Chapter 5

Analysis and discussion

We have seen how Novell/SUSE has opened up their development process to facilitate a developer-community external to the company. In some respects they are extending their own organization, but at the same time they are keeping externals at arms-length by not granting them authority to commit code to the system. Although these external developers are not part of the Novell organization, they are nevertheless united in a collaborative network that is linked to this organization.

There are several questions that arise concerning the topic of firm-sponsored communities, which I will make an effort to answer in this chapter. The first set of questions are related to my first research topic: *What is the (theoretical) distinction between the sponsor firm and the sponsored community?* To answer this question, we must try to explain the organizational phenomenon we are witnessing in this case. Can we understand Novell and the openSUSE community as one and the same organization, or as two separate ones? Is it even possible to regard the openSUSE community as an independent entity? In the following I will argue they should be regarded as separate autopoietic systems with different properties. The second issue that arises is the question of how Novell and the openSUSE community are connected? Since there are several issues that may cause difficulties in the collaboration, I will discuss which factors it is that makes this co-existence possible. This is more precisely formulated in my second research question: *What is the nature of the relationship between the organizational and the external interactive system, and how are the two systems bound together?* The third set of questions concern the future development of the openSUSE project. How will the community evolve in relation to Novell? What are the possible scenarios? As this type of community only has existed in a limited amount of time, little empirical

experiences exist to provide answers. The research on the area is also limited (O'Mahony, 2007a; J. West & O'Mahony, 2005). My hope is therefore that the theoretical framework used to address the previous questions may shed some light on alternative future scenarios. While addressing my two first research questions, I will simultaneously be pursuing my last research question as well: *How may the case of Novell strengthen our understanding of the theory of autopoietic systems and the theory of boundary objects, separately and in combination?* In the next chapter I will try to sum up some of the most important contributions this case has provided to these theories. First, we will start off by making sense of Novell and the openSUSE project.

Theorizing Novell and the openSUSE project

Novell initiated the openSUSE project in august 2005, and at the time of writing it has existed between 2 and 3 years. In regards to the original road map that was laid out for the project, the launch happened prematurely resulting in a community project that contained the bare minimum of elements necessary for an open source community. The community was bootstrapped upon the existing beta-testers already affiliated with SUSE Linux, and the media-attention following the announcement of the project drew more interested users to the site. The openSUSE community has evolved and grown substantially since its birth, primarily in numbers of users and developers affiliated with the project. The openSUSE Build Service represents the largest technical and organizational innovation within the community, and has shown a strong and interesting development. The openSUSE project has matured to the point where it now includes reliable development tools, established communication channels, a sustainable amount of external developers and contributors, and a large user-base setting expectations for the product. Some representative governance structures such as the openSUSE board have recently been established, and the GPL licensing scheme (that has been present from the start) creates a platform of trust upon which the voluntary community of contributors may continue to grow. In terms of product development, the community-building has started to yield some results, especially in the area of testing, bug-reporting and translation. A fairly little amount of code is contributed by the external community, compared to internal contributions, although the last year has provided examples of product-innovation born and bread in the external community. Novell have however made an effort to create a transparent development process, where most communication is public and contributions are appreciated.

Defining systems

Novell-engineers and openSUSE managers identify themselves as community members, and are acknowledged as such by external contributors. Managers of the openSUSE community are making an effort of erasing the distinction between the company and the community, as this manager illustrates: “When we have internal meetings and we say “the community”, we try to

avoid that or we try not to separate us from the community. Because we are also part of the community." (interview #10). What are really the boundaries between Novell and the external community? Is it possible to claim that they are erased? Drawing upon autopoietic organization theory, I would argue that the latter is unlikely with the current situation.

In Luhmann's terms, Novell represents an organizational system. In his view such a system consists of a network of *decisions* – a specific form of communication. All social systems are autopoietic, meaning that all communications are enabled by previous communications, and again make future communications possible. This idea of recursivity creates a link between process and structure (Bakken & Hernes, 2003), and explains how an organization shapes and reshapes itself over time. Novell has undergone a large transition, and decisions have created several changes in the organization. The decision of pursuing a Linux operating system led to the purchase of the SUSE Linux and the Ximian company, which again was a precondition for establishing the openSUSE community. Decisions within the system also involve communications at a more detailed level, such as decisions concerning the next feature to be implemented in a component of the software product, or even the daily commit of a piece of code into the code base through Autobuild. An illustration of Novell as a decision-network is provided in figure 11 below. Since it is and organizational system, it is represented as a square in the figure. Membership in the system is linked to positions that take part in this organizational decision-network. So where does the openSUSE community fit in relation to this system? When an organization is defined as a decision-network, it becomes clear who is *not* part of the system: those who can not make decisions within it. Luhmann's theory therefore distinguishes Novell as an organization *apart* from the voluntary openSUSE community, which must thus be defined as something separate that exists in Novell's external environment.

Figure 11. The Novell organizational system and the interactive openSUSE system

Linux product-development in Novell does however happen in collaboration with external members of the openSUSE community, and these *do* have a role in the work on the product. The openSUSE community is a heterogeneous network of people surrounding the development of the

openSUSE distribution, who interact closely with the organizational system. There is a strong internal network of communications within this community, facilitated by the electronic communication channels. As such, this chain of communicative events in the openSUSE community is a social system *in its own right*. The openSUSE network might not have the qualities of an organizational system, primarily since there are no decisions or positions in the system (all decisions concerning the openSUSE project are made by Novell). However, the active openSUSE community may be regarded as an *interactive* system, following its own autopoietic development. For this reason, it is illustrated as an elliptical to differentiate it from the organizational system. Although the people involved with communications in this system are not physically present with each other, Michéle Morner (2003) successfully argues that open source projects may act as interactive systems, since communications flow in public channels across the Internet: "In principle, they have the possibility to follow each and every communication because of its documentation in mailing lists or newsgroups" (2003, p. 264). Due to the fact that communications are accessible to everyone, modularized by topics and are reproducible, she argues further that the stored communication contributes to the *stabilization* of the system. Drawing upon Luhmann, she shows how these features contribute to the system's *communication connectivity* and *systemic memory* (ibid). This is normally an untypical feature for interactive systems, but explains how a community of voluntary individuals with no formal organization[16] may *persist over time*.

One might want to note that the interactive openSUSE system is not identical with the openSUSE community (whoever that may be). With the framework of autopoietic system theory, we are provided with limits to the parts of the community that are active in the development of the distribution. This helps us solve the issue of identifying the "contributor-community" stated in the previous chapter. Since the system is only based on *communicative events*, the large numbers of users whom otherwise do not communicate with the system are defined as external and part of its environment. Downloading the distribution for personal use does not count as a communication, as only one person is involved. According to Luhmann, even the active contributors – as in the people – are not part of the system. However, we can say that communicative events that constitute the system are *linked to* the senders and receivers to the system (as in the people). Thus, we can continue and say that the interactive openSUSE system is equivalent to the active individuals in the openSUSE community, by means of the communications that are made by these participants. Since people are not part of the system, it is

[16] One might argue that Novell, as a sponsor of the community, provides the necessary formal organization to the project. This is however no more true than for any other autonomous open source project, as Novell has no "jurisdiction" outside its own boundaries to, for example, assign people to tasks. All the organizing within the openSUSE community is done in a voluntary manner; community members must accept Novell's suggestions in the same way as any other independent individual.

possible to say that "the community" may hold any desirable definition (i.e. "anyone who is a user of openSUSE and has an interest in its development"), irrespective of the interactive system involved with product development.

System connectivity

The organizational and the interactive system are both active in the development of the openSUSE distribution, and thereby the enterprise Linux distribution as well (although the interactive system relates to this more indirectly). These two systems are balanced in collaboration on the openSUSE product, somehow. It is clear that these systems are empirically intertwined, but does Luhmann's theory support such an understanding? I argue it does, as it is not uncommon that various types of social systems overlap (Jönhill, 2003). For example, organizations are crucial for binding different functional systems (i.e. politics and economy) together, as they may inhabit several societal systems simultaneously. Furthermore, Luhmann's insistence on leaving the individuals outside the systems, allow them to move about and take part in communications within a several systems. It is not as if participation within one system keeps you out of others. The theory can then explain how an employee in Novell can take part in the openSUSE interactive system and other open source projects, while still making communications within the Novell organizational system.

Autopoietic systems theory does not exclude the possibility of participation within several systems, or tight collaborative relations between them. To the best of my knowledge, it may however seem that the theory suffers from a lack of explaining *how* systems are linked together. Connectivity between autopoietic systems is typically referred to as "structural coupling" (Brans & Rossbach, 1997), but Luhmann's literature seems to exclusively focus on connectivity between societal subsystems:

> "Structural coupling is a relationship between systems with each of them belonging to the other's environment. It involves a system making its own complexity available for the constitution of another system and vice versa. The systems involved in the relationship can take the existence of the other systems for granted and thus concentrate on their own tasks. All the major functional subsystems of society – politics, law, economy, and science – are structurally coupled in this way" (Brans & Rossbach, 1997, p. 426).

Another dimension of structural coupling Luhmann has investigated is the connection between systems in different domains, such as the physiological, psychological and social systems. (Mingers, 2003). However, I have failed to locate descriptions of structural coupling between social systems themselves, at a micro and meso-level. At this level, the perspective is usually that of *one* autopoietic system in relation to its environment. The Novell and openSUSE system are closely intertwined, but how can we explain this connectivity? In the following, I will argue that other theoretical elements such as boundary objects could be a useful supplement Luhmann's

theory, by offering more explanation to how systems can be bonded together. Pulling in the theory of boundary objects also adds more normative explanatory power to the analysis, which we are deprived of with pure autopoietic theory. I would argue that discussing collaboration between actors in the social world of open source - or any other context for that matter - with disregard to the interests that the various stakeholders represent is problematic, as this would ignore a very important aspect of the organizational reality. Luhmann's theory does not deny the existence of rationality within an organization, but can not explain what this rationality consists of and thus which direction the organization is headed:

> "We do not preclude that organizations avail themselves of certain ends ("Ziele"), and rules in the acquisition of means to achieve ends (e.g. views concerning costs), in order to identify themselves. But we leave this question to the clarification of singular cases (…) and maintain only as a basis that an organization exists only as long as its autopoiesis continues and decisions are reproduced from decisions" (Luhmann, 1988, p. 34).

Before going deeper into this analysis, I will first have a closer look at the nature of the boundaries between the two systems and what enables them to interact with each other.

The organizational membrane

It may be too simple to say that the organizational and the interactive system are merely overlapping. If we compare Novell and the SUSE company from five years back and today, something has happened with the organization itself, or rather, the boundaries of the organization. Earlier, the development process was closed off and there was a clear distinction between the world inside SUSE and the world on the outside. Access to the source code was only given after the software was fully developed and released. This is similar to the old Novell organization as well. The organization closed itself off as many proprietary companies do. John Mingers illustrates how organizations erect these barriers, in general terms (2003, p. 115):

> "The organization is in a continual process of re-creation of itself anew as communications generate further communications and thereby internal structure. It is the organization that defines its boundaries through its communications and thereby closes itself off, generating a quite **impermeable barrier** for the environment" [emphasis added].

With the openSUSE project, Novell started to offer transparency into the development process, even allowing external influence on the product. Today internal developers communicate extensively with external contributors, and may even receive code contributions from the outside. The boundaries of the organization have changed, and are no longer *impermeable*. What is the nature of the current boundaries?

Permeability is an expression borrowed from chemistry, and is used to describe the ability of a substance to let a liquid or gas pass through. In biology, for example, it is common to discuss the

permeability of cells' plasma-membranes. Within organization theory there are few words that describe the composites of boundaries, and it may therefore be fruitful to borrow some vocabulary from this metaphor. The cell's plasma-membrane is described as *semi-permeable*, as proteins located in the cell's membrane allow the transportation of certain ions such as Natrium and Calium in and out of the cell, while others substances may be rejected. The plasma-membrane controls the traffic selectively, and is therefore also said to exhibit *selective permeability*. A special form of protein, named aquaporins[17], enable the continuous and free flow of water molecules through the membrane regardless of this selectivity.

Without drawing this analogy too far, we could say that the organizational boundaries of Novell and SUSE have evolved from being *impermeable*, when product development was done in a largely proprietary manner, to becoming *semi-permeable*, as development has become more transparent and communication concerning product development continuously flows in and out of the company. Influences on product development are also admitted on occasion through these boundaries, but only on a selective basis – Novell has sovereignty over all decisions in the project. Novell are therefore also exhibiting *selective permeability* towards its environment. The communications that flow through Novell's boundaries and the activity that is centered upon them are facilitated by some important elements, namely the boundary objects and communication channels that are located at the intersection of the Novell organizational system and the interactive openSUSE system. We will now move on to see how these operate.

Binding systems

By creating the openSUSE project, Novell initiated the emergence of an interactive system that now surrounds the development of the openSUSE distribution, but which is held at arms-length by a semi-permeable boundary from the organization controlling it. How do these two systems interact? What binds them together? I argue there are three elements that ensure a tight coupling between the two systems. First, several developments tools and models serve as important *boundary objects* between the two systems, enabling joint development on a common product. Secondly, *shared communication channels* are vital in creating transparency and providing access through the boundaries of the systems. Thirdly, the *marginal people* whom have roles in both systems are crucial for balancing the needs of both of them. In the following, I will address these three aspects in closer detail, but my analytic emphasis will remain on the first and third of the three. This does not mean that communication channels are not equally important in bridging the systems, but they do not demand the same amount of explanation. The following discussion will show how Luhmann's theory may be supplemented and strengthened with the theory of

[17] The discovery of aquaporins happened recently, and in 2003 Peter Agre received the Nobel prize in chemistry for its discovery.

boundary objects. Moreover, it will show how the theory of boundary objects not only connects individuals and groups, but also *social systems*.

Boundary objects

I have previously discussed the importance of boundary objects in collaborative work, based on two crucial properties: their ability to contain and create coherence among separate interests, and their ability to carry common knowledge and identity at the boundaries between social groups in collaboration. There are several such objects that play an important role in the development of the openSUSE distribution. We will immediately start off by looking at one such example, before returning to a discussion of their role in the collaboration.

A boundary object example

In all software development work, bug-tracking tools are crucial for organizing quality assurance processes. Bugzilla is the name of the tool used in openSUSE, and is itself an open source software tool that is widely used and familiar to most open source developers. Bugzilla keeps track of bug-reports, which normally represent notifications of an error in the software. Bug-reports are created retrospectively, typically after a user has tested a software product and detected an error somewhere. The user can then create a bug-report describing the problem, whereby a developer is assigned to finding a solution. A discussion may then go on in the bug-report between the user, the developer and other users that have an interest or opinion in the matter. Discussion entries may include attachments, such as screen shots of the problem or a patch of code that might fix the problem. A generic model of a bug-report is provided in figure 12.

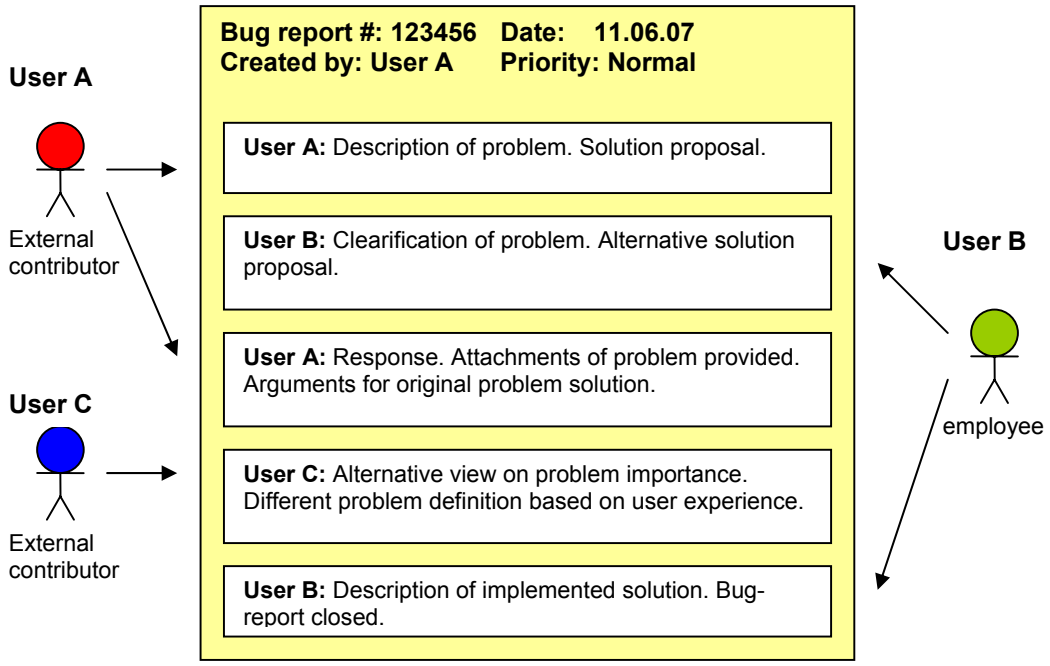


Figure 12. Model of a bug-report.

Although most bug-reports are rather non-controversial and straight-forward descriptions of a problem and its possible solutions, there may be different interests present and discussions can arise. Let us imagine that the following interests were present in the model above:

User A: Primary interest: get annoying problem solved. Secondary interest: provide a constructive contribution, and gain recognition for having identified something important.
User B: Employee. Primary interest to solve problem as quick as possible with least amount of time and cost.
User C: Interested in exposing himself as a competent user/developer. Secondary interest to solve problem.

The example is strictly hypothetical[18], but shows how the boundary object may contain multiple interests. Everyone wants the problem solved, but have other goals as well. The object is thus able to support simultaneous translations (Callon, 1986), as it does not fall apart even if there should be conflict (translations are attempts at creating an alliance by rallying the others to your problem definition). Defining the problem and suggesting a solution is a process of creating an obligatory passage point; the necessary path to take to achieve the common goal. In the case above, there might be three possible passage points; one for each user. Who will succeed in rallying the others

[18] I could have used a real example from Bugzilla to illustrate what a bug-report looks like and how it contains communication, but it would not be able to reveal the interests behind each statement. For that, a deeper analysis and empirical study would be needed. Therefore, a hypothical example will have to suffice to illustrate the point.

will depend on their arguments, strategies and positions. The original problem owner, user A, may use screenshots as interessement devices to enrole others into understanding how this problem must be solved. User B, on the other hand, is the user with the authority to actually commit a code solution to the code base, and thus holds the upper hand. If however user A or C provides a probable solution that saves him work-effort, he will be likely to become enroled and accept it.

While the discussion concerning the problem and its solution takes place *within* the bug-report, talk about the problem may still happen *outside* the object itself. Once placed in such a report, a problem in the software becomes an object that can be categorized, referred to, assigned to people, create discussion and trigger decisions. Below are two extracts from public openSUSE project meetings (organized every second week). Both employees and contributors participate, and one of the objectives in these meetings is to follow up work on bug-reports that concern non-technical issues. The first extract is an example of this process.

```
18:09 @<moderator>     Bug #343444
18:10 @<moderator>     contributor1: what about that one? :)
18:10 <contributor1>   never heard about that
18:10 @<moderator>     well read your emails dude its assigned to you
18:11 @<moderator>     caught
18:11 <contributor1>   yes.
18:11 <contributor1>   hmm
18:11 <contributor1>   ok, Asche on my haupt
18:11 <contributor1>   sorry, not yet on my radar
18:11 @<moderator>     okay
18:11 @<moderator>     no change then
```

Extract #1 from IRC project meeting.

This extract starts with reference to a specific bug-report that (#343444), and the moderator asks the contributor that is assigned to solve the problem about its status. Having forgotten all about it, the contributor admits he has not been able to do anything about it yet, and is "caught". He has, unfortunately, no change to report. This trivial extract illustrates how bug-reports are used to define and frame a problem that may be quite complex into an object that is easy for everyone to relate to. They contain shared knowledge and have a common identity, and are equally instrumental for contributors and employees alike in organizing their work. This second extract also includes a short discussion concerning the priority of a specific bug-report (my emphasis):

```
13:18 <@moderator>     Bug #293726
13:18 <bugbot>         openSUSE bug 293726 in openSUSE.org "Creation of Babel
                       wiki"
13:19 <@moderator>     that one also desperately needs closing
13:19 <employee2>      wiki update is more important for me
13:19 <employee2>      after that we can look into this
13:20 <@moderator>     c'mon. just be serious. its rotting since 6 months now
13:20 <@moderator>     there will always be something more important
13:20 <employee2>      it would be a nice to have, but we are not dieing
                       without it
```

```
13:21 <employee2>      and since john mentioned some problems with the setup, i
                       will not do a thing before the update ... new software
                       new problems ... hopefully
13:21 <@moderator>     okay
```

Extract #2 from IRC project meeting.

When the moderator says that the bug-report "needs closing", it means that the problem somehow needs to get solved so that the bug-report no longer is active. He also argues that it has been active for 6 months and needs to be solved soon. The employee however finds other tasks more important, and successfully argues that the problem should be postponed until a software upgrade has been done. This discussion only concerns the priority and time-frame for solving the problem. What the problem *is* and how it should be *handled* is captured within the bug-report itself, as shown previously. This extract shows how a software problem becomes a task. All the active bug-reports put together is therefore also understood a grand *todo-list*, perhaps more so by employees than external contributors. Since the employees are the ones officially responsible for dealing with the bug-reports on "their area" of work, the bug-reports can have another meaning for them (in addition to serving as collaborative objects and stages for conflicts of interest): Several employees I spoke with mentioned that one of the only ways of receiving feedback on their work was through Bugzilla. If there were few bug-reports on your project after a software release, you have done a good job:

> [Myself:] "In what ways do you receive feedback on what you do?"
> [Packager:] "Ha! Usually, never! But there is an indirect way: Bugzilla. If you have a lot of bugzilla's after release was done, then you know your work wasn't good. But usually, I hope I do my best and I don't get really much feedback" (interview #16).

Bugzilla is a collection of bug-reports, and includes many thousand entries. As a boundary object, Bugzilla can be described as a *repository* of objects (Star & Griesmer, 1989), and represents one of the important characteristics of the openSUSE project which distinguishes the present open development from the previous closed process in SUSE. When Bugzilla was "opened up", Novell managed to create a powerful boundary object that has proven crucial for binding the interactive openSUSE system with the organizational decision-network of Novell.

Other objects

In addition to Bugzilla, there are other examples of boundary objects found in the collaboration between Novell and the openSUSE community, including the openSUSE Build Service (OBS) and the Factory code base. The OBS is a repository of software projects, where each project may involve a number of developers ranging from one to several hundred. For each software project, the OBS includes supportive infrastructure for project management, communication and discussion. As a whole, the OBS is a remarkable boundary object. It holds room for unlimited

interests and translations. If a dispute breaks out in a software project, one of the parties can just fork the code, start their own software project in the OBS and rally their own developers. The OBS is a tool for software development, a showroom for projects and an infrastructure for alliance-building.

While the example of Bugzilla and the OBS so far has shown us how boundary objects manage to keep people together in collaboration - despite any boundaries or conflicts of interest - the Factory code base, on the other hand, displays another quality of boundary objects; namely that boundary objects are able to *connect and separate* people, groups and systems *simultaneously*. The Factory code base represents the absolute latest development of the distribution, and is therefore an object that all developers relate to. However, Factory only contains the official or accepted development projects; it does not contain all the various private development initiatives found in the OBS. There is a clear distinction between who may alter or impact the content of the Factory code base. Only employees have direct access to commit code updates that are eventually shuttled into Factory. The Factory object therefore illuminates the boundaries between the Novell organizational system and the interactive openSUSE system, while simultaneously binding them together.

As a contrast to the mentioned boundary objects, we can look at other collaborative objects which can *not* be considered boundary objects. For example, 'Autobuild' as a model of development and technical artifact is without doubt an important collaborative object, but this object remains within the Novell organizational system. It is surely an important boundary object between groups *within* Novell, but at this level of analysis it must be regarded as an internal naturalized object (different departments and teams within Novell is not our focus). As a naturalized object, having knowledge and familiarity with the object is an indicator of membership in the group (Bowker & Star, 1999). This is quite true in Novell's case: You will not get very far with your work in Novell's Linux R&D without any idea of what "Autobuild" means, nor does anyone outside Novell's Linux R&D have any clue of its meaning (there is in fact some confusion about this question internally as well). Returning our focus to the relationship between the Novell organizational system and the openSUSE interactive system, we may say that the boundary objects, put together, provide a framework that holds all the actors together - as individuals, groups and systems, while simultaneously supporting the qualities that separate the same people.

In sum, we have identified a few objects situated at the center of the Novell and openSUSE collaboration, and illustrated how they contain qualities as boundary objects. At this point I would like to start a theoretical discussion on the *role* of boundary objects, spurred by an observation made in the case of Novell and the openSUSE project: Neither the mentioned objects nor the theory of boundary objects itself can explain *why* the actors come together to

collaborate in the first place. What is the normative foundation for their joint venture? Star's theory would have us look other places for answers to this question, perhaps diverting us to economic business models and the open source literature's egoistic, altruistic, intrinsic and extrinsic incentives. I however believe that an answer to this question may exist within the boundary objects themselves. To illustrate: the most powerful boundary object of all in the collaboration in Novell is not one of the previously mentioned objects, but the end-goal of the collaboration itself: *the openSUSE distribution*. This is what all the collaboration is about; it is where all stakeholders hold their primary interests. It is also what drives all the actors together to collaborate. Before moving on with this discussion, it could be wise to sum up the simultaneous claims I am about to make: First, I am saying that the openSUSE distribution holds a quality as a boundary object that is different from other boundary objects such as Bugzilla, the OBS and Factory; Secondly, this quality is the openSUSE distribution's ability to create a normative motivation for collaboration among the actors; Finally, the two previous claims imply that a discussion and extension of Star's theory of boundary objects is desperately needed. Unfortunately, it is hard to explain everything at once, so I will have to ask the reader to be patient and bear with me while we walk through this line of thought. There are several questions that need to be answered (preferably at the same time), which I will address in the following order: First, how can the openSUSE distribution be a boundary object in the first place? Is it not just the product of collaboration? The second question I will pursue is this: What would the motivating aspect found in the openSUSE distribution mean for our understanding of boundary objects in general? Thereafter I will show how this claim is supported by some empirical findings, before I lastly look at what theoretical foundation we may find that may explain what I have witnessed in the case of Novell and the openSUSE project.

The openSUSE object

The openSUSE distribution is the product and the end result of the Linux development in Novell. I also find that it is an important boundary object situated at the intersection of all groups involved. It is reasonable to question whether the product actually can be a boundary object. However, I do not claim it is the distribution *as such* that should be considered the boundary object, as in the DVD with the running software on it. Rather, it is the *model* of what the distribution is and should be, as it is perceived among all collaborative participants. It is the idea of the product, and the meaning it holds to everyone involved. It is the word that is used whenever developers talk about "working on openSUSE", or the meaning they put into this specific software project when they compare it to other operating systems and software projects. In Star and Griesmer's terms, the openSUSE object represents an *ideal type* of boundary object: "It is abstracted from all domains, and can therefore be general enough to be applied in many contexts" (ibid, p.410).

Showing how the openSUSE model has characteristics of a boundary object is not very difficult. It is definitely a shared object between all stakeholders in the collaboration, and has a *mutual identity* among them all. Whether you are talking to a veteran Novell-employee from the "old" company, one of the executives at the headquarters, an openSUSE community developer or an un-experienced openSUSE user, they will most likely share a general understanding of what openSUSE is, as exemplified in figure 13 below. The figure contains a few basic keywords I have encountered in conversations with developers concerning the identity of the openSUSE distribution (but it is by no means any official description):

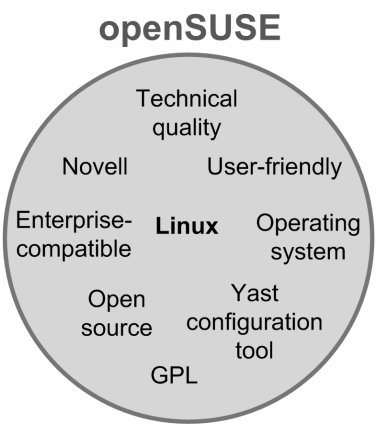


Figure 13. Keywords in the openSUSE object

Moving on, the object also contains *common knowledge* available to developers in all domains. This knowledge is directly linked to the physical distribution, as it is embedded into the software code. However, a developer does not have to look into the code base to know what the main packages in the openSUSE software architecture are, or which features and qualities it includes. The openSUSE object also holds knowledge about its historical development, which is important in forming its identity.

While the openSUSE object is able to hold many common qualities, it is even more adept at handling diversity and simultaneous translations. Consider, for example, the following statements. The purpose of the openSUSE distribution is to be:
- "a testbed for commercial products"
- "the best open distribution"
- "a development model for Novell"

These are just a few statements I have encountered in my conversations and observations with various informants. It would be possible to list a large number of interests related to the openSUSE distribution. The intriguing feature of the openSUSE object is its ability to handle many different interests such as these at the same time – they do not exclude each other! For

example, the object can support the interests of using the openSUSE distribution to appropriate returns in the software market, support proprietary enterprise solutions running on top of it *and* at the same time advocate principles of free software distribution. The object does not necessarily *harmonize* the interests; proponents of various views may be in hard conflict with each other. However, the object is able to make these interests *compatible* with one another. Note that there is a difference between which interest are present in the openSUSE object and those which may be *physically visible,* since the aspects and interests that are enacted *in practice* will depend on who comes out best with the multiple and ongoing translations surrounding the object. For example, we may recall that Novell's first release of the Linux product was labeled "Novell Linux Desktop". After this release, former SUSE employees in Novell and existing SUSE users where successful in establishing the "SUSE" brand as an obligatory passage point for the future development of the Linux product, which has lived on in the open***SUSE*** model. Figure 14 below is an illustration of how two interests are present in the openSUSE object, drawing respectively upon the normative structure of the commercial and the open source software model[19]:

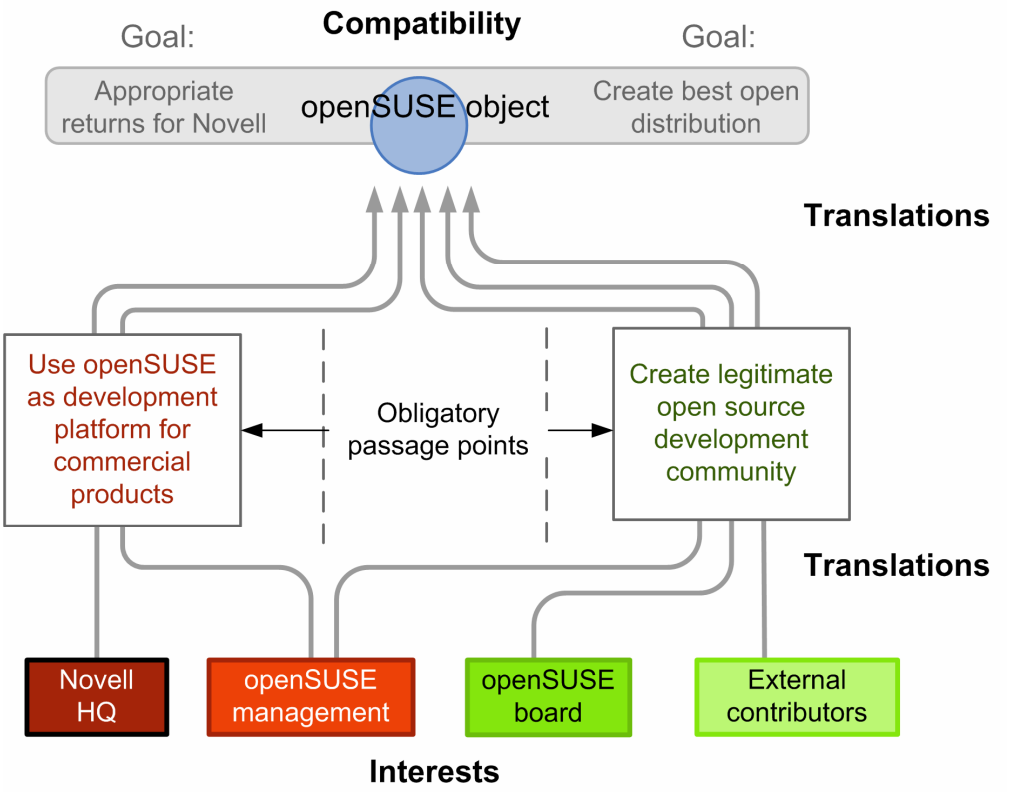


Figure 14. Illustration of translations in the openSUSE object. Model derived from Star and Griesmer (1989).

[19] Note that the model above is merely an illustration of how the openSUSE object may support n-way translations, and should not to be taken too literally. Other interests than the two illustrated surely also exist. The alliances that are drawn are also only used as examples, and may not represent the full picture.

As we have seen empirically, the obligatory passage points that are established by advocates of the interests present in the model above are able to exist simultaneously. For many developers that "choose" the path on the right, the general sentiment is that the left side is a necessary path to take *as well,* and does not interfere with the work of creating a strong open distribution. This open source advocate (employee) says he does not mind that Novell sell expensive proprietary products on top of the openSUSE product to enterprise customers:

> "No, these are not free [Groupwise, ZenWorks and other proprietary Novell products]. And actually I am not sure if these need to be free. Because most users for the desktop system don't need it. **And we have to make money somehow. And when the enterprise customer pays for it it's OK for me**" (interview #10 – my emphasis).

The role of the object

Now that we have shown how the openSUSE object is a legitimate (and important) boundary object, we can return to the unique features of this particular object and the point I am trying to make. I have mentioned that the openSUSE object has an ability to draw the collaborative actors together, and I further claim that it creates a fundamental motivation for the entire project. To illustrate this point I would like to revisit the data from my contributor-survey, and specifically figure 10 in chapter 4 (p.88). The figure shows that the most motivating factor among community developers is that they all believe that openSUSE is the best Linux distribution; this is what they claim is most important for their willingness to participate in the project. Whether openSUSE in fact *is* the best distribution or not, is beside the point. The impression that the openSUSE distribution holds undisputed quality is most likely created by the developers' own experience with the software, along with other community members' opinions that confirm this impression. This subjective experience then feeds right back into the identity they associate with the openSUSE object, which in turn motivates them to continue pursuing its development. This finding shows the importance the openSUSE object plays for the motivation of the contributors in the community.

Now is a good time to raise this discussion to a more theoretical level. What are the characteristics of the openSUSE object that makes it different from the other objects we have discussed? And what does this mean for our understanding of boundary objects? I believe we are witnessing that boundary objects can play different *roles* in collaboration. Bugzilla serves a different purpose than the openSUSE object in the cooperation between Novell and the external openSUSE community. I believe this finding addresses a weakness in the theory of boundary objects, as there is a lack of distinction of the object's individual role in the collaboration effort. It is clear that there is a difference between objects that represent the end-goal of the collaboration effort, and those that merely serve as a tool for reaching other aims among the respective collaborative

groups. I would therefore propose a distinction between those objects that are *means* and those that are *ends* of the collaboration effort, and suggest to name the former *supportive-objects* and the latter *target-objects* (since their development is the target of the collaboration). Bugzilla and the OBS are examples of supportive boundary objects, while the openSUSE object is clearly a target-object. This understanding provides a new impetus for explaining the motivating qualities of the openSUSE object: Since this object represents the ultimate end-goal of the collaboration, this role is obviously important in explaining one of the driving factors behind the project. We may understand the importance of this quality by reviewing other some theories that address some similar aspects; the concept of *epistemic objects* (Knorr Cetina, 1997, 2001; Rheinberger, 1997) together with our knowledge of *anthropological symbols* may provide strength to our understanding of target-objects.

Target-objects: towards a deeper theoretical understanding

Target-objects hold a property that the supportive-objects do not; they represent a *self-reinforcing motivational factor* in collaboration. To emphasize this point, we may borrow inspiration from another theoretical strand on objects in collaboration: The notion of epistemic objects in scientific work (Rheinberger, 1997) – also called knowledge objects (Knorr Cetina, 1997) – emphasizes the role of the object as the goal of collaboration. Since the object is the end-result, it has not yet become what it later will be. Epistemic objects are therefore "as much defined by what they are not (but will, at some point, have become) than by what they are" (Knorr Cetina, 1997, p. 15), and they "embody what one does not yet know" (Nicolini et al., 2007). This creates a "lack of completeness" in the object that ultimately spurs a desire among the people working on it. Knorr Cetina describes this as the object's *structure of wanting* (1997). The desire of working on the development of the object is common for all the actors, and is therefore one of the means that hold them all together. The openSUSE object contains these qualities. Consider, for example, this statement from a community developer (written in an open text field under the question "Why do you participate in the openSUSE community?" in the contributor survey):
*"[Because] I _**want**_ it to be the best distro... even though I think it still falls short in some areas."*
This statement includes *his* emphasis and illustrates his desire to make openSUSE the best possible Linux distribution. Although he might not believe openSUSE actually *is* the best distribution at the moment, it still drives him to contribute with what he can to reach that goal. It is a perfect illustration of how the openSUSE object contains the structure of wanting characteristic to epistemic objects. I therefore conclude that this is an aspect that can be generalized to all target-objects, as they all represent ends in collaboration. According to Nicolini et. al, any object that is "in the process of being materially defined, functions as an epistemic object". This could seem problematic for the case of the openSUSE object, since this already *is* materially defined. Knorr Cetina claims however that objects which are "simultaneously things-

to-be-used and things-in-a-process-of-transformation" (1997, p. 10) must be included in the category of epistemic things.

A second strand of thought that gives credit to the motivating capabilities of target-objects can be found in Emile Durkheim's classical work on social ritual theory (1965 [1912]). His theory emphasizes the role of objects (also referred to as symbols or totems) as centerpieces of attention among social groups:

> "Placed thus in the center of the scene, it becomes representative. The sentiments expressed everywhere fix themselves upon it, for it is the only concrete object upon which they can fix themselves" (Durkheim, 1965 [1912], p. 250).

When people come together in social rituals[20], the interaction generates sentiments – a "sort of electricity" (Durkheim) or what Randall Collins calls *emotional energy* (Collins, 2004, p. 38). We then use symbols and objects as representations of the abstract sense of collective consciousness that is created when we as collaborators come together, "[f]or we are unable to consider an abstract entity the source of the strong sentiments which we feel. We cannot explain them to ourselves except by connecting them to some concrete object of whose reality we are vividly aware." (Durkheim, 1965 [1912], pp. 250-252) The objects are thus charged with the sentiments that come out of our meetings:

> "What is mutually focused upon becomes a symbol of the group. In actuality, the group is focusing on its own feeling of intersubjectivity, its own shared emotion; but it has no way of representing this fleeting feeling, except by representing it as embodied in an object" (Collins, 2004, p. 37).

The powerful capability of Durkheim's symbols are that they able to unleash the sentiments they are charged with when the group is dissolved and individuals are apart from each other: "it is as though the cause which excited them in the first place continued to act." (Durkheim, 1965 [1912], p. 265). In this way the symbol has the ability to continue to unite the group when they are *not* together. Individuals are drawn towards the objects because of what they represent and the sentiments they contain. Lars Risan (Forthcoming) has studied how this social mechanism is manifested in the Debian Linux community, where he finds a "love for Debian" among the Debian developers. He describes this emotion as an "expression of sentiment, desire and belonging that goes far beyond economic rationality" (Risan, Forthcoming, p. 26). We can relate this back to the statement from the openSUSE contributor above, who desperately *wants* the openSUSE distribution to be the best distribution. It is likely that his emotional attachment to the openSUSE object is what explains this desire, rather than any rational motivation.

[20] Note that a social ritual does not need to have any religious association. Rituals are tied to human interaction and have a recurrent nature. They can exist on a large or small scale, from the gathering of a crowd (e.g. a demonstration), a board meeting or even a handshake (Collins, 2004, p. 34).

Can we append the capabilities of Durkheim's symbols to our understanding of target-objects? The theory certainly fills a critical gap in our thinking so far: namely accounting for a larger whole that is capable of holding together two geographically distributed autopoietic systems – an explanation that gives us *more* than an aggregate of various individual incentives and motivations, and that fits with what we have witnessed in the case of the openSUSE object. In the chapter 2 I mentioned that Star's theory of boundary objects may need an extension that can explain a larger unity that can hold the collaborating groups together. I argue that the symbolic value in objects such as the openSUSE object is the *manifestation of the symbolic unity within an organic solidarity*, and provides the "glue" to keep it all together despite diverging interests and other conflicts that may be.

It is however important to stress that target-objects and anthropological symbols are *not* one and the same. Durkheim's symbols are representations; they are placeholders for the sentiments of the collective's physical presence when it is not there. Target-objects are something *more* than that. The openSUSE object is not only a container of the communities sentiments. Drawing upon our understanding of epistemic objects above, we see that the object *itself* holds an intrinsic motivating quality, in addition to being a carrier of individual and collective sentiments in the sense of Collins and Durkheim[21].

I would like to make a last note on the topic of target-objects. In voluntary work, individual motivation may be more important than in other communities of work. I believe that what we now know about target-objects addresses a weakness in our research on open source developers and on explaining their motivation to contribute lots of hours of development for no pay. In the motivational grid drawn up in figure 2 in chapter 2 (p.11), I have grouped the various motivations found in the literature within the categories of egoistic, altruistic, intrinsic and extrinsic (the compilation is based on findings in (Dahlander & Magnusson, 2005; Demil & Lecocq, 2006; Hippel & Krogh, 2003; Lerner & Tirole, 2002; O'Mahony, 2003; Raymond, 2001; Stallman & Gay, 2002; Weber, 2004). This grid can account for most answers you will receive from any open source developer (it can for example explain each of the 70+ extra comments on reasons for participation, that were supplied in open text fields by respondents in my contributor-survey). However, I have argued adamantly for target-objects' importance in creating motivation in collaborative networks, yet this point does not show up elsewhere in the literature referred to above. Why? When individuals are asked about their reasons for doing something, they start reasoning and accounting for the motivations they are able to find within

[21] I would like to add that my use of Durkheim's understanding of symbols is coherent with Knorr Cetina's concept of object-centered sociality (Knorr Cetina, 1997). Although it is tempting to elaborate on the relationship between them, this thesis does not include enough room for a further discussion of this topic.

themselves, preferably looking for a rational explanation the rest of the world can acknowledge[22]. As researchers, we are also constantly looking for answers *within the individual*. Perhaps also because we are social scientists, we can loose sight of the role of material artifacts surrounding us. I would say that neither of these individual, accountable reasons explain why a developer or user wants to wear a t-shirt with the logo of the software product he is working for, or why he or she gets seriously upset when someone makes a bad remark about the software or say they prefer Microsoft Windows. Individual reasons also fall short of explaining how all these single developers are drawn together and united towards the same goal. As figure 15 above illustrates, the object exists in all grids, and makes the dichotomies between egoistic-altruistic, intrinsic and extrinsic rather irrelevant. The object exists *externally* to all individuals, yet they all identify themselves with it. Hopefully, the presentation and discussion of target-objects has provided some new insight on this area, and shown how an object can provide such motivation.

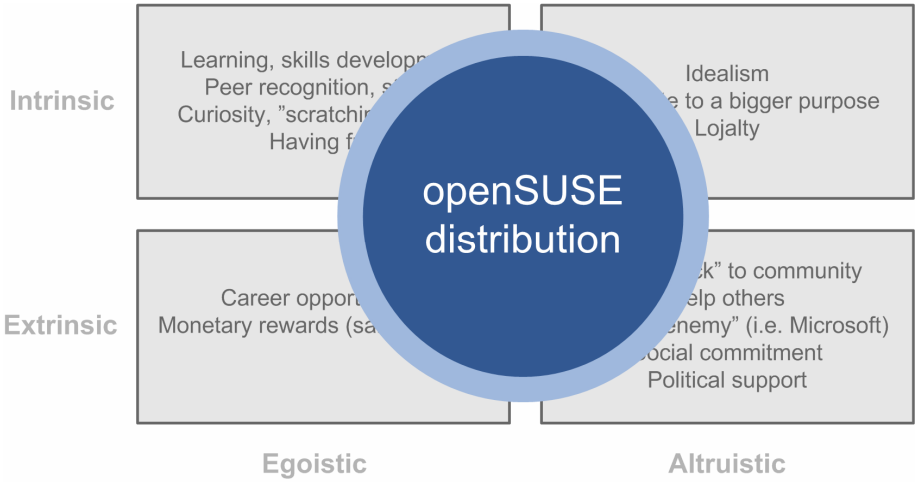


Figure 15. Target-object motivation.

We have now reached the end of this discussion on the role of boundary objects. The case of Novell and the openSUSE project has shown how we can improve our understanding of this theory, with the distinction of supportive- and target-objects. In return, the understanding of target-objects may also provide some advice for Novell: If they want to attract motivated contributors to the community, they need to give as many developers as possible access to the openSUSE object. We will return to this subject in our discussion of future scenarios towards the end of this chapter. Instead, we can sum up our discussion so far: We have seen how the

[22] Having been an active student-spokesman working full-time for over a year at a student-office for no pay, I have first-hand experience with the tedious process of explaining curios friends and concerned family about the reasons for such insanity.

autopoiesis of social systems creates boundaries between the Novell organizational system and the interactive openSUSE system. Boundary objects are situated in-between and ensure system-connectivity, whilst also maintaining separation. Target-objects, on the other hand, contribute to form a unity surrounding both systems, interlocking them together. This understanding above is however not *enough* to explain how the actors surrounding the openSUSE project stay united. Boundary objects are only *objects*; they do not act on their own. In a moment we will therefore look at which *people* play a particularly important role in activating these objects. First, we return to investigate the boundaries between Novell and its outside environment closer, and the channels that make important communication through the organizational membrane possible.

Shared communication channels

Although the boundary objects mentioned are vital for the collaboration, there are other important elements as well. The open communication channels are crucial for providing an arena where the collaboration can take place. The network of electronic communication channels consists primarily of the mailing lists, the wiki, and the IRC channels. These serve important purposes within each system, as mediators of communication. For the openSUSE interactive system they are especially crucial, as they are the only channels of communication in the system and since most members of the community are geographically distributed across the globe without being able to meet physically. Within Novell, other communication channels exist (meetings, coffee breaks, etc), but in Novell's Linux R&D the managers insist on using the open electronic channels for development work in order to create transparency and enable interaction with the external community members. According to my informants, almost all project development is communicated here; exchange of ideas, initiation of new projects, discussion about current and future solutions, problem solving on existing bugs and so on.

On the mailing lists, each participant in the community and the company can configure their own communication channels, related to their own interest. By selecting which lists to subscribe to, they can follow certain threads and topics and ignore others. The IRC channels support synchronous collaboration. The Wiki is especially suited for providing a portal to the other communication channels and for presenting static information, documentation and archiving communications on the mailing lists. Following Luhmann and Michéle Morner, we can say that these shared communication channels contribute to the *stabilization* of the interactive system (Morner, 2003), preventing it from dissolving over time. They key features in achieving stabilization are modularity of communications, accessibility to the channels and information, traceability and documentation of communications. These features are all provided by the mailing lists, IRC channels and the Wiki.

The communication channels in the organizational system and interactive system are shared. Before the openSUSE project was initiated and these channels were opened, communication concerning product development did not cross the organizational boundaries of Novell/SUSE. If we continue the metaphor from the previous section, one might say that these channels of communication are the *aquaporins* at of the organizational membrane. They ensure that communication concerning product development flows freely through the Novell's boundaries. The aquaporins escape the selective permeability that is exhibited by a cell's membrane; water can flow in and out whenever it pleases[23]. In a similar fashion, there is no censorship or control of communication on the community channels, anyone can post a statement that is transported throughout the entire network.

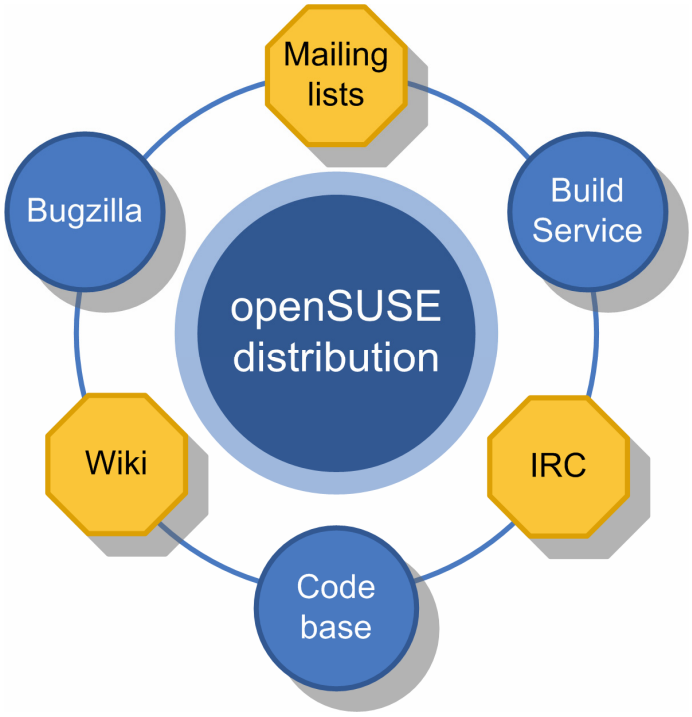


Figure 16. Network of boundary objects and communication channels.

The communication channels and the boundary objects are tied intricately together, as illustrated in figure 16 above. As shown in the example from the IRC chat, communications surrounding the objects often take place on community channels. Another example is provided by the fact that the mailing lists "Factory" and "Build Service" carry the very names of their respective objects. We can see how a set of objects and channels are important in binding the organizational and interactive system together, and form a sort of integrated network. While this set of objects and

[23] Water is the only substance that has is able to flow in and out of a cell unconditionally through the aquaporins, the traffic of other substances are "controlled" selectively by the cell's membrane.

channels contribute to system connectivity, there is also one last category of forces at work that bind the systems together: the people that communicate within both systems.

Marginal people

The distinction between the interactive openSUSE system and the organizational system in Novell will for some people seem artificial. Or, at least, they would rather argue that the boundaries between them are fuzzy and unclear. These are the people that actively communicate in both systems. They fill positions in the organizational network where they participate in decisions. At the same time they communicate actively with the openSUSE community which they are also a part of. They are on the one side the employees and managers in Novell's openSUSE department that are active in the community structures, and on the other side the external community members that have earned their stars and trust within the organizational system. We might say these are people with *dual membership*. In the terms of Susan Leigh Star, people whom inhabit multiple social worlds are called *marginal people* (Bowker & Star, 1999; Star & Griesmer, 1989), as they live at the margins of several domains. This understanding draws upon Robert Park's marginal man, "the one who has a double vision by virtue of having more than one identity to negotiate" (Bowker & Star, 1999, p. 302). In Novell and the openSUSE project these identities are related to the commercial software model on the one hand and the normative structure of the open source model on the other. Marginal people may typically experience tensions imposed by the multiple membership, and problems of identity and loyalty.

Who are they?

Within Novell these people are primarily found in Novell's Linux R&D, and may be identified as the most eager internal advocates and participants of the openSUSE community. In total there are approximately 300 people employed within this R&D team, but only a fraction of these may actually fit the description of marginal people. There are many employees that are used to the 'old way' of working in a closed development process, as this manager illustrates:

> "I encourage people [employees] to be more outside, I encourage them to communicate upstream and also on the openSUSE mailing lists, to actively participate, but for some of them it takes time. So even if we are doing open source development, a lot of people think in a closed source world, and have trouble communicating with the outside" (interview #14).

From my meetings with individuals at Novell who *are* active on community channels, I found that a common characteristic among them is a background from participating in other open source communities, previous to their employment in Novell. There are also examples of employees whom have never been involved in communities before, but have still become among the most active advocates and participants in the openSUSE community. I argue that the

experience and familiarity with the open source model (as a methodology and normative structure) among the former group has nevertheless been important in order for others to follow.

From the statistical analysis of the main developer-mailing-lists we can operationalize an indicator of these marginal people, and locate an approximate number of Novell-employees that may be described as such - in terms of their participation on these communication channels. For each mailing-list participant, the data includes the total number of entries and the number of months the entries are distributed over, by this participant *for each* mailing list (see overview of these lists and their participants in the previous chapter). The data is from the time period of September 2006 to October 2007. Based on these variables, it is possible to create an index in order to rank the participation of mailing list users. The following is an algorithm for such an index, which balances participation over time with amount of contributions:

$$\sum_{i=0}^{n} = x_i^2 + y_i$$

where n = 5 mailing lists (the developer-lists), x = months participated and y = number of entries (emails to the list). A participant whom has sent 60 emails to one list over 3 months will gain an index of 69, whilst another participant whom has sent 20 emails over 10 months to one list and 15 emails over 4 months to another will have an index of 151 (120 + 31). As the observant reader will notice the number of months you have participated increases exponentially in value over time, meaning that it is more important to contribute over time than to post a large amount of emails within a limited time-frame in order to get a high ranking. I tested the results of this and a few alternative ranking-algorithms with some of my key informants, and found that the algorithm above gave the results that fit their understanding best of which people are the most active participants on the lists. The next step we need to take, is to figure out the minimum amount of participation we would expect from someone belonging to the group of marginal people. My informants identify that the top 150-200 participants belong to the category they would describe as active community members (employees and non-employees). In order to be within this group an index-level of 30 is needed. It seems therefore that a reasonable index level could be set at 30, and would indicate noticeable participation on the lists. Sorting the list with the algorithm above and an index level above 30 yields the following results:

In total there are 174 users with index > 30 from this time period. Of these *40 users are Novell-employees*. Below is a chart where these *employees* are distributed according to their participation-rank in five equal categories (intervals of 40):

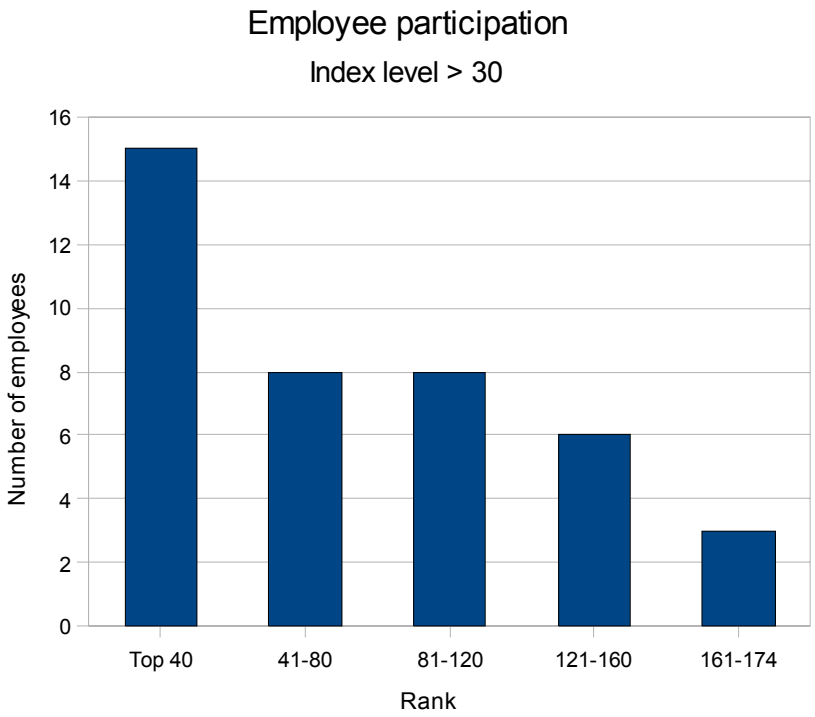


Figure 17. Participation by employees on mailing lists.

The chart in figure 17 may need a little explanation: Imagine that we were looking at the list of the 174 participants with index level above 30, ranked from the most active on the top to the least active at the bottom (the ones on the bottom will have an index level close to 30). The chart shows us that among the 40 most active people at the top of the list, there are 15 employees (the user ranked as number one on the entire list is in fact an employee). Among the next group from 41 to 80 in rank, there are 8 employees, and so forth. As we can see, the numbers drop increasingly in the categories that follow. This would indicate that the chosen index level includes those employees that in fact *are* active in the community, and that we do not need to worry about having excluded many employees that otherwise should have been taken into account. Based on this we can say that the amount of marginal people within Novell amounts to no more than 40 individuals, which is 13% of the employees in Novell's Linux R&D[24].

The other set of marginal people involves the community members on the other side of the organizational boundary, whom also are active in both systems. It may be more difficult to show how they can take part in Novell's organizational system without being employees. Although they are not able to make any direct decisions in this network, they may be rather influential on decisions with the organizational system as they are trusted with some level of responsibility.

[24] Note, however, that the employee-count in the chart does not differentiate between departments within Novell. Some of them might be from other departments than Novell's Linux R&D.

They may even be treated as if they inhabit positions in the organizational system. A community member I spoke with (interview #25) has taken the responsibility of maintaining a popular package for the desktop interface. Although a package maintainer in Novell is officially responsible for this software, the external community member does most of the work with this specific package. He therefore also has a natural level of influence on matters concerning this project. Other examples of marginal people on "the outside" may include community members in decision-making structures such as the openSUSE board, or members whom are assigned authority to work as editors of the wiki's news-pages. There are more examples such as these, but they are more the exception than the rule as Novell has released little control of the product to external community members.

The number of external members that are included by this definition is harder to count than in the previous case. We can not simply use the activity on the mailing lists as an indicator, as high communicative participation does not reveal whether you are a trusted and influential individual in the Novell system or not. We may however use data from the contributor-survey:

5.1. To what degree do you experience an ability to influence and take part in decisions involving the openSUSE project?
If you would like to answer "I do not know", please leave the question unanswered.

not at all	to a little degree	to some degree	**to a large degree**
19 %	31 %	37 %	**13 %**

N = 222

Table 8. Community-influence on decisions.

The question in table 8 above shows that 13% of the respondents experience a large degree of influence on decisions in the project. Since there are 222 respondents on this question, it accounts for 26 individuals[25]. This number seems more reasonable, compared to 135 external community members with index level above 30 on the mailing lists. Remember also that the survey only includes respondents that participated voluntarily and on their own initiative, and that the absolute number of people that experience high level of influence in the community therefore should be somewhat higher.

With the numbers in the table above, we can make an educated guess that there may be approximately 30 members in the community that belong to the group of marginal people. But to what degree can we trust that these numbers in fact represent the people we are after? Do they

[25] Actually, it accounts for 28 individuals. 2 of these are however employees, and I have therefore subtracted them from the number.

really have any influence? To investigate this question I searched for correlation between the respondents that experience a high level of influence with other variables in the survey. The results show that these same individuals also respond positively to being motivated by getting their footprint in the distribution, getting acknowledgement for their work and from the fact that the distribution has an enterprise user-base (questions 3.5, 3.6 and 3.8 in the survey, with Gamma above 0.45 on all cases). More importantly, high levels of influence also correlate with high contribution hours, meaning that contributors who put in more work also have more influence (or vice versa). Another interesting finding is that level of influence correlates moderately with having social ties (Gamma 0.37), and that this correlates again with amount of hours contributed. Put together, it seems that there may exist a group of core contributors having influence on decisions, on the product, and receiving acknowledgement for their work. They have some social ties since they communicate a lot together, and they are motivated by the fact that their work is "out there" among a large user base. These findings therefore offer credibility to the assumption that there exists a handful of marginal people on the external side of Novell's boundary. This finding is also backed by openSUSE managers in Novell that identify the same people as the external contributors they experience as most active and valuable to the project. Their number balances with the total amount of marginal people on the inside of the company as well. In total, this group of marginal people is not very big in numbers. They nevertheless play an immensely important role in making the model of collaboration between Novell and the openSUSE community work in practice. This is how:

The strategies and importance of the marginal people

The common denominator of the marginal people, whether they are employees or not, is their strong normative commitment to the openSUSE distribution and the joint development model. While the boundary objects and communication channels provide a collaborative infrastructure, they do not fully explain how the systems are held normatively together or how the link is protected from internal and external pressure threatening to break it. Examples of such internal pressure may be the old Novell organization's resistance to change and reluctance to fully embrace the open source development model, or the openSUSE community's desire to be an autonomous open source community. Externally, pressure may come from the commercial software market upon Novell to monetize their investments with short-term proprietary solutions. On the other side, the openSUSE community may be under attack from the larger open source community's lack of faith and criticism towards the project. Underlying the pressure for separation is the normative structure of the open source and commercial software models, contrasting each other. A figure of these examples is provided below.

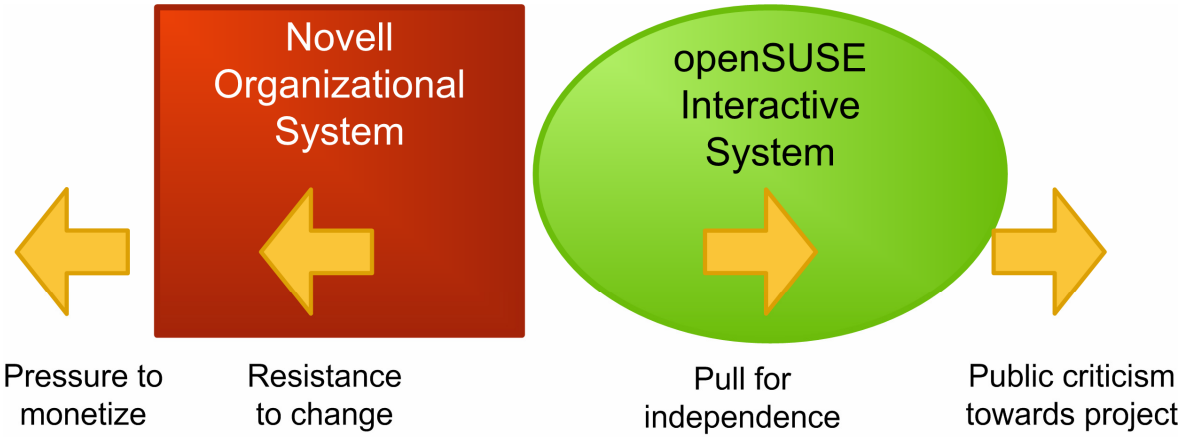


Figure 18. Internal and external pressure towards separation.

Although Novell's top executives and official strategies may provide the structural link between the two systems, it is the work of the collaborating developers that enacts this structure in practice. They represent forces that are important in explaining how the pressure modeled in figure 18 is countered. The practice of the marginal people contributes to the creation of a new social reality, where the hybrid software model (which combines the commercial and open source model) represents the dominating understanding. The strategies they employ (that I have identified) are either:

enforcing, such as putting in effort to make the collaborative development model work - and thereby proving its worth;

aggressive, such as advocating the model in public discussions and blogs, convincing colleagues to join in, or defending the model against external attacks; or

symbolic, as in leading by example and inspiring others to support the project by showing their own commitment.

An example of the latter is provided by this informant, after he was asked what could cause the community to abandon Novell: [W]hat you can do is to fire key people inside the company. For example, if we had fired [John] then other people [in the community] would get really pissed off." (interview #10). Novell have in fact been pursuing this strategy consciously, by recruiting respected and known open source developers to key positions in the company. By holding these positions in Novell, these key people visualize to the outside community that they perceive Novell as a legitimate open source actor themselves. (This does not mean, however, that these individuals do not voice their disagreements with Novell or that they pose as puppets on behalf of the company).

Examples of the aggressive strategy may be found everywhere, on mailing-lists, blogs, in the lunch room or at conferences and expositions. This is a strategy that within the community is commonly referred to as "evangelizing". Novell even has "evangelist" as an official job-description. Aggressive strategies may happen pro-actively or re-actively. The following is an example of the latter: During the autumn of 2006, Novell signed a deal with Microsoft which created controversy in the open source community. Some claimed the deal would threaten open source developers' motivation for working with Linux, and Novell in particular. (This issue is disputed, and I will not go into any more detail here). Shortly after the deal was announced, the top manager of an alternative Linux distribution posted a mail to the opensuse-project list, lamenting Novell's deal with Microsoft and inviting opensuse-developers to join "his" distribution instead. The first reaction that followed was the reply of an external community member who wrote:

> "I think the tone of this message is controversial. You can have any opinion you like against the Novell/Microsoft agreement, but taking this to try such hack against openSUSE developers is very far near a war declaration. I hope it's not. For I don't see [other distribution] better than openSUSE on any respect."

Several more replies followed, rejecting the invitation and defending the integrity of the openSUSE project. Many of the replies were sent by marginal people in the openSUSE community, employees and non-employees alike. The example illustrates the importance of their efforts: this publicly available email-tread must undoubtfully have been read by hundreds of openSUSE community-members, if not more. If the core group had not responded so quickly and decisively, the "attacker" might have succeeded in his mission[26].

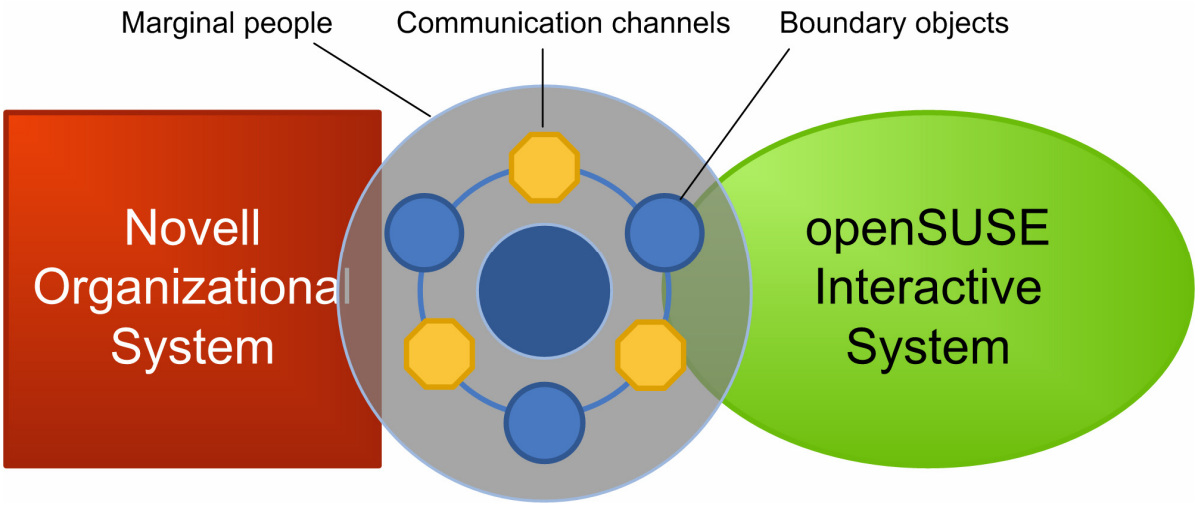


Figure 19. Elements of system connectivity.

[26] I base the assumption that he was rather unsuccessful on information from two openSUSE employees.

In summary, the analysis above has shown how two systems co-exist and collaborate in the creation of a software product. The discussion has focused on explaining how the collaboration is held together by the means of boundary objects, shared communication channels and the effort of marginal people in both systems. The three elements are intricately linked, and work in unison at weaving together the two groups, as shown in figure 19 above.

The underlying assumption is not an understanding of the two systems converging towards an equilibrium, as we can find in functionalist thought. The two systems are in constant reproduction and change, and their relations to each other are by no means static. In the next section we will therefore look at what these theories may tell us about the future development of Novell and the openSUSE project.

Future perspectives

The Novell organizational system and the interactive openSUSE system are autopoietic and under constant internal reconstruction. They can evolve in several possible directions. Michéle Morner (2003) argues that open source projects provide a good example of autopoietic evolution, as some projects seem to stabilize while others die once communications stop. With openSUSE, anything might happen. Novell, on the other hand, is a somewhat more stable system, but is also constantly evolving and shifting its boundaries between the two systems. It seems as if Novell's evolution is somewhat experimental, and adapts itself to changing conditions and responses from the environment. For example, Novell did not know if opening Bugzilla would be successful, and were prepared to close it again if the results did not turn out as expected. It however proved to be a large success, stimulating Novell to move on with openness in other areas.

Uncertainty exists as to how Novell will solve the dilemma of granting external contributors access to commit code to the code base and strengthen the collaboration with the community. We are standing at a point in time where many things may happen, and the tension between control and openness can drive the relationship between Novell and openSUSE in different directions. At the time of writing, the situation has already evolved and changed since the data was gathered in the autumn of 2007. I will address these changes towards the end of this chapter. In the following, I will present a few possible scenarios for the future evolution of Novell and the openSUSE community that are interesting to discuss empirically and theoretically. We will se that Novell have already started to implement the last of the scenarios that are presented here. However, the evolution of Novell and the openSUSE project is not a given, and a failure in one scenario may lead to another. Moreover, these scenarios could be applicable to any firm-sponsored open source community and can therefore be generalized to a larger population. An overview of the scenarios is provided in table 9 on the next page.

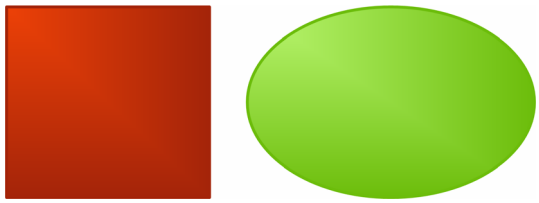
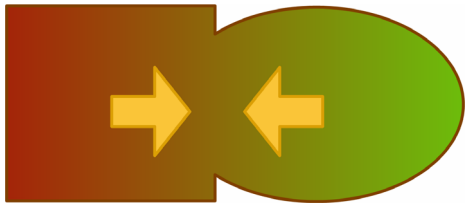
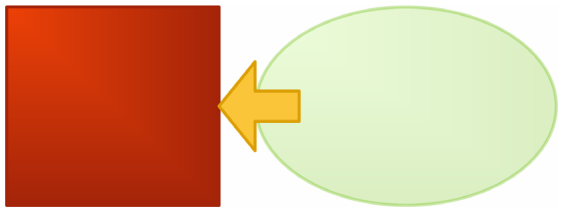
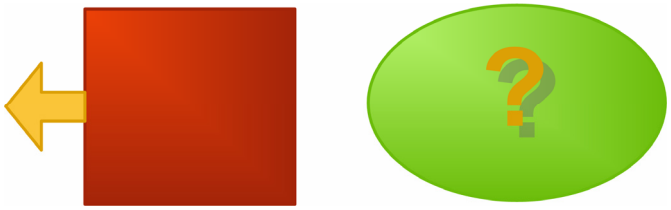
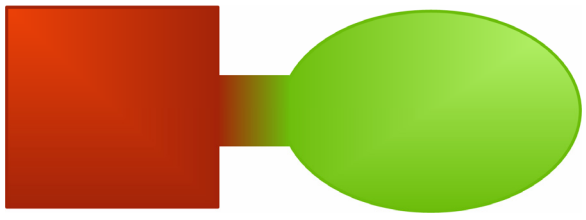
Status quo: The relationship between Novell and openSUSE remains as it is. Boundary: Continued *selective permeability.*	
Integration: Novell and openSUSE engulf each other and become one and the same system. Boundary: *Complete permeability.*	
Extinction: The interactive openSUSE-system dies, and the development goes back into Novell. Boundary: Back to *impermeability.*	
Divergence: Novell and openSUSE move in opposite directions, and openSUSE evolves into an autonomus organizational system. Boundary: *Diverse permeability*	
Failure: Novell's business model fails and the company pulls out of the openSUSE project. openSUSE's fate is uncertain and open for anyone to grab. Boundary: *Dissolved*	
Bridge: Novell and the openSUSE system continue to exist as today, but a clear bridge is built between the two systems, enabling mutual influence. Boundary: *Qualified permeability*	

Table 9. Future scenarios for the openSUSE project

Status quo

The first possible scenario is for the current relationship between Novell and the openSUSE system to continue to stay as it is. This would mean that Novell's boundaries will continue to exhibit selective permeability, meaning that all decisions are made by Novell and that any external input is filtered selectively. Basically, Novell can choose if and how they want to use any external contributions, and external developers need to go via employees if they want to commit code into the code base. This situation contributes to the tension between openness and control, but as I have shown in the previous discussion, the potency of mechanisms such as boundary objects and marginal people are strong enough to manage this tension. However, my data indicate that these mechanisms will not suffice indefinitely, rendering this scenario an unlikely permanent situation. Among community members I have been in touch with, the lack of ability to contribute directly is rated as the largest current challenge for the collaboration between Novell and the external community, and must be dealt with soon: "It's critical. Objective number one I would say (…) Because you can't on the one hand say that 'hey, be involved, join and help us!' and on the other hand say 'yeah, you can work on that, but you cannot touch those important things.' " (interview #18 with community member). The issue was also repeatedly mentioned by external contributors in the commentary field in the survey, as this example illustrates: "We need to make it easier for non-Novell people to improve the distribution and have their patches integrated." This pressure is acknowledged by Novell, who also agree that something needs to be done. One of the openSUSE managers claims that "we really need to provide a way to directly leave your footprint there if you want it." (interview #9). A product manager in R&D elaborates:

> "What we need is a defined process and rules for how external people can simply contribute to the code. As simple and directly as possible, and at the same time ensure the quality, the security and everything. And that's a tricky task" (interview #8).

So, although the current arrangement may hold for a while, it is likely that the relationship will evolve into one of the other scenarios in time.

Integration

One option is that the development of openSUSE is opened up completely, and all differences between employees and external developers are erased. In this scenario, we might witness a situation where Novell and openSUSE engulf each other. External developers will be included in the decision network and be embraced by Novell's organizational system. This would not mean that external developers become employees of Novell, but rather that anyone external to the company may gain rights to commit code and part-take in decisions at the same level as employees. Developers might also be given responsibilities as if they held positions within the organization. In O'Mahony's terms, it would mean that Novell's openness moves completely

from *transparency* to *accessibility* (2007a). Similarly, Novell's organizational system will extend beyond its boundaries of the employee-network and into the interactive system, and Novell will increasingly participate in external open source software projects. The company itself would have to shift more of its identity, as the openSUSE distribution will be increasingly *community managed* (O'Mahony, 2007b). I will return to this point in a moment. In the integrated model, the boundaries between the systems would be completely permeable¸ meaning that traffic and access into and out of Novell's development process would flow *indiscriminately*. In practice, it would mean that Novell and the openSUSE interactive system become one and the same system[27]. It would also mean that the goals of Novell and the openSUSE system not only remain compatible, but are in fact *harmonized* so they become one and the same. This leads us to our first obstacle.

There are some problematic issues with the integrated model, which might make it an unlikely outcome of the ongoing evolution. First, is it possible to fully integrate Novell's goals with the aims of the external developers? My data indicate that community members may have a hard time of incorporating the company's interests into theirs:

> [Myself]: "How much do you think the community members identify themselves with Novell, as opposed to openSUSE?"
> [openSUSE manager]: "Not so much. I guess there is also this fear about Novell having too much control. And every time Novell does a stupid move, it falls back on to the community. But I don't think that there are many people in the openSUSE community that identify themselves with Novell" (interview #9).

I investigated this relationship closer in the contributor survey. Here are some results:

4.1. I believe Novell is an open source company

not at all	to a little degree	To some degree	to a large degree
12 %	19 %	50 %	19 %

N = 273

4.2. openSUSE and Novell are one and the same to me

not at all	to a little degree	To some degree	to a large degree
37 %	28 %	26 %	10 %

N = 275

[27] Remember that the interactive system is not equivalent to the openSUSE community – there would still be an user-community surrounding the organization in this scenario, but it would make less sense to define external from internal contributors and users.

4.5. It would be fine with me if openSUSE was designed in red Novell colors

not at all	to a little degree	to some degree	to a large degree
55 %	21 %	15 %	25 %

N = 268

Table 10. Identity between Novell and the openSUSE community

Although many community members believe Novell is an open source company (at least to some degree [69%]), the results from the next questions in table 10 show that most community members still distance themselves from Novell. The quote and the survey-results therefore illustrate that a majority of the external contributors draw a clear line between the company and the community. This could indicate that it would be a challenge to succeed with harmonizing the two systems. However, these data only illustrate the *current* situation. They do not prove that the situation cannot change, only that it might be a challenging task.

This brings us to the second problem, which concerns the changes Novell would need to make in order for the integrated model to work. A closer examination of the data above indicate that Novell would have to become a "pure" open source company in order to wipe out the differences between the two systems: Statistical analysis shows there is a strong correlation between question 4.1 and 4.2 (Gamma = 0,55), meaning that many of the individuals who see Novell as an open source company are the same that see openSUSE and Novell as one and the same. This would indicate that in order for community members to identify with Novell, the company needs to "be more open source" in order to get the "rest" of the community to feel the same way as the 10% in question 4.2. There are however several reasons why this strategy would be problematic for Novell. Not only does the culture within the entire company need to change and more proprietary products become open, but Novell also needs to release complete control of the product development and let it become entirely community managed. Not all openSUSE engineers believe that Novell will be able to achieve this:

> "I don't see at the moment that Novell will give up complete control of openSUSE. And I am not sure about how this is communicated, and I see this as one of the bigger problems the project will face in the next couple of years. Because at *some point people will begin to expect something*. Because at the moment they are allowed to build packages, and to help other people on the mailinglist and on the wiki, but what they *say* is completely... Well, [a few of the managers] listen to it and we welcome ideas, but if there is a real conflict, for whatever technical thing in the distribution, *Novell will always win*" (interview #23, my emphasis).

The difficulties of implementing an integrated model does not mean that openness can not be achieved. There are alternative scenarios that may achieve accessibility in a different manner. We

will return to discuss these in short while. First, let us see what may happen if more openness is *not* achieved.

Extinction

If we for a moment imagine that Novell do not succeed in finding a solution to achieve some form of accessibility, they will risk losing the developers and contributors that really matter. According to the community members I have spoken with, there seems to be an anticipation within the community that the conflict of community-influence will soon be resolved. Developers currently seem to be waiting patiently for access to be given. If these expectations are not met within due time, Novell risks that the active openSUSE communty may fall apart since individual developers might move to other software communities instead. The effect of losing developers due to continued denied accessibility could result in that the openSUSE community becomes viewed as a pure "fanclub" in the larger open source community, which may in return inhibit any further recruitment of developers. Without contributors, the communicative events may stop and the system will die. The distribution will probably continue to be developed in-house in Novell, although it may still have big windows providing transparency into the development. In practice, the boundaries will nevertheless go back to being *impermeable*.

I find this model a rather unlikely outcome of the ongoing evolution, even in the event that no change happens in Novell's relationship to the community. Luhmann argues that systems stabilize when they contain *communication connectivity* and a *systemic memory* (Morner, 2003), as described previously. Since I have argued that these features are quite present in the openSUSE interactive system, it will not fall apart easily. I also believe that the potent boundary objects I have previously described are able to keep people drawn towards this collaboration and prevent any complete flight, as long as Novell keep the objects open and active. The community could however shrink or diminish over time, without reaching extinction.

In any case, the concern of losing the community (or never succeed in building a powerful community that makes a difference) could go away if Novell grant more access to external developers. Why is it not obvious for them to just do so? As previously mentioned, the SUSE Linux Enterprise product is closely tied to the openSUSE distribution since they share the same code base. The largest problem for Novell with allowing unconditional access to the development process is the company's ability to guarantee software quality to its enterprise customers, since alterations in the openSUSE code base will directly impact the enterprise distribution. Given this complexity, I believe there are two ways for Novell to provide more openness towards the community. The first option is to pursue a strategy where the code base is split in two, so that any "harm" that may come to the openSUSE code base does not have any direct impact on the enterprise distribution.

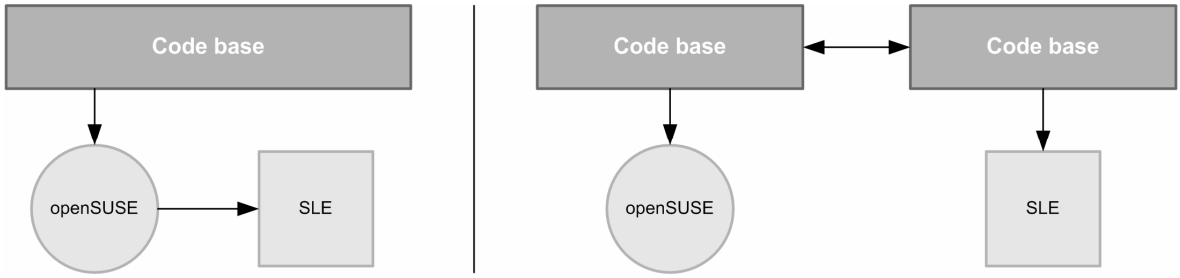


Figure 20. Split code base.

In this case alterations in one of the code bases would only selectively affect the other, as illustrated in figure 20 above. Therefore Novell can take the risk of giving direct access to external developers. According to my informants, this is a model that is used by Red Hat and their open Fedora distribution[28]. Let us have a look at the consequences of this scenario for Novell.

Divergence

If Novell were to pursue a strategy as described above, it would increase the likelihood of a scenario where Novell and the openSUSE community diverge apart. If the code base is split, the interactive openSUSE system would control its own distribution. This would definitely give rise to an organizational system surrounding the openSUSE object, as a structure needs to be in place in order to handle the decisions that need to be made. Positions and responsibilities within the community will be defined, and decision protocols will be developed. The openSUSE interactive system will thus develop its own decision network and evolve into an organizational system in its own right. In the illustration of this scenario, the openSUSE system is therefore visualized as a square rather than an elliptical. The boundaries of the Novell and openSUSE system will have *diverse permeability,* since there will be different access to the openSUSE development and SUSE Linux Enterprise development in Novell – the former being far more open.

In the event of this scenario, it is likely that Novell and the openSUSE system will drift more apart from each other. This is largely because the openSUSE object will alter its identity and shift its position, and thus lose some of its "power" to draw the Novell organization close. It will no longer be a target-object between the two systems, as Novell engineers will now work directly on the enterprise distribution rather than via openSUSE. The other supportive boundary objects will – instead of being one set of objects between the two systems – become duplicated in two sets and exist as naturalized objects within each of the systems. Their role as "glue" between the systems will therefore disappear. Communication channels will also be more concentrated on connecting the systems internally, and communication across systems will dampen.

[28] In Fedora, voluntary developers are able to achieve commit-rights to the code base through a trust-rating system. I do not have any data on numbers of developers or how successful this model is in their case.

Is this model desirable for Novell and the openSUSE community? This question is not for me to answer. One of my informants claims it will cost Novell a lot of money and recourses that they might not have, as it is more costly to maintain two code bases that simply one. My informant also has a theory that having only a single code base is a strength to the company and the community:

> "I think it is an advantage [to have the same code base]. Because we have... Suse linux has some reputation of being made in Germany and have some quality (...) We would give this up [by splitting the code base]. So what people told me, at exhibitions, is that they were happy that we do not give it up completely, that we do NOT do it like Fedora – just buy the brand name and let the community do the work. Because there is quality attached to it, and that we need the code base for the business products. So I think it is an advantage, but I think it is an advantage you have to communicate" (interview #23).

Again, I wanted to find out if my informants' impression was confirmed by the community:

3.8. I participate in the openSUSE community because my work for openSUSE may be used by many people, including enterprise level

not at all	to a little degree	to some degree	To a large degree
14 %	17 %	32 %	38 %

N = 263

The results show that the link to the enterprise product actually motivates a lot of people, but so do many of the other variables in the survey as well. However, the responses to this question correlate moderately with the question of how many hours the members contribute weekly to openSUSE (Gamma = 0,35). This means that for the majority of active developers in the openSUSE community (those who contribute the most), the fact that the product they influence is the same that is distributed through Novell's commercial network is important in explaining their motivation to participate. It would therefore seem as if my informant above has a point. This may indicate that openSUSE *might* lose developers or fail to attract enough talented effort in this scenario, since the openSUSE object can lose one of its important identity characteristics (engineered to high quality and distributed to enterprise users). However, the opposite might also be true. The increased accessibility to the distribution can recruit a large amount of developers to the community. In sum, I can not say whether this scenario is good or bad for Novell or the openSUSE community, but it is certainly an alternative.

Failure

Another scenario that may occur, and which dawned on me rather late, is that Novell might lose interest in openSUSE and abandon the project altogether. This may happen if the business model fails and if it no longer benefits the company to have a Linux operating system among their products. I could only speculate in the reasons for why this would happen, but it would most likely mean that the firm-sponsored community model failed (although it may be for other reasons than the development-model itself). The consequences for the openSUSE community – which already is a somewhat stabilized system – will depend on the fate of the Novell employees working on openSUSE; whether this part of the company is sold out so they can continue Linux development or directed to work on other tasks in Novell. One managers said that he hopes that "even if Novell would abandon the project, the project would be so lively that it would live on without the corporate sponsor" (08.05.08). Due to this uncertain fate, a question mark is put in the visualization of this scenario. In some form, the openSUSE distribution will continue to exist although the development effort is likely to be severely reduced.

Bridge

As mentioned previously in this chapter, I believe there are two realistic ways for Novell to provide more openness in their relationship to the community. The first is to split the code base and cut openSUSE loose so that full control of the enterprise distribution can still be maintained by Novell. The second approach is to take steps to scale up Novell's quality review processes and system of trust so that it can include certain external contributors. This is the final scenario I will present in this section, and the model Novell and the openSUSE community seem to be heading towards already.

In this model, accessibility is given *discriminately*, meaning that contributors have to qualify to earn the trust to gain similar rights as employees towards the code base. I therefore name this boundary *qualified permeability*, as it will permit certain individuals to roam freely between systems within a defined route of passage. This is visualized as a bridge between the two systems in the illustration of this scenario. In a way one could say that Novell would be extending their decision-network into the openSUSE interactive system, but they will still be separated as two systems since external developers will still not have the full rights (and obligations) as employees on other areas. In this scenario, Novell will trust certain external developers the access to submit changes and updates to the code base, and the opportunity to gain this status should be equal to all external contributors. The quality review process should treat internal and external developers in the same manner, and thus ensure that quality is maintained even if the code is received from outside the company.

The shrinking arm

Interestingly enough, Novell have already been able to move quite a step along the path of this scenario during the writing of this thesis. I constantly noticed new topics and threads concerning this issue on the project mailing lists over the next months after my visit to the company. I was also reminded about this development towards the end of my work. According to our agreement, I sent drafts of my text to Novell for their review, and received comments in return. A director of one of the departments in OPS R&D writes:

> "You speak about 'Holding the community at arm's length' - in several places [in the thesis]. I would like to see added that the arm is shrinking ;-) " (07.05.2008).

In a following phone conversation, he adds that his goal is to "erase the distinction between external and internal contributors" (08.05.2008). In May 2008, Novell will release a new version of the openSUSE Build Service, where "internal and external maintainers are handled technically equivalent. This means the Novell employees can use the same accounts and technical solutions as external developers" (written comment from OBS-maintainer, 08.05.2008). In other words, Novell will have made it technically possible for external developers to commit code in the same manner as internal engineers, and integrated the OBS with the Autobuild system. They will also have moved more of the employees' development process into the OBS, thus creating a joint arena for development between external and internal developers. In theoretical terms they are strengthening the use of the boundary object instead of the internal, naturalized object (Autobuild).

When this technical infrastructure is in place, the next step will be to create a system for granting access to trusted developers. How does one decide which individuals are qualified to commit to the code base? And who gets to make the decision? If the two systems are to be maintained as equal partners around the openSUSE distribution, it can not be up to Novell alone to define the qualification criteria or to decide who gets to be qualified contributors. In collaboration with the community, Novell are therefore also developing a *user-trust model* that will be technically integrated in the OBS. The goal of a user-trust model is to have a system that can calculate a trust-level[29] for all registered developers that wish to share software with other users. The point of such a system is based on the following principle: "In general, the trust in a certain software comes from the people behind it. If we can trust them, we can trust their software."[30] The rating of a developer in such a system can depend on a number of factors. A discussion-document on the project includes these working suggestions of factors that will affect developers' trust-level:

[29] For example, trust-level can be reported as a number between 1 – 100.
[30] Quoted from one of the working documents on the user-trust project, at http://en.opensuse.org/Build_Service/Concepts/Trust

- users accepts and signs a contract
- rating from other users
- user registers personal contact details
- user obliges to deliver updates for software project (ibid)

By defining criteria such as these, the community is also creating a system of incentives for stimulating developers to pursue trustworthy activities (in order to gain a higher ranking). When the user-trust model is in effect, every user that wants to download software from the OBS can see the rating of the author of the project and thus get an indicator of whether the software can be trusted. More importantly (at least for this discussion), the trust-rating system will also provide an instrument for Novell to qualify external developers, by using a measurement-tool and criteria that have legitimacy in the external openSUSE community. Once the qualified developers start submitting contributions towards the common code base, Novell's quality review processes will ensure that the contributions also meet the necessary standards, in the same manner as all contributions from employees are scrutinized.

In sum, Novell are constructing the bridge between the two systems without the need to drastically change the company or eat up the community. This technical and social bridge will be an important mechanism for managing the tension between openness and control between the Novell organizational system and the interactive openSUSE system. It is built by *empowering the objects* situated between them, especially by strengthening the role of openSUSE Build Service and giving externals more access to the openSUSE object. The second important consequence of pursuing this strategy is that it may *expand the group of marginal people*, on both sides. Employees that previously strictly did their development indoors through Autobuild, will expose their work in the Build Service and are thus likely to interact more with the external community. Likewise, external developers may achieve rights to commit code through the OBS, and thus be part of Novell's decision-network. I believe this may be a promising strategy for *managing openness*. But, although the structure of this scenario is falling into place for Novell and the openSUSE community, it however remains to be enacted in practice.