decompose this phase. REOPENED covers many different steps in the resolution process, and looks like a condensed version of a complete trajectory, mixing together features of all the other phases. Identifying distinct stages in this status might be useful for research, and may as well be for the collective itself. In the same way that the collective finds it useful to decompose the initial trajectory in several steps, there may be advantages in characterizing a reopening with more details.

4.4. INTERACTION AND BUG DURATION

One of the largest variations across the Bugzilla repository is the time bug reports stay open. Some reports stay open only a few hours, while others take years to be resolved. Figure 6 shows the distribution of the 80,000 resolved

reports contained in our Bugzilla snapshot, based on the duration of their resolution process.

As can be seen in Figure 6, there is a large percentage of bug reports that take a significant amount of time to be solved. More than half (60.4%) of the bug reports took more than thirty days to be resolved, and over a fourth (27.7%) took more than six months.

Bug resolution speed has been proposed as a measure of collective efficiency (Crowston et al., 2004) and can be used to study the impact of different practices on the outcome of the resolution process. Bug resolution depends on technical factors such as the complexity of the problem, but also on social and organizational factors, such as resource availability and coordination. Since coordination occurs in a large part through linguistic interaction, variations in patterns of interaction among participants may be related to variations observed in the duration of the resolution process. Our objective is therefore to investigate the eventual relationship between profiles of interaction and duration of the bug resolution process.

The analysis was carried out on the 155 completed reports (status = RESOLVED) present in our sample. Reports were split among four equal-size subsets based on their duration, with the following limits: less than 8 days, less than 56 days, less than 188 days, and over 188 days. Figure 7 shows the variation in dialogue act profiles for each category (significance levels are given in the figure; the statistics table is omitted as the testing procedure is identical to the previous experiment).

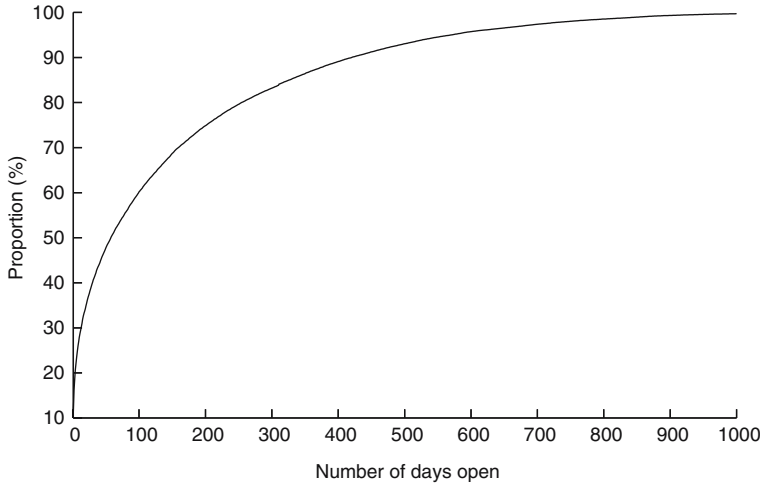

Figure 6. Cumulative distribution of bug reports over the duration of the resolution process ($N_r = 80881$).

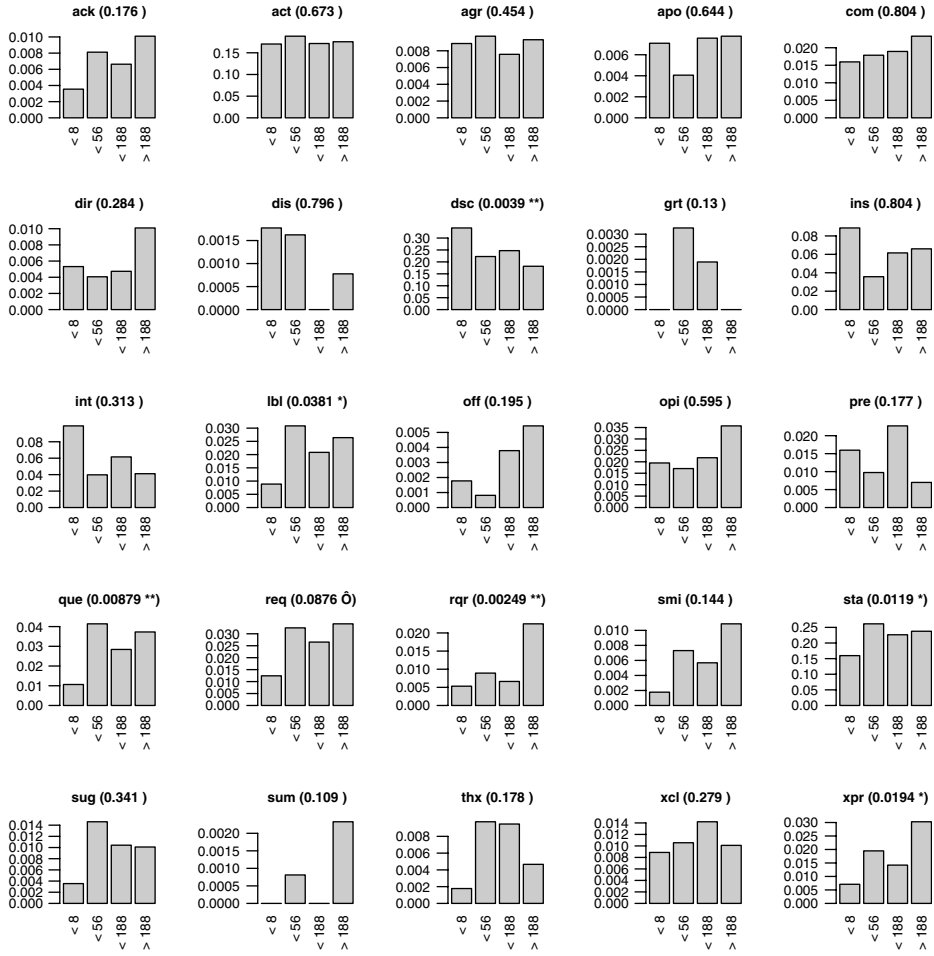

Figure 7. Dialogue act proportion by duration. Reports were grouped in categories of: less than 8 days, less than 56 days, less than 188 days, and more than 188 days (Kruskal–Wallis: $N_r = 155$; $N_{da} = 4140$; df = 3; p: *** < 0.001, ** < 0.01, * < 0.05, ' < 0.1).

Initial analyses show little correlation between interaction and duration. Only five categories show significant variations over bug duration at the 0.05 level. Some other categories display promising trends, but variations here are not as clear as the variations related to stages of resolution. While interpretations concerning the relationship between interaction and bug duration would be risky given the data, the results provide interesting observations about the problems faced by such studies.

First, as in every statistical measure, sample size is critical. The sample used in this analysis was almost half the size of the one used for the previous analysis, which is a likely cause for the lack of significant results. Increasing

the size of the sample should make it possible to draw more reliable conclusions, be they positive or negative.

However, beyond the size of the sample, we observe a wide variability in our current sample that may introduce too much noise. The sample used for this analysis was selected to be as representative of the corpus as possible, with the primary goal of training our automated tools; therefore, no parameters were controlled during the sample selection process. This results in a large variability on every dimension (bug duration, types of problems, proportions of dialogue acts, etc.). While it is suitable for our primary task, it weakens potential relationships among specific parameters. For instance, in our present analysis, duration is likely to be influenced by many other parameters besides patterns of interaction (e.g., technical difficulty, resource availability, etc.), which are not controlled in the sample. In order to properly test for the impact of interaction on duration, it would be necessary to control for all these other parameters.

Variability is also caused by the fact that bug duration may not be a precise enough measurement. Taken alone, process duration might be a misleading measure of collective activity, because it may cause distinct patterns of activity to be mixed together. For instance, many bugs take time due to insufficient resources, while others may be due to social misalignments. By only measuring the time a bug takes to be resolved, we may not be able to single out specific patterns of coordination from problems of resource allocation. It is only once the variability created by other factors is controlled that we can hope to see meaningful variations in interaction patterns.

This attempt at modeling relationships between duration and profiles of interaction therefore identifies the need for both better sampling based on an understanding of the sources of variability in the corpus, and for the careful selection of precise outcome measures that are relatively well understood in terms of dependencies with other parameters, so that specific phenomena can be singled out for study. As for increasing the size of the samples used for analysis, it is the focus of the following section.

5. Automating dialogue act recognition

In the previous section we demonstrated the interest of an approach based on the semantic modeling of linguistic interactions, and on the study of the relationships between such representations and more formal measures of collective activity. However, these studies relied on manual annotations of a small sample. This seriously limits the potential of the approach and the domain of possible analyses, as we found out in Section 4.4. In order to scale to the size of entire corpora containing years of interaction between participants in a collective, we need automated ways to annotate interaction with semantic information such as the dialogue acts we use in our analyses.

Our approach consists in using machine learning techniques to extract linguistic patterns that can reliably identify specific types of dialogue acts, based on a small initial sample of manually annotated data. Our method is based on the following observations:

– As we already mentioned, manually annotating entire corpora is not feasible. However, annotating a small number of reports can be done in a reasonable amount of time, and can be used as training data for a machine learning approach.
– Different collectives may present significantly different patterns of interaction. Using a machine learning approach makes it possible to train the tools on specific corpora with relatively little extra effort (simply requiring the annotation of a sample of the new corpora). Inversely, using approaches which attempt to directly describe the patterns (such as rule-based systems) instead of automatically learn them would require at least partial reengineering for each new corpora.
– Annotating reports is easier than elaborating extraction rules. A machine learning approach makes it more feasible to carry out studies on new corpora or using different semantic information, since it does not require as much technical expertise.

In the following paragraphs, we describe our procedure and present our current results in automated annotation, along with analyses of where future improvements can be made. The process is decomposed into three main phases: preprocessing, manual sample annotation, and machine learning.

5.1. PHASE 1: PREPROCESSING

The first phase consists in transforming the raw data into a form that can be further processed using automated tools. Comments are directly extracted from the Bugzilla database snapshot in raw textual form, and have to undergo a series of steps aimed at formatting the data and adding information necessary for the machine learning process. The three following steps are performed:

(a) *Segmentation of comments into tokens*. In most cases, tokens correspond to words, but may also represent specific entities such as times, URLs, or composed words.
(b) *Lexical and grammatical annotation of comments*. Information is added to each token, providing the part of speech (verb, noun, etc.) and lemmatized form of the token.
(c) *Segmentation of comments into utterances*. Utterances constitute the elementary entity used to annotate dialogue acts. In many cases, utterances correspond to sentences, but several utterances can occur in a single sentence.

In our sample, steps *a* and *b* were done automatically. Step *a* was carried out using a segmenter developed specifically to handle some of Bugzilla's specificities (such as program naming schemes and computer code entities). Step *b* relied on standard morphosyntactic tools. Step *c* was done manually to insure high reliability, as utterance segmentation poses a number of issues from a computational perspective that we have not yet addressed.

5.2. PHASE 2: MANUAL SAMPLE ANNOTATION

This phase consists in creating example data to be used as training data in the machine learning phase. A small sample was manually annotated using the taxonomy of dialogue acts detailed in Section 3.3, resulting in utterance/dialogue act pairs.

For this phase, we developed an annotation tool aimed at facilitating the task of annotating utterances. Our tool, Zentag, allows for the rapid annotation of dialogue acts through a simple graphical interface.

5.3. PHASE 3: MACHINE LEARNING

Machine learning techniques are used to identify patterns of linguistic features allowing for the annotation of the type of dialogue act performed by a given utterance. Once a learning algorithm is trained, it can be used to automatically annotate the entire corpus with dialogue act information. The learning step is composed of two phases.

5.3.1. *Feature extraction*

Before a correspondence between linguistic patterns and dialogue acts can be established, features on which to base the patterns have to be extracted from the utterances. We use a series of filters we implemented to extract information such as:

- *Information about words present in the utterance*. This includes the words themselves (either surface or lemmatized form), but also the potential presence of modals verbs, "wh-" forms (e.g., what, who, ...), evaluative language, etc.
- *Information about specific chains of words (n-grams)*. These chains take into account the relative position of the words in the chain. For instance, "you do" and "do you" would be two different chains, and are likely to have a different semantic role.
- *Information at the utterance level*. This includes the tense of an utterance, its form (interrogative, affirmative), etc.

5.3.2. *Learning process*

Feature extraction provides an alternative representation of utterances as a set of features and a corresponding category of dialogue act. A learning algorithm uses this representation to associate patterns of features to specific categories of dialogue act.

In the following section, we report results obtained with the Weka learning environment (Witten and Frank, 2000) using an implementation of the C4.5 decision tree algorithm. In some cases, better results may be attained from the tuning of more specific algorithms, but we currently are elaborating the entire learning procedure and are not focusing on optimization. Decision trees have the advantage of not requiring much tuning and being easily interpretable, thus reducing the amount of time needed for setting up experiments. We see tuning improvements as too costly for the experimental phase of our research, in which we favor ease of development and speed of experimentation.

5.4. EVALUATION OF AUTOMATED ANNOTATION OF DIALOGUE ACTS

Our evaluation is based on the manually annotated sample described in Section 4.1.[7] Evaluation was done using the complete sample following standard evaluation procedures in natural language processing (ten-fold cross-validation, percentages indicate accuracy).

5.4.1. *Results*

The first column (TOT) of Figure 8 shows the results of automated annotation obtained with varying sizes of the training set. The best overall accuracy is 64.2%, compared to a baseline[8] of 25.5%. These results are consistent with those summarized in (Stolcke et al., 2000, p. 365) given the number of categories learned and the size of the sample used.

The overall results (column TOT) have to be interpreted with care, as the distribution of dialogue acts is highly skewed (see Figure 3) and tends to bias the learning process. Because of the skew in distribution, the number of training instances used in the learning process varies considerably between categories. Categories such as `disagreement`, `offer` and `acknowledgement` have less than 50 instances each, while the three dominant categories (`action`, `description`, `statement`) range between 1000 and 2000 instances.

Since by default machine learning algorithms reward overall accuracy, an increase of 1% in accuracy in a category such as `statement` will weight much more than a 1% increase in the `offer` category, because of the total number of instances in each category (1947 vs. 27 instances). To reduce this effect, we can set a limit m to the number of examples of each category that will be used as the training set during the learning process, by randomly selecting at most m instances out of the set available for each category. By setting a limit, we ensure that the difference in the number of instances between frequent and

infrequent categories is smaller, and we can observe the impact on the per-category accuracy. However, limiting the number of instances also means that there will be less training instances overall, which is likely to significantly influence the learning process as well.[9]

Each shade of grey in Figure 8 shows a different run of the learning algorithm on a sub-sample using a different value of m (darker shades indicate lower limits). We see that as the training limit increases, overall performance increases. However, significant drops in infrequent categories are observed. Several categories (such as `commitment` or `expression`) see their accuracy decrease as the sample becomes more and more biased toward high-frequency categories. This is compensated at the global level by an increase in accuracy in the frequent categories (`description` and `statement`, and `action` to a lesser extent). As the unbalance increases between infrequent and frequent categories, accuracy for infrequent categories becomes less relevant and is overruled by accuracy in frequent categories. This results in better labeling of frequent categories but at the expense of other less frequent categories.

In addition to this tradeoff, two other phenomena can be observed. Some categories are stable and likely to have reached their maximum accuracy given the current approach. Simple categories, such as `apology`, `smiley` or `thank` do not take many linguistic forms in the corpus and thus do not require many training instances, yielding quick saturation of the algorithm for these categories. On the other hand, some categories are not learned at all, and correspond to categories with very few instances in the sample: `disagreement` counts only eight instances, and `summon` six.

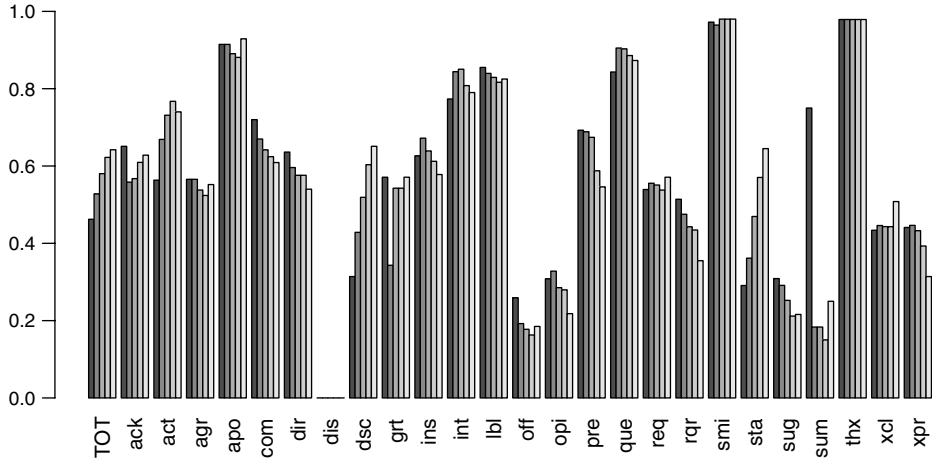

Figure 8. Accuracy of automated dialogue act annotation by category, with different training limits ($N_{da} = 7464$; per-category training limit $m = 100$, 250, 500, 1000, ∞; darker shades indicate lower limits).

5.4.2. *Improvements*

Given these results, increasing the size of the training data should increase both per-category and overall accuracy. The tradeoff illustrated above becomes less important if all categories contain more examples. In this way, more data can be used to train the learning algorithms without creating an unbalanced situation that favors frequent categories.

In addition to simply adding data, the use of more elaborate features should provide further increase in accuracy. Probably the most significant issue in machine learning is that most entities occur very few times, which makes identifying patterns a complex task, as the data presents large amounts of variation. Increasing the training size will always help, but other techniques aimed at abstracting utterances to increase similarity may be more viable. For instance, categorizing verbs into semantic classes might indicate that a sentence using the verb ''fix'' and another using the verb ''repair'' are semantically close, thus helping the learning process to establish similarities between the two sentences.

Finally, besides simply aiming for better overall performance, it is important to pay attention to the per-category accuracy, and to draw conclusions based on the contribution of each category to the final analysis, which is likely to vary depending on the experiments that are to be carried out with the data. Section 4 showed that not all dialogue act categories are equally informative in a given context of analysis. In our context, better performance does not mean better overall accuracy, but better suitability for a given analysis. Such increase in performance may be attained by defining priorities in the learning process concerning what categories are critical for experimentation. Even more simply, it may not be necessary to learn every category for a given analysis, in which case less complex learners (which generally translate into more efficient ones) can be trained on a subset of relevant categories, instead of on the complete set.

6. Discussion: Issues in computational interaction modeling

In the following paragraphs, we summarize the issues we have encountered so far, discuss what they mean in the context of studies of distributed practices, and provide directions on how we intend to address them in the future.

6.1. DATA AND AUTOMATED APPROACHES

Data collected from Bugzilla is very noisy, in that there are many typographical mistakes, incomplete sentences, specialized words, and even

computer code or other non-linguistic entities. Such problems will be present to a certain extent regardless of the collective studied. The quality of the corpus may vary significantly depending on the type of collective but also on the communication media employed, which will impact on the style of interaction. However, most natural interactions will be noisier than traditional corpora used in computational linguistics. Consequently, language processing tools that perform well on relatively clean corpora see their performance drop very significantly on corpora such as the Bugzilla corpus, and new approaches are needed to handle the higher noise level.

Fortunately, research in computational linguistics is increasingly preoccupied with "web data" and we can expect important improvements in natural language processing tools, which should translate directly into improvements for empirical computational approaches.

6.2. SAMPLING

Sample size is an important factor in statistical analyses, and given the relative complexity of our measures and the wide variation in the data, it is important to collect large samples. Despite this limitation, we were able to demonstrate some significant effects with a small sample, but there is no doubt that the scale and types of analyses are restricted by our current sample, as shown by our third experiment (Section 4.4).

Our approach offers a way to overcome this limitation by automating annotation, which constitutes the most time consuming step. As our tools' performances increase, more complex analyses will be possible on larger samples (complexity and size are tied, as complexity introduces more variability, which can be overcome statistically with larger sample sizes).

Beyond sample size, sample selection is critical. Getting more data is always preferable, but in cases such as Bugzilla, where variability is large and multiple constraints weight in the resolution process, it is important to select samples geared toward specific analyses. For instance, in our experiment on bug duration, stronger effects may have been found if the sample had been selected specifically to study this factor. By selecting only short- and long-duration bugs, the difference between the two groups would have been maximized. Instead, our current sample has been primarily designed for the training of our machine learning tools, and to allow for exploration of multiple hypotheses, and as such is an unbiased sample of the entire population of Bugzilla reports.

Further analyses should design samples that facilitate experimentation and maximize the chance of showing specific effects, by isolating variables of interest from other potential sources of variation.

6.3. MEASUREMENTS

The issue of sampling leads to another one: measurement. The success of models such as the one proposed in this paper depends on the selection of appropriate measures of the outcomes of a given process, since it is what provides a frame for the meaningful interpretation of observed patterns of interaction. As we found out in our last experiment (Section 4.4), selecting a good outcome measure is not a straightforward task. The selection not only requires that we be able to measure the outcomes, but also that we possess a good understanding of the relationships between the outcome measure and other processes. Computational techniques, by their inherent formality, require that measures be explicit and "formalizable". We saw in Section 4.4 that some seemingly simple measure such as bug duration can turn out to be too imprecise. Measures standing for very complex, entangled processes therefore need to be further taken apart before any model can be built.

With the increasing use of computational techniques, efforts have been started to identify useful measures for the study of collective activity in the context of FOSS research (Crowston et al., 2004). Creating such a "portfolio" of measures would be an invaluable asset for research on collective practices, by providing agreed upon, well understood, and meaningful measurements that could be used in studies such as ours. Furthermore, these measures could also be used more generally to establish links between different studies and to compare results.

6.4. SEMANTICS OF INTERACTION

Part of the difficulty with analyzing the semantic content of interactions is to make categories that are useful and sufficiently clearly defined so that they can be learned with computational approaches. Dialogue acts represent a tradeoff between what can possibly be extracted and the expressivity of the entity (the more semantic the structures, the harder it is to extract them reliably using computational techniques). Dialogue acts provide information about "what is going on" in the collective, and therefore are appropriate for the study of collective activity. This tradeoff between feasibility and expressivity is something that needs to be cautiously considered when designing new analyses, and also when moving to other types of corpora. For instance, in the case of large discussion forums, dialogue acts might not represent an appropriate level of analysis, and larger entities (such as arguments, themes, etc.) might be more relevant. However, in the present case, comments contain only a few utterances, and most often do not contain very developed arguments, making dialogue acts a useful representation.

Regardless of the type of corpus, dialogue acts do not constitute the only type of semantic marker, and research is needed to identify other entities that can contribute to a model of interaction *in the context of studying collective*

practices. Furthermore, once several levels of semantics have been identified, it becomes important to understand how to link the different levels into a coherent and meaningful model.

Many efforts in computational linguistics and artificial intelligence already aim at representing interaction for different purpose, such as question answering, information retrieval, or text mining (Mitkov, 2003, part 3). We believe the requirements of DCP studies (and of what we could call "computational sociology" in general) are significantly different and require an approach that explicitly addresses these particularities and needs. In the same way that linguistics provides an important theoretical foundation to language processing, computational tools such as the ones we are developing cannot ignore social models of distributed collectives.

6.5. GENERICITY

When designing a model of interaction, we have to ask if such a model can be generalized to other collectives (1) in similar settings, and (2) in different settings. A significant part of our motivation to automate analyses finds its root in the capacity to reproduce analyses on multiple corpora, in order to carry out comparative studies. Thus, models have to be able to function on different corpora. This involves several levels of complexity, as different corpora might mean:

- Varying levels of accuracy of computational tools, depending on the quality of the data, and the complexity of the interaction (Section 6.1).
- Different measures and outcome variables, due to the presence of different processes, structures, or organization in different collectives (Section 6.3). The importance of this issue is likely to increase as we compare collectives using 1) the same environment (e.g., two collectives using Bugzilla), (2) different ones, but similar in functionalities and representations (e.g., Bugzilla and Sourceforge's bug reporting system), and (3) completely different ones (e.g., Bugzilla and a developers' mailing list).
- Different levels of semantic analysis, required by different types of interaction (Section 6.4).

These differences across corpora and studies should be kept in mind when designing computational models, to keep the need for modifications in the tools and models to a minimum when moving to different corpora, and to be able to reproduce and compare analyses in a variety of contexts. We expect further research to lead to a better understanding of the modeling requirements in different contexts, leading in turn to a better understanding of the levels of representation, types of tools, and methods to be used to minimize the amount of development specific to a given corpus, and inversely to maximize the portability of the models across different corpora.

6.6. CONTRIBUTION: WHAT, AND FOR WHOM?

Finally, while we have focused so far on developing computational tools to support research on distributed collectives, it would be restrictive to think of our approach only in these terms. Models based on the semantics of linguistic interactions can benefit different types of users. First, as shown in this paper, they can help researchers to better understand collective practices. Second, they can provide models for designers to build better tools for the support of collective activity based on an improved understanding of the role of interactions in collective activity. Finally, they can offer new ways for the collective itself to access information and keep track of its evolution, by enabling a form of reflexivity.

Computational models can also be used with different objectives. In this paper, we have shown how to observe and evaluate the functioning of a collective, using descriptive techniques. Alternatively, once processes are better understood and relationships established, tools could be developed to predict future evolutions of a given collective, and eventually to run simulations to test hypotheses.

These many possibilities do not mean however that such tools and representations will be useful or appropriate in all of these distinct settings. A reflection on the uses and contributions of computational models, along with empirical validation in the form of tools providing the different functionalities we mentioned should be undertaken.

7. Conclusion

Our objective with this paper was to provide a picture of the different aspects involved in computational studies of distributed collective practices based on our experience in modeling interactions, implementing tools, and analyzing resulting representations. We have presented our tools and model, provided illustrations of how they could be used to highlight properties of the interactions and processes occurring in Bugzilla, and discussed issues that remain to be addressed.

Automated approaches to the study of collective practices are going to be needed to face the amount of data that is accumulating as the by-product of the activity of these collectives. More specifically, techniques that go beyond simply using structural information to also look at the semantics (the content) of the interactions will be necessary. Much research remains to be done to address the issues we have identified, and to develop computational tools that have the appropriate coverage, reliability and genericity to provide useful insights into distributed collective practices. Our research represents a first step in this direction.

Notes

1. A trajectory is the series of changes in status that have occurred throughout the life of a bug report, as recorded in the activity log of a bug report.
2. The following categories of dialogue acts were considered to be "direct" dialogue acts: `ack`, `agr`, `apo`, `dir`, `dis`, `grt`, `lbl`, `off`, `que`, `req`, `sug`, `sum`, `thx`.
3. The Switchboard score was obtained by adding the proportion of all the categories involved in "direct" dialogue listed in Table 2 in (Jurafsky et al., 1998).
4. The CLOSED status was excluded from the analysis as it did not appear in the sample and appears in less than 1% of the reports in the corpus. The `unknown` category of dialogue acts was excluded as it does not have a meaningful interpretation.
5. The Kruskal–Wallis test was used as the sample departs significantly from ANOVA assumptions.
6. This, however, cannot be seen simply by using dialogue acts. Information about the propositional content of the acts would be required. The observations given here about the content of the descriptions result from informal analyses made during the manual annotation process.
7. Not including the `unknown` category.
8. The baseline is computed by always assigning the most frequent category to every instance in the test data. In the present case, the most frequent category is `statement`, which represents 25.5% of the utterances in the sample.
9. In general, more data means better machine learning performances. This tendency diminishes as the learning algorithm reaches saturation. However, in our case the amount of data used for learning is relatively small, so we can expect that more data will lead to better performances.

Appendix A – Dialogue act taxonomy

Details of Table 1: For each category, a definition and a few examples are provided. Examples are left uncorrected.

Assertives

Describe an actual state of the world.

STATEMENT (`sta`)

Present objective information about a fact or the current state of the world.

`my pc has an amd k6 233 mhz cpu and 64 mb ram and a 4,3 gb quantum fireball se [1005-5]`

`it in not the exact same problem [18090-16]`

DESCRIPTION (`dsc`)

Present objective information about the behavior (or state) of the focus of the collective (which is to say Mozilla).

`mozilla 1998--10--03 crashed [1005-1]`

`on a freshly booted system, viewer.exe refuses to load anything more than the banner.gifs [4020-3]`

PRESCRIPTION (`pre`)

Present objective information about what the behavior (or state) of the focus of the collective should be. Prescriptions usually appear as contrast to

descriptions (descriptions detail how things actually are, and prescriptions how things are expected to be).

i the bookmarks manager should open and have usable menus and editable bookmarks [9380-1]

docshell should be setting validate_always [96480-21]

Commissives

Commit the speaker (to varying levels) to a future action (the action can itself be linguistic or not).

COMMIT (com)

Commit the speaker to a future course of action.

i'll take a look at this [15075-25]

this stuff will getnbsp;all fixed up again [17085-38]

OFFER (off)

Indicate that the speaker is willing to perform a future course of action (but does not commit to doing it through the enunciation of the utterance).

i have a fix for this [2010-17]

if necessary, i will file a seperate bug [10050-27]

Directives

Attempt (to varying levels) to make the hearer perform some action (linguistic or not).

REQUEST (req)

Request that an action be taken.

could you fix this one? [2010-3]

please verify [9045-20]

REQUIREMENT (rqr)

Request that a behavior (or state) related to the focus of the collective (Mozilla) be made to happen.

at the least, it needs to be corrected soon [7035-1]

we really need something that works on all platforms [97485-23]

QUESTION (que)

Request information about the current state of the world.

how's this look now? [4020-12]

can you reproduce under windows 98? [12060-11]

SUGGESTION (sug)

Suggest a course of action.

i think the border code is at the point where it needs a restructuring [2010-11]

rather than just delete after x days, it would be more useful to perform other actions [11055-17]

SUMMON (sum)

Request acknowledgement, with the main purpose of establishing, maintaining, or reestablishing contact.

eh? [21105-17]
Simon? [54270-27]

INSTRUCTION (ins)

Provide instruction to perform an action (in the way a recipe does). It does not constrain the interlocutor(s) to a course of action, but specifies how to perform the action in case the interlocutor would want to do it.

for definitions of milestones, see: http://www.mozilla.org/projects/seamonkey/milestones/50/m7plan.html, [5025-3]
go to bookmarks on the menu bar [9045-4]

Expressives

Describe a mental state of the speaker about the state of affairs described by the utterance.

OPINION (opi)

Present a subjective information or a personal position on a state of the world.

it's crystal-clear in the css1 spec [16080-14]
i love it it works great! [19095-4]

EXPRESSION (xpr)

Present a (subjective) information about the mental state of the locutor.

wish i could test more [16080-10]
i'd like to think this means the bug is fixed [18090-28]

ACKNOWLEDGEMENT (ack)

Acknolwedge a previous utterance.

ok [5025-4]
yeah [17085-37]

AGREEMENT (agr) / DISAGREEMENT (dis)

State agreement or disagreement with a previous utterance or a state of the world.

i agree with that [109545-38]
totally disagree with your assessement [42210-33]

EXCLAMATION (xcl) / SMILEY (smi)

All forms of exclamations without informational content. Smileys are a specific case.

grumble [10050-8]
good luck! [19095-42]
:-),; -), etc.

APOLOGY (apo) / GREETING (grt) / THANK (thx)

Respectively present an apology, a greeting, and a thank.

i am *really* sorry about the spam [2010-29]
hey [59270-22]
many thanks for looking into this [124620-113]

Performatives

Perform the action represented by the utterance, and bring about the corresponding new state of the world.

ACTION (act)

Perform an action directly or reports the performance of an action related to the utterance.

```
marking verified [1005-14]
*** this bug has been marked as a duplicate of X ***
```

Others

INTRODUCTION (int)

Any segment which purpose is to introduce or place in context another utterance.

```
here's an example, from one of my folders: [19095-21]
as you can see [19095-22]
```

LABEL (lbl)

Any segment used to explicitly address another utterance to a specific addressee.

```
Tom – [...] [12060-2]
Submitter: [...] [56615-4]
```

UNKNOWN (ukn)

Any segment which force is unknown or uncertain.

References

Allen, James F. and Mark G. Core (1997): *Draft of DAMSL: Dialog Act Markup in Several Layers*. NY: Department of Computer Science, University of Rochester.

Bannon, Liam (ed.) (1995): Commentary Section about the Suchman-Winograd Debate. *Computer Supported Cooperative Work*, vol. 3, no. 1, pp. 29–95.

Bugzilla, Team (2005a): Anatomy of a Bug. http://www.bugzilla.org/docs/tip/html/bug_page.html (visited: Nov 12, 2005).

Bugzilla, Team (2005b): Bug's Life Cycle. http://www.bugzilla.org/docs/tip/html/lifecycle.html (visited: Nov 12, 2005).

Bunt, Harry C. (2005): A Framework for Dialogue Act Specification. *Presented at the Fourth ISO/SIGSEM Workshop on Multimodal Semantic Representation*, Tilburg, The Netherlands. Available at: http://let.uvt.nl/general/people/bunt/docs/fdas.ps.

Carstensen, Peter H., Cartsen Sørensen and Tuomo Tuikka (1995): Let's Talk About Bugs!. *Scandinavian Journal of Information Systems*, vol. 7, no. 1, pp. 33–54.

Core, Mark G. and James F. Allen (1997): Coding Dialogs with the DAMSL Annotation Scheme. *Working Notes of the AAAI Fall Symposium on Communicative Action in Humans and Machines*, Boston, MA, pp. 28–35.

Crowston, Kevin (1997): A Coordination Theory Approach to Organizational Process Design. *Organization Science*, vol. 8, no. 2, pp. 157–175.

Crowston, Kevin, Hala Annabi, James Howison and Chengetai Masango (2004): Towards A Portfolio of FLOSS Project Success Measures. *Proceedings of the Fourth Workshop on Open*

Source Software Engineering, International Conference on Software Enginnering (ICSE'04), Edinburgh, Scotland.

Crowston, Kevin and James Howison (2005): The Social Structure of Free and Open Source Software Development. *First Monday*, vol. 10,, 2.

De Souza, Cleidson R.B., David F. Redmiles, Gloria Mark, John Penix and Maarten Sierhuis (2003): Management of Interdependencies in Collaborative Software Development. *Proceedings of the International Symposium on Empirical Software Engineering (ISESE'03)*, Rome, Italy, pp. 294–303.

Flores, Fernando, Michael Graves, Brad Hartfield and Terry Winograd (1988): Computer Systems and the Design of Organizational Interaction. *ACM Transactions on Office Information Systems*, vol. 6, no. 2, pp. 153–172.

Gacek, Cristina and Budi Arief (2004): The Many Meanings of Open Source. *IEEE Software*, vol. 21, no. 1, pp. 34–40.

Gasser, Les and Gabriel Ripoche (2003): Distributed Collective Practices and Free/Open-Source Software Problem Management: Perspectives and Methods. *Proceedings of the Conference on Cooperation, Innovation and Technologie (CITE'03)*, Troyes, France.

Gasser Les, Gabriel Ripoche and Robert J. Sandusky (2004). Research Infrastructure for Empirical Science of F/OSS. *Proceedings of the First International Workshop on Mining Software Repositories (MSR), International Conference on Software Engineering (ICSE'04)*, Edinburgh Scotland.

Gerson, Elihu, M. and Susan Leigh Star (1986): Analyzing Due Process in the Workplace. *ACM Transactions on Office Information Systems*, vol. 4, no. 3, pp. 257–270.

Hayes-Roth, Barbara and Frederick Hayes-Roth (1979): A Cognitive Model of Planning. *Cognitive Science*, vol. 3, no. 4, pp. 275–310.

Herbsleb, James D. and Rebecca E. Grinter (1999): Splitting the Organization and Integrating the Code: Conway's Law Revisited. *Proceedings of the International Conference on Software Engineering (ICSE'99)*, Los Angeles, CA, pp. 85–95.

Howison, James, Megan S. Conklin and Kevin Crowston (2005). OSSmole: A Collaborative Repository for FLOSS Research Data and Analyses. *Proceedings of the First International OSS Conference (OSS'05), Genova, Italy*.

Jurafsky, Daniel, Rebecca Bates, Noah Coccaro, Rachel Martin, Marie Meteer, Klaus Ries, Elizabeth Shriberg, Andreas Stolcke, Paul Taylor and Carol Van Ess-Dykema (1998): *Switchboard Discourse Language Modeling Project Final Report*. Center for Speech and Language Processing, Johns Hopkins University.

Mitkov, Ruslan (ed.) (2003): *The Oxford Handbook of Computational Linguistics*. New York, NY: Oxford University Press.

Mozilla Organization (2005): Mozilla's Bugzilla Repository. http://bugzilla.mozilla.org. (visited: Nov 12, 2005).

Raymond, Eric S. (2001): *The Cathedral and the Bazaar*. Sebastopol, CA: O'Reilly.

Ripoche, Gabriel and Les Gasser (2003): Scalable Automatic Extraction of Process Models for Understanding F/OSS Bug Repair. *Proceedings of the Conference on Software and Systems Engineering and their Applications (ICSSEA'03)*. Paris, France.

Sandusky, Robert J. (2005): *Information, Activity and Social Order in Distributed Work: The Case of Distributed Software Problem Management*. PhD Thesis, Graduate School of Library and Information Science, University of Illinois at Urbana-Champaign.

Scacchi, Walt (2002): Understanding the Requirements for Developing Open Source Software Systems. *IEE Proceedings Software*, vol. 149, no. 1, pp. 24–39.

Schmidt, Kjeld and Carla Simone (1996): Coordination Mechanisms: Towards a Conceptual Foundation of CSCW Systems Design. *Computer Supported Cooperative Work*, vol. 5, no. 2–3, pp. 155–200.

Schmidt, Kjeld and Carla Simone (2000). Mind the Gap! Towards a Unified View of CSCW. *Proceedings of the Fourth International Conference on the Design of Cooperative Systems (COOP'00)*. Sophia Antipolis, France.

Searle, John R. (1975): A Taxonomy of Illocutionary Acts. In K. Gunderson (ed.): *Language, Mind and Knowledge*.. Minneapolis, MN: University of Minneapolis Press, pp. 344–369.

Stolcke, Andreas, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Dan Jurafsky, Paul Taylor, Rachel Martin, Carol Ess-DykemaVan and Marie Meteer (2000): Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech. *Computational Linguistics*, vol. 26, no. 3, pp. 339–373.

Strauss, Anselm (1988): The Articulation of Project Work: An Organizational Process. *Sociological Quarterly*, vol. 29, no. 2, pp. 163–178.

Suchman, Lucy (1994): Do Categories Have Politics?. *Computer Supported Cooperative Work*, vol. 2, no. 3, pp. 177–190.

Suchman, Lucy (1996): Supporting Articulation Work. In R. Kling (ed.): *Computerization and Controversy: Values, Conflicts and Social Choices,* 2nd edition, San Diego, CA: Academic Press, pp. 407–423.

WebSideStory (2005): Firefox's Market Share Nears 7 Percent. `http://www.webside-story.com/products/web-analytics/datainsights/spotlight/05-10-2005.html` (visited: May 29, 2005).

Winograd, Terry (1987): A Language/Action Perspective on the Design of Cooperative Work. *Human-Computer Interaction*, vol. 3, no. 1, pp. 3–30.

Winograd, Terry (1994): Categories, Disciplines, and Social Coordination. *Computer Supported Cooperative Work*, vol. 2, no. 3, pp. 191–197.

Witten, Ian H. and Eibe Frank (2000): *Data Mining: Practical Machine Learning Tools with Java Implementations*. San Francisco, CA: Morgan Kaufmann.

Yamauchi Yutaka, Makoto Yokozawa, Takeshi Shinohara, and Toru Ishida (2000): Collaboration with Lean Media: How Open-Source Software Succeeds. *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW'00)*, Philadelphia, PA, pp. 329–338.