# A structurational perspective on leadership in Free/Libre Open Source Software teams

Kevin Crowston*, Robert Heckman*, Hala Annabi+ and Chengetai Masango*
* Syracuse University, School of Information Studies
Syracuse, NY, USA
{crowston, rheckman, cmasango}@syr.edu
+ University of Washington, The Information School
Seattle, WA, USA
hpannabi@u.washington.edu

*Abstract* – **In this conceptual paper, we present a structuration-based theory of leadership behaviours in self-organizing distributed teams such as Free/Libre Open Source Software development teams. Such teams are often composed of members of relatively equal status or who are so disparate in background that formal organizational status seems irrelevant, reducing the usual leadership cues provided by organizational status and title. Building on a functional view of leadership and structuration theory, we suggest that leaders are individuals who develop team structures that then guide the actions of team members. Specifically, we examine structures of signification in the form of shared mental models, structures of domination in the form of role structures and structures of legitimation in form of rules and norms. The main contribution of our paper is the integration of various social theories to describe emergent leadership behaviours in distributed teams. We develop a set of propositions and illustrate with examples taken from Free/Libre Open Source Software development teams. We conclude by suggesting practical implications as well as future research that might be conducted to test and further elaborate our theory.**

## I. INTRODUCTION

In this paper, we develop a theory of leadership behaviours in self-organizing distributed teams such as the core team of developers in Free/Libre Open Source Software development projects. FLOSS[1] development is an extreme example of self-organized distributed work and thus poses particular challenges for the development of leadership. Developers contribute from around the world, meet face-to-face infrequently (or not at all), and coordinate their activity primarily by means of information and communications technologies (ICT) [2, 3]. Projects often do not have appointed leaders [4]. As well, many (though by no means all) programmers contribute to projects as volunteers, without being paid or even working for a common organization, again minimizing the effectiveness of traditional organization and leadership. For example, in a team of volunteers, it is usually not possible for team leaders, if they exist, to formally assign a task to a particular individual and then monitor performance; rather they often must wait until an interested individual chooses to perform it. This heavy reliance on self-organization sets FLOSS projects apart from most other distributed teams and has particular implications for the development of leadership in these teams.

While distributed work has many potential benefits, distributed workers face many real challenges. Watson-Manheim, Chudoba, & Crowston [5] argue that distributed work is characterized by numerous discontinuities: a lack of coherence in some aspects of the work setting (e.g., organizational membership, business function, task, language or culture) that hinders members in making sense of the task and of communications from others [6], or that produces unintended information filtering [7] or misunderstandings [8]. These interpretative difficulties in turn make it hard for team members to develop shared mental models of the developing project [9, 10]. A lack of common knowledge about the status, authority and competencies of participants can be an obstacle to the development of norms [11] and conventions [12]. The separation between members may ultimately result in a failure of the team to be effective [13-16].

Distributed teams face particular problems with leadership. Though leadership is one of the most studied topics in organizational and management research, relatively little research has been conducted on the nature of leadership in self-organizing distributed teams [17-19]. Such teams are often composed of members of relatively equal status or who are so disparate in background that formal organizational status seems irrelevant, reducing the usual leadership cues provided by organizational status and title. Such teams often have no appointed leader, and their members may or may not have significant prior experience working with one another. In such cases, rather than being appointed or even elected, a leader or leaders may emerge gradually, and such emergent leadership may be completely unrelated to organizational position or status. As work teams become more distributed, these traits may become more pronounced. The most effective types of leadership behaviour in these new organizational forms may be very different than the behaviours appropriate to the centralized, hierarchical, single-leader paradigm.

In this conceptual paper, we build on a functional view of leadership and structuration theory to develop a theory of leadership behaviours in one particular type of distributed team, FLOSS project teams. The main contribution of our paper is the integration of various social theories to develop theoretical propositions about emergent leadership behaviours in self-organizing distributed teams. Our paper thus provides direction for future research by suggesting what concepts and relationships to study and what kinds of data to collect. In the following section, we introduce our theoretical bases and develop a set of propositions. We conclude by describing directions for future research to test or further refine our theory.

## II. A STRUCTURATIONAL PERSPECTIVE ON LEADERSHIP IN DISTRIBUTED TEAMS

In the two sections that follow, we briefly describe the two theoretical lenses that inform our analysis of leadership behaviours in self-organizing distributed teams. First we describe a functional view of leadership that we believe best suited to describe the form of emergent leadership in self-organizing distributed teams. Next we present our rationale for adopting a structuration perspective to conceptualize leadership in these teams.

[1] FLOSS is a broad term used to embrace software developed and released under an "open source" license allowing inspection, modification and redistribution of the software's source without charge ("free as in beer"). Much though not all of this software is also "free software", meaning that derivative works must be made available under the same unrestrictive license terms ("free as in speech", thus "libre"). In our writing, we use the acronym FLOSS rather than the more common OSS to emphasize this dual meaning.

### A. A functional view of leadership

We have noted that leadership in self-organizing distributed teams is often emergent rather than appointed. To understand the processes of emergent leadership, we adopt a functional approach to leadership. In this approach, some behaviours serve as leadership functions in that they help the team to achieve its goals and perform effectively. Through the interactions of the team members, one or more individuals emerge to perform the leadership behaviours that the team requires. More than one individual may perform leadership behaviours, and different individuals may perform the same leadership behaviours at different times [20]. A functional approach to leadership is better suited to the study of emergent leadership behaviours in teams without *a priori* leadership status or assignments.

Research has distinguished several different types of leadership behaviours. Most functionalist theories make a broad distinction between *task leadership* behaviours and *team maintenance* leadership behaviours. The former are concerned with organizing, coordinating and performing the task(s) that constitute the team's primary work, while the latter are concerned with maintaining team morale, motivation and communication. Bales [21] believed that the functions of task and maintenance behaviours are opposed, and that teams should strive to find a balance or equilibrium between them. The opposition between task and maintenance behaviours also suggested to Bales that it would be more likely that different people would emerge to perform task and maintenance roles [20]. In addition to the task and team maintenance functions which leadership must satisfy, Ancona and Caldwell [22] argued that there are also leadership functions involved with maintaining relations with individuals and teams outside the team. Effective emergent leaders may be those who are able to attend to both the relational (social) and task-related needs of the team, adapting to the situation and manifesting the requisite behaviours as required [15, 23-26].

More specific to leadership roles, the functional approach to leadership suggest that more than one individual may perform leadership behaviours, and different individuals may perform the same leadership behaviours at different times [20]. Hackman and Walton [27] argue that these functions have a critical role in facilitating team performance.

### B. Structuration theory

To conceptualize the way that individuals' actions can serve to provide leadership in FLOSS development teams, we adopt a structurational perspective. Structuration theory [28] is a broad sociological theory that seeks to unite action and structure. Numerous authors have used a structurational perspective to frame empirical analyses of team activities [e.g., 29, 30-33] and in particular, the development of virtual teams [e.g., 34]. We chose this framework because it provides a dynamic view of the relations between team structures and the actions of those that live within, and help to create and sustain, these structures and guides the development of theory. In particular, it provides a framework for analyzing how the actions of one member (a team leader) might shape the actions of others, even in the absence of traditional modes of authority.

Structuration theory might be described as a meta-theory: that is, rather than specifically describing the relations between specific factors of leadership, structuration theory describes the form that that theory should take. Specifically, structuration theory is premised on the duality of structures: it suggests that a theory of leadership in distributed teams should consider the structures and actions in these teams and how the two are interrelated. By structure, we mean the rules and resources that influence, guide or justify individual action. These structures are "encoded in actors' stocks of practical knowledge" [1] and "instantiated in recurrent social practice" [35]. We specifically consider three kinds of rules and resources [1, 36]:

1. interpretive schemes that create structures of signification,
2. authoritative and allocative resources that create structures of domination, and
3. norms that create structures of legitimation.

For example, a particular process for testing software modules (an individual action) may be followed by individual developers because that process is the accepted norm within the team (i.e., because of a structure of legitimation). It should be noted that these structures are not viewed as existing some how separately from team members. Rather the actors and their practices compose the structure, in the much the same way that dancers compose a ballet or football players compose a game [37].

By the "duality of structure", we mean these structural properties of a social system are both the means and the ends of the practices that constitute the social system. As Sarason [38] explains, in structuration theory:

> "The central idea is that human actors or agents are both enabled and constrained by structures, yet these structures are the result of previous actions by agents. Structural properties of a social system consist of the rules and resources that human agents use in their everyday interaction. These rules and resources mediate human action, while at the same time they are reaffirmed through being used by human actors or agents." (p. 48).

Simply put, by doing things, we create the way to do things. For example, the norm of using a particular testing strategy is not a given, but rather itself the outcome of prior actions by developers. By following the norm, developers reinforce its legitimacy ("we always do it this way"); by taking different actions (e.g., skipping testing because it is seen to be too time-consuming or using a different approach because the accepted approach seems unable to identify important problems), they undermine its legitimacy, perhaps eventually changing the norm.

By relating structure and action across time, structuration theory provides a framework for understanding the dynamics of a team [39] and the relations between actions. Barley and Tolbert [1] note that structuration is "a continuous process whose operations can be observed only through time" (p. 100). Cassell [40] says, "to study the structuration of a social system is to study the ways in which that system, via the application of generative rules and resources, in the context of unintended outcomes, is produced and reproduced through interaction" (p. 119). Figure 1, adapted from Barley and Tolbert [1] shows the relation between institution (which the authors use synonymously with structure) and action, and how both evolve over time. In this figure, the two bold horizontal lines represent "the temporal extensions of Giddens' two realms of social structure: institutions and action," while the "vertical arrows represent institutional constraints on action" and the diagonal arrows, "maintenance or modification of the insti-

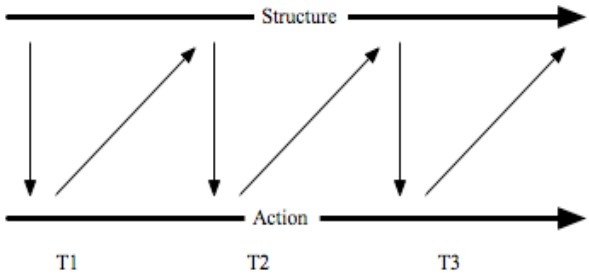


**Fig. 1.** A sequential model of the relation between structure and action (from [1], p. 101).

tution through action" (p.100). For example, the influence of a team norm on a developer to use a particular testing strategy is represented by a downwards vertical arrow, while reinforcement or changes to the norm due to actions is represented by an upwards diagonal arrow.

While it might first appear that a consideration of leadership would be relevant primarily to an understanding of the structures of domination, we expect leadership in self-organizing teams to be expressed through all three systems of structuration: signification, domination and legitimation. When viewed through the combined lenses of functional leadership theory and structuration theory, we argue that among the functions that are most essential to effective functioning of a team is the development of helpful structures that guide the actions of team members. In other words, we conceptualize leadership in self-organizing distributed teams in terms of developing structures—reinforcing existing structures or creating new ones—rather than simply control of resources. This view is one that can account for the patterns of fluid and emergent leadership found in self-organizing distributed teams.

## III. THEORY DEVELOPMENT: LEADERSHIP IN SELF-ORGANIZING DISTRIBUTED TEAMS

In the following three sections, we discuss each of the three systems of structure, developing propositions concerning the way structure guides effective team action (the downward vertical arrows in Figure 1) and the way team member actions generate structure (the upwards diagonal arrows) and describe leadership behaviours in these terms. To develop these propositions, we build on additional theories, which we introduce in each section.

Although the contribution of our paper is conceptual rather than empirical, to make the theory more concrete, we illustrate the generation and effects of structures with examples drawn from FLOSS development in two projects: the APACHE HTTPD project and the PLONE project. APACHE HTTPD (http://httpd.apache.org/) is the most commonly deployed Web server. PLONE (http://www.plone.org/) is a Web-based content management system. Both projects have been successful in harnessing the efforts of numerous developers from around the world to develop and distribute software. The examples are drawn from email or IRC (Internet Relay Chat) transcripts of developer interactions that are currently being analyzed in the empirical portion of our project. (Detailed description of our data collection approach would be inappropriate for this paper because our argument here is primarily conceptual, building on prior theory, rather than empirical, building on current data. The examples are offered as illustrations rather than proof of our theory.)

### *A. Interpretive schemes and structures of signification*

Individual actors' interpretive schemes create structures of *signification*. To describe how interpretive schemes influence collective team action and vice versa, we draw on theories of the role of shared mental models in team action. Shared mental models, as defined by Cannon-Bowers et al. [41],

> are knowledge structures held by members of a team that enable them to form accurate explanations and expectations for the task, and in turn, to coordinate their actions and adapt their behaviour to demands of the task and other team members" (p. 228).

The issue then is not so much whether developers have interpretive schemas, but rather the degree of similarity and sharing among the schemas of different developers.

Shared mental models are clearly important for team effectiveness. In a study of supply chains (another distributed work environment), Hult, Ketchen and Slater [42] found the level of shared meanings to be related to improved overall performance (specifically, reduced cycle time). On the other hand, without shared mental models, individuals from different teams or backgrounds may interpret tasks differently based on their backgrounds, making collaboration and communication difficult [43]. The tendency for individuals to interpret tasks according to their own perspectives and predefined routines is exacerbated when working in a distributed environment, with its more varied individual settings and less opportunity for informal discussion.

Research on software development in particular has identified the importance of shared mental models in the area of distributed software development [44]. Curtis et al. [9], note that, "a fundamental problem in building large systems is the development of a common understanding of the requirements and design across the project team." They go on to say that, "the transcripts of team meetings reveal the large amounts of time designers spend trying to develop a shared model of the design". In short, shared mental models are important as guides to effective individual contributions to, and coordination of the software development process.

In emphasizing the duality of structure, the structurational perspective draws our attention to how shared mental models are products of, as well as guides to, action. Walton and Hackman [45] identify an interpretive function of teams, which is to help members create a consistent social reality by developing shared mental models. The problem of developing shared mental models is likely to particularly affect FLOSS development, since FLOSS project members are distributed, have diverse backgrounds, and join in different phases of the software development process. To identify specific actions that can help to build shared mental models, we draw on the work of Brown and Duguid [46], who identify the importance of socialization, conversation and narration in building shared mental models.

First, new members joining a team need to be socialized into the team to understand how their work fits into the processes being performed or the existing code structure. An example drawn from the early stage of team development the APACHE HTTPD project illustrates the function of socialization. In this example a more senior member of the team explicitly addresses new members of the team in explaining why the code functions in a particular way (quotations from developers email or IRC chats are presented in typewriter font).

> For new members to the list, child processes would send lines such as these to the logging process with a child number attached. The child num would be used by the logging process to determine which pieces of info need to be teamed. When the logging process receives say the ETIME, it knows it is safe to log request.

Second, conversation is critical in developing shared mental models. It is difficult to build shared mental models if people do not talk to one another and use common language [47]. Meetings, social events, hallway conversations and electronic mail or conferencing are all ways in which team members can get in touch with what others are doing and thinking. Yoo and Alavi [18] found that leaders sent more messages than other team members. However, these kinds of conversations are less likely to occur spontaneously in highly distributed groups. Another example from APACHE HTTPD illustrates a conversation between a newer member and a more expert member to clarify actions and meaning of terms. The conversation between the two members and others in that particular instance leads to shared understanding of the meaning of terms and a deeper understanding of the code and the process.

Newer member:

> I think I'm missing something. What does that comment mean in this context? Doesn't static only define these variables to be "global" in this file? I don't understand how this relates to reformatting speed, whatever that is.

Expert member:

> Yes, that's what it means in this context. What it means is that I didn't want to make them globals, but decided it wasn't so bad to do that. The reformatting speed is when you use #config timefmt to change the string version of those variables, with these globals it doesn't have to call time(NULL) again or stat().

Finally, Brown and Duguid [46] stress the importance of narration. To keep shared mental models strong and viable, important events must be replayed, reanalyzed, and shared with newcomers. The history that defines who we are and how we do things around here must be continually reinforced, reinterpreted, and updated. A third example drawn from APACHE HTTPD, illustrates the function of narration. During this early stage in the team development, members of the Apache project were discussing their goals in terms of the product and process. In discussion of what members should include when modifications to the product were made, a member used a war story to illustrate a point:

> "So long as you remembered to put in the #ifdef. Sometimes, people forget. With RCS, this is not a problem. (A minor war story may be instructive, if only to let people know where I'm coming from. In the ai_httpd sources I've put up on ftp.ai.mit.edu, the nameserver cache is an option, so the code can be compiled at sites which don't do mmap(). My first cut at doing this left out an #ifdef around a line of modified code (the call to write_nameserver _cache in get_remote_host), meaning that while my modified server tested just fine, the base configuration could not be compiled after the patch. I fixed that, but this sort of human error is likely to happen again, and probably not just to me.)"

Based on the discussion above, we offer the following propositions. The two parts of proposition 1 correspond to the two links between structure and action shown in Figure 1: part a corresponds to the downward arrow showing the effect of structure on developers' actions, and part b, to the upward sloping arrow, showing the effect of developers' actions on structures.

***Proposition 1a:*** Highly developed shared mental models enable more effective contributions by FLOSS developers.

***Proposition 1b:*** Teams with practices that involve higher levels of socialization, conversation and narration will develop more highly developed shared mental models.

In a distributed team where members make diverse knowledge contributions [48], leaders may exercise their influence by means of their substantive expertise as well as through their coordinating and directing activities. An important leadership function may be supporting the development of shared mental models (consciously or unconsciously) by leading in the socialization of other team members through conversation and narration. A couple of examples from APACHE HTTPD illustrates that members who engage in building shared mental models of the code structure or coordinate tasks are often recognized as the experts in those areas. This recognition is shown in these quotations in the way that other members of the group often refer to specific team members for final decisions or to seek feedback specifically on the matter.

Example product leadership:

> Also, this would be greatly enhanced if instead of issuing a Redirect the server could respond "here's the jpg file, but you (the client) should be aware the URL this is really known as is http://host/path/mother.jpg". I didn't see any codes that matched that - **Roy?**

Example process leadership:

> Inactivity isn't healthy, so let's get something up and running, I belive that **rst** is in a position to give us a list of patches which are urgent. If there are no objections, lets have a look at this prioritised list, setup some pointers to the patches themselves, and try them out. One we get the urgent fixes out of the way, we can get back to the wish-list.

We therefore offer the following proposition linking structure, action and leadership:

***Proposition 2***: Team members who are more involved in socialization, conversation and narration will be recognized as leaders by other team members.

*B. Resources and structures of domination*

Second, the control of resources is the basis for power and thus for structures of *domination*. Resources include both allocative resources (control over things) and authoritative resources (control over people). For software development, these resources would seem to be less relevant: since the work is intellectual rather than physical and development tools are readily available, few "things" in short supply. Furthermore, most FLOSS projects are composed of volunteers and have a stated ethos of open contribution and lack of formal hierarchy, making control over people indirect at best. Nevertheless, developers do face important differences in access to expertise and in control of system source code (the primary resource) and documentation (a secondary resource), so structures of domination are still important. Structures of domination are inscribed in roles within project having differential access to code and documents. Researchers have described FLOSS projects as having a hierarchical or onion-like structure [49-52], with core developers contributing most of the code and overseeing the design and evolution of the project, with contributions from co-developers or active users. Core developers are distinguished by having write privileges on the source code. Research suggests that teams need members in all these roles to be successful [52].

Roles emerge from activities such as task division. The overall task of developing the system is divided into pieces suitable for different kinds of participants. An example drawn from the PLONE project, shows one member describing a division of labor in the production of documentation that involves several levels of contribution.

> > The normal flow is:
> > 1. Author adds documentation
> > 2. Reviewer publishes documentation
> > 3. User reads documentation, has
> > question/correction, adds comment
> > 4. Author gets email
> > 5. Author reads comment, corrects his article,
> > removes the comment
> > (and if we had events, we could send a "thank
> > you" mail here ;) (also note that author can edit
> > his content in-place after initial publication, no
> > need for another workflow process.)
> > 6. The flow starts at (3) again.

Based on the discussion above, we offer the following propositions:

***Proposition 3a:*** Clearly defined role structures with participants in all roles enable more effective contributions by FLOSS developers.

***Proposition 3b:*** Teams in which role definition functions such as task division are regularly performed will develop more clearly defined role structures.

In the early stages of APACHE HTTPD, there was not a clear identification of roles. However, the emergent behavior of members in how they contribute and facilitate to the development of the product or the process led to fellow members deferring to them in their matter of expertise. In example 1, members call on the "code experts" for feedback. In example 2, a member calls on one of the "process leaders."

Example 1 (product):

> So my question is, could set_content_type_and_parms() be removed as well? Or is the NULL default content type important somewhere? RST [*expert's initials*]?

Example 2 (process):

> RST - How do you want to distribute this? Shall I send it to you? Perhaps we need to start a module repository?

***Proposition 4:*** Team members who perform role definition functions such as task division will be recognized as leaders by fellow team members.

*C. Rules and norms and structures of legitimation*

Finally, actors' social norms and team rules embody structures of *legitimation*. The regulative function of teams, as presented by Walton and Hackman [45], describes one aspect of team functions as the creation of rules, implicit and explicit. As the team attempts to achieve its task, team interactions lead to the development of implicit and explicit rules for social or interpersonal interaction to guide team member behaviour in achieving its goals and functions. The creation and implementation of rules is a key competency for any team or organization [53]. A team or organization's ability to creatively create rules that are consistent with members' actions and represent organizational mission, values and process is critical to its effectiveness [53, 54]. For example, Fielding [4] describes the creation of decision making rules in the APACHE HTTPD project.

These developments are the result of integrating the knowledge of experts, through problem solving, political negotiation, and experiential learning [53], into the team's structure reflecting potential behavioural changes within a team over time, what March et al. [53] and Hayes and Allinson [55] refer to as learning on the team level. They also reflect what we have labeled procedural task leadership at work. Grant [48] similarly suggests that a firm (or team) creates coordination mechanisms, in the form of procedures and norms, to economize on communication, knowledge transfer and learning, thus reserving team decision making and problem solving for complex and unusual tasks.

***Proposition 5a:*** Clearer and more elaborate rules and norms enable more effective contributions by FLOSS developers.

***Proposition 5b:*** Teams with practices that involve high levels of collaborative, interactive problem solving, political negotiation, and experiential learning will develop clearer and more elaborate rules and norms.

Since development of structures is an important leadership function, team members who initiate the processes that result in the development of implicit or explicit rules are those most likely to be perceived as leaders by other members [56, 57]. As well, Barley and Tolbert [1] note that socialization frequently "involves an individual internalizing rules and interpretations of behaviour appropriate for particular settings" (p. 100). New members need to be encouraged and educated to interact with one another to develop a strong sense of "how we do things around here" (e.g., norms).

The following example shows the development of a set of explicit rules for the construction of documentation for the PLONE project. Note as well that the individual developing these rules explicit calls for them to be promulgated by a core developer to give them legitimacy, another example of recognition of leadership being tied to creation of structures.

> > I assume that we will be pushing a lot of
> > documentation in the next few weeks.
> > I think it would be very helpful if documentation
> > reviewers had a set of guidelines to follow for
> > what to accept as-is, what to edit and publish,
> > and what to reject. Things like
> >
> > o Short name format
> > o Descriptions
> > o Style/formatting of body text
> > o Version information
> > o Formatting
> > o Section organization
> > o Comments (when to add, when to remove)
> >
> > Perhaps the best thing would be to produce a
> > checklist against which submitters and
> > reviewers could gauge a piece of documentation.
> > Hopefully, this should remove some ambiguity
> > and resolve any disputes on what gets edited and
> > what gets accepted.
> >
> > I think it's important to do this sooner rather
> > than later, as we want to establish PHC as a
> > bonafide resource right from the outset. It
> > doesn't have to be long or overly detailed, but it
> > does have to be somewhat authoritative, which
> > means that Alex or someone else core should
> > produce the initial draft.

A second example shows a project leader in the PLONE project enforcing a norm by evicting from an IRC discussion channel an individual, RATATOSK, who persistently violated the norms for interactions in the project.

> [16:40] * Ratatosk () has joined #plone
> [16:41] <Ratatosk> f* hell
> [16:41] <Ratatosk> f* wankes in here
> [16:41] <Forsetim|win> Niord|dinner: well RSS can just escape it, but RSS can include anything ;)
> [16:41] * Sleipnir () Quit ("Leaving")
> [16:41] * spliter () Quit ("Leaving to have a rest")
> [16:41] * Forsetim|win goes back to looking at the dublin core spec
> [16:41] * Ratatosk hm
> [16:41] * Ratatosk Forseti is gay
> [16:42] <Niord|dinner> Ratatosk doesn't know how to use the /me command
> [16:42] * Sleipnir () has joined #plone
> [16:42] * Forsetim|win knows how to use the boot though
> [16:42] <Niord|dinner> indeed
> [16:42] <Tyr> hi Sleipnir
> [16:42] * Ratatosk hi
> [16:43] * Urd () has left #plone
> [16:43] <mehere> I'm wondering what's wrong with this code: ...
> [16:43] * Ratatosk was kicked by Niord|dinner (Ratatosk bye)

Based on this discussion, we offer a final proposition:

***Proposition 6:*** Members of a team who initiate the development of rules and norms, who implicitly or explicitly enforce rules and norms, and who socialize others in these rules and norms will be perceived as leaders by other members.

### D. Summary

Combining the discussion of the three aspects of structure described above results in the conceptual framework shown in Table 1. For each of the three aspects of structure, the table describes the embodiment of the structure as we have conceptualized it for self-organizing distributed teams, and the actions that reinforce or modify the structures.

## IV. CONCLUSION

In this paper, we presented a conceptual model and a set of propositions concerning the evolution of software development processes within distributed FLOSS development projects. Developing a theoretical framework consolidating a number of theories to understand the dynamics within a distributed team is itself a contribution to the study of distributed teams and learning within organization literature [58]. We should note that while we are particular interested in self-organizing teams in which leadership is emergent, we believe that these propositions may also apply to case in which leadership is assigned.

Of course, before it can be fully accepted, a theory must be tested against empirical evidence. We are currently testing our theory in a field study of FLOSS projects. To ground the concepts developed above, we are collecting a wide variety of evidence, including logs of ICT-supported interactions, bug reports, code changes and project documents, as well as interviews with developers. The illustrations presented in this paper are examples of these sources. These data will be analyzed primarily through content analysis, but also by creating process maps, cognitive maps and social networks.

Even in its current state of development though, the theory does have some implications for practice. The theory suggests that team leaders should be particular concerned with socialization of new members, definition of team roles and development of rules and norms. Understanding the processes of teams of independent knowledge workers working in a distributed environment is important to improve the effectiveness of distributed teams and of the traditional and non-traditional organizations within which they exist. The results of our study could serve as guidelines (in team governance, task coordination, communication practices, mentoring, etc.) to improve performance and foster innovation.

**Table 1.** Constructs for study: Embodiments of structures and actions that reinforce or modify structures.

| **Structure** | **Structural embodiment** | **Actions that create/reinforce/ modify structure** |
|---|---|---|
| Signification | Shared mental models | Socialization<br>Conversation<br>Narration<br>Task and social leadership |
| Domination | Roles with differential access to allocative and authoritative resources | Role definition<br>Role assignment<br>Socialization<br>Task and social leadership |
| Legitimation | Norms<br>Formal rules and procedures | Rule and norm creation and change<br>Problem solving<br>Political process<br>Experiential learning<br>Task and social leadership |

## V. ACKNOWLEDGEMENTS

This research was partially supported by NSF Grants 03-41475 and 04-14468. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

## VI. REFERENCES

[1] S. R. Barley and P. S. Tolbert, "Institutionalization and structuration: Studying the links between action and institution," *Organization Studies*, vol. 18, pp. 93–117, 1997.

[2] E. S. Raymond, "The cathedral and the bazaar," *First Monday*, vol. 3, 1998.

[3] P. Wayner, *Free For All*. New York: HarperCollins, 2000.

[4] R. T. Fielding, "Shared leadership in the Apache project," *Communications of the ACM*, vol. 42, pp. 42–43, 1999.

[5] M. B. Watson-Manheim, K. M. Chudoba, and K. Crowston, "Discontinuities and continuities: A new way to understand virtual work," *Information, Technology and People*, vol. 15, pp. 191–209, 2002.

[6] P. C. van Fenema, "Coordination and control of globally distributed software projects," in *Erasmus Research Institute of Management*. Rotterdam, The Netherlands: Erasmus University, 2002, pp. 572.

[7] P. S. de Souza, "Asynchronous Organizations for Multi-Algorithm Problems," in *Department of Electrical and Computer Engineering*: Carnegie-Mellon University, 1993.

[8] D. J. Armstrong and P. Cole, "Managing distance and differences in geographically distributed work groups," in *Distributed Work*, P. Hinds and S. Kiesler, Eds. Cambridge, MA: MIT Press, 2002, pp. 167–186.

[9] B. Curtis, D. Walz, and J. J. Elam, "Studying the process of software design teams," in *Proceedings of the 5th International Software Process Workshop On Experience With Software Process Models*. Kennebunkport, Maine, United States, 1990, pp. 52–53.

[10] J. A. Espinosa, R. E. Kraut, J. F. Lerch, S. A. Slaughter, J. D. Herbsleb, and A. Mockus, "Shared mental models and coordination in large-scale, distributed software development," presented at Twenty-Second ICIS, New Orleans, LA, 2001.

[11] D. Bandow, "Geographically distributed work groups and IT: A case study of working relationships and IS professionals," in *Proceedings of the SIGCPR Conference*, 1997, pp. 87–92.

[12] G. Mark, "Conventions for coordinating electronic distributed work: A longitudinal study of groupware use," in *Distributed Work*, P. Hinds and S. Kiesler, Eds. Cambridge, MA: MIT Press, 2002, pp. 259–282.

[13] F. Bélanger and R. Collins, "Distributed Work Arrangements: A Research Framework," *The Information Society*, vol. 14, pp. 137–152, 1998.

[14] E. Carmel and R. Agarwal, "Tactical approaches for alleviating distance in global software development," *IEEE Software*, pp. 22–29, 2001.

[15] S. L. Jarvenpaa and D. E. Leidner, "Communication and trust in global virtual teams," *Organization Science*, vol. 10, pp. 791–815, 1999.

[16] R. E. Kraut, C. Steinfield, A. P. Chan, B. Butler, and A. Hoag, "Coordination and virtualization: The role of electronic networks and personal relationships," *Organization Science*, vol. 10, pp. 722–740, 1999.

[17] W. F. Cascio and S. Shurygailo, "E-leadership and virtual teams," *Organizational Dynamics*, vol. 31, pp. 363–376, 2003.

[18] Y. Yoo and M. Alavi, "Emergent leadership in virtual teams: what do emergent leaders do?" *Information and Organization*, vol. 14, pp. 27–58, 2004.
[19] I. Zigurs, "Leadership in virtual teams: Oxymoron or opportunity?" *Organizational Dynamics*, vol. 31, pp. 339-351, 2003.
[20] C. Pavitt, "Small Group Communication: A Theoretical Approach (3rd Ed)," vol. 2004, 1998.
[21] R. F. Bales, "The equilibrium problem in small groups," in *Working papers in the theory of action*, T. Parsons, R. F. Bales, and E. A. Shils, Eds. Glencoe, IL: Free Press, 1953, pp. 111–161.
[22] D. G. Ancona and D. F. Caldwell, "Beyond task and maintenance: Defining external functions in groups.," *Group and Organization Studies*, vol. 13, pp. 468–494, 1988.
[23] S. L. Jarvenpaa, K. Knoll, and D. E. Leidner, "Is Anybody Out There? Antecedents of Trust in Global Virtual Teams," *Journal of Information Systems*, vol. 14, pp. 29–64, 1998.
[24] T. R. Kayworth and D. E. Leidner, "Leadership effectiveness in global virtual teams," *Journal of Management Information Systems*, vol. 18, pp. 7–40, 2002.
[25] K. L. Tyran, C. K. Tyran, and M. Shepherd, "Exploring emergent leadership in virtual teams," in *Virtual Teams That Work: Creating Conditions for Virtual Team Effectiveness*, C. B. Gibbon and S. G. Cohen, Eds. San Francisco: Jossey-Bass, 2003, pp. 183–195.
[26] Y. Yoo and M. Alavi, "Electronic Mail Usage Pattern of Emergent Leaders in Distributed Teams." Cleveland, OH: Weatherhead School of Management, Case Western Reserve University, 2002.
[27] J. R. Hackman and R. E. Walton, "Leading groups in organizations," in *Designing Effective Work Groups*, P. S. Goodman and Associates, Eds. San Francisco, CA: Jossey-Bass, 1986, pp. 72–116.
[28] A. Giddens, *The Constitution of Society: Outline of the Theory of Structuration*. Berkeley: California, 1984.
[29] S. R. Barley, "Technology as an occasion for structuring: Evidence from the observation of CT scanners and the social order of radiology departments," *Administrative Sciences Quarterly*, vol. 31, pp. 78–109, 1986.
[30] G. DeSanctis and B. M. Jackson, "Coordination of information technology management: Team-based structures and computer-based communication systems," *Journal of Management Information Systems*, vol. 10, pp. 85, 1994.
[31] M. Newman and D. Robey, "A social process model of user-analyst relationships," *MIS Quarterly*, vol. 16, pp. 249–266, 1992.
[32] W. J. Orlikowski, "The duality of technology: Rethinking the concept of technology in organizations," *Organization Science*, vol. 3, pp. 398–427, 1992.
[33] G. Walsham, *Interpreting Information Systems in Organizations*. Chichester: John-Wiley, 1993.
[34] S. Sarker, F. Lau, and S. Sahay, "Using an adapted grounded theory approach for inductive theory building about virtual team development," *DATA BASE for Advances in Information Systems*, vol. 32, pp. 38–56, 2001.
[35] W. J. Orlikowski, "Using technology and constituting structures: A practice lens for studying technology in organizations," *Organization Science*, vol. 11, pp. 404–428, 2000.
[36] E. W. Stein and B. Vandenbosch, "Organizational learning during advanced system development: Opportunities and obstacles," *Journal of Management Information Systems*, vol. 13, pp. 115–136, 1996.
[37] M. E. Kondrat, "Actor-centered social work: Revisioning "person-in-environment" through a critical theory lens," *Social Work*, vol. 47, pp. 435–448, 2002.
[38] Y. Sarason, "A model of organizational transformation: The incorporation of organizational identity into a structuration theory framework," *Academy of Management Journal*, pp. 47–51, 1995.
[39] D. Gregory, "Presences and absences: Time-space relations and structuration theory," in *Social Theory of Modern Societies: Anthony Giddens and His Critics*. Cambridge: Cambridge University Press, 1989.
[40] P. Cassell, "The Giddens Reader." Stanford, CA: Stanford University Press, 1993.
[41] J. A. Cannon-Bowers and E. Salas, "Shared mental models in expert decision making," in *Individual and Group Decision Making*, N. J. Castellan, Ed. Hillsdale, NJ: Lawrence Erlbaum Associates, 1993, pp. 221-246.
[42] G. T. M. Hult, D. J. Ketchen, Jr., and S. F. Slater, "Information processing, knowledge development and strategic supply chain performance," *Academy of Management Journal*, vol. 47, pp. 241–253, 2004.
[43] D. Dougherty, "Interpretive barriers to successful product innovation in large firms," *Organization Science*, vol. 3, pp. 179–202, 1992.
[44] L. L. Levesque, J. M. Wilson, and D. R. Wholey, "Cognitive divergence and shared mental models in software development project teams," *Journal of Organization Behavior*, vol. 22, pp. 135–144, 2001.
[45] R. E. Walton and J. R. Hackman, "Groups under contrasting management strategies," in *Designing Effective Work Groups*, P. S. Goodman and Assoc, Eds. San Francisco, CA: Jossey-Bass, 1986, pp. 168–201.
[46] J. S. Brown and P. Duguid, "Organizational learning and communities-of-practice: Toward a unified view of working, learning, and innovation," *Organization Science*, vol. 2, pp. 40–57, 1991.
[47] B. A. Bechky, "Sharing meaning across occupational communities: The transformation of understanding on a production floor," *Organization Science*, vol. 14, pp. 312–330, 2003.
[48] R. M. Grant, "Toward a knowledge-based theory of the firm," *Strategic Management Journal*, vol. 17, pp. 109–122, 1996.
[49] A. Cox, "Cathedrals, Bazaars and the Town Council," 1998.
[50] C. Gacek and B. Arief, "The many meanings of Open Source," *IEEE Software*, vol. 21, pp. 34–40, 2004.
[51] G. K. Lee and R. E. Cole, "From a firm-based to a community-based model of knowledge creation: The case of Linux kernel development," *Organization Science*, vol. 14, pp. 633–649, 2003.
[52] J. Y. Moon and L. Sproull, "Essence of distributed work: The case of Linux kernel," *First Monday*, vol. 5, 2000.
[53] J. G. March, M. Schulz, and X. Zhou, *The Dynamics of Rules: Change in Written Organizational Codes*. Stanford, CA: Stanford University Press, 2000.
[54] C. Argyris and D. A. Schön, *Organizational Learning*. London: Addison-Wesley, 1978.
[55] J. Hayes and C. W. Allinson, "Cognitive style and the theory and practice of individual and collective learning in organizations," *Human Relations*, vol. 51, pp. 847-871, 1998.
[56] D. C. Baker, "A qualitative and quantitative analysis of verbal style and the elimination of potential leaders in small groups.," *Communication Quarterly*, vol. 38, pp. 13–26, 1990.
[57] S. M. Ketrow, "Communication role specializations and perceptions of leadership," *Small Group Research*, vol. 22, pp. 492–514, 1991.
[58] D. Robey, H. M. Khoo, and C. Powers, "Situated-learning in cross-functional virtual teams," *IEEE Transactions on Professional Communication*, pp. 51–66, 2000.