WHICH WAY TO OPEN DESIGN?

Learning from Designer's Successes and Failures in Open Source Software Development Communities

Kovač Aleksandar*, Kittiwongsunthorn Warruntorn*, Katsuhiko Kushi*
**Kyoto Institute of Technology, Sakyo-ku, Matsugasaki-Goshokaido chou*
Kyoto 606-8585, Japan

Abstract: The internet has become a globally accessible space for people to create and innovate together, regardless of their physical location, geographical boundaries or cultural backgrounds. More importantly, it is proving to be a suitable context for open collaboration on a global scale. However, even in this seemingly ideal context, designing and the implementation of design solutions in the open source paradigm remains challenging. This paradigm does not provide organizational structures that support the designer's conventional role and without them, conventional design methodologies fail. Using the research methods from social sciences, three hindering categories and three facilitating categories of social circumstances found to influence the implementation of design solutions in OSSD projects are encoded and presented. These provide a basis to propose a model of adaptive behavior for designers as a set of heuristics aiming to improve a designer's ability to collaborate, contribute and design with and within an open-source community by alleviating the effects of hindering social circumstances, while retaining the facilitating effects. This model calls for three heuristic techniques: a) "Designer's socialization" – aimed at improving a designer's ability to collaborate on open projects, b) "Designer's cultivation" – aimed at improving a designer's ability to contribute in open source communities and c) "Designer's initiative" – aimed at improving a designer's ability to design within the open source paradigm.

Keywords: design methodology, open design, sociological aspects of open paradigm, open society

1. Introduction

1.1. Current condition

The "open source" paradigm is described as a production paradigm in which all the information necessary to produce a final product is freely accessible and redistributed [1]. Historically, this concept has always been a fundamental condition for the development of sciences, technologies, education, art, cuisine and many other human creative achievements.

The open source paradigm is particularly relevant and exciting today as it benefits remarkably well from the digital information technologies and seemingly ubiquitous connectivity of the present-day. The ability of people to communicate and act online regardless of their geographic location or cultural backgrounds presents a context for open source collaboration that could not have been imagined ever before. In this context, individuals and groups online can come forward with ideas and a means of production. They self-organize, collaborate and share information to create together. Despite its creative potential, creating and designing in the open source paradigm has not received the attention it deserves from the standpoint of design science or practical design methodology.

This kind of collaborative creativity is especially vibrant in online communities gathered around open source software development (OSSD) projects. OSSD project communities can be generally described as online communities of an indeterminate number of voluntary members assembled around the development of a particular software. Community members take on tasks by themselves, often according to a personal feeling of "fun, contribution or accomplishment" [2] and without detailed plans or schedules. In many cases, their work is made public a-posteriori, i.e. after submitting it to the community for assessment. Merited community members assess contributions and decide whether to include them in the next software version [3]. The ethos [4] of open source development creates a meritocratic social context where disputes are often resolved based on rational arguments and where free experimentation, innovation and collaboration is welcomed and supported.

As an OSSD project gains acceptance and the number of users and/or contributors grows, the inclusion of human-related design issues such as usability, user interaction, user experience or overall project identity becomes increasingly relevant. At this stage, designers often attempt to contribute design solutions for such issues, however, these contributions are often considered unacceptable to OSSD communities [5] and as a result, are rarely implemented. Furthermore, it has been observed that conventional design methodologies in general prove inefficient in an OSSD context, and very often fail.

1.2. Aim, goal and area of research

Aside from informed opinions and interpretations based on anecdotal evidence, existing scientific knowledge does not offer solid insights into the causes for design methodologies and/or designers failing in the open source paradigm. One possibility for the lack of scientific investigation is that open source development by online (and off-line) communities is an immensely complex, dynamic system that remains persistently difficult to analyze [6].

The goal of this research is to gain a structured insight into the collaborative design process within OSSD projects, with the aim of providing a nomothetic explanation of causal relationships between social circumstances involving designers in OSSD communities and the implementation of design solutions proposed by designers.

This field research addresses an emerging area of open source design that includes design science and methodology, social science, collaborative and open production models and digital networks. The ultimate goal is to contribute an updated design methodology applicable in the open-source production paradigm.

2. Methodology

Research approach was determined by the following three characteristics of OSSD:

1. The dynamism and complexity of OSS development, evident at even the basic levels of production and communication, is comparable to the dynamism and complexity of societies, thus, research methods from the social sciences were deemed appropriate.
2. The immense volume of correspondence in open source communities is unmanaged, often asynchronous and concurrent. At the same time, it is documented, archived online and openly accessible in the form of mailing list archives, blogs, microblogs, opinion pieces, software repositories etc. This superabundance of information provides a valuable source of research data.
3. As a consequence of the reduced social cues in online communication and the bias of open source culture for action, it is not possible, nor necessary to confirm the precise backgrounds of contributors in OSSD communities. Indeed, an OSSD contributor often takes on a combination of roles: "designer", "developer" and "user". In addition, contributors could be groups, individuals or entities with multiple online personalities.

With that in mind, our research approach was designed as an iterative triangulation method relying on grounded theory (GT), case studies and participatory action. These research methods were chosen from the social sciences. GT allowed for the extraction of relevant insights from the superabundance of information in the field, while case studies and participatory action offered a multilevel perspective on OSSD, from the level of an observer to the level of an active participant (a contributor) in the OSSD process. [Figure 1] Each triangulation cycle went through three phases (extraction, affirmation and testing) during which codes, concepts and categories were extracted.

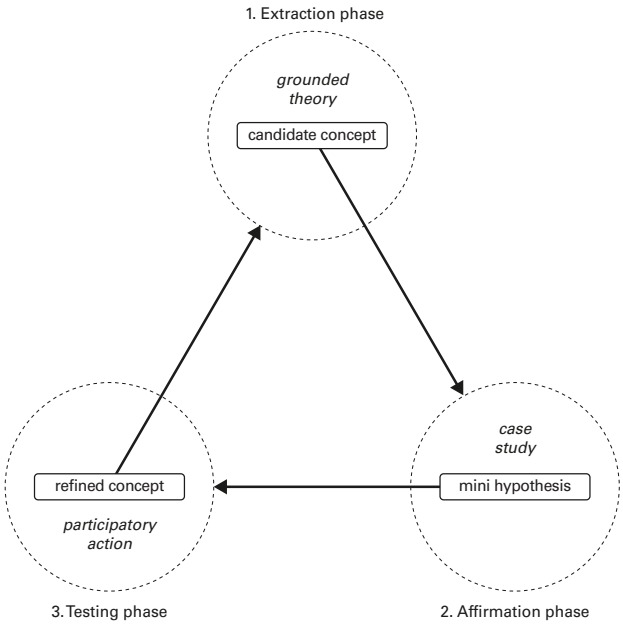

Figure 1. Conceptual diagram of the triangulation cycle showing the three phases (extraction, affirmation and testing) and the research methods used in each phase.

In the context of this paper, it is assumed that the role of "a designer" is a contributor (individual or legal entity) that creates or contributes to the creation of design solutions in OSSD projects [7]. A "developer" is a contributor that creates or contributes to the creation of programming and coding solutions.

2.1. Extraction Phase

Each triangulation cycle begins with a wider perspective on OSSD and relies on the GT method to extract "candidate concepts" from data sources. Following the GT method, "a candidate concept" is a formed statement or a direct quote extracted from data sources relevant to the field of research. Early on in the research, candidate concepts were mostly extracted key codes and concepts [Figure 2]. As the research progressed with new triangulation cycles, the candidate concepts allowed general categories and theories to be formed. For illustration, a candidate concept "programmer's code is easily tested, user experience is not" was extracted following a comment by an OSSD project contributor stating that focusing on user experience slows down the OSSD process since, unlike program code that "either works or not", it is "not possible" to provide an exact and universally acceptable evaluation of user experience. Since this topic seemed promising and relevant, "programmer's code is easily tested, user experience is not" was added as a candidate concept for the next phase. This phase allowed the extraction of many other candidate concepts such as "Good design ideas can trigger code development", "good looking projects are more desirable for developers", etc.

Figure 2. A screenshot of encoded data from the OSSD mailing list during the extraction phase. Columns, from left to right: e-mail message senders, correspondence threads, tags and codes attached to each message. The rightmost area chronologically displays content of one correspondence thread.

2.2. Affirmation Phase

Once a candidate concept is extracted in the extraction phase, the triangulation cycle refocuses to affirm the validity of that candidate concept relying on the case studies. A candidate concept is considered affirmed if it can be found to be true and applicable in various real accounts. Again, the transparency of the open source development process is essential in this phase since it enables access to numerous real accounts. An affirmed and validated concept can then be carried into the next phase as a "mini-hypothesis". Mini-hypotheses are simple abstracted statements stemming from candidate concept(s) but encoded from accounts of similar and opposing opinions concerning this topic. Example mini-hypotheses' structures are: "Doing (A) will contribute to acceptance of the idea."; "(B) reduces possibility for an idea's implementation."; "Approach (C) is an appropriate way to do (D)" etc.

2.3. Testing Phase

So far our insights, (candidate concepts and mini-hypothesis) although inducted from real world circumstances, have remained in the realm of theory. For additional verification of insights it is important to experience the "designer's" role in the first person, utilizing these insights. Therefore, it is necessary to refocus the research once more, to direct participation.

In the testing phase we tested the mini-hypotheses. Through direct participation,

mini-hypotheses were introduced to a project community in the form of: ideas, proposals, questions or opinions. The responses from the community usually arrived very fast. A refined concept that was accepted, then became a part of our refined data. An example of the verification process pulled from the research involves the testing of the mini-hypothesis: "The success of ideas in open source depends on their articulation." In the GIMP developer mailing list [8], a GIMP user started a thread that strongly demanded changes to the design (abbreviated) [9]:

> "...in fact, make GIMP really easy to use, and powerful. make animation part of the package instead of a separate piece. some of us are losing out. make menu items intuitive. ... adobe photoshop filters (Mac or PC) should be built in. ... people should have to install a plugin. (...) GIMP without photoshop filters is practically useless (...) GIMP should be powerful."

The lack of a clearly articulated problem or possible solution by the user provided the opportunity for us to test the mini-hypothesis with the following reply:

> "...The same factors that contribute to acceptance of some code, will contribute to the acceptance of your ideas. If your ideas are articulated, presented and clear enough, and just work for other people, (i.e. clearly indicate the 'excellence' of your approach) they will be created/ implemented/ improved somehow somewhere by someone. That's how open source works. Maybe I am painting an idealized picture, here?"

The answer from a prominent member of the GIMP community affirmed the mini-hypothesis in his statement:

> "You aren't. Peter started the brainstorm blog exactly with intention to get input from users and see a bigger picture. One of the points in that blog is that if you can't present your idea in a clear concise way, maybe you didn't really think much about it..."

In the previous example, it was possible to add the following refined concept: "The success of design ideas in open source depends on their articulation." At the end of the testing phase, a successfully affirmed mini-hypothesis, or a "refined concept" is then put back together with the GT data for further iterative triangulation cycles or GT axial coding.

2.4. Research Frame

The main portion of this field research focused on online documentation and correspondences from four OSSD project communities [Table 1] over a period of 18 months, from April 2010 through May 2012. During that period, over 11,000 messages were encoded. The following period, from May 2012 until February 2013 was used to refine and verify the findings. Besides the four OSSD communities mentioned, additional data sources included, but were not limited to: online documentation and correspondences from other project communities, blogs, microblogs, various social network groups, news portals and academic papers.

3. Findings

In OSSD communities, a designer faces social circumstances fundamentally different from the social circumstances encountered in conventional organizations. In conventional organizations, members communicate and act in adherence to recognized roles and plans within a well structured production environment where defined methods, goals, schedules, etc. play important roles in achieving production goals. Therefore, an organized and structured approach in such organizations is commonly perceived as "normal", "desired" or "expected". In contrast, only a few simple cultural conventions maintain the ethos of open source communities. Activities within OSSD communities are mostly individual, network

OSS project	Description
Syner [*www.syner.org*]	A project to develop a social network for social problem solving. The project experienced a surge of activity at its start. With the departure of the initiator and project leader, the project lost its steam and is currently inactive. 11 members used Google Wave for correspondence and documentation.
Inkscape [*www.inkscape.org*]	A rich scalable vector graphics (SVG) editing software. Community uses nine mailing lists and numerous forums for correspondence. The documentation is accessible on the website. The project is relatively mature with a stable community of more than ten thousand members.
GIMP [*www.gimp.org*]	A comprehensive bitmap image editing software. GIMP developers use six mailing lists for correspondence. GIMP is a mature project with a stable community of developers and users with immense amounts of documentation online.
Mozilla Thunderbird [*www.mozilla.org/projects/thunderbird/*]	An e-mail client. Thunderbird, once a very popular e-mail client, has since seen its development wane. Faced with software obsolescence, Thunderbird users overtime transformed the existing support forum into a valuable source of comments and ideas on the future direction of the project. [*http://getsatisfaction.com/mozilla_messaging*]

Table 1. OSSD projects whose online documentation and correspondences were focused on during the main portion of research.

distributed, asynchronous, unmanaged and concurrent. The conventional organizational structures necessary to support the conventional role of the designer do not exist and without them the conventional design methodologies also fail.

Using the research methodology described in the previous chapter, we were able to code six categories of social circumstances that arise in OSSD communities and directly influence the acceptance and implementation of proposed design solutions in OSSD projects. It was possible to divide these six categories of social circumstances into two key groups: hindering social circumstances and facilitating social circumstances.

3.1. Social Circumstances Hindering the Implementation of Proposed Design Solutions

"Hindering social circumstances" are social circumstances causing OSSD communities to reject proposed design solutions. The following subsections seek to explain three categories of hindering social circumstances involving designers and other contributors presented in the order of occurrence, as the designer would encounter them when contributing to an OSSD project. The categories are referred to as: "disregard for apprenticeship", "alien cultures" and "fragmented visions".

3.1.1. Disregard for Apprenticeship

The first category of hindering circumstances may arise at the very onset of a designer's effort to contribute in an OSSD community. This is caused by a disregard for apprenticeship, i.e. a contributor's unpreparedness for the task. The disregard for the apprenticeship can come both from the designer or the developer. The manifestation of the hindrance differs whether it is the designer or the developer causing it, but the effect on design implementation is nevertheless the same, it hinders a designer from contributing in an OSSD project.

In the case of the designer, the disregard for apprenticeship manifests as the designer's failure to exert the patience necessary to get acquainted with the open source paradigm or OSSD project they are attempting to contribute to. Familiarization with the specific OSSD project that a designer aims to contribute solutions to, will help them understand the appropriate modes of contribution. The failure of a designer to observe the culture of the OSSD project they aim to engage in, effectively excludes proposed design solutions from consideration by the community.

In the case of the developer, the disregard for apprenticeship manifests as the developer's failure to exert the patience necessary to get acquainted with the range of topics that design commonly deals with. Developers, for example, have the programming ability to implement user interfaces, but rarely exert the patience necessary to consider human fac-

tors or the reusability of such user interface implementations. The resulting substandard user interfaces are released and users become accustomed to them. Once this occurs, a designer's proposal for a user interface improvement will often be seen as destructive and therefore not be accepted.

3.1.2. Alien Cultures

"Alien cultures" refers to hindering circumstances that arise when tools, methods, techniques and jargon used in one field of expertise are not applicable in another, impeding successful collaboration between designers and developers in OSSD projects. For example, online tools for versioning and managing software code do not provide appropriate support for documenting and managing data formats for design development.

Another example of hindering effect of "alien cultures" can be seen when the implicit rationale behind the designer's design decisions being unintelligible among software developers and vice versa. A discussion from the GIMP developer mailing list [10] illustrates one instance from this category. A potential contributor repeatedly urged GIMP developers to include his color swatches palette as one of the default palettes with the next version of GIMP. However, the need for this inclusion is not clear to the developers. Since the rationale for it has not been clearly articulated by the contributor, the future implementation of the palette seems improbable. After the first rejection by the GIMP community, the palette contributor replies:

> "...before talking about a product you have to try and design with it, is the best proof that a product is good and consistent."

Verification that requires the direct experience of designing is not a rational one to the developer. In addition, the developer has second thoughts whether the proposed color palette meets the technical requirements for color management in the printing process and the conversion between additive and subtractive color models:

> "I tried your swatches. I didn't see anything else than just swatches. A lot of swatches. Now you say your printed book (from CMYK swatches, using a hand-picked generic CMYK profile) has no relationship with your sRGB swatches... What exactly should I try?"

Subsequently, the palette contributor failed to produce a satisfactory argument and the initiative to include the palette was ceased.

Designers contributing design proposals are often unable to develop software code themselves. As a result, design solution proposals depend on the developer's acceptance of an idea in order to be implemented. Due to the effect of "alien cultures", implementation of a design solution proposal is hindered by misunderstandings and incompatibilities between designers and developers.

3.1.3. Fragmented Visions

OSSD community members come from different backgrounds, therefore, it is inevitable that community members will have different personal conjectures regarding various aspects of the project. "Fragmented visions" refers to the inability to attain consensus in design implementation due to differences in personal visions that result in the rejection of design proposals. Characteristic of the open source paradigm, fragmented visions are common among OSSD community members.

Fragmented visions may also hinder the implementation of design proposals when misinformed personal conjectures about the design process and it's purpose have a negative influence on the stance of the community members towards design [11]. In an example from the Drupal mailing list, a developer communicated a personal conjecture suggesting that those designers that are not competent in template coding cannot help in developing the Drupal project. In this case, fragmented visions were potentially hindering the accep-

tance of any design proposals made by contributors without coding experience, despite the merit of their idea.

> "So, my point is, that a designer who can't override a theme template is not a Drupal designer, and therefore he/she can't really help me with Drupal development. The barriers to participation for designers are really low, but yes, there are barriers you just can't skip."

3.2. Social Circumstances Facilitating the Implementation of Design Solutions

"Facilitating social circumstances" increase the likelihood of design solutions being accepted by OSSD communities. The main characteristics of these facilitating social circumstances are the clear articulation and efficient filtering of information in design proposals by contributors. Articulated and filtered information benefits all parties involved by providing a "common ground" where the communication and exchange of ideas can happen.

The importance of information articulation and filtration can be exemplified in their essential role in turning online OSS support forums into a valuable resource for both users and developers [12]. In these support forums, solution seekers and solution providers have an incentive to be articulate in order to be understood. In turn, they must filter through other communications on the forum in order to extract relevant information. Users use support forums to articulate problems and filter relevant information in order to receive solutions. Developers filter questions and articulate answers to provide solutions to user problems while gaining insights that aid them in improving OSSD projects.

The following subsections discuss three categories of social circumstances that have been found to facilitate the implementation of design solution proposals in OSSD. They are referred to as: "design focused efforts", "hybrid approaches" and "concurrent efforts".

3.2.1. Design Focused Efforts

"Design focused efforts" are individually or communally maintained websites (such as open source design projects, blogs or research websites) where designers openly present their thinking, design development process and finalized design solutions. These efforts adhere to the open source production model, with a focus on design rather than coding. As a result, it is possible for such efforts to advance work pertaining to design practice and design theory without introducing the hindrances described in the previous sub-chapter. Websites for design focused efforts are an open source of both articulated and filtered design information. It has been observed that some design ideas that originated at such websites, spread very quickly and influenced various OSSD projects.

Open design information available through "design focused efforts" was found to include open design solutions, templates and tutorials, "bootstrapping" kits (e.g. Twitter bootstrap CSS and Java boilerplate for website scaffolding [13]). It also included the analyses of design issues in existing projects, explaining and emphasizing important design points and opinion pieces.

As a part of this research, a small, design-focused website was created that discusses "papercuts" [14], which can be defined as trivial and easy-to-fix usability bugs that the average user would encounter in everyday software use. This website focused on papercuts in Inkscape software [15] by providing an articulated and structured overview of each papercut together with a solution proposal [Figure 3]. Articulating design issues in this manner succeeded in attracting and activating Inkscape users and developers to participate in solving design-focused issues [16].

3.2.2. Hybrid Approaches

"Hybrid approaches" refers to a category of social circumstances where the design process is partially or entirely performed outside the open source paradigm, while the results of such design process are released under an open source license. Designing outside the open

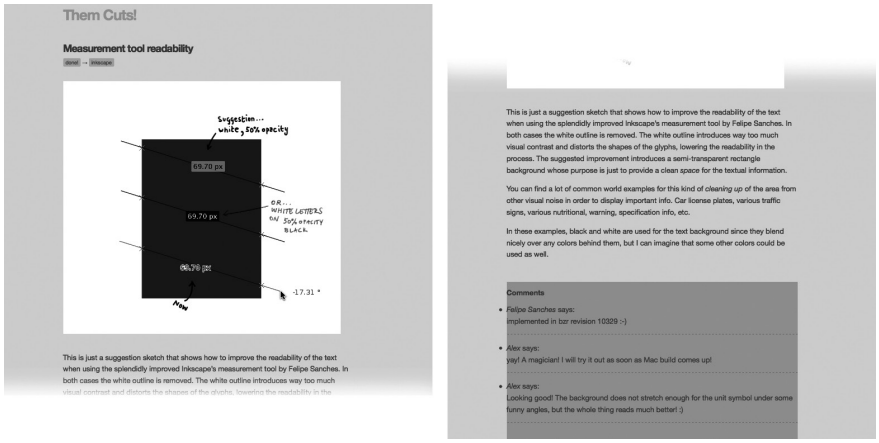

Figure 3. 'Papercuts' website screenshot showing an initial design suggestion and a short discussion that resulted in successful implementation of the suggestion. (http://cuts.thinking-garment.com/273)

source paradigm introduces unique collaboration conditions (e.g. the enlisting of professional designers) and sacrifices some of the advantages of open sourced production. In return, however, it benefits from a structured and organized approach to the design process.

"Hybrid approaches" facilitate the implementation of design solutions in OSSD in two ways. Primarily, design solutions are implemented for the purpose they were commissioned. Once implemented, these solutions are within the open source paradigm and can be freely used and modified which leads to further, secondary, implementations by other OSSD projects.

Hybrid approaches in OSSD are common. For illustration, we will mention three examples. In the first example, Canonical. Ltd [17], that leads the Ubuntu project (open source operating system) hired Dalton Maag type foundry [18] to design the "Ubuntu" font family for the "Ubuntu" GUI. Once designed, the font was released under an open font license [19] that functions similarly to open source software licenses. In the second example, Mozilla Foundation [20], the non-profit organization behind Mozilla Firefox browser and many other OSSD projects, maintains their own design department and retains the final design decisions. Nevertheless, design and production details are openly available and in adherence to the open source paradigm. An independent OSSD project, music player software called "Songbird" [21], is an example of secondary implementation. It uses Mozilla developed XUL (XML User Interface Language) for the design of it's own user interface. Finally, the third example is the development cycle of Google Android [22] where the codebase is opened by Google only after code stability has been reached.

3.2.3. Concurrent Development Efforts

Conventional organizations seek efficiency and cannot afford redundant work. In such a setting, the economies of scale [27] and scope [28] matter significantly. As a result, careful management of the production process is imperative for achieving profitability.
There is no such imperative in the open source paradigm. Concurrent activities done in order to achieve a common purpose or result are a prominent characteristic of the open source paradigm that stems directly from the definition of open source [23]. In practice, OSSD projects rely on various version control systems to support concurrent development efforts. Contributors are free to choose tasks, pursue individual approaches and reuse the existing solutions via branching, forking and "cloning". These circumstances offer a rich choice of openly shared solutions for OSSD contributors to consider, facilitating the implementation and reuse of articulated design solutions, as well as coding solutions, beyond the limits of the project of origin.

For example, a multi-platform toolkit for creating graphical user interfaces GTK+ [24] was created to assist in the development of GIMP, but it transcended that use and has been implemented by other OSSD projects that need toolkits for multi-platform GUIs. In another example, "Hyde" [25] a static HTML page generator written in the Python programming

branching: parallel development on revision-controlled version of code. *forking:* starting a project based on a version of the pre-existing project's source code, *"cloning"*: developing a project to mimic another project or product.

[David A. Wheeler: *Why Open Source Software / Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers!, Ch. A.6 Forking.* Retrieved from: http://www.dwheeler.com/oss_fs_why.html#forking]

language, clones the functionality of "Jekyll" [26], a static HTML page generator written in the Ruby programming language.

4. The Model of Adaptive Behavior for Designers in an Open Source Community

The open source paradigm presents many advantages for the design process. At the same time, the hindering social circumstances are emerging, hindering the design proces and implementation of design solutions in OSSD. Although these hindering social circumstances have no critical effects on OSSD, they are detrimental to the advancement of open source design. In conventional production models, structured organization can alleviate most of the aforementioned hindering circumstances through formalized communications and procedures optimized for achieving clearly stated goals. The open source paradigm does not provide an environment in which such structured organizational tools can be applied.

In working towards a future open source design methodology, the aforementioned findings provide a solid basis to propose a model of adaptive behavior for designers through which the effects of hindering social circumstances can be alleviated while retaining the facilitating effects when designing in the open source paradigm. This model follows three heuristic techniques, referred to respectively as: designer's socialization, designer's cultivation and designer's initiative [Figure 4] through which a designer acquires the essential ability to collaborate, contribute and design in the open source paradigm. It is worth noting that this is not a set of heuristic techniques specific to any particular OSSD community to address design issues, but rather, a general model of social adaptive behavior for a designer in the open source production paradigm.

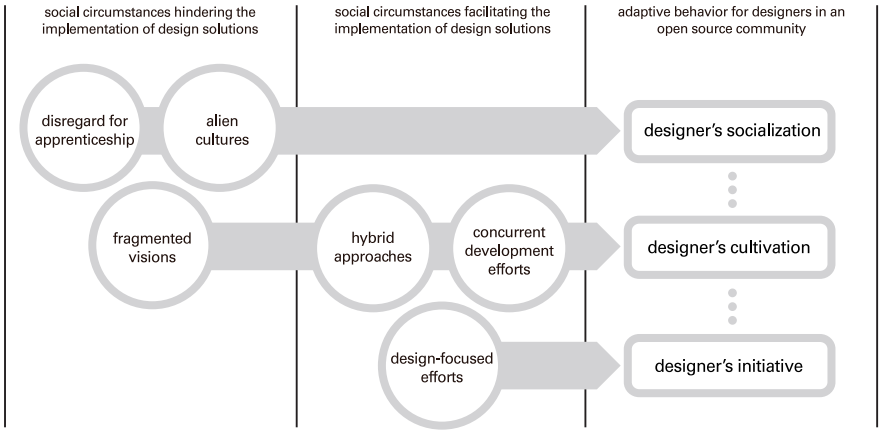

Figure 4. Hindering and facilitating social circumstances found, informed three heuristic techniques of adaptive behavior for designers in OSSD.

4.1. Designer's socialization

Designer's socialization is a heuristic technique aimed at improving a designer's ability to collaborate on OSSD projects. Socialization is a voluntary and often independent process. It begins with adapting the designer's mindset to the open source paradigm which involves the designer learning social patterns, techniques and jargon necessary for effective collaboration. In adapting, a designer acquires an accurate personal conception about the open source paradigm, along with various development and licensing models. The designer then learns basic information about a project that includes, but it is not limited to: the project's idea and purpose, the project's goals and visions and the process of submitting proposals. All of this is done by reading available documentation and taking part in the project's discussions on mailing lists, fora, etc.

In OSSD, the designer cannot rely on a pre-defined role or status of a "designer", as is

typical in conventional production environments. Through socialization, the designer can gradually evolve her position while gaining the trust of community members; a process that compels the designer to be patient and acquaint themselves with the project. Online wiki documentation for the Celestia OSSD project describes this process and stresses its importance as a form of "apprenticeship" [29]:

> "As with most software projects with established developer communities and protocols, a potential contributor to Celestia will have to undergo a form of apprenticeship. The program is large and complex and its underlying design philosophy has to be understood. It also takes a while to gain the trust of the people currently working on the program. By themselves, programming skills aren't enough. Too often individuals unfamiliar with the program or with the people involved have suggested significant changes and have become discouraged because their ideas weren't incorporated immediately. Don't let this happen to you. Read the forum. Read the developers mailing list. Participate in discussions. Contribute ideas. Try out things with the code on your own. Be patient. It'll take a while."

4.2. Designer's Cultivation

Designer's cultivation is a process of acquiring and persistently improving a designer's ability to filter and articulate information, with the purpose of improving a designer's ability to contribute design solutions in OSSD communities. The cultivation process is not a normative process. Rather, it instills a designer with the sense, ability and experience to connect and correlate knowledge and ideas; to act and design creatively and imaginatively within the community. This is accomplished through openly and actively contributing knowledge and ideas to the OSSD community. An example of this behavior in OSSD are Blender project leaders [30] who regularly inform the community on the current project activities, plans, issues, etc. Therefore, the activities of a "cultivated designer" sustain or "cultivate" the open source paradigm itself.

In practice, through cultivation a designer becomes able to articulate design solution proposals and filter information in a way that makes sense to a coding-focused community. OSSD projects are often initiated and maintained by developers who cultivate a programming-focused culture that benefits programming efforts within the community. An "uncultured" designer's effort to contribute design solutions rarely adheres to the programming-focused culture. These efforts are perceived as being against the "fun of programming" and thus, are considered a "burden" and "destructive" to the OSSD project. A developer from the Drupal project describes this situation metaphorically in response to an "uncultured" design contributor:

> "You've come into our front room, and, while we were making a cup of tea, you moved all the furniture around. Not only that, but you redecorated, changed the carpet, and removed all of our belongings." [31]

4.3. Designer's Initiative

In comparison with the "designer's socialization" and "designer's cultivation" techniques mentioned before, the designer's initiative covers the widest scope of the three heuristic techniques. "Designer's initiative" refers to a designer's ability to initiate and maintain open design processes that realize the open source principles for the field of design. In other words, it refers to a designer's ability to design within the open source paradigm. Designer's initiative should be understood as a part of a designer's general strategy; a methodological practice that allows design processes to evolve and combine.

It is important to mention that, in the open source paradigm, the designers are free to approach the superabundance of information found across the OSSD projects and to

use it analytically and synthetically. Additionally, there is a strong bias for reusability and modularity of solutions. These two principles strongly imply that the open design process is inevitably a collaborative, non-linear, de-centralized process. "Designer's initiative" therefore aims at the designer's ability to realize and advance these principles through design practice and theory, ultimately contributing to the development of design philosophy and methodology more appropriate to open source paradigm.

5. Conclusions

In the open source, organizational structures necessary to support the designer's role do not exist and without them the conventional design methodologies fail. As a part of an ongoing effort to help explain, develop and verify design methods for open source production models online and off-line, this field research investigated the social circumstances in OSSD communities which influence the implementation of design solutions, aiming to explain the observed ineptitude of designers to contribute design solutions to OSSD projects. Using the grounded theory method, together with case studies and direct participation, six key categories of social circumstances influencing the implementation of design solutions in the OSSD projects were encoded and presented. There are three hindering categories and three facilitating categories of social circumstances. The way in which the open source functions is inextricable from the social conditions within it. Thus, the development of future design methodologies for the open source paradigm compels a deep understanding of the cultural framework of open source communities.

Working towards a future open source design methodology, the findings provided a basis to propose a model of adaptive behavior for designers in an open source community. This model aims to alleviate the effects of hindering social circumstances, while retaining the facilitating effects when designing in the open source paradigm. The model consists of three heuristic techniques, referred to respectively as: a) "Designer's socialization" – aimed at improving a designer's ability to collaborate on OSSD projects, b) "Designer's cultivation" – aimed at improving a designer's ability to contribute in OSSD communities and c) "Designer's initiative" – aimed at improving a designer's ability to design within the open source paradigm.

In conclusion, the designer's role in an open source community involves much more than the creative contribution of design solutions. The designer must also be an active member of the community and a catalyst for the wider implementation of design through the designer's initiative and the opening of the designer's own design process. Designers are compelled to understand and utilize social circumstances and social interaction when they are collaborating, contributing and designing with a community. In other words, a designer's mindset should shift from designing for a community, towards designing with and within a community.

6. References

Note: All web addresses were successfully accessed on 2013-06-20.

1. Open Source Initiative: *The Open source Definition,* retrieved from: http://opensource.org/osd
2. Peter Sikking: *"...at the moment there is no sense of fun, contribution and accomplishment in the transformation tools topic...."*. Retrieved from: https://mail.gnome.org/archives/gimp-developer-list/2013-February/msg00102.html
3. Yamauchi, Y., Yokozawa, M., Shinohara, T., & Ishida, T.: *Collaboration with Lean Media: how open-source software succeeds*, ACM Conference on Computer Supported Cooperative Work (pp. 329-338). ACM, 2000.
4. Ibid.
5. Matthew Thomas: *Why Free Software usability tends to suck*. Retrieved from: http://web.archive.org/web/20041117091141/http://mpt.phrasewise.com/discuss/msgReader$173, (2002-04-19)
6. Sack, W., Détienne, F., Ducheneaut, N., Burkhardt, J.-M., Mahendran, D., & Barcellini, F.: *A Methodological Framework for Socio-Cognitive Analyses of Collaborative Design of Open Source Software*, Computer Supported Cooperative Work, 15, 2-3, (J. M. Atkinson, Ed.) Cambridge University Press. Retrieved from http://arxiv.org/abs/cs/0703009, 2007
7. Based on paraphrase of the first chapter of *Mozilla Public License Version 2.0*. Retrieved from: http://www.mozilla.org/MPL/2.0/
8. GIMP project: GIMP *developer mailing list archive*. Retrieved from: https://mail.gnome.org/archives/gimp-developer-list/
9. Jim Michaels et al.: *suggestion for new versions of* GIMP. Retrieved from: https://mail.gnome.org/archives/gimp-developer-list/2011-November/msg00050.html
10. GiveLifeCS, gespertino: *givelife color system*. Retrieved from: https://mail.gnome.org/archives/gimp-developer-list/2011-December/msg00020.html
11. E. Miles: *An observation about Designers versus Developers* [blog post], Angry Donuts blog, 2009-08-28. Retrieved from: www.angrydonuts.com/an-observation-about-designers-versus-developers
12. Lakhani, K. R., & Von Hippel, E., *How open source software works: "free" user-to-user assistance*, Research Policy, 32 (6), (pp. 923-943). Retrieved from http://www.sciencedirect.com/science/article/pii/S0048733302000951, 2003
13. *Twitter Bootstrap project*. Retrieved from: https://github.com/twitter/bootstrap#readme
14. "Papercuts", *PaperCut*, [project wiki], Retrieved from: https://wiki.ubuntu.com/PaperCut, (2011-02-15)
15. Inkscape papercuts. Retrieved from: http://cuts.thinking-garment.com
16. Aleksandar Kovač et al.: *Them Cuts! Guides around page to* [web page]. Retrieved from: http://cuts.thinking-garment.com/144
17. Canonical: *What We Do Overview* [web page]. Retrieved from: http://www.canonical.com/about-canonical/overview
18. Ubuntu font family, *About* [web page]. Retrieved from: http://font.ubuntu.com/about/
19. Ubuntu: *Ubuntu Font Licence, Version 1.0* [web page]. retrieved from: http://font.ubuntu.com/licence/
20. Mozilla Foundation: *Our mission is to promote openness, innovation & opportunity on the Web* [web page]. Retrieved from: http://www.mozilla.org/en-US/mission/
21. *Songbird Music Player* [web page]. Retrieved from: http://www.getsongbird.com/desktop/
22. Android project documentation: *Frequently Asked Questions, Why is Google in charge of Android?* [web page]. Retrieved from: http://source.android.com/source/faqs.html
23. Open Source Initiative: *The Open Source Definition (Annotated)*. Retrieved from: http://opensource.org/osd-annotated
24. The GTK+ Team: *What is* GTK+*, and how can I use it?*. Retrieved from: http://www.gtk.org/
25. Ringce: ++*Crebits*. This content is accessible from: http://ringce.com/hyde
26. Tom Preston-Werner, Nick Quaranto, et al.: *Jekyll*. Retrieved from: https://github.com/mojombo/jekyll#readme
27. Moore, Fredrick T.: *Economies of Scale: Some statistical Evidence*. Quarterly Journal of Economics (MIT Press) 73 (2): 232–245 (May 1959)
28. John C. Panzar and Robert D. Willig: *Economies of Scope*, The American Economic Review Vol. 71, No. 2, Papers and Proceedings of the Ninety-Third Annual Meeting of the American Economic Association (May, 1981), pp. 268-272
29. Celestia Project: *Celestia/Development* [project wiki]. Retrieved from: http://en.wikibooks.org/wiki/Celestia/Development
30. Sergey Kurdakov on Feb 26, 2013; 8:12am: *Blender developer meeting notes, 24 February 2013*, retrieved from: http://blender.45788.x6.nabble.com/Blender-developer-meeting-notes-24-February-2013-td104759.html
31. Boulton, M., *Design in Open Source*, ch. "Designers vs Developers", para. 2., [online journal], retrieved from: www.markboulton.co.uk/journal/comments/design-in-open-source, (2009-08-30)