A Model of Open Source Developer Foundations

Dirk Riehle, Sebastian Berschneider

A Model of Open Source Developer Foundations

Dirk Riehle[1] and Sebastian Berschneider[1]
[1] Friedrich-Alexander University Erlangen-Nürnberg, Computer Science Department, Martensstr. 3, 90158 Erlangen, Germany,
dirk@riehle.org, http://osr.cs.fau.de

Abstract. Many community open source projects are of high economic relevance. As these projects mature, their leaders face a choice of continuing the project as is, making the project join an existing foundation, or creating their own foundation for the project. This article presents a model of open source developer foundations that project leaders can use to compare existing foundations with their needs or to design their own. The model is based on a three-iteration qualitative study involving interviews and supplementary materials review. To demonstrate its usefulness, we apply the model to nine foundations and present their organizational choices in a comparative table format.

1 1 Introduction

Community open source projects are open source software projects that are owned by a community of stakeholders [10]. The Linux kernel, the Apache projects, and the Eclipse platform are examples of community open source software. Community owned projects are to be viewed in contrast to single-vendor open source projects like MySQL, SugarCRM, or Alfresco [8] that are owned and managed by a single stakeholder, the company. As community open source projects mature and become increasingly economically relevant, they face a choice: Continue as is, seek the protection of an existing open source foundation, or create their own foundation.

The benefits of an open source foundation for the projects are manifold [9]. Most notably, the foundation acts as the legal representative of the projects. Thus, it can represent the projects' interests in court and protect the individual contributing software developers. Also, a foundation increases the projects' credibility and makes them less dependent on individual people, which in turn increases industry involvement and makes the projects more sustainable. Thus, the creation of a foundation is a natural step in the life-cycle of successful maturing community open source projects.

Open source foundations are similar to traditional consortia. In fact, many foundations chose to incorporate using legal forms typically chosen for consortia. The main difference is the choice of an open source license for the software being developed. By using an open source license, a foundation cannot exclude non-members from utilizing the software. Thus, the software is typically not considered competitively differentiating. This is most forcefully demonstrated by some foundations choosing a for-public-benefit organizational form rather than the for-member-benefit form that traditional consortia typically take.

Today, dozens if not hundreds of such open source foundations exist. Next to large well-known foundations like the Linux Foundation, Apache Software Foundation, or the Eclipse Foundation, many smaller niche foundations exist, catering to the specific needs of their projects. These specific needs may be country, culture, or jurisdiction-specific (e.g. OpenPlans, Kuali, or WorldVistA), or they may be specific to a particular industry (e.g. TOPCASED, GENIVI, OpenAPC), or they may be specific to a particular horizontal layer in the technology stack (e.g. Drupal Foundation, Document Foundation, KDE e.V.) or any combination thereof.

Joining an existing foundation or creating a new one is a daunting task. First, a project considering joining an existing foundation has to ask itself whether the foundation matches the project's interests. Second, if the answer is no, the project has to consider creating its own foundation. For these processes, there is no guidance, yet the compatibility of a foundation with the projects' needs is a crucial factor in ensuring the project's sustainability.

In our analysis of open source foundations, we need to distinguish between developer foundations, created by software development firms, and IT user foundations, created by users of the software who are using open source to avoid vendor-lock-in and keep prices down. This paper is only about developer foundations.

This paper presents a qualitative analysis of the structure and processes of existing open source developer foundations. A three-step process of analyzing existing foundations led to a model of open source foundations that project leaders can use to compare existing foundations with their needs or design their own. The paper's resulting contributions are:

- A model of community open source developer foundations that has been derived from a three-step qualitative study process
- The comparison of nine open source foundations using this model and demonstrating its ability to capture the complexity of existing foundations

The paper is structured as follows. Section 2 outlines our research process. Section 3 presents a model of community open source developer foundations. Section 4 applies the model to nine existing open source foundations and demonstrates its usefulness. Section 5 reviews related work and section 6 concludes the paper.

2 Research Process

This paper shows the result of an exploratory theory generation process for developer foundations; it does not yet quantitatively validate this theory. We use the terms "theory" and "model" of developer foundations interchangeably in this article. The theory was generated in a three-step iterative process. The first step created the first model, the second and third step created refined models based on additional qualitative research. The first and second step were performed by the second author under the guidance of the first author, the third step was performed by the first author alone. The second author is a student of the first author.

2.1 First Iteration: Initial Model

For the first iteration, the first author chose six different foundations for initial analysis. These six foundations were the Free Software Foundation, the Linux Foundation, the Apache Software Foundation, the Eclipse Foundation, the GENIVI consortium, and the Albatross community. These foundations were chosen for their breadth and maturity. The second author then gathered and analyzed these foundations, using the following data:

- Telephone interviews with foundation representatives
- Email exchanges to clarify issues from the interviews
- By-laws and other materials from the foundations' websites

The second author analyzed the materials using an open coding approach. He grouped the codes into initial categories that then became the building blocks of the first model [4]. This led to the definition of sub-categories like "membership" and their possible values like "person" or "corporation".

2.2 Second Iteration: Qualitative Validation

For the second iteration, the first author provided four more foundations to the second author that were supposed to serve as a qualitative validation of the first-iteration model: Would the initial model be able to capture the complexity of these four new foundations? The four new foundations were the KDE, Mozilla, OpenAPC, and TOPCASED foundations.

The attempt to describe these four new foundations using the initial foundation model surfaced several problems with the initial model. For example, the initial model had not captured the possibility of financing a foundation through a for-profit subsidiary (Mozilla). Thus, the result of the second iteration was a revised model that enhanced several of the original model dimensions and added one more.

2.3 Third iteration: Categorization and Review

For the third iteration, the first author took over and created a stringent categorization hierarchy from the available data. For example, the rather broad single category Intellectual Property was split into subcategories Project License, Patent License Grant, and IP Ownership. The accompanying attribute values were rearranged accordingly and in-line with the original coding. The Albatross community was recognized as single-vendor-owned open source rather than community open source and consequently removed.

The resulting hierarchy of concepts and categories is called category, attribute, and possible value in Tables 1 and 2, the final analysis result after the third iteration. We found this nomenclature to be more useful than the general terminology of categories and sub-categories. The resulting model was presented to the original corre-

spondents of the second author from the first iteration of this research process. While not a representative survey, the generally positive feedback confirmed the quality of the model. A few minor corrections, mostly renaming of codes, was applied to the model. The result is the model presented in Section 3.

3 Model of Developer Foundations

3.1 Model overview

Tables 1 and 2 show the final result of the combined three-step exploratory analysis using interviews and supplementary web and literature research. It presents the model as a stringent category hierarchy with

- column 1, "category", being the top category,
- column 2, "attribute", being the first layer of sub-categories, and
- column 3, "possible values", being a third layer of categories.

Column 4 contains short codes for the possible values provided in column 3. Column 5 describes constraints between the categories, providing a short form of capturing category interactions. Column 6 then provides some explanation of the meaning of the attribute (the first level of sub-categories).

The following Subsections explain the model categories.

3.2 The General model category

The **General:Purpose** of a developer foundation is to benefit the public (G:PB) or its members (G:MB). Member benefit does not exclude public benefit, however, it is not a primary goal of the foundation then. This choice of public vs. member benefit strongly influences how the foundation incorporates.

The **General:Incorporation** of a developer foundation is typically as a non-profit foundation (G:F) or a non-profit consortium (G:C). The difference is that a foundation serves the public and a consortium serves its members. The actual choice of the corporate form depends on the jurisdiction of incorporation. In the U.S.A., for example, the Apache Software Foundation is a 501(c)3 non-profit foundation, while the Eclipse Foundation is a 501(c)6 non-profit consortium. In Germany, they might have chosen the form of an e.V. or a GmbH. Most countries have forms of incorporation that support for-public-benefit or for-member-benefit organizations.

The **General:Members** of a developer foundation can be both natural persons (G:NP) or juristic persons (G:JP). Natural persons are people, and juristic persons are incorporated organizations. Any combination is possible, as are many restrictions. For example, the Mozilla Foundation has no members, the Free Software Foundation only has natural people (developers) as members, and the Linux Foundation has both natural and juristic persons as members.

3.3 The Philosophy model category

The **Philosophy:Commercial Stance** of a developer foundation can be that of a free-software-enforced (P:FSE) or closed-software-allowed (P:CSA) stance. Taking a free-software-enforced stance prevents taking enhancements to the open source software private, thereby limiting the business models possible with the software. Taking a closed-software-allowed stance allows a much broader range of business models, including those in which the open source software is enhanced without those enhancements being open sourced as well. The stance strongly determines the choice of a license.

This is an abbreviation of a more complex discussion carried out elsewhere in more detail [11]. The Free Software Foundation and the KDE e.V. are examples of foundations that take the more restrictive free-software-enforced stance, while the Apache Software Foundation and the Eclipse Foundation are examples of foundations that take the more permissive closed-software-allowed stance. In general, the more commercially minded, the more likely a permissive stance.

The **Philosophy:Development Model** of a foundation can be open (P:OD) or closed (P:CD). In the case of open development, all or most of the development artifacts are available in public as they are being developed. In the case of closed development, the project communities keep the project artifacts to themselves and only release them to the public in major increments, if they release them at all. Legally, in the restrictive sense of a license choice, the software may still be open source, though. Closed development is fairly uncommon but it does happen; an example is GENIVI, a consortium developing a Linux-based stack for in-vehicle infotainment.

3.4 The Intellectual Property model category

The **Intellectual Property:Project License** that a foundation chooses can be any of the 50+ open source licenses that have been officially accepted by the Open Source Initiative (OSI) as such [6]. Sometimes, foundations develop their own license, which then has to pass the OSI's license review process. Simplifying, we reduce the choice of license to either a reciprocal (e.g. GPL or AGPL) (IP:RL) or a permissive license (e.g. BSD or Apache license) (IP:PL).

- A reciprocal license forces software developers who are embedding the reciprocally-licensed software in a product to open source the embedding code when distributing the product.
- A permissive license does not force the developer to open source the embedding software code when distributing the software.

There are many options, and this simplification does not do justice to the wealth of choices one faces in choosing or designing a license. German et al. discuss some these issues and their implications [5]. In general, a free-software-enforced stance (P:FSE) goes well with a reciprocal license, and an closed-source-allowed stance (P:CSA) goes well with a permissive license.

A foundation has the option of providing the same source code under multiple licenses to increase flexibility. Contributors typically are required to accept all licenses, as enforced through a contributor agreement, see below. The Mozilla Foundation, bundles three licenses (the foundation's MPL, the LGPL, and the GPL) into its so-called tri-license. Derivative projects can then chose which license to utilize.

Table 1: Model of Open Source Developer Foundations, Part 1

Model of Open Source Developer Foundations 1 / 2					
Category	Attribute	Possible Values	Value Short Code	Constraints	Comments
General	Purpose	• Public benefit • Member benefit	PB MB	Single choice	Implies form of incorporation
	Incorporation	• Foundation • Consortium	F C	Single choice	E.g. 501(c)3 vs. 501(c)6 (United States) or e.V. vs. GmbH (Germany)
	Members	• Natural persons • Juristic persons	NP JP	Multiple choice, including none	Any possible number of restrictions
Philosophy	Commercial stance	• Free software enforced • Closed software allowed	FSE CSA	Single choice	Implies license choice, including patent grant regulation
	Development model	• Open development • Closed development	OD CD	Single choice	
Intellectual Property (IP)	Project license	• Reciprocal • Permissive	RL PL	Multiple choice; at least one	May or may not use their own license; may provide multiple licenses
	Patent license grant	• No grant • Grant for use • Grant for use + embedding	NPG UPG EPG	Single choice	Usually regulated through project license
	IP ownership	• No IP ownership required • Relicensing rights grant • Copyright assignment	NIP RRG CA	Single choice	Realized through contributor agreement

Continued in Table 2

The **Intellectual Property:Patent License Grant** can be none (IP:NPG), for-use (IP:UPG), or for-use and embedding (IP:EPG).

- If no patent license grant is provided through the project license or a contributor agreement (see IP ownership below), then users may have to pay

Table 2: Model of Open Source Developer Foundations, Part 2

Model of Open Source Developer Foundations 2 / 2					
Category	Attribute	Possible Values	Value Short Code	Constraints	Comments
Governance	Board membership	• Democratic (elected) • Meritocratic (earned) • Autocratic (self-appointed)	DB MB AB	Single choice	Actual value set can be much larger
	Project membership	• Democratic (elected) • Meritocratic (earned) • Autocratic (appointed)	DP MP AP	Single choice	Actual value set can be much larger
	Natural member career	• User/developer/committer • Project manager/leader • Officer, board member	UDC PMC BRD	Multiple choice, including none	Typically applies to natural persons only; actual career steps can be much more fine-grain
	Juristic member level	• Financing level • Developer level	FL DL	Multi-dimensional choice	Typically applies to juristic persons only
Financing	Foundation	• Membership fees • Sponsorship • Gifts and grants • For-profit subsidiary	MF S GG FPS	Multiple choice, at least one	
	Projects	• Foundation • Members	PFF PMF	Multiple choice, at least one	Typically only single choice though
Operations	Infrastructure	• Foundation • Members	IFO IMO	Multiple choice, at least one	
	Back office	• Volunteers • Employees	BOV BOE	Multiple choice, at least one	

royalties to the patent holder. This option is uncommon, because one of the reasons for creating a foundation is to avoid such legal uncertainty.

- If a patent license grant is provided for-use, then the software can be used without paying royalties under the original open source license. A software developer who wants to sell software that embeds the open source code under a different license may have to pay the original patent holder royalties.
- If a patent license grant is provided for use and embedding, a software developer can embed the open source code in a commercial product and sell the software under a commercial license without having to pay royalties to the patent holder.

The **Intellectual Property:IP Ownership** determines the rights the foundation wants to acquire of the software code. It may be none (IP:NIP), a relicensing right (IP:RRG), or the full copyright (IP:CA). Some foundations do not require to receive any rights but rely on the project license to regulate intellectual property issues around the project. An example is the KDE e.V. (foundation), however, this is the more uncommon choice. (KDE offers a transfer but does not require it.) Most foundations ask contributors to sign a contributor agreement before they make their first contribution, which either provides the foundation with a relicensing right or directly signs over the copyright to the foundation.

- A relicensing rights grant asks a contributor to provide the foundation with the right to relicense the contributed software code. This is helpful should the foundation decide to change its license at a later point of time. It also provides the foundation control over the whole code base, because the foundation becomes the single and only holder of a complete relicensing right while each contributor holds rights only to their typically small piece.
- A copyright assignment asks a contributor to transfer all rights to the foundation, which then becomes the sole owner of the source code. This includes all the rights from the relicensing rights grant. In addition, it allows the foundation to act as the representative of the project, for example, in court. In particular, the Free Software Foundation requires a copyright assignment and declares its intent to enforce its intellectual property rights.

3.5 The Governance model category

The **Governance:Board Membership** of a foundation can be anything from democratic (GV:DB) on the one end through meritocratic (GV:MB) in between to autocratic (GV:AB) on the other end. Some foundations allow for elections of board members from the member base or even general public while others have a set board that answers only to itself. The Free Software Foundation and the Mozilla Foundation are examples of the latter, the Linux Foundation and the Apache Software Foundation are examples of the former.

The **Governance:Project Membership** can be anything from democratic (GV:DP) (elected by membership) through meritocratic (GV:MP) (earned status

through continued project work) to autocratic (GV:AP) (appointed by board). This is particularly important for the project leadership. Most foundations, for example, the Apache Software Foundation and the Eclipse Foundation, support a meritocratic model in which project members earn their stripes before being elected into respective positions, but the other options also occur.

The **Governance:Natural Member Career** determines the career that a natural person, typically a developer, can have inside the foundation. The Apache Foundation provides the original model of this career [2]. The career design varies between foundation, but one can repeatedly find these three components [7]:

- The traditional open source career steps of increasing status are user, contributor, and committer (a.ka. maintainer) (GV:UDC). A user uses the software and helps others, a contributor makes contributions that have to pass review before they are accepted, and a committer reviews the work that contributors submit and can make contributions to the project that don't have to pass anyone else's review.
- Foundations make the project management level explicit (GV:PMC) that is tied to the committer role in traditional open source projects: Developers can become members of a project management committee (PMC). The PMC determines a project's road-map in the overall scope of the foundation. Thus, it manages the project's direction. A PMC member can become the leader of one or more PMCs.
- Finally, beyond project management, developers can become officers and board members of the foundation (GV:BRD), determining and contributing to its overall strategy and direction.

The **Governance:Juristic Member Level** determines the role that a juristic person can play within the foundation. Usually, the member level is closely aligned with the resources that a member provides, most notably financing (GV:FL) and development resources (GV:DL). The Eclipse foundation, for example, provides four types of corporate membership (“associate”, “solution”, “enterprise” and “strategic” members) while the TOPCASED foundation, whose software builds on the Eclipse platform, provides only one type of membership.

3.6 The Financing model category

The **Financing: (of the) Foundation** can be any or all of membership fees (F:MF), sponsorships (F:S), gifts and grants (F:GG), or revenues from a commercial subsidiary (F:FPS).

- Membership fees have the benefit of being predictable, while sponsorship may or may not happen. Sponsorship money may be tied to a particular project. Gifts and grants are also not predictable, however, for gifts and grants the foundation has to put in work and advertise or apply for it. These three are the most common forms of financing a foundation and most foundations rely on them.

- In addition, a foundation can finance itself through commercial subsidiaries. For example the Mozilla Foundation, with its for-profit subsidiary the Mozilla Corporation, funds itself through the income derived from products and services it sells for its open source software. In 2009, most of the Mozilla Foundation's $101M income came from its search related activities (the Firefox search box etc.) [3].

The foundation's income is usually spent on providing infrastructure services for the projects it is maintaining, for back office work, lawyers and legal work, and a (typically) small staff.

The **Financing: (of) Projects** may come directly from the foundation (F:PFF) or from the project participants (F:PMF), that is, the foundation members working on the project. For example, the Eclipse Foundation pays for (and maintains) all the infrastructure for its projects, while TOPCASED and GENIVI rely on their members to set-up and maintain the infrastructure on a per-project basis.

3.7 The Operations model category

The **Operations:Infrastructure** of a foundation's projects may be provided by the foundation itself (O:IFO) or by its members (O:IMO). This may be closely aligned with the financing of projects. The common choice is for the foundation to provide the infrastructure, as the Free Software Foundation, the Apache Software Foundation, the KDE e.V. and many others do. Sometimes, members perform these project services, as is the case with TOPCASED and GENIVI foundations.

The **Operations:Back Office** handles most of the back office work like maintaining a member database, collecting contributor agreements, and watching over proper processes. Much of this work may be performed by volunteers (O:BOV), but typically there are at least a few full or part-time employees of the foundation (O:BOE), paid for from the foundation's income.

4 Application of Model to Developer Foundations

4.1 Application overview

A full confirmatory validation of the model has yet to be done. To demonstrate at least the usefulness of the model, we apply it to the nine foundations we used in the theory generation process. Tables 3 and 4 shows the result of this application.

4.2 Discussion of application

The model fits the foundations well and no necessary changes to the model became apparent. This is not surprising, given that the model had been derived from these

foundations. As described in Section 2, we took a multiple-step process, first deriving an initial model for six selected foundations, and then extending it by incorporating into the model what we learned from four more foundations. In a similar vein, if we were to apply it to more foundations, we might find more extensions to the model. We expect these possible modifications to be small and incremental.

Table 3: Application of Model to 9 Foundations, Part 1

Model of Developer Foundations				Application of Model to Developer Foundations 1 / 2									
Category	Attribute	Possible Values	Value Short Codes	Linux Foundation	Free Software Foundation	Apache Software Foundation	Eclipse Foundation	Mozilla Foundation	KDE e.V.	TOPCASED	GENIVI	OpenAPC
General	Purpose	• Public benefit	PB	-	PB	PB	-	PB	PB	-	-	-
		• Member benefit	MB	MB	-	-	MB	-	-	MB	MB	MB
	Incorporation	• Foundation	F	-	F	F	-	F	F	-	-	-
		• Consortium	C	C	-	-	C	-	-	C	C	C
	Members	• Natural persons	NP	NP	NP	NP	NP	-	NP	-	-	-
		• Juristic persons	JP	JP	-	-	JP	-	JP	JP	JP	JP
Philosophy	Open source stance	• Free software	FS	FS	FS	-	-	FS	FS	-	FS	-
		• Open source	OS	-	-	OS	OS	OS	-	OS	OS	OS
	Development model	• Open development	OD	OD	OD	OD	OD	OD	OD	OD	-	OD
		• Closed development	CD	-	-	-	-	-	-	-	CD	-
Intellectual Property (IP)	Project license	• Reciprocal	RL	RL	RL	-	-	RL	RL	-	RL	-
		• Permissive	PL	-	-	PL	PL	PL	-	PL	PL	PL
	Patent license grant	• No grant	NPG	-	-	-	-	-	-	-	-	-
		• Grant for use	UPG	-	-	-	UPG	-	-	UPG	-	UPG
		• Use + embedding	EPG	EPG	EPG	EPG	-	EPG	EPG	-	EPG	-
	IP ownership	• No IP required	NIP	-	-	-	-	NIP	NIP	-	-	-
		• Relicensing rights	RRG	RRG	-	RRG	RRG	-	-	RRG	RRG	RRG
		• Copyright assignment	CA	-	CA	-	-	-	-	-	-	-

Continued in Table 4

5 Related Work

West and O'Mahoney investigate corporate sponsorship of open source communities, some of which are organized as foundations [12]. They call "participation architec-

Table 4: Application of Model to 9 Foundations, Part 2

Model of Developer Foundations				Application of Model to Developer Foundations 2 / 2								
Category	Attribute	Possible Values	Value Short Codes	Linux Foundation	Free Software Foundation	Apache Software Foundation	Eclipse Foundation	Mozilla Foundation	KDE e.V.	TOPCASED	GENIVI	OpenAPC
Governance	Board membership	• Democratic	DB	-	-	-	-	-	DB	-	-	-
		• Meritocratic	MB	-	-	MB	-	-	-	-	-	-
		• Autocratic	AB	AB	AB	-	AB	AB	-	AB	AB	AB
	Project membership	• Democratic	DP	DP	-	-	-	-	DP	DP	DP	DP
		• Meritocratic	MP	-	-	MP	MP	MP	-	-	-	-
		• Autocratic	AP	-	AP	-	-	-	-	-	-	-
	Natural member career	• User/developer/committer	UDC	UDC	UDC	UDC	UDC	UDC	UDC	UDC	UDC	UDC
		• Project manager/leader	PMC	PMC	-	PMC	PMC	PMC	PMC	PMC	PMC	PMC
		• Board member	BRD	BRD	-	BRD	BRD	-	BRD	BRD	BRD	BRD
	Juristic member level	• Financing level	FL	FL	-	FL	FL	-	FL	-	-	-
		• Developer level	DL	DL	-	-	DL	-	-	DL	DL	DL
Financing	Foundation	• Membership fees	MF	MF	MF	-	MF	-	MF	MF	MF	MF
		• Sponsorship	S	-	-	S	S	-	S	-	-	-
		• Gifts and grants	GG	-	GG	GG	GG	GG	GG	-	-	-
		• For-profit subsidiary	FPS	-	-	-	-	FPS	-	-	-	-
	Projects	• Foundation	PFF	-	PFF	PFF	PFF	PFF	PFF	-	-	PFF
		• Members	PMF	PMF	-	-	-	-	-	PMF	PMF	-
Operations	Infrastructure	• Foundation	IFO	IFO	IFO	IFO	IFO	IFO	IFO	-	-	IFO
		• Members	IMO	IMO	-	-	-	-	-	IMO	IMO	-
	Back office	• Volunteers	BOV	-	-	BOV	-	-	BOV	-	-	-
		• Employees	BOE	BOE	BOE	BOE	BOE	BOE	BOE	BOE	BOE	BOE

ture" what we call "foundation model". Their research, like ours, was qualitative in nature and based on interviews and supplementary materials review. West and O'Mahoney's Production category matches our Development Model category, their Governance category matches ours, except that we determined a separate Financing category, and their Intellectual Property category matches ours as well, except that ours involves Patent Handling. We became aware of this work only after performing our own, so we view similarities as additional validation of our work. West and O'Mahoney missed several categories like the more fine-grain member levels and careers that we determined. In general, we are providing more details with our possible values for attributes and the constraints between them.

Xia et al. distinguish between output and process benefits that consortium members achieve [13]. They take the view of a single company trying to gauge returns on participating in a consortium. Output benefits are derived from having a stake in the consortium's output. Process benefits take the form of social capital and learning that a company derives from participating. A survey shows these benefits and we expect the process benefits to apply to members of open source foundations as well.

Among the diversity of real consortia, open source foundations are closest to research and development (R&D) consortia. Aldrich et al. present a comparative study of R&D consortia in the U.S.A. and Japan [1]. They find that Japanese consortia all show similar structures and formalize long-standing inter-company relationships. In contrast, U.S.-based consortia show significant diversity in terms of organizational structure and processes and were more fluid than their Japanese counterparts. In contrast to these consortia, open source foundations tend to have an international perspective. Thus they tend to have little dependency on governmental support and despite necessary incorporation in a particular jurisdiction are more adjusted to international needs than their local and more traditional non-open-source counterparts.

6 Conclusions

This article presents a qualitative study of open source developer foundations. In a multiple-step process, it distills a model of the structure and governance of these foundations. The model takes the form of category, attribute, and possible-value triples that explain how to design such a foundation. To demonstrate the models usefulness, it is successfully applied to nine such foundations. The goal of this research is to provide to the leaders of community open source projects a guide for choosing an existing foundation or a blueprint for designing their own foundation.

7 Acknowledgments

We would like to thank the anonymous reviewers for their comments. We would also like to thank everyone who participated in the interviews. Finally, we would like to thank Wayne Beaton (Eclipse Foundation), Gervase Markham (Mozilla Founda-

tion), Cornelius Schumacher (KDE e.V.) and Björn Schießle (Free Software Foundation Europe) for their help in improving this paper.

8 References

[1] Howard E. Aldrich, Michele Kremen Bolton, Ted Baker, Toshihiro Sasaki. "Information Exchange and Governance Structures in U.S. and Japanese R&D Consortia: Institutional and Organizational Influences ." *IEEE Transactions on Engineering Management,* vol. 45, no. 3, (August 1998). 263 pp.

[2] The Apache Software Foundation. "How It Works." URL: http://www.apache.org/foundation/how-it-works.html. Accessed: 2011-12-25. See http://www.webcitation.org/64CSTesol.

[3] Mitchell Baker. The State of Mozilla: Sustainability. 2010-11-18. URL: http://www.mozilla.org/foundation/annualreport/2009/sustainability.html. Accessed: 2011-12-25. See http://www.webcitation.org/64CSdMEdU.

[4] Juliet Corbin, Anselm Strauss. *Qualitative Research.* Sage Publications, 2008.

[5] Daniel German, M. Di Penta, J. Davies. "Understanding and Auditing the Licensing of Open Source Software Distributions", International Conference in Program Comprehension (ICPC 2010).

[6] Open Source Initiative. "Open Source Licenses." URL: http://www.opensource.org/licenses/index.html. Accessed: 2011-12-25. See http://www.webcitation.org/64CSEJk8G.

[7] Dirk Riehle. "A New Developer Career." 2010-06-10. URL: http://dirkriehle.com/2010/06/10/linux-tag-keynote-slides-a-new-developer-career/. Accessed: 2011-12-25. See http://www.webcitation.org/64CSLWhEV.

[8] Dirk Riehle. "The Single-Vendor Commercial Open Source Business Model." Information Systems and e-Business Management vol. 10, no. 1. Springer Verlag, 2012, 5-17.

[9] Dirk Riehle. "The Economic Case for Open Source Foundations." IEEE Computer vol. 43, no. 1 (January 2010). Page 86-90.

[10] Dirk Riehle. "The Economic Motivation of Open Source: Stakeholder Perspectives." IEEE Computer vol. 40, no. 4 (April 2007). Page 25-32.

[11] Richard Stallman. "Why Open Source Misses the Point." 2011-09-20. URL: http://www.gnu.org/philosophy/open-source-misses-the-point.html. Accessed: 2011-12-25. See http://www.webcitation.org/64CRzMBMq.

[12] Joel West, Siobhan O'Mahoney. "The Role of Participation Architectures in Growing Sponsored Open Source Communities." Industry and Innovation vol. 15, no. 2. Taylor & Francis, 2008.

[13] Mu Xia, Kexin Zhao, Joseph T. Mahoney. "Enhancing Value via Cooperation: Firms' Process Benefits From Participation in a Consortium." UIUC College of Business Working Paper 08-0109, 2008.