21ST INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED17
21-25 AUGUST 2017, THE UNIVERSITY OF BRITISH COLUMBIA, VANCOUVER, CANADA

TOWARDS A MODEL OF THE OPEN-DESIGN PROCESS: USING THE GROUNDED THEORY FOR MODELLING IMPLICIT DESIGN PROCESSES

Boisseau, Etienne; Bouchard, Carole; Omhover, Jean-François
Arts et Métiers ParisTech, France

Abstract

The open-source approach arose in the computer industry. It now also impacts physical goods. In order to benefit from alleged benefit of open-design for tangible artefact, it appears needed to model this design process. We thus studied the 12 projects that took part to the PoC21 innovation camp, and construct a model based on interviews of projects stakeholders we have conducted. To develop the model, we followed the methodology of the grounded theory. The two main contributions of this paper are: first, the model of the open-design; and second, the methodology we used to construct this model. We have found that it is possible to use the grounded theory framework for formalizing implicit processes. Our model depicts the open-design process as a process, which features resemble to both traditional product design processes, and open-source software development processes. However, it also presents specifics that are not shown in any of these models detailed in literature. This model can serve as a basic for future prescriptive research on open-design. Moreover, we recommend a grounded-theory based methodology to model undocumented and implicit design processes.

Keywords: Open-design, Grounded theory, Open source design, Process modelling, Research methodologies and methods

Contact:
Etienne Boisseau
Arts et Métiers ParisTech
Product Design and Innovation Laboratory
France
etienne.boisseau@ensam.eu

Please cite this paper as:
Surnames, Initials: *Title of paper*. In: Proceedings of the 21st International Conference on Engineering Design (ICED17), Vol. 2: Design Processes | Design Organisation and Management, Vancouver, Canada, 21.-25.08.2017.

1 BACKGROUND AND MOTIVATION

1.1 Designing: Why is it important to model how one designs?

Designing is an activity that has an effect on nearly every sphere of human life (Pahl, et al., 2007). Indeed, our world is constituted of products, which are man-made objects serving a purpose. All these products have been designed. Improving how one designs is critical for developing more efficiently better products. However, designing is a complex activity, each instance of which is unique: either the problem or context are changed, or the method taken is different. To improve designing, we need to understand how one designs, hence develop models of these designing instances, since one can hardly grasp every component, or activities occurring during the designing of a new artifact. These models represent this "*environment with greater or less veridicality and in greater or less detail, and consequently to reason about it*" (Simon, 1996, p. 22). The role of researchers is then to define success influencing factors, and to propose new tools and methods accordingly. Literature reports dozens of different models (Howard et al., 2008; Tomiyama, et al., 2009), which differ in terms of scope, aim, or purpose (Wynn and Clarkson, 2005). However, because designing is complex, there is no *"silver bullet method"* (ibid., p. 35) that would solve every (wicked) design problem.

1.2 Open: How does free software impact the product design process?

To understand *open*, as in "open-source" or "open-design", we need to understand where this concept comes from. It stemmed from the free-software movement, which arose in the 1970s in the USA and is notably personified by Richard Stallman. The free-software movement aims at taking back control over technology by granting end-users four freedoms: freedom to run a program without restriction (for every purpose); to study its functioning; to make and redistribute copies of it; and to improve the software and release these improvements (Stallman, 1985). This "*political*" movement is later generalized through a pragmatic offshoot: the open-source software movement. This approach focused on the practical consequences of open-source principles, rather than on related values. Enabling anyone to modify a software – which results in letting them take part to its design process – led to new practices and organizations (Raymond, 2001).

1.3 Open-Design: What stands where the open-approach meets product design?

In past decades, product manufacturing democratized, due to the digitalization of manufacturing processes, the development of low-cost manufacturing tools, and the development of new structures for producing (FabLabs, makerspaces, etc.). New stakeholders (i.e. the end-users) became interested in manufacturing and thus in designing. We similarly observed a digitalization and democratization of product design itself. With new stakeholders and a democratized process, new design practices arose – among them the open-design. Open-Design for tangible artifacts might have great potential. In the case of software, it indeed shows distinctive qualities (flexibility, auditability, accountability, stability) and led to great industrial successes (GNU/Linux, Apache, etc.). However, there are today only a few results on specific features of open-design processes, and no widespread dedicated design tools or methods.

2 STATE OF THE ART

We now present current models of designing and why they do not succeed in representing open-design instances. We also explain why we chose a grounded theory based approach to model open-design.

2.1 Design process: why one needs multiple specific models

Designing is part of a broader process that is product development (Ulrich, 2011). The latter correspond to the conceiving and delivering of an artifact, in order to address a currently unfulfilled need. The input of the development process is thus this need. The output is making a product (that answers this need) available on the market. This broader process can be divided into two sub-processes: first, product design, and second product manufacturing. Product design starts with the need. It occurs once, and ends with the unequivocal definition of the artifact that answer the targeted need (Cross, 2000). Then comes the product manufacturing that realizes (i.e. makes real) this product definition. This phase is iterated as

many times as artifacts are produced. Thus, the "designing process" aims to create a "solution definition", based on the "need" of an end-user.
Designing models can be characterized by three major criteria: the type of need addressed (input), the process itself, and of the product definition (output). The designing process is made up of three components: the method followed, the stakeholders taking part to the process, and the boundary objects used.
Regarding methods, models differ in their number of stages in the design process and their definitions. A collection of models with a comparison of their various stages can be found in Howard et al. (2008).
A characteristics of design models are stakeholders taking part in the process. Various specialists are listed in the literature: engineers, designers, ergonomists, etc. Each specialists has its own representation of the problem, and solves it in a specific way. At the interface between phases of the design process, knowledge and information are formalized and exchanged through boundary objects (Carlile, 2002). They are of different types: formal or not, digital or tangible. Lastly, Wynn and Clarkson (2005) distinguish two major types of designing processes models: procedural (or sequential) and dynamic ones. In procedural processes, steps are executed one after the other, when they are interconnected in dynamic ones. Thus, boundary objects are more formalized in procedural processes than in dynamic ones. In dynamic processes, activities are more connected one to the others, hence stakeholders forms multidisciplinary teams. At the opposite, boundaries between tasks in procedural processes lead to silo organizations.
However, software design introduced a new type of development approach: agile models. They consist of numerous iteration where a product definition is quickly defined, and incrementally refined by comparison with the targeted need. The objective here is to shift from an optimal solution to a responsive delivering of the solution (Nerur & Balijepally, 2007). Finally, with open-source software, another approach appeared: crowed-source design (von Krogh, 2003). The particularity of this development is that one uses contribution coming from a large number of persons who are not necessarily professionals. All these models and approaches, their common or distinctive features are synthesized in Table 1.

Table 1. Designing approaches

	Procedural	Dynamic	Agile	Crowd-sourced
Type	Designing model types		Development paradigms	
Need	Is scoped at the beginning of the project. Targeted need is not necessarily related to design team stakeholders			Evolves according to team members (and their priorities)
Method	Stage-Gate based	Concurrent accomplishment of each phase with feedback loops.	Frequent iterations of design sprints, aiming to incrementally improve the product definition	
Stakeholders	Specialized and silo organized	Multidisciplinary teams, with frequent informal interactions		Made of skilled members forming a core team, and numerous external contributors
Boundary objects	Formalized	Informal during design process	Formalized at the end of each design sprints	Fully digital, hosted on remote repositories, can be forked
Product definition	Formalized at the end of the process		Functionalities and quality increased after each iteration, or design sprint.	

These differences show that designing models should be adapted to depict distinctive features of specific designing instances, in order to help developing relevant tools and methods.

2.2 Open-Design: what it is and why it differs from traditional models

Warger defines open-source software as *"an approach to software development and intellectual property in which program code is available to all participants and can be modified by any of them"* (2002, p. 18). Via global digitalization and better communication means, the open approach spread over sectors outside software industry. Previous definitions of open-source software highlight two aspects in the open-source definition: the development process (here, of software), and the intellectual property. This makes the open-approach transferable to other sectors: art, data, law, governance, and hardware.

We thus use the following definition: *"Open means anyone can freely access, use, modify, and share for any purpose (subject, at most, to requirements that preserve provenance and openness)"* (Open Knowledge Foundation, 2015).
Based on this definition, we can highlight fundamental principles of open that are: the free access (technically and legally) to anyone, without any discrimination; the free use (and then the right to modify and redistribute - even commercially); and a potential limitation, in order to preserve the original work, and its open characteristics. These principles induce two aspects of open. First, the digital form of contents: to ensure the free access in practice, content must not be physically localized somewhere. It must thus be somehow digital. If hardware cannot be digital, its blueprint, electrical diagram, etc. can be so. Second, peer-to-peer collaboration: since everyone can access and (re-)use the content, a fostered consequence is that people (who are now peers) tend to join their efforts to fulfil their needs.

For product design, the process and the solution definition can be opened, but not the need. Indeed, the latter is independent of the design team. Thus, the openness of design can be assessed on two dimensions, as coined by Huizingh (2011) for open-innovation: the process, and the solution definition. Opening the process means interacting with persons outside of the design team, enabling others to fork our projects and/or to contribute to it. Opening the sources means sharing them online using a permissive licensing. Since design is a complex process, all of its facets might not be open: we thus consider the openness evaluation of the process and the outcome as a continuum (from not open to open), where traditional design lies where neither the process nor the outcome are open, and open-design where both are fully open. So we can define open-design as *"the theoretical state for a design project where both the process and the sources of its outputs are accessible and (re)usable, by anyone and for any purpose"*. This definition covers following aspects: Open-design is about both the process and the output. Pure open-design is an abstraction, since we do not think that full openness could be achieved in practice; it is thus a direction to pursue. Openness in open-design can be summarized as "accessible and (re)usable, by anyone and for any purpose". The definition applies to a design project (i.e. an instance of the product design process), because a process cannot be open *per se*; similarly, if two processes follow the same steps, one can be open when the other is not. Regarding the need addressed, no significant differences with traditional design processes have been observed. Balka et al. (2010) indeed report that *"open-design projects tackle both incremental improvements and radically new designs"*.
Regarding stages of the design process, Balka et al. (2009) found *"no formally distinguishable patterns"*. However, we can point out that new models for designing appeared in software industry: some designers switched from a *"cathedral"* to a *"bazaar"* (Raymond, 2001). In this later case, *"product development is organized as an evolutionary learning process that is driven by criticism and error correction and institutionalized as peer review"* (Raasch, 2011, p. 559). A major evolution is in the composition of the design team where contributors belong either to the *"core"* development team, or the *"periphery"* (*ibid.*). Traditional roles become blurred (Stappers et al.). Then, if boundary objects would appear crucial for collaboration, in practice, verbal communication is identified as a key component of successful projects (Filson and Rohrbacher, 2011). Bonvoisin and Boujut (2015) claim that new online collaborative platforms are needed to further foster the rise of open-design.

New stakeholders, with new roles, skills, and organization; lose coordination of participants' contributions; digitized boundary objects; etc. All these specific features make difficult to use existing models for representing open-design. It thus appears needed to model existing practices.

2.3 Design process modeling: what method to create a new model

Smith and Morrow (1999) detail criteria a model of the design process must meet: addressing an important managerial issue, enabling to make decision based on information that is available and accurate, reasonable assumptions and simplifications of the model, and being computationally tractable. However, the most notable point is the need for both *"academic- and practitioner-oriented components"* in these models. However, we found no consensus on how to scientifically build a design process model from scratch.
Nonetheless, the grounded theory is a method for experience based qualitative research (Strauss and Corbin, 1998), falling within the constructivist paradigm. It aims at constructing a new theory or model through the analysis of raw data, via a rigorous step-by-step formalization and abstraction of these data. These data are first divided into segments (chunks of data conveying a single meaning) that are coded

and later grouped into concepts and categories. Based on these categories, a model is constructed without any *a priori* knowledge about existing theories (Paillé, 1994). This whole process occurs iteratively, and enables to unveil valid models of implicit social constructs. We thus chose it as it appear as a relevant method to generate a model that fits practices when little information about the phenomenon it available.

3 METHODS: MODELLING OPEN-DESIGN THROUGH GROUNDED THEORY

3.1 Aim and global structure of the study

Our study aims at modelling the open-design process. Indeed, we saw that current models appear not to represent distinctive features of open-design. Developing a new model should then serve to develop appropriate tools and methods to make the most of this new approach of design.
For this, we first selected a homogeneous group of open-design projects. We interviewed their stakeholders about the project and their contribution (Phase 1). Then, taking a grounded theory approach, we constructed a model based on the qualitative analysis of these interviews (Phase 2).

3.2 Sample: Projects studied

We studied the 12 projects having been selected for the "PoC21" boot camp. PoC21 is an innovation camp, held in Paris area (France) in late summer 2015. It *"brought together 100+ makers, designers, engineers, scientists and geeks"* during five full-time weeks, aiming to boost the development of 12 projects that were at the "proof of concept" stage (e.g. an easy-to-build wind turbine made of reclaimed, a household energy monitoring system, a thermal energy producing solar concentrator, etc.). This way, projects selection have not been done by author (but by the PoC21 team), avoiding selection bias. Moreover, these projects cover a wide variety of applications and share 5 key criteria: proposing a concrete solution to the climate change issue, developing a hardware product, having already reached the prototyping phase, being open-source, and being able to take part to the innovation camp. They had been selected after a call for participation in early 2015 (Conrad and Wolf, 2016).

3.3 Phase 1: Data collection

Once projects were selected, we conducted 30 to 45 minutes interviews, each with one stakeholder of one project. We chose one stakeholder per project only for practical reasons (availability of participants), and for not overweighting projects having more stakeholders. Language talked was English, unless interviewee chose French. Two interviews have been conducted physically, the others online.
We asked semi-directive questions, based on the 3 major criteria defining the design process,: what are the steps or phases followed during the design process (process), who took part to this process (stakeholders), and what were information exchanged (boundary objects). We also asked them about solution definition: how was released the product.
These interviews were audio recorded with authorization of interviewee. We then transcripted interviews using the *Sonal* software. We chose naturalized transcripts (Schegloff, 1997) – smoothing however involuntary vocalizations, pauses, and recurring non-verbal language idiosyncrasies.

3.4 Phase 2: Data analysis

Then, we divided transcripts into "segments", that are excerpt of verbatim in which one single idea only is expressed. Segments contains from couple of words to a few sentences.
These segments were then coded using pluri-nominal categorization. That is, we attributed one major "code", as well as one or several minor codes, to each segment. Codes symbolize the idea expressed. They consist of a few words. Major codes describe the general idea expressed ("product development process" in the example above), when minor ones define the singularity of the verbatim regarding the major concepts ("no method followed"). We used the *Sonal* software for that.
Then, we assigned a "concept" to each segment. These concepts are more abstract ideas gathering several codes. Therefore, we extracted a list each single major concepts used. Then, we assigned each major code to one or several concepts (that way, assigning categories to every code having been tagged with this major code). Concepts have an internal consistency: Codes were created without restriction (i.e. we looked at the segments' verbatim, and created concepts accordingly). However, we tried to minimize the number of concepts by merging little represented concepts into more generic ones.

Then, we clustered concepts into categories. The latter are even more abstract classes made up of several concepts. These categories were described. For this, we looked at concepts and codes belonging to this category (and their verbatim), and we abstracted their recurrent specific features

Based on these categories, we built our model. The model aims to describe common specific and iconic features of the different designing processes followed by the 12 projects. For that, we summarized recurrent patterns found in categories' descriptions, notably focusing on "Process" and "Boundary objects". We then synthetized these results in the model presented below.

All these stages, from data collection to model definition were conducted concurrently and iteratively. Of course, we built categories only after having conducted interviews, but we began to cluster concepts after a few interviews only. Without changing the scope of following interviews, it narrowed down specific points to explicit. Similarly, looking at concepts within a category made us revise some major codes and their categorization.

4 RESULTS: FROM POC21 PROJECTS TO CODES' CATEGORIZATION

4.1 Raw data, and database

We have conducted 11 interviews from stakeholders of 11 (out of 12) different project. We failed in contacting one participant, who conducted alone his project. All interviewed person attempt to the PoC21 innovation camp. Interviews were conducted between May and December 2016. Recordings add up to 8h19 (min: 28min/ mean: 46 min/max: 1h09), what equals to 58016 words of transcription (min: 1629/ mean: 5274/max: 9569).

These verbatims were divided into a total of 1072 codes (min: 66/ mean: 98/max: 135).

4.2 Concepts, Categories and Clusters

Each segment is tagged by at least two codes (a major and a minor one). 1069 single codes have been used, 347 of them being a major code at least once (in some rare cases, a code can be used as major for one segment and as minor for another).

These codes have been grouped into 41 concepts. Each concept represents from 3 to 41 major codes (mean: 10). There are from 3 to 91 (mean 28) segments gathered into a single category.

The software *Sonal* were used for the coding of segments. However we later used an *R* database. We wrote a Python script to interconnect the R database with Sonal .Rtr files. Finally, these concepts were clustered into 7 categories, as depicted in Table 2.

Table 2. Cluster, categories, and number of major codes

Category	**# of segments**	**Concept (# of segments)**
Human	361	core team (91), contributors (81), skills (68), human interactions (66), management (26), collaboration (15), geographical location (6), project nebula (4), communication (4)
Process	294	development process (82), design process (63), start (39), design process phase (39), design (19), iteration (19), tools (14), development process phase (10), time (9)
Project	158	issues (40), project (37), objective (27), motivation (21), constraint (16), business (13), strategy (4)
Boundary objects	148	boundary objects (43), sources (27), 3D (22), contribution (17), prototyping (16), inputs (15), documenting (8)
Open	102	open (62), PoC (40),
Product	28	product (16), hardware (6), user (3), digital (3)
N/A	323	questions (165), intro and closing (108), interruption (50)

5 ANALYSIS: CONSTRUCTING A MODEL BASED ON CATEGORIES

5.1 Model presentation

Once categories defined, we described and characterized those using concepts and codes it contains – as the last step of the grounded theory. We abstracted recurrent and distinctive patterns emerging from interviews. Based on these descriptions, we built our model of the Open-Design process depicted in Figure 1. It is composed of seven major steps with two feedback loops. Main activities are similar to the ones found in traditional models. The singularity is that steps 2 to 6 have no clear boundaries: they are conducted concurrently, with frequent informal iteration and feedbacks. The internal feedback loop occurs within the project core team (composed of long-term participants to the project): design activities are iterated until a satisfying point. Note that occasional external contributions can occur, notably during the design phase. Once a sufficient functionality level reached, the design is frozen and detailed sources (including building instructions) are detailed and publicly released. A new iteration begins, resetting the objective of the new development cycle (including new functions, or increasing success criteria).

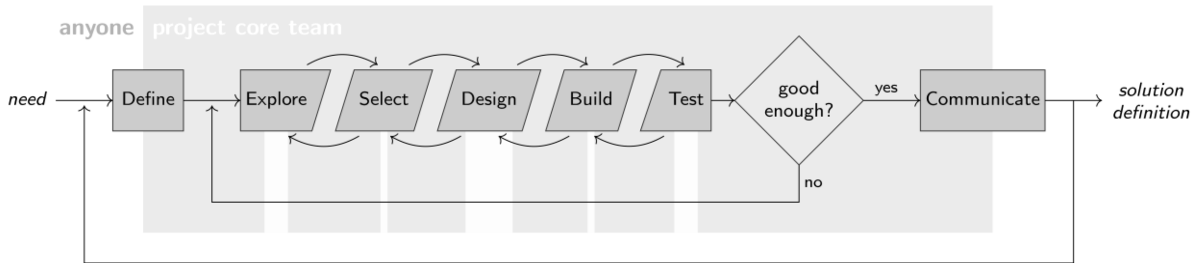

Figure 1. Open-Design process model

The seven steps constituting our model and in-between boundary objects are the following.
Need: As detailed above, the design process is based on an initial need. In the case of open-design, this need is often personal to project initiators.
Define: The purpose of this phase is to define and select functions to be implemented in the developed product. The first iteration is often a proof of concept. During this step, the functions to be embodied are selected among those the ideal artifact should do. They increase in complexity and integrate auxiliary functions after each iteration.
Requirements: This list of function is often tacit and made of expected behaviors. Two projects only explicitly specified requirements.

1. *Explore*: During this step, stakeholders look for already existing systems, which can fulfil one of several of the listed functions. This inventory is not exhaustive. This step may also be skipped because of a priori from the design team (and might be challenged after prototype tests).
2. *Select*: Among the inventoried existing systems, one is chosen in order to be implemented. Main selection criteria are time, cost, and available know-how.
3. *Design*: Before prototyping, participants defines the global architecture of the product, as well as where and how to implement chosen system into the existing state of the project. Little CAD tools are use, even if the system is later 3D-modeled for allowing CNC-manufacturing of parts.
4. *Build*: The system is implemented in a prototype. There is only one prototype built at a given time. For incremental iteration, the prototype is modified. In case of a major iteration, a new prototype might be created.
5. *Test*: The prototype is tested in order to check the successful implementation of the subsystem. However, test are little formalized and not compared to expected performances.

Go-No Go: If test succeed (in all but one project, neither requirement nor success criteria were not explicitly defined; in this case, criterion is "does is works well enough?"), design is frozen, and the team prepare a public release. Otherwise, a new iteration starts at step

6. *Communicate*: Once the design frozen, sources are edited and proper documentation is created. They are then broadcasted outside of the design team via dedicated platforms and project's website.

Solution definition: The release contains both the sources of the project (3D model, 2D technical drawings, CNC digital files), as well as building instructions (text, pictures, and/or videos).

5.2 Comparison with existing models

The model we constructed shows similarities to traditional models of the design process depicted in the literature, such as Pahl and Beitz's model (2007): (*"clarification of the task"*: step 1; *"conceptual design"* to *"embodiment design"*: steps 2 to 6; *"detail design"*: step 7). Similarly, we find steps present in Cross' model (2000): The macro feedback loop serves to *identify opportunities* by including new functions to embody, or new use-cases to adapt to. In Step 1 ("Define"), we can observe similarities with stages *clarifying objectives*, *establishing functions* and *setting requirements* of Cross' model. "Explore" stands for *generating alternatives*, and "Select" for *evaluating alternatives*. However, if the global process of open-design is coherent with existing design methodologies (that argue for Open-Design being an adaptation of traditional processes to specific requirement and context, rather than a brand new way to design), we still observe differences with above mentioned models: frequent iteration cycles, and low formalization of boundary objects. Moreover, the late formalization of the design, its broadcast outside the design team ("Communicate") and the content of these sources challenges traditional models.
Regarding traditional process, we observe an overweighting of generative phases (i.e. "doing the design"), compared to planning ones ("preparing and setting boundaries to generative phases"). Quality is thus reached via an increase of design cycle iterations, rather than in efficiency of initial design process. This might be due to the lack of design methodologies mastered by project stakeholders. This feature is also distinctive of agile methodologies, notably observed in software development.
Like in open-source software design, we also observe crowd-sourced contributions. There still is a core design team responsible for strategically decisions. Participants of design activities might however not belong to this team. The main steps are nevertheless coordinated by major participants. The model we constructed is indeed made up of two feedback loops: the first one, aiming at improving current system, is internal to the project team. Then, if the prototype is not validated, a new internal iteration starts.
Once sufficient quality is reached, the design team prepare a release of the product by freezing design and formalizing product sources, as well as auxiliary documents like building instructions, bill of material, 3D rendering, etc. Once done, these sources are publicly broadcasted and a new design cycle starts, with a change in design objectives (more functions to embody, or higher success criteria). Since sources have been publicly released in-between, we consider it as an "external" iteration cycle.
These feedback loops resemble the agile development methodology, notably found in open-source software development, with "design sprints" and frequent checks to verify if design objective are met (and changes of the latter, regarding current state of the project). Some specificities of this process also reminds to open-source software development: the initial formation of a team on a common shared idea, relationships with contributors not regulated by formal contracts (von Krogh, 2003). Specificities of open-source software development highlighted by Mockus et al. (2002): implicit mechanism with low specification and formalization of interfaces between design steps (favoring internal informal communication), a team of a few persistent and skilled members forming the core team of the project, as well a release-based development scheme, with internal pre-release activities.
However, we can highlight two specific features of the open-design model: First, we have observed that only one prototype is developed at time. This is due to both the rival nature of atoms (and hence products, compared to atoms) and its non-zero marginal cost: one cannot duplicate an object for free, and one product can be in only one location at time. Thus, it is costly and hard to maintain several prototype at the same state in different location for the development team. Since all projects we observed are small scaled (1 to 5 core members), they all developed only one prototype at time. The reason for a new prototype to be built, is that the improvement will not be incremental, and thus, deep changes will be made – what justifies to build a new one from scratch.
Since only one prototype is built, we observe that – unlike in the development of open-source software – the geographical location of core team members matters. In other words, for most projects, all team members are located in the same areas, or at least geographically gather for design activities - unless design activities are delegated to one person only, or when the development is made digitally only. One member expressed that living in a country far from the center of Europe was a hurdle to the development of its project.

6 DISCUSSION

We have presented a model of the open-design process for the development of tangible product, constructed using the grounded theory framework. This model does not radically differs from traditional

ones. Elements from both agile methodologies and crowed-sourced approaches can however be also found in this model: notably both feedback loops and a constant redefinition of goals, as well as contributions made by persons outside the design team. We also find traits due to amateur design, such as poorly structured phases and detailed boundary objects.
The most distinctive feature of this model is the public release and broadcast of detailed sources, which are edited once the design is validated.
We must however acknowledge limitations of our results. They can be assessed on three different levels: first, the raw data used, then the process for constructing a model based on these data, and, lastly, the significance of our model. Regarding data used, we recognize that only one stakeholder only per project has been interviewed. This have been done in order to give equal weight to each project, and for practical reasons. However, our results could have been verified by interviewing other stakeholders; but we then should circular reasoning. We should also note that even if the pool of project we choose is homogeneous, it consequently do not represent the whole variety of open-design projects. For what concerns the process, its interest and quality is ensured by the rigorous following of the grounded theory methodology. A refinement in the methodology would have been to assign categories to each code manually, rather that automatically. Due to time constraint, we chose to assign the same category for every codes sharing a same major concept. However, the robustness of the process have been ensured by the numerous iteration of coding and memoing, in order to obtain homogeneous categories. Finally, the significance of the model still need to be assessed by field study. This would enable us to confirm if its scope of application can be generalized to all open-design processes.
This model appear thus as a starting point for improving and increasing industrial implementability of open-design, by providing insights on its distinctive features for providing adequate tools and methods.

7 CONCLUSION

Open-Design is a new approach of designing rooted in the open-source movement. This new practice appear not to fit with existing design models. To make the most of open-design for tangible artifacts, it appeared needed to model this design process. We thus studied the 12 projects that took part to the PoC21 innovation camp, and construct a model based on interviews of projects' stakeholders we have conducted. To develop the model, we followed the methodology of the grounded theory.
The main contribution of this paper is the model of the open-design process presented in Figure 5. Another originality of our research is the use of the grounded theory to model a design process. We have found that it is possible to use such approach to formalize implicit processes, as shown by the model produced and its internal robustness. For future research, we thus recommend a grounded-theory based methodology to model undocumented and implicit design processes.
Our model depicts the open-design process as a process, which features resemble to both traditional product design processes, and open-source software development processes. However, it also present distinctive features rooted in agile and crowed-sourced approaches. Its accuracy and range of application will now be tested on industrial cases in various sectors, in a positivist approach. This should reinforce its significance, and enable future researchers to develop new tools and methods in order to increase efficiency and industry-compliancy of open-design based processes.

REFERENCES

Balka, K., Raasch, C., & Herstatt, C. (2009), Open source enters the world of atoms: A statistical analysis of open design. *First Monday, 14*. doi:10.5210/fm.v14i11.2670
Balka, K., Raasch, C., & Herstatt, C. (2010), How Open is Open Source? – Software and Beyond. *Creativity and Innovation Management, 19*, 248-256. doi:10.1111/j.1467-8691.2010.00569.x
Bonvoisin, J., & Boujut, J.-F. (2015), Open design platform for open source product development: current state and requirements. In: C. Weber, et al. (Eds.), *Proceedings of the 20th International Conference on Engineering Design, 8 - Innovation and Creativity*, pp. 11-20. Milano.
Carlile, P. R. (2002), A pragmatic view of knowledge and boundaries: Boundary objects in new product development. *Organization science, 13*(4), 442-455.
Charmaz, K. (2003), Grounded Theory. In: *The SAGE Encyclopedia of Social Science Research Methods.* SAGE Publications.
Conrad, A., & Wolf, C. (2016), *We Are All Crew.* Open State and OuiShare.
Cross, N. (2000), *Engineering Design Methods* (3rd ed.). John Wiley & Sons, Ltd.

Cross, N. (2001), Designerly ways of knowing: design discipline versus design science. *Design Issues*, *17*, 49-55. doi:10.1162/074793601750357196

Feller, J., & Fitzgerald, B. (2002), *Understanding open source software development.* Addison-Wesley.

Filson, A., & Rohrbacher, G. (2011), Design offered up: control and open outcomes in a digitally enabled design process. In Y. Luo (Ed.), *Cooperative Design, Visualization, and Engineering* (pp. 7-13). Springer. doi:10.1007/978-3-642-23734-8

Glaser, B., & Strauss, A. (1967), The Discovery Of Grounded Theory ; Strategies for Qualitative Research. Aldine de Gruyter.

Howard, T. J., Culley, S. J., & Dekoninck, E. (2008), Describing the creative design process by the integration of engineering design and cognitive psychology literature . *Design Studies , 29*, 160-180. doi:10.1016/j.destud.2008.01.001

Huizingh, E. K. (2011), Open innovation: State of the art and future perspectives . *Technovation , 31*, 2-9. doi:10.1016/j.technovation.2010.10.002

Initiative, O. S. (Éd.). (2015, 1), Open Source Definition. Retrieved from http://opensource.org/osd

Krause, F.-L., Kind, C., & Voigtsberger, J. (2004), Adaptive Modelling and Simulation of Product Development Processes. *CIRP Annals - Manufacturing Technology, 53*(1), 135–138.

Mockus, A., Fielding, R., & Herbsleb, J. (2002), Two Case Studies of Open Source Software Development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology, 11*(3), 309–346.

Nerur, S., & Balijepally, V. (2007), Theoretical Reflections on agile development methodologies. *Communications of the ACM, 50*(3), 79-83.

Ong, C.-A. (2004, 5), (Re-)Integration Dynamics of the PC Platform. Master's thesis, MIT.

Open Knowledge Foundation. (2015), The Open Definition. Retrieved from http://opendefinition.org/

Open Source Initiative. (2006), FAQ. http://www.opensource.org/advocacy/faq.html

Pahl, G., Beitz, W., Feldhusen, J., & Grote, K.-H. (2007), *Engineering Design* (éd. 3). Springer.

Paillé, P. (1994), L'analyse par théorisation ancrée. *Cahiers de recherche sociologique*, 147-181. doi:10.7202/1002253ar

Raasch, C. (2011), Product Development in Open Design Communities: A process perspective. *International Journal of Innovation and Technology Management, 8*, 557-575. doi:10.1142/S021987701100260X

Raymond, E. S. (2001), *The Cathedral & the Bazaar.* O'Reilly Media.

Schegloff, E. A. (1997), Whose Text? Whose Context? *Discourse and Society, 8*(2), 165-187.

Simon, H. A. (1996), *The Sciences of the Artificial* (3rd ed). Cambridge, MA: MIT Press.

Smith, R. P., & Morrow, J. A. (1999), Product development process modeling. *Design Studies, 20*(3), 237–261.

Stallman, R. M. (1985), The GNU Manifesto. *Dr. Dobb's Journal of Software Tools, 10*.

Stallman, R. M. (2008), Re: MAINTAINERS file. Récupéré sur http://lists.gnu.org/archive/html/emacs-devel/2008-03/msg00635.html

Stallman, R. M., & Williams, S. (2010), *Free as in Freedom (2.0): Richard Stallman and the Free Software Revolution.* Free Software Foundation. Retrieved from http://www.fsf.org/faif

Stappers, P. J., Visser, F. S., & Kistemaker, S. (s.d.). Creation & Co: User participation in design.

Strauss, A., & Corbin, J. (1998), Basics of qualitative research: Techniques and procedures for developing grounded theory. Sage Publications.

Tomiyama, T., Gu, P., Jin, Y., Lutters, D., Kind, C., & Kimura, F. (2009), Design methodologies: Industrial and educational applications . *CIRP Annals - Manufacturing Technology, 58*, 543-565. doi:10.1016/j.cirp.2009.09.003

Troxler, P. (s.d.). Libraries of the peer production era.

Ulrich, K. T. (2011), *Design : Creation of Artifacts in Society* (éd. 1). University of Pennsylvania.

von Krogh, G. (2003), Open-Source Software Development. *MIT Sloan Managment Review*, 14-18.

Warger, T. (2002), The Open-Source Movement. *The Edutech Report, 18*.

Warsta, J., & Abrahamsson, P. (2003), Is Open Source Software Development Essentially an Agile Method? *Proceedings of the 3rd Workshop on Open Source Software Engineering*, 143-147.

Wynn, D., & Clarkson, J. (2005), Models of designing. In *Design process improvement* (34-59). Springer.

Yazdani, B., & Holmes, C. (1999), Four Models of Design Definition: Sequential, Design Centered, Concurrent and Dynamic. *Journal of Engineering Design, 10*, 25-37. doi:10.1080/095448299261407